Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Módulo para la creación de juegos del tipo observación en el *software* ATcnea.

Autor(es):

Lennon Acosta La O

Luis Manuel Lugo Marquez

Tutor(es):

MSc. Tatiana Leyva Estrada

Ing. Yaritza B. González Ramírez

Ing. Yunier Broche Guevara

Co-Tutor(es):

Ing. Cesar A. González Rodríguez

La Habana, 2017

“En Cuba, nadie ha hecho tanto en tan poco tiempo.”

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizo al Centro de tecnologías para la formación (FORTES) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Lennon Acosta La O

_____

Luis Manuel Lugo Marquez

_____

MsC. Tatiana Leyva Estrada

_____

Ing. Yunier Broche Guevara

_____

Ing. Yaritza B. González Ramírez

_____

Ing. Cesar A. González Rodríguez

_____

AGRADECIMIENTOS DE LENNON ACOSTA LA O

Deseo agradecer antes que todo a quien me dio la vida, tú madre, que a pesar de la distancia siempre estabas ahí, siempre me brindase apoyo y siempre, siempre, me impulsaste, tus pasos son los que sigo hoy, no importa cuán difícil sea el camino, contigo alcanzaré la meta. Te amo mamá.

A mi padre, que a pesar de no estar aquí presente, sé que me da su fuerza, tu ejemplo me guió en mis primeros pasos, y hoy ejerzo lo positivo porque así me enseñaste a hacerlo. Te quiero viejo.

Mi hermano, que aun siendo menor te estas portando mayor, y me ayudaste a tomar decisiones que solo no podía, gracias Harry.

A mi bella esposa, Gissy, quién durante 2 años y dos meses, ha vencido esta batalla conmigo, quien juntó su fuerza a la mía para hacer un equipo invencible, esta tesis se logró también gracias a ti, te amo mi vida.

A la familia de mi esposa, mi suegra Janny, mi abuela también Gissy, porque así veo a María, a Daniel, a Manuel y en fin, a todos, gracias.

A mi familia que sé que aunque no están presentes están felices de tener un ingeniero en la familia, mis abuelos, mi abuela Andrea, mis tíos, mis primos. Su apoyo también fue importante.

Los amigos que más que amigos fueron familia, mis hermanos de causa, el apartamento se volvió una casa mientras pasaban los años, y allí todos logramos conquistar esta meta, felicidades a todos los que junto a mí, estamos alcanzando el triunfo, ser Ingenieros.

Agradecimiento especial a quienes hicieron de la Universidad un festival, ELEMENTRIX, Elvis, Asiel y Daniel. Con ustedes hice lo que no había hecho en la vida, y espero continuar el próximo año en la historia de ELEMENTRIX, Gracias.

A mis tutores que durante el transcurso del año nos apoyaron y mostraron el camino del triunfo, especialmente a Yaritza y te aclaro, no somos turistas, gracias.

AGRADECIMIENTOS DE LUIS MANUEL LUGO MARQUEZ

A mi mamá, siempre te estaré agradecido por todo lo que has sacrificado por mí y para que este hoy aquí, no eres capaz de imaginar cuánto.

A mi papá, tu apoyo nunca me faltó.

A mis tutores, su apoyo y preocupación nunca faltó, más que agradecido siempre estaré.

A mis amigos, por siempre estar ahí cuando las cosas empezaban a ponerse duras, nunca fallaron.´

A otras personas que con tan solo preguntar ¿cómo van las cosas?, mostraron su preocupación e interés porque yo esté hoy aquí.

DEDICATORIA DE LENNON ACOSTA LA O

Dedico el triunfo de hoy a mi madre, tú me enseñaste a luchar.

A mi padre, tú me diste la fuerza.

A mi hermano, tú luchaste conmigo.

A mi esposa, tú me amaste en todo el camino, y deseo me ames el resto del mismo.

A mi familia y la familia de mi esposa, quienes me apoyaron y desearon lo mejor.

A mis tutores, quienes se están volviendo a graduar con nosotros.

DEDICATORIA DE LUIS MANUEL LUGO MARQUEZ

A mi mamá, por estar ahí siempre que te he necesitado sin importar que tan difíciles han sido los momentos, por guiarme a ser lo que hoy soy, por confiar en que podía lograrlo, nunca podré estar lo suficientemente agradecido, especialmente para ti.

A mi papá, por el apoyo en todo este tiempo, por confiar en que podía lograrlo.

A mis hermanas, sé que ustedes llegarán aún más lejos, no importa lo duro que pueda ser el camino, confío en ustedes.

RESUMEN

La educación en el país se encuentra inmersa en un proceso de desarrollo y cambio, lo cual implica que se inserten en las escuelas las Tecnologías de la Información y las Comunicaciones (TIC), esto supone entonces la creación de aulas tecnológicas que devienen de la relación Educación-Tecnología. Actualmente en la Universidad de las Ciencias Informáticas (UCI) se desarrolla el *software* ATcnea que formará parte de las aulas tecnológicas cubanas, a pesar de las grandes ventajas que brinda, este no cuenta con prácticas dinámicas por lo que se hace necesaria su introducción en el mismo. El objetivo del presente trabajo es desarrollar un módulo para la creación de juegos del tipo observación en el *software* ATcnea. Para definir las características de la propuesta de solución se analizaron *software* para aulas tecnológicas con el fin de comprobar la existencia de módulos de juegos didácticos. En correspondencia con la metodología de desarrollo de *software* AUP, en su variante para la actividad productiva en la universidad, se identificaron, modelaron y describieron los requisitos funcionales identificados. Para la validación de la solución implementada se realizaron pruebas de sistema y de aceptación que permitieron verificar el correcto funcionamiento del módulo desarrollado.

Palabras clave: aulas tecnológicas, prácticas dinámicas, juegos educativos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La escuela como principal centro de formación del ser humano ha transitado por un proceso de desarrollo que no solo va encaminado a la adquisición de nuevas formas de conocimiento, sino también a nuevos métodos que ayuden y promuevan el avance tecnológico por el que la sociedad actual se encuentra cursando, siendo estas nuevas herramientas lo más novedoso en materia de educación. Utilizar las posibilidades que brinda esta nueva forma de enseñar estaría tributando directamente no solo a mejorar las condiciones en que se prepara a los educandos, sino también a romper con el tradicionalismo que caracteriza las aulas de hoy en día, constituyendo este un modo dinámico de aprender e implicar más activamente estos métodos en la tarea de educar (Martin, 2010).

En los últimos años, con el desarrollo tecnológico y pedagógico de los entornos virtuales de enseñanza se desarrollaron distintas alternativas. Estas posibilitan la mediación de las propuestas educativas con las Tecnologías de la Información y las Comunicaciones (TIC) a través de la creación de nuevos dispositivos y de nuevas formas de planificar, interpretar y comprender el rol docente así como también la clase misma.

Resulta esperanzador mirar hacia atrás y descubrir que ya están superadas las viejas discusiones y concepciones acerca del papel de las TIC en la educación. En la actualidad es cada vez menos frecuente encontrarse con docentes que cuestionen su valor e importancia para incorporarlas a las propuestas de enseñanza y el proceso de aprendizaje, pero que se encuentran preocupados por desarrollar saberes específicos acerca de la introducción de las tecnologías digitales en sus propuestas de enseñanza. En contextos de convergencia tecnológica la enseñanza necesita reinventarse, y es en este sentido que las TIC ofrecen posibilidades para que ello ocurra. Los nuevos entornos tecnológicos brindan un entramado acerca de los modos en que el conocimiento se construye, se distribuye y complejiza (Martin, 2010).

El avance de las TIC ha constituido un gran impulso para disímiles áreas dentro de la sociedad. Entre los entornos favorecidos se encuentra la educación, dando surgimiento a nuevas formas de enseñar y por tanto nuevas maneras de aprender. Un ejemplo de esto son las aulas tecnológicas que impulsan el aprendizaje mediante la interacción profesor-estudiante mediado por la tecnología.

Las llamadas aulas tecnológicas, también encontradas en algunas bibliografías como aulas interactivas o aulas digitales, constituyen una realidad actual que brinda una solución educativa contemporánea para el método de enseñanza-aprendizaje. Estas últimas muestran una experiencia única en el aula, con marcada intencionalidad para el nivel primario y secundario, que es donde se encuentran en formación los "nativos" en informática, pero sin dudas, requeridas también en todos los niveles (Dopico, 2014).

Este novedoso tema, surge producto de la aplicación de las nuevas tendencias educativas y su vinculación con las tecnologías. El objetivo de las aulas tecnológicas, término más abarcador que las identifica, es la creación de un ambiente colaborativo, que propicia la introducción de tecnologías como medios, por lo que forma parte de la didáctica y constituye un modo de enriquecimiento del contenido académico que imparte la figura docente. Además permite a profesores y educandos establecer una profunda comunicación, cuya interactividad en el intercambio de ideas e información y formas de colaboración, motivará la participación y profundización en los temas objeto de estudio. También desarrolla un nuevo entorno de aprendizaje, donde el estudiante podrá manejar un conjunto organizado de contenidos que le permita ser capaz de analizar el mundo que le rodea y tomar decisiones (Dopico, 2014).

Diferentes autores han plasmado varias definiciones de las aulas tecnológicas, un ejemplo de estas es la definición realizada por Felipe Segovia Olmo como una: *"Comunidad de aprendizaje, cuyo objetivo principal es el desarrollo de la inteligencia y de los valores de los alumnos, que planifican, realizan y regulan su propio trabajo, bajo la mediación de los profesores, por medio de métodos didácticos diversificados y tareas auténticas, evaluados por alumnos y profesores, en un espacio multiuso abierto, tecnológicamente equipado y*

organizado según los principios de la calidad total en la gestión" (Olmo, 2003) se ajusta al concepto integrador que promueven las tendencias educativas del siglo XXI.

Antonia Lozano Díaz en su reseña del libro de Segovia Olmo las delimita como, *"el sistema educativo aula inteligente es un constructo creativo, un conjunto de saberes que se plasman en una pedagogía singular. Propugna un cambio de modelo de educación a través de la reingeniería total del sistema educativo; lo hace partiendo de una determinada conceptualización de lo que sería la calidad en educación"* (Díaz, 2004). Este cambio de actitud, aptitud y mentalidad, constituye un reto en la actualidad para la educación moderna.

La composición de estas aulas se diseña sobre la base de los modelos educativos y didácticos que se pretenden aplicar y se debe tener en cuenta, además de los aspectos arquitectónicos, ambientales, de acabado o mobiliario, como elemento fundamental el equipamiento físico y lógico básico. Considerándose como tal las PC o computadoras, tabletas digitales y teléfonos inteligentes, el *software* compatible y conectividad adecuada que garantice desde la integración del equipamiento, hasta las aplicaciones para el desarrollo colaborativo de los contenidos e intercambios como son los entornos virtuales de enseñanza-aprendizaje, multimedias educativas, teleconferencias o intercambios en trabajos grupales en foros, wiki, blogs, etc. (Dopico, 2014).

Aunque las aulas tecnológicas describen un avance potencial en el desarrollo del proceso educativo es necesario tener en cuenta aspectos esenciales que ponen en peligro el correcto funcionamiento y dinámica de las mismas. Por tanto, para lograr un óptimo aprovechamiento de aquellas facilidades que brinda un aula de esta índole hay que basarse principalmente en dotar a los usuarios de las aulas tecnológicas, dígase alumnos y profesores, de aquellas competencias imprescindibles para el correcto manejo de las tecnologías, así como el balance con otras actividades educativas que también tributen al proceso, y además tener en cuenta una selección de aquellas actividades formativas que han de llevarse a cabo para facilitar el desarrollo exitoso de la clase (González, 2005).

Por el propio hecho de la inserción de las nuevas tecnologías en el aula y la aparición de las aulas tecnológicas, el manejo de prácticas dinámicas que contribuyan al desarrollo escolar de los niños ha ido en ascenso, se ha demostrado que es realmente útil la implementación de las mismas en las escuelas como apoyo directo a la educación escolar. Además, son fuente de motivación ante la adquisición de los nuevos conocimientos, por lo que constituye una ventaja su uso como herramienta colateral en el proceso educativo. Dichas prácticas dinámicas aportan un valor añadido al proceso lúdico, que suele estar relacionado con aspectos como la concientización, así como el manejo de habilidades que sustentan un proceso educativo más acertado y atractivo para los escolares (Extremadura, 2001).

El uso de las prácticas dinámicas en el aula ayudan al profesor a erradicar problemas tales como (Martínez, 2014):

- Aburrimiento.
- Falta de compromiso y motivación.
- Carencia de destrezas y habilidades.
- Ausencia de trabajo colaborativo.
- Incapacidad de comprensión y expresión por medios tradicionales.

Durante el curso 2015-2016 comienza en el Centro de Tecnologías para la Formación (FORTES) de la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI), el desarrollo de un *software* para la gestión de una clase en un aula con el empleo de las TIC, generalmente conocido como aula tecnológica. En este caso, el *software* se ha denominado ATcnea y se ha desarrollado para su funcionamiento sobre tecnología HAIER, ensamblada por la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica (GEDEME) y con la colaboración de soluciones libres brindadas por el Centro de Soluciones Libres (CESOL) de la Facultad 1 de la UCI. El conjunto consistiría en dos aplicaciones informáticas con fines educativos, una PC, con sistema operativo GNU/Linux y una tableta con sistema operativo Android. La comunicación entre las dos aplicaciones le posibilita al profesor, desde

su PC, guiar el proceso de enseñanza-aprendizaje, usando los medios que proveen las aulas tecnológicas, y a los estudiantes, desde la tableta, recibir e interactuar con el contenido. Entre los requerimientos de *software* a tener en cuenta como prioridad principal, se encuentra desarrollar los sistemas informáticos para la distribución Nova para sistemas GNU/Linux y Novadroid para sistemas Android.

Aunque a nivel internacional existen productos de este tipo se requiere desarrollar el mismo con el objetivo de lograr la soberanía tecnológica en el país, expandiendo por Cuba y el mundo un *software* y sistemas operativos cubanos.

Dicho producto ya se encuentra en su versión 1.0, sin embargo el hecho de reproducir el contenido en la tecnología no implica que, en los estudiantes se supriman del todo durante el transcurso de las actividades que se realizan las conductas de aburrimiento, la falta de compromiso con la tarea y la desmotivación, además se pueden manifestar trabas para obtener los conocimientos de una manera más dinámica y menos tradicional.

Por esta razón se hace necesaria la introducción de prácticas dinámicas en el *software* ATcnea, pues de manera más positiva y eficaz estaría contribuyendo a eliminar aquellos obstáculos que puedan interferir en la aprehensión efectiva de los contenidos que se le imparten a los educandos, y así enriquecer tanto el aprendizaje como la interacción de los estudiantes con la clase.

Teniendo en cuenta lo antes planteado, se define como **problema a resolver:** ¿Cómo incluir las prácticas dinámicas al *software* ATcnea en su versión 1.0?

La presente investigación tiene como **objeto de estudio:** Módulo de juegos didácticos en aulas tecnológicas.

El **campo de acción** de la investigación está enfocado en: Módulo de juegos del tipo observación en el *software* ATcnea.

Se define como **objetivo general:** Desarrollar un módulo para la creación de juegos del tipo observación en el *software* ATcnea.

Para darle cumplimiento al objetivo general, se identifican los siguientes **objetivos específicos**:

- Realizar un análisis de los aspectos teóricos relevantes y soluciones homólogas existentes.
- Analizar la metodología, tecnologías y herramientas que sustenten el desarrollo del módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Diseñar el módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Implementar el módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Validar la solución propuesta a través de la aplicación de diferentes tipos de pruebas.

Sobre la base de la situación anteriormente planteada se formularon las siguientes **tareas de investigación**:

- Analizar las diferentes aulas tecnológicas investigadas por el grupo de desarrollo del *software* ATcnea para verificar el uso de módulos de juegos didácticos.
- Realizar un estudio de la metodología utilizada por el grupo de desarrollo del *software* ATcnea.
- Realizar un estudio de las herramientas utilizadas por el grupo de desarrollo del *software* ATcnea.
- Definir las características que posea el módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Realizar el diseño con el fin de describir el módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Analizar el patrón arquitectónico y los patrones de diseño para la implementación del módulo para la creación de juegos del tipo observación en el *software* ATcnea.

- Implementar el módulo para la creación de juegos del tipo observación en el *software* ATcnea.
- Realizar las pruebas de sistema, integración y aceptación que validen el módulo para la creación de juegos del tipo observación en el *software* ATcnea.

Para la realización de la siguiente investigación se utilizaron los siguientes **Métodos Científicos**:

Métodos teóricos

Analítico-sintético: Permitió estudiar y realizar el análisis de documentos y bibliografías que sustenten de forma teórica los elementos asociados a los módulos de juegos didácticos en aulas tecnológicas.

Histórico-lógico: Se aplica para desarrollar un estudio de las tendencias actuales en el desarrollo de los video juegos y cómo han evolucionado en el tiempo, lo cual permite determinar lo más factible para el desarrollo del módulo.

Análisis documental: Se utilizó para realizar consultas a la literatura especializada en el tema abordado en la presente investigación.

Métodos empíricos

Observación: Se utiliza para realizar el análisis de las características de los sistemas similares al módulo para la creación de juegos del tipo observación en el *software* ATcnea.

La presente investigación se encuentra estructurada como se detalla a continuación:

Capítulo I Fundamentación teórica: En este capítulo se analizan y presentan los principales enfoques teóricos con el objetivo de generar el marco teórico de la investigación. Además se realiza el estudio y selección de las herramientas y tecnologías a utilizar para el desarrollo de la solución.

Capítulo II Características y diseño del sistema: En este capítulo se describe el modelo de dominio, y los conceptos asociados al mismo. Se generan las historias de usuarios como

artefactos fundamentales asociados a la metodología según los requisitos funcionales y no funcionales identificados. Además se realiza el diseño de la solución propuesta identificándose el patrón arquitectónico y los patrones de diseño utilizados en el desarrollo de la misma. Además se elaboran los diagramas de clase del diseño y los diagramas de secuencia.

Capítulo III Implementación y prueba: En este capítulo se describen los principales aspectos de la implementación. Incluye además la estrategia seguida para aplicar las pruebas al sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1. Introducción

En este capítulo se realiza un estudio de las aulas tecnológicas existentes en el mundo así como de los juegos didácticos y video-juegos del tipo observación para confeccionar el marco teórico de la investigación. Además se detallan las herramientas, tecnologías y lenguajes de programación a utilizar así como la metodología de desarrollo a emplear durante la implementación del módulo para la creación de juegos del tipo observación en el *software* ATcnea.

1.1 Historia de los video-juegos

Sobre la década del 40 a finales de la segunda guerra mundial, se empiezan a concretar los primeros intentos del hombre por crear máquinas automáticas para el procesamiento de información. Esto se debió en gran medida, al uso de la máquina electromecánica ***Bombe*** diseñada por Alan Turing para descifrar los códigos nazis. Así se evidenció la importancia de automatizar los procesos por medio de máquinas, y surge entonces el término computación. Las primeras computadoras eran aparatos inmensos, que podían pesar varias toneladas y abarcar grandes espacios; ejemplo de esto es la “ENIAC”, desarrollada en la universidad de *Pennsylvania*, que pesaba varias toneladas y ocupaba todo un sótano. Con el paso del tiempo las computadoras fueron disminuyendo en tamaño y aumentando su procesamiento con la aparición de los primeros microchips (Jiménez, 2016).

Junto a las computadoras, los video-juegos fueron desarrollándose paralelamente, y aunque en sus inicios no fueron considerados video-juegos como tal, sino que se definían como juegos-electrónicos, puesto que su funcionamiento se basaba en un artefacto electro-mecánico (Jiménez, 2016).

Los primeros intentos de juegos-electrónicos fueron estudios e investigaciones, dado que en el momento no se veían como algo prometedor. Ejemplo de esto, fue el ***OXO*** desarrollado por Alexander Shafto Douglas en 1952 como complemento de su tesis de doctorado, que

tenía como tema principal la interactividad entre seres humanos y computadoras. El juego consistía en el famoso “cruz y raya” y permitía jugar a un humano contra la computadora. Un poco más adelante se puede ver el juego ***Tenis for two***, creado por William Higinbotham en 1958 del Laboratorio Nacional de *Brookhaven* el cual consistía en un juego de tenis simulado en la pantalla de un osciloscopio. Este juego tenía como objetivo entretener a los visitantes del laboratorio y por este motivo años más tarde es desmantelado para emplear sus piezas en investigaciones más serias (Jiménez, 2016).

Así es como se sentaron las bases de los juegos-electrónicos y se abren las puertas a un nuevo mundo que es el de los video-juegos, que no es hasta la década del 70 cuando toman un gran auge con la aparición de compañías dedicadas a comercializar todo tipo de artefactos para el entretenimiento. Algunas de las compañías más conocidas que aún subsisten son (Jiménez, 2016):

- ***NINTENDO***: Es una empresa japonesa que en los inicios de los 70 presentó dispositivos de juegos para salas recreativas y más adelante crea una consola que permitía jugar varios video-juegos con solo conectarse a un televisor.
- ***ATARI***: Es una empresa estadounidense famosa por la creación de la videoconsola doméstica *Atari Pong*, que se conectaba a un aparato de televisión y permitía jugar el famoso video-juego *Pong*.
- ***KONAMI***: Empresa japonesa que en sus inicios se dedicaba a la creación de máquinas arcade y videoconsolas para otras compañías, luego se dedicó a realizar sus propios video-juegos y es una de las empresas más conocidas hasta la actualidad.

