Módulo para juegos serios de palabras en el software ATcnea en su versión 1.0.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores

Greter Hernández Piñero

Danger Germán Fernández Bastida

Tutores

Ing. Yordankis Matos López

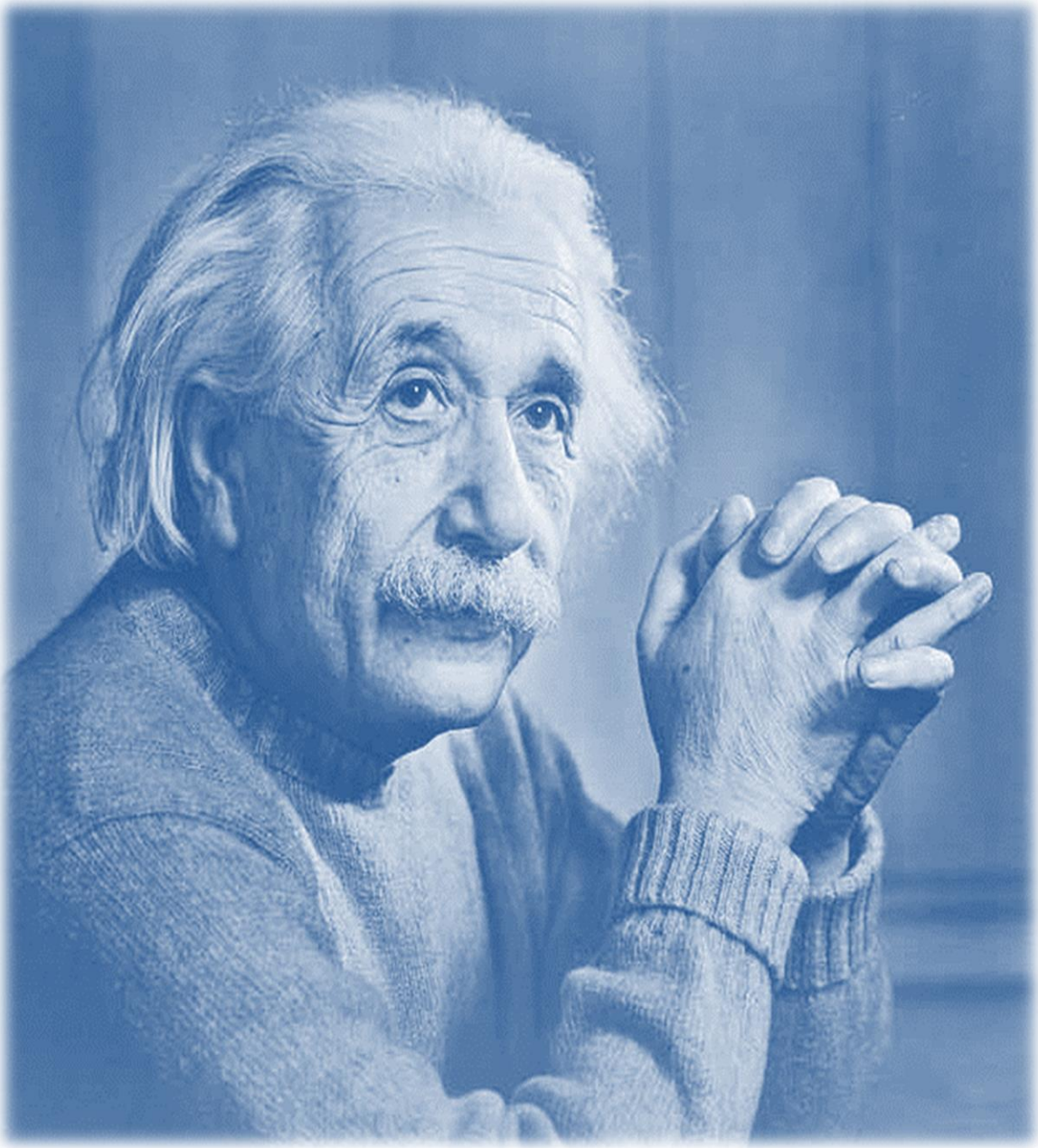
Ing. Carlos Montenegro Amador

Ing. Guillermo González Jiménez

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

La Habana, junio de 2017



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad. "

Albert Einstein

Declaración de Autoría

Declaramos ser los autores del presente trabajo de diploma y otorgamos 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 _____.

_____ Firma del Autor
Greter Hernández Piñero

_____ Firma del Autor
Danger Germán Fernández Bastida

_____ Firma del Tutor
Ing. Yordankis Matos López

_____ Firma del Tutor
Ing. Carlos Montenegro Amador

_____ Firma del Tutor
Ing. Guillermo González Jiménez

Agradecimientos

A mi mamá: por hacerme feliz, por ser lo más grande en mi vida, por ser la más fuerte guerrera que exista, por sus consejos siempre certeros, por sus abrazos, por las enseñanzas, por sus besos, por su calor, por su cariño, por todo.

A mi papá: por guiarme, por los consejos, por ser mi amigo, padre, hermano, confidente, por ayudarme a cumplir este sueño de pequeño, por todo.

A mis abuelos: por su cariño, amor, comprensión, consentimiento y dedicación, por los consejos, por siempre estar a mi lado, por todo.

A mi compañera de tesis: por ayudarme en todo, por entenderme, por su amor, dedicación, apoyo incondicional, por amarme, por todo.

A mis tutores: por guiarme en todo el proceso, por su participación activa, por los conocimientos transmitidos, por todo.

A mis amigos: por las experiencias vividas, por la amistad, comprensión, consejos, por todo.

A la Universidad de las Ciencias Informáticas: por la oportunidad de cumplir un sueño.

Danger

A mi mamá: por sus consejos, por sus abrazos, por las enseñanzas, por sus besos, por su calor, por su cariño, por hacerme feliz, por ser lo más grande en mi vida, por todo.

A mi papá: por guiarme, por los consejos, por ser mi amigo, padre, confidente, por ayudarme a cumplir este sueño de pequeña, por todo.

A mis abuelas: por su cariño, amor, comprensión, consentimiento y dedicación, por los consejos, por siempre estar a mi lado, por todo.

A mi compañero de tesis: por ayudarme en todo, por entenderme, por su amor, por todo.

A mis tutores: por guiarme en todo el proceso, por los conocimientos transmitidos, por todo.

A mis amigos: por las experiencias vividas, por la amistad, consejos, por todo.

Greter

Dedicatoria

Le dedico este trabajo a mi mamá, a mi papá, mi abuela, mi hermana, mi abuelo, a mi novia, al resto de la familia, a mis amigos, a mis profesores y todas aquellas personas que lo hicieron posible.

Danger

Le dedico este trabajo a mi mamá, a mi papá, mi abuela, mi tía, a mi novio, al resto de la familia, a mis amigos, a mis profesores y todas aquellas personas que lo hicieron posible.

Greter

Resumen

El avance de las Tecnologías de la Información y las Comunicaciones, ha beneficiado a la sociedad en todos los sectores, desde la comunicación hasta la educación. Estas tecnologías tienen un impacto relevante en el proceso de enseñanza-aprendizaje, debido a que favorecen el surgimiento de nuevos espacios educativos nombrados aulas tecnológicas. El proyecto ATcnea es un fiel ejemplo de lo antes mencionado, surge a partir de la solicitud que llega a la Universidad de las Ciencias Informáticas, específicamente al centro FORTES de la facultad 4, de realizar un *software* que en conjunto con los productos entregados por las entidades GEDEME-HAIER integrarían un aula tecnológica. El *software* ATcnea es un sistema de gestión del aula que permite a un profesor guiar el proceso de enseñanza-aprendizaje. Carece de recursos didácticos que permitan contribuir a la atención, motivación, desarrollo de la observación, apoyo a la toma de decisiones, tolerancia a la frustración y habilidades del lenguaje como la redacción y la ortografía en los estudiantes. Por esta razón, se definió como objetivo desarrollar un módulo para juegos serios de palabras que permita aumentar los recursos didácticos disponibles en el software ATcnea. Para la implementación del módulo fue necesario utilizar las tecnologías y herramientas definidas por el proyecto para el desarrollo del *software*. Seguidamente se transitó siguiendo la metodología de desarrollo AUP UCI para guiar todo el proceso y se realizaron pruebas de *software* para validar la solución propuesta.

Palabras clave: aula tecnológica, juegos serios de palabras, proceso de enseñanza-aprendizaje, recursos didácticos.

Índice

Índice de figuras

Índice

Introducción

En la antigüedad cuando el hombre inicia sus procesos de aprendizaje lo hace de manera espontánea y natural, debido a la necesidad de conocer y adaptarse al medio ambiente. Al pasar los siglos, estos conocimientos adquiridos por el hombre se almacenan y se estructuran en asignaturas con el fin de trasmitirlos a futuras generaciones. En este contexto surge la enseñanza intencional, como una actividad estratégica y constructiva que se desarrolla en un espacio educativo denominado aula tradicional (1).

Un aula tradicional es el lugar donde un grupo de estudiantes comparten la misma educación, dirigidos por un profesor que transmite sus conocimientos de forma directa (2). En este tipo de aula, las Tecnologías de la Información y las Comunicaciones (TIC) no son utilizadas para involucrar a los estudiantes en el proceso de enseñanza-aprendizaje[1]. Actualmente el papel de las TIC en la sociedad es muy importante ya que ofrecen recursos, herramientas y programas que se utilizan para procesar, administrar y compartir la información como: internet, correo electrónico, juegos, servicios web, entre otros recursos. La ausencia de las TIC en la educación provoca en los estudiantes la falta de motivación, el interés por aprender y la apatía hacia la escuela (3).

En la constante búsqueda de erradicar la falta de motivación, el interés por aprender y la apatía hacia la escuela de los estudiantes, nacen nuevas formas, estrategias y métodos cuyo propósito es mejorar la calidad de la educación. En este ámbito surgen las aulas tecnológicas como un ambiente en donde la gestión del conocimiento y la relación profesor-estudiante genera un cambio en la forma de educar. Estas se componen de tecnologías como pueden ser: pizarra digital interactiva, video proyector, cámara de documentos, audio, datos, video, e internet. Además de contar con un sistema de gestión del aula que permite la interacción profesor-estudiante (4).

Cuba no queda ajena a estos cambios, puesto que durante el curso 2015-2016 llega a la Universidad de las Ciencias Informáticas (UCI) la solicitud de desarrollar un *software,* que en conjunto con los productos entregados por las entidades GEDEME[2]-HAIER[3] integrarían un aula tecnológica. Aunque a nivel internacional existen productos de este tipo, GEDEME

[1] Proceso de enseñanza-aprendizaje: es el procedimiento mediante el cual se transmiten conocimientos especiales o generales sobre una materia.

[2] GEDEME: empresa industrial para la informática, las comunicaciones y la electrónica.

[3] HAIER: multinacional de electrónica de consumo china y compañía de electrodomésticos.

realiza la solicitud con el objetivo de lograr la soberanía tecnológica en el país expandiendo por Cuba y el mundo un *software* cubano. De conjunto con el Centro CESOL[4], se le asigna al Centro de Tecnologías para la Formación (FORTES) la tarea de desarrollar el producto ATcnea.

El *software* ATcnea es un sistema de gestión del aula compuesto por dos aplicaciones, una destinada al profesor o moderador de la clase desarrollada en el lenguaje Java y otra destinada a los estudiantes o receptores desarrollada utilizando la plataforma Android. Entre ambas existe comunicación para que el profesor desde su PC[5] pueda guiar el proceso de enseñanza-aprendizaje usando los medios que proveen las aulas tecnológicas y los estudiantes desde las tabletas, recibir e interactuar con el contenido. Cuenta con funcionalidades que permiten a un profesor guiar el proceso de enseñanza-aprendizaje. Dentro de ellas se encuentran: atención diferenciada a los estudiantes, herramienta chat, creación de actividades grupales, creación de preguntas con varias tipologías y exámenes, registro de asistencia, compartir audio y video, envío de archivos, control de los terminales de los estudiantes de manera remota, reportes de asistencia, evaluaciones, entre otras.

El *software* ATcnea tiene el propósito de convertirse en una herramienta educativa[6] que contenga recursos didácticos[7] disponibles para ejercitar y desarrollar las habilidades de cada parte, tanto de quien enseña como de quien aprende, para ser utilizado en cualquier nivel de enseñanza. ATcnea carece de funcionalidades que permiten al profesor contribuir a la atención, motivación, desarrollo de la observación, apoyo a la toma de decisiones, tolerancia a la frustración y habilidades del lenguaje como la redacción y la ortografía en los estudiantes.

Por estas razones se identificó como ***problema a resolver***: ¿Cómo aumentar los recursos didácticos disponibles en el *software* ATcnea?

[4] CESOL: Centro de Soluciones Informáticas.

[5] PC: Personal Computer (computadora personal).

[6] Herramienta educativa: programa educativo didáctico que es diseñado con el fin de apoyar la labor de los profesores en el proceso de enseñanza- aprendizaje.

[7] Recursos didácticos: materiales que tienen utilidad en el proceso educativo.

Con el fin de solucionar el problema identificado, se define como **objeto de estudio** de la presente investigación: juegos serios en sistemas de gestión de aulas tecnológicas. Enfocando el **campo de acción en**: juegos serios de palabras en el *software* ATcnea.

Teniendo en cuenta el problema a resolver se define como **objetivo:** desarrollar un módulo para juegos serios de palabras que permita aumentar los recursos didácticos disponibles en el *software* ATcnea.

Para guiar la investigación se define como **idea a defender:** el desarrollo de un módulo para juegos serios de palabras permitirá aumentar los recursos didácticos disponibles en el *software* ATcnea.

Para lograr el objetivo se definen como **resultados esperados:**

- Documentación asociada a la investigación.
- Artefactos generados durante la etapa de diseño.
- Un módulo para juegos serios de palabras en el *software* ATcnea.
- Componentes reutilizables para desarrollar juegos de palabras en la aplicación destinada al estudiante.

Métodos Investigativos:

Para realizar la investigación se hace uso de métodos teóricos y empíricos que a continuación se mencionan:

Métodos teóricos:

- ***Analítico-Sintético:*** se empleó para el análisis y extracción de los principales conceptos como aula tecnológica y juegos serios a incluir en el marco teórico.
- ***Histórico-Lógico:*** se empleó para realizar el estudio del estado del arte e investigar acerca de sistemas de gestión del aula como soluciones similares.
- ***Modelación:*** se utilizó para realizar los diagramas que modelan la solución del módulo para juegos serios de palabras.

Métodos empíricos:

- ***Observación:*** método que permitió estudiar los juegos serios para aulas tecnológicas, sus ventajas y desventajas.
- ***Entrevista:*** se empleó con el objetivo de analizar y comprender el funcionamiento del *software* ATcnea.

Para la elaboración del presente trabajo de diploma se decidió dividir la información en tres capítulos, los cuales quedan conformados de la siguiente manera:

Capítulo 1. Fundamentación teórica: en este capítulo se realiza un estudio de los elementos teóricos que sustentan el trabajo. Se analizan los sistemas similares existentes y se estudian la metodología, tecnologías y herramientas a utilizar.

Capítulo 2. Características y diseño del módulo: se realiza el diseño y descripción del módulo propuesto, así como los elementos que permiten describir la solución propuesta, tales como: modelo de dominio, requisitos funcionales y no funcionales, historias de usuario, prototipos de interfaz de usuario y el modelado de diseño de las funcionalidades.

Capítulo 3. Implementación y pruebas del módulo: se desarrolla la solución propuesta y se describen las pruebas realizadas al sistema, una vez que concluye la implementación para asegurar que este cumpla con las especificaciones requeridas.

Capítulo 1 Fundamentación teórica

Para la fundamentación del presente trabajo es necesario establecer ciertos conceptos básicos fundamentales para comprender la investigación. También es necesario un estudio a nivel mundial sobre sistemas similares y describir la metodología, herramientas y tecnologías a usar en la propuesta de solución.

1.1 Aula tecnológica

Un aula tecnológica es una solución educativa que brinda una experiencia única de aprendizaje. Tiene como objetivo la creación de un ambiente colaborativo, donde la tecnología enriquece el contenido académico de cada asignatura y permite al profesor-estudiante establecer una amplia comunicación (5).

Muchos son los autores que han definido el aula tecnológica. A continuación, se citan algunos conceptos.

- Segovia define aula tecnológica como: *"comunidad de aprendizaje, cuyo objetivo principal es el desarrollo de la inteligencia…de los alumnos…bajo la mediación de los profesores, por medio de métodos didácticos diversificados…en un espacio multiuso abierto, tecnológicamente equipado y organizado…"* (5).

- Antonia Lozano Díaz precisa: *"el sistema educativo aula inteligente es un constructo creativo, un conjunto de saberes que se plasman en una pedagogía singular. Propugna un cambio de modelo de educación a través de la reingeniería total del sistema educativo; lo hace partiendo de una determinada conceptualización de lo que sería la calidad en educación"* (6).

- Antonia Lozano expone: *"Un aula tecnológica o interactiva reúne aspectos arquitectónicos, ambientales, de acabado o mobiliario, como elemento fundamental el equipamiento físico y lógico básico, considerándose como tal las PC o dispositivos móviles, el software compatible y la conectividad adecuada que garantice la integridad del equipamiento"* (6).

