Universidad de las Ciencias Informáticas
Facultad 10

Módulo SMProg para el Entorno Virtual de Aprendizaje de la Universidad de las Ciencias Informáticas.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores
Bárbara Marta Zerquera Zerquera
Yosbany Martínez Milanés

Tutores
Lic. Dunia Suárez Ferreiro
Lic. Daynel Marmol Lacal

Consultante
Ing. Ludiel Castro Barrios

Ciudad de la Habana, mayo 2010

"Año del 52 Aniversario de la Revolución"

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Bárbara Marta Zerquera Zerquera

_____

Yosbany Martínez Milanés

_____

Firma de los Autores

Lic. Dunia Suárez Ferreiro

_____

Lic. Daynel Marmol Lacal

_____

Firma de los Tutores

DATOS DE CONTACTO

Tutora: Dunia Suárez Ferreiro
Especialidad de graduación: Licenciada en Ciencias de la Computación.
Categoría docente: Asistente
Categoría Científica: ninguna
Años de experiencia: 4
Años de graduado: 4
Correo electrónico: dsuarezf@uci.cu

Tutor: Daynel Mármol Lacal
Especialidad de graduación: Licenciado en Ciencias de la Computación.
Categoría docente: Asistente
Categoría Científica: ninguna
Años de experiencia: 4
Años de graduado: 4
Correo electrónico: dmarmol@uci.cu

Autora: Bárbara Marta Zerquera Zerquera
Correo electrónico: bmzerquera@estudiantes.uci.cu

Autor: Yosbany Martínez Milanés
Correo electrónico: ymmilanes@estudiantes.uci.cu

Agradecimientos

Dedicatoria

Resumen

Con la creación del Entorno Virtual de Aprendizaje (EVA) en la Universidad de las Ciencias Informáticas (UCI), se ha aumentado la integración de diferentes proyectos productivos y de asignaturas donde el estudiante interactúa de forma directa con éstas y se evalúa mediante Moodle.

Las asignaturas de corte de programación por la complejidad y novedad de la materia que representa para los estudiantes, requiere de mucho interés y motivación por parte de éstos, para poder alcanzar buenos resultados en las evaluaciones. Para propiciar esta motivación se ha venido desarrollando una herramienta que va integrada con la Plataforma de Entorno Virtual de Aprendizaje; SMProg, pensada como respuesta para satisfacer esta necesidad de una mayor interacción profesor-estudiante. Para la integración de esta herramienta a la Plataforma Virtual de Aprendizaje es necesaria la implementación de la lógica de juego a partir de diseño de tableros previamente realizado.

En la presente investigación se pretende desarrollar una herramienta educativa para el EVA - UCI, que permita la utilización de Juegos Didácticos, la gestión de asignaturas, evaluación y el diseño dinámico de juegos; además que disminuya el tiempo presencial del profesor e incremente el aprendizaje del estudiante de una forma agradable.

Palabras Claves:

Herramienta educativa, proceso enseñanza-aprendizaje, módulos de juegos didácticos.

Índice

Introducción

Con el devenir de los años, el avance de las tecnologías ha traído consigo el desarrollo de espacios virtuales de intercambio en tiempo real para la comunicación interpersonal. El desarrollo del software educativo como herramienta de enseñanza permite una mejor integración de los estudiantes con el proceso de enseñanza-aprendizaje (PEA).

Esto trajo consigo que los sistemas de enseñanza se sumaran a este desarrollo debido al impacto que estaba cobrando a nivel mundial el uso de la red de redes y la Internet de forma particular, creando así los software de enseñanza que son una parte muy importante para la interacción profesor–estudiante.

Para la aplicación de esta novedosa tecnología existen los Entornos Virtuales de Aprendizaje (EVA), que propician la necesaria interacción profesor-estudiante para que el estudiante se sienta respaldado y seguro a la hora de enfrentarse a un primer impacto con una asignatura que no conoce o en el momento de evaluarse en un tema en el que tenga problemas para aprender.

Para la solución de este problema, en los sistemas de enseñanza de muchos países se utilizan juegos didácticos para contrarrestar la falta de interés hacia el estudio consecuente y voluntario del estudiante.

A partir del año 2005 en la Universidad de las Ciencias Informáticas (UCI) se comienza a utilizar la plataforma de Entorno Virtual de Aprendizaje (EVA), la cual desde ese entonces ha brindado una fuente de información e interacción, ya que el diseño y el desarrollo del EVA está basado en la filosofía de que "*la colaboración entre profesor-alumno transforma al alumno en protagonista del proceso de enseñanza-aprendizaje*" (Pages, 2006).

En el Departamento de Especialidades de la Facultad 10 se llegó a la conclusión después de realizar varias encuestas y observaciones de que existe una insuficiente motivación de los estudiantes hacia las asignaturas del programa de estudio de la carrera, especialmente las relacionadas con la disciplina de programación, donde los estudiantes por lo novedoso e imperante q resulta esta materia se encuentran de "manos atadas", por lo que es necesario llevar a cabo un método para realzar la autoestima del estudiante y de esta manera disminuir el trabajo presencial del profesor, ayudando de una manera grata al estudiante en su desarrollo.

En la Universidad de las Ciencias Informáticas se desarrolla un proyecto de innovación pedagógica para implementar módulos de juegos didácticos en el lenguaje Java, en particular la herramienta SMProg, que está diseñada para utilizarse como actividad dentro del Entorno Virtual de Aprendizaje (EVA). Esta herramienta se encuentra en fase de desarrollo, en su primera versión, aunque sólo permite los juegos didácticos de tablero, por ser juegos conocidos y de buena aceptación general, se propone la implementación de la lógica del juego para la interacción del estudiante con las asignaturas. Como parte de la herramienta SMProg se desarrollaron módulos para el diseño del tablero y la gestión de asignaturas, por lo que se requiere la construcción de un módulo que permita la integración de éstos dos y el juego según las reglas definidas.

A partir de la **situación problémica** existente se plantea el siguiente **problema científico**: ¿Cómo integrar los módulos de la herramienta educativa SMProg para el Entorno Virtual de Aprendizaje de la Universidad de las Ciencias Informáticas EVA - UCI?

Como **Objeto de estudio** se tienen las Herramientas para el Entorno Virtual de Aprendizaje en la Universidad de las Ciencias Informáticas (EVA-UCI) y como **Campo de acción**, Módulos de juegos didácticos para el Entorno Virtual de Aprendizaje (EVA-UCI).

Se ha trazado el siguiente **Objetivo General** de la investigación: Diseñar e implementar una aplicación que permita la integración de los módulos de la herramienta educativa para el EVA - UCI "SMProg" y además interactuar con Juegos Didácticos.

Idea a defender: Con la creación de un módulo para la implementación de la lógica para juegos didácticos se garantizará que los estudiantes interactúen con los tableros de juegos didácticos en la herramienta SMProg.

Para el desarrollo de la investigación se proponen los siguientes **objetivos específicos**:

1. Realizar un estudio referente a módulos de juegos didácticos para fundamentar la investigación.

2. Diseñar una aplicación donde se implemente la lógica de juego para propiciar la interacción del estudiante con los tableros de juegos didácticos.
3. Diseñar los componentes visuales de la aplicación.
4. Implementar la aplicación para la interacción del estudiante con los tableros de juegos didácticos.
5. Realizar pruebas a la aplicación.

Para darle respuesta a los objetivos anteriores es necesaria la realización de las siguientes **tareas investigativas**:

1. Realización de un estudio de las aplicaciones similares que existen a nivel mundial.
2. Diseño de la herramienta educativa con todos sus componentes.
3. Implementación de la herramienta educativa en correspondencia con las características diseñadas.
4. Realización de las pruebas a la aplicación.

Métodos teóricos utilizados

Durante el desarrollo de la investigación se emplea el método **Histórico-Lógico** para realizar un estudio profundo de las anteriores herramientas que no han podido dar la respuesta necesaria para el adecuado aprendizaje de los estudiantes, logrando una mejor integración de los conocimientos adquiridos.

Se usa el método **Análisis y Síntesis** para el análisis de las características de los diferentes tipos de juegos con tableros para lograr un correcto diseño de la implementación de la aplicación.

Se utilizó también el método de **Modelación** para representar el objeto de estudio de la aplicación y la posible solución del problema científico. Con este método se realiza el diseño de la herramienta a implementar.

El presente documento se encuentra estructurado en introducción, tres capítulos, conclusiones, recomendaciones y anexos.

En el **Capítulo 1** se presenta la fundamentación teórica acerca de los sistemas de aprendizajes didácticos y de las herramientas que permiten la adecuada

implementación para juegos que existen en el mundo, haciendo un acercamiento al ámbito nacional y de la UCI en general y las posibles soluciones que se le dan a estos problemas. Se tratará la descripción de las características generales de la aplicación.

En el **Capítulo 2** se presenta el modelo de negocio y el levantamiento de requisitos, se definen los casos de uso, actores y se muestran los diagramas de casos de uso del sistema y las respectivas descripciones textuales.

En el **Capítulo 3** se presenta el análisis y el diseño de la herramienta SMProg donde se presentan cada artefacto generado en este flujo de trabajo, diagramas y descripciones de clases que intervienen en el desarrollo de la aplicación.

En el **Capítulo 4** se presenta la implementación de la herramienta SMProg donde se presentan los artefactos generados en este flujo de trabajo, los diagramas correspondientes y las pruebas realizadas a la aplicación.

Capítulo 1: Caracterización y Definición del estado del arte para la herramienta educativa SMProg.

Desde inicios de la computación el uso de herramientas fáciles y amigables para la interacción del usuario final y el computador, ha permitido una mayor eficacia en el momento del aprendizaje y captación del conocimiento a partir del uso de la información y la final aclimatación por parte del usuario. El uso de las tecnologías a nivel mundial ha traído consigo un próspero adelanto en relación con el desarrollo del intelecto humano, aunque con la llegada de Internet se ha monopolizado de manera contundente el flujo de información, por lo que se ha extendido la captación de conocimientos por vía electrónica.

Fundamentalmente en el sistema de educación desde que se inicia el proceso enseñanza – aprendizaje se tiene en cuenta que el proceso de la enseñanza en sí es bastante complejo, por lo que se requiere de ciertas habilidades que son indispensables para la adquisición de conocimientos; de ahí que empiecen a tomar parte desde años tempranos los juegos didácticos como una forma de ayudar al acondicionamiento del estudiante con la materia que va a recibir, por eso, el empleo de los medios didácticos, que facilitan información y ofrecen interacciones facilitadoras de aprendizajes a los estudiantes, suele venir prescrito y orientado por los profesores, tanto en los entornos de aprendizaje presencial como en los entornos virtuales de enseñanza.

"El acto didáctico define la actuación del profesor para facilitar los aprendizajes de los estudiantes. Su naturaleza es esencialmente comunicativa." (1)

"La didáctica es el arte de enseñar y constituye la rama fundamental de la Pedagogía. Su objeto de estudio está bien delimitado: el Proceso de Enseñanza – Aprendizaje (PEA). El PEA está compuesto por dos procesos en si complejos, la enseñanza y el aprendizaje. En este proceso el docente puede actuar como guía o facilitador del aprendizaje de los estudiantes a partir de los métodos de enseñanza que utilice, creando un proceso de interacción entre los conocimientos que trasmite y la forma en que los estudiantes se apropian del mismo" (2).

En los sistemas educacionales que ya se han integrado con el desarrollo de la Informática existen distintos procesos de enseñanza que permiten una adecuada

integración de los estudiantes con las disciplinas que se les imparten, ya sea de manera directa con un profesor o por vía no presencial, o sea por vía e-learning como son los Entornos Virtuales de Aprendizaje (EVA), que traen como ventaja que exhiben una amplia bibliografía para consultar y disponibilidad de materiales de investigación como referencia. La utilización de juegos didácticos como método de aprendizaje en todas las enseñanzas ha concedido un mejor entendimiento por parte de los educandos a la hora de aprender.

El uso de software educativo, conjuntamente con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), han ofrecido progresos considerables en el desarrollo intelectual de aquellos que están integrados al Proceso Enseñanza - Aprendizaje (PEA) en cualquiera de sus ramas.

Con la evolución del software educativo se fueron definiendo algunas características que los mismos deben permitir:

- La interactividad con los estudiantes, retroalimentándolos y evaluando lo aprendido.
- Facilitar las representaciones animadas.
- Incidir en el desarrollo de las habilidades a través de la ejercitación.
- Permitir simular procesos complejos.
- Reducir el tiempo de que se dispone para impartir gran cantidad de conocimientos facilitando un trabajo diferenciado, introduciendo al estudiante en el trabajo con los medios computarizados.
- Facilitar el trabajo independiente y a la vez el tratamiento individual de las deficiencias de los estudiantes.

1.1. E-Learning.

1.1.1 ¿Qué es el e-learning?

E-Learning es una manera flexible y poderosa mediante la cual individuos y grupos adquieren nuevos conocimientos y destrezas con apoyo de tecnología de redes de computadoras. Ésta permite diseminar y tener acceso a información multimedia, hacer uso de simuladores, al tiempo que permite interacción y colaboración con aprendices que pueden estar dispersos alrededor del mundo.

El *e-learning* se desarrolla en la actualidad valiéndose de la Internet; sin embargo, en el futuro podría incluir *computadoras portátiles* con comunicación inalámbrica móvil, teléfonos celulares, y dispositivos de interacción que están articulados en objetos y artefactos de uso cotidiano.

El campo del *e-learning* se desarrolla muy rápidamente gracias a cuatro factores principales:

- Disponibilidad de redes de gran velocidad, para ofrecer información y servicios.
- Necesidad creciente de "trabajar con sabiduría" y con actualización continua de habilidades y destrezas.
- Conveniencia de que la educación sea justo a tiempo (a menudo "desde cualquier parte, cuando se necesite").
- Es una alternativa costo-efectiva a la educación y entrenamiento corporativos presenciales, en salón de clase.

1.2 EVA.

1.2.1 Entorno Virtual de Aprendizaje

Un Entorno Virtual de Aprendizaje EVA es un espacio con accesos restringidos, concebido y diseñado para que las personas que acceden a él; desarrollen procesos de incorporación de habilidades y saberes, mediante sistemas telemáticos, (3) teniendo en cuenta que para la creación del mismo es imprescindible la existencia de:

- Recursos de Contenidos en formato de páginas Web (Programas, Temas, Casos Prácticos, etc.).
- Repositorio de Recursos de Conocimiento, que recoge dinámicamente (inclusión de la URL y un resumen) los hallazgos de la utilización de Agentes Inteligentes y su inclusión como Web.
- Servicio de Clases Virtuales por medio de Streaming de Audio y Video de la asignatura, de modo que el alumno pueda asistir virtualmente a ellas en tiempo real o en tiempo diferido.
- Utilización del Chat para tutorías virtuales, junto a un servicio de mensajería personal instantánea y/o de videoconferencia ya sea por Internet u otros medios que incluyan el uso de la red.

- Dispositivos de Comunicación Virtual. Estos dispositivos representan la popularización del conocido "*groupware*" (conjunto de programas informáticos colaborativos) en un modo no centralizado (*peer to peer*).

Además, destacar que se puede definir: *"como un conjunto heterogéneo de recursos que comparten el soporte digital y la tecnología de Internet de modo sinérgico para posibilitar un nuevo modo de interacción humana orientado a la exploración y el aprendizaje".* (4)

Se puede decir que partiendo de estas perspectivas que nos queda como conclusión que un EVA no es más que una herramienta educativa en forma de espacio virtual donde administradores, profesores y alumnos se autentican y mediante el uso de repositorios para materiales, foros y todo un sistema de evaluaciones permiten disminuir el lapso presencial del profesor y propician que el estudiante pueda interactuar con el material ya sea de forma sincrónica o asíncrona, contribuyendo así al desarrollo del proceso enseñanza-aprendizaje como instrumento de mediación. En términos generales, un EVA establece una red de comunicación total entre todos sus usuarios, potenciando el aprendizaje, la cooperación, la creación de nuevas iniciativas, etc., con resultados altamente positivos en todas las direcciones.

1.2.2 Tipos de EVA y repercusión de su uso.

En el mundo se pueden encontrar diversos tipos de Entornos Virtuales de Aprendizajes que son ineludibles en el PEA y que permiten la interacción aunque no siempre en tiempo real de los profesores y los alumnos que intervienen en dicho proceso, sino también de que exista siempre la comunicación ya sea en foros o en conversaciones de tipo interpersonal. Entre estos se encuentran además los AVA (Ambiente Virtual de Aprendizaje) y las PVA (Plataformas Virtuales de Aprendizaje) que no dejan de ser en sí herramientas educativas para abrir una puerta virtual al conocimiento y por consiguiente el enriquecimiento del intelecto.

Los EVAs tienen una gran importancia en el PEA ya que facilitan el trabajo tanto a estudiantes como a profesores, permiten el intercambio de ideas y conocimientos entre los usuarios. No sólo sirven para impartir o recibir un contenido determinado, sino que es posible encontrar muchas fuentes de información que facilitan una preparación acorde a los deseos de cada estudiante.

Como se menciona con anterioridad entre los EVA más utilizados actualmente se encuentran:

- Dokeos
- ILIAS
- Sakai
- Claroline
- ATutor
- LRN
- BlackBoard
- Moodle (5).

Dokeos

Es un LMS (Learning Management Sistem) bien concebido que también ofrece autoría de contenidos y herramientas de videoconferencia. Soporta conversión de documentos Microsoft Office a Adobe Flash. Ofrece sincronización con sistemas de Recursos Humanos tales como Oracle y SAP quiénes utilizan el modelo ERP. Esta característica para administrar contenidos incluye distribución de contenidos, calendario, proceso de entrenamiento, chat en texto, audio y video, administración de pruebas y guardado de registros. Hasta el 2007, estaba traducido en 34 idiomas (y varios están completos) y es usado por más de mil organizaciones. (6)

ILIAS

En alemán (Integriertes Lern-, Informations- und Arbeitskooperations System), en inglés podría traducirse por Integrated Learning, Information and Cooperation System y en español como Sistema Integrado de Cooperación, Información y Aprendizaje; provee herramientas para test y tutoría así como elementos de colaboración como chats y foros y tecnologías de distribución como RSS y *podcasts* que no son más que proyectos abiertos y cooperativos sobre la temática, en forma de wiki y con listas de correo . (7)

Sakai

Concebido por Universidades reconocidas, este es un mapa de desarrollo del proyecto el cual ha avanzado considerablemente en los últimos años. Basado en Java/Tomcat es mantenido por la Sakai Foundation que gestiona las relaciones con

los apoyos educacionales y comerciales. El nombre Sakai proviene del cocinero Hiroyuki Sakai y entre las universidades que la integran se encuentran:

- Universidad de Indiana.
- Universidad de Michigan.
- Universidad de Yale.
- Universidad de Stanford.
- Universidad Politécnica de Valencia.
- Universidad del Valle de Guatemala. (8)

Claroline

Es una herramienta enfocada más bien al entorno educacional que al corporativo, este sistema está basado en principios pedagógicos específicos. Soporta contenido SCORM (*Sharable Content Object Reference Model*) así como herramientas de contenido como Wiki y otras herramientas online. (9)

Atutor

Es un programa diseñado en PHP, Apache, MySQL, trabaja sobre plataformas Windows, GNU/Linux, Unix, Solaris, soporte a 32 idiomas, contiene herramienta de Gerencia y administra alumnos, tutores, cursos y evaluaciones en línea, herramienta de Autoría incorporada, herramienta de Colaboración incorporada. (10)

LRN

(*Learn Research Network*), es un LMS completo de código abierto que cuenta con un sofisticado sistema de portales que integra herramientas para administrar cursos, contenidos y herramientas de colaboración. (11)

BlackBoard contiene 2 grupos que son:

- ***Blackboard Academic Suite*** que consiste de:

- ***Blackboard Learning System***, un entorno de manejo de cursos. Actualmente esta plataforma está siendo usada a nivel mundial por diversas instituciones relacionadas con la educación, tal es el caso del Servicio Nacional de Aprendizaje SENA en Colombia. También la utilizan la Universidad Nacional de Colombia y la Pontificia Universidad Javeriana. En México tiene presencia desde hace más de 10 años en diversas universidades e instituciones como el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM).
- **Blackboard Community System**, para comunidades en línea y sistemas de portales.
- **Blackboard Content System**, un sistema para el manejo de contenido.

- ***Blackboard Commerce Suite,*** *que consiste de****:***

- **Blackboard Transaction System**, un sistema de procesamiento de transacciones (tarjeta débito) para identificaciones de universidades.
- **Blackboard Community System**, Un sistema para transacciones de comercio electrónico.
- **Bb One**, una red comercial para procesar transacciones de tarjetas débito patrocinadas por BlackBoard. (12)

Moodle

Por sus siglas en inglés significa *Modular Object-Oriented Dynamic Learning Environment* (Entorno de Aprendizaje Dinámico Modular Orientado a Objetos), es un sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conocen como LMS (Learning Management System). Es multiplataforma y permite poder incluir o enlazar (link), al igual que múltiples blogs, web-quest, imágenes, videos o documentos, que hacen mucho más rico, variado y dinámico el trabajo. (13)

En nuestra universidad se utiliza EVA Moodle haciendo posible la integración del estudiante con las tecnologías, el cual contribuye de forma eficaz y certera con el proceso enseñanza–aprendizaje. El uso de la misma aún no soluciona la desmotivación por parte de los estudiantes hacia las asignaturas que incluye la plataforma, por lo cual se desarrollan módulos como parte de la herramienta SMProg para su posterior integración con el EVA Moodle.

Acarreando esto como premisa, se presenta la inexistencia del uso de juegos didácticos para poder conquistar el estímulo del intelecto de aquellos estudiantes que no alcanzan un "entusiasmo" por las asignaturas que están en la plataforma, por lo que la utilización de algún apoyo externo siempre es bienvenido a la hora de mejorar el PEA.

1.3 Juegos Didácticos.

1.3.1 Tipos de Juegos Didácticos.

El acto didáctico define la actuación del profesor para facilitar el aprendizaje de los estudiantes. Su naturaleza es esencialmente comunicativa. Los juegos didácticos como base de sustento para el aprendizaje y como ejercicio para poder incrementar el coeficiente intelectual, favorecen la agilidad mental y disminuyen en sí el uso de otros métodos poco ortodoxos que empobrecen la calidad de una idea bien impartida.

A medida que aumenta la edad de los estudiantes y el nivel de enseñanza que cursan, en las instituciones docentes disminuye la utilización de juegos didácticos, por lo que su aplicación decrece de la enseñanza primaria hacia la universitaria. (14)

Acorde a las características de los juegos didácticos: la participación, el dinamismo, el entretenimiento, la interpretación de papeles y la competencia estos también constan de tres fases:

- Introducción, ideada con el objetivo de iniciar al "jugador", dependiendo de algunas normas y pautas a seguir.
- Desarrollo, se lleva a cabo la actuación del jugador según las reglas del juego.
- Culminación cuando el jugador llega a la meta que no es más que el resultado de todo aquello que aprendió o ejercitó durante el desarrollo del juego.