Durante algún tiempo luego de los 70, los video-juegos fueron pasando de ser jugados en consolas personalizadas que necesitaban de medios como un televisor, hacia dispositivos únicamente destinados al juego con todos los periféricos necesarios integrados. Es aquí cuando llega la década de los 90 donde las computadoras personales tienen un inmenso auge, y revolucionan la industria de los video-juegos convirtiéndola en el monopolio que representa en la actualidad. Con los juegos para computadoras los usuarios tenían nuevas

funcionalidades, nuevos modos y formas que permitieron un desarrollo paulatino de los mismos (Jiménez, 2016).

Es así que con el surgimiento de nuevas tecnologías, tales como las tecnologías móviles, no fue mucha la espera para encontrarse un mercado abarrotado de video-juegos y recursos disponibles para estos medios. Ya sea para sistemas operativos *iOS* (de la reconocida firma *Apple*) como de *Android* (que en la actualidad se encuentra bajo la tutela de *Google*) o de otros no tan conocidos pero no menos importantes como *Windows Mobile*. Resulta que en la actualidad es posible encontrar en cualquier medio informático uno de esos apasionantes y entretenidos video-juegos que remontan su historia a los rústicos juegos de los 70 (Jiménez, 2016).

1.2 Juegos del tipo Observación

¿Qué entendemos por observación?

El concepto de observación varía según el tiempo y el contexto en el que se aplique. En cualquier caso y de manera simplificada, debería entenderse como un proceso que requiere de atención voluntaria e inteligente, orientado por un objetivo con el fin de obtener información (Camacho, 2011).

Según (Boisvert, 2015) las categorías seleccionada para los juegos de acuerdo con la habilidad o competencia que desarrollan, son:

- Observación
- Atención
- Aptitud musical
- Destrezas manuales
- Ingenio e imaginación
- Agilidad y destrezas motoras

Para el presente estudio se hace énfasis en los juegos del tipo Observación los cuales se dividen en varios tipos según (Boisvert, 2015):

1 ¿Cuántos ves?

2 El detective

3 Un guiño de ojo

4 ¿Qué falta?

5 ¿De qué comida se trata?

6 ¿Qué es lo que ha cambiado?

7 Cuestión de memoria

8 Una cuestión de imagen

Como parte de la investigación los autores se centrarán en los juegos Descubre la imagen y Descubre las diferencias que forman parte de los juegos del tipo observación respondiendo a los siguientes tipos: El detective y ¿Qué es lo que ha cambiado? respectivamente. En estos últimos los usuarios que interactúan con los mismos deben observar diferentes imágenes para identificar objetos o diferencias usando la lógica, los conocimientos y la experiencia.

1.3 Ejemplos de Video-juegos educativos

El uso de video-juegos por parte de los niños y jóvenes se ha vuelto muy común, y han sido objeto de reflexiones y críticas, tanto por sus contenidos como por el alto porcentaje de tiempo que esta población pasa frente a un televisor, computador, teléfono móvil o tabletas.

Para Gifford (1991) existen siete características que hacen que estos recursos sean un medio de aprendizaje más atractivo y efectivo:

1. Posibilitan el ejercicio de la fantasía, sin limitaciones en el tiempo y en el espacio.
2. Facilitan el acceso a otros escenarios de aprendizaje diferentes al aula de clase.
3. Favorecen la repetición y el intentarlo otra vez, en un ambiente que no reviste riesgos.
4. Permiten el dominio de habilidades; aunque parezca difícil, los niños tienen la opción de repetir las acciones, hasta llegar a dominarlas, consiguiendo la sensación de control.

5. Facilitan la interacción con otros amigos, contrario a lo que usualmente sucede en un aula de clase, de una manera no jerárquica.
6. Hay claridad en los objetivos que se persiguen. Normalmente el niño o el joven no tiene claro qué es lo que está estudiando en sociales, matemáticas o ciencias, cosa que no pasa cuando utiliza el video juego debido a que este establece una tarea clara y concreta: abrir una puerta, rescatar a alguien, hallar un tesoro, etc., lo que proporciona un alto nivel de motivación.
7. Favorece la atención y el autocontrol, dado que se evidencia la noción de que cambiando el entorno y no al niño, se puede propender por el éxito individual.

Hace varios años la posibilidad de pensar que un videojuego tuviera connotaciones educativas, era impensable en los padres y docentes. Hoy en día son innegables las bondades que esta clase de herramientas tienen en el proceso de aprendizaje de los niños y jóvenes, y es prioritario que el sistema educativo lo tenga en cuenta para el mejor logro de los objetivos propuestos.

1.3.1 MinecraftEDU. Videojuego que se adapta al plan de estudios

MinecraftEDU es la versión académica del famoso videojuego *Minecraft*. Este, además de enseñar sobre Informática, Física o Matemáticas, permite que cualquier profesor personalice el *software* para adaptarlo a su plan de estudios. *MinecraftEDU* ofrece, también, talleres de capacitación para maestros y tutoriales para estudiantes. Joel Levin, *'Minecraft Teacher'* y cofundador de la empresa creadora de esta versión del videojuego, describía, en una entrevista con Yorokobu, su experiencia en el aula: “Cuando los niños se meten en su rol y se enfrentan a las situaciones dentro de un juego que les encanta, conectan con el contenido” (Paredes, 2015).

De este juego se puede adquirir el dinamismo del mismo para ser personalizado por cada profesor a su clase y estilo, así como arrojar a los estudiantes un conocimiento entretenido.

Imagen 1 Juego *MinecraftEDU* (Paredes, 2015)

1.3.2 Proyecto *Kokori*. Biología en primera persona

A los mandos de una pequeña nave, llamada '*nanorobot'*, los estudiantes se introducirán en una célula y se enfrentarán a siete misiones centradas en la asignatura de Biología. El objetivo del proyecto *Kokori* es que los alumnos descubran, en primera persona, cómo son las células y sus componentes y conozcan el desarrollo de varios procesos biológicos desde dentro (Kokori, [2010]).

De este juego se puede adquirir la capacidad de estilizarse solo a una asignatura y un tema en específico, además de la riqueza de usar imágenes de gran escala y encontrar diferencias pequeñas, por poner un ejemplo, diferenciar células benignas de células malignas.

Imagen *2* Proyecto *Kokori* (Kokori, [2010])08

1.3.3 Simple Machines. Física con un simpático robot

Es un juego diseñado por el Museo de Ciencia e Industria de *Chicago* para ayudar a los estudiantes a aprender los principios físicos básicos implicados en el uso de niveles, poleas, planos, ejes y ruedas. Los alumnos tendrán que ayudar a *Twitch*, un robot muy simpático, a reunir las piezas necesarias para construir una máquina (Paredes, 2015).

De dicho juego se puede adquirir el uso de herramientas para descifrar un objetivo, además de la posibilidad de saber que se puede desarrollar juegos educativos para un lugar.

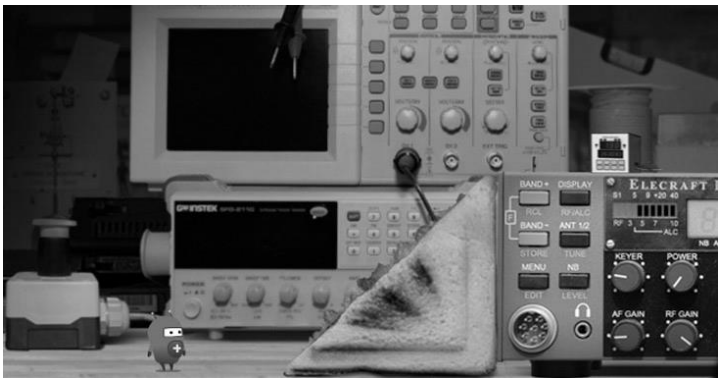


Imagen 3 Juego *Simple Machines* (Paredes, 2015)

1.4 Aulas Tecnológicas analizadas

Tomando como punto de partida el estudio realizado por los integrantes del proyecto Desarrollo del producto ATcnea sobre las diferentes aulas tecnológicas existentes a nivel internacional tales como: *Mythware*, *Gradelink*, *Radix*, *NetSoporte School* entre otras; el equipo de desarrollo profundiza en el estudio con el objetivo de verificar si estas últimas poseen un módulo para la creación de juegos del tipo observación.

A continuación se muestra una tabla donde se evidencia un resumen de las aulas analizadas:

Tabla 1 Análisis de las Aulas tecnológicas

Software	**Módulo de juegos**	**Características y principales funcionalidades**

Mythware Classroom Management Software	NO	**Características:** Vendedor: *Nanjing Universal Networks* País: China URL: www.mythware.com Fundado: 2007 Prueba gratis: Sí Despliegue: Web Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Pizarra Interactiva, Control Web, Cuestionario con Respuesta, Enseñanza de Grupo, Gestión de Grupo.
Gradelink	NO	**Características:** Vendedor: *Gradelink* País: Estados Unidos URL: www.gradelink.com Fundado: 2002 Prueba gratis: Sí Despliegue: Web Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Gestionar Registros del Estudiante, Basado 100% en la Web, Sitios web de las Escuelas (servicio *premium*).
Radix	NO	**Características:** Vendedor: *Radix Technologies* País: Israel URL: www.radix-int.com Fundado: 1992 Despliegue: Instalado, Móvil Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Soporta entorno operativo de Clases Mixto, Transmisión de Pantalla, Pizarra Interactiva Colaborativa.
NetSoporte School	NO	**Características:** Vendedor: *NetSoporte* País: Estados Unidos URL: www.netSoporte-inc.com Prueba gratis: Sí Despliegue: Instalado Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Pizarra virtual, Medición y control de internet y de aplicaciones, Recursos online para exámenes y compartir contenido.

Impero Education Pro	NO	**Características:** Vendedor: *Impero Software* País: Reino Unido Fundado: 2002 URL: www.impero*software*.co.uk/products/education-pro-classroom-management-*software*/ Prueba gratis: Sí Despliegue: Instalado, Móvil, Web Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Ejecutar archivos/sitios web en máquinas seleccionadas, Deshabilitar/Habilitar Internet, Impresora, Memorias USB, Enviar y coleccionar archivos desde/hacia estudiantes, Añadir o adjuntar direcciones web e imágenes a mensajes instantáneos, Restringir usuarios hacia un listado de sitios web permitidos.
AB Tutor	NO	**Características:** Vendedor: *Globe Microsystems* URL: www.abtutor.com País: Reino Unido Fundado: 2000 Prueba gratis: Sí Despliegue: Instalado, Móvil Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Automáticamente lanzar sitios web o abrir archivos de forma remota, la red de vistas de tamaño reducido, ver y registrar la actividad de los estudiantes (aplicaciones, sitios web, impresión y pulsaciones de teclas).
LanSchool	NO	**Características:** Vendedor: *Stoneware* URL: www.lanschool.com/ País: Estados Unidos Fundado: 2000 Prueba gratis: Sí Despliegue: Instalado Soporte: Sí Licencia: Privativa

		Principales funcionalidades: Carrito inalámbrico y soporte 1: 1, Mostrar Estudiante a otros Estudiantes, Enviar y Recolectar Archivos y Directorios.
XClass	NO	**Características:** Vendedor: *Sun-Tech International Group* URL: www.suntechgroup.com País: Hong Kong Fundado: 1997 Despliegue: Instalado Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Inhabilitar la Web, Control de Aplicaciones, Conexión remota, Pizarra Interactiva.
Netop Vision Pro	NO	**Características:** Vendedor: *Netop* País: Dinamarca URL: http://www.netop.com/ Prueba gratis: Sí Soporte: Sí Licencia: Privativa **Principales funcionalidades:** Control de Acceso a Internet, NUEVO: Centro de Aprendizaje, Supervisar el trabajo de los estudiantes.

Una vez analizadas las aulas mencionadas anteriormente se concluye que a pesar de que cada una de estas posee gran cantidad de funcionalidades, no tienen módulos propios que le permitan al profesor crear su propio juego teniendo en cuenta la materia que desea impartir en un momento determinado, sino que utilizan juegos web así como disímiles herramientas web disponibles en internet para satisfacer este objetivo.

ATcnea a pesar que brinda las opciones de lanzar aplicaciones y sitios web de forma remota en el terminal de los estudiantes, no posee tampoco una herramienta que le brinde al profesor la posibilidad de incluir en cualquier momento de la clase juegos creados por él sobre un tema específico de la materia que imparte. Para ello tendría que acceder a la web para utilizar juegos que se encuentren disponibles en la misma o utilizar juegos que ya estén instalados

en los terminales de los estudiantes y que a su vez respondan a la materia que se encuentra impartiendo.

En la mayoría de los casos, cuando el profesor necesita utilizar un juego para lograr una mayor interacción de los estudiantes con el contenido se hace necesario buscar un juego que responda al tema en cuestión. Esto se hace un poco difícil debido a la gran diversidad de temas sobre los cuales se puede impartir una clase trayendo consigo que muchas veces no existen juegos disponibles en la web ni juegos desarrollados que complementen las necesidades formativas del profesor en un momento determinado por lo que se hace necesario incluir dentro del propio software un módulo para la creación de juegos del tipo observación. El módulo le permitirá al profesor incluir como parte de sus clases un juego el cual pueda ser elaborado teniendo en cuenta sus propias especificaciones y que a su vez satisfaga sus necesidades. Estos tipos de juegos se desarrollan con el objetivo de que el estudiante pueda desarrollar la habilidad de identifica, tan necesaria para el desarrollo de determinados niveles de asimilación por parte del estudiante en cualquier carrera o enseñanza.

1.5 Metodología de desarrollo

Una metodología de desarrollo de *software* es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto *software* desde que surge la necesidad del producto hasta que se alcanza el objetivo por el cual fue creado (Garzás, 2009).

1.5.1 Metodología de desarrollo para la Actividad productiva de la UCI

Para el desarrollo de la investigación se utilizará la metodología AUP en su variante UCI, utilizada por el equipo de desarrollo de ATcnea para la implementación del software en aras de cumplir con la metodología definida para la actividad productiva de la universidad, utilizándose para describir el sistema el escenario 4 encapsulando los requisitos en historias de usuarios.

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN), Descripción de Procesos del Negocio (DPN) o Modelo Conceptual (MC)) y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema (CUS), Historias de Usuario (HU) o Descripción de Requisitos por Proceso (DRP)), surgen cuatro escenarios en la metodología AUP en su variante UCI para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

Escenario No 1: proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.	
Escenario No 2: proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.	
Escenario No 3: proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.	
Escenario No 4: proyectos que no modelen negocio solo pueden modelar el sistema con HU.	

1.6 Ambiente de desarrollo

A continuación se exponen las herramientas, tecnologías y lenguajes de desarrollo y modelado a utilizar para darle solución al objetivo planteado. Para la selección de las mismas no se realiza un estudio debido a que se utilizarán las mismas empleadas por el equipo de desarrollo del software ATcnea. Seguidamente se muestra una breve descripción de cada una.

1.6.1 Lenguaje de desarrollo

Java es un lenguaje de programación concurrente y orientado a objetos, diseñado para ser ejecutado en cualquier dispositivo sin la necesidad de ser recompilado, esto gracias a que las aplicaciones en Java, son ejecutadas por una plataforma que es la encargada de interpretar la aplicación y ejecutarla. Siendo esta característica una de las mayores ventajas de este lenguaje, ha hecho que se encuentre dentro de los más usados en la actualidad para el desarrollo de *software*. Android como uno de los Sistema Operativo (SO) para móviles más usados, tiene como lenguaje nativo Java, convirtiéndolo en el lenguaje por excelencia de dicho SO (Tutorials, 2015).

1.6.2 Entorno Integrado de Desarrollo (IDE)

Un ambiente de desarrollo integrado (DE, por sus siglas en inglés) es una aplicación que proporciona un conjunto de herramientas y servicios para facilitar el desarrollo de *software*. Consta de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría tiene auto-completado inteligente de código y algunos contienen un compilador, un intérprete, o ambos (ORACLE, 2010).

1.6.2.1 Android Studio 2.0

En la actualidad Android Studio es el IDE por excelencia para el desarrollo en SO (Sistema Operativo) Android, (Sgoliver, 2014). Android Studio cuenta con una gran cantidad de prestaciones y recursos que facilitan el desarrollo, como: refactorización del código, renderización en tiempo real y plantillas para la creación rápida de aplicaciones.

La licencia bajo la que se encuentra Android Studio es Apache 2.0, que es una licencia de *software* libre y código abierto que permiten el uso del mismo sin problemas legales incluso en el ámbito de Cuba (Studio, 2016).

1.6.2.2 NetBeans 8.0

NetBeans IDE 8.0 cuenta con características que contribuyen a simplificar y agilizar el proceso de desarrollo, es multiplataforma y sin restricciones de licencia y tiene una gran comunidad de usuarios y desarrolladores de todo el mundo. Entre las características más

notables de este IDE cabe destacar que posee un editor que le permite a los desarrolladores programar código limpio y organizado, permite la integración con repositorios de control de versiones tales como Subversión, Mercurial, y Git., facilita el diseño de interfaces GUI para aplicaciones Java SE, HTML5, Java EE, PHP, C/C++, y Java ME y se integra de forma natural con el servidor web Apache. (ORACLE, 2010).

1.6.3 Lenguaje de modelado

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software UML puede usarse en las diferentes etapas del ciclo de vida del desarrollo del software cuenta con un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos. (Jacobson, 2000).

UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios (Cornejo, 2008). Para el desarrollo de la solución se hace uso en su versión 2.0.

1.6.4 Git

Git es un sistema de control de versiones distribuido que se origina a partir del desarrollo del kernel de Linux y es usado por muchos proyectos populares Open Source como son Android o Eclipse, así como tantos otros proyectos comerciales. La principal diferencia entre Git y cualquier otro sistema de control de versiones es cómo Git modela sus datos como un conjunto de instantáneas de un mini sistema de archivos. (Fiqus, 2010). Para el desarrollo de la solución se hace uso de la versión 2.7.4.

1.6.5 Herramienta CASE para el modelado

Visual Paradigm es una herramienta CASE que se utilizada para el modelado UML. La misma es de libre distribución y se caracteriza por confiabilidad y estabilidad, por este motivo, es la herramienta perfecta para Ingenieros de *Software*, Analistas de Sistemas, entre otros, que necesiten desarrollar bajo un sistema UML orientado a objetos (Software, 2016). Para el

desarrollo de la solución se hace uso de la herramienta en su versión 8.0, la misma será utilizada para el modelado de los diagramas mostrados en la investigación.

1.7 Conclusiones del Capítulo

Una vez concluido el desarrollo del capítulo se arriba a las siguientes conclusiones:

- Las aulas tecnológicas analizadas no cuentan con módulos para juegos sino que utilizan los juegos web así como disímiles herramientas web.
- Los videojuegos educativos garantizan la motivación en el proceso de enseñanza-aprendizaje.
- Las herramientas y tecnologías seleccionadas, así como la metodología permitieron seleccionar el ambiente de desarrollo para la posterior implementación.

CAPÍTULO 2: CARACTERÍSTICAS Y DISEÑO DEL SISTEMA

2 Introducción

Para el desarrollo de una aplicación de *software* una de las partes más complicadas es decidir lo que se va a entregar como respuesta, algo en lo que tanto el cliente como el desarrollador deben quedar satisfechos. En el presente capítulo se realiza una descripción de la propuesta de solución a la situación problemática planteada. Se especifican y explican los requisitos funcionales y no funcionales y se describen las historias de usuario. También se realiza el diseño del módulo como parte del flujo de trabajo que propone AUP-UCI. Se generan los artefactos correspondientes al modelo del diseño, además se diseña el modelo de base de datos y el modelo de despliegue.

2.1 Modelo de Dominio

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes *software*. No se trata de un conjunto de diagramas que describen clases *software*, u objetos *software* con responsabilidades (Larman, 2003).

Es posible tomar como punto de inicio del proyecto al modelo de dominio para el diseño del sistema. Cuando se programa orientado a objetos el sistema de forma interna imita una realidad en cierta medida, por lo cual el mapa de conceptos del modelo de dominio representa una primera versión del sistema (Larman, 2003).

