Universidad de las Ciencias Informáticas

Facultad 4

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

Módulo de gamificación basado en ranking para la plataforma educativa ZERA

Autor:

Bernardo Sarría Fuentes

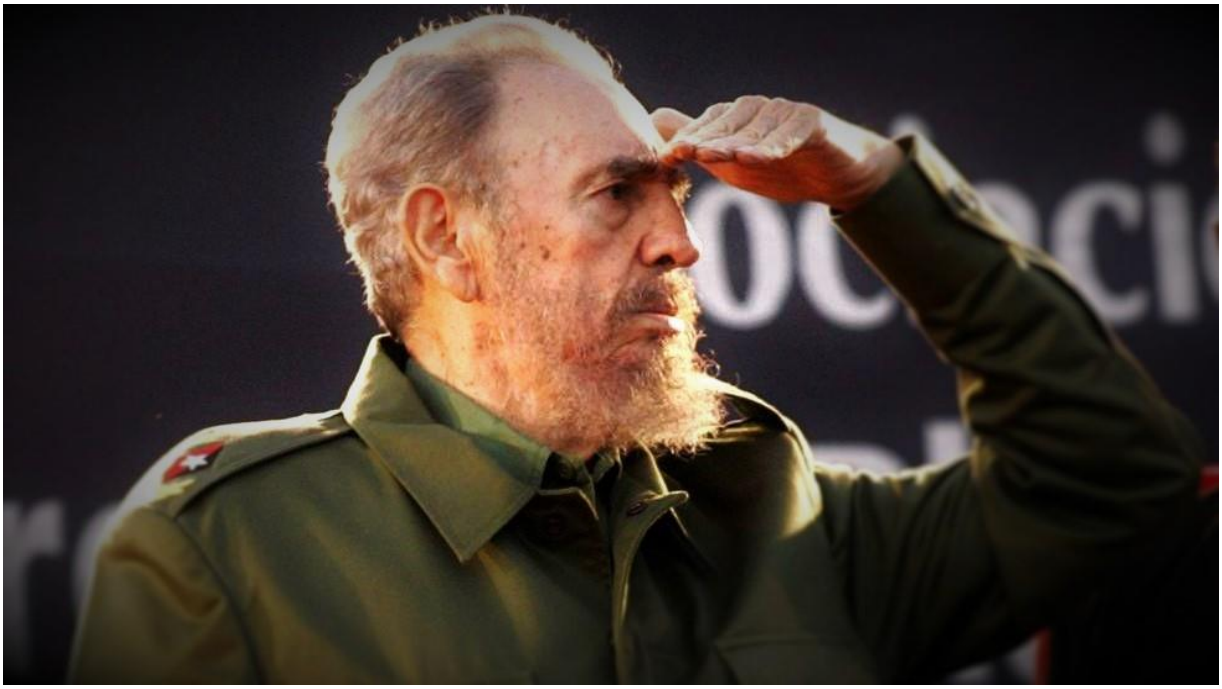
Tutores:

MSc. Luis Eduardo Benítez Oliva

MSc. Dania Domínguez Alvarez

La Habana, 2019

“Año 60 de la Revolución”



"No puede concebirse una revolución sin educación, no puede concebirse progreso sin educación, no puede concebirse un futuro esplendoroso para la nación cubana sin educación, no puede concebirse un mejoramiento en todos los órdenes de la vida sin educación"

Fidel Castro Ruz

Declaración de Autoría

Declaro ser el único autor del presente Trabajo de Diploma y reconozco al Centro de Tecnologías para la Formación (FORTES) de la Facultad 4 de la Universidad de las Ciencias Informáticas, los derechos patrimoniales del mismo, con carácter exclusivo, para que hagan el uso que estimen pertinente con el mismo.

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

_____

Firma del Autor
Bernardo Sarría Fuentes

_____

Firma del Tutor
MSc. Luis Eduardo Benítez Oliva

_____

Firma del Tutor
MSc. Dania Domínguez Alvarez

Dedicatoria

Le dedico este trabajo a mis padres, especialmente a mí mamá, por ser la madre que es con sus virtudes y defectos, ella sabe que la quiero muchísimo, a mi novia, a mis suegros, a mis hermanos, a mis tíos, a mis amigos, a mis profesores y todas aquellas personas que hicieron posible este sueño.

Agradecimientos

Agradecerle primeramente a mi familia, mis tíos, mis hermanos, mis suegros, a mis padres. por haber sido una fuente de inspiración y lucha durante estos 5 años. Gracias por haber estado conmigo en todo momento, por el apoyo incondicional que siempre me han brindado, por los jalones de orejas y sermones que fueron necesarios en su momento, por brindarme su ayuda en cualquier circunstancia y por ser los padres que son. Por ustedes soy quien soy el día de hoy. Este logro va para ustedes.

Gracias a mis tutores Dania Domínguez Alvarez y Luis Eduardo Benítez Oliva, por la dedicación y ayuda que me otorgaron, esto no hubiese sido posible sin ustedes. Mencionar a la profesora Zobeida que al igual que mis tutores estuvieron día a día conmigo en todo momento. Muchas gracias.

A mi novia María de la Caridad por ser mi compañera durante estos largos años, quien me diera su apoyo en todo momento. Gracias por estar ahí para mí, siempre serás muy importante en mi vida.

Agradecerle a la profesora Saylin y al profesor Luis Manuel. Profesores que desde que los conocí, jugaron un papel importante en mi vida. ¿Qué hubiese sido de mi sin ustedes? No tengo palabras para agradecerles todo lo que han hecho. Ustedes también son parte de mi familia. Gracias

A mis compañeros de aula, los que estuvieron conmigo desde mis inicios en la carrera, y los que encontré en el transcurso de la misma. Gracias por los momentos que pasamos juntos, y estoy muy agradecido de haberlos conocido a todos y cada uno de ustedes, espero a que no perdamos el contacto y podamos seguir juntos por mucho tiempo, recordando los momentos de intenso estudio y de risas interminables, además de un apoyo mutuo y desinteresado.

A Adrián Céspedes, Israel Machín, Andy, Amado, Lorenzo, que más que amigos se convirtieron en hermanos, que pasamos por tantas cosas juntos y que nos ayudábamos mutuamente. Saben que los llevaré siempre conmigo. Gracias

A Carlos y Reydel; bueno que decir de ellos. Ellos saben que los quiero muchísimo y que no se van a librar de mi tan fácil, apartando que fueron los primeros amigos que conocí en mis inicios de estudio en la universidad. Gracias a los dos.