Los juegos didácticos se dividen en las siguientes categorías:

- Juegos Creativos.
- Juegos Profesionales.
- Juegos Didácticos.

Juegos Creativos: Permiten desarrollar en los estudiantes la creatividad, estimulan la imaginación y la producción de nuevas ideas.

Juegos Profesionales: Estimulan la rapidez y la motivación para entender y aprender. Se centran en la materia que se va a impartir.

Juegos Didácticos: Resulta un método muy eficaz, sobre todo para la enseñanza problémica. Existen diversas variantes de cómo emplear estos:

- **De tipo Competitivo**: Encuentros de conocimiento, olimpiadas.
- **De tipo Profesional**: Análisis de situaciones concretas, análisis de casos, interpretaciones de papeles, simulación. (15)

De acuerdo con Armando Tesla en su libro:" Aprendizaje sobre juegos" , los principios que más caracterizan a los juegos didácticos son los siguientes:

- De participación activa.
- De dinamismo.
- De entrenamiento.
- De interpretación de roles.
- De carácter problémico.
- De obtención de resultados concretos.
- De competencia.

Los juegos didácticos pueden contribuir, en cualquier enseñanza, a elevar la calidad del PEA, siempre que se tengan en cuenta las características que deben cumplir para su efectividad. Entre los resultados que se han obtenido históricamente con su utilización destacan:

- Elevar el estudio individual de los estudiantes.
- Mejorar los resultados docentes.
- Lograr una interacción cooperativa y competitiva entre un colectivo de estudiantes.
- Motivar al estudiantado.
- Encontrar nuevas formas de evaluación. (16)

En la búsqueda de métodos y estrategias de enseñanza y de aprendizaje cada vez más adaptados al entorno telemático y, a la vez, capaces de contribuir a los objetivos pedagógicos se crea como representación un juego didáctico que cumpla con las

expectativas de integrarse con asignaturas del EVA Moodle, que son aquellas que se estudian en la carrera de Ingeniería en Ciencias Informáticas.

1.4 Módulos de Juegos Didácticos para asignaturas de corte de programación.

En este marco el empleo de los medios didácticos, que facilitan información y ofrecen interacciones facilitadoras de aprendizajes a los estudiantes, suele venir prescrito y orientado por los profesores, tanto en los entornos de aprendizaje presencial como en los entornos virtuales de enseñanza.

La selección de los medios más adecuados a cada situación educativa y el diseño de buenas intervenciones educativas que consideren todos los elementos contextuales (contenidos a tratar, características de los estudiantes, circunstancias ambientales...), resultan siempre factores clave para el logro de los objetivos educativos que se pretenden.

Existen actualmente ciertos módulos diseñados con diferentes objetivos, algunos de ellos como apoyo para asignaturas determinadas de distintas carreras ingenieriles. Aunque apoyan al PEA sólo Jclic, que es un entorno para la creación, realización y evaluación de actividades educativas multimedia, desarrollado en la plataforma Java, es capaz de integrarse a Moodle como módulo ya que permite el diseño y la utilización de algunos juegos didácticos (crucigramas, sopa de palabras, asociaciones de palabras, etc.) para la enseñanza primaria y secundaria preferentemente. De ahí que exista la posibilidad de adaptarlo a la enseñanza universitaria para vincularlo a las asignaturas de corte de programación como se hace necesario particularmente en la carrera de Ingeniería en Ciencias Informáticas.

Entre las principales características de Jclic se encuentran las siguientes:

- Esta implementado sobre la plataforma Java, con un formato de almacenamiento de la información en ficheros XML.
- Sus componentes son
    - **Jclic- applet:** es un applet que permite que se integre a una aplicación Web, en este caso a Moodle.
    - **Jclic-reports**: es un módulo de recogida de datos sobre las actividades que realizan los estudiantes.

  - **Jclic-author**: permite crear y modificar proyectos Jclic, en un entorno visual muy intuitivo e inmediato.
- Tiene arquitectura abierta que permite ampliar o adaptar sus funcionalidades.
- Es un proyecto de software libre. (17)

Sin embargo, para los requerimientos del nivel universitario esta herramienta no es suficiente, pues está definida con un alcance menor, no obstante es bueno resaltar que algunas de sus características comunes con respecto a SMProg, se tienen en cuenta para el desarrollo más rápido de esta nueva herramienta educativa. (Ing. Ludiel Castro).

Por lo tanto en la universidad se lleva a cabo un proyecto de innovación pedagógica que permite la creación de una aplicación a partir de un ya realizado diseño de tableros para juegos didácticos y de acuerdo a las características del juego se realiza la implementación de la Lógica de Juego, para después de fusionarla con un sistema gestor de bases de datos integrarla con EVA Moodle.

1.5 SMProg

La herramienta educativa SMProg (Software – Motivación - Programación) permite al profesor definir el sistema de preguntas que desea se muestren a los estudiantes que se encuentren "jugando". Las preguntas podrán agruparse por temas y saldrán en un inicio de forma aleatoria. De acuerdo a los temas en que más se confunda el estudiante, el propio software mostrará un mayor número de preguntas de ese tipo y realizará la evaluación del mismo. Esta herramienta también permite al profesor realizar el diseño del juego y su vinculación con el sistema de preguntas previamente definido, así como la interacción del estudiante con la aplicación.

En la primera versión de la herramienta educativa se incluyó sólo el diseño e implementación de Juegos Didácticos con tableros debido a que son bien conocidos y tienen buena aceptación general. En la primera versión del software se desarrollan tres módulos diferentes estrechamente vinculados entre sí y con similar ambiente que son los siguientes:

- Módulo Diseño de Tableros (**DesBoarD**): Herramienta creada para la modelación por parte del profesor del tablero correspondiente de acuerdo al

tipo de juego a jugar, ya sea Avanza hasta la Meta, Bingo o Sube y Baja, desarrollada por el Ing. Ludiel Castro.

- Módulo Gestión de Asignaturas (**VirtualKnow**): Las funcionalidades de este módulo requieren en gran medida la interacción con la plataforma Moodle. En caso de que se esté interactuando con la misma, su funcionalidad principal sería la definicón de temas, sistemas de preguntas y de evaluación de las asignaturas en caso de no estar vinculado a Moodle.
- Módulo Juegos didáctico con Tableros (**SMProg**): Desde este módulo se integran los restantes, es por esto que se nombra módulo principal. Desde el mismo es posible gestionar la asignatura, diseñar los tableros de juegos o iniciar el juego, con sus características propias, además de administrar la aplicación (roles - usuarios), manteniendo, en posteriores versiones, la comunicación con Moodle a través de applets y de la base de datos.

1.5.1 Módulo de Juegos Didácticos con Tableros.

Este módulo permite que los estudiantes de una forma sencilla y recreativa amplíen sus conocimientos y desarrollen habilidades educativas mediante una interfaz que permite, de forma automática, mostrar las preguntas en orden creciente de complejidad.

Juegos Didácticos con Tableros consta de dos subsistemas:

- Diseño de Tablero.
- Implementación de juegos de tableros.

El Diseño de Tablero permite al profesor diseñar el tablero teniendo en cuenta las características de sus estudiantes, ya sea motivación por la asignatura o dificultades en la misma.

En esta interfaz visual se puede elegir el tipo de juego de tablero deseado para posteriormente ubicar obstáculos dentro de las casillas, así como definir casillas de tipo didáctico.

Una vez diseñada la interfaz visual del juego es necesario poder interactuar con la misma, aquí es donde entra la Implementación de juegos de tableros que permite a los estudiantes jugar y aprender.

En esta interfaz se lleva a cabo la integración de los restantes módulos de esta herramienta que consta de 3 tipos de tableros que según sus propiedades se dividen en:

- **Avanzar hacia la meta**: en este tipo de tablero las casillas no tienen asociado ningún comportamiento especial. Simplemente se avanza hacia la meta (hacia adelante) según el número obtenido en el lanzamiento de los dados.
- **Bingo**: las casillas de este tablero carecen de un comportamiento particular, para poder concluir el juego es necesario que se complete una cantidad prefijada de casillas contiguas horizontal, vertical o diagonalmente.
- **Sube y Baja**: el tablero consta de casillas en las que se permanece, avanza o retrocede hacia otra casilla ubicada en el mismo, con la utilización de un dado, ya sea hacia adelante, atrás, arriba o abajo.

Y de una base de datos que va a tener toda la información almacenada e implementada sobre el gestor libre PosgreSql. Entre la información almacenada se encuentran los resultados de los estudiantes por temas y por asignatura (historial del estudiante), los temas que conforman la asignatura, el sistema de preguntas por temas con diferentes niveles de complejidad, así como el sistema de evaluación de la asignatura en cuestión.

Desde este módulo (**SMProg**) se integran los restantes, es por esto que lo llamamos módulo principal. Desde el mismo es posible gestionar la asignatura, diseñar los tableros de juegos e iniciar el juego, con sus características propias, además de administrar la aplicación (roles - usuarios), manteniendo la comunicación con Moodle a través de applets y de la base de datos una vez que sea integrado a este.

1.5.2 Lógica de Juego.

La implementación de la Lógica de Juego debe cumplir con las especificaciones de las características del juego para cada uno de los tableros que van a ser utilizados en el módulo a integrar.

Este subsistema es capaz de guiar a los estudiantes por un sistema de preguntas que al inicio es a través de una pregunta por cada tema, analizando cuál es el primer tema de la asignatura que el estudiante debe ejercitar, dado sus malos resultados en el mismo, para mostrar un mayor número de preguntas de ese tipo.

Ésta va a viabilizar la interacción del estudiante con las asignaturas de corte de programación primeramente y en un futuro con el resto de las asignaturas que estén vinculadas a Moodle y que de esta forma cree en sí la esperada motivación carente en la Universidad hacia este tipo de asignatura.

Mediante una interfaz amigable se va a garantizar que el estudiante se interese en el juego y se preocupe por llegar siempre a la meta o ganar de forma general en conocimientos y agilidad mental.

1.6 Metodología de desarrollo de Software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, métodos y técnicas que ayudan a organizar el trabajo y la documentación con el objetivo de facilitar el desarrollo del producto. Es decir, son una guía de pasos a seguir, donde además indican qué persona y qué papel debe tener la misma para realizar una actividad específica en el desarrollo del software. Además detallan la información que se debe producir después de terminada una actividad.

Las metodologías se dividen en dos grandes grupos:

- Metodologías Ligeras. (XP (Programación extrema), *Crystal Light Methods, etc*).
- Metodologías Pesadas. (SW-CMM (*SoftWare Capability Maturity Model*), RUP (Proceso Unificado de Rational).

Las ligeras se basan en conseguir su objetivo por medio del orden y la documentación, mientras que las pesadas se basan en la comunicación directa e inmediata entre aquellos que intervienen en el proceso.

Se entiende como desarrollo ágil de software a un paradigma de desarrollo de software basado en procesos ágiles. Los procesos ágiles de desarrollo de software, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos ágiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (*bullpen* en inglés). La

oficina debe incluir revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica.

Algunas metodologías ágiles de desarrollo de software son:

- *Adaptive Software Development* (ASD).
- *Agile Unified Process* (AUP).
- *Crystal Clear.*
- *Essential Unified Process (*EssUP).
- *Feature Driven Development* (FDD).
- *Lean Software Development* (LSD).
- *Open Unified Process* (OpenUP).
- Programación Extrema (XP).
- *Scrum*.
- SXP (Scrum/XP)(17)

Entre las metodologías pesadas citadas podemos describir que:

Para SW-CMM existen 3 Niveles de madurez definidos que son:

- 3.1 Nivel 1: Inicial
- 3.2 Nivel 2: Repetible
- 3.3 Nivel 3: Definido
- 3.4 Nivel 4: Gestionado
- 3.5 Nivel 5: Optimizado (18)

El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Entre sus principales características tenemos:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software. (19)

En este caso en particular se utiliza la metodología RUP para el desarrollo de la aplicación, por todas las ventajas de organización que brinda, por dividir el trabajo en roles, por ser adaptable a cualquier tipo de proyecto, ya sean grandes o sencillos y por ser una metodología orientada a objetos. Además de que la aplicación forma parte del proyecto de Innovación Pedagógica “Herramienta Educativa sobre Software Libre para las asignaturas de programación en la Universidad de las Ciencias Informáticas”, que se desarrolla con esta metodología.

1.6.1 Proceso Unificado de Rational (RUP).

RUP es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), conforma la metodología más utilizada en el mundo tanto para el desarrollo de pequeños proyectos como para sistemas de software complejos pues está pensada para adaptarse a diferentes áreas de aplicaciones y diferentes organizaciones.

RUP se divide en cuatro fases (Inicio, Elaboración, Construcción, Transición) y presenta nueve flujos de trabajo de los cuales seis son de ingeniería de software y tres de apoyo, estos son:

- Modelo del negocio.
- Requerimientos.
- Análisis y Diseño.
- Implementación.

- Prueba.
- Instalación o distribución.
- Configuración y administración del cambio.
- Administración de proyectos.
- Ambiente.

Como una de sus principales características está el hecho de que en cada ciclo de iteración se hace exigente el uso de artefactos.

1.7 Tecnologías actuales a considerar.

Para el desarrollo de esta aplicación se utiliza el lenguaje de programación Java que no es más que un lenguaje orientado a objetos desarrollado por *Sun Microsystems*. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Actualmente existen muchas aplicaciones gráficas de usuario basadas en Java. El entorno de ejecución Java (*JRE - Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java*-) se ha convertido en un componente habitual en los PC de usuario de los sistemas operativos más usados en el mundo. Además, muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se ejecuten en cualquier PC. (20)

1.7.1 Java como lenguaje de desarrollo.

Se escogió este lenguaje por dos razones fundamentales:

- Portabilidad de su plataforma.
- Se puede incrustar en otro programa.

Portabilidad de su plataforma:

El compilador del lenguaje Java genera un código binario especial (el Java *bytecode*) que es interpretado por una Máquina Virtual (JVM) sus siglas en ingles significan Java Virtual Machine.

La Máquina virtual de Java (JVM **–** *Java Virtual Machine* -) es un programa nativo, es decir, es un ejecutable en una plataforma específica, capaz de comprender tanto el

bytecode como el sistema en el que se va a ejecutar la aplicación. Así, cuando se escribe una aplicación Java, se hace pensado para que se ejecute en una máquina virtual de Java, siendo ésta la que en última instancia convierte de código *bytecode* a código nativo del dispositivo final.

La gran ventaja de la máquina virtual de java es su portabilidad, de manera que su creación para diferentes arquitecturas logra que un programa escrito en Windows pueda ser interpretado en un entorno libre, disponiendo solamente de la máquina virtual para dichos entornos. De esto se desprende el famoso axioma que sigue a Java, "escríbelo una vez, ejecútalo en cualquier parte". (21)

Se puede alojar en otro programa:

Una de las funcionalidades que potencian el uso del lenguaje de programación Java es la incorporación de los applets.

Un applet es un componente de Java que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet, a diferencia de otros programas, no puede ejecutarse de manera independiente, debe ejecutarse siempre en un contenedor. Ésta es la principal razón por la que se ha escogido este lenguaje ya que puede permitir que la aplicación desarrollada pueda ser ejecutada por cualquier usuario con tan sólo cargar la página Web en su navegador. (22)

1.7.2 Ambiente de desarrollo.

Actualmente existe gran variedad de IDEs (*Integrated Development Enviroment*) para el desarrollo de aplicaciones Java, como son *WebSphere Studio Application Developer*, *Rational Application Developer*, *JBuilder* entre otros, aunque solo se prestará atención a los IDEs *Open Source* (código abierto), en especial el NetBeans que ha sido escogido para el desarrollo de la aplicación, aunque existan otras herramientas como Eclipse, IntelligentJ con su versión libre, Jedit, etc.

NetBeans es un proyecto exitoso con una gran comunidad en constante crecimiento lo que facilita el trabajo con el mismo, fue fundado como proyecto de código abierto por Sun MicroSystems quien continúa siendo su patrocinador principal. Es un entorno de desarrollo con el que se puede escribir, compilar, depurar y ejecutar programas. Está escrito en el propio Java pero se puede utilizar para cualquier otro lenguaje de programación que soporte esta herramienta. Cuenta con un número importante de

módulos para extenderlo, es un producto libre y gratuito sin restricciones de uso. La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear grandes aplicaciones de escritorio, ofrece soporte para los plug-in AWT/SWING entre otros, aunque solo estos serán utilizados en la creación de la aplicación. El NetBeans viene integrado con todas las herramientas necesarias para el desarrollo Web o Desktop a diferencia del Eclipse, que es necesario ir instalando pluggins según sea necesario.

En este trabajo no se pretende demostrar que el NetBeans es el mejor IDE Open Source para el desarrollo en Java solo que ofrece la ventajas necesarias para lograr una conformidad en la realización de la aplicación.

1.7.3 Herramienta Case.

Se cuenta con la herramienta Case (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) Visual Paradigm 3.4 que utiliza “UML” como *lenguaje* de modelado, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Lista de características:

- Soporte de UML versión 2.1
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento
- Modelado colaborativo con CVS y Subversión (nueva característica)
- Interoperabilidad con modelos UML2 (meta-modelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).
- Ingeniería de flujo normal e ingeniería inversa.
- Ingeniería inversa - Código a modelo, código a diagrama
- Ingeniería inversa Java, C++, Esquemas XML, XML,.NET exe/dll, CORBA IDL Generación de código - Modelo a código, diagrama a código
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso

- Diagramas EJB - Visualización de sistemas EJB.
- Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
- Diagramas de flujo de datos
- Soporte ORM - Generación de objetos Java desde la base de datos
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación
- Generador de informes para generación de documentación
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML
- Importación y exportación de ficheros XMI
- Integración con Visio - Dibujo de diagramas UML con plantillas (esténcils) de MS Visio
- Editor de figuras. (23)

Además permite la integración con las siguientes herramientas Java:

- *Eclipse/IBM WebSphere.JBuilder.*
- *NetBeans IDE.*
- *Oracle Jdeveloper.*
- *Sun ONE.*
- *IntelligentJ.*

En especial con el NetBeans que es el IDE a utilizar en el desarrollo de la aplicación, aportando facilidad y comodidad en el generador de código o de diagramas.

En la realización de esta etapa del trabajo han sido abordados los principales conceptos que de una forma u otra se relacionan con el trabajo como son las definiciones de:

- Qué es Moodle y sus principales ventajas y desventajas.
- Qué es un (EVA) y su impacto en el mundo.
- La importancia de los juegos didácticos.
- Qué es SMProg.

Conclusiones

Partiendo de estos análisis y estudiando la situación problémica planteada se propone realizar un módulo que permita mediante la implementación de la Lógica de Juego agrupar los módulos para una posterior integración con el EVA - Moodle. También se hizo un estudio de las principales tecnologías, metodologías y herramientas de diseño a considerar para el desarrollo de la aplicación. Con esto se concluye la caracterización y la definición del estado del arte para la herramienta educativa SMProg.

Capítulo 2: Características del Sistema para la herramienta educativa SMProg.

En el siguiente capítulo se describen las características del sistema que se propone para dar solución al problema existente, se describen breves elementos del proceso de creación de tableros y de la utilización de juegos didácticos. Se exponen las diferentes clases del diseño junto con la explicación de cada una de ellas. Se hace referencia a los principales Casos de Uso del sistema junto con sus Diagramas de Clases. Se definen conceptos y se identifican entidades que se relacionan en un modelo de negocio realizado previamente, el cual permite determinar los requisitos necesarios que debe cumplir la aplicación para cubrir las necesidades que lo originaron.

2.1 Modelo de Negocio

Debido a que en la metodología del proceso de desarrollo del software está bien definido el modelo de negocio en la fase de inicio, se procura viabilizar la implementación de la Lógica de Juego por lo que se retoman todos los conceptos definidos que se utilizaron en el diagrama mediante un **glosario de términos:**

El **usuario** es aquella persona que posee los privilegios de acceder a la aplicación.

La **aplicación** es la herramienta capacitada de cargar los tableros de juegos didácticos y su configuración correspondiente de acuerdo a cada uno de los tableros.

El **tablero** es un fichero que contiene la información necesaria para ser utilizada en el desarrollo del juego didáctico seleccionado.

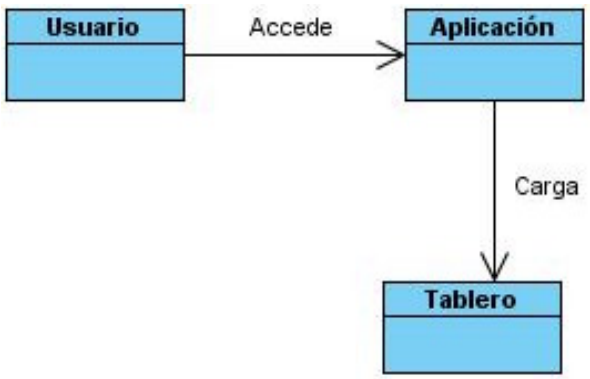


Figura 1 Diagrama del modelo del negocio

En la figura No.1 se muestra la relación entre las entidades definidas por el Modelo de Negocio, donde es posible observar como los usuarios acceden a la aplicación y obtienen como resultado final un fichero, el cual está previamente creado y contiene el tipo de juego solicitado.

2.1.2 Casos de Uso del Negocio

Un caso de Uso debe estar basado en la forma en que el sistema es visto externamente por un usuario sin detallar interfaces externas, formatos de datos, reglas de negocio ni fórmulas complejas, sino que es un artefacto narrativo que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. A continuación se muestra el diagrama de Casos de Uso correspondiente al modelo de negocio.

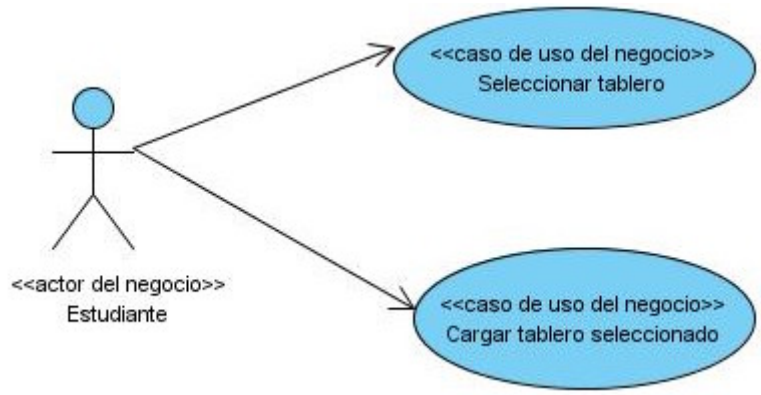


Figura 2 Diagrama de Casos de Uso del Negocio

En la figura 2 se muestra la relación que existe entre el actor del negocio, el estudiante y los respectivos casos de uso del negocio, los cuales van a estar vinculados con otros casos de uso u otras funcionalidades que se les van a agregar a su desarrollo posterior.