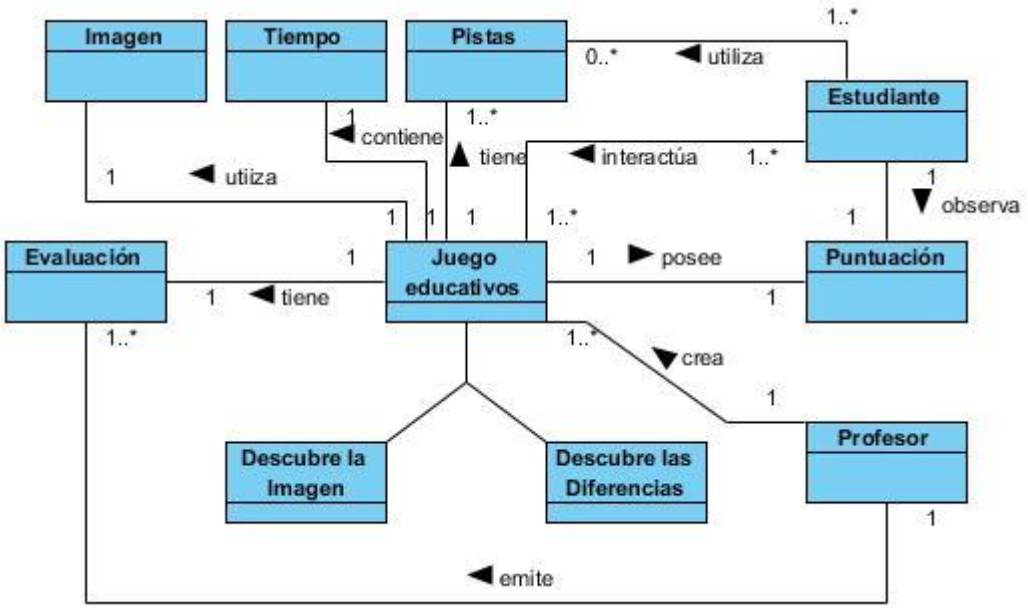


Figura 1 Modelo de Dominio

2.1.1 Definición de las clases del Modelo de Dominio

Se muestra una breve descripción de cada una de las clases que intervienen en el Modelo de Dominio:

Estudiante: Usuario que interactúa con el juego educativo creado.

Profesor: Usuario encargado de controlar y monitorear toda la clase mediante la PC, así mismo será el encargado de crear el juego teniendo en cuenta los contenidos educativos a impartir en una clase determinada, asignando a cada juego una imagen, un tiempo y la cantidad de pistas que desee.

Juegos educativos: Juegos creados por el profesor con fines educativos que podrán ser utilizaos en una clase.

Imagen: Representación visual de los contenidos educativos utilizados por el profesor en una clase y que puede ser utilizado en la elaboración de un juego.

Evaluación: Nota que emite el profesor a cada estudiante cuando termina de jugar.

Descubre la imagen: Juego de observación del tipo descubrir la imagen o identificar objetos dentro de la imagen.

Descubre las diferencias: Juego de observación del tipo descubrir las diferencias entre imágenes.

Pistas: El profesor las especifica en el juego que crea, mientras que el estudiante las puede utilizar a su favor para dar respuestas a los juegos.

Tiempo: El profesor debe asignar el tiempo que cada juego debe durar.

Puntuación: Cada juego tiene un valor asignado y cada estudiante puede ver mientras interactúa con el juego que puntuación va obteniendo.

2.2 Propuesta de solución

La solución que se propone tiene como objetivo apoyar el aprendizaje de los estudiantes que interactúan con el producto ATcnea mediante la utilización de juegos del tipo observación acorde a los contenidos que están recibiendo. Dicha solución permite la creación de dos tipos de juego: Descubre la imagen y Descubre las diferencias.

El módulo a su vez está compuesto por dos aplicaciones: uno encargado de la gestión del juego con el cual el profesor tendrá interacción directa y otro para el estudiante el cual se encarga de gestionar toda la lógica del juego recibido.

El módulo a desarrollar permite al profesor la creación, modificación y eliminación de estos tipos de juegos personalizados según las necesidades de la clase, de una manera rápida y sencilla además de poder enviárselo a los estudiantes en el momento deseado. Para realizar cualquier acción entorno a los juegos, el profesor debe acceder a la opción *Administrar juego* donde se mostrará el listado de juegos.

Para crear cualquiera de los tipos de juegos el módulo brinda al profesor la opción de introducir diferentes datos tales como: el nombre del juego, el tiempo que posee el estudiante para interactuar con el mismo y la cantidad de pistas. Estas últimas representan algunas de las selecciones realizadas por el profesor por lo que siempre serán menor o igual que la cantidad de objetos a encontrar. Así mismo permite al profesor cargar una imagen cualquiera acorde al contenido con el que desee vincular el juego, una vez cargada la imagen el sistema se encargará de redimensionar automáticamente la misma a la resolución (1208x530px). En

el caso específico del tipo de juego Descubre las diferencias el profesor deberá cargar una imagen simétrica para garantizar que contenga las diferencias a encontrar ubicadas en ambas mitades de la imagen en la misma posición una con respecto a la otra.

El profesor será el encargado de realizar las selecciones de objetos a encontrar, tantas como este estime conveniente, ubicando las coordenadas de dicha selección en la imagen ya sea de forma rectangular o de forma elíptica. Al seleccionar el objeto a encontrar el sistema debe permitir al profesor indicar el nombre del mismo, el cual será usado en el caso del juego descubre la imagen como objeto a encontrar y que se incluirá en el listado de objetos. Para el caso del tipo de juego Descubre las diferencias el profesor solo realizará las selecciones de las diferencias en la primera mitad de la imagen, la aplicación del estudiante se encargará por si misma de ubicar en ambas mitades la(s) diferencia(s) encontrada(s) resaltando en cada una que posee diferente.

Una vez creado el juego el profesor podrá realizar las acciones de editar o eliminar dicho juego siempre seleccionando el juego al cual le desea realizar la acción. En caso de que el profesor desee eliminar un juego el sistema debe pedir confirmación para realizar la acción. El profesor podrá editar un juego modificando los datos establecidos para la creación del mismo.

El profesor podrá enviar el juego creado cuando estime conveniente a uno o más estudiantes siempre que se encuentren conectados a la clase, visualizando estos en cada uno de sus terminales la cantidad de pistas disponibles, el tiempo de juego en decremento, la lista de los nombres de los objetos a encontrar, la imagen sobre la cual deben buscar los objetos o las diferencias así como la puntuación que irán obteniendo a medida que identifiquen un objeto.

La puntuación que los estudiantes irán obteniendo se establece teniendo en cuenta el porciento que representa un objeto del total de objetos a encontrar, en caso de que el estudiante utilice una pista el sistema no dará puntuación por el objeto mostrado. Por otro lado, cada vez que el estudiante seleccione en un lugar de la imagen donde deberá encontrar los objetos o las diferencias y esta selección no se corresponda con ninguna diferencia u

objeto de los que debe encontrar, el estudiante será penalizado restándole 10 segundos al tiempo de juego.

Una vez concluido el tiempo o encontrados todos los objetos el sistema enviará al profesor una notificación especificando que uno o más estudiantes han terminado el juego, permitiendo al mismo visualizar los resultados obtenidos por los estudiantes. Dichos resultados se mostrarán en un listado donde se visualizará el nombre del estudiante, la fecha en que jugó el juego, el nombre del juego, el tipo de juego, y la evaluación obtenida por el estudiante.

La evaluación obtenida se calculará haciendo uso del porciento de objetos encontrados, teniendo en cuenta la evaluación establecida en los ajustes de la clase, cualitativa (M, R, B, E), y cuantitativa. La evaluación cuantitativa se realizará de 2 a 5 puntos para el caso de que en la configuración de la clase el profesor haya seleccionado la escala de evaluación del nivel de enseñanza del Ministerio de Educación Superior (MES) y de 0 a 100 puntos en caso de que haya seleccionado la escala de evaluación para el nivel de enseñanza del Ministerio de Educación (MINED).

2.3 Especificación de requisitos funcionales

La ingeniería de requisitos del *software* es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al *software*. La especificación de requisitos parece una tarea sencilla pero no lo es, la misma permite especificar las características operacionales del *software* (función, datos y rendimientos), indica la interfaz del *software* con otros elementos del sistema y establece las restricciones que debe cumplir el *software* (Pressman, 2002).

2.3.1 Requisitos funcionales

Un requisito funcional define una función del sistema de *software* o sus componentes. Pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir (Sommerville, 2006). Luego de analizada la propuesta de solución se definieron los siguientes requisitos funcionales:

Tabla 2 Descripción de los requisitos funcionales

Requisitos de la aplicación Desktop		
Nº	**Nombre**	**Descripción**
RF 1	Crear juego	El sistema debe permitir al profesor crear un juego en el sistema debe permitir especificar los siguientes datos: • Nombre del juego • Tipo de juego • Tiempo de juego • Cantidad de pistas • Cargar imagen
RF 2	Editar juego	El sistema debe permitir al profesor editar un juego creado con anterioridad y también modificar los siguientes datos: • Nombre de juego • Tipo de juego • Tiempo de juego • Cantidad de pistas • Cargar imagen
RF 3	Eliminar juego	El sistema debe permitir al profesor eliminar un juego creado con anterioridad.
RF 4	Cargar imagen	El sistema debe permitir al profesor cargar una imagen para crear un juego.
RF 5	Guardar fichero de juego	El sistema debe permitir al profesor guardar el fichero del juego creado, para ello debe estar autenticado en el sistema y debe estarse creando un juego.
RF 6	Redimensionar imagen	Una vez cargada la imagen para crear un juego el sistema debe redimensionar la misma de forma automática.

RF 7	Enviar juego a la aplicación cliente	El sistema debe permitir al profesor enviar un juego creado con anterioridad a la aplicación cliente.
RF 8	Seleccionar juego a enviar	El sistema debe permitir al profesor seleccionar del listado de juegos el juego que desea enviar a los estudiantes.
RF 9	Listar juegos	El sistema debe permitir al profesor visualizar el listado de juegos creados en el sistema.
RF 10	Incluir nombre de la selección	El sistema debe permitir al profesor una vez seleccionada el área a identificar, especificar el siguiente dato: • Nombre de la selección
RF 11	Listar objetos a encontrar	El sistema debe mostrar en el juego enviado a los estudiantes un listado con los objetos a encontrar.
RF 12	Buscar juego en la lista de juegos	El sistema debe permitir buscar un juego en el listado de juegos especificando el siguiente dato: • Nombre del juego
RF 13	Listar estudiantes	El sistema debe permitir al profesor visualizar el listado de estudiantes conectados a la clase a los cuales les enviará el juego.
RF 14	Mostrar resultados del juego	El sistema debe permitir al profesor visualizar el resultado obtenido por los estudiantes en un juego. En el listado se mostrarán los siguientes datos: • Recibido de • Tipo de juego • Nombre de juego • Evaluación

RF 15	Eliminar resultados	El sistema debe permitir al profesor eliminar los resultados obtenidos del listado de resultados.
RF 16	Notificar recibo de resultados	El sistema debe notificar al profesor al recibir los resultados de los estudiantes.
Este requisito se manifiesta en ambas aplicaciones con un uso diferente		
RF 17	Seleccionar área	El sistema debe permitir tanto al profesor como a los estudiantes seleccionar un área en la imagen, para ello los usuarios deben estar autenticados en el sistema como profesor o como estudiante, cada uno en su aplicación.
Requisitos de la aplicación Android		
Nº	**Nombre**	**Descripción**
RF 18	Notificar recibo de juego	El sistema debe notificar a los estudiantes una vez que el profesor envíe un juego.
RF 19	Mostrar juego	El sistema debe permitir al estudiante visualizar el juego enviado por el profesor y además mostrar los siguientes datos: • Cantidad de pistas • Tiempo del juego • Puntuación
RF 20	Mostrar pistas	Una vez que el estudiante se encuentre interactuando con el juego y acceda a la opción Pistas el sistema debe mostrar las pistas especificadas por el profesor al crear el juego.
RF 21	Mostrar puntuación	El sistema debe mostrar al estudiante a medida que el mismo interactúe con el juego la puntuación que se va alcanzando una vez que identifique los objetos o las diferencias en las imágenes.
RF 22	Mostrar objeto	El sistema debe mostrar una vez que el estudiante seleccione un objeto en la imagen que dicho objeto se muestre.

RF 23	Mostrar tiempo	El sistema debe mostrar el tiempo de forma automática en la parte superior derecha una vez que el estudiante ejecute el juego.

Los requisitos Mostrar juego, Mostrar pistas, Mostrar puntuación, Mostrar tiempo y Mostrar objetos forman parte del paquete Mostrar.

2.3.2 Requisitos no funcionales

En la ingeniería de *software*, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar (Sommerville, 2006).

Seguridad

RNF1: El sistema debe garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo al tipo de usuario.

Usabilidad

RNF2: Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad (Universidad de las Ciencias Informáticas, 2014).

Software

RNF 3: El sistema debe permitir la instalación de la herramienta en tabletas que cuenten con Android en la versión 5.0 y 6.0.

RNF 4: Permitir la instalación de la herramienta en PC que cuenten con Nova 2015 o superior.

Hardware

RNF 5: Se recomienda 1 GB RAM o superior, 7 pulgadas o superior de pantalla, resolución de 1280x800px o superior, CPU/GPU *Dual Core* 1.0GHz o superior, para las tabletas.

RNF 6: Se recomienda que tenga 4GB RAM o superior, CPU *Core* 2 E6300 o superior, 1GB Tarjeta de video o superior, para las pc.

2.4 Descripción de los usuarios del sistema

Tabla 3 Descripción de los usuarios del sistema

Usuario	Descripción
Profesor	Encargado de gestionar los juegos que tendrá el sistema.
Estudiantes	Encargado de brindar solución a los juegos y recibir evaluación.

2.5 Historias de usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del *software*. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento del proyecto las historias de usuario pueden romperse en varias historias de usuario, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en solo unas semanas (Goggi, 2012).

A continuación se muestra la HU del requisito funcional crear juego, el resto de las HU de los restantes requisitos funcionales se encuentran en el Anexo#1.

Tabla 4 HU requisito funcional crear juego

Número: 1	**Nombre del requisito:** Crear juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:**30días

Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir la creación de un juego en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para la creación de un juego:

- El usuario debe estar autenticado como profesor.
- Realizar la carga de los datos necesarios para la creación del juego.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor crear un juego en el sistema accediendo a la opción Crear.

Una vez seleccionada la opción el sistema debe permitir especificar los siguientes datos:

- Tipo de juego
- Tiempo de juego
- Cantidad de pistas
- Cargar imagen
- Nombre del juego

Una vez cargada la imagen el sistema debe permitir realizar las siguientes acciones:

- Seleccionar área
- Limpiar pantalla

Además debe permitir realizar una de las siguientes acciones:

- Aceptar
- Cancelar

Al seleccionar la opción Aceptar el sistema crea y guarda el juego.

Al seleccionar la opción Cancelar el sistema cierra la ventana de crear.

En caso de que haya campos vacíos o campos incorrectos se debe mostrar un

mensaje de información.
Observaciones: La creación de un juego permite al profesor tener más herramientas de evaluación.
Prototipo de interfaz: 

2.6 Patrón Arquitectónico

Un patrón arquitectónico expresa un esquema estructural para la construcción de un sistema. Son estrategias que ofrecen soluciones a problemas de arquitectura de *software* en ingeniería de *software*. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados (Tabares, 2015).

2.6.1 Patrón Modelo - Vista - Controlador

El marco de trabajo tanto de Android como JavaFX está basado en el patrón arquitectónico Modelo Vista Controlador (MVC). Este patrón divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

Modelo *(Model)*: Administra y maneja todo lo relacionado con los datos del sistema, da respuesta a peticiones de información sobre el estado de la aplicación y responde con instrucciones de cambio de estado (generalmente del controlador a la vista) (Welicki, 2016).

Vista (*View*): Obtienen los datos del modelo y muestran la información al usuario. Cada vista tiene asociado un componente controlador (Welicki, 2016). Las vistas están representadas por ficheros escritos en xml para Android y fxml para JavaFX que se encargan de construir las vistas en los formatos anteriormente mencionados con las que interactúa el usuario.

Controlador (*Controller*): reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("*service requests*") para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores (Welicki, 2016).

2.6.2 MVC en Android

En Android el patrón arquitectónico MVC posee una particularidad en cuanto a la vista y el controlador, definiéndose estos dos como un par Controlador-Vista. En este caso particular el controlador notifica al modelo y el modelo notifica a todos los dependientes (pares Controlador-Vista) asociados a él. Para este par la vista contiene una instancia de su controladora y la controladora además posee una instancia de la vista. Las vistas (*Layout*) son archivos *xml* que contienen los componentes (*widgets*) que conforman la Interfaz de Usuario (UI), las clases controladoras (*Activity*/*Fragment*) contienen las instancias de cada uno de estos componentes responsabilizándose por el control de los mismos, mientras que el modelo contiene una estructura de datos de objetos complejos o simplemente datos primarios.

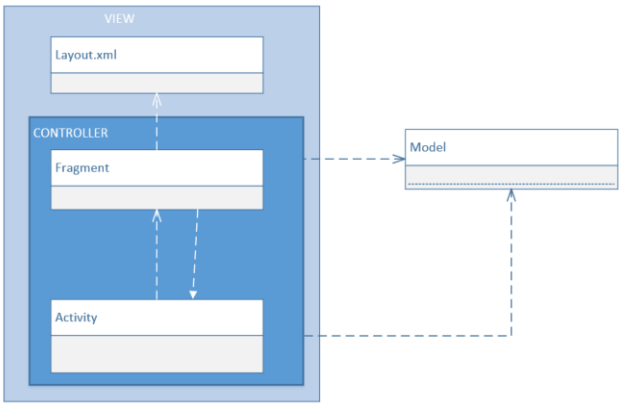


Imagen 4 Patrón arquitectónico Modelo Vista Controlador para Android

2.6.3 MVC en Java FX

En el *framework* Java FX la vista es un archivo fxml que solo contiene la referencia del componente. Se comunica con su controladora a través de eventos ejecutados desde la interfaz visual. La clase controladora tiene la responsabilidad de inicializar los elementos y acceder a los datos de los componentes. Por esta razón se puede considerar al subconjunto vista-controlador como una única unidad lógica.

La siguiente imagen muestra la composición del *framework*, donde el escenario es el elemento principal de la vista, seguidamente le sigue la escena. Un escenario puede estar contenido por disímiles escenas. Las escenas contienen los componentes visuales de la aplicación, dígase: *radiobutton*, *checkbox*, *textfield, button, pane*, *anchor pane*, etc. Conocer el ciclo de vida de estas aplicaciones que utilizan el modelo-vista-controlador garantiza su correcta implementación.

Imagen 5 Patrón arquitectónico Modelo Vista controlador para Java FX

2.6.4 MVC en el módulo desarrollado

El módulo está implementado sobre la misma arquitectura del producto ATcnea por lo que se utilizó al igual que en este el patrón arquitectónico MVC.

A continuación se muestra una imagen en la que se evidencia el uso del patrón MVC en el módulo desarrollado, la cual representa las clases controladoras, las vistas y el modelo en cada uno de las aplicaciones. Además se muestra un paquete conexión que contiene los comandos *ImageGameComand.java* encargado de enviar desde la aplicación del profesor los datos necesarios del juego para su visualización en la aplicación del estudiante y *EvaluateImageGameCommand.java* encargado de enviar desde la aplicación del estudiante la puntuación obtenida por este después de concluido el juego a la aplicación del profesor para su posterior evaluación. El envío y recepción de estos comandos se realiza mediante el empleo de un *socket* servidor en la aplicación del profesor y un *socket* cliente en la aplicación del estudiante. Cada uno de estos se encuentra a la escucha de que se envíen o se reciban datos, retroalimentándose a través de la función *execute()*, la cual se encarga de realizar una operación determinada una vez se ha recibido un comando:

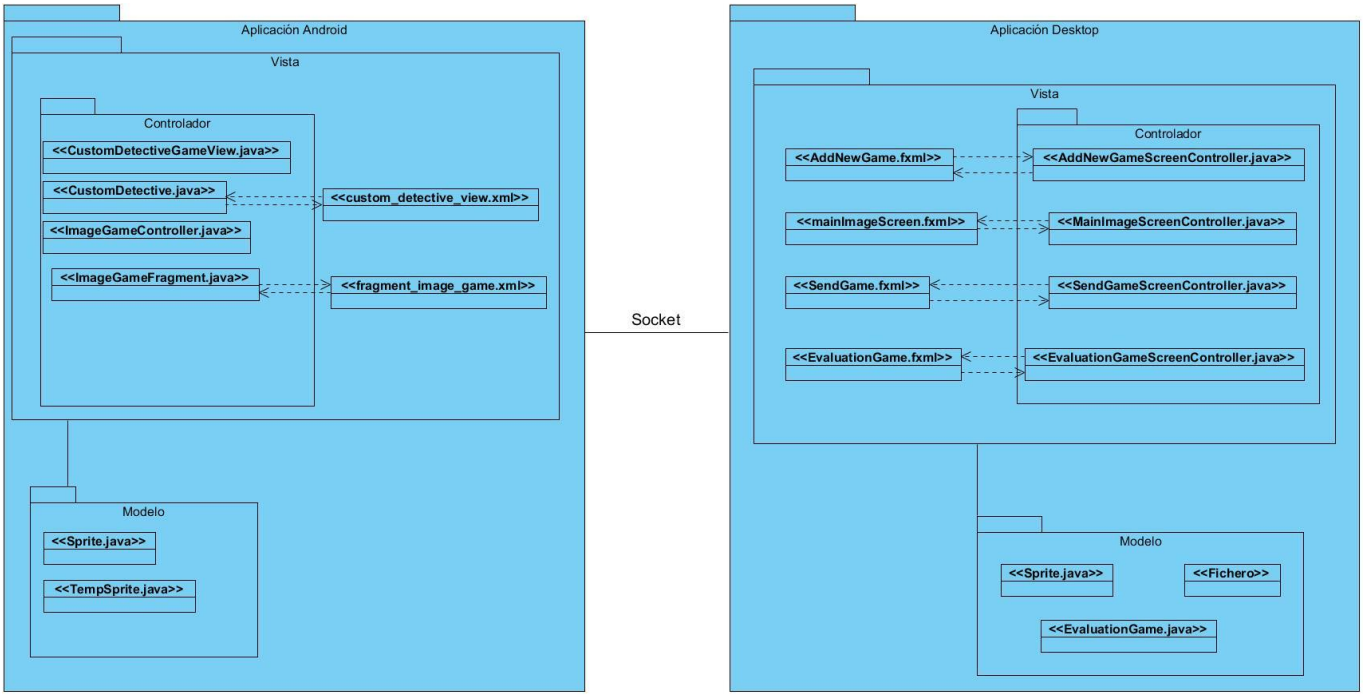


Figura 2 Arquitectura del módulo desarrollado

2.7 Patrones de diseño

Los patrones de diseño son soluciones a problemas repetidos en la construcción de *software* y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos (Kaisler, 2005). A continuación se describen los patrones de diseños utilizados en el desarrollo de la solución utilizados en el presente trabajo.