Una vez analizados los conceptos anteriores, los autores de la presente investigación entienden por aula tecnológica: ambiente educativo compuesto por tecnologías donde el profesor hace uso de recursos didácticos para desarrollar de forma ágil e interactiva el proceso de enseñanza-aprendizaje.

1.1.1 Beneficios de un aula tecnológica

El aula tecnológica aporta múltiples beneficios a estudiantes y profesores. Mejora la calidad del proceso de enseñanza-aprendizaje en las instituciones educativas, condicionado por el avance tecnológico que en estas se emplea. A continuación, en la siguiente tabla se muestran algunos beneficios (7):

Tabla 1 Beneficios de un aula tecnológica

Profesor	Estudiante
❖ Uso y comprensión de la tecnología. ❖ Programación de contenidos. ❖ Seguimiento a los avances individuales y de grupo de los estudiantes. ❖ Incremento de la eficacia y eficiencia en el momento de impartir el contenido.	❖ Vive intensamente el aprendizaje en todas sus formas: visual, oral, escrita y auditiva. ❖ Desarrolla su creatividad y estimula su potencial innovador. ❖ Fortalece su aprendizaje autónomo. ❖ Interés por la ciencia y la tecnología.

1.2 Juego

Se define como la actividad que realizan uno o más jugadores, en el cual emplean su imaginación o herramientas para crear una situación con un número determinado de reglas, con el fin de proporcionar entretenimiento o diversión. Existen juegos competitivos, donde los jugadores tienen que lograr un objetivo, y juegos no competitivos, donde los jugadores buscan simplemente disfrutar de la actividad (8).

Los juegos se clasifican en diferentes tipos dentro de los cuales se encuentran (9):

- ❖ Juegos populares
- ❖ Juegos tradicionales
- ❖ Juegos de reglas
- ❖ Juegos de construcción
- ❖ Juegos de mesa
- ❖ Juegos de naipes
- ❖ Video juegos
- ❖ Juegos de rol

- Juegos didácticos

Precisamente son los juegos didácticos los utilizados como una herramienta fundamental para el proceso de enseñanza-aprendizaje, hacen posible elevar el trabajo independiente de los estudiantes y resolver situaciones problemáticas en la actividad práctica. El juego proporciona nuevas formas de explorar la realidad y estrategias diferentes para trabajar sobre la misma. Además, los juegos permiten a los estudiantes desarrollar su imaginación, pensar en numerosas alternativas para un problema, descubrir diferentes modos y estilos de pensamiento, y favorecen el cambio de conducta (10). Dentro de los juegos didácticos se encuentran los llamados juegos serios o juegos formativos que se basan en aplicar dinámicas propias de los juegos a la integración de los estudiantes con el proceso de enseñanza-aprendizaje.

1.2.1 Juegos serios

Muchos son los autores que han definido los juegos serios. A continuación, se citan algunos de ellos:

Primeramente, Clark Abt expone: *"tienen un propósito educativo explícito y cuidadosamente planeado, y no están pensados para ser jugados únicamente por diversión"* (11).

Mike Zydia actualizó el concepto dándole un alcance mayor, concibiendo los juegos serios como: *"una prueba mental a través de un ordenador que tiene reglas específicas y que utiliza el entretenimiento como forma de entrenamiento gubernamental o corporativo, y con finalidades educativas, sanitarias, de políticas, públicas y de comunicaciones estratégicas"* (12).

Finalmente, María Sánchez Gómez plantea*: son objetos y/o herramientas de aprendizaje que poseen en sí mismos y en su uso objetivos pedagógicos, didácticos, autónomos, autosuficientes y reutilizables, que posibilitan a los jugadores (estudiantes) a obtener un conjunto de conocimientos y competencias predominantemente prácticos* (13).

Una vez analizados estos conceptos los autores de la presente investigación entienden por juegos serios: recursos didácticos dirigidos al proceso de enseñanza-aprendizaje, que ofrecen a los estudiantes una mejor forma de aprender los contenidos y desarrollar sus habilidades.

1.2.2 Ventajas de los juegos serios para los estudiantes

Los juegos serios están dirigidos a la educación, tienen la finalidad de involucrar al máximo a los estudiantes en el proceso de enseñanza-aprendizaje. Al mismo tiempo son materiales de gran utilidad en el proceso educativo que permiten apoyar la función del profesor de generar conocimiento.

Los juegos serios Según Bernabéu y Goldstein:

- Facilitan la adquisición de conocimientos y el desarrollo de capacidades cognitivas superiores.
- Dinamizan el proceso de enseñanza-aprendizaje, mantienen y acrecientan el interés del estudiante y aumenta su motivación por el estudio.
- Fomentan la cohesión del grupo y la solidaridad entre iguales.
- Favorecen el desarrollo de la creatividad, la percepción, la inteligencia emocional, y aumentan la autoestima.
- Aumentan los niveles de responsabilidad de los estudiantes, ampliando también los límites de libertad (14).

1.2.3 Juegos serios para aulas tecnológicas.

Los juegos serios para aulas tecnológicas son aquellos que adaptan sus normas y funciones a los objetivos específicos de cada asignatura. Permiten ofrecer un entorno atractivo donde los usuarios “aprenden haciendo” (14). Poseen gran importancia ya que son aplicados a través de las tecnologías, hacen posible que el estudiante adquiera y demuestre sus conocimientos.

Según Sánchez, cuando se crea un juego para ser usado en la educación, se debe pensar en la experiencia del grupo al que será aplicado, y plantear una actividad apropiada para lograr una reproducción de esa experiencia en un mundo lúdico (14).

Los juegos serios elevan los niveles neuronales de dopamina[8] en los estudiantes, permiten incrementar la capacidad de resolución de problemas, memorización de reglas y estrategias

[8] Dopamina: neurotransmisor encargado de motivarnos en los momentos difíciles con la promesa de una recompensa (45).

para ganar. Además, desarrollan la observación, apoyo a la toma de decisiones, tolerancia a la frustración, rapidez e intuición (15).

1.2.4 Juegos serios de palabras.

En la actualidad el uso de juegos serios de palabras en aulas tecnológicas es indispensable ya que son productores de un gran efecto educativo. Proporcionan múltiples beneficios como: mejorar la lectura, escritura, vocabulario y ortografía en los estudiantes. Potencian la memoria, previenen la aparición y desarrollo del envejecimiento de la misma, favorecen el pensamiento y el razonamiento lógico (10).

Algunos juegos de palabras son:

- Anagrama
- Colgado o ahorcado
- Crucigrama
- Dilema
- Paradoja
- Palabras cruzadas
- Sopa de letras
- Scrabble
- Trabalenguas

1.3 Aplicaciones Similares.

Hoy en día el uso de aulas tecnológicas ha revolucionado el proceso de enseñanza-aprendizaje, de forma tal que proporciona el surgimiento de nuevos proyectos educativos, con el objetivo de mejorar los métodos y herramientas de enseñanza. Uno de sus principales componentes es el sistema de gestión del aula encargado de establecer interacción profesor-estudiante. A continuación, se hace un análisis de las principales funcionalidades de dichos sistemas (16).

Tabla 2 Funcionalidades de los sistemas similares

Funcionalidades	***Mythware Classroom Management Software***	***iTALC***	***NetSupport School***	***Impero Education Pro***

Pizarra interactiva	X	X	X	X
Abrir sitio web de forma remota	X		X	X
Transmisión por cámara	X	X		X
Transmisión de pantalla	X	X		X
Bloquear pantalla al estudiante	X	X		
Enviar mensaje a los estudiantes en forma de texto	X	X	X	X
Entrega y recogida de archivos		X	X	
Chat en grupo o a nivel individual	X	X	X	X
Evaluación módulo de preguntas y respuestas	X			X
Pruebas y exámenes para compartir contenido		X	X	
Añadir o adjuntar direcciones web e imágenes a mensajes instantáneos	X	X	X	X
Recursos online para exámenes y compartir contenido y consola técnica	X	X	X	X
Bloquear las estaciones de trabajo	X		X	
Visualización de las pantallas de los estudiantes en tiempo real		X	X	
Enviar preguntas rápidas			X	X
Gestión de la clase	X	X	X	X
Transmisión por cámara		X	X	

Película en red	X		X	
Vista en miniatura de las pantallas de los usuarios en tiempo real	X	X		
Gestionar servicios	X			

Los sistemas anteriormente estudiados permiten reconocer la necesidad de la investigación, ya que no poseen juegos serios como funcionalidades nativas del *software*. Estos sistemas por sus condiciones tecnológicas usan los juegos a través de sitios en internet, donde se otorga un permiso previo a los estudiantes. De esta forma se genera como principal desventaja para cualquier institución educativa, tener que adaptar su plan de enseñanza a los juegos y no viceversa. Sin embargo, sirvió como guía para la actual investigación.

1.4 Descripción de la metodología, tecnologías y herramientas a utilizar

La metodología, herramientas y tecnologías a utilizar son las seleccionadas por el proyecto para el desarrollo del *software* ATcnea. A continuación, se expone una descripción de las mismas.

1.4.1 Lenguaje de programación

Un lenguaje de programación es un lenguaje formal diseñado para dar instrucciones a una máquina, particularmente a un computador. Contiene notaciones que permiten especificar algoritmos computacionales para escribir programas (17). Un lenguaje de programación es el procedimiento de escritura del código fuente de un *software*. Muestra al programa informático, qué acción tiene que llevar a cabo y cuál es el modo de concretarla.

Java en su versión 8.0

Lenguaje de programación de propósito general concurrente, basado en clases, orientado a objetos y específicamente diseñado para tener pocas dependencias de implementación como sea posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. Posee mecanismos que garantizan la seguridad durante la ejecución (18).

1.4.2 Sistema Gestor de Bases de Datos

Los sistemas gestores de bases de datos (DBMS[9], por sus siglas en inglés), son programas que permiten organizar datos en una o más tablas relacionadas. Se utilizan en todo el mundo en una amplia variedad de aplicaciones. Prácticamente cualquier aplicación que necesite almacenamiento, accesos y análisis de datos estructurados hace uso de algún tipo de DBMS (19).

HSQLDB (*Hyperthreaded Structured Query Language Database*)

Sistema de gestión de base de datos relacional escrito en Java. Como principal ventaja tiene su velocidad y su reducido tamaño. Además, puede mantener la base de datos en memoria o en ficheros en el disco duro. Se pueden realizar las operaciones más habituales de los sistemas de gestión de bases de datos (altas, bajas, modificaciones, consultas) usando sintaxis SQL, soporta triggers e integridad referencial (20).

1.4.3 Herramienta de mapeo

Hibernate en su versión 5.1.0. final

Capa de persistencia objeto/relacional y generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos, de una manera muy rápida y optimizada en cualquiera de los entornos soportados. Se integra en cualquier tipo de aplicación justo por encima del contenedor de datos (21).

1.4.4 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE[10] por sus siglas en inglés), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusivo a un solo lenguaje o bien puede utilizarse para varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

***Intellij IDEA* en su versión *community edition* 16**

IntelliJ IDEA permite escritura de código sin complicaciones. Gracias a su profunda comprensión de los lenguajes y tecnologías, crea un entorno adecuado en el que todos los miembros del equipo pueden trabajar juntos de manera eficiente. Integración transparente

[9] DBMS: *database manager system.*

[10] IDE: *integrated development environment.*

con una amplia variedad de sistemas de control de versiones. Permite a los miembros del equipo permanecer en sincronía con los cambios de otros, asegurando que todas las contribuciones sean productivas. El IDE constantemente valida la calidad del código y ofrece soluciones inmediatas para los problemas encontrados en todos los niveles, desde la instrucción individual para arquitectura global, utilizando las inspecciones de código avanzado y análisis de matriz de dependencia (22).

***Android Studio* en su versión 2.1.3**

IDE oficial para el desarrollo de aplicaciones para *Android* que se basa en *IntelliJ IDEA*, posee un robusto editor de códigos y potentes herramientas para desarrolladores. *Android Studio* ofrece funciones que aumentan la productividad durante la compilación de aplicaciones para *Android*, como las siguientes:

- Sistema de compilación flexible basado en *Gradle*.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Integración de plantillas de código y *GitHub*, para ayudarte a compilar funciones comunes de las aplicaciones e importar ejemplos de código.
- Gran cantidad de herramientas y *frameworks* de prueba.
- Herramientas *Lint* para detectar problemas de rendimiento, uso, y compatibilidad de versión (23).

1.4.5 Tecnologías para el Marco de Trabajo de Desarrollo

JavaFX en su versión 8.0

Basado en Java**,** proporciona a los desarrolladores de la aplicación crear e implementar fácilmente aplicaciones de escritorio que se comportan de la misma forma en distintas plataformas y permite utilizar cualquier biblioteca de Java en aplicaciones JavaFX. Los desarrolladores pueden ampliar sus capacidades y utilizar la tecnología de presentación que JavaFX proporciona para crear experiencias visuales que resulten atractivas (24).

1.4.6 Herramienta para el control de versiones

La versión es el estado en el que se encuentre un producto en un momento dado de su desarrollo o modificación. El control de versiones es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o configuración del mismo (25).

Git en su versión 8.16.1

Software de control de versiones diseñado por *Linus Torvalds*[11], para la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente (26). Es un sistema de control de versiones distribuido que no depende de acceso a la red o un repositorio central.

Características de Git:

- Diseñado para manejar proyectos grandes.
- Es extremadamente rápido.
- Formato de archivo muy sencillo y compacto.
- 100% distribuido.
- Se puede sincronizar por cualquier medio.

1.4.7 Tecnología para la edición de imágenes

GIMP (***GNU Image Manipulation Program***) **en su versión 2.8**

Programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas especializadas.

1.4.8 Lenguaje Unificado de Modelado en su versión 2.0

El Lenguaje Unificado de Modelado (UML) ofrece soporte para clases abstractas, relaciones entre clases, comportamiento por iteración, empaquetamiento, entre otros. Estos elementos se pueden representar mediante tipos de diagramas como: de clases, de objetos, de casos de usos y de secuencia.

UML permite representar el sistema de forma gráfica, se pueden entender y especificar fácilmente sus características. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Permite realizar una verificación y validación del modelo realizado (27).

1.4.9 Herramienta CASE de modelado

Las herramientas CASE (*Computer Aided Software Engineering*, o Ingeniería de *Software* Asistida por Computadora) son un conjunto de métodos, utilidades y técnicas que ayudan

[11] Linus Torvalds: ingeniero de *software* finlandés conocido por iniciar y mantener el desarrollo del núcleo de Linux.

al desarrollo de *software*. Aumentan la productividad y logran una reducción en el costo de tiempo y dinero. Al utilizar estas herramientas se puede abstraer el código en un nivel donde la arquitectura y el diseño son más fáciles de entender y modificar (28).

Visual Paradigm en su versión 8.0

Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una construcción más rápida de aplicaciones, con un menor costo y mejor calidad. Permite realizar todos los tipos de diagramas de clases, obtener código inverso y generar código para una amplia gama de lenguajes. Otra de las características es que produce documentación del sistema en formato PDF y HTML.

1.4.10 Metodología de desarrollo de *software*.

Para lograr una planificación eficiente y predecir resultados durante el proceso de desarrollo de *software* han surgido varias metodologías. Estas últimas son una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de *software* en sus esfuerzos por implementar nuevos sistemas de información (29). De acuerdo a la filosofía de desarrollo, estas metodologías se clasifican en tradicionales o ágiles.

Para el desarrollo del módulo se hace uso de la metodología de desarrollo ágil AUP UCI, teniendo en cuenta que es la utilizada en la actividad productiva de la UCI y en el desarrollo del software.

El Proceso Unificado Ágil (AUP, por sus siglas en inglés) es una versión simplificada del Proceso Unificado de *Rational* (RUP, por sus siglas en inglés). Describe de manera fácil como desarrollar aplicaciones de *software* de negocio usando técnicas ágiles de XP y conceptos que aún se mantienen validos en RUP.

Esta metodología tiene como característica principal que posee tres fases que se ejecutan de manera consecutiva, siete disciplinas asociadas, once roles y cuatro posibles escenarios, que permiten su adaptación al proyecto (30).

AUP UCI permite la utilización del modelo ágil para los proyectos que necesitan, por sus características, encapsular sus requisitos funcionales en historias de usuarios, de requisitos

por procesos o casos de uso (31). De acuerdo a las características principales se trabaja sobre el cuarto escenario de la metodología AUP UCI nombrado historia de usuario.

1.5 Conclusiones parciales

- La identificación de los conceptos relacionados con aulas tecnológicas y juegos serios de palabras para el *software* ATcnea, contribuyó a un mejor entendimiento del entorno en que se desenvuelve la investigación.
- El estudio de sistemas similares arribó a que estos no brindan juegos serios como aplicaciones nativas del *software*, trayendo como desventaja para cualquier institución, tener que adaptar su plan de enseñanza a los juegos y no viceversa, por lo que se demuestra la necesidad de dicho módulo.
- Se describió la metodología, herramientas y tecnologías que permitirá guiar el proceso de diseño e implementación del módulo para juegos serios de palabras.

Capítulo 2: Descripción de la solución propuesta

En el desarrollo del *software* es de vital importancia definir los procesos que intervienen en el mismo para lograr un correcto desarrollo del módulo para juegos serios de palabras. Se identifican los requisitos con los que debe cumplir. Además de las historias de usuarios para la descripción de los mismos y la arquitectura y diseño del sistema a desarrollar.