2.2 Levantamiento de Requisitos

En la primera fase del *Rational Unified Process* (RUP), se tiene el flujo de trabajo de la Captura de Requisitos la cual es imprescindible para la realización certera de todo proyecto de *software* ya que los mismos son la base para otras fases de desarrollo e implican a todos los miembros del equipo de trabajo.

La *IEEE Standard Glossary of Software Engineering Terminology* define un requerimiento como: Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. También es conocido como condición o capacidad definida como representación documentada que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. (23)

2.2.1 Requisitos Funcionales

Un Requisito Funcional no es más que la capacidad o condición que el sistema debe cumplir. Definen qué debe hacer el software. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

Los requisitos funcionales que identifican este sistema son:

1. El sistema debe permitir la utilización de tableros.
   - 1.1 Permitir seleccionar tipo de tablero (Avanzar hasta la Meta, Bingo o Sube y Baja).
   - 1.2 Permitir cargar el tablero seleccionado.
   - 1.3 Permitir jugar en el tablero seleccionado.
2. Seleccionar la Configuración del Juego.
   - 2.1 Seleccionar el tema a ejercitar.
   - 2.2 Seleccionar si va a jugar con vinculación a otras asignaturas o no.
3. Permitir que el estudiante empiece a jugar.
   - 3.1 Permitir al estudiante el lanzamiento del dado.
   - 3.2 Permitir al estudiante la utilización del bombo.
   - 3.3 Permitir al estudiante responder una pregunta.
   - 3.4 Permitir al estudiante avanzar en el tablero si responde correctamente la pregunta.
   - 3.5 Permitir al estudiante retroceder en el tablero si responde incorrectamente la pregunta.
4. Cumplir las reglas generales del Juego.
   - 4.1 Mostrar las preguntas al estudiante en orden creciente de complejidad una de cada tema la primera vez.
   - 4.2 Mostrar un mayor número de preguntas del peor tema.
   - 4.3 Permitir al jugador resolver las casillas didácticas.

4.4 Permitir al jugador sortear los diferentes obstáculos.

2.2.2 Requisitos No Funcionales

Son propiedades o cualidades que el producto debe tener. Una vez que se conozca qué debe hacer el sistema puede determinarse como ha de hacerlo. En muchos casos son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales (1). Los Requisitos No Funcionales que identifican a este sistema son:

Interfaz externa: Se propone una interfaz amigable y sencilla para el disfrute de los usuarios involucrados en el proceso.

Interfaz interna: Interactúa con un Gestor de Bases de Datos desarrollada en PostgreSQL.

Hadware: Se puede utilizar una PC Pentium III o superior con un mínimo de 128 MB de memoria RAM para el buen funcionamiento del programa.

Portabilidad: El sistema deberá ser multiplataforma, para esto se utilizará Java Virtual Machine (JVM) 6.0 o superior.

Restricción en el diseño la implementación: Se programará en lenguaje Java.

Software: Se utilizará la herramienta Netbeans 7.0 para la programación del sistema a implementar.

Soporte: Se brindará una ayuda o un manual de usuario para los jugadores.

Usabilidad: Será fácil de usar por parte de los usuarios aunque sea la primera vez que interactúen con la aplicación.

2.3 Modelo de Casos de Uso del Sistema

El estudiante es el único actor del sistema que interactúa con la aplicación, iniciando todos los casos de uso del sistema. En la figura No. 3 se observan los casos definidos para representar el flujo de los eventos que permiten obtener un resultado de valor para los estudiantes durante la interacción con el juego.

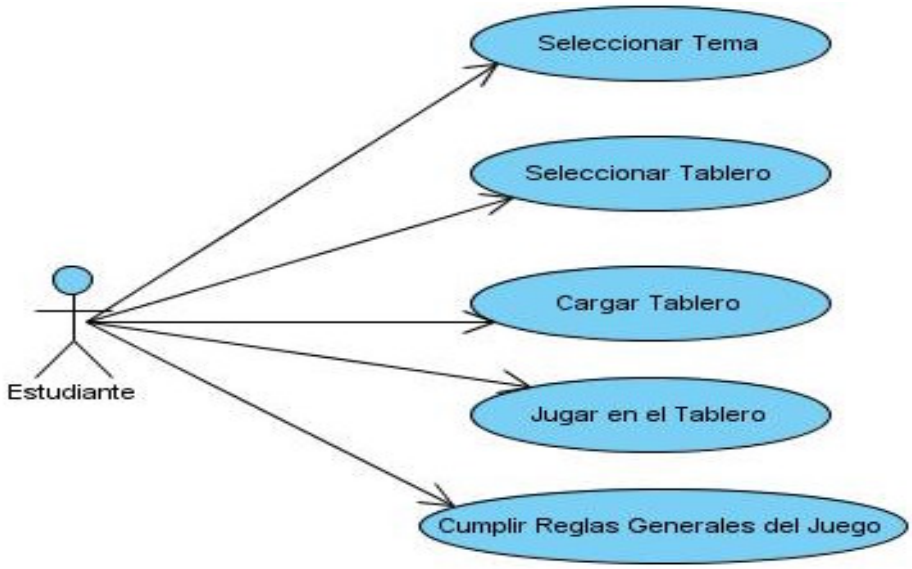


Figura 3 Diagrama de Casos de Uso del Sistema

Definición de los actores.

Tabla 1: Actor del sistema

Actor	Descripción
Estudiante	Es el encargado de seleccionar el tablero de los juegos, así como de cargar dicho tablero y definir la forma en que va a jugar.

2.3.1 Casos de Uso del Sistema

Tabla 2: Caso de Uso Seleccionar Tablero

CU-1	Seleccionar Tablero
Actor	Estudiante
Descripción	El estudiante selecciona el tipo de tablero deseado. El tablero ya va a estar previamente diseñado y va a contener el tipo de juego a jugar.
Referencia	R1

Tabla 3: Caso de Uso Seleccionar Tema

CU-1	Seleccionar Tema
Actor	Estudiante
Descripción	El estudiante selecciona el tema deseado. El tema va a estar previamente definido.
Referencia	R2

Tabla 4: Caso de Uso Cargar Tablero

CU-2	Cargar Tablero
Actor	Estudiante
Descripción	El estudiante carga el tipo de tablero seleccionado. El tablero ya va a estar previamente diseñado y va a contener el tipo de juego a jugar.
Referencia	R1

Tabla 5: Caso de Uso Jugar en el Tablero

CU-3	Jugar en el Tablero
Actor	Estudiante
Descripción	El estudiante empieza a jugar en el tipo de tablero seleccionado. En dependencia del tipo de tablero seleccionado el jugador empezará el juego con dado o no.
Referencia	R3

Tabla 6: Caso de Uso Cumplir Reglas del Juego

CU-4	Cumplir Reglas Generales del Juego
Actor	Estudiante
Descripción	El estudiante debe ser capaz de cumplir con las reglas

	establecidas para cada tipo de juego según el tablero seleccionado.
Referencia	R4

2.3.2 Descripción de Casos de Uso

Describe cómo se lleva a cabo y se ejecuta un caso de uso determinado, en términos de las clases del análisis y de sus objetos en interacción.

Tabla 7: Descripción textual del Caso de Uso Cumplir SeleccionarTema

CU-1	**Seleccionar Tema**
Actores:	Estudiante
Propósito:	Permitir al estudiante seleccionar tema con el que va a ejercitar.
Resumen: El estudiante podrá ver los temas con los que va a ejercitar.	
Referencias:	R2
Precondiciones:	Debe existir una conexión con una base de datos.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	Respuesta del sistema
1) El estudiante escoge el tema que desea ejercitar.	1.1) El sistema guarda en la configuración del Juego para el estudiante en particular los Temas seleccionados. 1.2) El sistema muestra un listado con todas las asignaturas con que puede interactuar el estudiante.
2) El estudiante selecciona las asignaturas con las que va a interactuar durante el Juego.	2.1) El sistema guarda en la configuración del Juego para el estudiante en particular las asignaturas seleccionadas. Finaliza el Caso de Uso Seleccionar tema.
Poscondiciones:	El sistema almacena en una Base de Datos la configuración del Juego para el estudiante.

Tabla 8: Descripción textual del Caso de Uso Cumplir Seleccionar Tablero

CU-1	**Seleccionar Tablero**
Actores:	Estudiante
Propósito:	Permitir al estudiante seleccionar el tipo de tablero con el que va a interactuar.
Resumen: El estudiante podrá ver los tipos de tableros (Avanzar hasta la Meta, Bingo, Sube y Baja).	
Referencias:	R1
Precondiciones:	Debe existir una conexión con una base de datos.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	Respuesta del sistema
1) El estudiante accede al área de Tableros.	1.1) El sistema muestra los tipos de tableros disponibles.
2) El estudiante selecciona el tipo de tablero con el que va a jugar.	2.1) El sistema muestra el listado de los Temas que el estudiante puede escoger para la asignatura.
Poscondiciones:	El sistema almacena en una Base de Datos la configuración del Juego para el estudiante.

Tabla 9: Descripción textual del Caso de Uso Cargar Tablero

CU-2	**Cargar Tablero**
Actores:	Estudiante
Propósito:	Permitir al estudiante cargar el tipo de tablero seleccionado y con el que va a interactuar.
Resumen: El estudiante podrá ver el tipo de tablero seleccionado (AvanzarMeta, Bingo, SubeBaja).	
Referencias:	R1

Precondiciones:	El estudiante debe seleccionar el tablero y el tema.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	Respuesta del sistema
1.1) El estudiante ve el tablero seleccionado.	1) El sistema muestra el tipo de tablero seleccionado. Finaliza el Caso de Uso Seleccionar tablero.
Poscondiciones:	El sistema muestra en la aplicación el tipo de tablero seleccionado por el estudiante autenticado.

Tabla 10: Descripción textual del Caso de Uso Jugar en el Tablero

CU-3	**Jugar en el tablero**
Actores:	Estudiante
Propósito:	Que el estudiante pueda ejercitar sus conocimientos de forma didáctica.
Resumen: Permitir al estudiante jugar en el tablero seleccionado.	
Referencias:	R1,R3
Precondiciones:	El estudiante debe tener el juego cargado.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	Respuesta del sistema
1) El estudiante lanza el dado.	1.1) El sistema muestra una ficha en la salida dependiendo del tipo de tablero y cuando para el dado mueve la ficha y lanza una ventana con una pregunta de acuerdo al nivel de complejidad que le corresponde a dicha casilla en la que se encuentre la ficha.
2) El estudiante responde correctamente la pregunta.	2.1) El sistema guarda en el historial del estudiante que la pregunta fue respondida correctamente.

	2.2) Avance o retroceso del estudiante en el tablero según el número alcanzado en el lanzamiento de los dados y del tipo de tablero. Finaliza el caso de uso Jugar en el tablero.
CURSO ALTERNO DE LOS EVENTOS 1	
Viene del paso 2 del curso normal de los eventos.	
1) El estudiante no responde correctamente la pregunta.	1.1)El sistema guarda en el historial del estudiante que la pregunta no fue respondida correctamente. 1.2)El sistema muestra una nueva pregunta de similar nivel de complejidad. 1.3)Se retorna al paso 2 del curso normal de los eventos.
CURSO ALTERNO DE LOS EVENTOS 2	
Viene del paso 2 del curso normal de los eventos.	
1) El estudiante no responde correctamente la pregunta en una cantidad de ocasiones igual al número máximo de intentos prefijados por el profesor para preguntas de ese nivel de complejidad.	1.1) El sistema guarda en el historial del estudiante que la pregunta no fue respondida correctamente. 1.2) El sistema muestra una vía de estudio para vencer las dificultades del estudiante. 1.3) El sistema reinicia el Juego hasta tanto el Jugador no esté listo para volverlo a intentar. Finaliza el caso de uso Jugar en el tablero.
Poscondiciones:	El historial del estudiante en la asignatura queda actualizado en la base de datos.

Tabla 11: Descripción textual del Caso de Uso Cumplir Reglas Generales del Juego

CU-4	**Cumplir Reglas Generales del Juego**
Actores:	Estudiante
Propósito:	Que el estudiante juegue siguiendo las reglas del tipo de juego seleccionado.
Resumen: Permitir al estudiante que se acote y cumpla cabalmente las reglas del juego.	
Referencias:	R4
Precondiciones:	El estudiante debe tener el juego cargado.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	Respuesta del sistema
1.1) El estudiante responde la pregunta.	1) El sistema muestra una pregunta de acuerdo al nivel de complejidad que le corresponde responder al estudiante.
2) El estudiante responde correctamente la pregunta.	2.1) El sistema guarda en el historial del estudiante que la pregunta fue respondida correctamente. 2.2) Avance del estudiante en el tablero según el número alcanzado en el lanzamiento de los dados. 2.3) El sistema le permite al jugador resolver la casilla didáctica. 2.4) El sistema le permite al jugador sortear los diferentes tipos de obstáculos. Finaliza el caso de uso Cumplir Reglas Generales del Juego.
3) El estudiante avanza o retrocede en el tablero.	3.1) El sistema le permite mediante y en dependencia del tipo de tablero avanzar o retroceder en el tablero o de marcar la ficha en la casilla correspondiente.
CURSO ALTERNO DE LOS EVENTOS 1	

<table>
<tr><td colspan="2">Viene del paso 2 del curso normal de los eventos.</td></tr>
<tr><td>1) El estudiante no responde correctamente la pregunta.</td><td>1.1) El sistema guarda en el historial del estudiante que la pregunta no fue respondida correctamente.
1.1) El sistema muestra una nueva pregunta de similar nivel de complejidad.
1.2) El sistema muestra un mayor número de preguntas del peor tema.
Se retorna al paso 2 del curso normal de los eventos.</td></tr>
<tr><td colspan="2">CURSO ALTERNO DE LOS EVENTOS 2</td></tr>
<tr><td colspan="2">Viene del paso 2 del curso normal de los eventos.</td></tr>
<tr><td>1) El estudiante no responde correctamente la pregunta en una cantidad de ocasiones igual al número máximo de intentos prefijados por el profesor para preguntas de ese nivel de complejidad.</td><td>1.1) El sistema guarda en el historial del estudiante que la pregunta no fue respondida correctamente.
1.2) El sistema muestra una vía de estudio para vencer las dificultades del estudiante.
1.3) El sistema reinicia el Juego hasta tanto el Jugador no esté listo para volverlo a intentar.
Finaliza el caso de uso Cumplir Reglas Generales del Juego.</td></tr>
<tr><td>Poscondiciones:</td><td>El historial del estudiante en la asignatura queda actualizado en la base de datos.</td></tr>
</table>

Se realizó la presentación del Modelo de Negocio, donde se han descrito los procesos con la especificación de sus actividades y actores involucrados, así como los casos de uso del negocio. Se realizó además el levantamiento de los requisitos funcionales y posteriormente la presentación de los requisitos no funcionales, que constituyen el paso fundamental para adentrarse en este modelo y el correcto funcionamiento del sistema a implementar. Se presentó el Diagrama de Casos de Uso del Negocio y la respectiva descripción de cada uno de los casos de uso que intervienen en el mismo, haciendo evidente la forma en la que quedará estructurado el sistema.

Capítulo 3: Análisis y Diseño del Sistema para la herramienta educativa SMProg

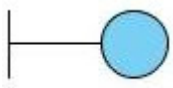
El Análisis y el Diseño es el centro de atención al final de la Fase de Elaboración y el comienzo de las iteraciones de Construcción. Una vez que se analizan y especifican los requisitos del sistema, este flujo de trabajo constituye una de las tres actividades técnicas (diseño, generación de código y pruebas) que se requieren para construir y verificar el software. El Análisis y el Diseño como flujo de trabajo de un software es tanto un Proceso como un Modelo. El "Proceso" de Análisis y el Diseño son una secuencia de pasos que hacen posible que el analista y el diseñador describan todos los aspectos del software que se va a realizar mediante el cual los requisitos se traducen en un "plano" para construir el software.

3.1 Características Fundamentales del Análisis.

Análisis: Consiste en obtener una visión del sistema, sin tomar en cuenta el lenguaje de programación o la plataforma en que se ejecute la aplicación y no precisa como se implementa la herramienta. El Análisis se preocupa solo de ver que hace el sistema, de modo que solo se interesa por los requisitos funcionales, permitiendo estructurar los requisitos de manera que nos facilite su comprensión su modificación y su mantenimiento. (24)

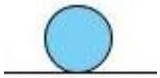
Las clases que se utilizan para el modelado de los diagramas de clases son de tres tipos: Interfaz, Control y Entidad.

Interfaz: Modelan la interacción entre el sistema y sus actores.



Control: Coordinan la realización de uno o unos pocos Casos de Uso coordinando las actividades de los objetos que implementan la funcionalidad del Caso de Uso.

Entidad: Modelan la información que posee larga vida y que es a menudo persistente.



3.1.1 Diagrama de Clases del Análisis

Caso de Uso: Seleccionar Tablero

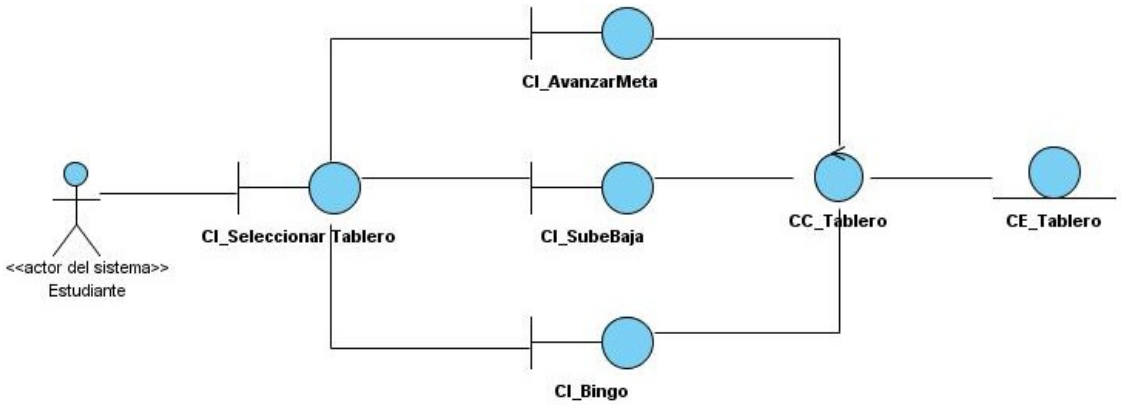


Figura 4: Diagrama de Clases del Análisis del CU Seleccionar Tablero.

Caso de Uso: Cargar Tablero

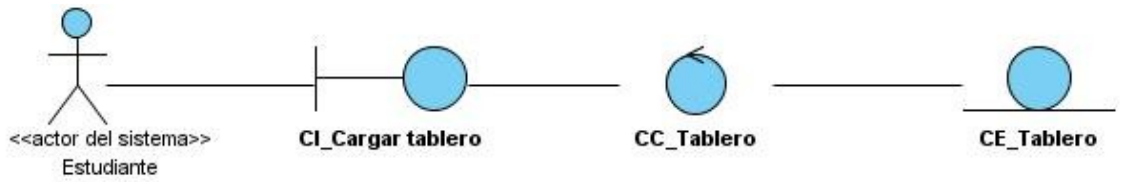


Figura5: Diagrama de Clases del Análisis del CU Cargar Tablero.

Caso de Uso: Jugar en el Tablero

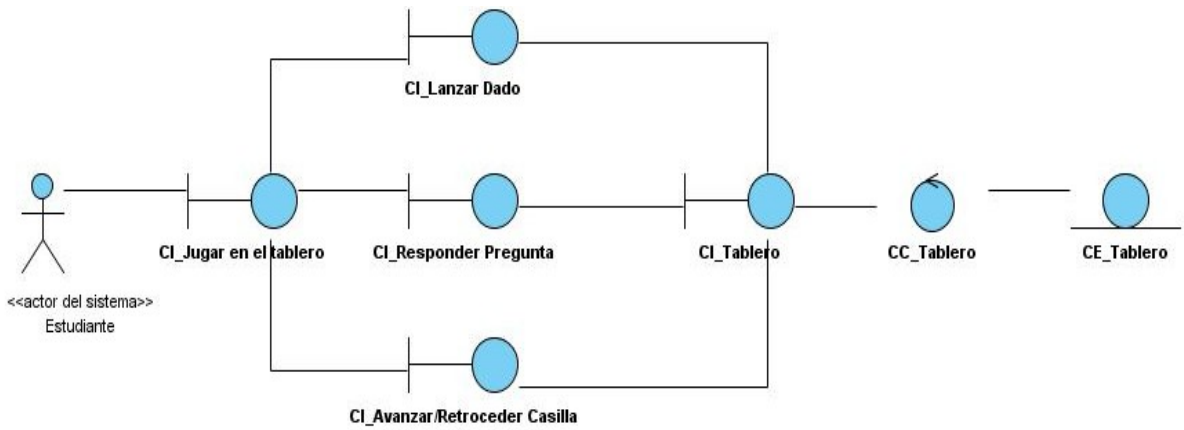


Figura 6: Diagrama de Clases del Análisis del CU Jugar en el Tablero.

Caso de Uso: Cumplir Reglas Generales del Juego.

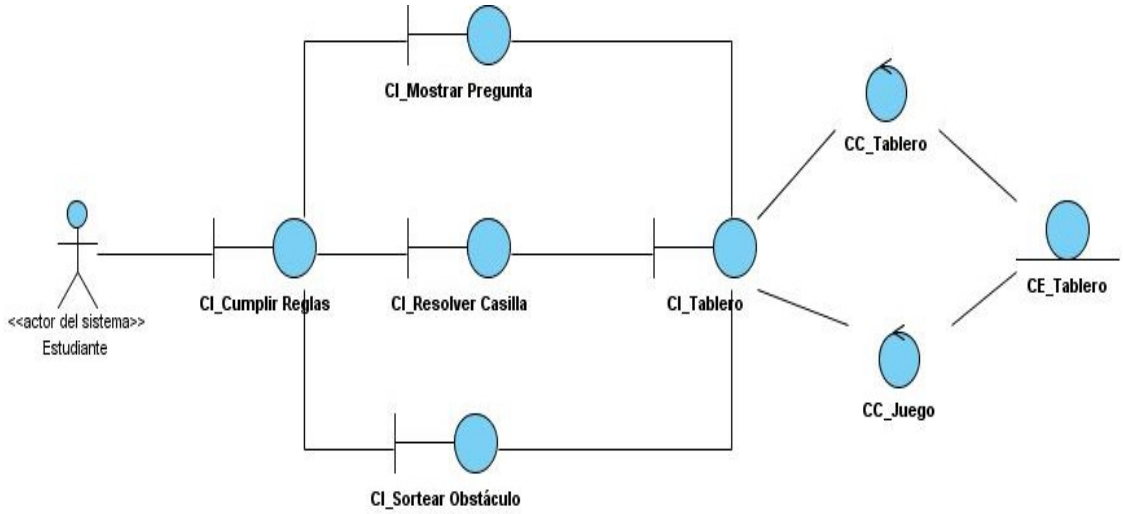


Figura 7: Diagrama de Clases del Análisis del CU Cumplir Reglas Generales del Juego.