2.7.1 Patrones GRASP

Los Patrones de Asignación de Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (UML, 2016).

- **Creador**: se encarga de guiar la asignación de responsabilidades relacionadas con la creación de objetos. Su intención es encontrar un creador que necesite conectarse al objeto creado en alguna situación (Kaisler, 2005). En el sistema, la clase

AddNewGameScreenController.java es la encargada de “crear” instancias de la entidad *BGraphic.java*.

• **Controlador**: el patrón controlador funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la interfaz quien recibe los datos del usuario y los envía a las distintas clases según el método invocado (Kaisler, 2005). El uso de este patrón se evidencia en la clase *MainImageScreenController.java* que permite acceder a la información disponible de los juegos en los ficheros almacenados.

• **Alta cohesión**: las responsabilidades que almacena una clase deben ser pequeñas y enfocadas. Los métodos y atributos deben de ser sencillos para implementar dichas responsabilidades (Pressman, 2010). El controlador *MainImageScreenController.java* asigna responsabilidades de controlar el flujo de eventos del sistema a clases específicas. Este delega en otras clases (ejemplo *AddNewGameScreenController.java*) las actividades con las que mantiene un modelo de alta cohesión.

```
private void ButtonAction() {
 addButton.setOnAction(new EventHandler<ActionEvent>() {
     @Override
     public void handle(ActionEvent event) {
         myScreenManager.openWindow(SceneConfigGame.ADD_NEW_GAME,
                 ATcneaSinglenton.getInstance().getI18N().getString("manage.game.add.label")
                 ,StageStyle.UTILITY,addNewGameScreenController);
     }
 });
```

Imagen 6 Patrón Alta cohesión

• **Experto:** consiste en realizar una correcta asignación de responsabilidades en las clases utilizadas de tal forma que la tarea de implementar un método o crear un objeto, debe recaer sobre aquella que conoce toda la información necesaria para llevar a cabo dicha acción. De este modo se obtiene un diseño con mayor cohesión, la información se mantiene encapsulada y se disminuye el acoplamiento (Carmona, 2012). La clase

AddNewGameScreenController.java es la encargada de crear el fichero que contiene los datos del juego creado.

```
public class AddNewGameScreenController extends AnchorPane implements Initializable, ControlledScreen {
    ScreenManager myController;
    ResourceBundle i18n;
    //Tipo de herramienta en uso
    private BGraphic.FigureType toolInUse;
    private ATOptionButton btnSelected;

    //Objeto de la pizarra
    private BGraphic board;

    //Objeto en edicion
    private BGraphic remoteTools, editTool;

    //Variable global de uso de la pizarra
    private boolean isCreate, isEditing;
```

Imagen 7 Patrón Experto

2.7.2 Patrones Gof

Son patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño usando modelado UML. Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Gómez Giraldo, y otros, 2011).

• **Singleton** (Instancia única): este garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. La clase CustomDectectiveGameView.java es la controladora del juego, y maneja toda la lógica y recursos, por este motivo es necesario tener una única instancia de la misma y que se encuentre disponible en cualquier momento.

```
public class CustomDetectiveGameView extends SurfaceView implements SurfaceHolder.Callback {
    public class SurfaceThread extends Thread {
        private SurfaceHolder surfaceHolder;
        private CustomDetectiveGameView view;
        public boolean run;

        public SurfaceThread(SurfaceHolder surfaceHolder,CustomDetectiveGameView view){
            this.surfaceHolder=surfaceHolder;
            this.view= view;
            run = false;
```

Imagen 8 Patrón Singleton

• **Decorator:** este patrón responde a la necesidad de añadir características o funcionalidades a objetos de forma dinámica. La aplicación de este patrón en el manejo de las interfaces gráficas permite definir plantillas extensibles y ampliamente reutilizables (UML, 2016). La clase *CustomDetectiveView.xml* contiene la plantilla global para el componente encargado de mostrar los datos relacionados con el tiempo, la puntuación y las pistas que es común para los dos tipos de juegos que el sistema permite crear.

```
public class CustomDetective extends LinearLayout {
    private TextView txtView;
    private ImageView imgView;
    private TextView txtView2;
    private TextView txtViewTime;
    int time = 300;
    CustomDetectiveGameView customDetectiveGameView;
```

Imagen 9 Patrón Decorator

2.8 Diagramas de clases del diseño

Los diagramas de clases del diseño (DCD) representan la vista estática del sistema y modelan los conceptos asociados al dominio de la aplicación así como los conceptos internos definidos para la parte de la implementación. No se describe el comportamiento del sistema dependiente del tiempo y se representa mediante clases y sus relaciones (Jacobson, 2000). A continuación se muestra el DCD perteneciente al aplicación desarrollado en JavaFX:

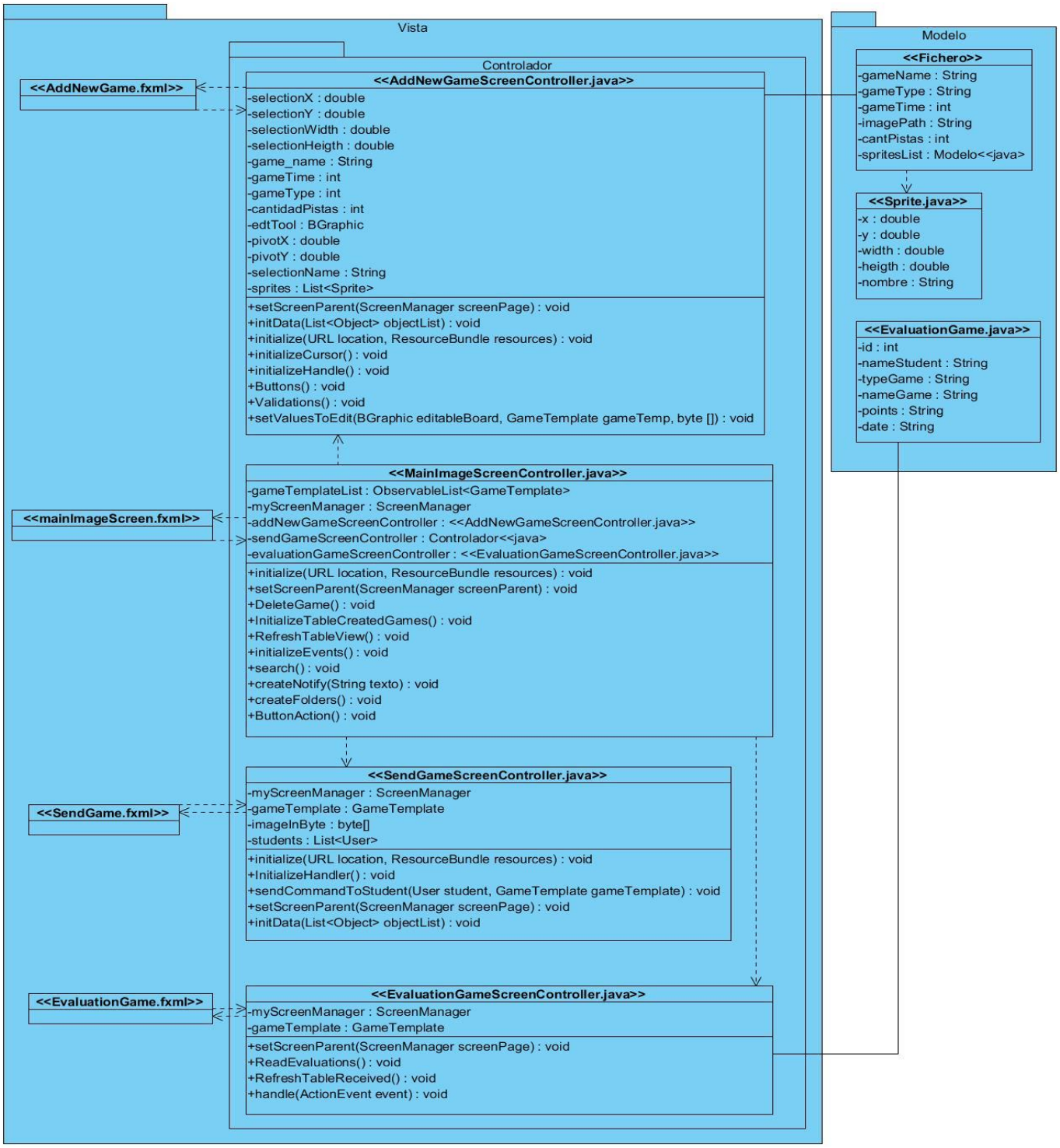


Figura 3 Diagrama de clases de la aplicación JavaFX

A continuación se muestra el diagrama de clases del diseño de la aplicación desarrollada en Android:

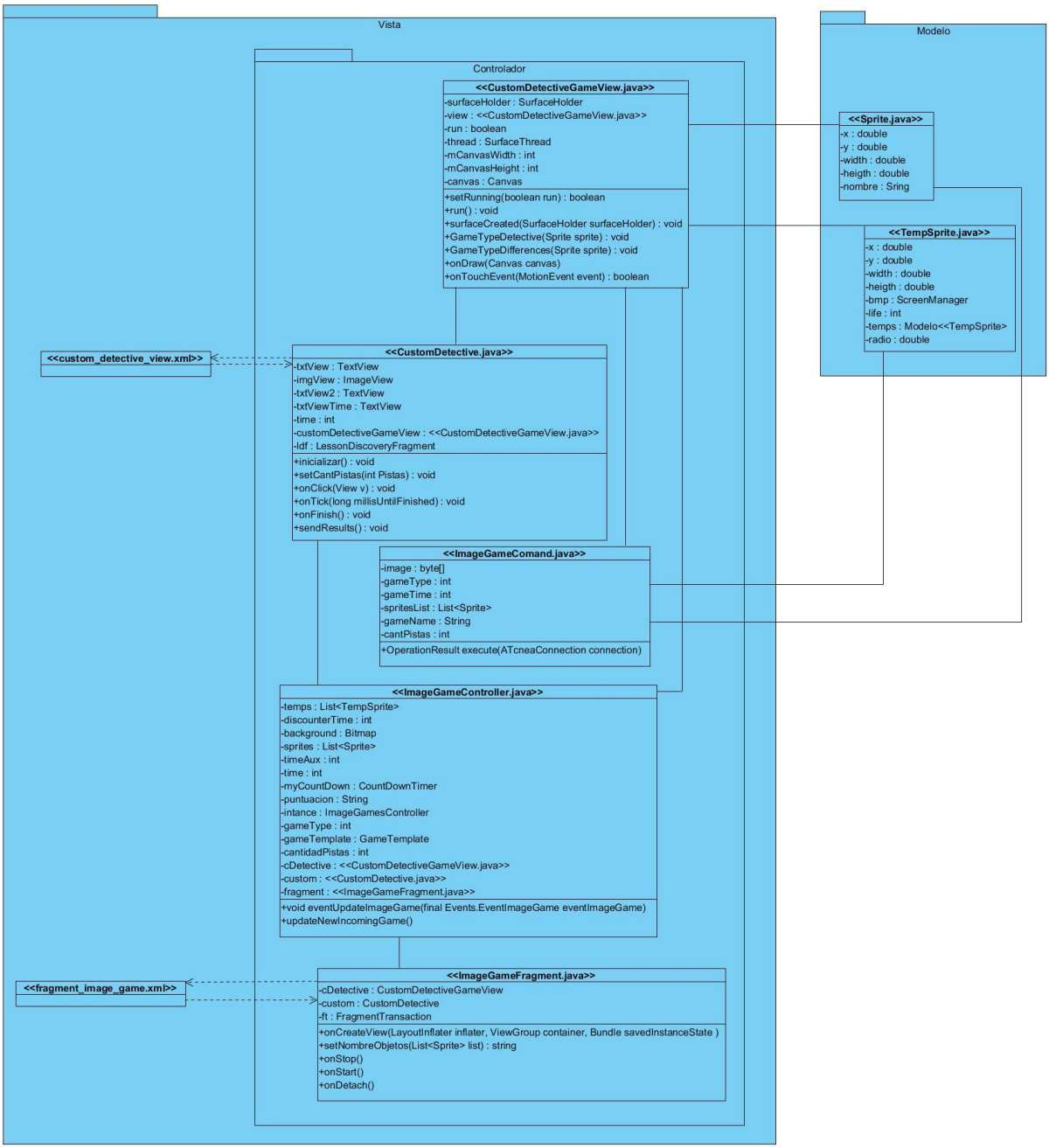


Figura 4 Diagrama de clases de la aplicación Android

2.9 Modelo de Datos

Un modelo de datos es un modelo abstracto que organiza elementos de datos y estandariza cómo se relacionan entre sí y con las propiedades del mundo real (McCaleb, 1999). El mismo tiene el propósito de garantizar que los datos persistentes sean almacenados de manera coherente y eficaz, así como definir el comportamiento que debe ser implementado en la base de datos (Pressman, 2010).

2.9.1 Descripción de la tabla del Fichero

Para el desarrollo del módulo se hace uso de un fichero donde se almacenan los datos de cada juego y un Sprite para recopilar los datos asociados a las selecciones. Seguidamente se muestra el diagrama entidad-relación correspondiente a la solución:

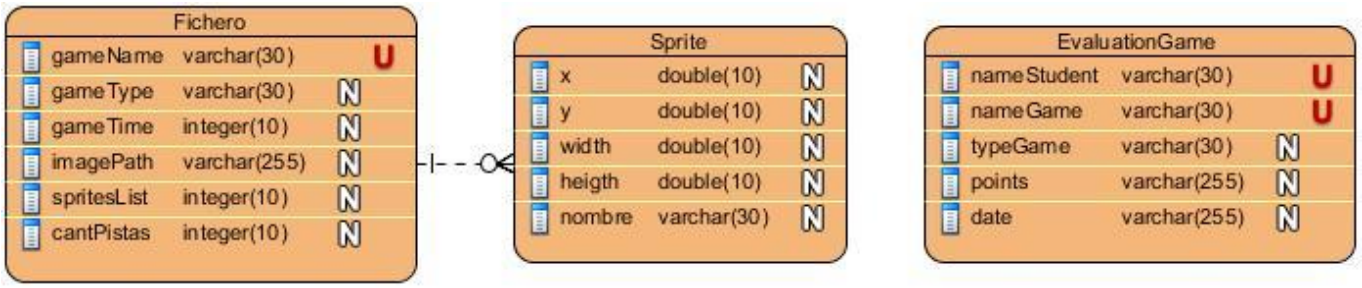


Figura 5 Modelo de datos

A continuación se presenta la descripción de la tabla Fichero para una mejor comprensión la descripción de las restantes se encuentran en el Anexo 2:

Tabla 5 Descripción de los atributos del fichero

Atributos	Tipo	Descripción
gameName	varchar(30)	Nombre del juego, no mayor de 30 caracteres alfanuméricos.
gameType	varchar(30)	El tipo de juego se define en os tipos, Descubre la imagen y al tipo Encuentra las diferencias.
gameTime	integer(10)	El tiempo que posee cada juego.

imagePath	varchar(30)	Especifica la imagen que tendrá el juego en el fondo, es decir, sobre la que se juega.
spritesList	integer(10)	Contiene la lista de coordenadas de los puntos a encontrar y además el nombre del objeto que se encuentra en cada uno de dichos puntos.
cantPistas	integer(10)	La cantidad de pistas que posee cada juego.

Tabla 6 Descripción de los atributos de la tabla Sprite

Atributos	**Tipo**	**Descripción**
x	Doublé	Coordenada en el eje x de la selección.
y	Doublé	Coordenada en el eje y de la selección.
width	Doublé	Ancho de la selección.
heigth	Doublé	Alto de la selección.
nombre	varchar(30)	Nombre de la selección.

Tabla 7 Descripción de los atributos de la tabla EvaluationGame

Atributos	**Tipo**	**Descripción**
nameStudent	varchar(30)	Nombre del estudiante
nameGame	varchar(30)	Nombre del juego
typeGame	varchar(30)	Tipo de juego
points	varchar(255)	Puntuación del estudiante en el juego
date	varchar(255)	Fecha en que se juega

2.10 Diagrama de Despliegue

El Diagrama de despliegue modela la arquitectura en tiempo de ejecución del sistema, en él se representan las dependencias entre los componentes para poder determinar el impacto de un cambio, queda especificado qué *hardware*, sistemas operativos, *software* de interfaces y soporte conformarán el nuevo sistema (Rumbaugh, y otros, 2000).

A continuación se presenta el diagrama que representa la distribución física del sistema a través de nodos, el mismo está compuesto por una computadora (laptop) que deberá tener instalado JAVA 8.0 y sistema operativo NOVA 2015 o superior y por las terminales de los estudiantes que en este caso serán las tabletas que deberán tener instalado el sistema operativo NOVADROID 6.0. La comunicación entre la laptop donde el profesor creará la clase y las tabletas de los estudiantes se realizará a través del protocolo UDP (Protocolo de Datagrama de Usuario) mientras que las tabletas se conectarán con la laptop del profesor a través del protocolo TCP (Protocolo de Control de Transmisión). La conexión entre ambos dispositivos se realizará mediante conexiones inalámbricas (WIFI) a través de un punto de acceso.

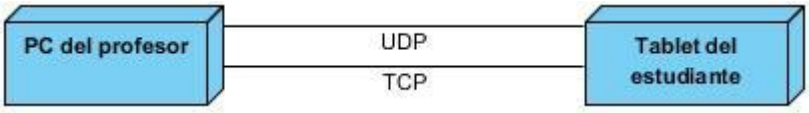


Figura 6 Diagrama de despliegue

2.11 Conclusiones del Capítulo

Una vez concluido el presente capítulo se arribó a las siguientes conclusiones:

- La elaboración del modelo de dominio estableció un punto de partida para el diseño de la solución.
- La identificación de los requisitos funcionales y no funcionales del sistema, y su descripción mediante historias de usuario, sirvió de guía para la implementación de las distintas funcionalidades de la solución propuesta.

- Los artefactos generados durante el diseño de la solución contribuyeron al mejor entendimiento del sistema para dar paso a la implementación de la solución propuesta.
- La selección de un fichero como modelo de datos permitió un uso más eficiente de los datos que se almacenan en dicho módulo.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3 Introducción

En este capítulo se describen los principales aspectos relacionados con la implementación de los elementos identificados durante el diseño de la propuesta de solución. Incluye el diagrama de componentes, que muestra la organización y las dependencias lógicas entre componentes. Se describe además la estrategia de prueba a seguir para validar el correcto funcionamiento del sistema, así como los principales resultados del proceso de ejecución de dichas pruebas.

3.1 Estándares de codificación

Las convenciones de código son importantes para los programadores por un gran número de razones (UCI, 2016):

- El 80% del coste del código de un programa va a su mantenimiento.
- Casi ningún *software* lo mantiene toda su vida el auto original.
- Las convenciones de código mejoran la lectura del *software*, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- Si distribuyes tu código fuente como un producto, necesitas asegurarte de que está bien hecho y presentado como cualquier otro producto.

3.1.1 Convenciones de nombres

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código (UCI, 2016).

3.1.2 Paquetes

Reglas para nombrarlos: El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente com, edu, gov, mil, net, org, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981. Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. Dichas convenciones pueden especificar que algunos nombres de los directorios correspondan a divisiones, departamentos, proyectos o máquinas (UCI, 2016).

Ejemplos

com.sun.eng

com.apple.quicktime.v2

edu.cmu.cs.bovik.cheese

3.1.3 Clases

Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML) (UCI, 2016).

class Cliente;

class ImagenAnimada;

3.1.4 Interfaces

Los nombres de las interfaces siguen la misma regla que las clases.

interface

ObjetoPersistente;

interface Almacen;

3.1.5 Métodos

Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula (UCI, 2016).

```
ejecutar();

ejecutarRapido();

cogerFondo();
```

3.1.6 Variables

Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje (UCI, 2016).