De manera general, gracias a todas y cada una de las personas que hicieron posible que este momento fuese una realidad. Por el apoyo, ánimos y la amistad que me brindaron. Gracias.

Resumen

Los avances de las Tecnologías de la Información y las Comunicaciones en la actualidad brindan la oportunidad de enseñanza a partir de la web. Como consecuencia muchos profesores han buscado soluciones que faciliten y agudicen el interés de los estudiantes por la utilización de dichos medios. La aparición de los videojuegos ha tenido un fuerte impacto en la juventud y es cada vez más notable en la actualidad, trayendo consigo que los jóvenes hayan generado una actitud adictiva a estos. Además, la aplicación de los videojuegos en otros contextos como la educación y los negocios, podrían generar grandes beneficios. A partir de esto surge la idea de la gamificación. En la Universidad de las Ciencias Informáticas se desarrolló la plataforma educativa ZERA, sistema que dado sus características puede ser utilizado tanto a nivel nacional como internacional. Pero desde sus inicios no se concibió bajo la integración de la gamificación, la cual ha tenido un gran auge en los últimos años y ha presentado buenos resultados en el ámbito de la enseñanza, razón por la que se desea incluir en la versión 2.1. La presente investigación se centra en la necesidad de desarrollar un módulo que integre mecánicas de gamificación a la plataforma educativa ZERA. Dando como resultado del estudio realizado, un módulo que utiliza mecánicas de gamificación haciendo referencia al ranking.

Palabras claves: gamificación, módulo, plataforma educativa, ranking, videojuego.

Índice:

Índice de Ilustraciones

Índice de Tablas

Índice de Ecuaciones

Introducción

El acelerado desarrollo de las tecnologías de la información y las comunicaciones (TIC) ha provocado cambios en diversas áreas y procesos, sin quedar exento el proceso de formación (1). En este ámbito se puede apreciar potenciales cambios en el proceso de enseñanza-aprendizaje (PEA), fundamentalmente en la manera en que el estudiante recibe y materializa los conocimientos y en la forma en que el profesor imparte o elabora las clases o cursos para influir positivamente en las actividades y procesos educativos encaminando el aprendizaje de los estudiantes estableciendo nuevos métodos de comunicación entre maestro, estudiante, sociedad y conocimiento. Vale resaltar que hoy día, cada vez se multiplican más los programas académicos relacionados con las modalidades: presencial con apoyo de TIC, b-learning o aprendizaje mixto con apoyo de las TIC, o el aprendizaje e-learning, aprendizaje en línea.

En estos escenarios del proceso educativo, la evaluación, como en cualquier área del campo educativo, ha de estar siempre acorde a estos aspectos, evidenciándose en los Sistemas de Gestión de Aprendizaje, conocidos por sus siglas en inglés como LMS (Learning Management System) (2). Esta evaluación en el proceso educativo puede conceptualizarse como un proceso dinámico, continuo y sistemático, enfocado hacia los cambios de las conductas y rendimientos. Esta a su vez permite ser utilizada para generar tablas de posiciones (ranking) a partir de las evaluaciones en las plataformas LMS, se establece un conjunto de elementos tales que, para uno o varios criterios, el primero de ellos presenta un valor superior al segundo, este a su vez mayor que el tercero y así sucesivamente (2).

Las plataformas LMS son entornos virtuales de aprendizaje orientados a facilitar la capacitación a distancia, tanto para instituciones educativas como empresas. Este sistema permite la creación de aulas virtuales, así como, la interacción entre tutores y alumnos. También, se pueden hacer evaluaciones, intercambiar archivos, participar en foros y chats entre otras herramientas complementarias (3).

Dentro de un LMS la interacción se establece entre un contenido virtual (el material de autoaprendizaje) y el estudiante o entre dos usuarios, sean ambos estudiantes o maestro-estudiante a través de la interfaz de usuario que provee el sistema. Dichas interacciones en el PEA se evidencian utilizando las tecnologías (4).

En distintos entornos virtuales de aprendizaje suelen ponerse en práctica el uso de los diferentes elementos de juegos (dinámicas y mecánicas), donde estos se unifican utilizando el término de gamificación. Donde, las dinámicas se basan en: emoción, competencia y el aprendizaje al que puede llegar el estudiante según su esfuerzo y motivación. Por otra parte, las mecánicas tienen

como objetivo potenciar el interés a partir de ciertas actividades, tales como: acumular puntos y definirlos en un ranking, escalar niveles, premios y reconocimientos. Todos estos elementos, casi siempre de una forma u otra, se encuentran integrados dentro de los LMS (2). La incorporación de estos elementos en sistemas de aprendizaje se ha extendido y generalizado, lo que ha posibilitado que los usuarios sean atraídos por estas nuevas maneras de aprender, colaborar y competir, trayendo como consecuencias el aumento de la interactividad en los sistemas de gestión de aprendizaje y la retroalimentación en actividades educativas (2).

La gamificación implica tener en cuenta un sistema de posiciones. De este modo, incluso si los objetivos a alcanzar son difíciles o requieren mucho esfuerzo siempre se encontrará distintas posiciones (primera, intermedia y última), fomentando la competitividad e interés entre los usuarios. La gamificación entre sus principales elementos se encuentra la tabla de posiciones conocida como *ranking*, donde cada usuario es posicionado acorde a su desempeño en comparación con el resto de los participantes. La aplicación de esta mecánica de juego en una plataforma LMS fomenta a que se puedan aplicar las demás mecánicas de juegos (5).

Cuba, quien demuestra ser una potencia en materia de educación, incursiona en estos sistemas, por lo que fortalece su papel en el PEA e impulsa la utilización de las TIC en todos sus niveles de enseñanza. Un ejemplo de ello, es la creación de la Universidad de las Ciencias Informáticas (UCI), la cual tiene como objetivo principal formar profesionales comprometidos con la Patria y altamente calificados en la rama de la Informática, además de producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y a su vez, servir de soporte a la industria cubana en la Informática (7).En esta institución existen varias plataformas de software libre a código abierto, entre las que se encuentran Xauce, Xabal, Xavia, Xilema, Xedro (6).