3.2 Características Fundamentales del Diseño

Diseño: El modelo de diseño es planteado como un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar, constituyendo una entrada principal en la actividad de implementación. (25)

3.2.1 Diagrama de Colaboración

Caso de Uso: Cumplir Reglas Generales del Juego.

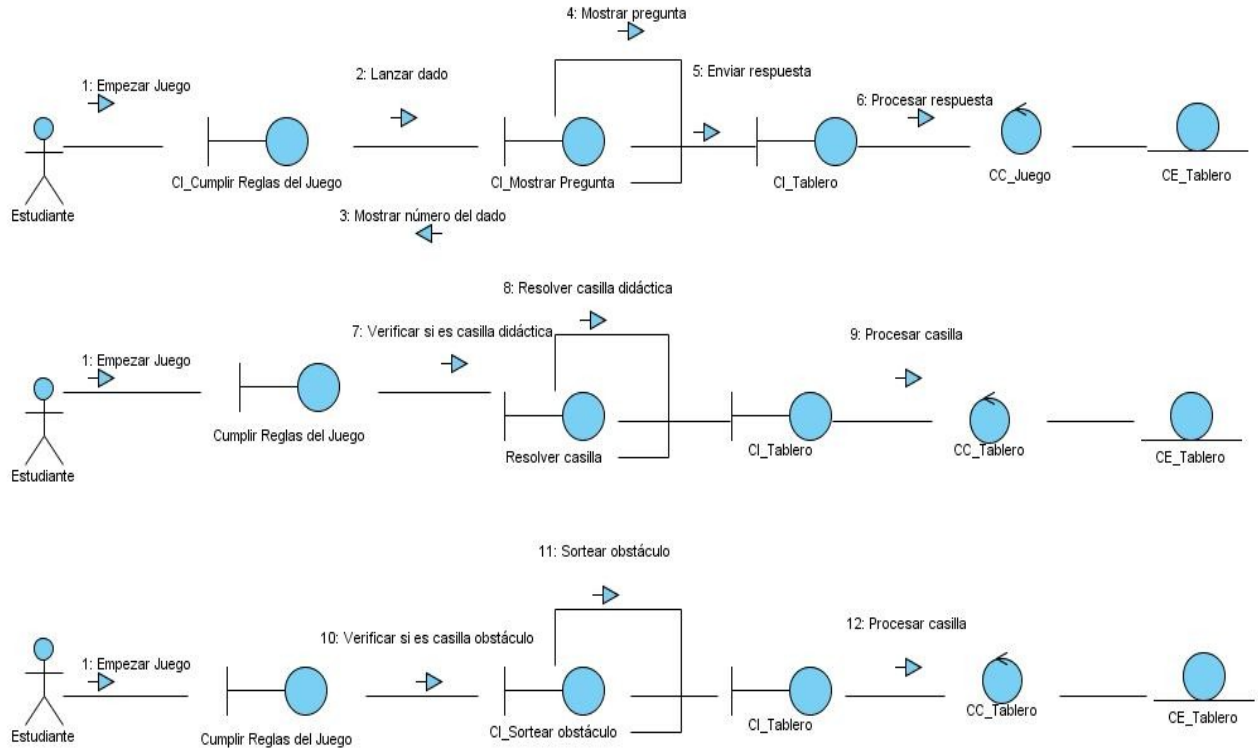


Figura 8: Diagrama de Colaboración del CU Cumplir Reglas Generales del Juego.

Caso de Uso: Jugar en el tablero.

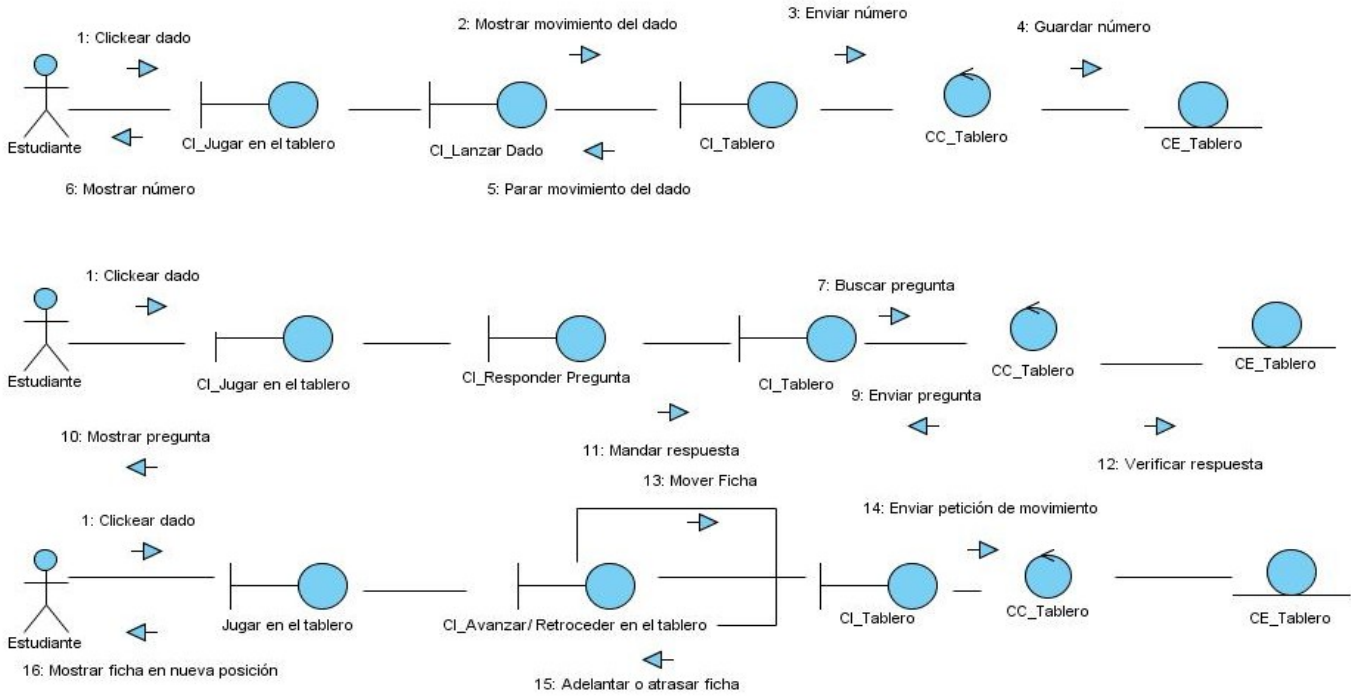


Figura 9: Diagrama de Colaboración del CU Jugar en el tablero.

3.2.2 Diagrama de Clases

El siguiente diagrama muestra las clases que dan respuesta al problema, destacándose entre las mismas las clases: *CasillaVP, DadoVP, FichaVP, Juego, TablaVP y TableroVP;* las cuales forman la columna vertebral del programa por lo que se hace necesario que estén relacionadas entre sí, y con otras clases que le dan terminación al programa añadiéndole valor agregado al mismo, mediante relaciones de tipo “agregación”, “composición” y ”generalización/especialización”.

En el diagrama se representa la clase *TableroVP,* la cual está compuesta por objetos de las clases *TextoVP*, *DadoVP, FichaVP, ImágenesVP, CasillaVP, TablaVP* (o sea, puede existir un tablero sin ninguna instancia de estas clases, que en este caso sería un tablero vacío). Esta clase es quien va a almacenar la mayor cantidad de las funcionalidades de peso de la aplicación y va a tener dependencia funcional fuerte con las clases *AnvanzarMeta, Bingo y SubeBaja*, quienes van a heredar de la clase *Juego* las cuales son imprescindibles en el momento de implementar las funcionalidades que tengan que ver directamente con los diferentes tipos de juego, además de tener

relación con *FichaVP, DadoVP* y *CasillaVP,* que a su vez esta última tiene dependencia las clases *ObstaculoVP y DidacticaVP*, quienes son las encargadas de tratar el comportamiento de las casillas de forma individual según el tipo de tablero escogido por el jugador cuando carga el juego, esto es teniendo en cuenta que el tablero cargado contiene los tipos de juegos AvanzarMeta o SubeBaja que son los responsables de contener el tipo de casilla específico, ya que el Bingo solo presenta casillas de tipo *CasillaVP*.

Se observan también en el diagrama varias relaciones de generalización/especialización principalmente entre las clases de los eventos, tales como la clase Eventos (la cual es una clase genérica) de la cual heredan las clases *EventosTableros, EventosCasilla, EventosDado,* entre otras.

Con el uso de estas clases se detalla la abstracción de la dependencia entre los componentes del tablero creado por la herramienta y los elementos adicionales que intervienen en el mismo para la creación del juego, así como las acciones que se pueden realizar una vez que se carga el juego seleccionado para que el estudiante pueda empezar a interactuar con la aplicación.

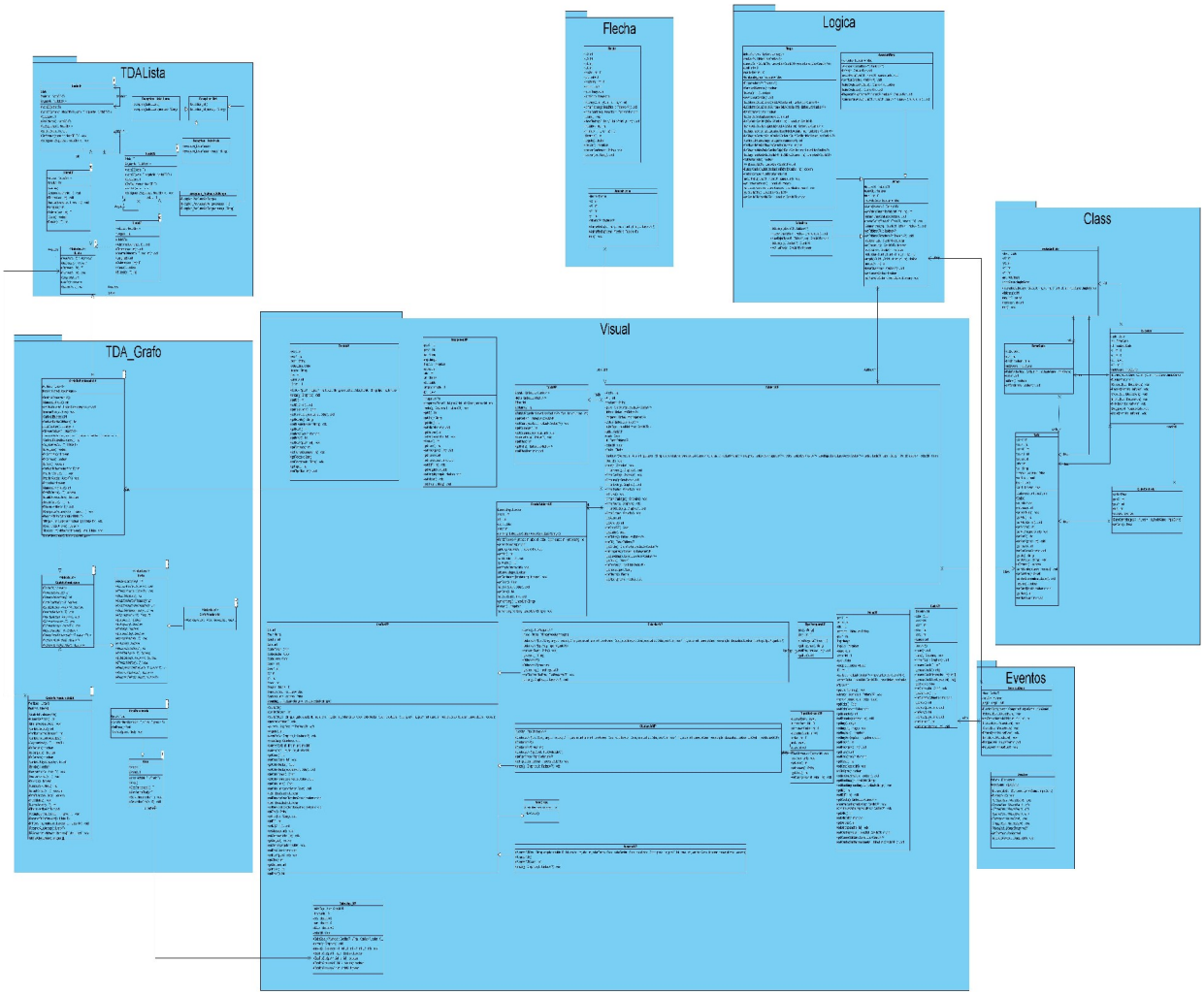


Figura 10: Diagrama de Clases agrupado en paquetes.

3.2.3 Descripción de Clases

A continuación se realiza la descripción detallada de las principales clases que conforman el sistema en orden alfabético, especificando los atributos y algunas operaciones que las mismas realizan.

Tabla 12: Descripción de la clase AvanzarMeta

Nombre: AvanzarMeta: Juego	
Tipo de clase: Controladora	
Atributo	**Tipo**
yamostro	boolean
Atributos heredados del padre: Juego	

num	Int
randObj	Random
tableroVP	TableroVP
Para cada responsabilidad:	
Nombre:	AvanzarMeta(TableroVP tableroVP)
Descripción:	Inicializa los atributos de la clase.
Nombre:	Void irMeta(CasillaVP c)
Descripción:	Mueve la ficha hasta la meta directo.
Nombre:	Void moverFicha(FichaVP fichaVP, int numero)
Descripción:	Este método mueve la ficha y verifica a donde está parada la ficha.
Nombre:	Void tratarObstaculo(CasillaVP c)
Descripción:	Este es el que trata los obstáculos dependiendo del tipo que sean.
Nombre:	Void CaminarParaAtras(FichaVP fichaVP, CasillaVP casillaVP, int numero)
Descripción:	Este es el método para mover la ficha N casillas hacia atras.
Nombre:	Void RegresarSalida(FichaVP fichaVP, CasillaVP casillaVP)
Descripción:	Este es el método para mover la ficha hasta la meta.
Nombre:	Void VerificarCasilla(CasillaVP c)
Descripción:	Este es el método que llama a moverficha() para verificar en que casilla está la ficha.
Nombre:	boolean esDidactica(), boolean esMeta(), boolean esObstaculo(), int genEnteroAleatorio(int rinic,rfin), TableroVP getTableroVP(), void iniciarFichasCasillasdeSalida(), void setTableroVP(TableroVP tableroVP), void tratarDidactica()
Descripción:	Métodos heredados del padre: Juego

Tabla 13: Descripción de la clase Bingo

Nombre: Bingo: Juego	
Tipo de clase: Controladora	
Atributos	**Tipo**
Atributos heredados del padre: Juego	**Tipo**
num	Int
randObj	Random

tableroVP	TableroVP
Para cada responsabilidad:	
Nombre:	Bingo(TableroVP tableroVP)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	boolean buscar(int n)
Descripción:	Aquí se busca el número del método GeneraelNunmero()para saber si está o no en la lista de números.
Nombre:	BuscarCasillaEnListaCasillasConFicha(int idCasilla)
Descripción:	Busca en la lista de casillas con fichas si la casilla que le pasas está dentro de esta lista.
Nombre:	Int CantidaddeCasillaDiagonalDerechaI(int numero)
Descripción:	Se le entra el id de una casilla y devuelve cuántas casillas tiene que contar para esa diagonal.
Nombre:	Int CantidaddeCasillaDiagonalIzquiedaI(int numero)
Descripción:	Recibe como parámetro un id de una casilla y devuelve cuantas casillas tiene que contar para esa diogonal.
Nombre:	Boolean EstaB(int numero)
Descripción:	usca en que tramo de filas está el número que le paso por parámetros.
Nombre:	CasillasConcecutivasParaGanarI()
Descripción:	Método que devuelve la cantidad de casillas consecutivas necesarias para poder ganar. Toma el menor de las columnas y las filas.
Nombre:	Boolean GeneraelNunmero()
Descripción:	A partir del número que genera el método genEnteroAleatorio lo ubica en la lista de Enteros.
Nombre:	Int LaCantDeCasillasBajar(int numero)
Descripción:	Devuelve la cantidad de casillas que faltan para llegar al final de la fila en dependencia del número pasado por parámetro.
Nombre:	Boolean LaCasillaTieneUnaFichaB(CasillaVP casillaVP)
Descripción:	Recibe una casilla y devuelve si tiene una ficha o no.
Nombre:	LinkedList<CasillaVP> LaColumnaDeLaCasillaX(int IdDeLACasilla)
Descripción:	Recibe un id de una casilla X y retorna la columna completa para esa casilla.
Nombre:	LinkedList<CasillaVP> LaDiagonaldeDerechaDeLaCasillaX(int IdDeLACasilla)

Descripción:	Este método es el que devuelve una lista de Casillas de la Diagonal de derecha a izquierda.
Nombre:	LinkedList<CasillaVP> LaDiagonaldeIzqDeLaCasillaX(int IdDeLACasilla)
Descripción:	Este método es el que devuelve una lista de Casillas de la Diagonal de Izquierda a derecha.
Nombre:	LinkedList<CasillaVP> LaFilaDeLaCasillaX(int IdDeLACasilla)
Descripción:	Recibe un id de una ficha X y retorna la fila completa para esa ficha.
Nombre:	Void moverFicha(FichaVP fichaVP, int numero)
Descripción:	Mueve la ficha de casilla, después de pasarle la ficha y el número de la casilla para donde va.
Nombre:	Void seleccionarCasilla()
Descripción:	Pinta la ficha en las casillas que lanza el número random.
Nombre:	Win(java.util.LinkedList<CasillaVP> listcasillaVPs)
Descripción:	Recibe una lista de casillas y devuelve si las casillas de esa lista están o no en la lista de casillasConFichas.
Nombre:	Boolean YaHasGanado()
Descripción:	Este es el método que determina si el estudiante ganó o no.

Tabla 14: Descripción de la clase CasillaVP

Nombre: CasillaVP	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Alto	int
Ancho	int
Angulo	double
banderaCaso	boolean
banderaLetra	boolean
Caso	int
colorBorder	java.awt.Color
colorFondo	java.awt.Color
colorLetra	java.awt.Color
ID	int
nombImg	LinkedList<java.lang.String>

posX	Int
posY	int
Tex	java.lang.String
X	Int
Y	int
Para cada responsabilidad:	
Nombre:	CasillaVP()
Descripción:	Inicializa los atributos de la clase.
Nombre:	CasillaVP(int caso)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	CasillaVP(java.lang.String tex, double angulo, int ID, int ancho, int alto, java.awt.Color colorFondo, java.awt.Color colorBorder, java.awt.Color colorLetra, int posX, int posY, int caso, boolean banderaCaso, boolean booolean banderaLetra)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	Double Angulo()
Descripción:	Devuelve en una variable entera el ángulo donde está ubicada la casilla.
Nombre:	Int getColorBorder()
Descripción:	Obtiene el color del borde de la casilla.
Nombre:	Int getColorFondo()
Descripción:	Obtiene el color del fondo de la casilla.
Nombre:	Int getID()
Descripción:	Obtiene el identificador de la casilla.
Nombre:	Int getPosX()
Descripción:	Obtiene la posición X de la casilla.
Nombre:	Int getPosY()
Descripción:	Obtiene la posición Y de la casilla.
Nombre:	Int[] obtenerX(int ini, int fin, int an)
Descripción:	Devuelve un arreglo con la coordenada x de la casilla.
Nombre:	Int[] obtenerY(int ini, int fin, int al)
Descripción:	Devuelve un arreglo con la coordenada y de la casilla.
Nombre:	Void pintar(java.awt.Graphics g, TableroVP t)
Descripción:	Pinta la casilla.

Tabla 15: Descripción de la clase DadoVP

Nombre: DadoVP	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Num	int
Para cada responsabilidad:	
Nombre:	DadoVP()
Descripción:	Inicializa los atributos de la clase.
Nombre:	int generarCordX()
Descripción:	Devuelve en una variable entera la coordenada X del dado.
Nombre:	int generarCordY()
Descripción:	Devuelve en una variable entera la coordenada Y del dado.
Nombre:	int generarCordXfrente(int des)
Descripción:	Devuelve en una variable entera la coordenada X frontal del dado.
Nombre:	int generarCordYfrente(int des)
Descripción:	Devuelve en una variable entera la coordenada Y frontal del dado.
Nombre:	Void paint(java.awt.Graphics g)
Descripción:	Este es el método para pintarle los puntos al dado.
Nombre:	pintar3D(java.awt.Graphics g1)
Descripción:	Método para pintar el resto del dado.

Tabla 24: Descripción de la clase TableroVP

Nombre: TableroVP	
Tipo de clase: Controladora	
Atributos	**Tipo**
dado	Dado
Para cada responsabilidad:	
Nombre:	TableroVP(int Ancho, int Alto, java.lang.String tipoJuego, GrafoNoPonderadoLA<CasillaVP> grafo, java.util.LinkedList<FichaVP> fichas, java.util.LinkedList<ImagenesVP> imagenes, java.util.LinkedList<TextoVP> textos, java.util.LinkedList<Arco<CasillaVP>> subeBaja, TablaVP tabla, Dado

	dado, FondoTableroVP ft, java.awt.Color colorSB)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	Int getAlto()
Descripción:	Obtiene el alto de la tabla.
Nombre:	Int getAncho
Descripción:	Obtiene el ancho de la tabla.
Nombre:	Dado getDado()
Descripción:	Obtiene el dado.
Nombre:	LinkedList<FichaVP> getFichas()
Descripción:	Obtiene la lista de fichas.
Nombre:	FondoTableroVP getFt()
Descripción:	Obtiene el fondo del tablero.
Nombre:	GrafoNoPonderadoLA<CasillaVP> getGrafo()
Descripción:	Obtiene la lista de adyacencias de las casillas.
Nombre:	LinkedList<ImagenesVP> getImagenes()
Descripción:	Obtiene la lista de imágenes.
Nombre:	LinkedList<Arco<CasillaVP>> getSubeBaja()
Descripción:	Obtiene la lista de arcos del juego SubeBaja.
Nombre:	TablaVP getTabla()
Descripción:	Obtiene la tabla.
Nombre:	LinkedList<TextoVP> getTextos()
Descripción:	Obtiene los textos de las casillas.
Nombre:	String getTipoJuego()
Descripción:	Obtiene el tipo de juego.
Nombre:	Void InitTabla()
Descripción:	Inicializa la tabla.
Nombre:	Void paint(java.awt.Graphics g)
Descripción:	Método para pintar.
Nombre:	Void PintarCasilla(java.awt.Graphics g)
Descripción:	Pinta la casilla.
Nombre:	Void PintarDado(java.awt.Graphics g)
Descripción:	Pinta el dado.
Nombre:	Void PintarFicha(java.awt.Graphics g)

Descripción:	Pinta la ficha.
Nombre:	Void pintarFichaBingo(java.awt.Graphics g)
Descripción:	Pinta la ficha del Bingo.
Nombre:	Void PintarFondo(java.awt.Graphics g)
Descripción:	Pinta el fondo de la tabla.
Nombre:	Void PintarImg(java.awt.Graphics g)
Descripción:	Pinta la imagen.
Nombre:	Void PintarSubeBaja(java.awt.Graphics g)
Descripción:	Pinta la casilla SubeBaja.
Nombre:	Void PintarTabla(java.awt.Graphics g)
Descripción:	Pinta la tabla.
Nombre:	Void PintarTexto(java.awt.Graphics g)
Descripción:	Pinta el texto.