Los nombres de las variables deben ser cortos pero con significado. La elección del nombre de una variable debe ser un mnemónico, designado para indicar a un observador casual su función. Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales. Nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres (UCI, 2016).

```
int i;

char c;

float miAnchura;
```

3.2 Diagrama de Componentes

El Diagrama de Componentes se usa para modelar la estructura del *software*, incluyendo las dependencias entre los componentes de *software*, los componentes de código binario, y los componentes ejecutables. En el diagrama de componentes se modelan los componentes del

sistema, a veces agrupados por paquetes, y las dependencias que existen entre componentes (y paquetes de componentes) (UML, 2016).

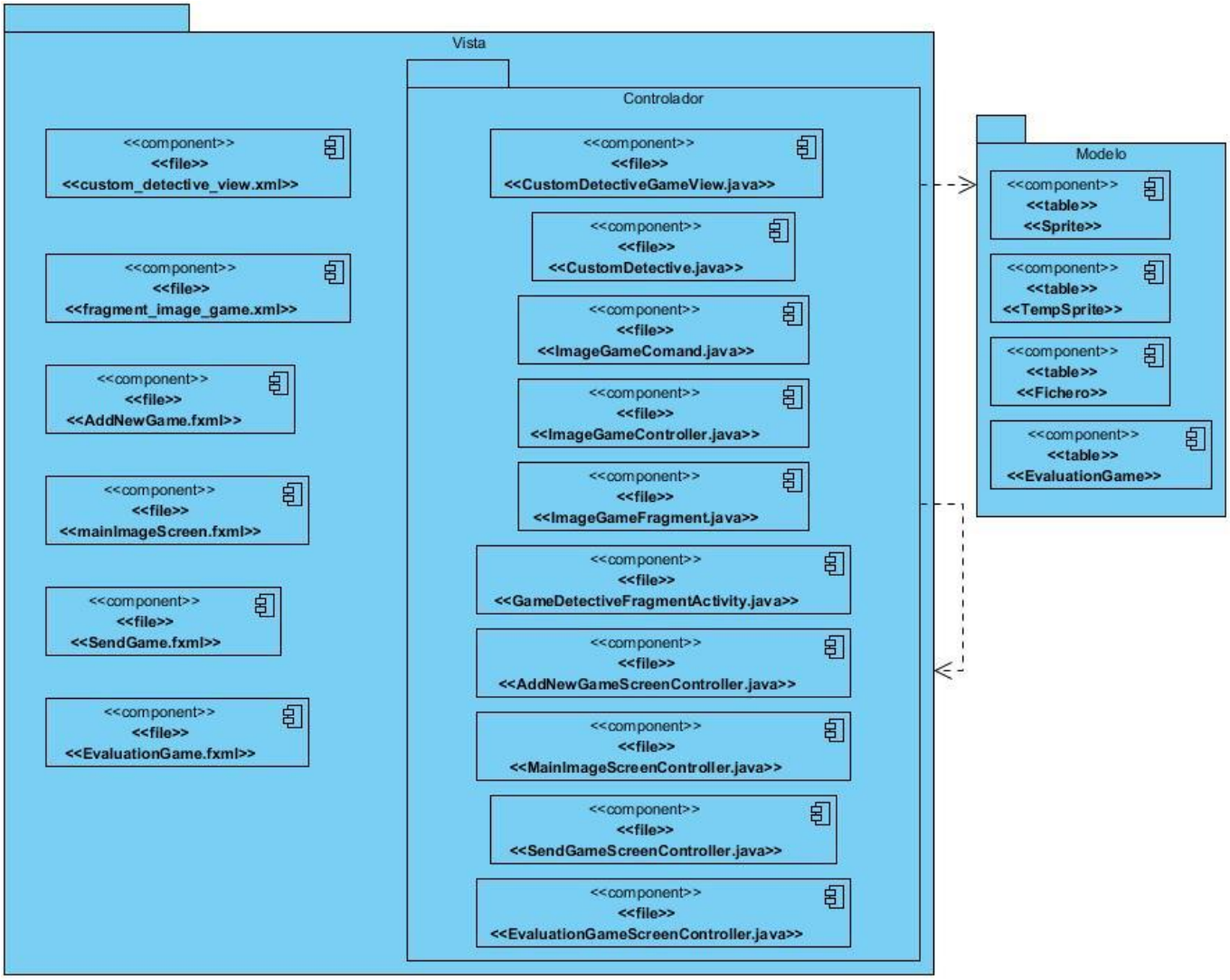


Figura 7 Diagrama de componente

3.3 Pruebas de *software*

Las pruebas de *software* son un proceso iterativo que implican ejercer una implementación del *software* con datos de prueba. Durante su aplicación se examinan las salidas del *software*

y su entorno operacional para comprobar que funciona tal y como se quiere. Las pruebas son una técnica dinámica de verificación y validación (Sommerville, 2005).

3.3.1 Estrategia de prueba

La estrategia de prueba describe el enfoque y los objetivos generales de la actividad de prueba. Incluye los niveles a que se realizarán las pruebas, el tipo de prueba a ser ejecutada y los métodos que se emplearán. Se especifican además las técnicas de prueba y herramientas que se usarán, así como los casos de prueba diseñados para lograr los objetivos. Debe permitir comenzar a evaluar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el *software* en su conjunto (Blanco, [2010]).

3.3.2 Niveles de prueba

Las pruebas de *software* se realizan en varios niveles, de acuerdo a determinados objetivos. Entre los niveles de prueba se encuentran: el nivel de pruebas del sistema y el nivel de pruebas de aceptación. Se detallan a continuación aspectos significativos de los mismos.

Pruebas de sistema: en este nivel se verifica el comportamiento del sistema en su conjunto, siendo el más adecuado para comprobar los requisitos no funcionales relacionados con la seguridad, velocidad, exactitud y fiabilidad. Los tipos de pruebas que se aplican permiten comprobar además utilidades, unidades físicas, entornos operativos e interfaces externas con otros sistemas (Blanco, [2010]).

Pruebas de aceptación: en este nivel se verifica el comportamiento del sistema frente a los requisitos del cliente, generalmente participa el mismo cliente o los usuarios.

3.3.3 Métodos de prueba

Un método de prueba se define como un procedimiento definitivo que produce un resultado de una prueba. Incluye la identificación, medición y evaluación de una o más cualidades, características o propiedades (Wilhelm, 2009). A nivel internacional en el desarrollo de *software*, se reconocen dos métodos de prueba fundamentales: caja blanca y caja negra. Se describen a continuación ambos métodos:

Prueba de caja blanca: se verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones. Hacen énfasis en la reducción de errores internos (Rodríguez, 2012). Se comprueban los caminos lógicos de *software*, examinando el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

Prueba de caja negra: Se verifica el correcto manejo de funciones externas provistas o soportadas por el *software*, así como que el comportamiento observado se apegue a las especificaciones del producto y las expectativas del usuario. Los casos de prueba se construyen a partir de las especificaciones del sistema. El componente se ve como una caja negra cuyo comportamiento solo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas.

El probador presenta las entradas al componente o sistema y examina las correspondientes salidas. Si las salidas no son las esperadas entonces la prueba ha detectado un problema con el *software* (Sommerville, 2005). No se requiere del conocimiento de la lógica del sistema, solo debe conocerse la funcionalidad que debe realizar.

Se diseñaron los casos de prueba según las funcionalidades descritas por cada historia de usuario. Se persigue con estos artefactos lograr una compresión específica de las condiciones que la solución debe cumplir. Cada planilla de casos de pruebas recoge la especificación de una historia de usuario, dividida en secciones y escenarios, detallando las funcionalidades descritas en ella y describiendo cada variable.

3.3.4 Diseño de casos de prueba

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en las que se diseñan las entradas y salidas esperadas, para probar el sistema. El objetivo de este proceso es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el *software* satisface sus requerimientos (Sommerville, 2005).

Los casos de prueba de caja negra pretenden demostrar que: las funciones del *software* son operativas, la entrada se acepta de forma correcta, se produce la salida esperada y la integridad de la información externa se mantiene.

A continuación se muestra el caso de prueba perteneciente al requisito funcional Cargar Imagen:

Tabla 8 Caso de prueba del requisito funcional Cargar imagen

Descripción general								
Permitir la creación de un juego en el sistema.								
Condiciones de ejecución								
Para la creación de un juego: - El usuario debe estar autenticado como profesor. - Realizar la carga de los datos necesarios para la creación del juego. Para la creación de un juego: - El usuario debe estar autenticado como profesor. - Realizar la carga de los datos necesarios para la creación del juego.								
Escenario	**Descripción**	**Cargar Imagen**	**Tiempo de juego**	**Nombre del juego**	**Cantidad de pistas**	**Tipo de juego**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Crear	Una vez en la vista Administrar juego el usuario selecciona la opción Crear.	N/A	N/A	N/A	N/A	N/A	El sistema debe permitir especificar los siguientes datos: (*) Nombre del juego. (*) Tipo de juego. (*) Cantidad de pistas, (*) Cargar imagen, (*) Tiempo de juego. El sistema permite además seleccionar una de las siguientes acciones: Aceptar y Cancelar.	Clase X/Administrar Juegos/Crear
EC 1.2 Opción Aceptar	Una vez especificados los datos para crear un juego el usuario selecciona la opción Aceptar.	V	V	V	V	V	El sistema crea el juego. Además actualiza y muestra el listado de los juegos, en el mismo se mostrará el Nombre y el Tipo de juego.	Clase X/Administrar Juegos/Crear/Aceptar
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	N/A	N/A	N/A	N/A	N/A	El sistema muestra la interfaz Administrar juegos.	Clase X/Administrar juegos/Crear/Cancelar
EC 1.4 Datos incorrectos	Al especificar los datos para crear un juego el usuario especifica datos incorrectos.	I	V	V	V	V	El sistema muestra un mensaje de información.	Clase X/Administrar Juegos/Crear/Aceptar
		V	I	V	V	V		
		V	V	I	V	V		
		V	V	V	I	V		
		V	V	V	V	I		

3.4 Resultados de las pruebas

A continuación se muestran los resultados obtenidos una vez realizadas las diferentes pruebas:

3.4.1 Resultados de las Pruebas de sistema

Con el objetivo de verificar el cumplimiento de los requisitos funcionales establecidos para la presente investigación se hace uso de las **Pruebas de Caja Negra**, teniendo en cuenta la técnica de **partición por equivalencia**. Esta técnica permite examinar los valores válidos e inválidos de las entradas existentes en el *software* y descubrir de forma inmediata una clase de errores. La partición equivalente se basa en la definición de casos de pruebas que descubran errores.

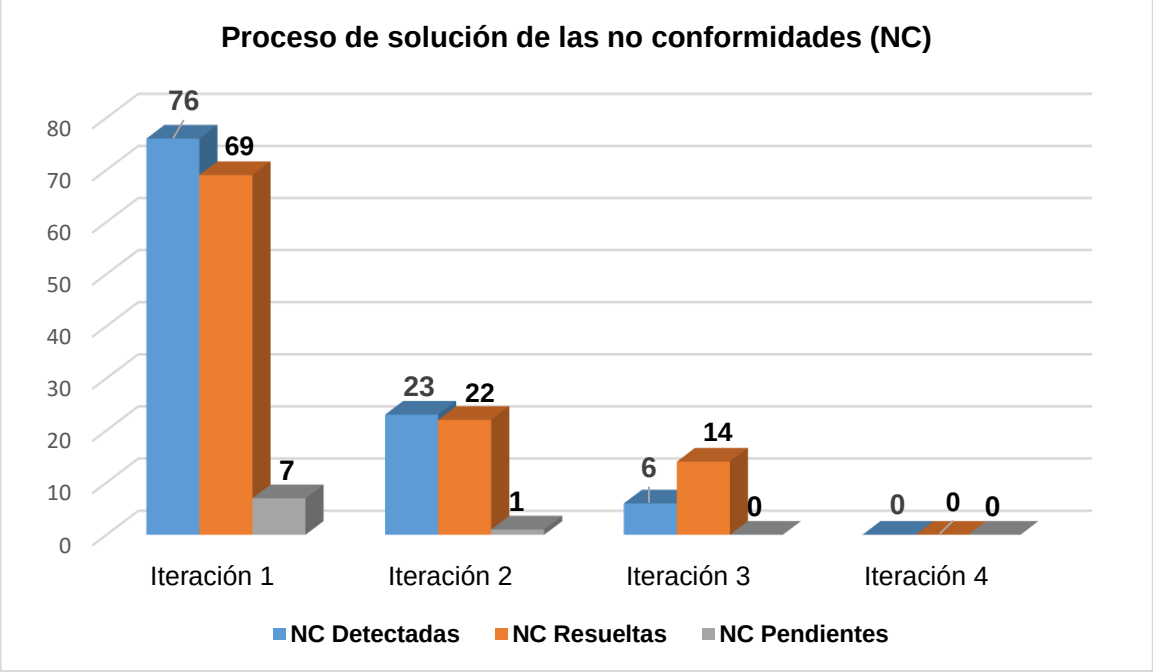
Además, se hace uso de los casos de prueba generados durante este flujo de trabajo con el fin de detectar la mayor cantidad de no conformidades posibles en las funcionalidades del componente realizándose cuatro iteraciones de prueba.

A continuación se muestra una tabla donde se evidencian los resultados obtenidos una vez realizadas las pruebas en cada una de las iteraciones:

Tabla 9 Resultado de las pruebas de sistema

Iteraciones				
	Alta	Media	Baja	Total
1	15	25	36	76
2	7	9	7	23
3	-	2	4	6
4	-	-	-	-

En el siguiente gráfico se ejemplifica el proceso de solución de las no conformidades detectadas, mostrándose por iteración la cantidad de no conformidades identificadas, las resueltas y las que quedaron pendientes:



3.5 Conclusiones del capítulo

Una vez concluido el presente capítulo se arribó a las siguientes conclusiones:

- Los estándares de codificación permitieron una mayor organización durante la implementación del módulo.
- La elaboración del diagrama de componentes permitió presentar la vista de implementación del sistema.
- Las pruebas de *software* aplicadas, permitieron validar el módulo para la creación de juegos.

CONCLUSIONES

Con la culminación de la presente investigación se obtuvo como resultado la implementación de un Módulo para la creación de juegos del tipo observación en el *software* ATcnea. A continuación se muestran las conclusiones generales:

- El análisis realizado a las aulas tecnológicas arrojó como resultado que las mismas no cuentan con módulos de juegos sino que usan distintos juegos web así como disímiles herramientas web.
- La metodología, las tecnologías y herramientas seleccionadas garantizaron el correcto desarrollo del módulo.
- El diseño de la solución facilitó una mejor comprensión del sistema para la posterior implementación.
- La realización de las pruebas al módulo desarrollado, permitió detectar y corregir los errores presentes en el mismo.
- Se obtuvo un módulo para la creación de juegos del tipo observación en el *software* ATcnea.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos en la investigación, para próximas investigaciones se recomienda:

- Incluir nuevos tipos de juegos al módulo que aporten nuevos valores educativos.
- Incluir una herramienta para la selección libre de los objetos permitiendo una selección más óptima de los mismos que proporcione mayor dificultad a los juegos creados.

REFERENCIAS

(FIB), Facultad de Informática de Barcelona. 2008. Historia de los videojuegos. *Retro Informática.* [En línea] 20 de 3 de 2008. [Citado el: 5 de 2 de 2015.]

Blanco, Bueno, Carlos. [2010]. *Ingeniería de Software - Construcción y pruebas.* [En línea] [2010]. [Citado el: 28 de marzo de 2016.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf.

Boisvert, Francine. 2015. *Juegos divertidos en educación primaria, Para desarrollar la observación, la memoria, la reflexión ,el ingenio...* Madrid : Narcea S.A., 2015. ISBN 978-84-277-2080-0.

Camacho. **Camacho, M. Teresa FuertesLa observación de las prácticas educativas como elemento de evaluación y de mejora de la calidad en la formación inicial y continua del profesorado. 2011.** 237-258, Catalunya, España : Revista de Docencia Universitaria, 2011, Vol. 9. ISSN 1887-4592.

Carmona, Juan García. 2012. Solid y GRASP. *Buenas prácticas hacia el éxito en el desarrollo de software.* 2012.

Cheptsov, Andrey. 2016. *IntelliJ IDEA 2017.1 EAP is Open.* 2016.

Cornejo, José Enrique González. 2008. ¿Qué es UML? El Lenguaje de Modelado Unificado. [En línea] Enero de 2008. [Citado el: 6 de 4 de 2017.] http://www.docirs.com/uml.htm.

Díaz, Antonia Lozano. 2004. El aula inteligente: ¿hacia un nuevo paradigma educativo? *Revista Electrónica de Investigación Educativa.* [En línea] 2004. http://redie.uabc.mx/vol6no2/contenido-lozano.html.

DISTANCIA, UNIVERSIDAD NACIONAL ABIERTA Y A. 2016. Lenguaje de modelado unificado UML. [En línea] 2016. http://stadium.unad.edu.co/ovas/10596_9836/index.html.

Dopico, MSc. María Josefina Vidal Ledo Dra. Ileana Morales Suárez MSc. Rosa Moraima Rodríguez. 2014. Scielo. [En línea] 24 de 1 de 2014. http://scieloprueba.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412014000200018&lng=es&nrm=iso.

Extremadura, J DE. 2001. *Sociedad de la información y Educación .* 2001.

Figueroa, Gustavo. 2005. *LA METODOLOGIA DE ELABORACION DE PROYECTOS.* 2005. ISSN 0718-1701.

Fiqus. 2010. Git - Manual de usuarios Version 1 PDF. *Cooperativa de Trabajo Fiqus LTDA.* [En línea] Cooperativa de Trabajo Fiqus LTDA, 2010. http://www.fiqus.com.

Garzás, Javier. 2009. *El Laboratorio Nacional de Calidad del Software de INTECO.* 2009.

Goggi, Lucila Ana Cuccaro. 2012. *Adecuación de la metodología de desarrollo Extreme Programming a proyectos llevados a cabo en la materia Laboratorio III de la Facultad de Ingeniería de la Universidad Austral.* Austral : s.n., 2012.

Gómez Giraldo, y otros. **software., Una ontología para la representación de conceptos de diseño de. 2011.** 3, s.l. : 09, 2011, Avances en Sistemas e Informática, Vol. 8. ISSN.

González, Atilio Bustos. 2005. *Estrategias didácticas para el uso de TiC's en la docencia universitaria presencial. Un manual para los ciudadanos del Ágora.* s.l. : Pontificia Universidad Católica de Valparaíso, 2005.

Grifol, Daniel. 2016. *Metodologías de desarrollo ágil: Scrum.* 2016.

HISTORIA DE LAS COMPUTADORAS. **García, Gerardo Ignacio Hernández. 2011.** México : s.n., 2011.

Historia de los Videojuegos: Empresas. *http://indicelatino.com.* [En línea] [Citado el: 5 de 2 de 2015.] http://indicelatino.com/juegos/historia/empresas/.

JACOBSON, GRADY BOOCH JAMES RUMBAUGH IVAR. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000. ISBN: 9788478290369;.

Jacobson, Ivar Rumbaugh James Booch Grady. 2000. *Lenguaje Unificado de Modelado. Manual de referencia.* Madrid, España : Addison Wesley, 2000. ISBN 84-7829+037-0.

Jiménez, Guillermo González. 2016. *DESARROLLO DEL VIDEO-JUEGO "RECICLA CONMIGO" DE LA COLECCIÓN MUNDOCLICK PARA DISPOSITIVOS MÓVILES CON SISTEMA OPERATIVO ANDROID.* 2016.

Kaisler, Stephen H. 2005. *Software Paradigms.* New Jersey : Hoboken Sons : Inc, 2005. ISBN-0471483478.

Kokori, Proyecto. [2010]. Proyecto Kokori. *http://www.kokori.net/win.php.* [En línea] [2010]. http://www.kokori.net/win.php.

LanSchool. 2000. LanSchool. *LanSchool.* [En línea] Estados Unidos, 2000. http://www.lanschool.com/.

Larman, Craig, Hall, Prentice. 2003. *UML y Patrones.* 2003.

Lou, T. 2016. *Eindhoven University of Technology, A comparison of Android Native App Architecture.* 2016.

Martin, María Mercedes. 2010. *Aulas virtuales, convergencia tecnológica y formación de profesores.* 2010.

Martínez, Edgar Rubén Cosío. 2014. Prezi. [En línea] 14 de 11 de 2014. https://prezi.com/vstabfy31lb7/aprender-jugando-o-jugando-a-aprender-los-videojuegos-en-el-aula/#_=_.

McCaleb, Michael R. 1999. *A Conceptual Data Model of Datum Systems.* National Institute of Standards and Technology : s.n., 1999.

MSc. María Josefina Vidal Ledo, Dra. Ileana Morales Suárez,MSc. Rosa Moraima Rodríguez Dopico. 2014. Scielo. [En línea] 24 de 1 de 2014. http://scieloprueba.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412014000200018&lng=es&nrm=iso.