2.1 Descripción del sistema propuesto

El módulo para juegos serios de palabras en el *software* ATcnea estará dividido en dos submódulos, un submódulo para la aplicación destinada al estudiante desarrollado utilizando la plataforma *Android* y un submódulo para la aplicación destinada al profesor desarrollado utilizando la plataforma Java.

El submódulo para la aplicación destinada al estudiante incluirá componentes reutilizables que permitirán a desarrolladores crear juegos de palabras con ahorro de trabajo y tiempo. Por otra parte, el submódulo para la aplicación destinada al profesor permitirá realizar una total gestión de los juegos, enviar a estudiantes, controlar el desarrollo y resultados de los mismos.

A continuación, se muestra el funcionamiento del módulo a través de la siguiente ilustración donde se describen cada uno de los elementos principales del módulo.

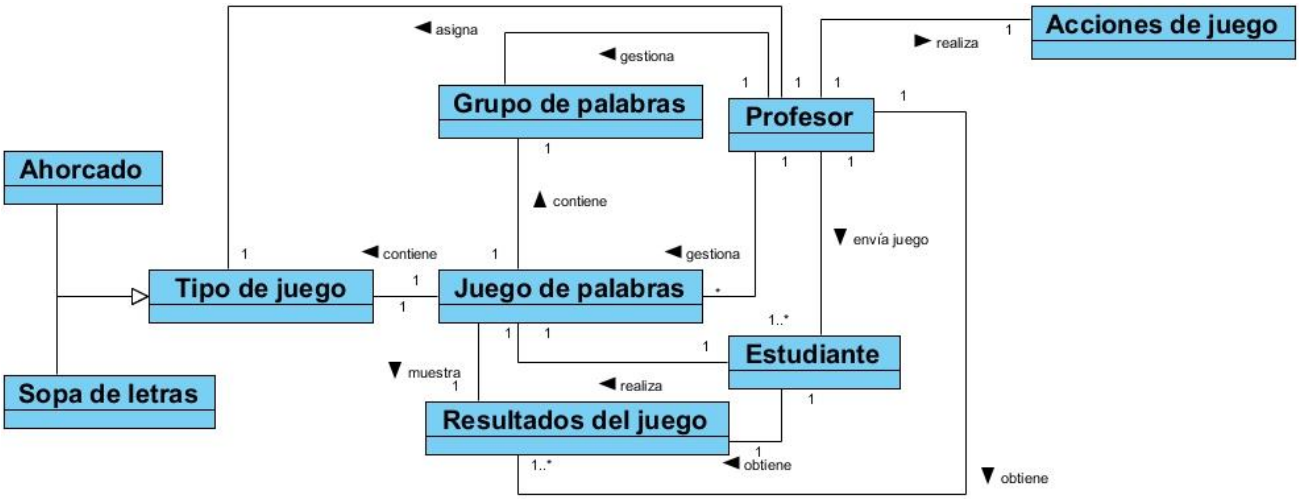


Ilustración 1 Representación del módulo para juegos serios de palabras

Resultados del juego: datos que muestra el sistema al profesor, una vez que finalice el juego.

Estudiante: usuario que puede navegar en el sistema con los permisos otorgados a un estudiante.

Profesor: usuario que puede navegar en el sistema con los permisos otorgados a un profesor.

Juego de palabras: juego creado por el profesor para enviar a los estudiantes.

Grupo de palabras: palabras que asigna el profesor en la creación de un juego.

Acciones de juego: acciones en tiempo real del juego, que puede realizar el profesor como: pausar, reanudar o finalizar.

Tipo de juego: diversos juegos de palabras disponibles para ser utilizados por el profesor.

Ahorcado: tipo de juego de palabras existente en la aplicación destinada al estudiante.

Sopa de letras: tipo de juego de palabras existente en la aplicación destinada al estudiante.

2.2 Requisitos del sistema

El proceso de desarrollo del *software* comprende en sus etapas más tempranas la definición de tareas orientadas a captar las necesidades o características para satisfacer el sistema que se desea crear o modificar. A continuación, se muestran los requisitos funcionales y no funcionales que debe cumplir el sistema creado.

2.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas (32).

Tabla 3 Requisitos funcionales

No.	Requisito	Descripción	Prioridad	Complejidad
RF1	Crear juego de palabras	El sistema debe permitir al profesor crear un juego de palabras e incluir los siguientes datos: (*) Nombre	Alta	Alta

		(*)Grupo de palabras (*) Tipo de juegos (*) Horas (*) Min (*) Partida automática (*) Filas (*) Columnas		
RF2	Modificar juego de palabras	El sistema debe permitir al profesor editar un juego de palabras. (*) Nombre (*) Grupo de palabras (*) Tipos de juegos (*) Horas (*) Min (*) Partida automática (*) Filas (*) Columnas	Alta	Alta
RF3	Eliminar juego de palabras	El sistema debe permitir al profesor eliminar uno o varios juegos.	Media	Baja
RF4	Enviar juego de palabras	El sistema debe permitir al profesor enviar un juego de palabras a los estudiantes.	Alta	Media
RF5	Listar juego de palabras	El sistema debe permitir listar todos los juegos existentes.	Media	Media

RF6	Buscar juego de palabras	El sistema debe permitir al profesor buscar un juego.	Media	Media
RF7	Visualizar juego de palabras	El sistema debe permitir al profesor visualizar los datos del juego.	Media	Baja
RF8	Crear grupo de palabras	El sistema debe permitir al profesor crear un grupo de palabras, e incluir los siguientes datos: (*) Nombre de grupo (*) Palabras	Alta	Alta
RF9	Modificar grupo de palabras	El sistema debe permitir al profesor editar un grupo de palabras, e incluir los siguientes datos: (*) Nombre de grupo (*) Palabras	Media	Media
RF10	Eliminar grupo de palabras	El sistema debe permitir al profesor eliminar uno o varios grupos de palabras.	Media	Baja
RF11	Visualizar grupo de palabras	El sistema debe permitir al profesor visualizar un grupo de palabras.	Media	Media
RF12	Listar grupo de palabras	El sistema debe permitir listar todos los grupos creados.	Media	Media
RF13	Buscar grupo de palabras	El sistema debe permitir al profesor	Media	Media

		buscar un grupo de palabras.		
RF14	Listar estudiantes conectados	El sistema debe permitir listar los estudiantes conectados a la clase.	Alta	Media
RF15	Buscar estudiantes conectados	El sistema debe permitir al profesor buscar un estudiante conectado.	Media	Media
RF16	Eliminar resultados de juego	El sistema permite al profesor eliminar uno o varios resultados del juego.	Media	Media
RF17	Visualizar resultado de juego	El sistema debe permitir al profesor visualizar un resultado de juego.	Media	Media
RF18	Buscar resultado del juego	El sistema debe permitir al profesor buscar un resultado.	Media	Media
RF19	Listar resultados del juego	El sistema debe permitir listar los resultados del juego.	Media	Media
RF20	Jugar tipo de juego Sopa de letras	El sistema debe permitir al estudiante jugar el juego Sopa de letras.	Alta	Alta
RF21	Jugar tipo de juego Ahorcado	El sistema debe permitir al estudiante jugar el juego Ahorcado.	Alta	Alta
RF22	Reanudar partida	El sistema debe permitir al profesor reanudar la partida de juego.	Media	Media

RF23	Pausar partida	El sistema debe permitir al profesor pausar la partida de juego.	Media	Media
RF24	Finalizar partida	El sistema debe permitir al profesor finalizar la partida de juego.	Media	Media
RF25	Listar estudiantes jugando	El sistema debe permitir listar todos los estudiantes con juegos disponibles en sus dispositivos.	Media	Media
RF26	Mostrar ayuda al estudiante	El sistema permite mostrar al estudiante la ayuda del juego.	Media	Media

2.2.2 Requisitos no funcionales.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares (32).

Tabla 4 Requisitos no funcionales

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Conformidad
Objetivo	Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad.
Origen	UCI
Artefacto	Interfaces del sistema.
Entorno	En línea
Estímulo	

1.a Realizar una acción en el sistema.	**Respuesta: Flujo de eventos (Escenarios)**
Se crea o inserta/edita/elimina un elemento.	Se muestra un mensaje con el resultado de la acción.
1.b Informar al usuario	
Se ubica sobre un botón que se identifica por un ícono/hipervínculo.	El color del elemento cambia.
Se solicita información del usuario.	Cada campo debe tener asociado un pequeña ayuda para facilitar la entrada de datos por parte del usuario.
Medida de respuesta	
1. Si se crea o inserta/edita/elimina un elemento se debe mostrar un mensaje con el resultado de la acción. 2. El color de los íconos/hipervínculos debe cambiar si el usuario se ubica encima de ellos.	

Atributo de Calidad	Fiabilidad
Sub-atributos/Sub-características	Tolerancia a fallos
Objetivo	Capacidad del módulo para operar según lo previsto en presencia de fallos de *hardware* o *software*.
Origen	Interno al sistema/ externo al sistema.
Artefacto	Canales de comunicación/Almacenamiento persistente.
Entorno	Operación normal/Modo degradado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Fallo por omisión/accidente	

Se interrumpen las comunicaciones de red entre el ordenador del profesor y los terminales de los estudiantes.	El sistema debe ser capaz de notificarle al usuario la pérdida de conexión.
Medida de respuesta	
Tiempo para notificarle al usuario la pérdida de conexión en no más de 10 segundos.	

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-característica	Jugabilidad
Objetivo	Capacidad del producto que demuestra la calidad del juego en términos de sus reglas de funcionamiento y diseño, sería todo aquello que hace el estudiante.
Origen	Estudiante
Artefacto	Las pantallas pertenecientes a todos los mecanismos del juego.
Entorno	ATcnea
Estímulo **1. a. Interactividad del jugador en el juego**	**Respuesta: Flujo de eventos (Escenarios)**
Interactuar con el entorno del juego.	1. El estudiante se desplaza por el entorno con el objetivo de encontrar las palabras indicadas. 2. El estudiante obtiene puntos y avanza en el juego a medida que encuentra la palabra y el tiempo no termine. 3. El estudiante tiene la oportunidad de consultar ayuda.
Medida de respuesta	
Interactuar con el juego.	

Atributo de Calidad	Portabilidad
Sub-atributos/Sub-características	Adaptabilidad
Objetivo	Capacidad del producto que permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de *hardware*, *software*, operacionales o de uso.
Origen	Arquitecto de *software*.
Artefacto	Todo el sistema
Entorno	Sistema desplegado
Estímulo **1. a. Capacidad del módulo de adaptarse de forma efectiva al *software* ATcnea**	Respuesta: Flujo de eventos (Escenarios)
El sistema está diseñado con tecnologías que permiten que este se adapte al *software* ATcnea.	N/A
Medida de respuesta	
El ambiente de despliegue del *software* ATcnea.	

Atributo de Calidad	Eficiencia del desempeño.
Sub-atributos/Sub-característica	Utilización de recursos.
Objetivo	Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de *software*.
Artefacto	Todo el sistema.

<table>
<tr><td>Entorno</td><td>Sistema desplegado.</td></tr>
<tr><td>Estímulo
1. a. Características de Hardware</td><td>Respuesta: Flujo de eventos (Escenarios)</td></tr>
<tr><td>Aplicación destinada al profesor:
CPU Core 2 E6300 o superior.
4GB RAM o superior.
Capacidad del Disco Duro de 2 Giga.
Acelerador gráfico de 256 Mb
Red de cable:
10Mbytes/100Mbytes/1000Mbytes Red compartida o Red conmutada
Red inalámbrica:
802.11b/g/n Red inalámbrica.
Aplicación destinada al estudiante:
. Se recomienda 1 GB RAM o superior
7 pulgadas o superior de pantalla
CPU/GPU Dual Core 1.0GHz o superior.
Red inalámbrica:
802.11b/g/n Red inalámbrica.</td><td>1. El módulo para que funcione correctamente en el software ATcnea.</td></tr>
<tr><td>Estímulo
1. a. Características de Software</td><td>Respuesta: Flujo de eventos (Escenarios)</td></tr>
<tr><td>Permitir la instalación de la aplicación destinada al estudiante en tablets que cuenten con NovaDroid.
Permitir la instalación de la aplicación destinada al profesor en Nova 2015 o superior.</td><td>1. El módulo para que funcione correctamente en el software ATcnea.</td></tr>
<tr><td colspan="2">Medida de respuesta</td></tr>
</table>

Hacer uso del módulo para juegos serios de palabras.

Atributo de Calidad	Eficiencia del desempeño.
Sub-atributos/Sub-característica	Comportamiento temporal.
Objetivo	Grado en que los tiempos de respuesta, procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de *software*.
Artefacto	Módulo para juegos serios de palabras.
Entorno	El sistema desplegado.
Estímulo **1. a. Tiempo de respuesta ante alguna acción realizada por el usuario**	**Respuesta: Flujo de eventos (Escenarios)**
Se realiza la acción de envió, pausa, resumen y finalización de juego.	El sistema muestra respuesta a las acciones realizadas en un tiempo mínimo de 5 segundos.
Medida de respuesta	
Realizar acciones en el módulo para juegos serios de palabras.	

2.2.3 Descripción de Historias de Usuario

Una historia de usuario es una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales, comúnmente utilizada en las metodologías de desarrollo ágiles para la especificación de requisitos.

En el cuarto escenario de AUP, en su variante UCI los requisitos se administran en historias de usuarios por lo que para la solución propuesta se describieron un total de 26 Historias de usuario, de las cuales se presenta la correspondiente al RF1: Crear juego de palabras. Las historias de usuario restantes generadas se encuentran en el Anexo 2.

Tabla 5 Historia de usuario Crear juego de palabras

Número: 1	Nombre del requisito: Crear juego de palabras.
Programador: Danger Germán Fernández Bastida.	Iteración asignada: 1ra
Prioridad: Alta	Tiempo estimado:4 días
Riesgo en desarrollo: N/A	Tiempo real: 5 días

Descripción:

1- Objetivo:

Crear juegos de palabras en el sistema.

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

Para crear un juego de palabra hay que:

- Estar autenticado en el sistema con usuario profesor.
- Debe existir al menos un tipo de juego.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Los campos siguientes son obligatorios.

Nombre del juego: campo de texto que inicia con mayúscula, longitud no mayor de 30 caracteres, solo caracteres alfanuméricos.

Tipo de juego: campo de selección que debe de tener como mínimo un juego de palabras.

Grupo de palabras: campo de selección debe de tener como mínimo un grupo de palabras.

Horas: campo de selección que no admite estar vacío.

Min: campo de selección que no admite estar vacío.

En caso de que se utiliza el tipo de juegos Sopa de letras además de esos campos se tienen que llenar los siguientes:

Filas: campo de selección que no admite estar vacío.

Columnas: campo de selección que no admite estar vacío.

El campo siguiente no es obligatorio.

Partida automática: campo de selección.

4- Flujo de la acción a realizar:

- El sistema debe permitir crear un juego de palabra, esta acción puede realizarse al seleccionar la opción Crear juego que se muestra en la vista Administrar juego.
- El usuario crea el juego, selecciona la opción Aceptar, se muestra un mensaje de información en la vista Administrar juego.
- Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción.
- Si selecciona la opción Cancelar regresa a la vista previa.

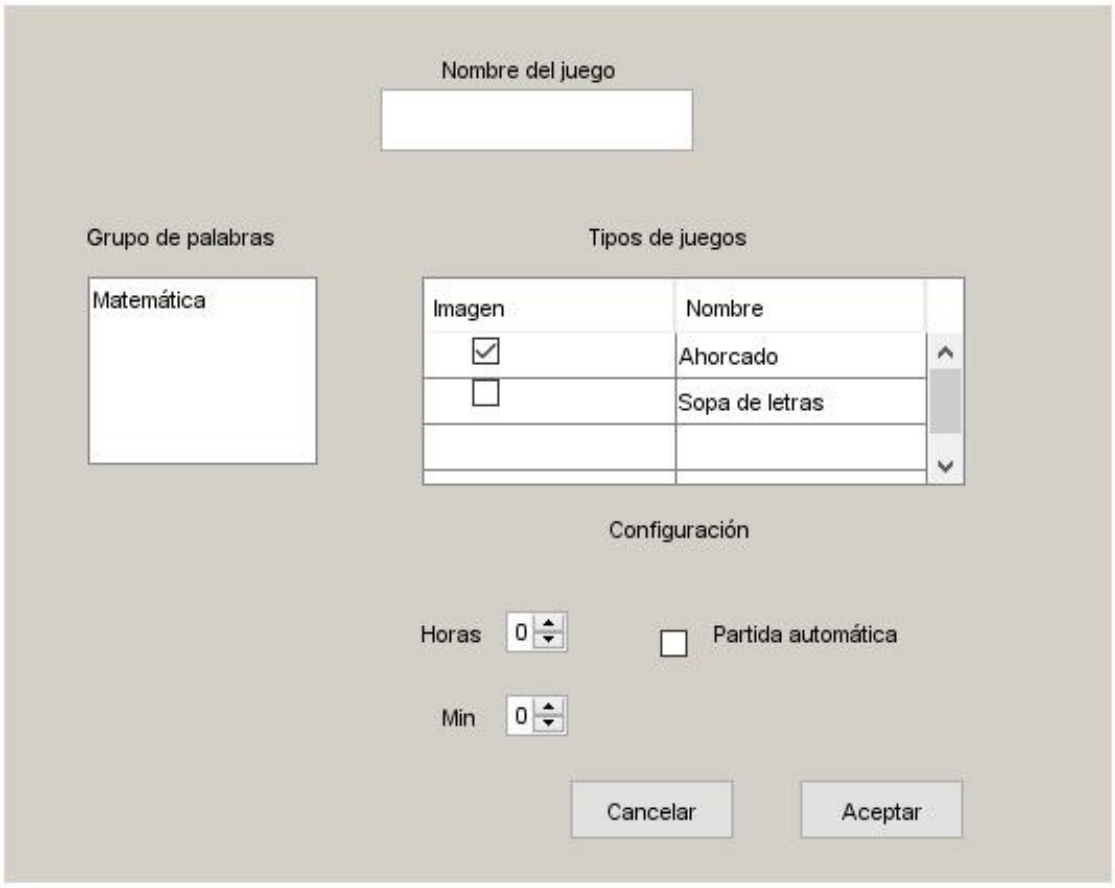
Observaciones

Esta acción solo puede ser realizada por el profesor.