En la UCI se encuentra el Centro de Tecnologías para la Formación (FORTES), que tiene entre sus objetivos el desarrollo de aplicaciones en función del PEA. Este centro cuenta con varios proyectos enmarcados en la línea XAUCE dedicada a la educación entre los cuales se encuentra la plataforma educativa de aprendizaje ZERA (6).

La plataforma educativa ZERA al igual que el resto de las plataformas LMS posibilita, entre otros aspectos, el diseño de cursos y llevar un seguimiento detallado de las acciones del educando y sus avances. Esta plataforma sirve de apoyo al profesor, el cual tendrá a su disposición opciones para gestionar el aprendizaje de sus estudiantes, aumentando su preparación como profesionales (7).

Actualmente la plataforma presenta poca utilización de los estudiantes hacia ella, en la que los estudiantes dejan de ver contenido que en ella se exponen.

Se aplicó una encuesta a 40 estudiantes entre 3er, 4to y 5to año de la facultad 4 pertenecientes a la UCI y se detectaron algunos problemas en la utilización de la plataforma. A continuación, se mencionan los problemas detectados:

- Un 90% percibe que la plataforma solo se utiliza para consultar contenido académico, no como un entorno interactivo de aprendizaje, por lo que los estudiantes no se sienten atraídos a usarla.
- A un 75% le gustaría que se incorporaran mecánicas de juego en la plataforma.

A partir de los resultados obtenidos en la encuesta, el problema tecnológico derivado de la plataforma educativa Zera es que no cuenta con un módulo de gamificación basado en *ranking*, por lo que no se aplica mecánicas de gamificación.

Teniendo en cuenta lo antes planteado, se define como **problema a resolver** en la investigación: ¿Cómo gestionar el ranking en la plataforma educativa Zera?

Se delimita como **objeto de estudio**: Las plataformas LMS.

Se define como **objetivo general**: Desarrollar un módulo de gamificación basado en ranking para la plataforma educativa ZERA.

El **campo de acción** de la investigación está enfocado en: Gamificación basada en ranking.

Para darle cumplimiento al objetivo general, se definen las siguientes **tareas a cumplir:**

- Análisis de los principales aspectos teóricos relacionados con la gamificación basada en ranking.
- Caracterización de los sistemas similares que utilicen esta forma de gamificar, para un mejor entendimiento del módulo a desarrollar.
- Selección de la metodología y tecnologías de desarrollo con el objetivo de realizar el módulo de gamificación basado en ranking para la plataforma educativa ZERA.
- Definición de los requisitos funcionales y no funcionales del módulo de gamificación propuesto por la investigación.
- Implementación de los requisitos funcionales.
- Validación del módulo mediante la estrategia de pruebas definidas.

Posibles resultados: Una vez finalizado el presente trabajo, se obtendrá como resultado un módulo de gamificación basado en ranking para la plataforma educativa ZERA.
Los métodos de investigación empleados son:

Métodos teóricos:

- **Análisis histórico-lógico**: Se utilizó con el objetivo de realizar un estudio de los referentes teóricos, metodológicos y tecnológicos que sustentan el desarrollo de plataformas virtuales en el PEA y las tendencias actuales en que el mismo se expresa.
- **Analítico y sintético**: Permitió profundizar en el conocimiento y estudio de los antecedentes y aportó significativos elementos para llegar a la propuesta de solución. Se manifiesta en el estudio de diferentes fuentes, que integran en sus reflexiones y prácticas el tema de las TIC en la educación.
- **Modelación:** Se utilizó con el objetivo de crear modelos y diagramas que son abstracciones del producto final, lo que permitió tener un dominio inicial de la información que se va a modelar, representar de forma estática los requisitos y obtener una versión del sistema original para validar los requisitos con el cliente.

Métodos Empíricos

- **Encuestas:** Permite detectar los problemas existentes en la plataforma educativa ZERA.

El presente trabajo de diploma está compuesto por tres capítulos:

Capítulo 1: Fundamentación teórica. En este capítulo se hace referencia a los elementos teóricos en los cual está basado la investigación y donde se incluye un estudio del estado del arte de este tema. Se exponen los lenguajes, metodologías, herramientas y tecnologías utilizadas en el desarrollo de la solución, así como los principales conceptos relacionados con el contenido.

Capítulo 2: Propuesta de solución. En este capítulo se describe la solución propuesta para llevar a cabo el desarrollo del módulo. Se detallan las principales características, así como la especificación de los requisitos funcionales a implementar y los no funcionales. Además, se realiza una descripción de la arquitectura y los patrones de diseño presentes en la propuesta.

Capítulo 3: Implementación y Prueba. En este capítulo se describen las fases de implementación y pruebas. Se realiza la implementación de todas las funcionalidades identificadas, logrando un módulo que satisfaga las principales necesidades del cliente. Se detallan, además, las pruebas realizadas al sistema durante el proceso de implementación, para asegurar que el módulo cumple con las especificaciones requeridas por el cliente.

Capítulo I: Fundamentación Teórica

En el presente capítulo se expone el marco teórico en el que se desarrolla la investigación, creando una base sólida para el desarrollo del proyecto. Para ello, se analizan conceptos sobre la investigación que se lleva a cabo y los distintos sistemas que utilizan la gamificación basada en sistemas de puntuación, así como las características que presentan y el aporte que brinda su utilización. Además, se especifica la metodología, lenguajes y herramientas a utilizar en el desarrollo de la solución.

1.1. Gamificación

Varios autores asocian los conceptos de gamificación y ludificación, e incluso lo referencian como sinónimo uno de otro. Así también hay quienes utilizan el término ludificación como la traducción más cercana de gamificación. En la aplicación en entornos educativos estos presentan ciertas diferencias que invitan a plantear el presente análisis (8).

La diferencia entre la ludificación y gamificación se presenta en la utilización del término, puesto que no fue sino hasta el 2010 que la gamificación empezó a ganar popularidad como lo señala Rodríguez y Santiago (2015). En algunos textos se mantiene ludificación y en otros ya se habla de gamificación, considerando esta última como el término más empleado en la actualidad y con diferencia. Solo unos pocos utilizan el término ludificación y además es un término que se ha empezado a utilizar mucho después, sobre todo por personas inicialmente ajenas a este movimiento, y más cercanas al periodismo o lingüistas. La gamificación propone un sistema concreto que toma características y acciones de la concepción del juego para llevarlas al entorno educativo, apostándole a potenciar la motivación, concentración, esfuerzo, fidelización y otros valores positivos, hasta llegar al resultado final: el aprendizaje (5).