Olmo, Felipe Segovia. 2003. *El aula inteligente. Nuevas perspectivas.* Madrid : Espasa Calpe, 2003.

ORACLE. Core J2EE Patterns - Front Controller. [En línea] http://www.oracle.com/technetwork/java/frontcontroller-135648.html.

—. 2010. NetBeans. *NetBeans.* [En línea] 2010. http>//netbeans.org/community/releases/67/.

Paredes, Leyre. 2015. Fundación Telefónica España. [En línea] 22 de 10 de 2015. http://www.fundaciontelefonica.com/2015/09/22/aplicaciones-y-videojuegos-utilizar-aula/.

Pérez, Oiver Andrés. 2011. *Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM.* 2011.

Pressman, Roger. 2002. *Ingeniería del Software. Un enfoque práctico. 5ta. Edición.* 2002. ISBN 84-481-3214-9.

Pressman, Roger S. 2010. *Software Engineering: an practitioner's approach 7ma Edition.* New York: Higher Education : s.n., 2010. 978-0-07-337 597 -7.

Pro, Impero Education. 2002. Impero Education Pro. *Impero Education Pro.* [En línea] Reino Unido, 2002. http://www.imperosoftware.co.uk/products/education-pro-classroom-management-software/.

Pupilpad. Pupilpad. *Pupilpad.* [En línea] Jordan. http://www.pupilpad.net.

Ray. 2011. Historia de los Videojuegos: El Origen y los Inicios. *otakufreaks.com.* [En línea] 10 de 10 de 2011. [Citado el: 1 de 2 de 2015.] http://www.otakufreaks.com/historia-de-los-videojuegos-el-origen-y-los-inicios/.

Rodríguez, Tello, Eduardo A. 2012. *Estrategias y técnicas de prueba del software.* [.pdf] CINVESTAV-Tamaulipas : s.n., 2012.

Roselló, Reynaldo Rosado. 2016. Gestor de Documentos Administrativos. [En línea] 2016. [Citado el: 28 de Junio de 2016.] https://excriba.prod.uci.cu/page/context/shared/document-details?nodeRef=workspace://SpacesStore/7f2a24b8-928e-4455-891a-8f2c3d48beb8.

Rumbaugh, J., Jacobson, I. y Booch, G. 2000. El Lenguaje Unificado del Modelado. Manual de Referencia. *UML.* [En línea] 2000. [Citado el: 2 de Febrero de 2016.] https://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf. ISBN: 84-7829-037-0.

School, NetSoporte. NetSoporte School. *NetSoporte School.* [En línea]

Sgoliver. 2014. *Entorno de desarrollo Android (Android Studio).* 2014.

Software, Design Tools for Agile Teams, with UML, BPMN and More. 2016. Software Design Tools for Agile Teams, with UML, BPMN and More. *www.visual-paradigm.com.* [En línea] 2016. [Citado el: 15 de 12 de 2016.] https://www.visual-paradigm.com/.

Software, Ingeniería de. 2011. MODELO DE ANÁLISIS. [En línea] 2011. [Citado el: 6 de 4 de 2017.] https://mundokramer.wordpress.com/2011/05/20/modelo-de-analisis-software/.

Software, Mythware Classroom Management. 2007. Mythware Classroom Management Software. *Mythware Classroom Management Software.* [En línea] China, 2007. http://www.mythware.com.

Sommerville, Ian. 2006. *Software Engineering.* s.l. : 8th ed, 2006. ISBN 0-321-31379-8. .

—. 2005. *Ingeniería de Software 7ma edición.* s.l. : Pearson-Addison Wesley, 2005.

Studio, Meet Android. 2016. Meet Android Studio. *developer.android.com.* [En línea] 2016. [Citado el: 15 de 12 de 2016.] https://developer.android.com/studio/intro/index.html.

Tabares, Fernando Maillane. 2015. PREZI. [En línea] 20 de 4 de 2015. https://prezi.com/avl1wwvvsknx/patrones-de-arquitectura-y-diseno/.

Trastorno del Déficit de atención e hiperactividad. **Benítez, Manuel y Hierro, M. D. 2002.** 15-22, s.l. : Programa de Educación en Pediatría, 2002, Vol. 5.

Tutorials, The Java™. 2015. The Java™ Tutorials. *http://docs.oracle.com.* [En línea] 2015. [Citado el: 15 de 12 de 2016.] http://docs.oracle.com/javase/tutorial/getStarted/TOC.html.

UCI, Programa de Mejora Universidad de las Ciencias Informáticas, Basado en Convenciones de Código para el lenguaje de programación JAVA™ por Scott Hommel Sun Microsystems Inc. Traducción por Alberto Molpeceres. 2016. *Estándar de codificación para Java.* 2016.

UML, Modelado de Sistemas com. 2016. Modelado de Sistemas com UML. [En línea] 2016. https://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x208.html.

Universidad de las Ciencias Informáticas. 2014. Estrategia Marcaria de los Productos UCI. [En línea] UCI, 2014. [Citado el: 26 de 6 de 2017.] http://iux.prod.uci.cu/.

Universidad Unión Bolivariana. 2014. METODOLOGIAS AGILES “PROCESO UNIFICADO AGIL (AUP)”. [En línea] 2014. [Citado el: 2016 de Enero de 26.] http://ingenieriadesoftware.mex.tl/images/18149/METODOLOGIAS%20AGILES.pdf.

Welicki, León. 2016. *Patrones y Antipatrones: una Introducción - Parte II.* [En línea] 2016. [Citado el: 1 de Marzo de 2016.] https://msdn.microsoft.com/es-es/library/bb972251.aspx.

Wilhelm, Richard. 2009. ASTM International. *Método de prueba, práctica ¿u otra cosa?* [En línea] Octubre de 2009. [Citado el: 20 de Marzo de 2016.] http://www.astm.org/SNEWS/SPANISH/SPSO09/ruls_regs_spso09.html.

ANEXOS

Anexo 1 Historia de usuario de los requisitos funcionales restantes

Tabla 10 HU requisito funcional editar juego

Número: 2	**Nombre del requisito:** Editar juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:**25días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir la edición de un juego en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para la edición del juego:

- El usuario debe estar autenticado como profesor.
- Existir al menos un juego en el sistema.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor editar un juego creado con anterioridad accediendo a la opción Editar una vez seleccionado e juego a modificar.

Una vez seleccionada la opción el sistema debe permitir modificar los siguientes datos:

- Tipo de juego
- Tiempo de juego
- Cantidad de pistas

- Cargar imagen

Una vez cargada la imagen el sistema debe permitir realizar las siguientes acciones:

- Seleccionar área
- Limpiar pantalla

Además debe permitir realizar una de las siguientes acciones:

- Aceptar
- Cancelar

Al seleccionar la opción Aceptar el sistema crea y guarda el juego.

Al seleccionar la opción Cancelar el sistema cierra la ventana de crear.

En caso de que haya campos vacíos o campos incorrectos se debe mostrar un mensaje de información.

Observaciones: La edición de los juegos permite a los profesores mejorar los juegos ya creados.

Prototipo de interfaz:

Tabla 11 HU requisito funcional eliminar juego

Número: 3	**Nombre del requisito:** Eliminar juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 10días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir la eliminación de un juego en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para la eliminación de un juego tiene que:

- El usuario estar autenticado como profesor.
- Existir al menos un juego en el sistema.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor eliminar un juego creado con anterioridad seleccionando del listado de juegos el juego a eliminar y luego la opción Eliminar.

Una vez seleccionada la opción el sistema debe pedir confirmación para realizar la acción, además debe permitir al usuario seleccionar una de las siguientes opciones: Aceptar o Cancelar.

- Si el profesor selecciona la opción Cancelar el sistema no elimina el juego.
- Si el profesor selecciona la opción Aceptar el sistema elimina el juego seleccionado y actualiza el listado de juegos.

Observaciones: La eliminación de los juegos permite al profesor tener mayor capacidad para actualizaciones del mismo juego o para juegos mejores.

Prototipo de interfaz:

Tabla 12 HU requisito funcional cargar imagen

Número: 4	**Nombre del requisito:** Cargar imagen
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:**10 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir cargar imágenes en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para cargar las imágenes tiene que: - El usuario estar autenticado como profesor. - El profesor estar creando un juego.	

3- Flujo de la acción a realizar:

El sistema debe permitir profesor cargar una imagen para crear un juego seleccionando la opción Examinar, una vez seleccionada la opción el sistema debe permitir seleccionar una imagen.

Además permite seleccionar una de las siguientes opciones: Aceptar o Cancelar.

- Al seleccionar la opción Cancelar el sistema cierra el formulario que permite seleccionar la imagen.

- Al seleccionar la opción Aceptar el sistema debe evaluar si la dirección es válida y si el archivo seleccionado es una imagen. En caso de que el archivo seleccionado no sea una imagen o sea una dirección incorrecta el sistema debe lanzar un mensaje de información.

Observaciones: Las imágenes son para la creación de los juegos o para la modificación del diseño de los juegos.

Prototipo de interfaz:

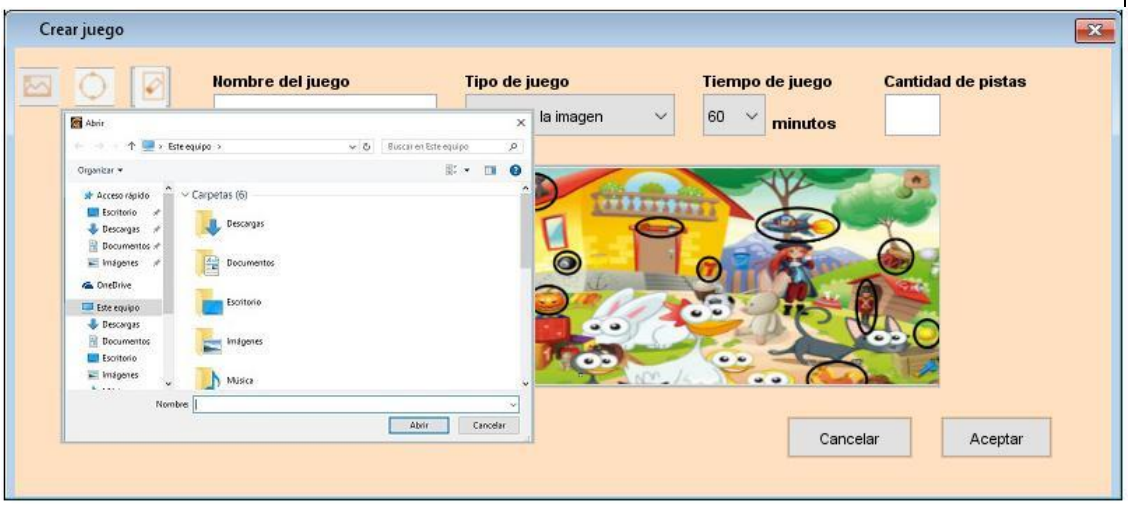


Tabla 13 HU requisito funcional seleccionar área

Número: 5	**Nombre del requisito:** Seleccionar área
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:**15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir al profesor seleccionar un área determinada en el juego.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para seleccionar un área:

- El usuario debe estar autenticado como profesor.

-Existir al menos una imagen en el sistema en la cual el profesor pueda hacer la selección de un área.

3- Flujo de la acción a realizar:

El sistema debe permitir tanto al profesor como a los estudiantes seleccionar un área en la imagen, para ello los usuarios deben estar autenticados en el sistema como profesor o como estudiante, cada uno en su aplicación.

En caso de ser profesor se debe haber cargado previamente una imagen para crear el juego y en caso de ser estudiante se debe estar ejecutando el juego en el terminal.

El sistema debe permitir al profesor escoger la opción Seleccionar área,

una vez seleccionada esta opción el sistema debe permitir al profesor seleccionar en la imagen el área deseada arrastrando el mouse desde un punto a otro lo que se representará a través de un círculo. Luego esta área es guardada en el juego como una selección correcta. En caso del estudiante una vez que el juego ha sido ejecutado en su terminal el sistema debe permitir que el mismo al seleccionar (hacer clic) en cualquier sitio de la imagen, si es un área seleccionada por el profesor en la creación del juego la misma se señalará a través de un círculo.
Observaciones: Las selecciones de áreas son para la creación de los juegos y la edición por parte del profesor y para la respuesta de los estudiantes cuando estén jugando.
Prototipo de interfaz: 

Tabla 14 HU requisito funcional guardar fichero de juego

Número: 6	**Nombre del requisito:** Guardar fichero de juego
Programador: Luis Manuel	**Iteración Asignada:** 2da

Lugo y Lennon Acosta	
Prioridad: Alta	**Tiempo Estimado:**20 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir al profesor guardar el fichero del juego creado en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para guardar el fichero de un juego:

- El usuario debe estar autenticado como profesor.
- Existir al menos un juego en el sistema creado por el profesor.
- Existir conexión con el servidor para guardar el fichero.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor guardar el fichero del juego creado, para ello debe estar autenticado en el sistema y debe estarse creando un juego.

Una vez que se está creando el juego el sistema permite seleccionar la opción Aceptar.

Una vez seleccionada la opción Aceptar el sistema guarda el juego creado con las especificaciones realizadas por el profesor y actualiza el listado de juegos.

Observaciones: Las ficheros de juego son usados por los profesores para la creación de nuevos juegos o la modificación de los mismos. También para el envío de los juegos a los estudiantes.

Prototipo de interfaz:

Tabla 15 HU requisito funcional mostrar juego

Número: 7	**Nombre del requisito:** Mostrar juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:**15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir visualizar un juego en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar el juego: - El usuario debe estar autenticado como estudiante. -Existir al menos un juego en el sistema. **3- Flujo de la acción a realizar:** El sistema debe permitir al estudiante visualizar el juego enviado por el	

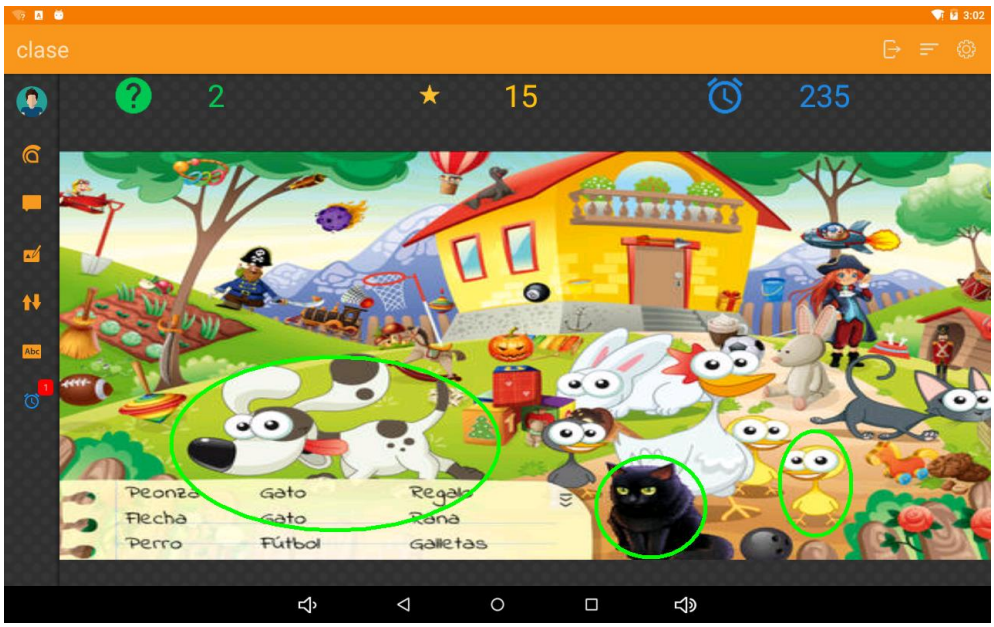
profesor accediendo a la opción que le permite visualizar el juego, el sistema debe mostrar el juego enviado y además los siguientes datos: • Cantidad de pistas • Tiempo del juego • Puntuación El sistema debe permitir al estudiante interactuar con el juego que está visualizando.
Observaciones: La muestra de los juegos es para que el profesor pueda editarlo y para que el estudiante pueda jugarlo.
Prototipo de interfaz: 

Tabla 16 HU requisito funcional redimensionar imagen

Número: 8	**Nombre del requisito:** Redimensionar imagen
Programador: Luis Manuel	**Iteración Asignada:** 1ra

Lugo y Lennon Acosta	
Prioridad: Alta	**Tiempo Estimado:**20días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir la redimensión de una imagen en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para la redimensión de una imagen:

- El usuario debe estar autenticado como profesor.
- Existir al menos una imagen en el sistema o realizar la carga de la imagen.

3- Flujo de la acción a realizar:

Una vez cargada la imagen para crear un juego el sistema debe redimensionar la misma de forma automática.

Observaciones: La redimensión de las imágenes permiten la mejor edición de los juego y del diseño de los mismos.

Prototipo de interfaz:

Tabla 17 HU requisito funcional enviar juego a la aplicación cliente

Número: 9	**Nombre del requisito:** Enviar juego a la aplicación cliente
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:**15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir el envío de juegos a la aplicación cliente en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para el envío de juegos a la aplicación cliente:

- El usuario debe estar autenticado como profesor.

-Existir al menos un juego en el sistema.

-Existir conexión con entre la aplicación profesor y la aplicación cliente.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor enviar un juego creado con anterioridad seleccionando para ello el juego a enviar y luego la opción Enviar.

Una vez seleccionada la opción Enviar el sistema muestra el listado de estudiantes conectados a la clase donde el profesor deberá seleccionar a los estudiantes a los que desea enviar el juego y posteriormente la opción Enviar.

Observaciones: El envío de los juegos a la aplicación cliente permite al profesor enviarle a un estudiante o a muchos el juego de imágenes con el que los va a evaluar posteriormente.

Prototipo de interfaz:

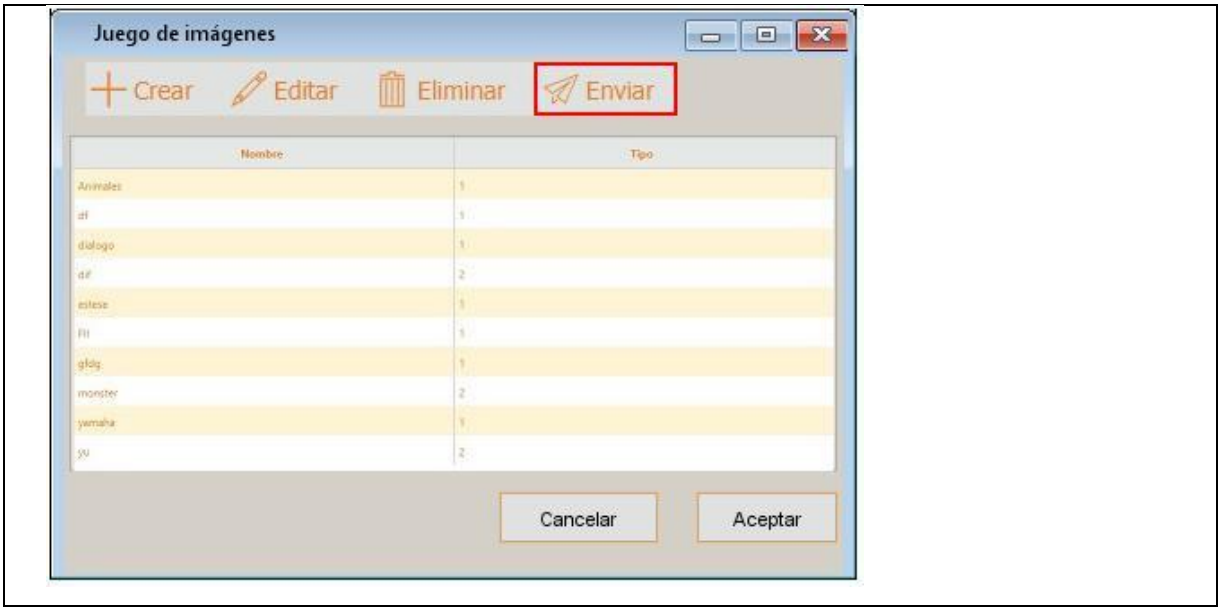


Tabla 18 HU requisito funcional seleccionar juego a enviar

Número: 10	**Nombre del requisito:** Seleccionar juego a enviar
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:**1 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir la selección de un juego en el sistema para su posterior envío. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para seleccionar un juego: - El usuario debe estar autenticado como profesor. **3- Flujo de la acción a realizar:**	

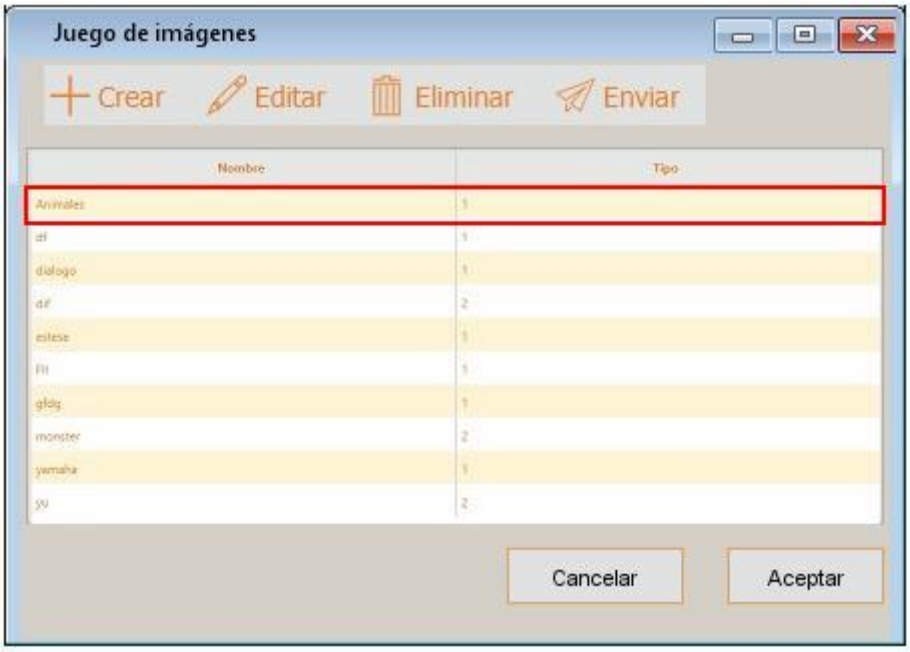
El sistema debe permitir al profesor seleccionar del listado de juegos el juego que desea enviar a los estudiantes haciendo clic encima del mismo. El juego seleccionado se marcará de otro color.
Observaciones: La selección del juego permite al profesor elegir entre los juego que desea enviar.
Prototipo de interfaz: 

Tabla 19 HU requisito funcional listar juegos

Número: 11	**Nombre del requisito:** Listar juegos
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**10días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir listar juegos en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para listar juegos:

- El usuario debe estar autenticado como profesor.
- Existir al menos un juego en el sistema.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor visualizar el listado de juegos creados en el sistema accediendo a la opción Administrar juego.