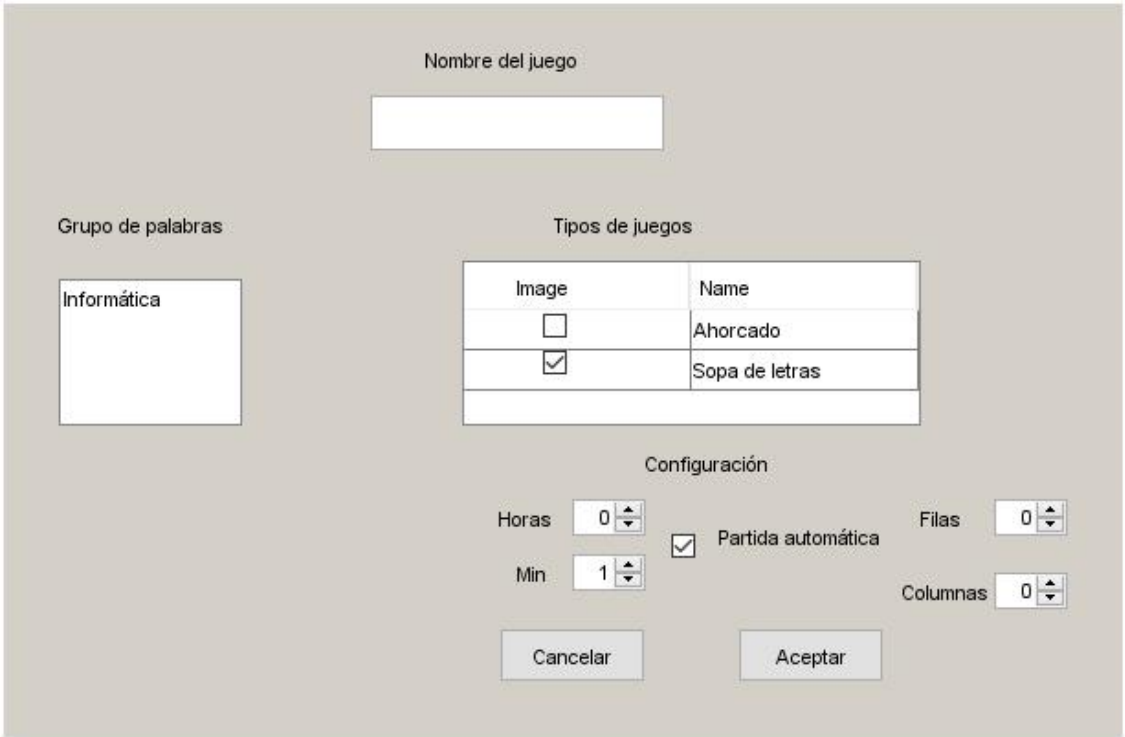
Prototipo de interfaz: Crear juego de palabras.

Vista Administrar juego de palabras.

Para juego Ahorcado.



Para juego Sopa de letras.



2.3 Arquitectura

La arquitectura de *software* de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del *software*, sus propiedades externas visibles y las relaciones entre ellos (33).

Para la realización del módulo se utilizó como patrón arquitectónico Modelo Vista Controlador (MVC). Su elección se fundamenta en lo siguiente:

MVC divide el sistema en tres componentes, como su nombre lo específica, modelo, vista y el controlador:

- Modelo: es la representación de la información con la cual el sistema opera, gestiona todos los accesos a dicha información, como consultas o actualizaciones. Las peticiones de acceso o manipulación de información llegan al Modelo a través del Controlador.
- Vista: es la interfaz gráfica que se muestra a los usuarios.
- Controlador: es el encargado de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

Condicionado por el uso del *framework* JavaFX, y la arquitectura por defecto que usan las aplicaciones nativas de Android. El patrón arquitectónico MVC utilizado en el desarrollo del módulo para juegos serios de palabras presenta la particularidad, que la Vista no se comunica con el Modelo y la función del Controlador es la de mediar entre ambos. De tal forma que una Vista no puede ser creada sin asociarle un Controlador. A continuación, se ejemplifica a través de la ilustración lo antes expuesto.

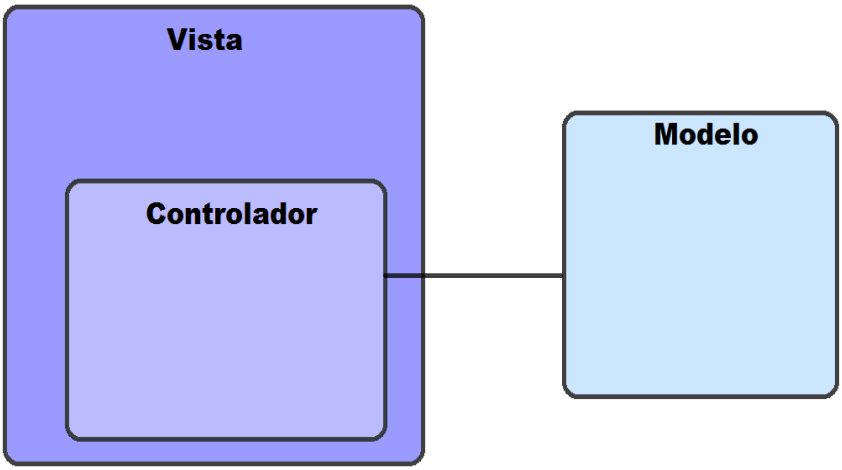


Ilustración 2 MVC

2.4 Patrones de diseño

Son soluciones a problemas repetidos en la construcción de *software* y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos (34). A continuación, se muestra una breve descripción de los patrones usados, así como la forma en que se utilizaron en la propuesta de solución.

2.4.1 Patrones GRASP

Los Patrones de Asignación de Responsabilidades (GRASP[12] por sus siglas en inglés) describen los principios fundamentales del diseño de clases y la asignación de responsabilidades, expresados como patrones (35). A continuación, se describen los utilizados durante el desarrollo del módulo.

- Creador: asigna a la clase A la responsabilidad de crear una instancia de la clase B (35). Se ejemplifica su uso en la ilustración 10 donde la clase *ManageGameScreenController* se le asigna la responsabilidad de crear una instancia de la clase *AddGameScreenController*.
- Experto: expresa que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (35). Se evidencia su uso en la ilustración 10, donde la clase *WordsGame* es la responsable en conocer toda la información necesaria de un juego de palabras.

[12] GRASP: *General Responsibility Assignment Software Patterns.*

- Alta cohesión: se refiere al grado o la fuerza con que se relacionan los elementos. Un elemento con alta cohesión, realiza tareas relacionadas entre sí (35). En pocas palabras, para lograr alta cohesión debemos agrupar funciones similares en clases individuales. Se evidencia su uso en la ilustración 10, donde la clase *WordsGame* agrupa todas las funcionalidades relacionadas con un juego de palabra.
- Bajo acoplamiento: plantea la idea de tener las clases lo menos ligadas entre sí. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (35). Se evidencia su uso en la ilustración 10.
- Controlador: establece una clara separación entre la interfaz de usuario y núcleo de procesamiento de la aplicación, donde se halla la lógica de negocio. La idea básica es crear una clase que implemente métodos dedicados a escuchar o atender los eventos del sistema (35). Se evidencia su uso en la ilustración 10, en la cual la clase controladora *AddGameScreenController* maneja las peticiones o eventos de la interfaz gráfica *CreateGame*.

2.4.2 Patrones GOF

Los patrones GOF[13] favorecen la reutilización de código y ayudan a construir un *software* basado en la reutilización (36). A continuación, se describen los utilizados durante el desarrollo de módulo para juegos serios de palabras.

- *Singleton* (instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto (36). Se evidencia su uso en la siguiente ilustración.

[13] GOF: *Gang of Four*.

Ilustración 3 Ejemplo de uso del patrón Singleton

- *Templates* (patrón plantilla): es un patrón de comportamiento que define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, en donde los pasos serán redefinidos en las subclases, tiene como objetivo definir un esqueleto para un algoritmo (36). Se evidencia su uso en la siguiente ilustración.

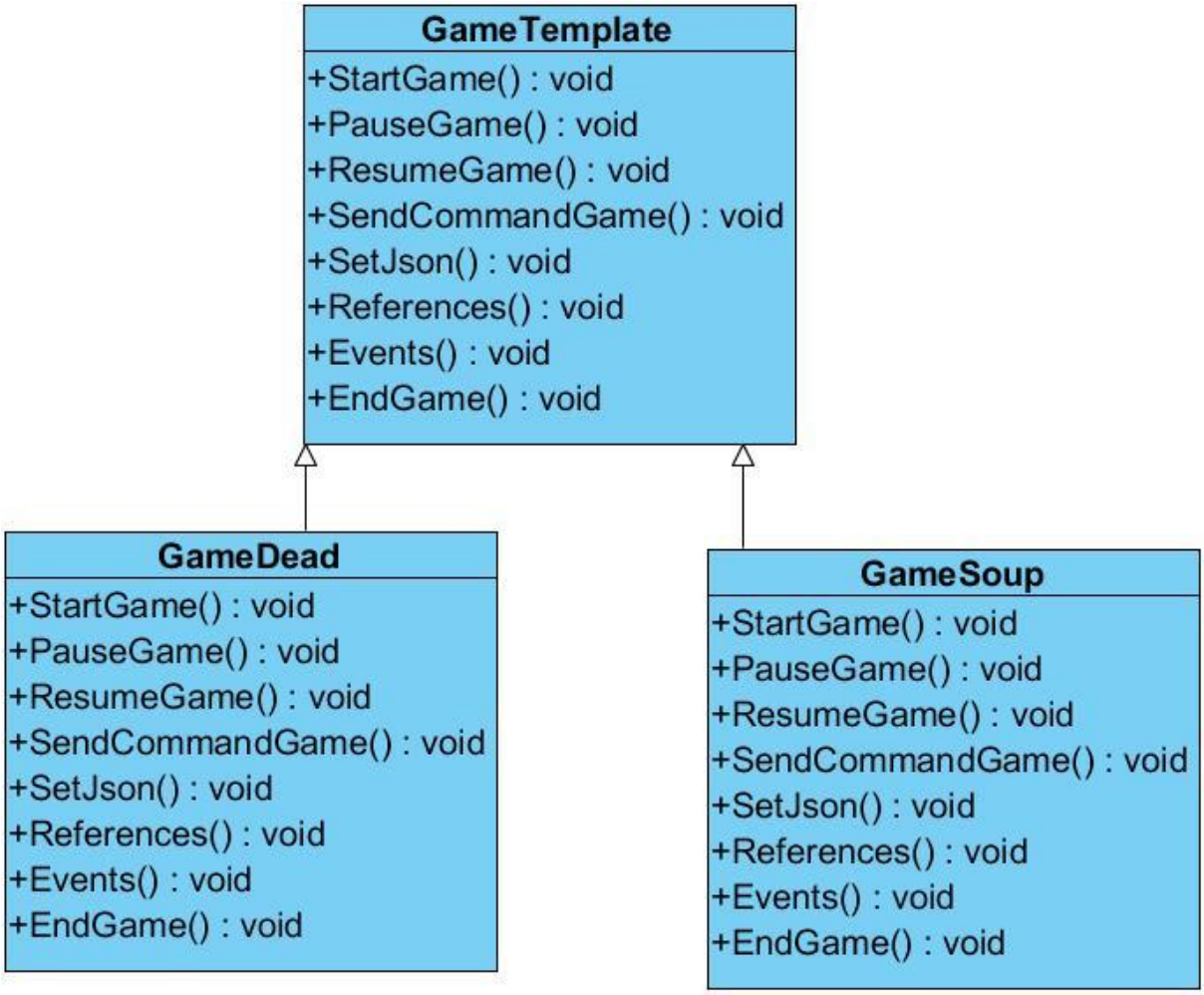


Ilustración 4 Ejemplo de uso del patrón Templates

2.5 Modelo de diseño

Representa todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, constituyendo la entrada principal a las actividades de la fase de implementación

A continuación, se describe de manera general el significado de los principales elementos presentes en el diagrama de clases del diseño que corresponde al módulo para juegos serios de palabras.

Paquete Vista: contiene las clases que muestran la información al usuario.

Paquete Controlador: contiene las clases que realizan el tratamiento de eventos.

Paquete Modelo: contiene las entidades generadas en correspondencia con las tablas de la base de datos que almacena toda la información que maneja el módulo.

Útil: contiene las clases auxiliares relacionadas con los controladores.

Se muestra el diagrama de clases del diseño que corresponde al RF1: Crear juego de palabras. Los diagramas de clases del diseño restantes generados se encuentran en el Anexo 3.

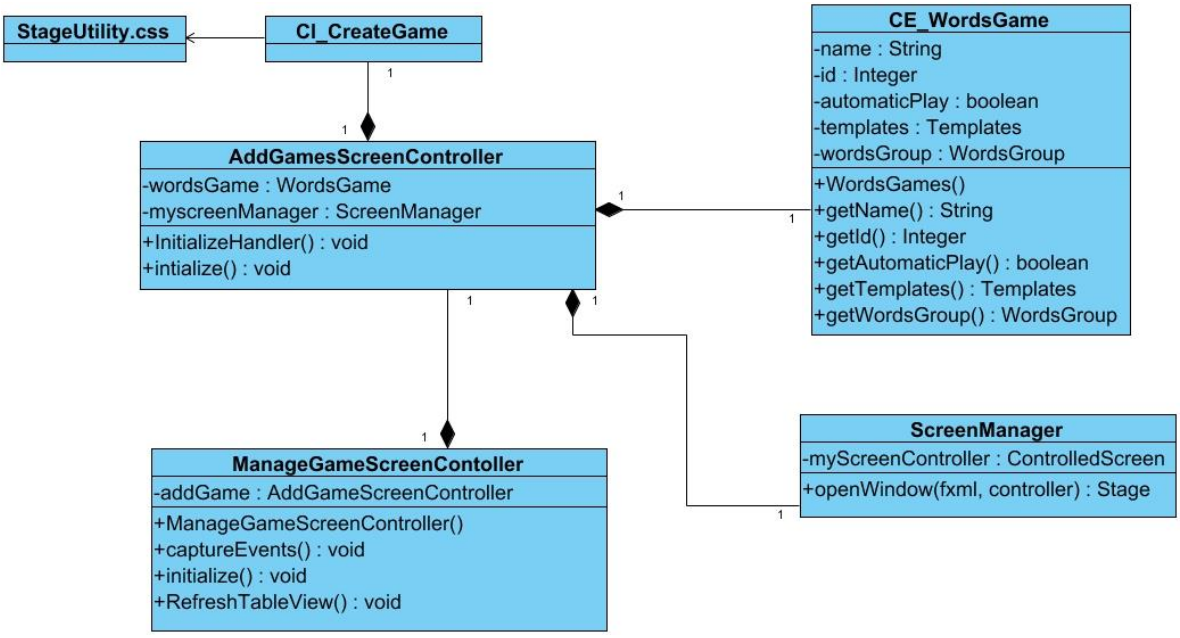


Ilustración 5 Diagrama de clase del diseño Crear juego de palabras

2.6 Modelo de datos

Modelo abstracto que organiza elementos de datos y estandariza como se relacionan entre sí con las propiedades del mundo real (37).

Para implementar la solución propuesta se añadieron cinco tablas nuevas al modelo de datos de ATcnea. En la siguiente ilustración se presentan las tablas utilizadas en la base de datos del módulo para juegos serios de palabras.