3.3 Arquitectura de la herramienta

RUP asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. (26)

La arquitectura constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes facilitando así la comunicación entre todas las partes implicadas en el desarrollo del software.

La Arquitectura seleccionada para la realización de esta herramienta es Cliente – Servidor, que no es más que un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan. Aquí definimos cuatro aspectos fundamentales que se describen a continuación:

1. Vista de Casos de Uso significativos.
2. Vista Lógica.
3. Vista de Despliegue.
4. Vista de Implementación.

3.3.1 Vista de Casos de Uso significativos

Entre los Casos de Uso más significativos, definidos por su complejidad y trabajo al realizarlos en el desarrollo de la aplicación tenemos:

1. Jugar en el tablero.
2. Cumplir Reglas Generales del Juego.

A continuación se muestra una figura con la dependencia existente entre el actor quien es el que inicia los procesos y los casos de uso.

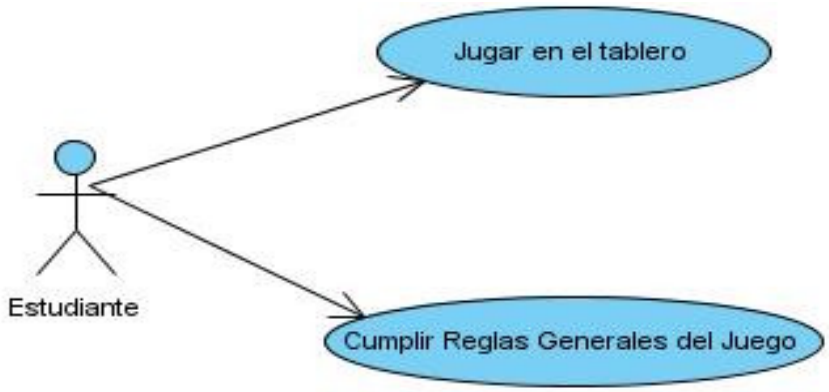


Figura 11. Diagrama de Casos de Usos más significativos arquitectónicamente.

De acuerdo al flujo de tareas que se desglosan en cada uno de los Casos de Uso descritos se tienen en cuenta una serie de aspectos que van a resultar imprescindibles con la implementación de la Lógica de Juego, donde dan a luz una serie de componentes y funcionalidades necesarias para darle respuesta a los requisitos definidos, por eso es que se escogen estos dos casos de uso como los más significativos arquitectónicamente.

3.3.2 Vista Lógica

Debido a la utilización de la arquitectura Cliente – Servidor se debe tener en cuenta que siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y los servidores en procesadores departamentales o de grupo.

Los clientes realizan generalmente funciones como:

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con redes de área local o extensa (1).

Entre las clases más importantes definidas por la arquitectura están las que interactúan directamente con el usuario, quienes van a ser las clases de interfaz, agrupadas en paquetes y desde las cuales se ven las dependencias funcionales que tienen entre ellas. A continuación se muestra un diagrama con las principales clases agrupadas en paquetes y sus dependencias.

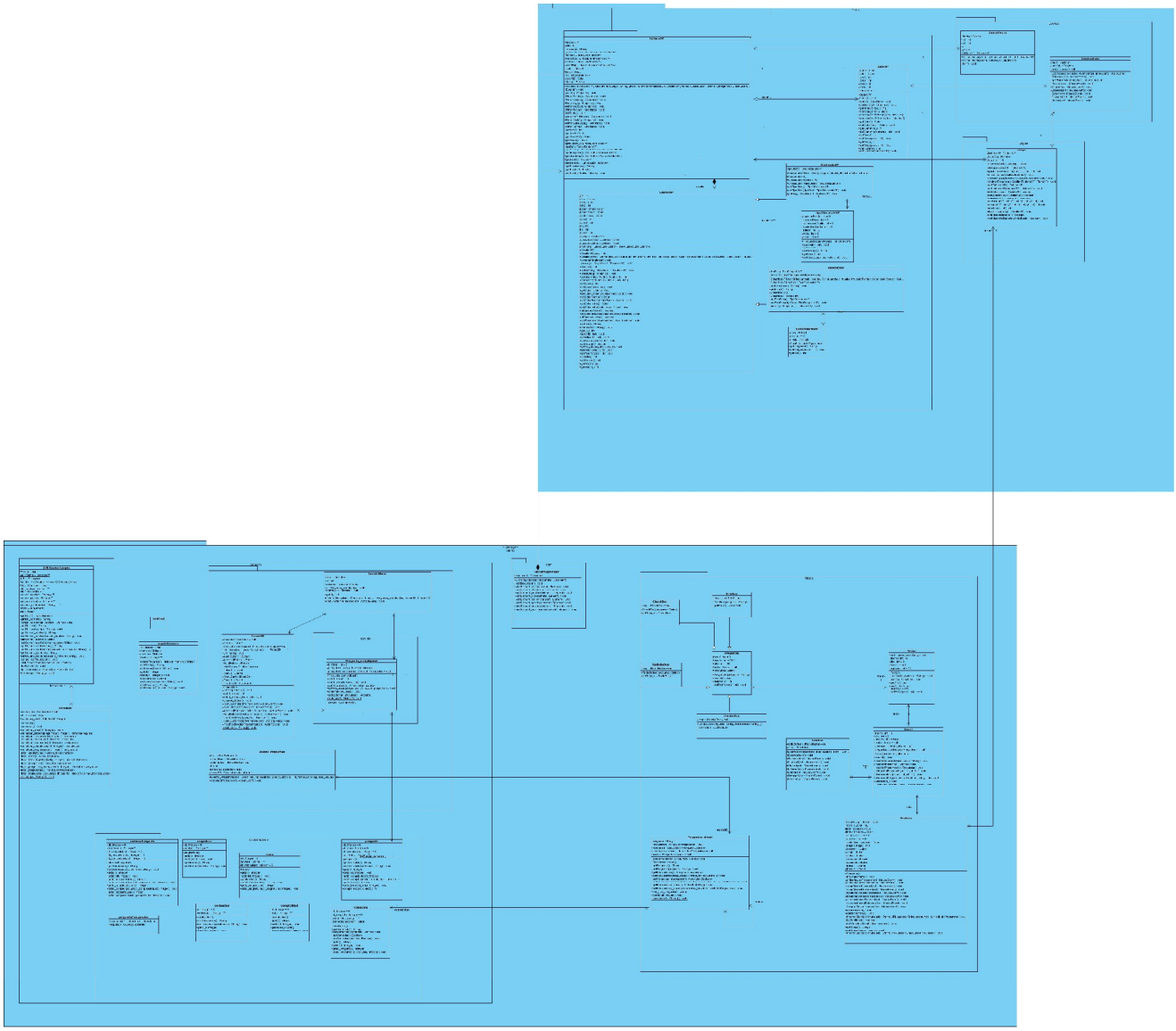


Figura 12. Diagrama de Clases más significativas agrupadas en paquetes según arquitectura Cliente - Servidor.

En el diagrama se representan las clases agrupadas en paquetes siguiendo la arquitectura trazada para la solución propuesta. En un paquete aparecen las clases que intervienen con el Cliente y en el otro paquete las clases que intervienen con el servidor y las relaciones que existen entre los dos paquetes y las clases que los componen.

3.3.3 Vista de Despliegue

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, así como las conexiones entre estos elementos en el sistema. El

modelo consiste en uno o más nodos, dispositivos y conectores, entre estos. El modelo de despliegue también mapea procesos dentro de estos elementos de procesamiento, permitiendo la distribución del comportamiento a través de los nodos que son representados. A continuación se muestra el diagrama de despliegue modelado para la aplicación a desarrollar.

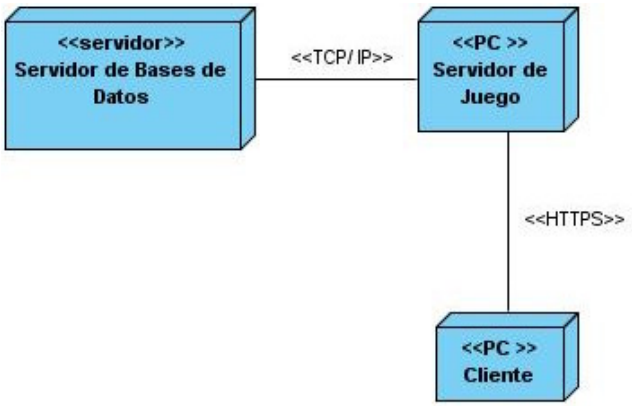


Figura 13. Diagrama de Despliegue.

Aquí se define en su conjunto cómo el usuario que va a estar en la PC Cliente utilizando el Juego Didáctico se conectará mediante conexión web segura *https,* una vez que el módulo esté integrado al *EVA-MOODLE*, para acceder al servidor donde va a estar el fichero que posibilita cargar el juego. Existe además la conexión mediante el protocolo *TCP/IP* con una base de datos que va a ser la responsable de mostrar las preguntas que correspondan con el tipo de casilla independientemente del tipo de tablero que sea cargado por parte del usuario que esté interactuando con el sistema.

3.3.4 Vista de Implementación

En la vista de Implementación nos enfocamos fundamentalmente en el diagrama de componentes, que siguiendo la arquitectura definida vemos que: acorde a RUP este es el modelo encargado de mostrar la estructura de alto nivel del Modelo de Implementación, específicamente los Subsistemas de Implementación y sus dependencias de importación. También se usan para mostrar ficheros de código fuente y sus dependencias de compilación; ficheros de aplicación y sus dependencias en tiempo de ejecución; relaciones derivadas entre ficheros de código fuente y ficheros resultantes de la compilación. (27)

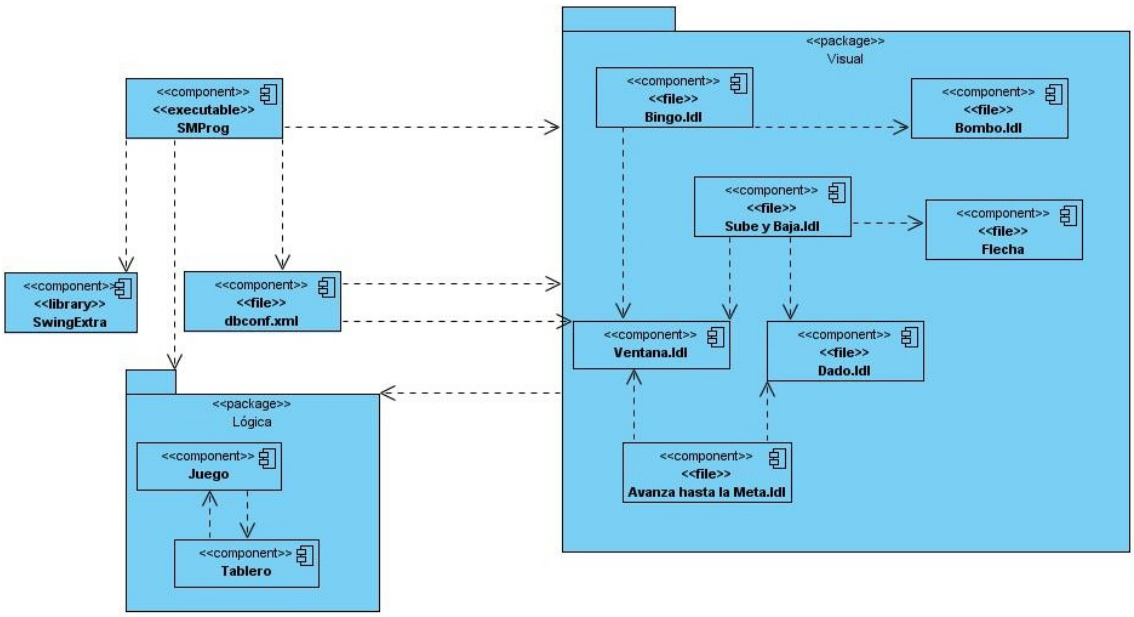


Figura 14. Diagrama de Componentes.

En el diagrama presentado aparecen los componentes y paquetes imprescindibles que intervienen en todo el proceso, donde el ejecutable es quien domina al resto de las clases, pues es quien las enlaza a todas.

El ejecutable SMProg es el componente con el que va a interactuar el cliente y que va a tener dependencias con los demás paquetes que integran el diagrama y la librería SwingExtra.

Los paquetes de Lógica y Visual son los contenedores de las clases que intervienen en los tipos de tableros, en el primer paquete se ven como las dependencias entre las clases Juego y Tablero y en el segundo paquete se ven como los 3 tipos de tableros utilizan el componente ventana que a su vez esta se relaciona con el fichero (dbconf) que carga las consultas de la base de datos. Aquí también se evidencia la presencia del dado que es utilizado por los tipos de tableros Sube y Baja y Avanza hasta la Meta, así como la flecha es solamente la usa el tipo de tablero Suba y Baja. El tipo de juego Bingo maneja solamente el bombo.

3.4 Tratamiento de Errores

Es necesario que la interacción del sistema con el usuario sea confiable y segura, para ello se lleva a cabo el tratamiento de errores para detectar los posibles fallos que se puedan encontrar en el sistema. Siempre que ocurra un error o se lance una

excepción el sistema notificará inmediatamente al usuario. La notificación de los errores se realiza mediante mensajes de texto para el usuario, de tal modo que este entienda la naturaleza del error y pueda corregirlo en el próximo intento de juego. Otra de las alternativas del sistema para tratar los errores es la validación de la entrada de datos, verificando que los mismos sean del tipo de dato correcto, pero esto hace un poco engorroso el sistema y aumenta la complejidad temporal de los métodos, por lo que el tratamiento de errores va a ser preferentemente mediante mensajes de texto al usuario y lanzamiento de errores por parte del sistema.

3.5 Interfaz

La interfaz del sistema es sencilla y fácil de entender. La aplicación le permite al estudiante seleccionar el tablero con el tipo de juego (AvanzarMeta, Bingo o SubeBaja) que se desea cargar, para que posteriormente el estudiante pueda interactuar con el mismo y empiece a jugar.

Una vez cargado el tablero con el juego (AvanzarMeta, Bingo o SubeBaja), el sistema activa la opción de lanzar el dado si es un tablero del tipo AvanzarMeta o SubeBaja. Una vez cargado el juego debe mostrar los diferentes tipos de casillas (Obstáculo y Didácticas para AvanzarMeta y solo Didáctica para SubeBaja y Bingo), además de las fichas que representarán a los jugadores y establecer la conexión entre las casillas definiendo así el camino por el que se desplazará el jugador.

El juego cuenta con un distintos fondos de tableros que van a ser diseñados por el profesor, imprescindibles para cargar los tableros de los juegos con los que van a interactuar los estudiantes una vez que accedan a la aplicación.

En el juego, dependiendo del tipo de tablero, se va a poder lanzar el dado con el cual, dependiendo del tipo de la casilla en la que se encuentre el jugador, se podrá avanzar, retroceder o regresar a la meta o ir directo a la salida. Las casillas que van a tener tratamiento diferenciado se van a distinguir de las casillas normales en las que no hay que responder preguntas por una imagen particular que define el comportamiento de la misma. En el bingo las fichas que van a ocupar las casillas van a estar etiquetadas de acuerdo a la veracidad de la respuesta dada por el estudiante, sabiendo este cuando pueda ganar de acuerdo a la cantidad de casillas consecutivas señaladas, ya sea vertical, horizontal o diagonal, ya sea inversa o reversa.
A continuación algunas figuras que muestran la interfaz visual de la herramienta.

Figura 36. Avanza hasta la Meta.

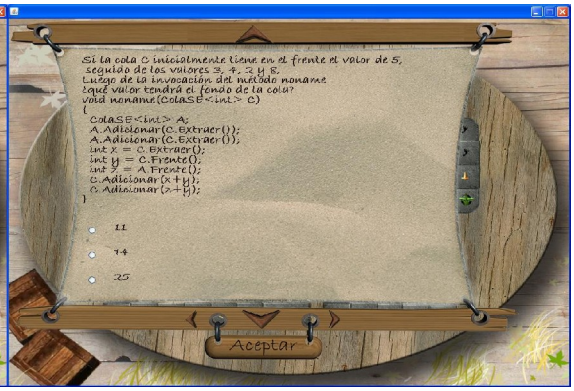


Figura 37. Casilla obstáculo o didáctica.

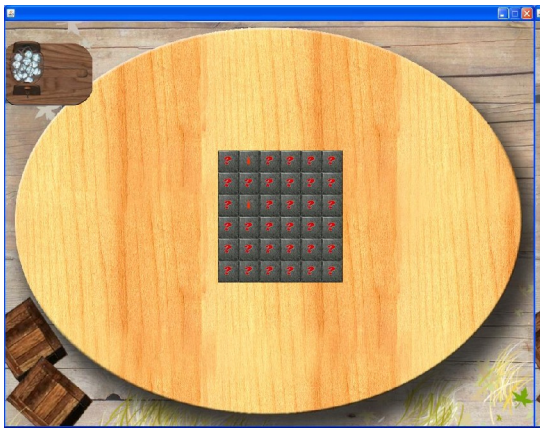


Figura 38. Bingo.

Figura 39. Sube y Baja.

Durante el desarrollo de este capítulo se ha visto el resultado de la Disciplina de Análisis y el Diseño en el proceso de desarrollo de la implementación en la “Lógica de Juego”, donde RUP propone una serie de flujos de trabajo y artefactos que se han ido viendo en los diagramas de clases del Análisis, acorde a los casos de uso del sistema, los diagramas de interacción entre el usuario y el sistema, el diagrama de Clases y las descripciones textuales de las clases más importantes que intervienen en el proceso, se realiza además, una breve descripción de cómo se tratan los errores y cómo interactúa el usuario con el sistema.

Capítulo 4. Implementación y Pruebas del Sistema

Según RUP, la Implementación es el centro de la Fase de Construcción, aunque también se lleva a cabo durante la Fase de Elaboración para crear la línea base ejecutable de la arquitectura y durante la de Transición para tratar defectos que se hayan encontrado en ese momento, en caso de que existan.

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes, es decir, ficheros de código fuente o binario, scripts, ejecutables o similares. Aquí se implementan las clases elaboradas durante el Diseño y se muestra su organización.

4.1 Componentes Desarrollados

Se cuenta con una serie de componentes visuales y no visuales que facilitan la programación, realizando todas las restricciones y validaciones en tiempo de diseño de la aplicación. En tiempo de ejecución estas validaciones se verifican de forma autónoma por los componentes para que sean cumplidas, lo que facilita al programador la posibilidad de sólo dedicarse a la programación de las funcionalidades, olvidándose por completo de las validaciones de aquellos datos que pudieran ser entrados por el usuario.

Bombo: Permite que el estudiante pueda seleccionar una pelota con el número de la casilla que va a ocupar en el tablero.

RadioButton: Permite validar antes de ejecutar el evento si se cumplen las validaciones de los componentes de entrada.

CheckBox: Encargado de la entrada de parámetros booleanos, verdadero o falso.

Textbox: Permite la entrada de datos por parte del usuario como cadenas (string) y números (int16, int32, int 64, double, decimal).

Dado: Permite que el estudiante pueda lanzar el dado para obtener un número con la cantidad de casillas a mover la ficha.

Flecha: Permite indicar al jugador hacia qué casilla va a estar dirigida la ficha después de responder la pregunta acorde al tipo de casilla.

Ficha: Permite indicar al jugador en qué posición del tablero se encuentra de acuerdo al tipo de juego.

Ventana: Permite visualizar al jugador la pregunta afín con el tipo de casilla correspondiente. Este componente contiene a su vez a **RadioButton, TextBox** y **Checkbox.**

4.2 Estándares de Codificación

Los Estándares de Codificación son reglas específicas a un lenguaje que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Durante el desarrollo de un software estos estándares ayudan a los ingenieros a producir un código de alta calidad y a entender y a utilizar el código de sus colegas. Pero también realzan considerablemente la capacidad de mantenimiento y reusabilidad a largo plazo del producto final.(1)

Con esto se reduce la posibilidad de introducir errores en el código haciéndolo más legible y fácil de leer ayudando así al posterior mantenimiento del software.

4.2.1 Reglas a seguir

En el momento de la programación existen ciertas reglas y estándares a seguir para un buen uso de la programación

1. Los nombres de las clases deben ser sustantivos o frases de sustantivos y no se deben usar prefijos.

2. Para las clases interfaces, utilizar sustantivos, frases en forma sustantiva o adjetiva que describan comportamiento.

3. Todas las clases deben tener prefijos o sufijos que las identifiquen y seguidos de una letra mayúscula que sería la primera del nombre de la clase.

4. En los enumerados se recomienda no poner prefijos (sufijos) a los valores ni al identificador, el cual tiene que estar en singular.

5. Para los campos de solo lectura y constantes se deben utilizar sustantivos o frases de éstos.

6. Para los parámetros y campos no constantes se deben usar nombres descriptivos que deben ser lo suficientemente explícitos para determinar el significado de la variable y su tipo, preferentemente su significado.

7. Utilizar i, j, k, l, m, n para contadores en ciclos triviales, en caso contrario usar nombres de variables más descriptivos.

8. Los métodos deben ser nombrados con verbos o frases verbales.

9. Las propiedades deben ser nombradas utilizando sustantivos o frases en forma sustantiva considerando siempre que sean nombradas con el mismo nombre de su tipo.

10. Nombrar los manejadores de eventos con el sufijo *Eventos* y las clases que sean para pasar argumentos con *Eventos* + "Nombre de la clase" Ej. *EventosDado*.

11. No hacer público ninguna instancia o campo de una clase, deben ser privados o protegidos según la dependencia de la clase en cuestión.

12. Excepto el código relacionado con la interfaz gráfica de la aplicación, todos los nombres de variables y campos deben tener al principio la abreviatura del tipo dado.

13. Todas las declaraciones de funcionalidades se deben hacer preferentemente en idioma Español para un mejor entendimiento y utilizando el estilo Pascal que capitaliza la primera letra de cada palabra; descontado los campos protegidos y privados, las variables locales y los parámetros, que deben ser en estilo Camello, que capitaliza la primera letra de cada palabra, exceptuando para la primera palabra.