Una vez seleccionada la opción el sistema muestra el listado de juegos creados con anterioridad y permite realizar una de las siguientes acciones:

- Crear
- Editar
- Eliminar
- Enviar

El sistema permite además seleccionar una de las siguientes opciones:

- Cancelar
- Resultados

Observaciones: Listar los juegos permite la organización de los mismos en el sistema para una mejor utilización de los mismos.

Prototipo de interfaz:

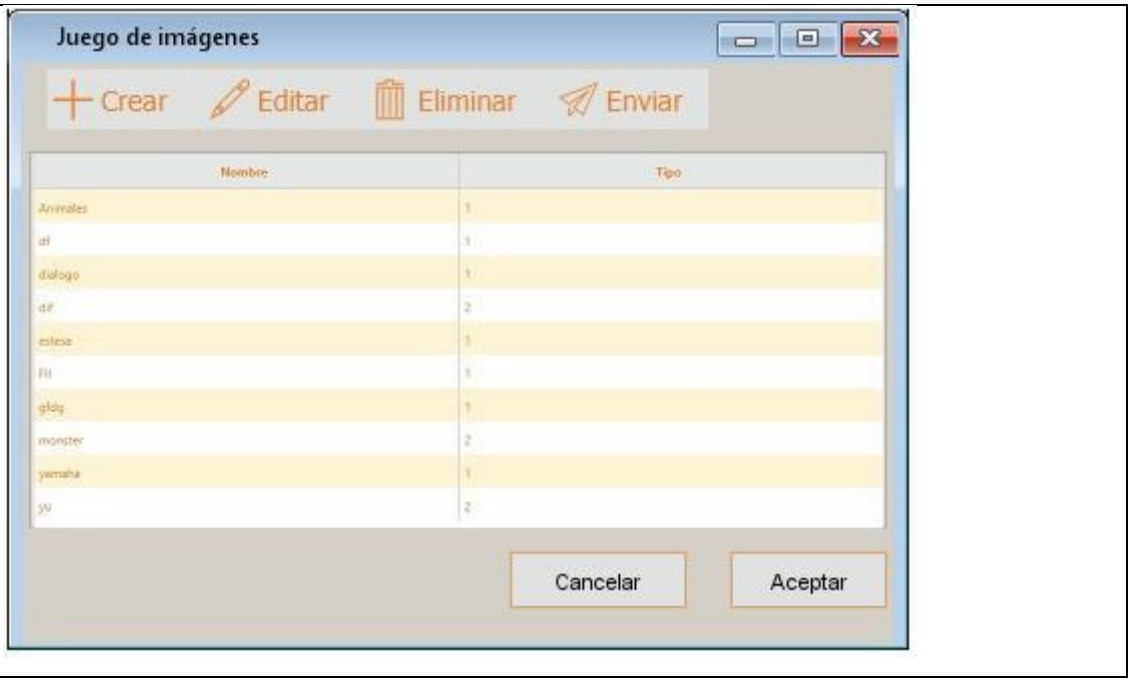


Tabla 20 HU requisito funcional mostrar pistas

Número: 12	**Nombre del requisito:** Mostrar pistas
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir visualizar las pistas en el juego. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para mostrar pistas tiene que: - El usuario debe estar como estudiante.	

- Existir al menos un juego en el sistema.
- El juego debe tener la opción de pistas.

3- Flujo de la acción a realizar:

Una vez que el estudiante se encuentre interactuando con el juego y acceda a la opción Pistas el sistema debe mostrar las pistas especificadas por el profesor al crear el juego.

El número de pistas especificadas por el profesor debe ir disminuyendo a medida que el estudiante acceda a la opción Pistas, así mismo al estudiante consultar una pista no recibirá evaluación por el objeto mostrado.

Observaciones: Las pistas ayudan a los estudiantes menos aventajados a alcanzar la solución del juego.

Prototipo de interfaz:

Tabla 21 HU requisito funcional incluir nombre de la selección

Número: 13	**Nombre del requisito:** Incluir nombre de la selección
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**5días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir nombrar la selección de un área en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para nombrar una selección:

- El usuario debe estar autenticado como profesor.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor una vez seleccionada el área a identificar, especificar el siguiente dato:

- Nombre de la selección

El nombre especificado por el profesor representará el objeto a buscar y formará parte del listado de objetos a encontrar.

Observaciones: Nombrar la selección le permitirá al profesor saber lo que crea y al estudiante saber lo que busca.

Prototipo de interfaz:

Tabla 22 HU requisito funcional listar objetos a encontrar

Número: 14	**Nombre del requisito:** Listar objetos a encontrar
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**12días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir listar los objetos a encontrar en un juego en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar los objetos en el juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema. - El juego incluir un grupo de objetos. **3- Flujo de la acción a realizar:**	

El sistema debe mostrar en el juego enviado a los estudiantes un listado con los objetos a encontrar, los mismos deben haber sido especificados por el profesor una vez que ha seleccionado el área a buscar.
Observaciones: El listado de los objetos a encontrar en el juego permite a los jugadores una mejor organización a la hora de jugarlos.
Prototipo de interfaz: 

Tabla 23 HU requisito funcional mostrar puntuación

Número: 15	**Nombre del requisito:** Mostrar puntuación
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**10días

Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir visualizar la puntuación obtenida en un juego en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para mostrar la puntuación en un juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema. **3- Flujo de la acción a realizar:** El sistema debe mostrar al estudiante a medida que el mismo interactúe con el juego la puntuación que se va alcanzando una vez que identifique los objetos o las diferencias en las imágenes. La puntuación se irá incrementando a medida que el estudiante identifique las imágenes.	
Observaciones: La muestra de la puntuación en el juego permite al jugador autoevaluarse.	
Prototipo de interfaz:	

Tabla 24 HU requisito funcional mostrar objeto

Número: 16	**Nombre del requisito:** Mostrar objeto
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:**20 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al estudiante visualizar un objeto encontrado en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para mostrar un objeto en un juego:	

- El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema creado por el profesor. **3- Flujo de la acción a realizar:** El sistema debe mostrar una vez que el estudiante seleccione un objeto en la imagen que dicho objeto se muestre, es decir, se señale como objeto encontrado y lo borre de la lista de objetos a encontrar.
Observaciones: Permite a los estudiantes enfocarse en otro objeto una vez encontrado el anterior.
Prototipo de interfaz: 

Tabla 25 HU requisito funcional mostrar tiempo

Número: 17	**Nombre del requisito:** Mostrar tiempo

Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir al estudiante visualizar el tiempo en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para mostrar el tiempo de un juego:

- El usuario debe estar autenticado como estudiante.
- Existir al menos un juego en el sistema creado por el profesor.

3- Flujo de la acción a realizar:

El sistema debe mostrar el tiempo de forma automática en la parte superior derecha una vez que el estudiante ejecute el juego.

El tiempo disminuirá y se detendría en caso de que el estudiante gane o en el momento en que llegue a 0, mostrando en ambas opciones un mensaje de información.

Observaciones: Permite a los estudiantes dividirse en tareas de la fácil a la difícil ya que están contra el reloj.

Prototipo de interfaz:

Tabla 26 HU requisito funcional buscar juego en la lista de juegos

Número: 18	**Nombre del requisito:** Buscar juego en la lista de juegos
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al profesor encontrar un juego determinado rápidamente. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

Para buscar un juego tiene que:

- El usuario debe estar autenticado como profesor.
- Existir al menos un juego en el sistema creado por el profesor.

3- Flujo de la acción a realizar:

El sistema debe permitir buscar un juego en el listado de juegos especificando el siguiente dato:

- Nombre del juego

El sistema debe mostrar el juego o los juegos relacionados con el nombre introducido.

De no existir ninguno mostrar un mensaje de información.

Observaciones: Permite al profesor una mejor navegación y más rápida por los juegos que posea el *software*.

Prototipo de interfaz:

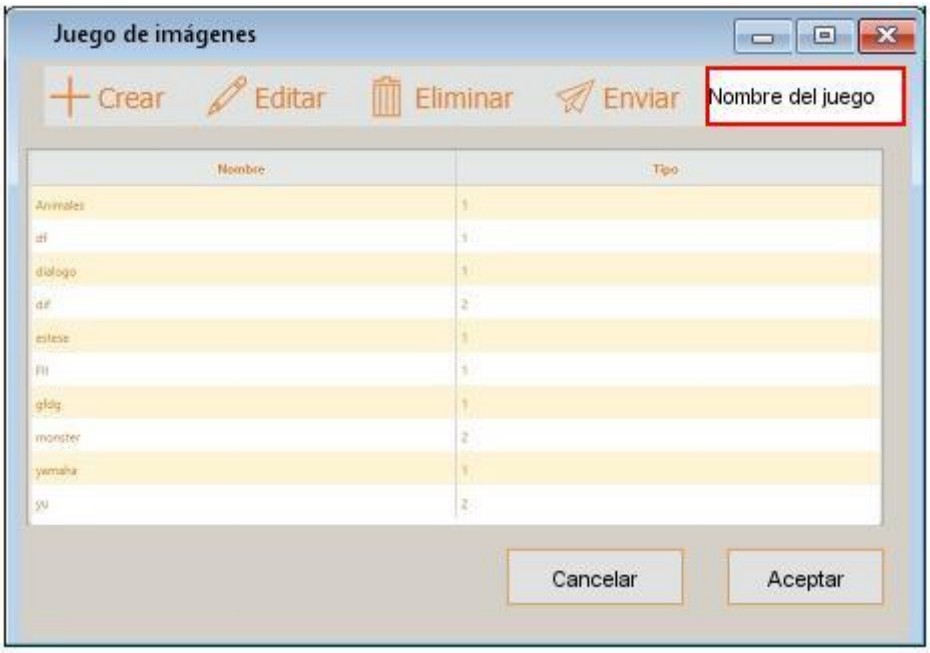


Tabla 27 HU requisito funcional listar estudiantes

Número: 19	**Nombre del requisito:** Listar estudiantes
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:**2 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir listar estudiantes en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para listar estudiantes: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema. **3- Flujo de la acción a realizar:** El sistema debe permitir al profesor visualizar el listado de estudiantes conectados a la clase a los cuales les enviará el juego. Luego de elegir la opción enviar juego y seleccionar el juego a enviar se muestra el listado de estudiantes conectados a la clase.	
Observaciones: Listar los estudiantes permite la organización de los mismos en el sistema para un mejor control de ellos.	
Prototipo de interfaz:	

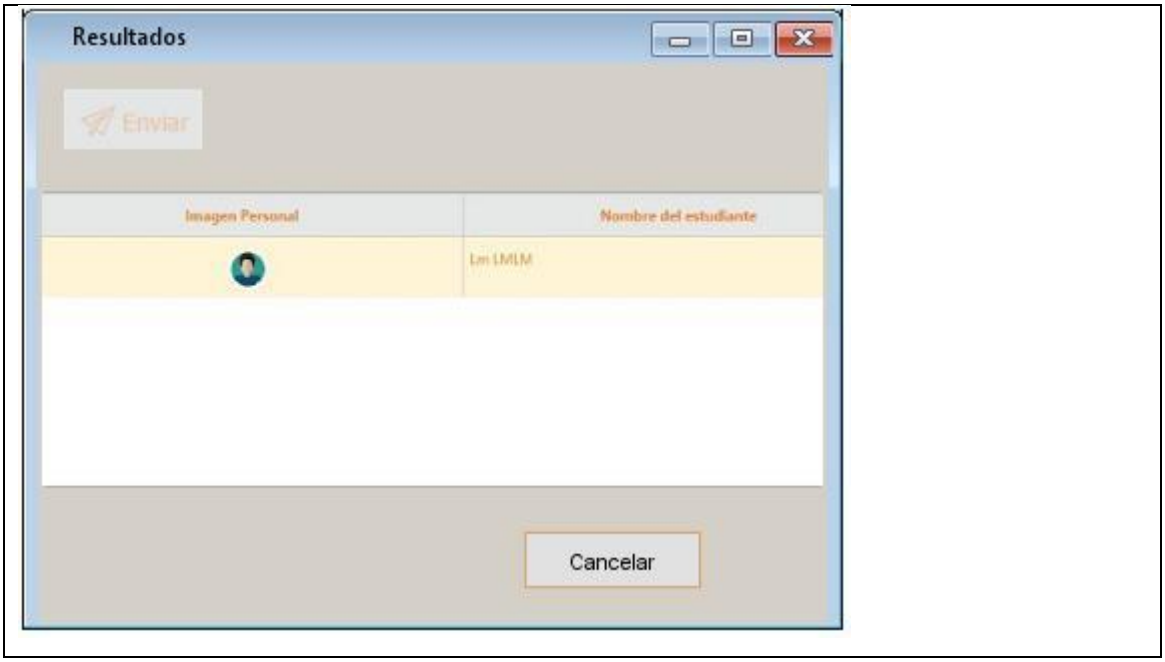


Tabla 28 HU requisito funcional mostrar resultados del juego

Número: 20	**Nombre del requisito:** Mostrar resultados del juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir visualizar los resultados de todos los estudiantes en un juego en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar los resultados del juego:	

- El usuario debe estar autenticado como profesor.
- Existir al menos un juego en el sistema.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor visualizar el resultado obtenido por los estudiantes en un juego accediendo a la opción Resultados el cual muestra un listado con los resultados obtenidos.

En el listado se mostrarán los siguientes datos:

- Recibido de
- Tipo de juego
- Nombre de juego
- Evaluación

El sistema debe permitir al profesor seleccionar la opción Cancelar.

Observaciones: La muestra de los resultados de los juegos es para que el profesor pueda visualizarlos y tomar notas.

Prototipo de interfaz:

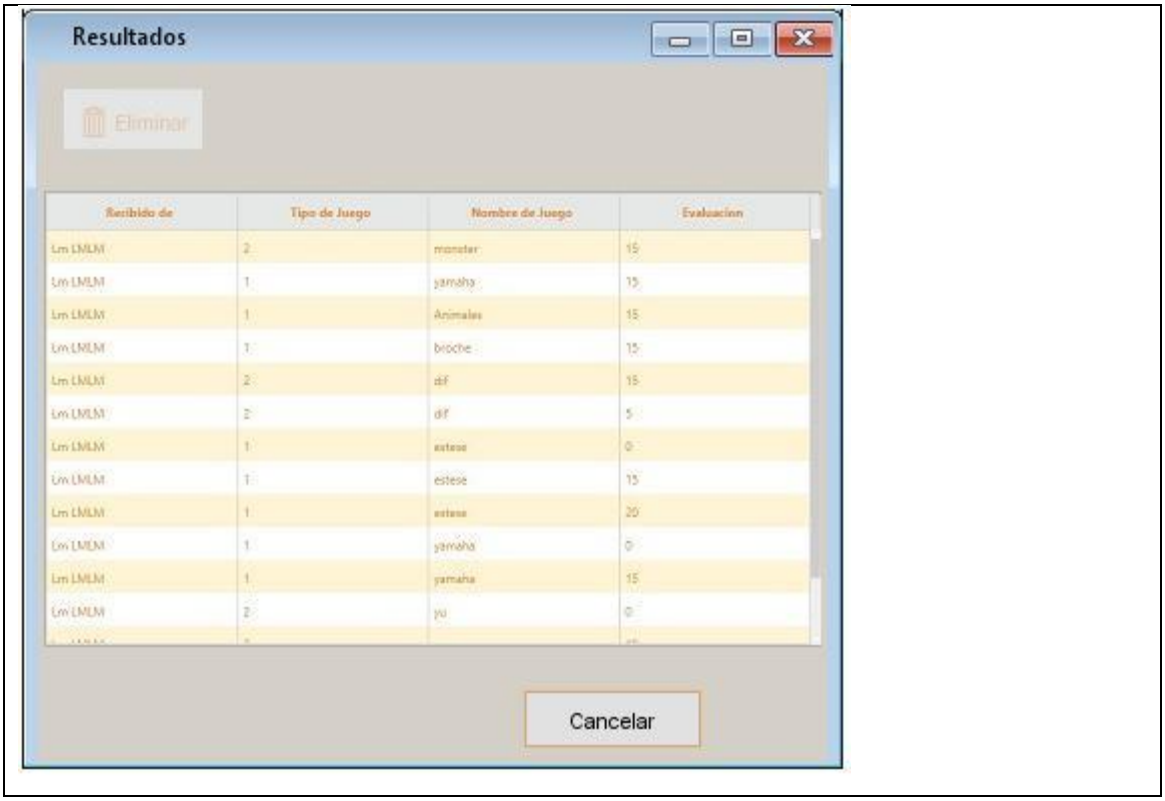


Tabla 29 HU requisito funcional eliminar los resultados

Número: 21	**Nombre del requisito:** Eliminar los resultados
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:**5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permite eliminar los resultados de los estudiantes del sistema.	

2- Acciones para lograr el objetivo (precondiciones y datos):

Para la selección tiene que:

- El usuario estar autenticado como profesor.

3- Flujo de la acción a realizar:

El sistema debe permitir al profesor eliminar los resultados obtenidos del listado de resultados seleccionando el resultado a eliminar y la opción Eliminar. El sistema debe pedir confirmación de la acción.

El sistema debe permitir seleccionar una de las siguientes opciones:

- Aceptar
- Cancelar

Observaciones: Permite al profesor eliminar los resultados que se muestran en la lista de resultados del sistema.