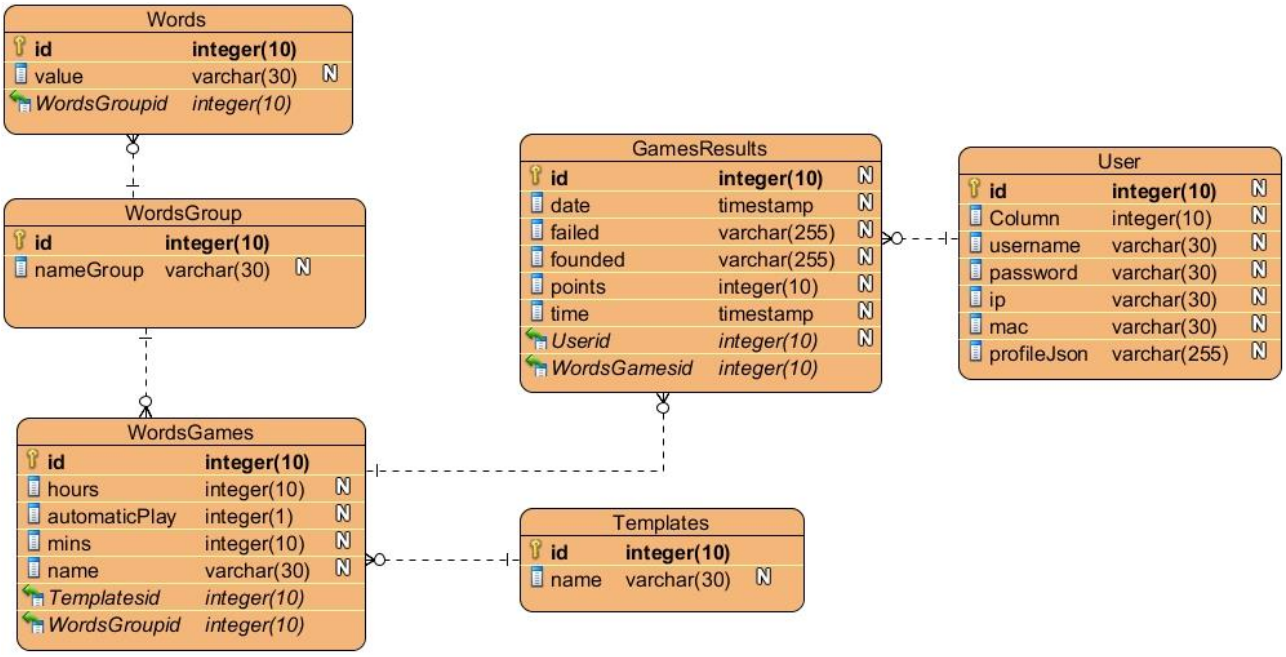


Ilustración 6 Modelo de datos

En la tabla 6 se muestra la descripción de una de las tablas creadas en el modelo de datos. Las descripciones de las tablas restantes se encuentran en el Anexo 4.

Tabla 6 Descripción de la tabla WordsGame

Words_Games		
Atributo	**Tipo**	**Descripción**
id	Integer (10)	Etiqueta única que identifica al juego de palabras.
automaticPlay	Integer (1)	Almacena modo de ejecución del juego de palabras.

Name	Varchar (30)	Almacena el nombre del juego de palabras.
Hours	Integer (10)	Almacena las horas del juego de palabras.
Mins	Integer (10)	Almacena los minutos del juego de palabras.

2.7 Diagrama de despliegue

Modela la arquitectura en tiempo de ejecución y muestra la disposición física de los nodos que componen el sistema. Presenta la configuración de los elementos de *hardware* (nodos) y muestra como los elementos del *software* se encuentran conectados por enlaces de comunicación (33). A continuación, se muestra el referente al módulo para juegos serios de palabras y se describen sus componentes.

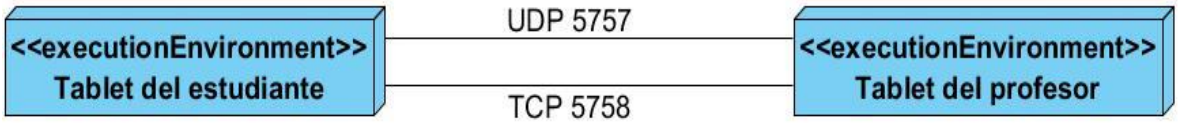


Ilustración 7 Diagrama de despliegue

Tablet del estudiante: dispositivo que utiliza el estudiante para interactuar con la clase.

PC del profesor: dispositivo que utiliza el profesor para guiar la clase.

UDP: protocolo a través del cual el tablet del estudiante encuentra las clases disponibles, mediante el puerto 5757.

TCP: protocolo a través del cual la PC del profesor envía datos a los estudiantes conectados a la clase mediante el puerto 5758.

2.8 Conclusiones parciales

- La identificación de los requisitos funcionales y no funcionales del sistema, y su administración mediante historias de usuarios, sirvió de guía para la implementación de las distintas funcionalidades del módulo para juegos serios de palabras.
- La selección del patrón arquitectónico Modelo Vista Controlador permitió generar una estructura de sistema robusta y establecer una correcta relación entre sus componentes.
- El uso de los patrones de diseño permitió dar solución a problemas en la construcción del *software*, estandarizar y reutilizar código, para un mejor entendimiento y desarrollo del módulo.
- Los artefactos generados durante el diseño de la solución contribuyeron a un mejor entendimiento del sistema para dar paso a la implementación de la solución propuesta.

Capítulo 3: Desarrollo de la solución propuesta

En el presente capítulo se definen los componentes utilizados en la implementación del módulo y los paquetes en los que estará dividido. Se muestran imágenes de las principales interfaces de la solución propuesta. Además, se exponen los resultados de las pruebas de *software* realizadas como demostración del correcto funcionamiento del sistema. Se persigue como objetivo fundamental evaluar la calidad del producto desarrollado y garantizar que el módulo para juegos serios de palabras diseñado e implementado cumpla con las funcionalidades previamente definidas.

3.1 Diagrama de componentes

El diagrama de componentes muestra los elementos de diseño de un sistema de *software*. Permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces (38). A continuación, se muestran los diagramas de componentes asociados a la aplicación destinada al estudiante y a la aplicación destinada al profesor.

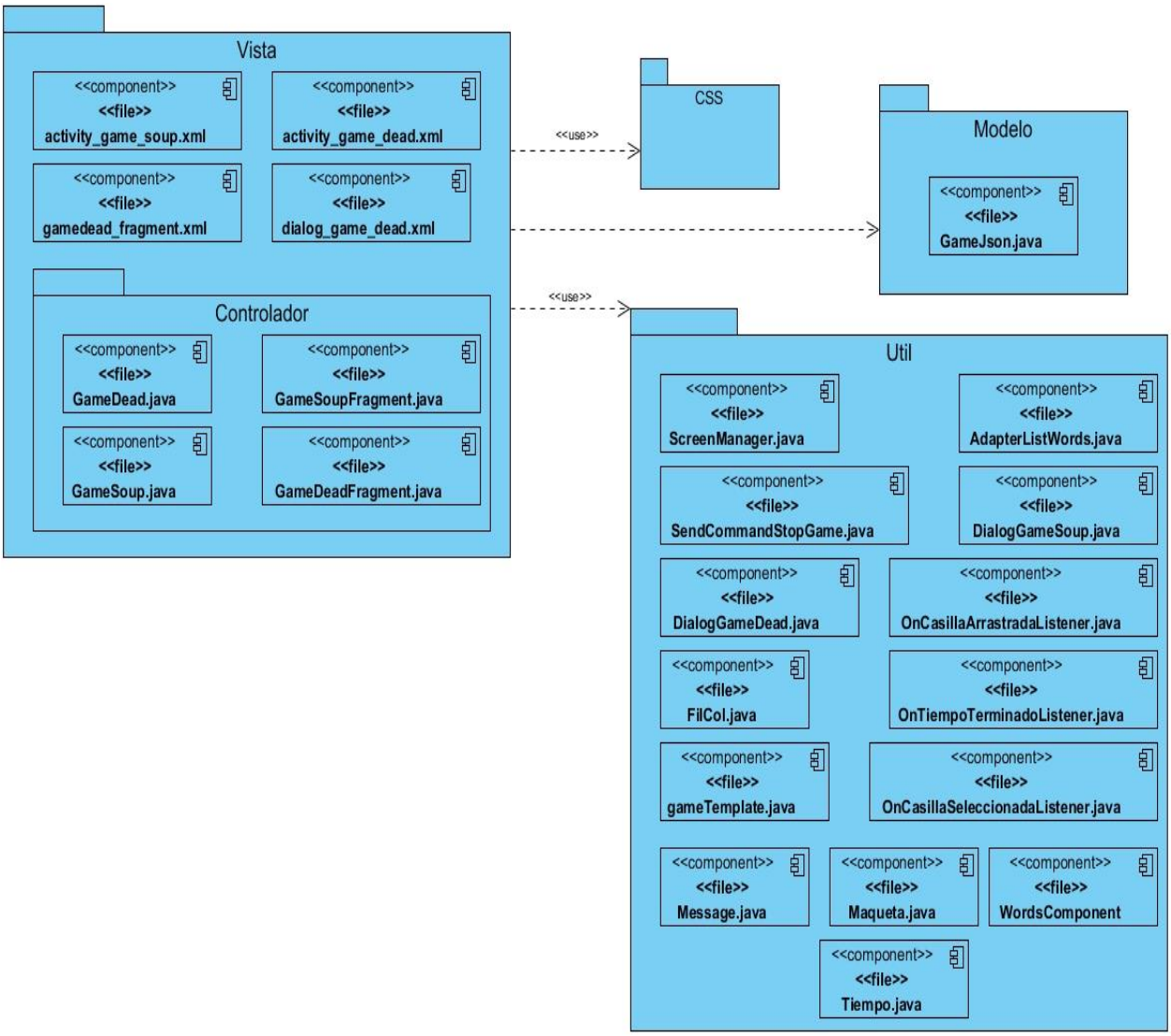


Ilustración 8 Diagrama de componentes para la aplicación destinada al estudiante

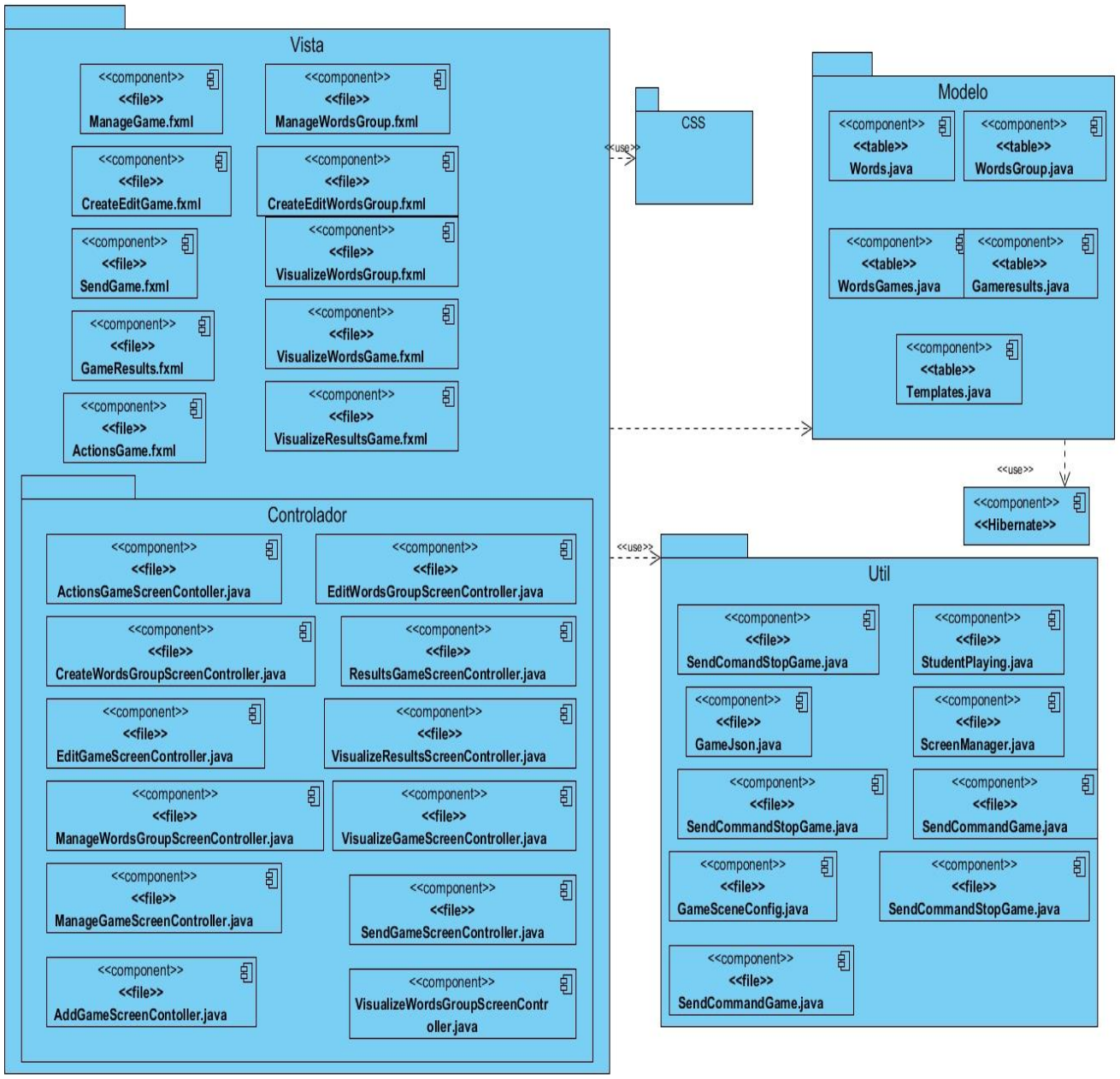


Ilustración 9 Diagrama de componentes para la aplicación destinada al profesor

3.2 Estándares de codificación

Inicialización: intentar inicializar las variables locales al ser declaradas. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

Colocación: poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}".) No esperar al primer uso para declararlas; puede confundir a programadores no preavisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

void myMethod() {

int int1 = 0; // comienzo del bloque del método

if (condición) {

int int2 = 0; // comienzo del bloque del "if"

...

}

}

La excepción de la regla son los índices de bucles for, que en Java se pueden declarar en la sentencia for.

for (int i = 0; i < maximoVueltas; i++) {}

Evitar las declaraciones locales que ocultan declaraciones de niveles superiores, por ejemplo, no declarar la misma variable en un bloque interno:

int cuenta;

...

miMetodo () {

if (condición) {

int cuenta = 0; // evitar esto

...

}

...

}

Declaraciones de clases e interfaces: al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- La llave de cierre "}" empieza una nueva línea para ajustarse a su sentencia de apertura correspondiente, excepto si no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{".

class Ejemplo extends Object {

int ivar1;

int ivar2;

Ejemplo (int i, int j) {

ivar1 = i;

ivar2 = j;

}

int metodoVacio () {}

...

}

Sentencias if, if-else, if else-if else: la clase de sentencias if-else debe tener la siguiente *forma:*

if (

condición

) {

sentencias;

}

if (

condición

) {

sentencias;

} else {

sentencias;

}

if (

condiciòn

) {

sentencia;

} else if (

condición

) {

sentencia;

} else {

sentencia;

}

Nota: las sentencias if usan siempre llaves {}, evitar la siguiente forma, propensa a errores:

if (condición)

sentencia; //evitar esto

Sentencias for: una sentencia for debe tener la siguiente forma:

for (inicialización; condición; actualización) {

sentencias;

}

Variables: Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje. Los nombres de las variables deben ser cortos, pero con significado. Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales.