La definición de gamificación dada por (9) es la que mejor se adecúa, definida como:

… es el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos. Se trata de una nueva y poderosa estrategia para influir y motivar a grupos de personas**.**

Esto define claramente lo que se busca en las plataformas LMS, intentar motivar al alumnado en entornos monótonos y exigentes para que se esfuercen a la hora de realizar las tareas encomendadas por el docente, y a su vez, una forma de interacción en la plataforma.

Gamificación y juego no son lo mismo. El juego es una actividad libre y voluntaria, que busca el entretenimiento del usuario y donde se puede ganar o perder. Por su parte la gamificación selecciona aquellas mecánicas y dinámicas del juego que se pueden aprovechar para desarrollar determinadas

destrezas, entrenar en determinadas tareas, reforzar o cambiar comportamientos. Además, la gamificación es un proceso activo donde nunca se pierde, porque realmente está orientado a la práctica y mejora de un contenido o competencia. En definitiva, desarrolla experiencias con elementos de juego para progresar hacia un objetivo. La ilustración 1 resume las diferencias entre cada uno de esos conceptos de forma gráfica (10):

Ilustración 1. Diferencias y similitudes de la gamificación y el juego (Fuente: Díaz Fernández, 2018)

1.2. Elementos fundamentales de la gamificación

La gamificación se estructura sobre las mecánicas y las dinámicas de juego, dos conceptos íntimamente relacionados. Se definen estos dos conceptos ya que algunas de las principales mecánicas y dinámicas del juego serán empleadas en el proyecto (11).

1.2.1 Dinámicas

Las dinámicas de juego son aquellas necesidades e inquietudes humanas que motivan a las personas. Las dinámicas son los conceptos, la estructura implícita del juego, cómo el jugador se siente al participar en el proceso de aprendizaje. Las dinámicas de juego son tan diversas como necesidades tiene el ser humano (11):

- **Aprendizaje:** A medida que el jugador usa el sistema gamificado va adquiriendo nuevos conocimientos de estrategias que permiten superar retos complejos.
- **Retos:** Éstos deben ser claros y el usuario deberá saber cuándo ha superado los mismos.
- **Socialización:** Las alianzas entre jugadores permiten que el usuario viva experiencias distintas.

- **Emociones:** Cada una de las acciones que se están llevando a cabo provocan unas emociones determinadas.
- **Competición:** Los sistemas gamificados pueden favorecer la cooperación entre los usuarios. En un entorno competitivo se crea mayor interés y motivación en el jugador, bien sea en su desempeño individual o en un grupo.
- **Identidad:** En este aspecto se trataría en como el usuario se adentra en la historia y la vive como si estuviera dentro del juego, de esta manera el jugador vive una experiencia que puede ser compartida y repetida.
- **Narrativa:** La historia presente en el sistema gamificado es la puerta de entrada del usuario a un mundo por conocer.

1.2.2 Mecánicas

Las mecánicas del juego son las distintas acciones, comportamientos, técnicas y mecanismos de control que se utilizan para convertir en juego una actividad. Los procesos mecánicos son aquellos que llevan al desarrollo del sistema y pueden ser de diferentes tipos según el comportamiento, (basado en el comportamiento y psiquis humana), la retroalimentación (relativa al ciclo de interacción en el flujo de las actividades) o la progresión (acumulación de habilidades a través de la experiencia) (11). Estas mecánicas se utilizan para lograr que la consecución de objetivos sea precisa y completa. Su uso logra conseguir una alta motivación en el usuario.

Algunas de las principales mecánicas de juego son:

- **Puntos:** Se otorgan cuando se realiza algún tipo de acción. Crean una sensación de progreso para el usuario y le devuelven inmediatamente información sobre las acciones que realiza, correctas o incorrectas.
- **Niveles:** Los niveles son unos indicadores que aportan reconocimiento y respeto una vez se han cumplido unos hitos determinados. Los niveles son unas de las motivaciones más fuertes para los jugadores
- **Desafíos:** Los desafíos permiten que los usuarios compitan y se reten entre sí para obtener la puntuación más alta en alguna actividad. Normalmente, el jugador con la puntuación más alta es recompensado.
- **Misiones o retos:** Normalmente se construyen bajo un sistema basado en puntos y están enfocados en la motivación de los usuarios de finalizar tareas más complicadas.

- **Ranking:** Esta mecánica proporciona deseo de aspiración, fama y que el nombre del usuario aparezca resaltado por encima de otros. También es un indicador que permite conocer como lo está haciendo el usuario en comparación con los demás jugadores.
- **Premios:** Los retos y misiones que plantea un juego intentan hacer sentir al usuario que el juego tiene una finalidad, una meta. Esa finalidad viene representada por los premios, que son la recompensa tangible (bien física o virtualmente) a la consecución de un objetivo mediante una acción o serie de acciones. Los premios pueden clasificarse en trofeos, medallas o logros que suelen ser visibles para otros usuarios con el fin de obtener reconocimiento y alimentar la motivación del resto de jugadores.

1.2.3 Componentes estéticos

Los componentes estéticos son las implementaciones específicas de las dinámicas y mecánicas: avatares, insignias o medallas, colecciones, niveles, equipos, bienes virtuales, etc. Los componentes estéticos más populares son los puntos e insignias. Los componentes estéticos son los elementos que representan la implementación específica que responden a una mecánica, y esta a su vez es la base que propicia las dinámicas de juego en entornos de aprendizaje (11).

1.3. Ranking en la gamificación

El propósito de una tabla de posicionamiento es mostrar a los jugadores dónde se clasifican en un sistema gamificado. Los de arriba disfrutan del liderazgo que aporta; en cuanto a todos los demás, la tabla de posicionamiento les muestra dónde están parados en relación con sus compañeros.