Prototipo de interfaz:

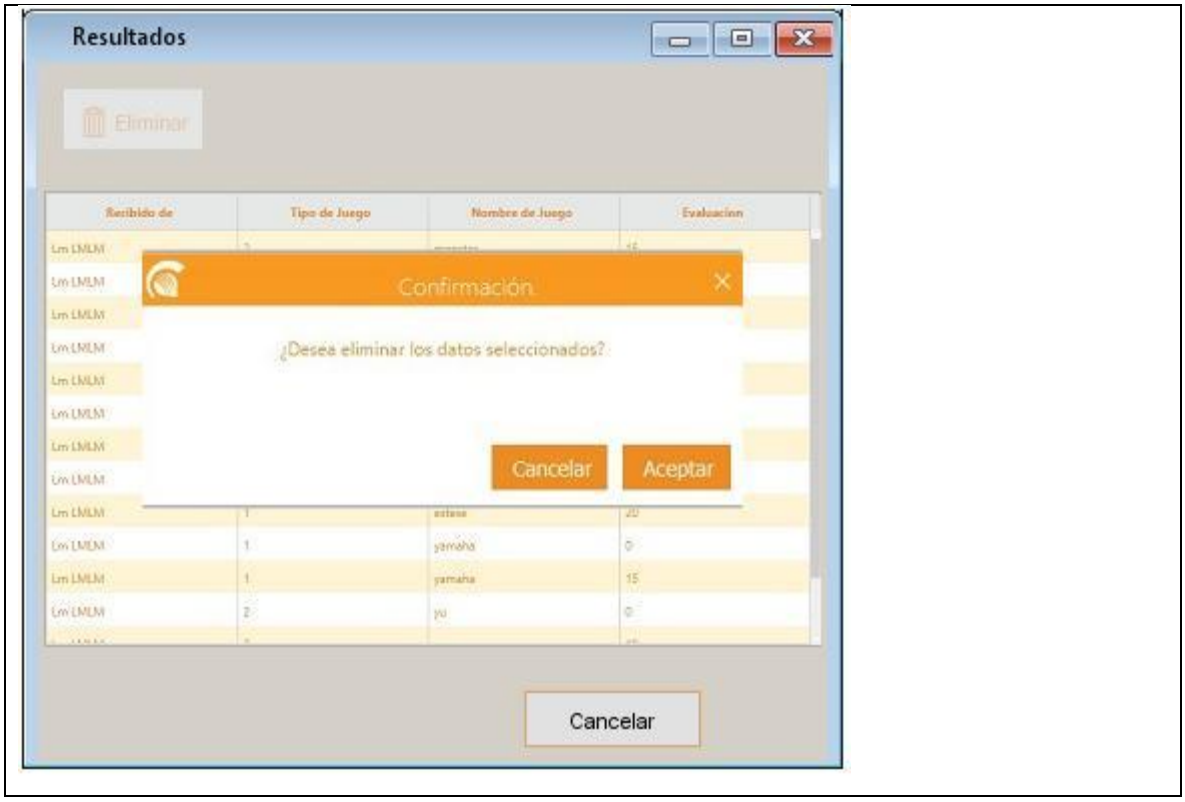


Tabla 30 HU requisito funcional notificar recibo de resultados

Número: 22	**Nombre del requisito:** Notificar recibo de resultados
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al profesor ver la notificación. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

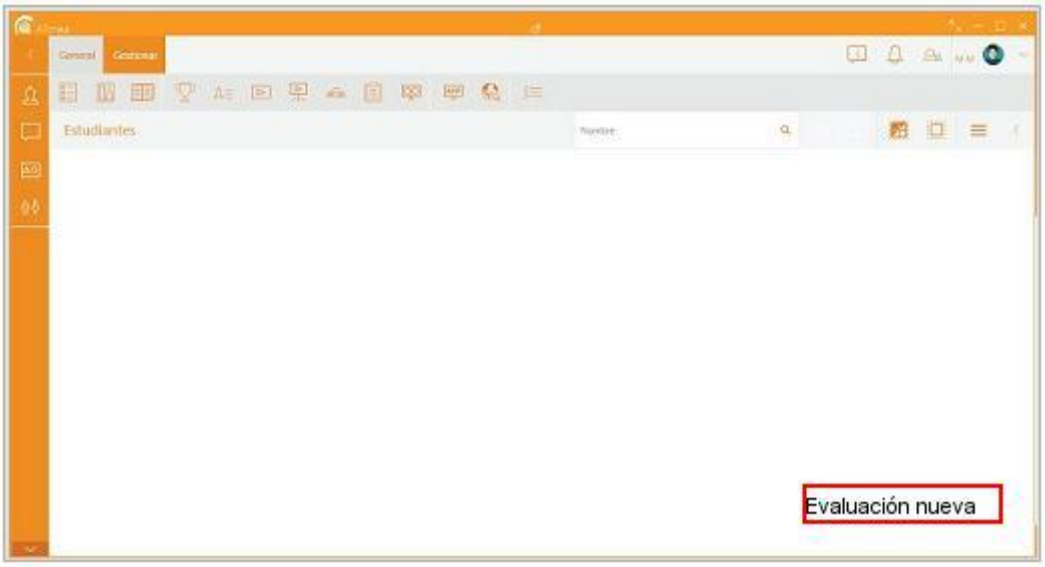
Para notificar al profesor el recibo de un resultado: - El usuario debe estar autenticado como profesor. **3- Flujo de la acción a realizar:** El sistema debe notificar al profesor al recibir los resultados de los estudiantes.
Observaciones: Permite al profesor revisar al instante las respuestas de los estudiantes.
Prototipo de interfaz: 

Tabla 31 HU requisito funcional notificar recibo de juego

Número: 23	**Nombre del requisito:** Notificar recibo de juego
Programador: Luis Manuel Lugo y Lennon Acosta	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:**5 días

Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permite notificar al estudiante. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para notificar al estudiante tiene que: - El usuario estar autenticado como estudiante en su aplicación. - El profesor haber enviado un juego. **3- Flujo de la acción a realizar:** El sistema debe notificar a los estudiantes una vez que el profesor envíe un juego.	
Observaciones: Permite al estudiante ver lo enviado por el profesor.	
Prototipo de interfaz:	


clase
2
15
235
Peonza
Gato
Regalo
Flecha
Gato
Rana
Perro
Fútbol
Galletas

Anexo 2 Casos de Prueba de los requisitos funcionales restantes

Tabla 32 Caso de prueba del requisito editar juego

Descripción general								
Permitir la edición de un juego en el sistema.								
Condiciones de ejecución								
Para la edición del juego: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema.								
Escenario	**Descripción**	**Cargar Imagen**	**Tiempo de juego**	**Seleccionar área**	**Cantidad de pistas**	**Tipo de juego**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Editar juego	Una vez en la interfaz Administrar juego y de haber seleccionado el juego a editar el usuario selecciona Editar juego	N/A	N/A	N/A	N/A	N/A	El sistema debe permitir especificar los siguientes datos: Imagen de fondo, Áreas a seleccionar, Objetos a encontrar Tiempo de juego, Pistas. El sistema permite además seleccionar una de las siguientes acciones: Aceptar y Cancelar.	Clase X/Administrar juego/ Editar juego
EC 1.2 Opción Aceptar	Una vez especificados los datos para editar un juego el usuario selecciona la opción Aceptar.	V	V	V	V	V	El sistema edita el juego. Además actualiza y muestra el listado de los juegos, en el mismo se mostrará el Nombre y el Tipo de juego.	Clase X/Administrar Juegos/Editar juego/ Aceptar
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	N/A	N/A	N/A	N/A	N/A	El sistema muestra la interfaz Administrar juegos.	Clase X/Administrar juegos

								/Editar juego/ Cancelar
EC 1.5 Datos incorrectos	Al especificar los datos para editar un juego el usuario especifica datos incorrectos.	I	V	V	V	V	El sistema muestra un mensaje de información.	Clase X/Administrar Juegos/Editar juego/ Aceptar
		V	I	V	V	V		
		V	V	I	V	V		
		V	V	V	I	V		
		V	V	V	V	I		

Tabla 33 Caso de prueba del requisito Eliminar Juego

Descripción general			
Permitir la eliminación de un juego en el sistema.			
Condiciones de ejecución			
Para la eliminación de un juego tiene que: - El usuario estar autenticado como profesor. - Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Eliminar	Una vez seleccionado el o los juegos a eliminar el usuario selecciona la opción Eliminar.	El sistema debe pedir confirmación de eliminación antes de realizar la acción a través de un mensaje. El sistema debe permitir al usuario seleccionar una de las siguientes opciones para confirmar o cancelar la acción: Aceptar Cancelar	Clase X/Administrar juego/Eliminar
EC 1.2 Opción Aceptar	Una vez que el sistema muestra el mensaje de confirmación el usuario selecciona la opción Aceptar.	El sistema elimina él o los juego seleccionados y actualiza el listado de juegos.	Clase X/Administrar juego/Eliminar/Aceptar
EC 1.3 Opción Cancelar	Una vez que el sistema muestra el mensaje de confirmación el usuario	El sistema muestra nuevamente el listado de juegos.	Clase X/Administrar juego/Eliminar/Cancelar

	selecciona la opción Cancelar.		

Tabla 34 Caso de prueba del requisito cargar imagen

Descripción general				
Permitir cargar imágenes en el sistema.				
Condiciones de ejecución				
Para cargar las imágenes tiene que: - El usuario estar autenticado como profesor. - El profesor estar creando un juego.				
Escenario	**Descripción**	**Cargar Imagen**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Examinar	Una vez en la vista Crear juego el usuario selecciona la opción Examinar.	N/A	El sistema debe permitir buscar el siguiente dato: (*) Imagen, con cualquier formato. El sistema permite además seleccionar una de las siguientes acciones: Aceptar y Cancelar.	Clase X/Administrar juego/Crear juego/Examinar
EC 1.2 Opción Aceptar	Una vez especificados la imagen el usuario selecciona la opción Aceptar.	V	El sistema realiza la carga de la imagen que será utilizada para el juego.	Clase X/Administrar juego/Crear juego/Examinar/Aceptar
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	N/A	El sistema muestra la interfaz anterior.	Clase X/Administrar juego/Crear juego/Examinar/Cancelar

EC 1.5 Campos vacíos	Al especificar los datos para la carga de la imagen el usuario deja campos vacíos.	V I	El sistema muestra un mensaje de información.	Clase X/Administrar juego/Crear juego/Examinar/Aceptar

Tabla 35 Caso de prueba del requisito seleccionar área

Descripción general				
Permitir al profesor seleccionar un área determinada en el juego.				
Condiciones de ejecución				
Para seleccionar un área: - El usuario debe estar autenticado como profesor. -Existir al menos una imagen en el sistema en la cual el profesor pueda hacer la selección de un área.				
Escenario	**Descripción**	**Área**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Seleccionar Área	Una vez seleccionada la opción Crear juego el profesor procede a Seleccionar las áreas.	N/A	El sistema debe permitir editar el siguiente dato: (*) Área. El sistema realiza la selección del área especificada.	Clase X/Administrar grupo/Crear juego/Seleccionar área
EC 1.5 Datos incorrectos	Al especificar los datos para seleccionar el área el profesor especifica datos incorrectos.	I V	El sistema muestra un mensaje de información.	Clase X/Administrar grupo/Crear juego/Seleccionar área

Tabla 36 Caso de prueba del requisito guardar fichero de juego

Descripción general			
Permitir al profesor guardar el fichero del juego creado en el sistema.			
Condiciones de ejecución			
Para guardar el fichero de un juego: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema creado por el profesor. - Existir conexión con el servidor para guardar el fichero.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Seleccionar Área	Una vez seleccionada la opción Crear juego el profesor procede a Seleccionar las áreas.	El sistema debe permitir editar el siguiente dato: (*) Área. El sistema realiza la selección del área especificada.	Clase X/Administrar grupo/Crear juego/Seleccionar área

Tabla 37 Caso de prueba del requisito mostrar juego

Descripción general			
Permitir visualizar un juego en el sistema.			
Condiciones de ejecución			
Para visualizar el juego: - El usuario debe estar autenticado como estudiante. -Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar juego	Una vez seleccionado el juego se selecciona la opción Mostrar juego.	El sistema muestra el juego en el Tablet del estudiante.	Clase X/Juegos/Mostrar juego

Tabla 38 Caso de prueba del requisito redimensionar imagen

Descripción general			
Permitir la redimensión de una imagen en el sistema.			
Condiciones de ejecución			
Para la redimensión de una imagen: - El usuario debe estar autenticado como profesor. - Existir al menos una imagen en el sistema o realizar la carga de la imagen.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Cargar imagen	Una vez seleccionadla opción Cargar imagen y la imagen es cargada.	El sistema realiza la redimensión automáticamente.	Clase X/Crear juego/Cargar imagen

Tabla 39 Caso de prueba del requisito enviar juego a la aplicación cliente

Descripción general			
Permitir el envío de juegos a la aplicación cliente en el sistema.			
Condiciones de ejecución			
Para el envío de juegos a la aplicación cliente: - El usuario debe estar autenticado como profesor. -Existir al menos un juego en el sistema. -Existir conexión con entre la aplicación profesor y la aplicación cliente.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Enviar	Una vez que el usuario selecciona el juego pulsa la opción Enviar y selecciona él o los estudiantes que están conectados y le envía el juego.	El sistema muestra un mensaje donde se especifica que el juego va a ser enviado. EL sistema permite elegir entre las opciones: Aceptar, Cancelar	Clase X/Administrar juegos/Seleccionar juego/Enviar juego/Seleccionar estudiante/Enviar
EC 1.2 Aceptar	Una vez el usuario ha dado Enviar selecciona la opción Aceptar.	El sistema envía el juego a la aplicación cliente y muestra un mensaje de información.	Clase X/Administrar juegos/Seleccionar juego/Enviar juego/Seleccionar estudiante/Enviar/Aceptar
EC 1.3 Cancelar	Una vez el usuario ha dado Enviar selecciona la opción Cancelar.	El sistema no envía el juego y refresca la vista Administrar juegos.	Clase X/Administrar juegos/Seleccionar juego/Enviar juego/Seleccionar estudiante/Enviar/Cancelar

Tabla 40 Caso de prueba del requisito seleccionar juego a enviar

Descripción general			
Permitir la selección de un juego en el sistema para su posterior envío.			
Condiciones de ejecución			
Para seleccionar un juego: - El usuario debe estar autenticado como profesor.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Seleccionar juego	Una vez en la vista Lista de juego el profesor procede a seleccionar los juegos.	El sistema debe permitir seleccionar todos los juegos que el profesor desee seleccionar.	Clase X/Administrar Juegos/Lista de

			juegos/Seleccionar juego

Tabla 41 Caso de prueba del requisito listar juegos

Descripción general			
Permitir listar juegos en el sistema.			
Condiciones de ejecución			
Para listar juegos: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Listar	Una vez seleccionado Administrar juego el sistema muestra una tabla con todos los juegos existentes	El sistema muestra el listado de juegos y permite realizar las siguientes acciones: Crear, Editar y Eliminar.	Clase X/Administrar juegos/Listar juego

Tabla 42 Caso de prueba del requisito mostrar pista

Descripción general			
Permitir visualizar las pistas en el juego.			
Condiciones de ejecución			
Para mostrar pistas tiene que: - El usuario debe estar como estudiante. - Existir al menos un juego en el sistema. - El juego debe tener la opción de pistas.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar pistas	Una vez seleccionada la opción Mostrar juego el estudiante selecciona la opción Mostrar pista	El sistema muestra en el juego el objeto aleatorio que la pista debe mostrar.	Clase X/Mostrar juego/Mostrar pista

Tabla 43 Caso de prueba del requisito incluir nombre de la selección

Descripción general				
Permitir nombrar la selección de un área en el sistema.				
Condiciones de ejecución				
Para nombrar una selección: - El usuario debe estar autenticado como profesor.				
Escenario	**Descripción**	**Nombre de la selección**	**Respuesta del sistema**	**Flujo central**

EC 1.1 Opción Seleccionar Área	Una vez que seleccionas el área el sistema muestra un mensaje de información pidiendo el nombre de la selección.	N/A	El sistema debe permitir especificar el siguiente dato: (*) Nombre de la selección.	Clase X/Crear juego/Seleccionar Área
EC 1.2 Opción Aceptar	El usuario selecciona la opción Aceptar.	V	El sistema actualiza el nombre de la selección que será usada para ubicarla en la lista de objetos o para identificar las diferencias.	Clase X/Crear juego/Seleccionar Área/Aceptar
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	N/A	Regresa a la vista anterior.	Clase X/Crear juego/Seleccionar Área/Cancelar

Tabla 44 Caso de prueba del requisito listar objetos a encontrar

Descripción general			
Permitir listar los objetos a encontrar en un juego en el sistema.			
Condiciones de ejecución			
Para listar los objetos en el juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema. - El juego incluir un grupo de objetos.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar objetos	Una vez seleccionada la opción Mostrar juego el sistema lista los objetos.	El sistema muestra la lista de objetos a encontrar.	Clase X/Mostrar juegos/Mostrar objetos

Tabla 45 Caso de prueba del requisito mostrar puntuación

Descripción general			
Permitir visualizar la puntuación obtenida en un juego en el sistema.			
Condiciones de ejecución			
Para mostrar la puntuación en un juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**

EC 1.1 Opción Mostrar puntuación	Una vez seleccionada la opción Mostrar juego el estudiante interactúa con el juego.	A medida que el estudiante juegue y descubra objetos la puntuación se va actualizando y mostrando.	Clase X/Mostrar juego/Mostrar puntuación

Tabla 46 Caso de prueba del requisito mostrar objeto

Descripción general			
Permitir al estudiante visualizar un objeto encontrado en el sistema.			
Condiciones de ejecución			
Para mostrar un objeto en un juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema creado por el profesor.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar objeto	Una vez seleccionada la opción Mostrar juego el estudiante debe encontrar un objeto.	El objeto que el profesor señaló como verdadero es mostrado por el sistema en el juego.	Clase X/Mostrar juego/Mostrar objeto

Tabla 47 Caso de prueba del requisito mostrar tiempo

Descripción general			
Permitir al estudiante visualizar el tiempo en el sistema.			
Condiciones de ejecución			
Para mostrar el tiempo de un juego: - El usuario debe estar autenticado como estudiante. - Existir al menos un juego en el sistema creado por el profesor.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar tiempo	Una vez seleccionada la opción Mostrar juego el estudiante interactúa con el juego.	El sistema muestra el tiempo que debe jugar el estudiante en una cuenta regresiva.	Clase X/Mostrar juego/Mostrar tiempo

Tabla 48 Caso de prueba del requisito buscar juego en la lista de juegos

Descripción general				
Permitir al profesor encontrar un juego determinado rápidamente.				
Condiciones de ejecución				
Para buscar un juego tiene que: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema creado por el profesor.				
Escenario	**Descripción**	**Nombre del juego**	**Respuesta del sistema**	**Flujo central**

EC 1.1 Opción Buscar juego	El usuario selecciona la opción Buscar juego	V	El sistema debe permitir especificar el siguiente dato: (*) Nombre del juego. El sistema muestra el listado de los juegos con ese nombre.	Clase X/Administrar Juego/Buscar juego

Tabla 49 Caso de prueba del requisito listar estudiantes

Descripción general			
Permitir listar estudiantes en el sistema.			
Condiciones de ejecución			
Para listar estudiantes: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Listar estudiantes	Una vez que selecciona la opción Enviar juego se procede a seleccionar los estudiantes.	El sistema debe permitir al profesor visualizar el listado de estudiantes conectados a la clase a los cuales les enviará el juego.	Clase X/Administrar juegos/Enviar juego/Listar estudiantes

Tabla 50 Caso de prueba del requisito mostrar resultados del juego

Descripción general			
Permitir visualizar los resultados de todos los estudiantes en un juego en el sistema.			
Condiciones de ejecución			
Para visualizar los resultados del juego: - El usuario debe estar autenticado como profesor. - Existir al menos un juego en el sistema.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Mostrar resultados del juego	Una vez seleccionada la opción Mostrar resultados del juego, el sistema debe mostrarlos.	El sistema debe permitir al profesor visualizar los resultados de los estudiantes en el juego. En el listado se mostrarán los siguientes datos: • Recibido de • Tipo de juego • Nombre de juego • Evaluación	Clase X/Administrar juegos/Mostrar resultados

Tabla 51 Caso de prueba del requisito eliminar resultados

Descripción general			
Permite eliminar los resultados de los estudiantes del sistema.			
Condiciones de ejecución			
Para la selección tiene que: - El usuario estar autenticado como profesor.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Eliminar	Una vez seleccionado el o los resultados a eliminar el usuario selecciona la opción Eliminar.	El sistema debe pedir confirmación de eliminación antes de realizar la acción a través de un mensaje. El sistema debe permitir al usuario seleccionar una de las siguientes opciones para confirmar o cancelar la acción: Aceptar Cancelar	Clase X/Administrar juego/Lista de resultados/Eliminar
EC 1.2 Opción Aceptar	Una vez que el sistema muestra el mensaje de confirmación el usuario selecciona la opción Aceptar.	El sistema elimina el o los resultados seleccionados y actualiza el listado de resultados.	Clase X/Administrar juego/Lista de resultados/Eliminar/Aceptar
EC 1.3 Opción Cancelar	Una vez que el sistema muestra el mensaje de confirmación el usuario selecciona la opción Cancelar.	El sistema muestra nuevamente el listado de resultados.	Clase X/Administrar juego/Lista de resultados/Eliminar/Cancelar

Tabla 52 Caso de prueba del requisito notificar recibo de resultados

Descripción general			
Permitir al profesor ver la notificación.			
Condiciones de ejecución			
Para notificar al profesor el recibo de un resultado: - El usuario debe estar autenticado como profesor.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Administrar juegos	Una vez seleccionada la opción Administrar juegos, el sistema muestra la notificación.	El sistema debe permitir al profesor visualizar la notificación enviada desde la	Clase X/Administrar juegos

		aplicación cliente una vez que el estudiante termina de jugar.	

Tabla 53 Caso de prueba del requisito notificar recibo de juego

Descripción general			
Permite notificar al estudiante.			
Condiciones de ejecución			
Para notificar al estudiante tiene que: - El usuario estar autenticado como estudiante en su aplicación. - El profesor haber enviado un juego.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Clase X	Una vez seleccionada la Clase X, el sistema muestra la notificación.	El sistema debe permitir al estudiante visualizar la notificación de que ha recibido el juego enviado desde la aplicación del profesor.	Clase X

Anexo 3 Guía de observación

Durante la investigación con el método de observación se buscó identificar:

- Características similares que se puedan incluir en el módulo para la creación de juegos del tipo observación

GLOSARIO

Práctica dinámica: Juegos didácticos para que el proceso educativo sea de mayor interés para el educando.

Mnemónico: En informática, un **mnemónico** o **nemónico** es una palabra que sustituye a un código de operación (lenguaje de máquina), con lo cual resulta más fácil la programación, es de aquí de donde se aplica el concepto de lenguaje ensamblador.