3.3 Principales pantallas de la aplicación destinada al profesor

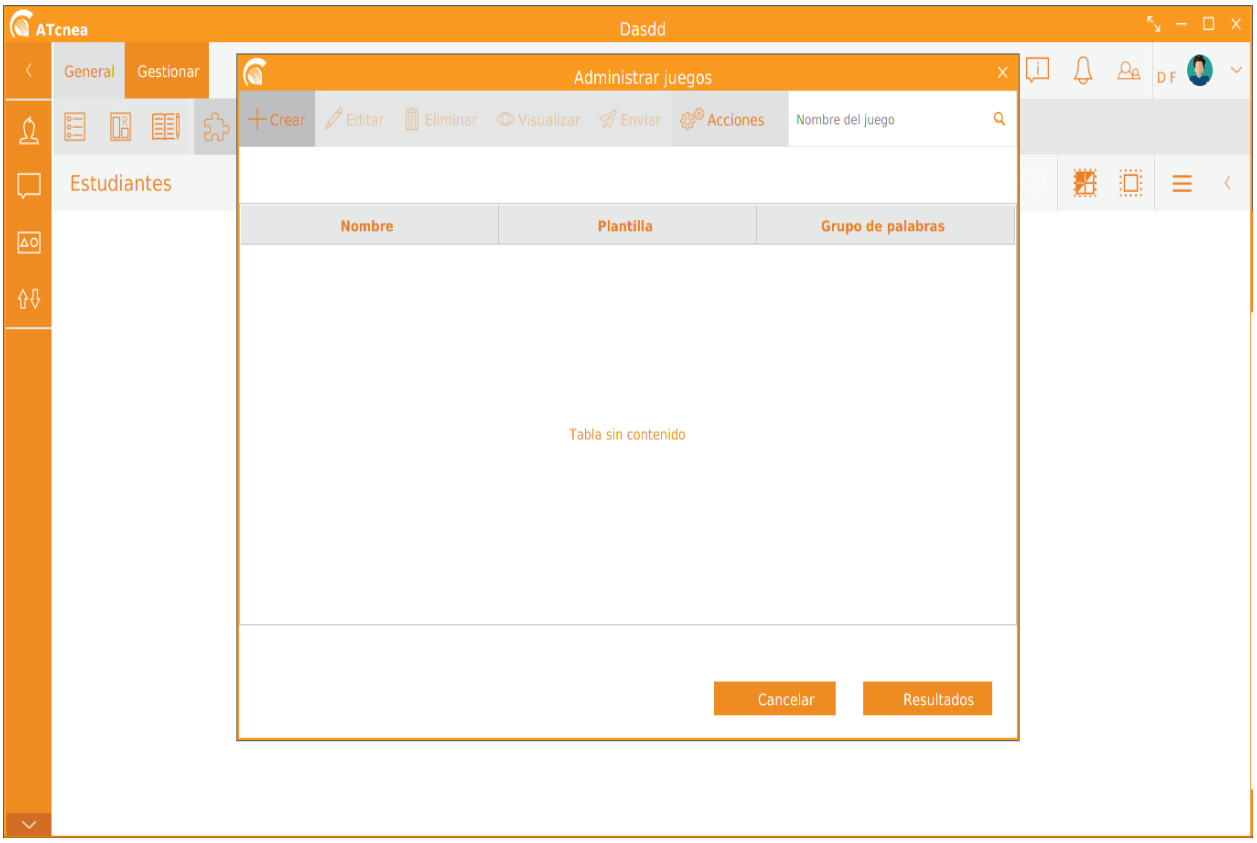


Ilustración 10 Interfaz Administrar juegos de palabras

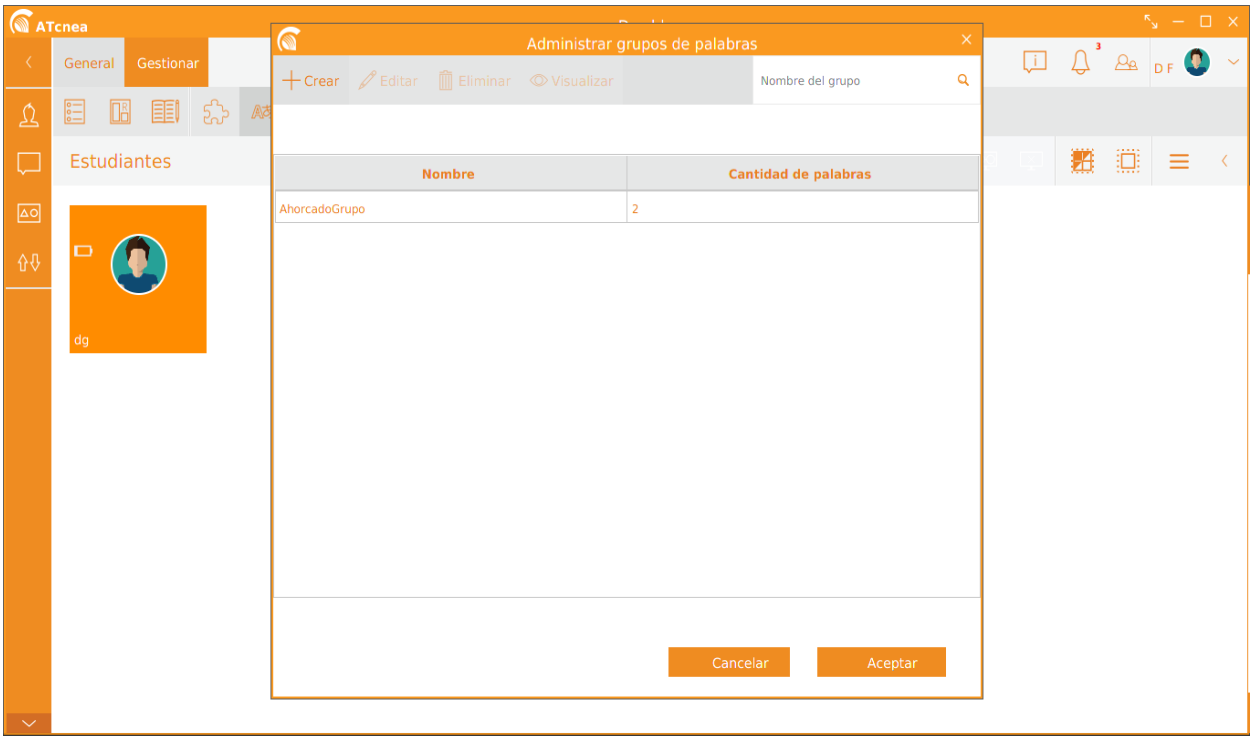


Ilustración 11 Interfaz Administrar grupos de palabras

Ilustración 12 Interfaz Crear grupo de palabras

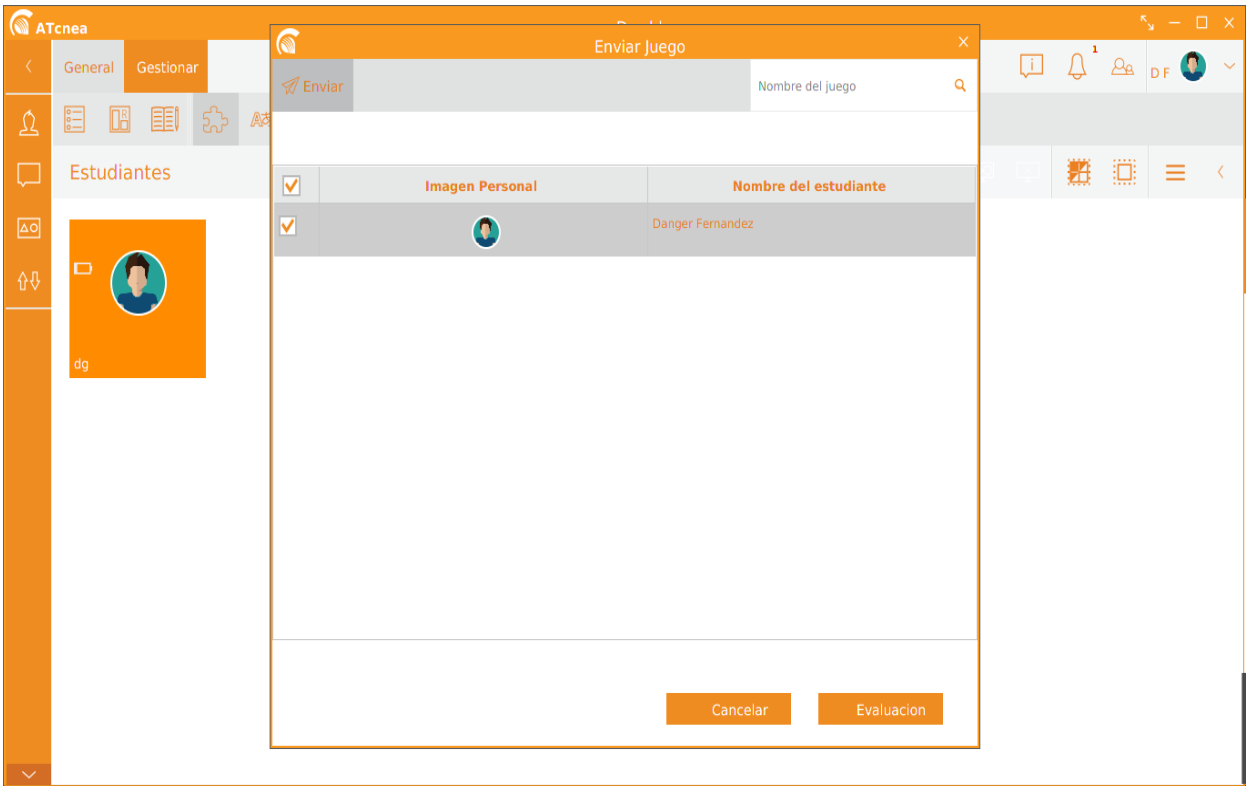


Ilustración 13 Interfaz Enviar juego de palabras

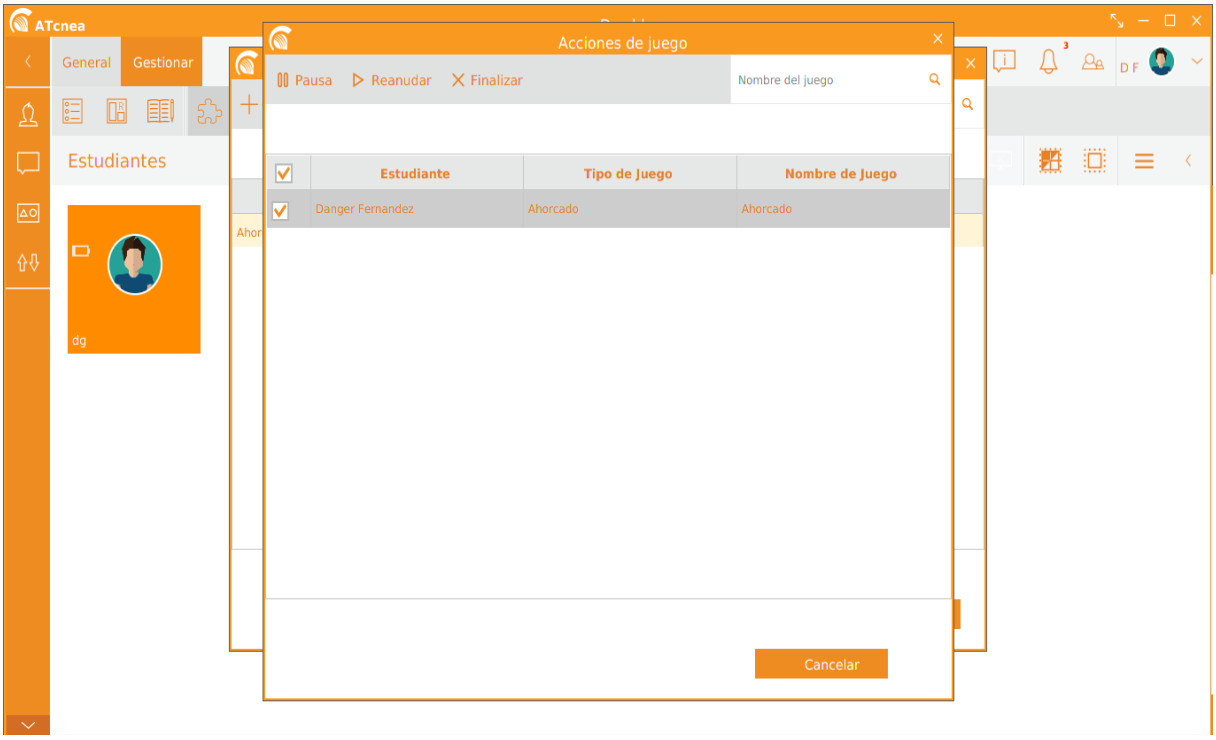


Ilustración 14 Interfaz Acciones de juego

3.4 Principales pantallas de la aplicación destinada al estudiante

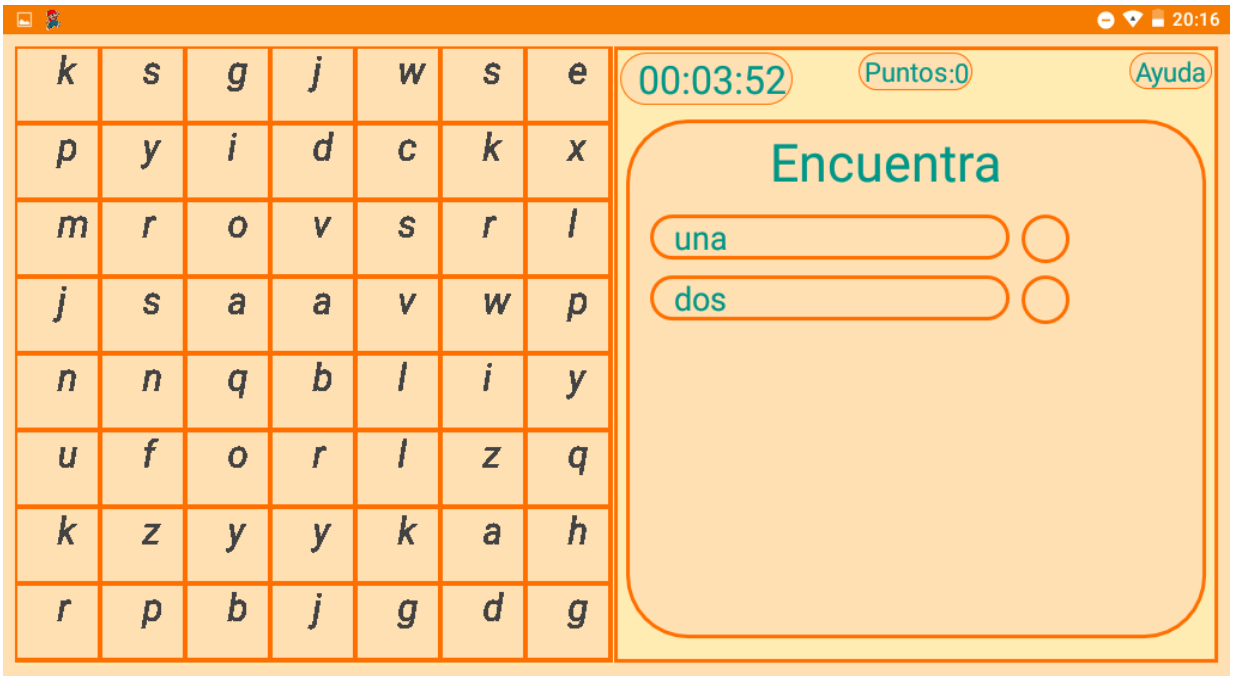


Ilustración 15 Interfaz tipo de juego sopa de letras

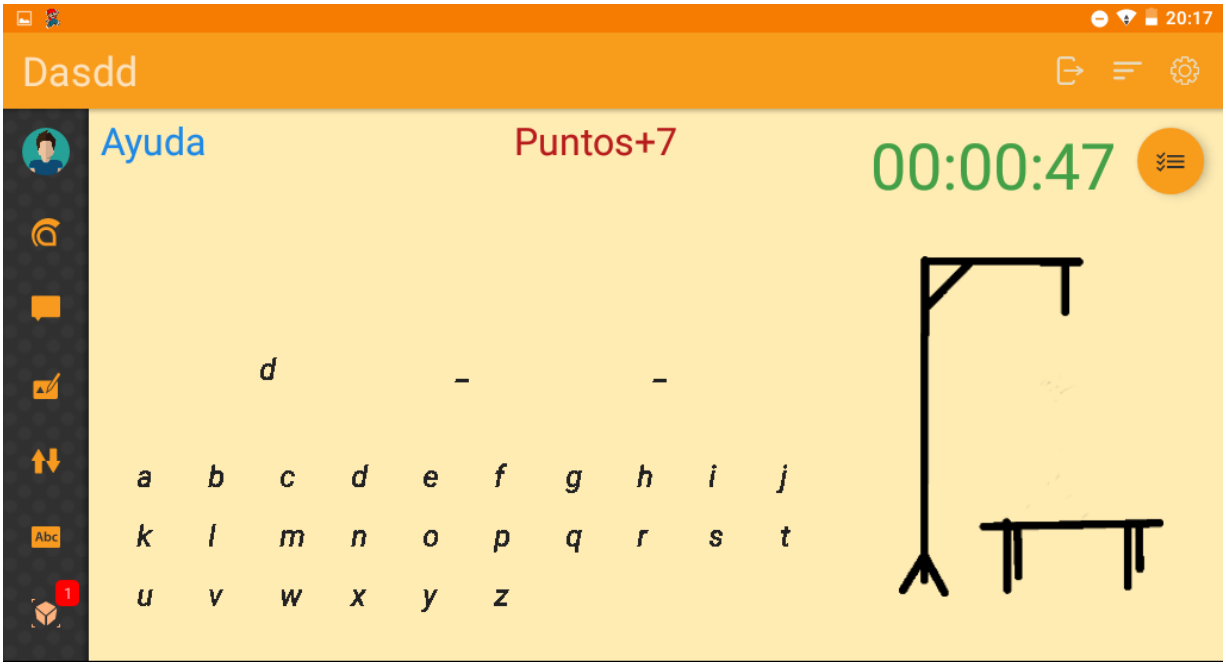


Ilustración 16 Interfaz tipo de juego ahorcado

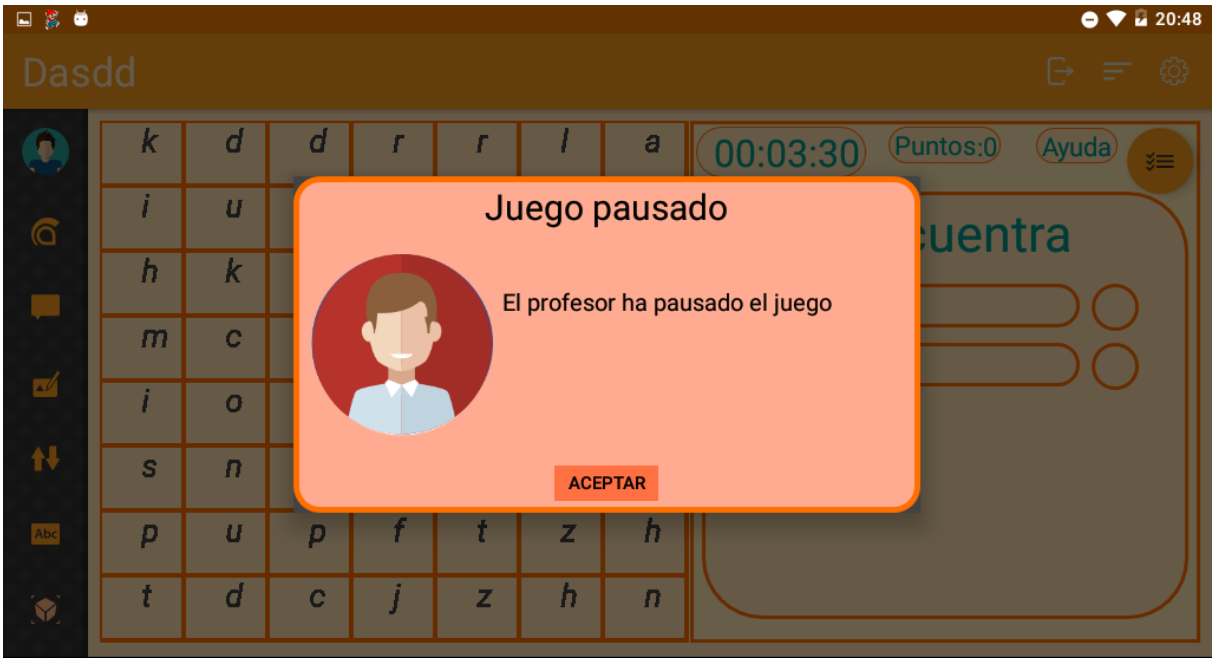


Ilustración 17 Interfaz juego pausado por el profesor

3.5 Pruebas de *software*

Para que el módulo se considere listo debe someterse previamente a una etapa de pruebas rigurosa, con el fin de analizar si cumple con las funcionalidades requeridas.