A menudo, la sola presencia de una tabla de posicionamiento puede provocar el deseo de jugar. El simple objetivo de elevar el ranking sirve como un poderoso motivador para continuar. La gente le gusta mantener la puntuación. Entender esto y proporcionar maneras fáciles de hacerlo es una excelente manera de fomentar el compromiso en el juego. Para algunos, la mera vista de su rango en la tabla de posicionamiento es toda la recompensa que buscan.

Cuando se trata de mecánicas de gamificación, nada mejor que la tabla de posicionamiento. El truco para las tablas de posicionamientos es diseñarlos de tal manera que alienten a los jugadores a permanecer en el juego. La tabla de posicionamiento o ranking a desarrollar en la plataforma ZERA es la evidencia o reflejo de un sistema de puntos que evalúa el desempeño de los estudiantes al participar en actividades evaluativas de cada curso con el fin de motivarlos (12).

1.4. Plataforma Educativa XAUCE ZERA

La Plataforma Educativa XAUCE ZERA es una plataforma educativa que se desarrolla en el Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas. Es una plataforma innovadora, flexible, fácil de usar, interactiva y adaptable capaz de guiar el PEA en empresas, organizaciones, comunidades e instituciones de cualquier nivel de enseñanza a partir de un conjunto poderoso de herramientas centradas en los aprendices y ambientes de aprendizaje colaborativo, que le dan poder, tanto a la enseñanza como al aprendizaje (7).

Es accesible desde cualquier dispositivo móvil y soporta contenidos interactivos, así como recursos multimedia a partir de plantillas previamente definidas que se entremezclan con documentos, presentaciones y otros materiales junto a foros de discusión, cuestionarios y disímiles tipologías de ejercicios que pueden ser evaluados de forma automática, por coevaluación o directamente por el profesor (7).

La plataforma cuenta con una interfaz moderna y fácil de usar, diseñada para ser responsiva y accesible. De igual forma cuenta con actividades y herramientas colaborativas que pueden ser evaluadas de diferentes formas; realiza una gestión conveniente de los archivos y notificaciones en tiempo real, además de contar con un editor de texto simple e intuitivo. Permite una autenticación segura por la correcta gestión de permisos y roles de usuario y que los mismos vean el contenido del curso y aprendan en su propio idioma debido a su capacidad multilingüe (7).

1.4.1. Análisis de Plataformas LMS

Actualmente existen plataformas que utilizan gamificación en cuanto a sistemas de puntuación. A continuación, se realizará un análisis de dichas plataformas a nivel internacional en cuanto al sistema de posiciones:

Moodle

Moodle fue desarrollada en PHP, su distribución es libre a través de la licencia GPL (General Public Licence) y fue lanzada en el 2001, su última versión estable es la 3.4, actualizada el 15 de enero del 2018 y está disponible en español, a nivel mundial es la plataforma LMS más popular (13). La gamificación en este sistema ha sido conseguida a través de *plugins* (aplicación que se relaciona con otra para agregarle una función nueva y generalmente muy específica) desarrollados por la comunidad. Uno de estos *plugins* es denominado *Ranking Block*, provee un tablero de posiciones a partir de un seguimiento de las actividades que completa un estudiante en una página HTML (14). La cantidad de puntos por defecto que acumula un estudiante es según el tipo de tarea que sea completada más el

grado con que el profesor le evalúa. Los puntos por defecto se pueden configurar como parte del *plugin* según el tipo de actividad (15).

El ranking estructura su información en tres columnas: la posición, el nombre y los puntos acumulados. Para la plataforma educativa ZERA estos son también los datos indispensables a mostrar. En un inicio solo se aprecia parcialmente el ranking, mostrándose los primeros cinco estudiantes, para ver al resto, se debe presionar un botón que muestra el tablero completo. Esta previsualización tiene como objetivo mostrar solo los líderes, como reconocimiento a su desempeño. Debido a la incompatibilidad en cuanto a las tecnologías empleadas para el desarrollo de ZERA y Moodle, no se puede integrar *Ranking Block* directamente. El sistema de evaluación empleado en Moodle está definido como la nota que recibe un estudiante por un profesor en línea o según la haya definido en una actividad que el sistema pueda evaluar.

Schoology

Schoology es un LMS para colegios que engloban primaria y secundaria, instituciones de educación más alta, y empresas que permite a sus usuarios crear, dirigir y compartir contenidos y recursos, es propietaria dirigida por la empresa de igual nombre y fue fundada el 1 de mayo de 2009. Un rasgo notable en el sistema de posiciones de Shoology es que en la configuración de calificaciones permite asociar a cada actividad un peso relativo en cuanto a su importancia para el curso (16). Por ejemplo, si en un curso están definidas cuatro actividades y todas tienen un 100% de peso entonces cada una influye en un 25% a la nota final del curso. Este criterio de importancia permite en un contexto determinado evaluar justamente el desempeño de cada estudiante. En la plataforma ZERA cada ejercicio tiene asociado un nivel de dificultad, por tanto, al completarse satisfactoriamente un ejercicio debe tenerse en cuenta, al igual que en *Schoology* qué peso o complejidad este aporta para evaluar el desempeño total.

Canvas

Canvas es una LMS desarrollada por *Instructure Inc.,* fue lanzada en febrero del 2011 bajo la licencia GPL v3 como código abierto. Esta plataforma es usada por más de trescientos colegios universitarios alrededor de 12 distritos, con aproximadamente nueve millones de usuarios para finales del 2013. Fue implementada con el *framework* para *Ruby* (*Ruby on Rails*) y *PostgreSQL* como sistema gestor de bases de datos (14). Una de las particularidades de Canvas es el empleo de un sistema de posiciones que tiene en cuenta las expectativas cumplidas al realizar una actividad. Cuando un ejercicio es terminado por el estudiante este puede haber excedido, cumplido o no satisfecho las expectativas, para cada uno de estos criterios se asocia un índice (peso) que incide en el promedio de su calificación. La

fórmula empleada para evaluar a un estudiante es conocida como "*Decaying Average".* Este método calcula el promedio asociando un peso mayor a la última evaluación que al resto (17). En la plataforma ZERA los estudiantes se enfrentan a actividades que brindan más de un intento, evaluar el desempeño de estos ejercicios priorizando la evaluación del intento final es posible si se emplea esta fórmula.