4.3 Estándares de Interfaz

Para el adecuado diseño de la aplicación "desktop" a implementar se proponen seguir una series de estándares que van encaminados a garantizar la consistencia de este. Estos estándares se definen a continuación:

- Brindar una interfaz sencilla, de manera tal que cualquier persona con un mínimo dominio de la computación pueda aprender a trabajar con la aplicación.

- Garantizar la legibilidad de manera que exista contraste de los colores de los textos con el fondo del tablero específico y el tamaño de la fuente sea lo suficientemente adecuado a la vista del usuario.

- Mostrar al usuario solamente aquellas opciones a las que tiene derecho a acceder.

4.4 Pruebas al sistema

Probar es imprescindible para verificar el funcionamiento y la calidad de un software. La prueba es un proceso de ejecución de un programa con la intención de comprobar que el producto satisface los requerimientos y se comporta como se desea, por eso cualquier producto de ingeniería puede ser probado de una de estas formas:

1. Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.

2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que “todas las piezas encajen”, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

4.4.1 Pruebas de caja blanca

Las técnicas de caja blanca o técnicas estructurales se basan en un minucioso exámen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. Los pasos a realizar para aplicar esta técnica son:

- Representar el programa en un grafo de flujo.

- Calcular la complejidad ciclomática.

- Determinar el conjunto básico de caminos independientes.

- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

4.4.2 Pruebas de caja negra.

Las técnicas de caja negra o funcionales son las que se realizan sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba.

La Prueba de Caja Negra no es una alternativa a las técnicas de Prueba de Caja Blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de Caja Blanca (1).

4.5 Casos de Pruebas principales

Luego de tener las clases válidas e inválidas definidas, se procedió a definir los Casos de Prueba, para los Casos de Uso más significativos que se encuentran organizados en tres unidades diferentes para los juegos de tipo Avanzar hasta la Meta y Sube y Baja:

1. Seleccionar el Tablero.
2. Jugar en el tablero.
3. Cumplir Reglas del Juego.

4.5.1 Seleccionar Tablero

A este caso de prueba se le hicieron pruebas de selección de tema y cargado de tablero.

CPR 1: Seleccionar Tema

Flujo Central:

1. El estudiante ordena Seleccionar Tema.
2. El sistema muestra la Interfaz Seleccionar Tema para seleccionar el tipo de tema a escoger.
3. El estudiante selecciona el tema o los temas con los que va a efectuar el juego.
4. El sistema muestra interfaz con los diferentes temas disponibles.

5. El estudiante ordena aceptar la operación.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
La opción Seleccionar tema está habilitada.		El sistema muestra en una ventana los tipos de tablero a seleccionar.	Satisfactorio	
El tipo o los tipos de tema determinado se pueden seleccionar.		El sistema muestra la selección del o los temas especificados.	Satisfactorio	
La opción Aceptar esta habilitada.		El sistema permite seleccionar la opción Aceptar.	Satisfactorio	
La opción Cancelar esta habilitada.		El sistema permite seleccionar la opción Cancelar.	Satisfactorio	
El sistema guarda los cambios y cierra la ventana cuando se oprime el botón Aceptar.		El sistema guarda los cambios y cierra la ventana.	Satisfactorio	.
El sistema cierra la ventana cuando se oprime el botón Cancelar.		El sistema cierra la ventana.	Satisfactorio	

CPR 2: Cargar Tablero

Flujo Central:

1. El estudiante ordena Seleccionar el Tablero.

2. El sistema muestra la Interfaz Seleccionar el Tablero para seleccionar el tipo de tablero a escoger.
3. El estudiante selecciona el tipo de tablero (AvanzaMeta, Bingo, SubeBaja).
4. El responsable ordena aceptar la operación.
5. El estudiante ordena Cargar Tablero seleccionado.
6. El estudiante ordena aceptar la operación.
7. El sistema acepta la operación mostrando el tipo de tablero seleccionado.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
La opción Seleccionar tablero está habilitada.		El sistema muestra en una ventana los tipos de tablero a seleccionar.	Satisfactorio	
El tipo de tablero determinado se puede seleccionar.		El sistema muestra la selección del tablero especificado.	Satisfactorio	
La opción Aceptar esta habilitada.		El sistema permite seleccionar la opción Aceptar.	Satisfactorio	
La opción Cancelar esta habilitada.		El sistema permite seleccionar la opción Cancelar.	Satisfactorio	
El sistema guarda los cambios y cierra la ventana cuando se oprime el botón Aceptar.		El sistema guarda los cambios y cierra la ventana.	Satisfactorio	.
El sistema cierra la ventana cuando se		El sistema cierra la ventana.	Satisfactorio	

oprime el botón Cancelar.				

4.5.2 Jugar en el tablero

A este caso de prueba se le hicieron pruebas de Lanzar Dado, Recoger ficha del Bombo, Mostrar y Responder Pregunta y Avanzar o Retroceder en el tablero.

CPR 1: Lanzar Dado

Flujo Central:

1. El estudiante ordena Lanzar Dado.
2. El sistema muestra en la Interfaz el dado moviéndose.
3. El sistema muestra interfaz con la posición en la que paró el dado.
4. El sistema muestra interfaz con el número que generó el lanzamiento del dado.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El dado está habilitado.		El sistema muestra el dado	Satisfactorio	
	El dado no está en el tablero.	El sistema ha cargado un tablero que no utiliza dado (Bingo).	Satisfactorio	
El dado se encuentra en movimiento.		El sistema muestra el movimiento del dado en el tablero.	Satisfactorio	
El dado para el movimiento.		El sistema muestra el dado con el número.	Satisfactorio	

CPR 2: Recoger ficha del bombo.

Flujo Central:

1. El estudiante ordena Recoger ficha del bombo.
2. El sistema muestra en la Interfaz el bombo moviéndose.
3. El sistema muestra interfaz con la pelota que salió del bombo.
4. El sistema muestra interfaz con el número que generó el lanzamiento del bombo.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El bombo está habilitado.		El sistema muestra el bombo.	Satisfactorio	
	El bombo no está en el tablero.	El sistema ha cargado un tablero que no utiliza bombo (Avanzar hasta la Meta o Sube y Baja).	Satisfactorio	
El bombo se encuentra en movimiento.		El sistema muestra el movimiento del bombo.	Satisfactorio	
El bombo para el movimiento.		El sistema muestra la pelota que salió del bombo con el número.	Satisfactorio	

CPR 3: Mostrar / Responder Pregunta.

Flujo Central:

1. El sistema muestra en la Interfaz una pantalla con una pregunta que corresponda acorde con la casilla en la que se encuentre la ficha.
2. El estudiante responde en la Interfaz nueva la pregunta.

3. El estudiante envía la respuesta presionando el botón Aceptar.
4. El sistema muestra interfaz anterior sin la pantalla de la pregunta.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
La pantalla está en el tablero		El sistema muestra la pantalla.	Satisfactorio	
La pantalla muestra la pregunta		El sistema muestra la pregunta en la pantalla.	Satisfactorio	
Se puede responder en la pantalla.		El sistema muestra la o las posibles respuestas a la pregunta.	Satisfactorio	
El botón Aceptar está habilitado.		El sistema permite presionar el botón Aceptar y cierra la ventana.	Satisfactorio	

CPR 4: Avanzar / Retroceder en el tablero.

Flujo Central:

1. El sistema muestra en la Interfaz el movimiento de la ficha.
2. El estudiante visualiza el movimiento de la ficha.
3. El sistema muestra en la Interfaz la ficha en la nueva casilla.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
La ficha está en la casilla		La ficha se ve en la casilla del	Satisfactorio	

		tablero.		
La ficha se traslada de casilla.		Se ve el movimiento de la ficha en el tablero.	Satisfactorio	
La ficha retrocede su posición en dependencia del tipo de casilla.		Si la casilla es de tipo obstáculo y la condición de esa casilla es retroceder, entonces la ficha retrocede de casilla.	Satisfactorio	
La ficha avanza su posición en dependencia del tipo de casilla.		Si la casilla es de tipo didáctica y responde correctamente la pregunta o es de tipo obstáculo y la condición de esa casilla es avanzar, entonces la ficha avanza de casilla.	Satisfactorio	

4.5.3 Cumplir Reglas del juego

A este caso de prueba se le hicieron pruebas de Resolver Casilla y Sortear Obstáculo.

CPR 1: Resolver Casilla.

Flujo Central:

1. El sistema muestra en la Interfaz el tipo de casilla.
2. El estudiante visualiza el tipo de casilla.

3. El sistema muestra en la Interfaz la pregunta o indica el movimiento de la ficha de acorde al tipo de casilla.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
El tipo de casilla tiene un comportamiento determinado.		En cada casilla se visualiza el tipo.	Satisfactorio	
El tipo de casilla didáctica se resuelve con preguntas.		La interfaz muestra una ventana con preguntas y respuestas a seleccionar.	Satisfactorio	
El tipo de casilla obstáculo se resuelve con avance o retroceso.		La interfaz muestra un mensaje al usuario indicándole lo que va a realizar la ficha, avance o retroceso.	Satisfactorio	

En este cuarto capítulo presentado se obtiene un sistema completamente diseñado y construido en términos de clases. De igual forma fueron señalados los principios de implementación que debían tenerse en cuenta como línea base del proyecto. Se generaron además cada uno de los artefactos y diagramas referentes al flujo de trabajo de implementación y se realizaron pruebas al sistema final donde se han definido varios casos de pruebas en las que se verifica la funcionalidad de la aplicación. Al concluir este capítulo se culmina la implementación de la Lógica de Juego.

Conclusiones Generales

Acorde a los objetivos y tareas trazadas y el trabajo realizado en esta investigación se alcanzaron los siguientes resultados:

- Se realizó un estudio de las herramientas similares existentes a nivel mundial.
- Se diseñó la aplicación con todos sus componentes visuales en consecuencia a los requisitos trazados.
- Se implementó la aplicación con todas sus funcionalidades en correspondencia con las características diseñadas para la utilización de juegos didácticos.
- Se realizaron pruebas a la aplicación para determinar el funcionamiento correcto de la interfaz visual y las funcionalidades implementadas.

Por todo lo anteriormente expuesto es posible concluir que los objetivos trazados para el presente trabajo han sido cumplidos satisfactoriamente.

Recomendaciones

1. Investigar e implementar acerca de las nuevas funcionalidades que pueden adicionarse a la herramienta.
2. Ajustar el tiempo de respuesta a las preguntas que se muestran en los tableros de juegos didácticos.
3. Integrar la aplicación a la Plataforma Virtual de Aprendizaje EVA-MOODLE.

Referencias Bibliográficas

1. **Entorno de Aprendizaje. ATutor**. [En línea] [Citado el: 26 de 11 de 2009.] http://www.atutor.ca/.

2.**Herramienta Educativa sobre Software Libre para apoyar las asignaturas de Programación en la Universidad de las Ciencias Informáticas**.C. Habana

3. **Kaplún, M.** Los Materiales de autoaprendizaje. Marco para su elaboración. . Santiago, Chile : s.n., 1995. pag. 55.

4. **Castro Barrios, Ludiel.** Módulo para la construcción de tableros de juegos didácticos de la herramienta educaiva SMProg para el Entorno Virtual de Aprendizaje. Ciudad de la Habana : s.n., 2009.

5. **Botella, Dr. Antonio Grandío.** EVAI- Entorno Virtual de Aprendizaje Interactivo- elearning. [En línea] [Citado el: 26 de 11 de 2009.] http://www.evai.net/evaih/index.php.

6. **Herramienta Case**.Visual Paradigm. [En línea] [Citado el: 26 de 11 de 2009.] http://www.visual-paradigm.com/.

8. **Metodologías Ligeras**. SW-CMM. *Wikipedia.* [En línea] [Citado el: 29 de 11 de 2009.] http://es.wikipedia.org/wiki/SW-CMM.

9. **Entorno de Aprendizaje. Sakai**. [En línea] [Citado el: 26 de 11 de 2009.] http://sakaiproject.org/ .

10. **Proceso Unificado de Rational**. *Wikipedia.* [En línea] [Citado el: 29 de 11 de 2009.] http://es.wikipedia.org/wiki/RUP#Principales_caracter.C3.ADsticas.

11. **Didáctica aplicada.**Pangea. [En línea] [Citado el: 26 de 11 de 2009.] http://www.pangea.org/peremarques/mdidact.htm.

12. **Herramienta IDE .**Netbeans. [En línea] [Citado el: 26 de 11 de 2009.] http://netbeans.org.

13. **Entorno de aprendizaje.**Moodle. *moodle.org.* [En línea] [Citado el: 26 de 11 de 2009.] http://moodle.org.

14. **Entorno de aprendizaje.**Lrn. [En línea] [Citado el: 26 de 11 de 2009.] http://www.slideshare.net/.

15. **Lenguaje de programación Java.** [En línea] [Citado el: 22 de 1 de 2010.] http://es.wikipedia.org.

16. **Entorno de aprendizaje.** Ilias. [En línea] [Citado el: 26 de 11 de 2009.] http://en.wikipedia.org/wiki/ILIAS.

17. **Entorno Virtual de Aprendizaje**. *EVA-UCI.* [En línea] UCI. [Citado el: 26 de 11 de 2009.] http://eva.uci.cu.

18. **Entorno de aprendizaje.**Dokeos. [En línea] [Citado el: 26 de 11 de 2009.] http://www.dokeos.com/.

19. **Desarrollo Ágil de Software**. *Wikipedia.* [En línea] [Citado el: 26 de 11 de 2009.] http://es.wikipedia.org/wiki/Metodolog%C3%ADas_%C3%A1giles#Metodolog.C3.ADas_.C3.A1giles.

20. **ConocimientosWeb**.*ConocimientosWeb.net.*http://www.conocimientosweb.net/zip/article108.html.

21. **Entorno de aprendizaje.**Claroline. *Claroline.net.* [En línea] [Citado el: 26 de 11 de 2009.] http://www.claroline.net/.

22. **Entorno de aprendizaje.** BlackBoard. [En línea] [Citado el: 26 de 11 de 2009.] http://www.blackboard.com/.

23. **Pressman, Roger S.** *INgeniería de Software, un enfoque práctico.* s.l. : Editorial Mc Graw Hill, 2002.

24. **Alvarez, Sara.** Arquitectura cliente-servidor. [En línea] 30 de 08 de 2007. [Citado el: 13 de 03 de 2010.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

Bibliografía

1. Equipo de desarrollo de Netbeans. Netbeans.org. [En línea] [Citado el: 19 de 1 de 2009.] http://www.netbeans.org/index_es.html.

2. Equipo de desarrollo de Sun Microsystems. Sun Microsystems. [En línea] 1994. [Citado el: 8 de 2 de 2009.] http://java.sun.com/docs/books/tutorial/uiswing/.

3. Equipo de desarrollo de Visual Paradigm. [En línea] [Citado el: 14 de 3 de 2009.] http://www.visual-paradigm.com/.

4. Equipo de desarrollo de IBM. IBM - Rational Unified Process (RUP). [En línea] IBM. [Citado el: 26 de 4 de 2009.] http://www-01.ibm.com/software/awdtools/rup/.

5. Java en castellano. [En línea] 1999. http://www.programacion.com/java/tutoriales/.

6. Módulo para la construcción de tableros de juegos didácticos de la herramienta educativa SMProg para el Entorno Virtual de Aprendizaje Moodle. Ing. Ludiel Castro Barrios, 2009.

7. Pressman, Roger S. INgeniería de Software, un enfoque práctico. s.l. : Editorial Mc Graw Hill, 2002.

8. Kaplún, M. Los Materiales de autoaprendizaje. Marco para su elaboración. . Santiago, Chile : s.n., 1995. pag. 55.

9. Manual de java. [En línea]. [Citado el: 26 de 09 de 2009.] http://sunsite.unam.mx/java.html

Anexos

Figura 15. La ficha se encuentra en la casilla de Salida.

Figura 16. Momento de la rotación del dado.

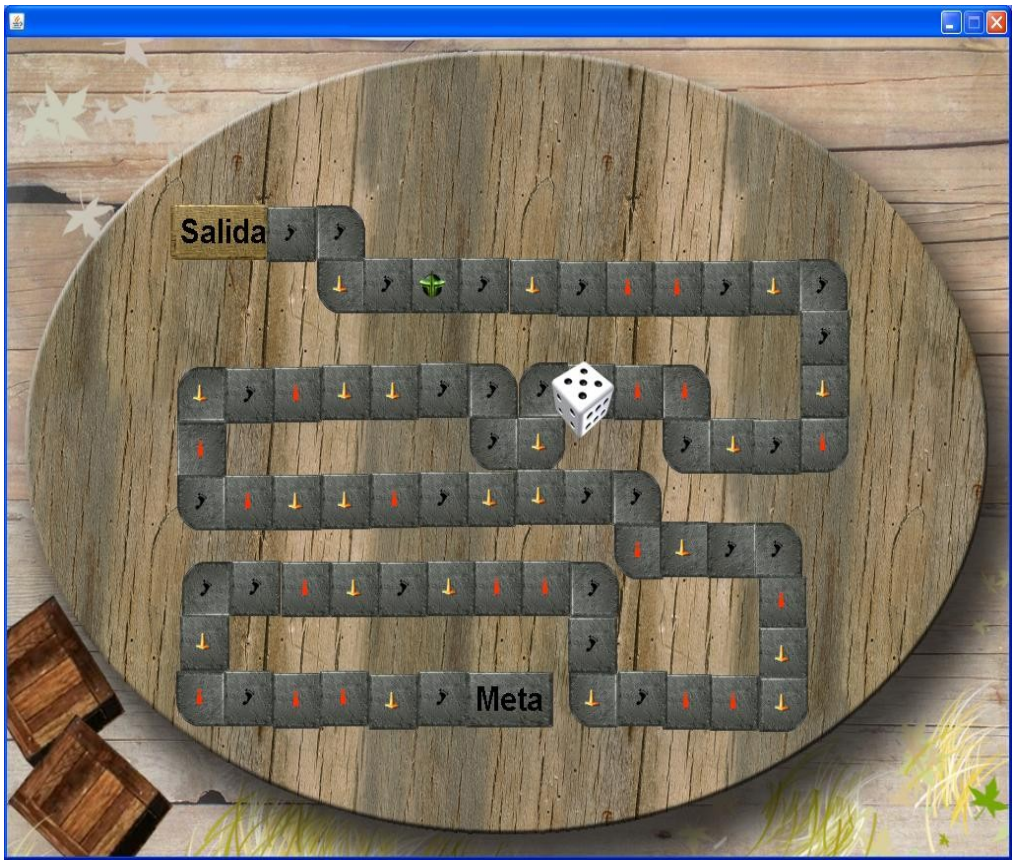


Figura 16. Las casillas que caminó la ficha después del lanzamiento de dado.

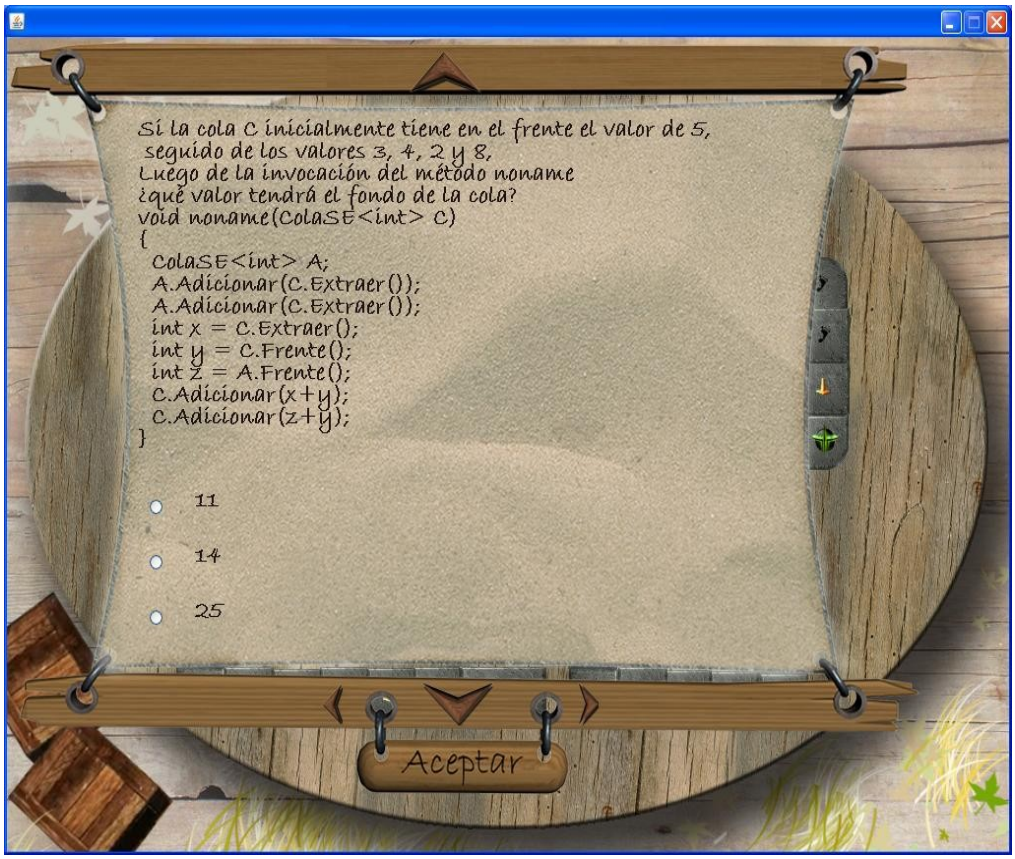


Figura 17. Resultado de la conexión con la base de datos después de que la ficha cae en una casilla de tipo didáctica.

Figura 18.Tablero Avanza hasta la Meta.

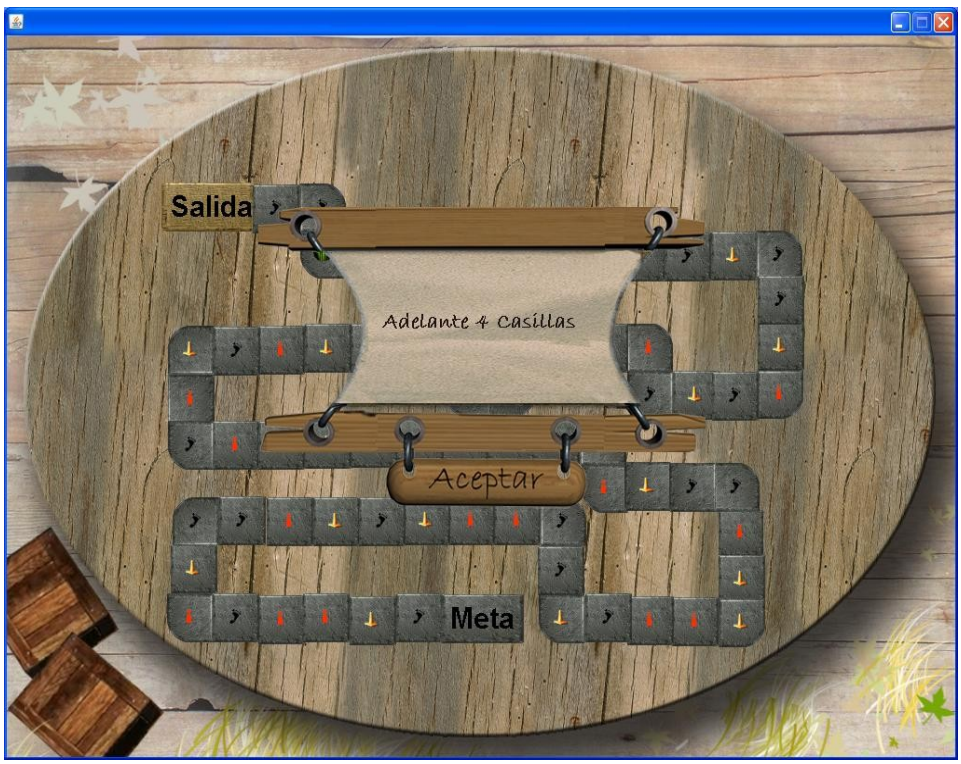


Figura 19. Cómo se comporta una casilla de tipo obstáculo.