Las pruebas de *software* son un elemento importante dentro del ciclo de vida de un proyecto. Permiten detectar defectos, verificar que todos los requisitos se hayan implementado correctamente y comprobar la integración adecuada de los componentes (32).

3.5.1 Tipos de pruebas

Son diferentes formas de verificar y validar un producto de *software*. Es un proceso que conlleva la realización de un conjunto de tareas a lo largo del ciclo de vida del sistema.

De acuerdo con el estándar IEEE 1012-1986 el conjunto mínimo de pruebas que se deben realizar son (39):

- Prueba modular, prueba unitaria o prueba de componentes.
- Prueba de integración.
- Prueba del sistema.
- Prueba de aceptación.

A continuación, se plantean las pruebas realizadas al módulo para juegos serios de palabras con el fin de mostrar la calidad del mismo.

Prueba unitaria: Forma de probar que un módulo de código funcione correctamente. Ello permite garantizar que cada uno funcione de manera eficiente por separado (40).

Prueba de integración: se realiza a medida que los diferentes módulos del sistema se integran en el mismo. Ya se han realizado pruebas unitarias o de unidad, y todos los componentes están correctamente implementados. El objetivo fundamental de esta prueba es comprobar que las interfaces entre los distintos módulos son correctas. Para realizar pruebas de integración a un sistema es necesario establecer una estrategia de integración. A continuación, se muestran dichas estrategias.

- De big-bang: se integran todos los componentes y entonces se prueba el sistema como un todo. Esta estrategia se basa en acoplar todos los módulos del proyecto de una vez con el objetivo de reducir la cantidad de pruebas.
- De arriba abajo (top-down): consiste en empezar la integración y la prueba por los módulos que están en los niveles superiores de abstracción, e integrar incrementalmente los niveles inferiores.
- De abajo a arriba (bottom-up): consiste en empezar la integración y la prueba por los módulos que están en los niveles inferiores de abstracción, e integrar incrementalmente los niveles superiores (33).

Prueba de sistema: son las pruebas que verifican el comportamiento del sistema en su conjunto. Funcionan para verificar que los elementos del sistema se hayan integrado de manera adecuada y que se realicen las funciones asignadas (33). De manera general este nivel de prueba es preparado y ejecutado por un grupo independiente al desarrollador, y consiste en validar que el *software* cumpla con los requerimientos especificados por el cliente.

Prueba de rendimiento: se realiza para estudiar el comportamiento de un sistema en el instante que se ve sometido a acciones realizadas por período de tiempo. De igual forma para evaluar un sistema si varios procesos son ejecutados simultáneamente (33).

Prueba de aceptación: se realiza una vez que el sistema se ha implantado en su entorno real de funcionamiento, y su objetivo es demostrar al usuario que el sistema satisface sus necesidades (33).

3.5.2 Métodos de pruebas

Un método de prueba es un procedimiento definitivo que produce un resultado de prueba. Se puede probar cualquier producto de ingeniería en dos formas: conociendo la función específica para la cual fue diseñado el mismo y conociendo el funcionamiento del producto. Al primer enfoque se le denomina prueba de caja negra y al segundo, prueba de caja blanca (33).

Pruebas de caja blanca: es donde se comprueban los componentes internos. Según Pressman, se basan en un examen detallado de los procedimientos y caminos lógicos del sistema para determinar si el estado real coincide con el esperado (41).

Pruebas de caja negra: son nominadas pruebas funcionales o de comportamiento y se centran en los requisitos funcionales del *software*. Se llevan a cabo sobre la interfaz del *software* buscando errores en cada una de las funcionalidades (41).

Para validar la propuesta de solución se empleó el método de caja blanca que permitió determinar si el estado real coincide con el esperado, y el de caja negra para comprobar la validez en las respuestas de las funcionalidades, antes las acciones del usuario y la calidad de las salidas en dependencia de las entradas.

3.5.3 Diseño de casos de prueba

Es una parte de las pruebas de componentes y sistemas en las que se diseñan entradas y salidas esperadas para probar el sistema (32). A continuación, se presenta el diseño del caso de prueba perteneciente a la historia de usuario Crear juego de palabras, el resto de los artefactos de este tipo se encuentran en el Anexo 5.

Tabla 7 Caso de prueba Crear juego de palabras

Escenario	Descripción	Nombre del juego	Tipos de juegos	Grupo de palabras	Min	Horas	Filas	Columnas	Partida automática	Respuesta del sistema	Flujo central
	En la vista	N/A		N/A						El sistema	

EC 1.1 Opción Crear	Administrar juego el usuario selecciona la opción Crear.		N/A		N/A	N/A	N/A	N/A	N/A	debe permitir especificar los siguientes datos: (*) Nombre del juego (*) Tipo de juego. (*) Grupo de palabras. (*) Min. (*) Horas. (*) Partida automática (*) Filas. (*) Columnas.	Administrar Juegos/Crear juegos
EC 1.2 Opción Aceptar	Una vez especificados los datos para crear un juego el usuario selecciona la opción Aceptar.	V	V	V	V	V	V	V	V	El sistema crea el juego. Actualiza y muestra el listado de los juegos.	Administrar Juegos/Crear/ Aceptar
EC 1.3 Opción Cancelar	El usuario selecciona la opción Cancelar.	N/A	N/A	N/A	N/A	N/A			N/A	El sistema muestra la interfaz Administrar juegos**.**	Administrar juegos/Crear/ Cancelar
EC 1.4 Datos vacíos	Al especificar los datos para	I	V	V	V	V	V	V	V	El sistema muestra un mensaj	Administrar Juegos/Agregar
		V	I	I	I	I	I	I	I		
		I	I	V	I	I	I	I	I		
		I	I	I	V	I	I	I	I		

	crear un juego de palabras el usuario deja datos vacíos.	I	I	I	I	V	I	I	I	e de información.	
		I		I							
			I		I	I	V	I	I		
EC 1.5 Datos incorrectos	El sistema especifica datos incorrectos.	V	I	I	I	I	I	I	I	El sistema muestra un mensaje de información.	Administrar Juegos/Aceptar
		I		I							
			V		I	I	I	I	I		
		I	I	V	I	I	I	I	I		
		I	I	I	V	I	I	I	I		
		I	I	I	I	V	I	I	I		
		I	I	I	I	I	V	I	V		

Tabla 8 Descripción de las variables del caso de prueba Crear juego de palabras

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre del juego	Campo de texto	No	Inicia con mayúscula, longitud no mayor de 30 caracteres, solo caracteres alfanuméricos.
2	Grupo de palabras	Campo de selección	No	Debe ser seleccionado
3	Tipos de juegos	Campo de selección	No	Debe ser seleccionado
4	Horas	Campo de selección	No	Debe ser seleccionado
5	Min	Campo de selección	No	Debe ser seleccionado
6	Partida automática	Campo de selección	No	No es obligatorio
7	Filas	Campo de selección	No	Debe ser seleccionado
8	Columnas	Campo de selección	No	Debe ser seleccionado

3.5.4 Resultados obtenidos

Resultados de las pruebas unitarias

Para la realización de pruebas unitarias al código del módulo para juegos serios de palabras se utilizó el *framework* JUnit en su versión 4.12 y el *framework* Mockito en su versión 2.8.

JUnit es un conjunto de clases que permite realizar la ejecución de código Java de manera controlada, para poder evaluar si los funcionamientos de cada uno de los métodos de una clase se comportan correctamente. Es decir, en función de algún valor de entrada se evalúa

el valor de retorno esperado; si el valor obtenido es correcto, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba. En caso de que el valor esperado sea diferente al valor obtenido, JUnit devolverá un fallo en el método correspondiente (42).

Mockito es un *framework* de prueba de código abierto para Java. Permite la creación de objetos dobles en pruebas de unidades automatizadas con el propósito de desarrollo impulsado por pruebas o desarrollo impulsado por comportamientos. Aplica la técnica de prueba donde los componentes reales se sustituyen por objetos que tienen un comportamiento predefinido (43).

Durante la implementación de las funcionalidades se realizaron pruebas unitarias, donde las no conformidades encontradas se resolvieron a medida que estas fueron aplicadas. A continuación, se ejemplifica una de las pruebas unitarias realizadas a la clase Maqueta.java en la aplicación destinada al estudiante, específicamente el método Llenar (char [] arr), donde el resultado de la prueba arroja no conformidades y posteriormente fueron resueltas.

Ilustración 18 Prueba unitaria en la aplicación destinada al estudiante fallida

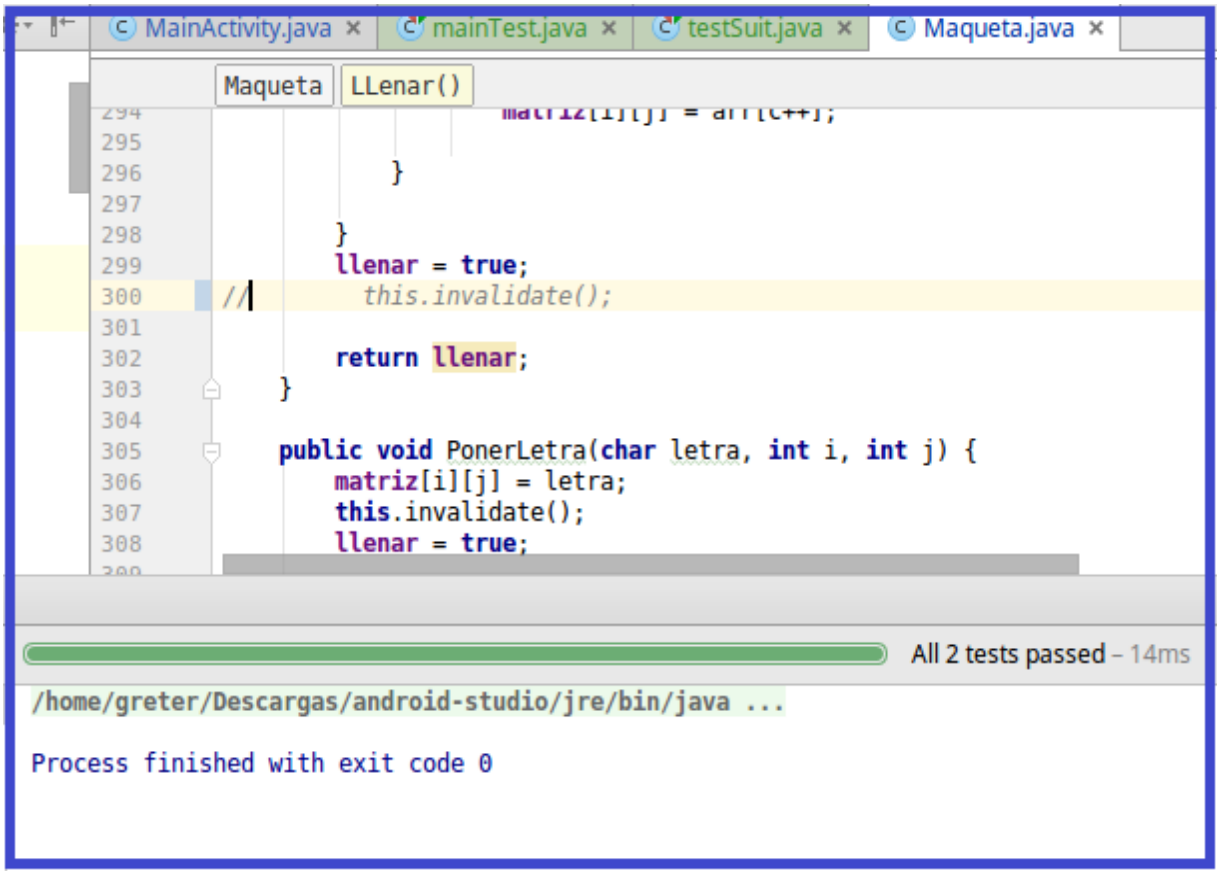


Ilustración 19 Prueba unitaria en la aplicación destinada al estudiante con éxito

Resultados de las pruebas de integración

Para la aplicación de las pruebas de integración al módulo para juegos serios de palabras en el *software* ATcnea, se utilizó la estrategia de *big-bang*. Apoyando la elección en las siguientes ventajas y desventajas:

Ventajas

- Útil para la detección de errores en el instante que se encuentren todos los módulos en construcción.
- Aplicable antes de la entrega del proyecto para evaluar el trabajo de todo el sistema con diversos escenarios.
- Se puede evaluar la interacción entre módulos para agilizar los procesos.
- Es apta para aplicar diversos escenarios para poder analizar el trabajo de todos los módulos en diversas situaciones (44).

Desventajas

- Solo es posible aplicarla hasta esté avanzado el proyecto.
- Se tienen que trabajar con todos los módulos, no se puede hacer un análisis individual.
- Si el proyecto es de corto plazo no se tiene mucho tiempo para la aplicación de la prueba y realizar las correcciones.

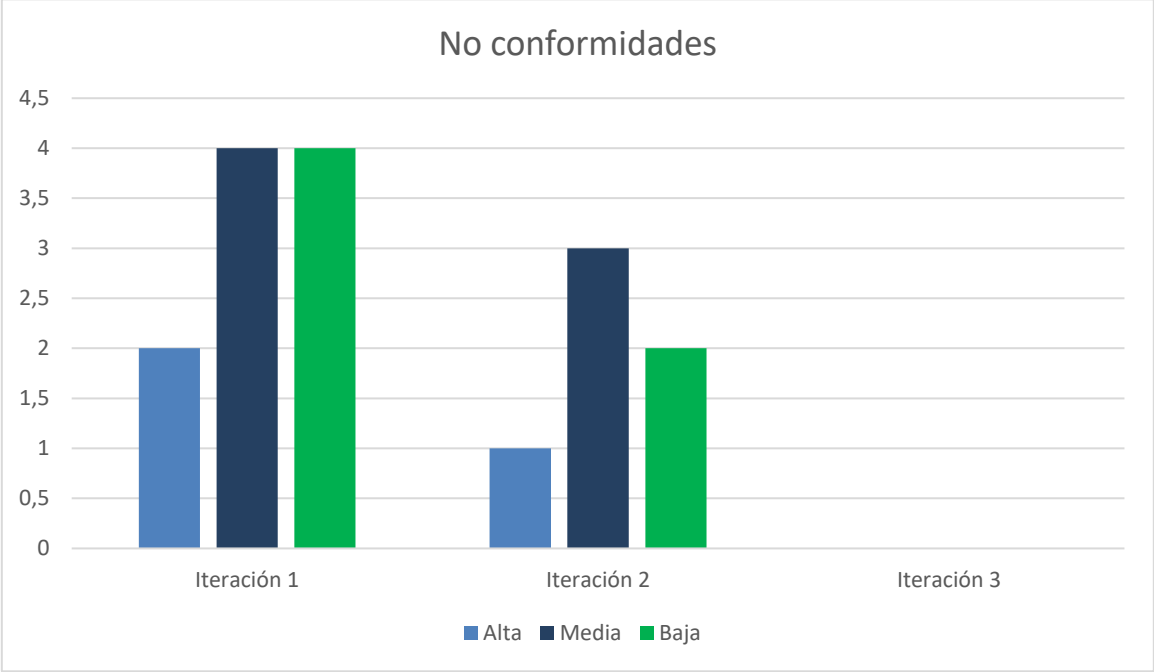


Ilustración 20 Resultados de las pruebas de integración

En la ilustración anterior se muestra la ejecución de tres iteraciones de pruebas de integración, donde en cada iteración fueron resueltas cada una de las no conformidades identificadas, hasta llegar a la total corrección de las mismas. Las no conformidades detectadas se dividieron en prioridad alta, media y baja. Los principales problemas resueltos fueron:

- El acceso a estructuras de datos globales.
- La duplicidad en nombres de variables.
- La actualización de datos.
- La no uniformidad en el diseño de la interfaz.