A nivel nacional, también se estudió una plataforma que usa elementos de gamificación:

COJ (Caribbena Online Judge)

COJ es un juez en línea para programación competitiva, este tipo de plataformas también han sido influenciados por la gamificación. Este juez en línea fue desarrollado en la UCI, y es la plataforma principal de competencia y entrenamiento del movimiento de programación Tomás López Jiménez.

Un rasgo que distingue a estos sistemas es que la puntuación de un usuario está sujeto a muchos factores en cuanto a los ejercicios de programación que son automáticamente evaluados por la aplicación. Debido a que en una plataforma educativa el número de usuarios registrados es elevado, para facilitar la usabilidad del ranking, es necesario un filtro que permita localizar rápidamente a un usuario en dicho listado. COJ permite filtrar según el nombre, la posición y si está conectado o no. Como característica adicional permite comparar dos perfiles, para ello expone el comportamiento de ambos usuarios respecto a todos los indicadores que son tomados en cuenta en el análisis evaluativo. En la plataforma ZERA los estudiantes deben ser capaces de comparar sus resultados con los de sus compañeros, los profesores también deben ser capaces de visualizar dicho análisis dados dos de sus estudiantes. Esta funcionalidad permite a organizadores de torneos que se celebran en dicha plataforma analizar el desempeño individual y colectivamente.

Análisis del estudio de aplicaciones similares

A pesar de que se pueda acceder a las plataformas mencionadas anteriormente por su condición de gratuidad, existen desventajas que limitan el trabajo con las mismas:

- En Moodle, el manejo de las evaluaciones del sistema está definida en base a dos puntos, pues el modelo de datos está estructurado de acuerdo a la forma en que se manejan las evaluaciones, lo que restringe el trabajo con los cursos y actividades.
- Con Schoology para lograr que los estudiantes se registren en la plataforma, estos deben poseer un código de invitación que solo un instructor o administrador puede enviarle vía correo electrónico, impidiendo de esta forma que cualquier usuario que desea trabaje con las actividades de la plataforma.

- En la plataforma Canvas no siempre funcionan todas las funcionalidades en el sistema, por lo que los usuarios si quieren usar una funcionalidad más allá de la plataforma, esta puede costarle dinero.
- Una de las más grandes desventajas de COJ es que es una plataforma en inglés, por lo que se precisa al menos de un dominio técnico en dicho idioma. De igual forma, las comparaciones de los usuarios en esta te muestran solo el número asociado a la actividad.

A pesar de las limitaciones, el estudio de estas plataformas permitió entender el funcionamiento de como realizan el ranking en cada plataforma, así como las formas de evaluación que utilizan. Por ello, se optó por seleccionar algunas de las funcionalidades que las plataformas manejan para poder llevar a cabo el desarrollo del módulo más viable.

1.5. Entorno de desarrollo usada bajo la arquitectura Xalix

En este epígrafe, se hace énfasis en las principales herramientas y tecnologías utilizadas en el desarrollo del módulo, priorizándose las definidas como parte de la arquitectura de desarrollo de software de la Plataforma Educativa XAUCE ZERA.

1.5.1. Lenguajes de desarrollo

PHP

Preprocessed Hypertext Pages (PHP) es un lenguaje del lado del servidor (esto significa que PHP funciona en un servidor remoto que procesa la página web antes de que sea abierta por el navegador del usuario), cuyas características principales son la independencia de plataforma y su gratuidad y que es especialmente creado para el desarrollo de páginas web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo (18).

PHP es completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Por otra parte, el código fuente escrito en PHP es invisible al navegador y al cliente. El servidor es el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable (18).

HTML

El Lenguaje de Marcado de Hipertexto, en inglés, *Hypertext Mark-Up Language*, (HTML) es el lenguaje de marcado predominante para la construcción de páginas web. Permite representar el contenido enriquecido en forma de texto, así como complementar el texto con objetos, como el caso de las

imágenes. HTML describe la estructura del contenido, además, puede manejar la apariencia de un documento y también su comportamiento a través de un script, por ejemplo, JavaScript (18).

CSS

Las Hojas de Estilo en Cascada, del inglés, *Cascade StyleSheets* (CSS) es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML[1]. CSS es la mejor forma de separar contenido y presentación; es imprescindible para crear aplicaciones web complejas. Separar contenido y presentación, brinda numerosas ventajas, ya que obliga a crear documentos HTML/ XHTML bien definidos, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (19).

El lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, el formato de tablas, la separación, el color, tamaño y tipo de letra de titulares y/o textos, la tabulación con la que se muestran los elementos de una lista o menú (19).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos Intermedios (20).

Para el desarrollo del módulo se utilizaron los lenguajes de desarrollo UML v2.4.1, PHP v7.0, HTML v5.0, CSS v3.0 y JavaScript v2.1 por sus disímiles ventajas, además de ser los definidos y utilizados en el proyecto al cual va dirigido el presente trabajo de diploma.

1.5.2. Frameworks y biblioteca para el desarrollo

Un framework es un conjunto de clases cooperativas que construyen un diseño reutilizable para un tipo específico de software. Este proporciona la arquitectura partiendo el diseño en clases abstractas y definiendo sus responsabilidades y colaboraciones. Un desarrollador realiza una aplicación haciendo subclases y componiendo instancias a partir de las clases definidas por el framework (21).

[1] XHTML o Extensible Hypertext Markup Language es una versión XML de validación de HTML.

De igual forma, un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (22).

Symfony

Symfony es un framework de PHP basado en la arquitectura Modelo-Vista-Controlador (MVC); su primera versión surgió por el año 2005 como un proyecto de software libre que se ha convertido en uno de los frameworks más populares de PHP que existe en la actualidad. Su fundador y mayor impulsor, el francés Fabien Potencier, dedica la mayor parte de su tiempo a corregir los problemas que puedan existir y que son identificado por la gran comunidad de la cual se rodea este framework (23).
Por más que Symfony puede ser utilizado para otros tipos de desarrollos no orientados a la web, fue diseñado para optimizar el desarrollo de aplicaciones web, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto (23).

Bootstrap

Bootstrap es un framework basado en HTML y CSS, creado por Twitter y liberado en 2012. Desde entonces ha ganado muchos adeptos, hasta el nivel de contar con la mayor comunidad de Github del mundo. Este framework ayuda a agilizar la creación de la interfaz de nuestra página web. Su uso permite que el sitio web sea adaptable a la pantalla del dispositivo con el que se accede, ya sea un ordenador, tablet, smartphone, televisión. Esto significa que tendremos una web sensible o adaptativa (24).