Figura 20.Tablero Avanza hasta la Meta.

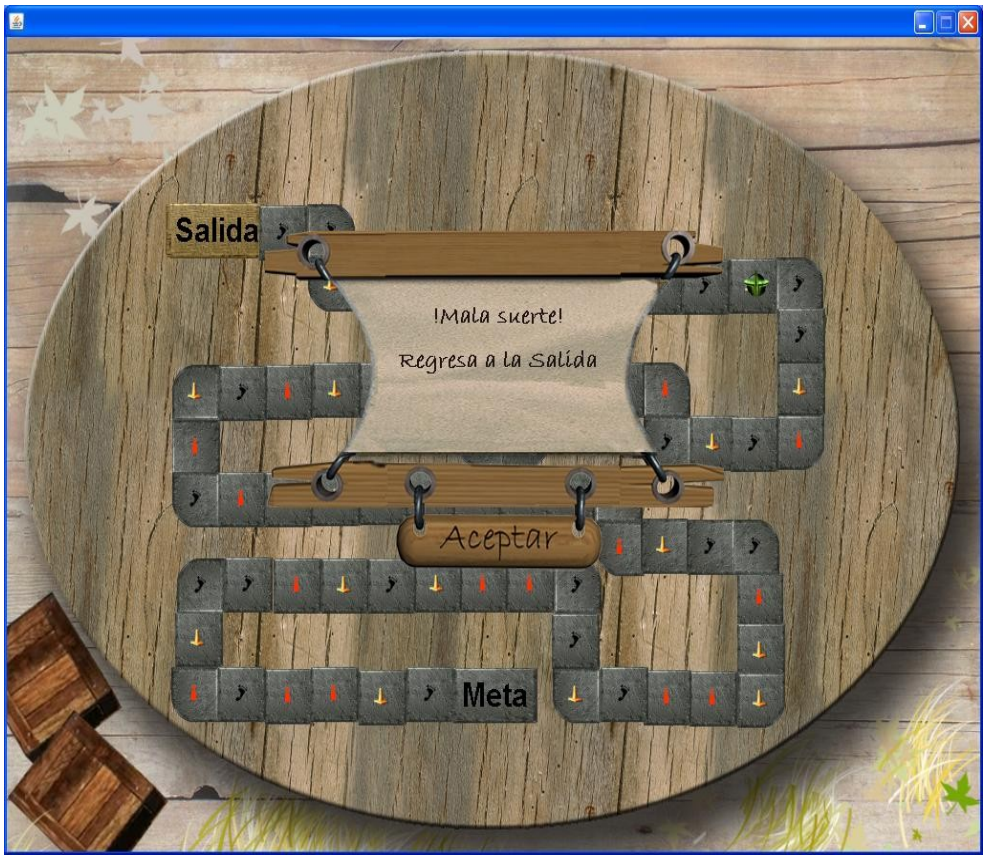


Figura 21. Comportamiento de la casilla obstáculo donde se retrocede.

Figura 22. El jugador llega a la meta y gana el juego.

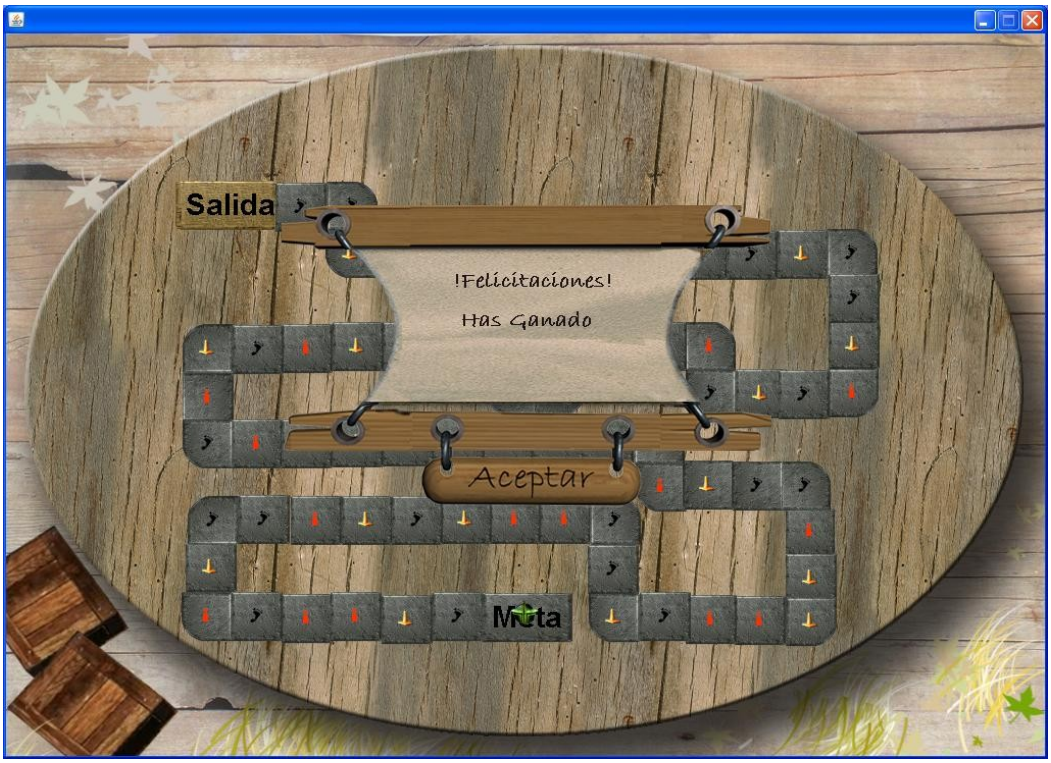


Figura 23. Comportamiento de la casilla de tipo obstáculo.

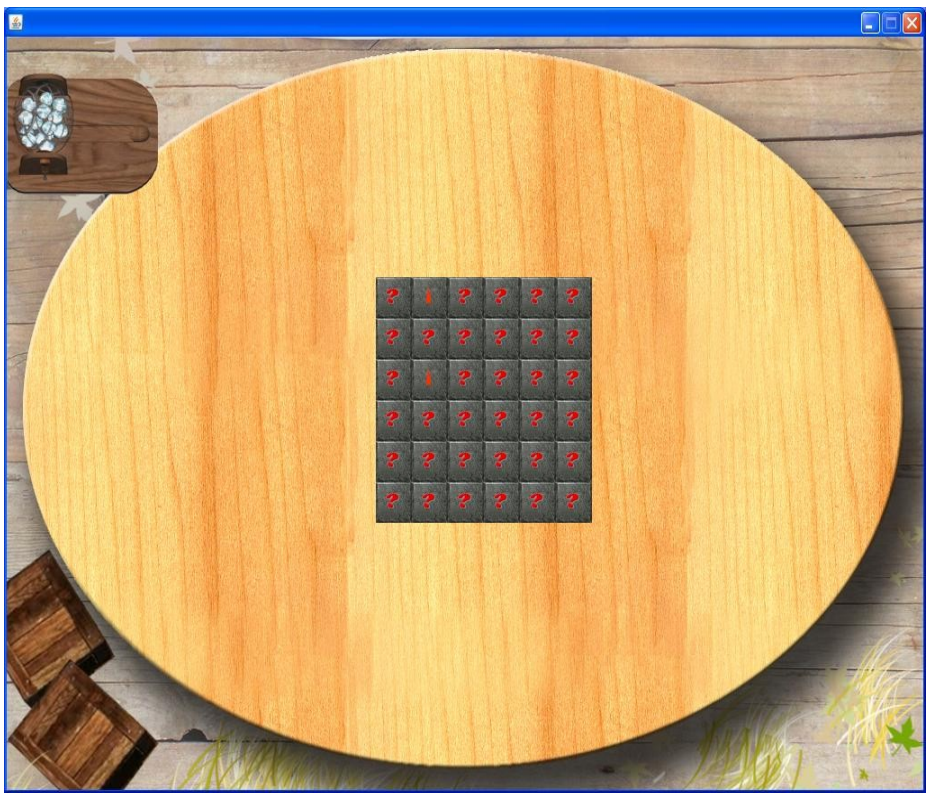


Figura 24.Ppodemos ver la utilización del bombo.

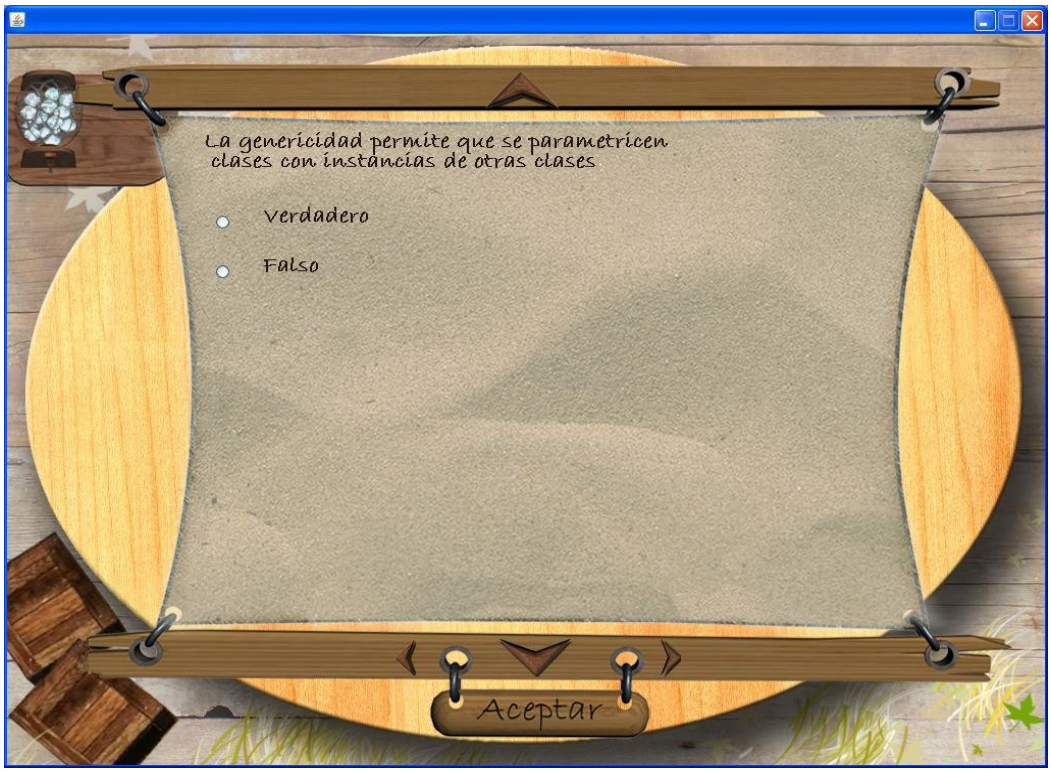


Figura 25. Todas las casillas contienen preguntas por lo que también hay conexión con la base de datos.

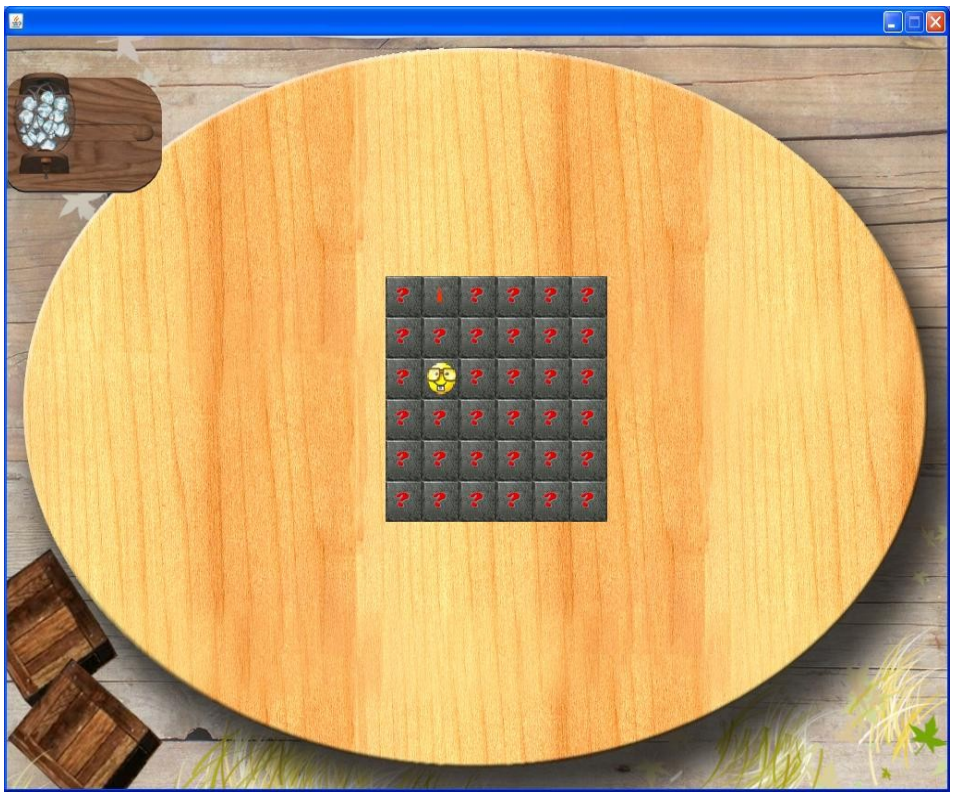


Figura 26.Si responde mal la pregunta el tipo de ficha cambia.

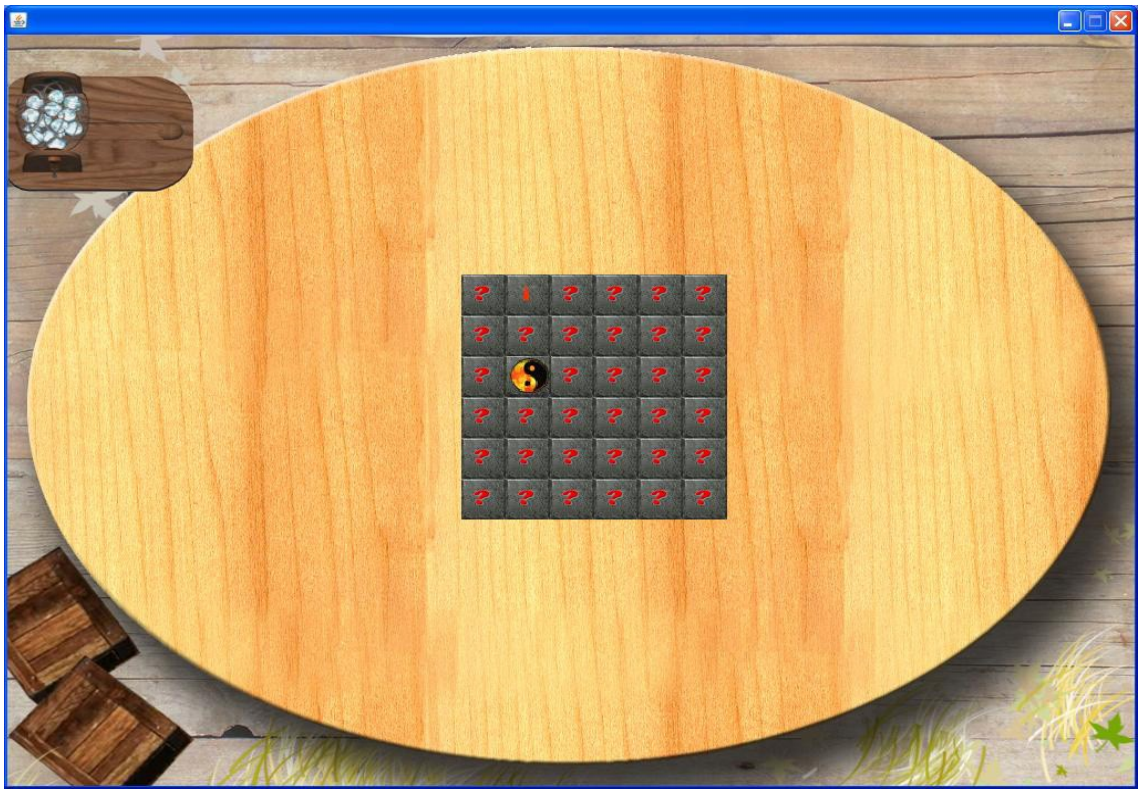


Figura 27. Este es el tipo de ficha que sale si responde correctamente.

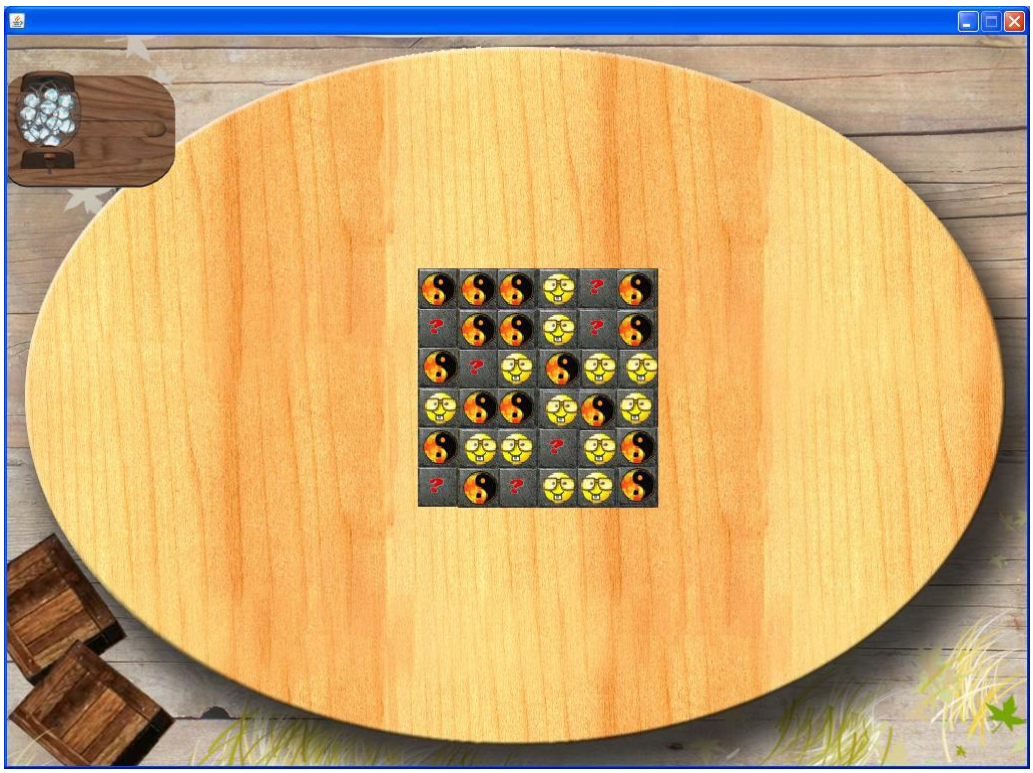


Figura 28. Cuando se gana se puede ver de esta forma.

Figura 29.Tablero Bingo con mensaje de felicitación.

Figura 30. Cuando no gana se le muestra este tipo de mensaje.

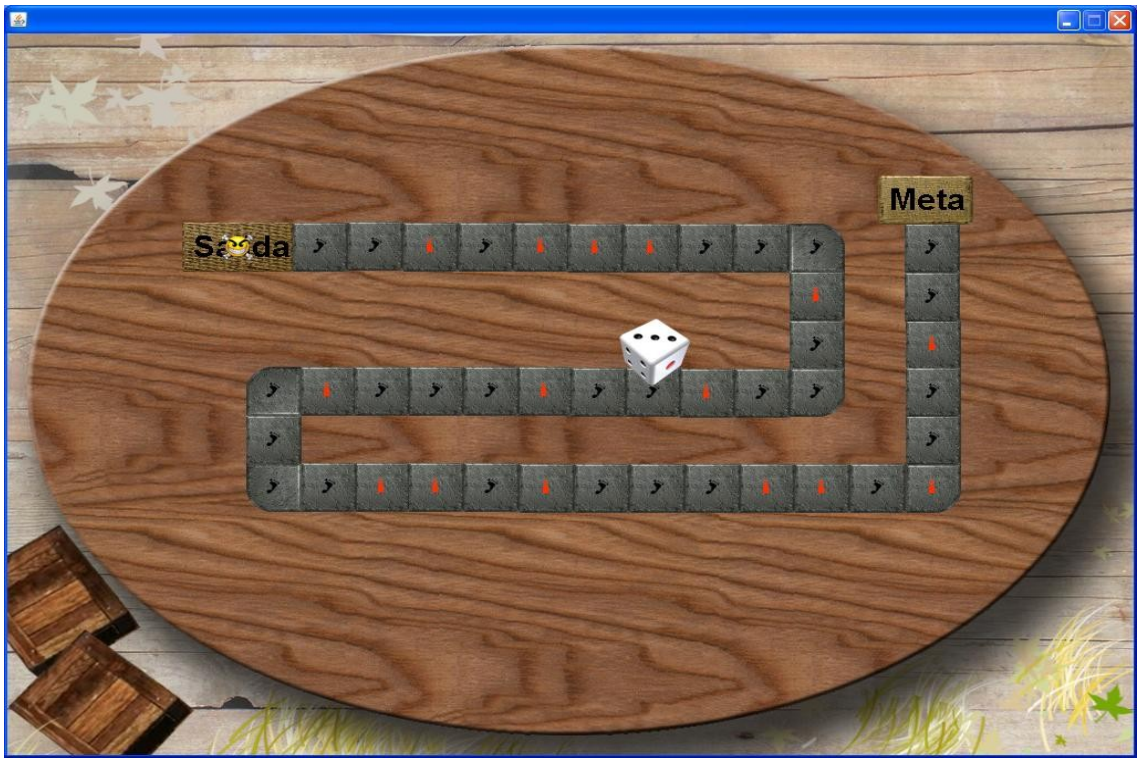


Figura 31.Tablero Sube y Baja.