Resultados de las pruebas de rendimiento

Con el objetivo de evaluar el comportamiento del sistema cuando está sometido a una carga que actúa de manera concurrente se realizaron las siguientes pruebas de rendimiento:

Pruebas de estabilidad: se realizó con el objetivo de comprobar que no existía degradación del servicio por un uso prolongado del sistema. Es decir, el sistema debe funcionar sin incidencias durante 24 horas. Se realizó una sola iteración, donde no se identificaron problemas en la estabilidad del sistema.

Pruebas de Capacidad: se realizó en la aplicación destinada al estudiante con el objetivo de encontrar los límites de funcionamiento del sistema después de integrar el módulo. Las pruebas se centraron en 3 recursos fundamentales del sistema como: CPU, memoria RAM y red. Para el desarrollo de esta prueba se usó como herramienta el medidor de rendimiento Android Monitor del IDE Android Studio. A continuación, se muestra una imagen de la prueba realizada.

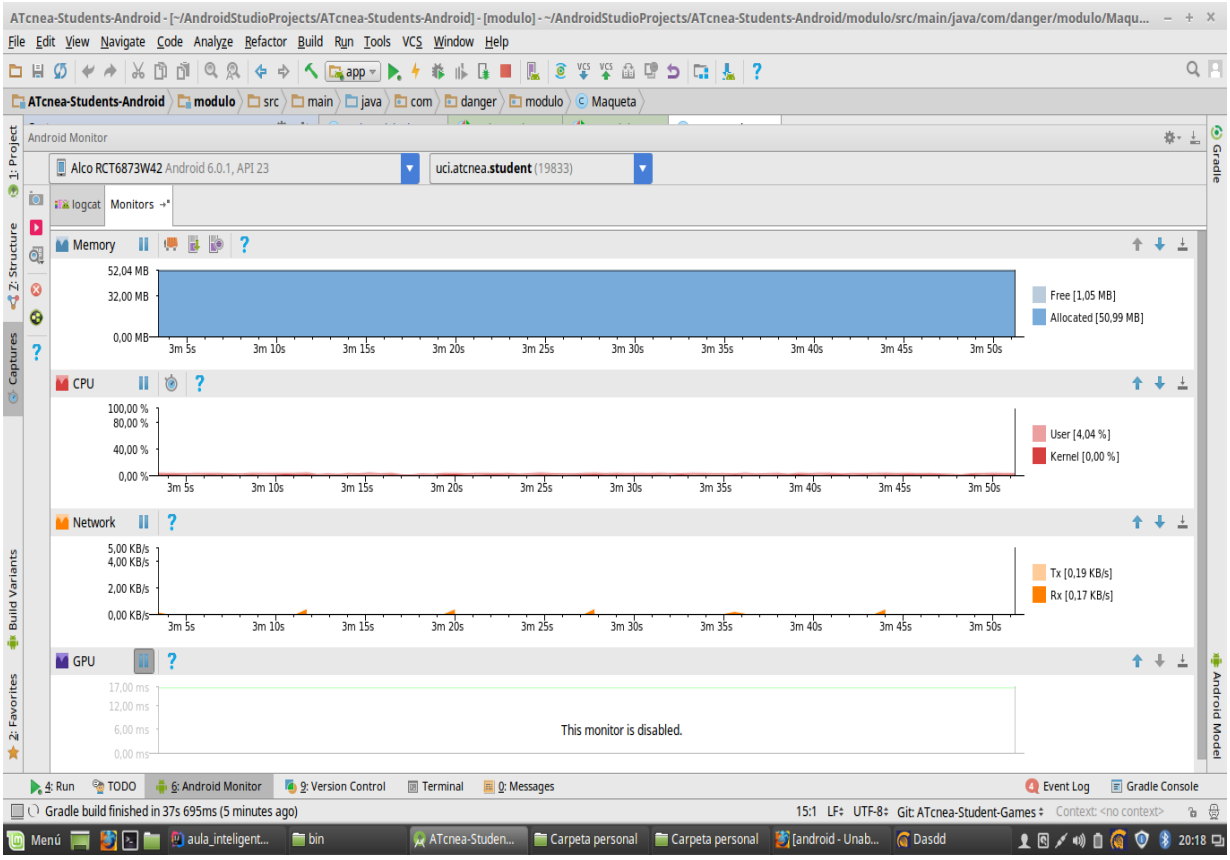


Ilustración 21 Prueba de capacidad

Como se evidencia en la imagen anterior el consumo de los recursos de sistema CPU, memoria RAM y red se mantuvieron constantes durante el período de ejecución del módulo. De esta forma no se identificaron no conformidades en la primera y única iteración de pruebas de capacidad.

Resultados de las pruebas de sistema

Para la aplicación de este tipo de prueba se empleó el método de caja negra. Este se ejecutó sobre las interfaces gráficas del sistema haciendo uso de los casos de pruebas, donde se muestran los resultados obtenidos con la ejecución de las mismas.

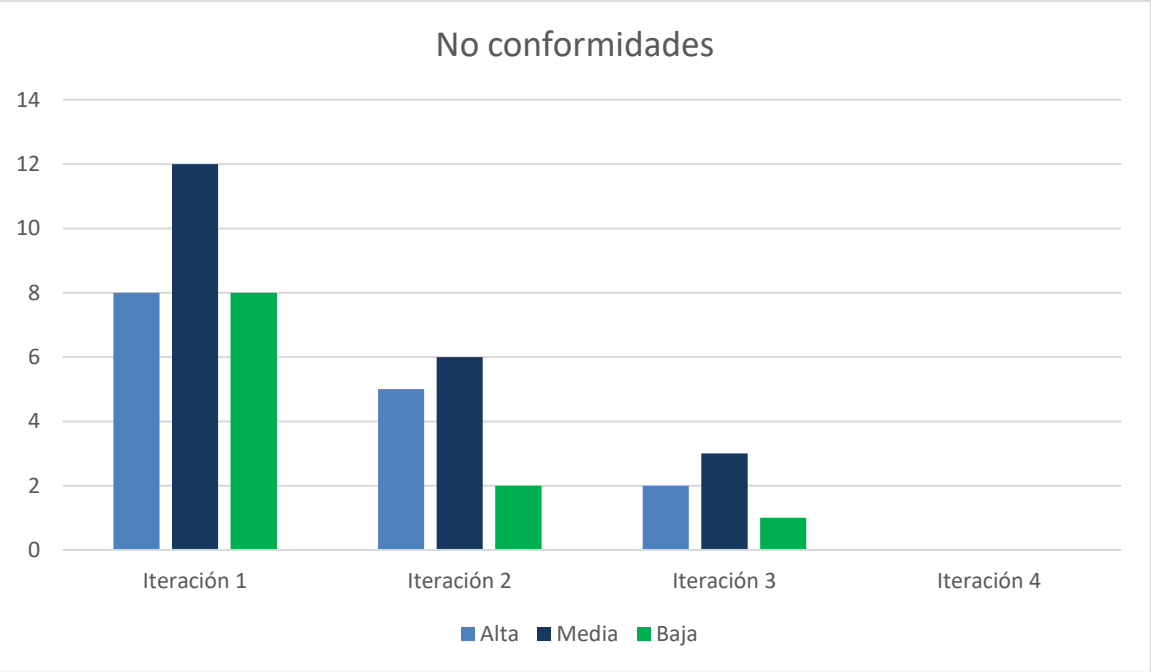


Ilustración 22 Resultados de las pruebas de sistema

En la ilustración anterior se muestra la ejecución de cuatro iteraciones, donde en cada iteración fueron resueltas cada una de las no conformidades identificadas.

3.6 Conclusiones parciales

- La creación de un diagrama de componentes permitió realizar un desglose de los diferentes elementos de *software* que componen el módulo para un mejor entendimiento de cómo está conformado.
- La realización de las pruebas unitarias, de integración, de sistema y de rendimiento permitieron la validación del módulo para juegos serios de palabras tras identificar y corregir las no conformidades encontradas.

Conclusiones Generales

Una vez concluida la presente investigación en la cual se realizó la implementación del módulo para juegos serios de palabras en el *software* ATcnea, se puede arribar a las siguientes conclusiones:

- El estudio del marco teórico, evidenció la necesidad de desarrollar un módulo para juegos serios de palabras en el *software* ATcnea, utilizando la metodología, tecnologías y herramientas seleccionadas.
- Se desarrolló un módulo para juegos serios de palabras que permitió aumentar los recursos didácticos disponibles en el *software* ATcnea.
- Se realizaron pruebas de *software* al módulo para juegos serios de palabras que permitieron identificar y corregir no conformidades, mejorando así la calidad del producto desarrollado.

.

Recomendaciones

Se recomienda realizar un estudio e implementación de otros tipos de juegos serios que se puedan incluir en el *software* ATcnea.

Referencias bibliográficas

1. Rojas , Freddy Velásquez. *Enfoques sobre el aprendizaje humano.* Junio de 2001.

2. G.*, Tevni Grajales. *El problema fundamental del aula tradicional.* Montemorelos, NL: Editorial Montemorelos. : s.n., 1997.

3. *Innovación docente y uso de las TIC en la enseñanza universitaria.* Salinas, Jesús. s.l. : Revista Universidad y Sociedad del Conocimiento, Noviembre de 2004, Vol. 1. ISSN 1698-580X.

4. Alonso, Jesus Tapia. *¿Qué es lo mejor para motivar a mis alumnos? Análisis de lo que los profesores saben, creen y hacen al respecto. .* Madrid : s.n., 1992.

5. Segovia Olmo, Felipe. *El aula inteligente. Nuevas perspectivas.* Madrid : s.n., 2003.

6. *El aula inteligente: ¿hacia un nuevo paradigma educativo?* Díaz, Antonia Lozano. 2, España : s.n., 2004, Vol. 6.

7. *The impact of classroom technology on student behav.* Angeline M. Lavin, Leon Korte, Thomas L. Davies. Dakota : s.n., 2010.

8. Piaget, Jean. *Piaget y el valor del juego en su teoría Estructuralista.*

9. Lares, Miguel J. *Juego e infancia.* Buenos Aires : Grupo Editorial Lumen, 2014.

10. *EL JUEGO EN LA ENSEÑANZA DE ELE.* María José Labrador Piquer, Pascuala Morote Magán. Valencia : s.n., 2008. ISSN 1576-7809.

11. C., Abt Clark. *Serious Game.* s.l. : Viking Press, 1970.

12. *From Visual Simulation to Virtual Reality to Games, en Computer.* Zyda, Mike. 2005.

13. Sánchez, M. *Buenas prácticas en la creación de Serious Games (Objetos de Aprendizaje Reutilizables).* Universidad de Málaga. Facultad de Ciencias de la Comunicación Campo de Teatinos. Málaga, España : s.n., 2010.

14. FELICIA, PATRICK. *Videojuegos en el aula. Manual para docentes.* 2009.

15. *Dopamina sintésis, liberación y receptores en el Sistema Nervioso Central.* Ricardo Bahena- Trujillol, Gonzalo Flores, José A. Arias-Montano. 2000, Vol. Vol.11.

16. Centro de Tecnologías para la Formación, Fortes. *Estudio de Homólogos de Sistemas de Gestión del Aula.* La Habana, Cuba. : s.n., 2016.

17. Gutiérrez González, MsC Angel. *Fundamentos de la computación .* INSTITUTO POLITÉCNICO NACIONAL DE MÉXICO : s.n., DICIEMBRE 2009.Cited: Enero 15, 2016.

18. Domínguez-Dorado, M. *Todo Programación.* Madrid : Editorial Iberprensa , 2005.

19. Matos, Rosa María. *Diseño de base de datos.* 1999.

20. HSQLDB - 100% Java Database. *HSQLDB - 100% Java Database.* [En línea] [Citado el: 9 de Diciembre de 2016.] http://hsqldb.org/.

21. González, Héctor Suárez. *Manual Hibernate.*

22. JetBrains. Intellij IDEA. *Intellij IDEA sitio oficial.* [En línea] JetBrains. [Citado el: 5 de Diciembre de 2016.] https://www.jetbrains.com.

23. Studio, Android. [En línea] https://developer.android.com/studio/index.html?hl=es-419.

24. *JavaFX.* Jasper Potts, Nancy Hildebrandt, Joni Gordon, Cindy Castillo. August 2014. E50607-02.

25. Pilato, C. Michael, Fitzpatrick, Brian W. and Collins-Sussman, Ben. *Version Control with Subversion. s.l.:O'Reilly.* 2004.

26. Git. [En línea] 2016. https://git-scm.com/.

27. Booch, G., Rumbaugh, J. and Jacobson, I. *El lenguaje unificado de modelado.* Madrid : Addison-Wesley, 1999.

28. Krall, César. *Qué es y para que sirve UML, el lenguaje Unificado de Modelado. [pdf].* 2012.

29. Avison, David and Fitzgerald, Guy. *Information systems development: methodologies, techniques and tools (3rd edition).* McGraw Hill, 2003.

30. Brito, Kerenny Acuña. *Metodología de Desarrollo.* 2005 Cited Febrero 3, 2016.

31. *Metodología de desarrollo para la Actividad productiva de la UCI.* Universidad de las Ciencias Informáticas La Habana. : s.n.

32. Sommerville, Ian. *Ingeniería del software Séptima edición .* Madrid : s.n., 2005.

33. Pressman, Roger S. *Software Engineering and practitioners approach. Septima.* New York: Hair Education : s.n., 2010.

34. Blaha, Michael. *Patterns of Data Modeling .* CRC Press, 2010.

35. Larman., Craig. *UML y Patrones. 2ª Edición.* 2003.

36. Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides,. *"Design Patterns: Elements of Reusable Object-Oriented Software",.* s.l. : Addison-Wesley, 0-201-63361-2, 1995.

37. Mc Caleb, MichaelR s.l: National Institute of Standars and Technology,. *A Conceptual Data Model of Datum Systems.* 1999.

38. Microsoft. *Diagramas de componentes de UML: Referencia.* [Cited: abril 26, 2016].

39. Black, R.,. *Managing the Testing Process,.* Segunda Edición, 2001.

40. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El lenguaje Unificado de Modelado.* s.l. : Manual de Referencia .. s.l. Adisson Wesley.

41. S.PRESSMAN., ROGER. *Ingeniería del Software:Un enfoque práctico.* 5ta Edición. 2002.

42. Maven. JUnit. *JUnit sitio oficial.* [En línea] Maven, 1 de Enero de 2017. [Citado el: 16 de Diciembre de 2016.] http://junit.org.

43. Geeks, Java Code. *Mockito Programing CookBook.* New York : s.n., 2014.

44. Pressman, Roger S. *Ingeniería de Software. Un enfoque práctico.* México : María Tereza Zapata, 2010. 978-607-15-0315-5.

45. J.L., Ramírez. *"Gamificación, Mecánicas de juegos en tu vida personal y profesional".* Madrid : s.n., 2014.

46. MARÍN, I. e HIERRO, E. *"Gamificación. El poder del juego en la gestión empresarial y la conexión con los clientes".* 2013.

47. Olmo, Felipe Segovia. *El aula inteligente.* 2003.

48. MARGULIS, LUCIO. *Juego Serio una metáfora de la dinámica de las organizaciones modernas. .* 2010.

49. CAMY, PAULA. *Juegos Serios/Serious Games.* 2006.

50. Belayev, Eugene. *IntelliJ IDEA in Action .* 2005.

51. Elgin, Ben. *Google Buys Android for Its Mobile Arsenal.* 2005.

52. Driskell, J. E. *Games, motivation and learning, Simulation & Gaming.* diciembre de 2002.

53. Letier, Patricio, Canós, José H. and Penadés, Ma Carmen. *Metodologías ágiles para el desarrollo de software.* España: Universidad Politécnica de Valencia. : s.n., 2003.

54. Bahit, Eugenia. *El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.* 2011.

55. Villa Betancur, Alejandro and Gira Plaza, Jorge E. *Automatización de pruebas unitarias de códigos PHP. s.l Scientia et Technica,.* 2012. pp. 147–151. Vol.2.

56. Netbeans. Netbeans. *Netbeans.* [En línea] 2016. https://netbeans.org/index_es.html.

57. Massachusetts Institute of Technology. *Massachusetts Institute of Technology sitio oficial.* [En línea] http://www.mit.edu/.

58. Mythware. *Mythware sitio oficial.* [En línea] [Citado el: 10 de Diciembre de 2016.] http://www.mythware.com.

59. ITALC. *ITALC sitio oficial.* [En línea] [Citado el: 10 de Diciembre de 2016.] http://italc.sourceforge.net/.

60. *NetSupport.* [En línea] NetSupport sitio oficial. [Citado el: 10 de diciembre de 2016.] www. NetSupport -inc.com .

61. *https://www.imperosoftware.co.uk/wp-content/uploads/2015/02/Impero-Education-Pro-Product-Feature-Datasheet.pdf., Impero Solutions Ltd. School Network & Device Management Software. Impero. [Online] 1. [Consulta: 1 febrero 2016.].*

62. http://www.capterra.com/classroom-management-software/spotlight/133791/AB%20Tutor/Globe%20Microsystems., Reviews of AB Tutor :

Free Pricing & Demos : Classroom Management Software. [en línea] [sin fecha]. [Consulta: 26 enero 2016]. Disponible en:.