JQuery

JQuery es una librería o framework de JavaScript creado inicialmente por John Resig que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología Ajax a páginas web. Al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Con las funciones propias de esta librería se logran resultados en menos tiempo y espacio (25).

Para el desarrollo de la solución se utilizaron los marcos de trabajo JQuery v2.0 para el trabajo con el lenguaje JavaScript, Bootstrap v3.0 para el trabajo con CSS y como framework PHP se utilizó Symfony v2.7.16. La elección de estos marcos de trabajo viene dada por sus características y ventajas, además de ser los definidos y utilizados en el proyecto al cual va dirigido el presente trabajo de diploma.

1.5.3. Tecnologías

Ajax

Ajax son las siglas de *Asynchronous JavaScript And XML*, (JavaScript asíncrono y XML). No es una tecnología en sí mismo, sino que se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes (26).

Ajax permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. Disminuye, además, el tiempo de espera ante la petición de un usuario y reduce el tráfico al servidor (26).

Se utilizó Ajax en el desarrollo del módulo para lograr que se realicen las peticiones sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en el módulo.

1.5.4. Entorno de desarrollo integrado

Un Entorno de Desarrollo Integrado, en inglés *Integrated Development Environment* (IDE), es un programa que está compuesto por un conjunto de herramientas que son usadas por el programador. Los IDE fueron diseñados para proporcionar un único programa en el cual se pueda llevar a cabo todo el desarrollo de un sistema y aumentar la productividad de los programadores, y así proporcionar componentes necesarios para la creación de interfaces de usuarios (26).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, entre otros. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (26).

NetBeans

Netbeans es un entorno de desarrollo gratuito y de código abierto que se encuentra actualmente en su versión 8.2. Posee todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java. Incluye también el control

de versiones, lo cual representa una venta a debido a que permite administrar las diferentes versiones del código fuente (27).

Para el desarrollo del módulo deseado se utilizó como IDE NetBeans v8.0, puesto que es de código abierto, multiplataforma y soporta lenguajes dinámicos como PHP y JavaScript. Además, permite la integración con el marco de trabajo Symfony 2, el cual posee algunas bibliotecas que el programador puede agregarle a su aplicación.

1.5.5. Sistema Gestor de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener bases de datos, proporcionando acceso controlado a las mismas. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos. Su objetivo fundamental consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modelo de almacenamiento de los datos en la computadora, ni el método de acceso empleado (28).

PostgreSQL

PostgreSQL es un sistema de administración de bases de datos relacionales de código abierto (DBMS) desarrollado por un equipo mundial de voluntarios. PostgreSQL no está controlado por ninguna corporación u otra entidad privada y el código fuente está disponible de forma gratuita. PostgreSQL destaca por su amplísima lista de prestaciones que lo hacen capaz de competir con cualquier SGBD comercial (29).

1.5.6. Servidor web

Un servidor web es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Mientras que comúnmente se utiliza la palabra servidor para hacer referencia a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones antes descritas (30).

Nginx

Nginx es un servidor web y proxy inverso de código abierto ligero de alto rendimiento, creado por Igor Sysoev y lanzado en octubre de 2004 bajo la Licencia BSD[2] simplificada, que incluye servicios de correo electrónico con acceso al internet Message Protocol (IMAP)[3] y al servidor Post Office Protocol (POP)[4] (31).

Una de las principales razones para utilizar el servidor web Nginx es que se trata de un software que es asíncrono, a diferencia de Apache que está basada en procesos. La ventaja principal de ser asíncrono, es su escalabilidad. En un sistema basado en procesos, cada conexión simultánea requiere de un hilo, lo que puede llevar a sobrecargar el servidor, mientras que en un servidor asíncrono se gestionan las peticiones en muy pocos hilos, reduciendo las posibilidades de sobrecarga en el servidor (31).

1.5.7. Sistema de control de versiones

El control de versiones (VCS de aquí en adelante) es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante (32).

El VCS permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa que, si se dañan o pierden archivos, se pueden recuperar fácilmente (32).

Git

Un sistema de control de versiones distribuido diseñado por Linus Torvalds en el año 2005. Surge como alternativa a BitKeeper[5], un control de versiones privativo que usaba en ese entonces para el kernel (32).

Git es liberado bajo una licencia GNU GPLv2 [6]y su última versión estable fue publicada a inicios de abril de 2017. Por sus disímiles ventajas, se ha convertido en uno de los más usados alrededor del mundo,

[2] Licencia de software otorgada principalmente para la distribución de software Berkeley.

[3] Protocolo de aplicación que permite el acceso a mensajes almacenados en un servidor de Internet.

[4] Protocolo utilizado en clientes locales de correo para obtener los mensajes de correo electrónico almacenados en un servidor remoto.

[5] Es un sistema de control de versiones distribuido para el código fuente de los programas producidos a partir de BitMover Inc.

[6] Licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto que garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

puesto que no depende de acceso a la red o un repositorio central; además, está enfocado en la velocidad, uso práctico y manejo de proyectos grandes (32).
Para el desarrollo del módulo se utilizó como SGBD PostgreSQL v9.5.12, como Servidor web Nginx v1.10.3 y como VCS Git v2.7.4.

1.6. Metodologías de desarrollo de software

Las metodologías de desarrollo de *software* imponen un proceso de forma disciplinada con el objetivo de hacerlo más predecible y eficiente, definiendo una representación que permite facilitar la manipulación de modelos y el intercambio de información entre todas las partes involucradas en la construcción de un sistema (33).

Las metodologías se clasifican en dos grupos:

Metodologías tradicionales: Están orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán.

Metodologías ágiles: Están orientadas a la interacción con el cliente y el desarrollo incremental del *software*, mostrando versiones parcialmente funcionales del *software* al cliente en determinados intervalos de tiempo, para que pueda evaluar y sugerir cambios en el producto.

La tabla 1 muestra las diferencias entre estos dos grupos de metodologías (34).

Tabla 1: Comparación sobre las metodologías de software. (Fuente: Elaboración propia).