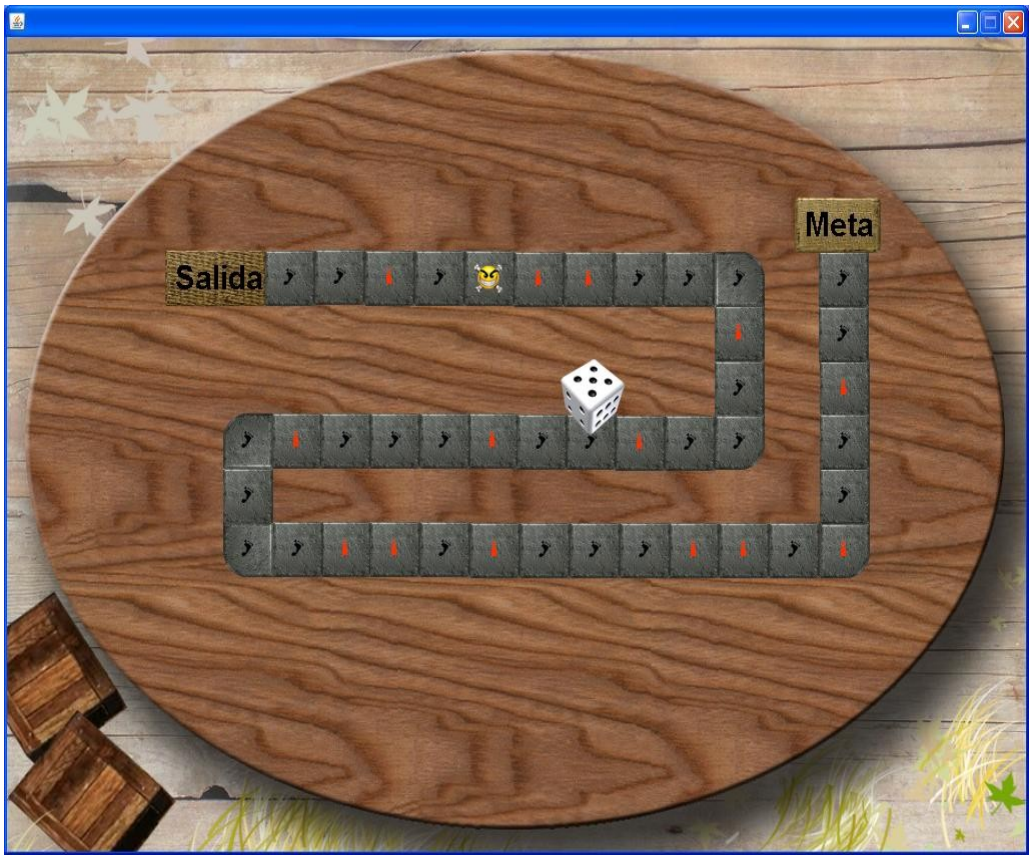


Figura 32. La ficha camina en el tablero después del lanzamiento del dado.

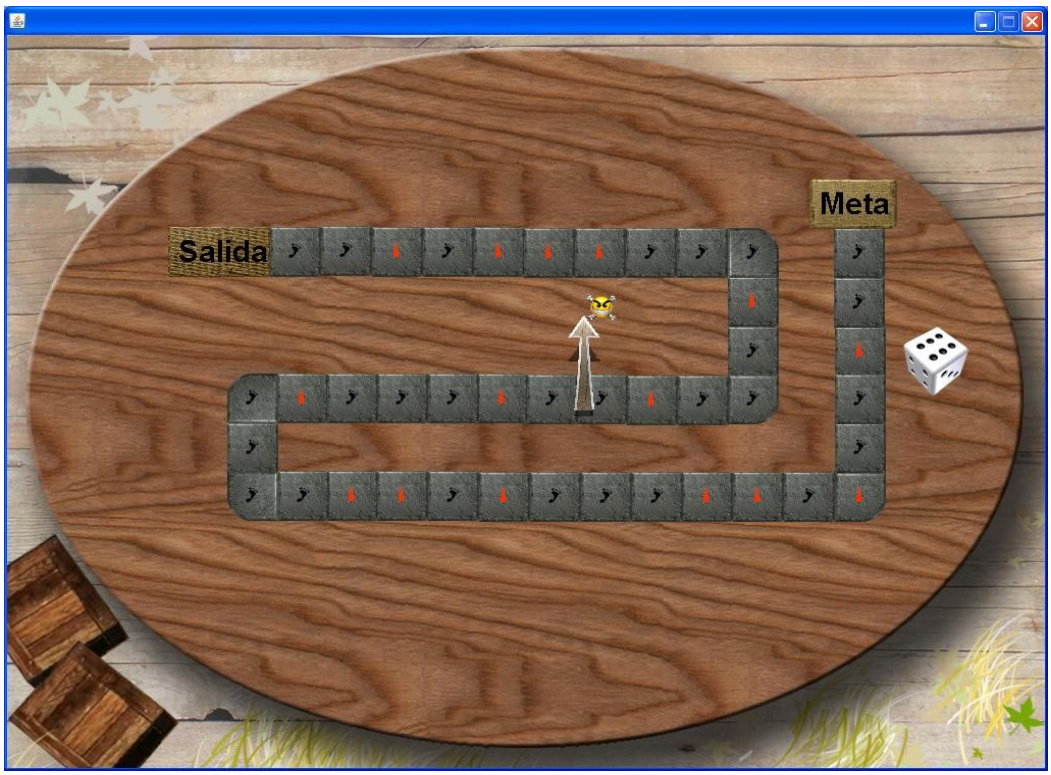


Figura 33. La ficha sube de posición en dependencia de la casilla en la que se encuentre.

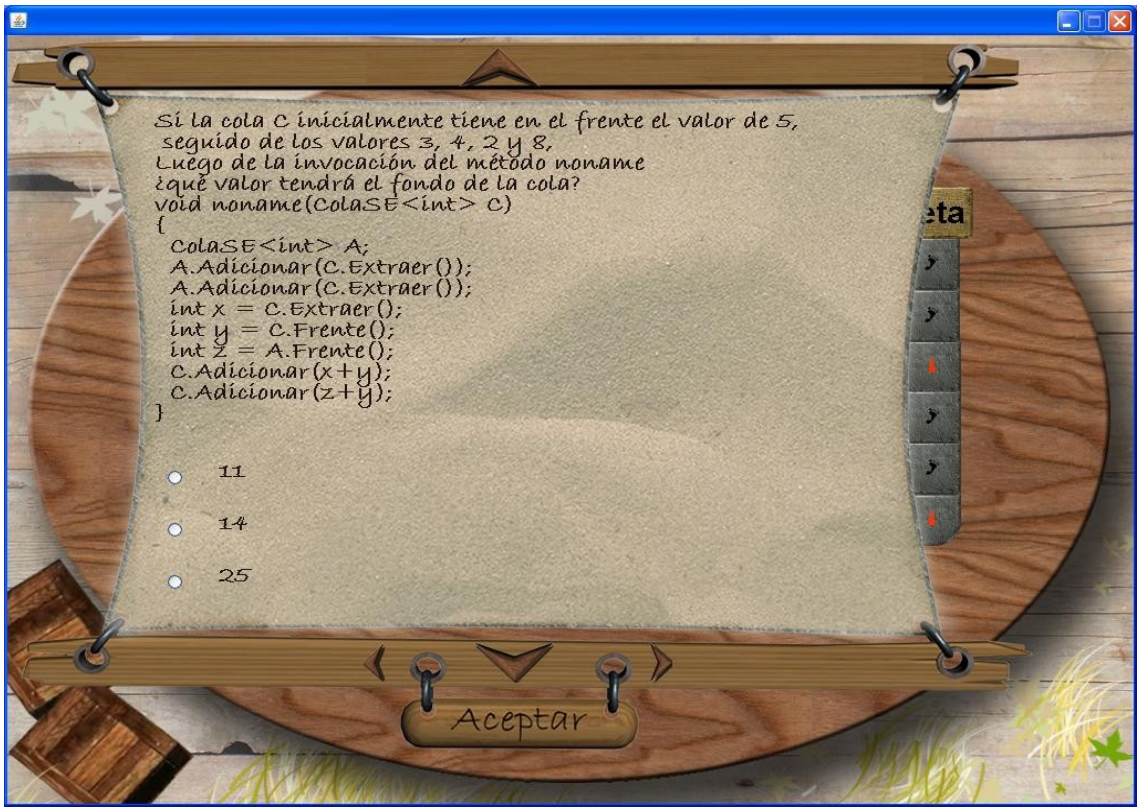


Figura 34. La pregunta que le corresponde cuando se conecta a la base de datos.

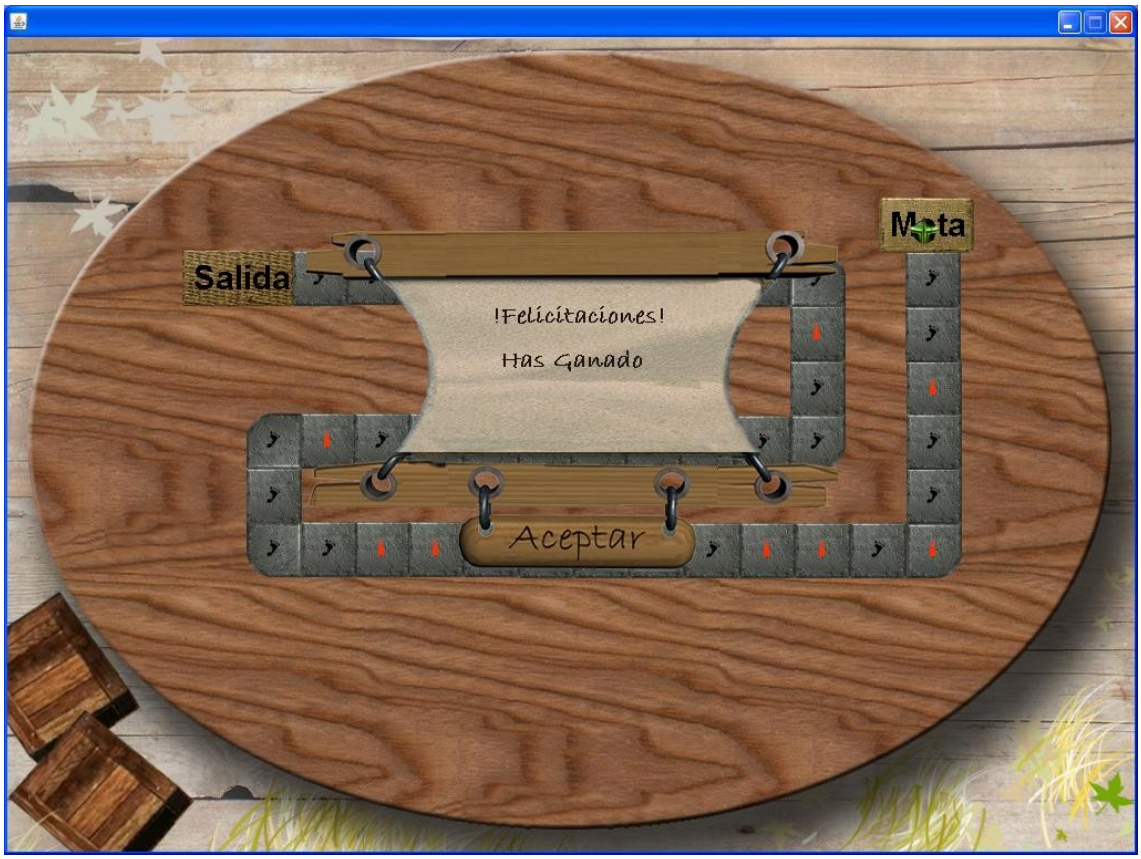


Figura 35. Cuando el estudiante gana.

Tabla 16: Descripción de la clase DidacticaVP

Nombre: DidacticaVP: CasillaVP	
Tipo de clase: Interfaz	
Atributo heredados del padre	**Tipo**
Alto	int
Ancho	int
Angulo	double
banderaCaso	boolean
banderaLetra	boolean
Caso	int
colorBorder	Color
colorFondo	Color
colorLetra	Color
ID	int
nombImg	LinkedList<string>
posX	int

posY	int
Tex	String
X	Int
Y	int
Para cada responsabilidad:	
Nombre:	DidacticaVP()
Descripción:	Inicializa los atributos de la clase.
Nombre:	DidacticaVP(java.lang.String tex, double angulo, int ID, int ancho, int alto, java.awt.Color colorFondo, java.awt.Color colorBorder, java.awt.Color colorLetra, int posX, int posY, int caso, boolean banderaCaso, boolean banderaLetra, TipoPreguntaVP tipoPreg)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	pintar(java.awt.Graphics g, TableroVP t)
Descripción:	Pinta la casilla didáctica.
Nombre:	TipoPreguntaVP getTipoPreg()
Descripción:	Obtiene el tipo de pregunta.
Nombre:	Angulo(), generarNomImag(), getAlto(), getAncho(), getAngulo(), getCaso(), getColorBorder(), getColorFondo(), getColorLetra(), getID(), getPosX(), getPosY(), getTex(), isBanderaCaso(), isBanderaLetra(), obtener(), obtenerY(), pintarDis(), pintarPre(), setAlto(), setAncho(), setAngulo(), setBanderaCaso(), setBanderaLetra(), setCaso(), setColorBorder(), setColorFondo(), setColorLetra(), setID(), setPosX(), setPosY(), setTex()
Descripción:	Métodos heredados de la clase CasillaVP

Tabla 17: Descripción de la clase FichaVP

Nombre: FichaVP	
Tipo de clase: Interfaz	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	FichaVP(int diametro, java.awt.Color color)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	FichaVP()
Descripción:	Inicializa los atributos de la clase.
Nombre:	Void CaminoHastaSalida(CasillaVP casillaiCvp)

Descripción:	Adiciona la casilla a la lista si no está repetida con el objetivo de tener un camino de casillas sin repetir para poder caminar hacia atrás sin que la ficha se comporte de forma no adecuada.
Nombre:	LinkedList<CasillaVP> getCasilla()
Descripción:	Obtiene la lista de casillas .
Nombre:	Color getColor()
Descripción:	Obtiene el color de la ficha.
Nombre:	Int getDiametro()
Descripción:	Obtiene el diámetro de la ficha.
Nombre:	Int getID()
Descripción:	Obtiene el id de la ficha.
Nombre:	Image getImage()
Descripción:	Obtiene la imagen de la ficha.
Nombre:	ImageIcon getImgAux()
Descripción:	Obtiene la imagen de la ficha como icono.
Nombre:	Int getPosX()
Descripción:	Obtiene la posición x de la ficha.
Nombre:	Int getPosY()
Descripción:	Obtiene la posición x de la ficha.
Nombre:	Void insertarCasilla(CasillaVP casilla)
Descripción:	Este es el método que hace el historial completo de las fichas en el tablero y tiene cada movimiento de la ficha.
Nombre:	Boolean isDisOpre()
Descripción:	Devuelve verdadero si esta presionado y falso si no.
Nombre:	Void pintar(java.awt.Graphics g, TableroVP t)
Descripción:	Método pintar de la clase casilla.
Nombre:	Void retroceder(int n)
Descripción:	Retrocede la ficha según el número de casillas entrada por parámetros.

Tabla 18: Descripción de la clase ImagenesVP

Nombre: ImagenesVP	
Tipo de clase: Interfaz	
Atributos	**Tipo**
Para cada responsabilidad:	

Nombre:	ImagenesVP() ()
Descripción:	Inicializa los atributos de la clase.
Nombre:	ImagenesVP(int posX, int posY, java.lang.String url, int ancho, int int posX, int posY, java.lang.String url, int ancho, int alto alto)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	Int Angulo()
Descripción:	Devuelve el ángulo en una variable entera.
Nombre:	Int getAlto()
Descripción:	Obtiene las dimensiones.
Nombre:	Int Ancho()
Descripción:	Obtiene las dimensiones.
Nombre:	Double getAngulo()
Descripción:	Obtiene las dimensiones.
Nombre:	Int getID()
Descripción:	Obtiene el id de la imagen.
Nombre:	getPosX()
Descripción:	Obtiene las coordenadas.
Nombre:	getPosY()
Descripción:	Obtiene las coordenadas.
Nombre:	getUrl()
Descripción:	Obtiene la dirección de donde se cargo la imagen.
Nombre:	pintar(java.awt.Graphics g, TableroVP t)
Descripción:	Método pintar.

Tabla 19: Descripción de la clase ImpleGame

Nombre: ImpleGame	
Tipo de clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	ImpleGame()
Descripción:	Inicializa los atributos de la clase.
Nombre:	Void addTablero()
Descripción:	Adiciona un nuevo tablero.
Nombre:	Juego getJuego()
Descripción:	Obtiene un juego determinado.

Tabla 20: Descripción de la clase Juego

Nombre: Juego	
Tipo de clase: Controladora	
Atributos	**Tipo**
num	int
randObj	random
tableroVP	TableroVP
Para cada responsabilidad:	
Nombre:	Juego(TableroVP tableroVP)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	Void AnimarFicha(CasillaVP cvp, FichaVP fichaVP)
Descripción:	Este es el método que anima la ficha en cualquier dirección para que se desplace por todo el tablero.
Nombre:	Int distancia(int x0, int y0, int xf, int yf)
Descripción:	Calcula la distancia entre dos coordenadas para realizar una serie de cálculos para la flecha y la sombra.
Nombre:	Boolean esDidactica(CasillaVP c)
Descripción:	Pregunto si es didáctica o no.
Nombre:	esMeta(CasillaVP cvp)
Descripción:	Me dice cuando la ficha llegó a la meta.
Nombre:	esObstaculo(CasillaVP c)
Descripción:	Pregunto si es obstáculo o no.
Nombre:	Int genEnteroAleatorio(int rinic, int rfin)
Descripción:	Genera un número entero aleatorio.
Nombre:	TableroVP getTableroVP()
Descripción:	Obtiene el tablero.
Nombre:	Void iniciarFichasCasillasdeSalida()
Descripción:	Pone la ficha en la casilla de salida.
Nombre:	Void moverFicha(FichaVP fichaVP, int numero)
Descripción:	Mueve la ficha la cantidad de casillas pasada por parámetros.
Nombre:	Void setTableroVP(TableroVP tableroVP)
Descripción:	Muestra el tablero.
Nombre:	tratarDidactica(CasillaVP c)

Descripción:	Este es el método que obtiene las preguntas y respuestas de la Base de Datos.

Tabla 21: Descripción de la clase ObstaculoVP

Nombre: ObstaculoVP: CasillaVP	
Tipo de clase: Interfaz	
Atributos heredados del padre	**Tipo**
alto	Int
ancho	Int
angulo	double
banderaCaso	boolean
banderaLetra	Boolean
caso	Int
colorBorder	Color
colorFondo	Color
colorLetra	Color
ID	Int
nombImg	LinkedList<String>
posX	Int
posY	int
tex,	string
X	Int
Y	int
Para cada responsabilidad:	
Nombre:	ObstaculoVP()
Descripción:	Inicializa los atributos de la clase.
Nombre:	ObstaculoVP(java.lang.String tex, double angulo, int ID, int ancho, int alto, java.awt.Color colorFondo, java.awt.Color colorBorder, java.awt.Color colorLetra, int posX, int posY, int caso, boolean banderaCaso, boolean banderaLetra, TipoObstaculoVP tipoObst)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	ObstaculoVP(TipoObstaculoVP tipoObs)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	TipoObstaculoVP getTipoObst()
Descripción:	Obtiene el tipo de obstáculo.

Nombre:	Void pintar(java.awt.Graphics g, TableroVP t)
Descripción:	Método para pintar.

Tabla 22: Descripción de la clase SubeBajaVP

Nombre: SubeBaja_VP: Juego	
Tipo de clase: Interfaz	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	SubeBaja_VP(CasillaVP vInicial, CasillaVP vFinal, java.awt.Color color)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.

Tabla 23: Descripción de la clase TablaVP

Nombre: TablaVP	
Tipo de clase: Interfaz	
Atributos	**Tipo**
casilla	LinkedList<CasillaVP>
columnas	Int
filas	int
Para cada responsabilidad:	
Nombre:	TablaVP(java.util.LinkedList<CasillaVP> casilla, int filas, int columnas)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	LinkedList<CasillaVP> getCasilla()
Descripción:	Obtiene la lista de casillas.
Nombre:	Int getColumnas()
Descripción:	Obtiene las columnas.
Nombre:	LinkedList<FichaVP> getFicha()
Descripción:	Obtiene la lista de fichas.
Nombre:	Int getFilas()
Descripción:	Obtiene las filas.
Nombre:	Void insertarFicha(FichaVP f)
Descripción:	Inserta una ficha en el tablero.

Tabla 25: Descripción de la clase Bombo

Nombre: Bombo
Tipo de clase: Interfaz

Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	Bombo(int posX, int posY)
Descripción:	Inicializa los atributos de la clase con los parámetros asignados.
Nombre:	Bombo()
Descripción:	Inicializa los atributos de la clase.
Nombre:	Void bound()
Descripción:	Pinta un círculo dentro de un rectángulo (genérico del IDE).
Nombre:	Int getAlto()
Descripción:	Obtiene el parámetro alto de la figura pintada.
Nombre:	Int getAncho()
Descripción:	Obtiene el parámetro ancho de la figura pintada.
Nombre:	Int getPosX()
Descripción:	Obtiene el parámetro posición x de la figura pintada.
Nombre:	Int getPosY()
Descripción:	Obtiene el parámetro posición y de la figura pintada.
Nombre:	Void setAlto(int alto)
Descripción:	Modifica el parámetro.
Nombre:	Void setAncho(int ancho)
Descripción:	Modifica el parámetro.
Nombre:	Void setPosX(int posX)
Descripción:	Modifica el parámetro.
Nombre:	Void setPosY(int posY)
Descripción:	Modifica el parámetro.

Tabla 26: Descripción de la clase Ventana

Nombre: Ventana	
Tipo de clase: Interfaz	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	Ventana()
Descripción:	Crea nueva forma Ventana
Nombre:	Void abrirventana()
Descripción:	Abre una ventana
Nombre:	Void cerrarventana()

Descripción:	Cierra una ventana
Nombre:	Juego getJuego()
Descripción:	Obtiene el juego.
Nombre:	Void InsertarEjercicio(java.lang.String enunciado, java.util.LinkedList<Preguntas> preguntas)
Descripción:	Inserta una pregunta en la ventana de acuerdo a la lista de preguntas disponibles.
Nombre:	Boolean isVisibilidad()
Descripción:	Pregunta si es visible o no la ventana.
Nombre:	Void setJuego(Juego juego)
Descripción:	Modifica el parámetro.
Nombre:	Void setVisibilidad(boolean visibilidad)
Descripción:	Modifica el parámetro.