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos.
Pocos Roles, más genéricos y flexibles.	Más Roles, más específicos.
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes) y equipos pequeños (menos de 10 integrantes).	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente usadas en proyectos grandes y con equipos posiblemente dispersos.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
La arquitectura se va definiendo y mejorando a lo largo del proyecto.	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.	Énfasis en la definición del proceso: roles, actividades y artefactos.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

A continuación, se hace un breve análisis sobre la metodología ágil AUP y como esta se adapta a la variación UCI.

Agile Unified Process (**AUP**, Proceso Unificado Ágil según sus siglas en inglés) es una versión simplificada del Proceso Unificado de Rational (RUP) desarrollada por Scott Ambler. AUP aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado ágil, gestión de cambios ágil y refactorización de base de datos para mejorar la productividad (35).

Esta metodología pasa por diferentes fases en su ciclo de vida como son:

- **Inicio:** Se encarga de definir el alcance del proyecto, los costes y plazos, los riesgos y la factibilidad del mismo.
- **Elaboración:** Encargada de identificar y validar la arquitectura.
- **Construcción:** Tiene como objetivo la creación de la documentación de apoyo, así como elaboración del *software* de forma incremental.
- **Transición:** El propósito es realizar las pruebas al proyecto para su posterior liberación funcional.

1.6.1. Metodología AUP-UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se tomará como referente el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (34).

1.6.2. Descripción de las fases de AUP en su variación UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mejor comprensión se muestra la tabla 2 (34)..

Tabla 2: Descripción de la fase de AUP-UCI

Fases AUP	**Fases Variación AUP-UCI**	**Objetivos de las fases (Variación AUP-UCI)**
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.6.3. Escenarios en cuanto a Requisitos

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

Escenario No1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Ilustración 2. Escenario No1 Metodología AUP-UCI

Escenario No2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

*Ilustración 3***.** Escenario No2 Metodología AUP-UCI

Escenario No3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

*Ilustración 4***.** Escenario No3 Metodología AUP-UCI

Escenario No4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Ilustración 5. Escenario No4 Metodología AUP-UCI

1.6.4. Selección de la metodología de software AUP-UCI

Se ha seleccionado para el desarrollo de la herramienta para el despliegue en el Centro FORTES la metodología AUP en su variación UCI, debido a que describe una manera fácil y ágil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantiene validos en RUP. El AUP-UCI aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas.
- Modelado Ágil.
- Gestión de cambios ágil.
- Refactorización de Bases de Datos para mejorar la productividad.
- La metodología AUP-UCI es la adopción de muchas de las técnicas ágiles de XP y otros procesos ágiles que mantiene RUP.
- Dentro de la metodología se seleccionó el escenario número cuatro, ya que este se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

Por tanto, esta metodología guiará al desarrollo de la aplicación ya que se adapta a las condiciones existentes y además está avalada por el departamento de Calidad-UCI.

1.7. Lenguaje Unificado de Modelado (UML)

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas, más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML es un lenguaje que proporciona un vocabulario y reglas para permitir una

comunicación, este se centra en la representación gráfica de un sistema. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones (36):

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo pueda entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura visión.

1.7.1. Utilidad de UML

UML es un lenguaje para modelamiento de propósito general evolutivo, ampliamente aplicable. Se aplica a una multitud de diferentes tipos de sistemas, dominios y métodos o procesos.

- Como lenguaje de propósito general, se enfoca en un conjunto de conceptos para la adquisición, compartición y utilización de conocimientos emparejados con mecanismos de extensión.
- Como un lenguaje para modelamiento ampliamente aplicable, puede ser montado en diferentes tipos de sistemas, dominios (negocios, *software*) y métodos o procesos. Como un lenguaje para modelamiento soportable por herramientas, estas están disponibles para soportar la aplicación del lenguaje para especificar, visualizar, construir y documentar sistemas.
- Es utilizado para el modelamiento industrialmente estandarizado, es un lenguaje abierto y totalmente extensible reconocido por la industria (37).

1.7.2. Herramienta CASE

Visual Paradimg

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas, además es considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. Teniendo en cuenta sus características y los beneficios que brinda para la construcción de *software*, especialmente

referente al modelado, se decidió utilizar Visual Paradigm para UML para el modelado de la aplicación (38).

Características

- Soporta aplicaciones Web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Ventajas

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X y Solaris.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo *Visio y Rational Rose*.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Conclusiones parciales

Los conceptos asociados al dominio del problema enmarcado en este capítulo, enriquecieron el conocimiento sobre Sistemas Informáticos, Sistemas de Gestión de Aprendizaje y actividades que utilizan elementos de la gamificación. De igual forma, aunque las plataformas similares estudiadas no satisfacen en su totalidad las necesidades actuales del módulo, el estudio de las mismas permitió

entender cómo funcionan el sistema de posiciones de las mismas para generar el ranking y aportaron a la selección de algunas características importantes para el desarrollo del módulo.

Además, la metodología AUP en su variante UCI permitió guiar el proceso de desarrollo de software, logrando estandarizar el ciclo de vida del software, dando cumplimiento a las buenas prácticas que define CMMI-DEV v1.3.

Capítulo II: Propuesta de Solución

Luego de elaborar el marco teórico y seleccionar las herramientas y la metodología de desarrollo de software a emplear, se han creado las condiciones para realizar una propuesta de solución que satisfaga las necesidades del problema de investigación. Este capítulo está enmarcado en el análisis y diseño para la futura implementación del módulo y se tiene como objetivo describir la propuesta que se plantea para establecer una solución, realizar la especificación de los requisitos del software y establecer los patrones que ayuden a modelar y entender los objetos del sistema. Además, se generan una serie de artefactos que conllevan a la etapa de implementación del módulo, cuyo nombre se ha definido como Módulo de gamificación basado en *ranking*.

2.1. Análisis y Diseño

Análisis y Diseño es una disciplina de la metodología de desarrollo de software AUP en su variante UCI. En la misma, si es necesario, se considera que los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte los requisitos, incluyendo los requisitos no funcionales (34).

2.1.1. Modelo de Dominio

Un modelo de dominio o conceptual (ilustración 6) explica los conceptos significativos en un dominio del problema; es el artefacto más importante a crear durante el análisis orientado a objetos. La designación de modelo de dominio ofrece la ventaja de subrayar fuertemente una concentración en los conceptos del dominio, no en las entidades del software (36).

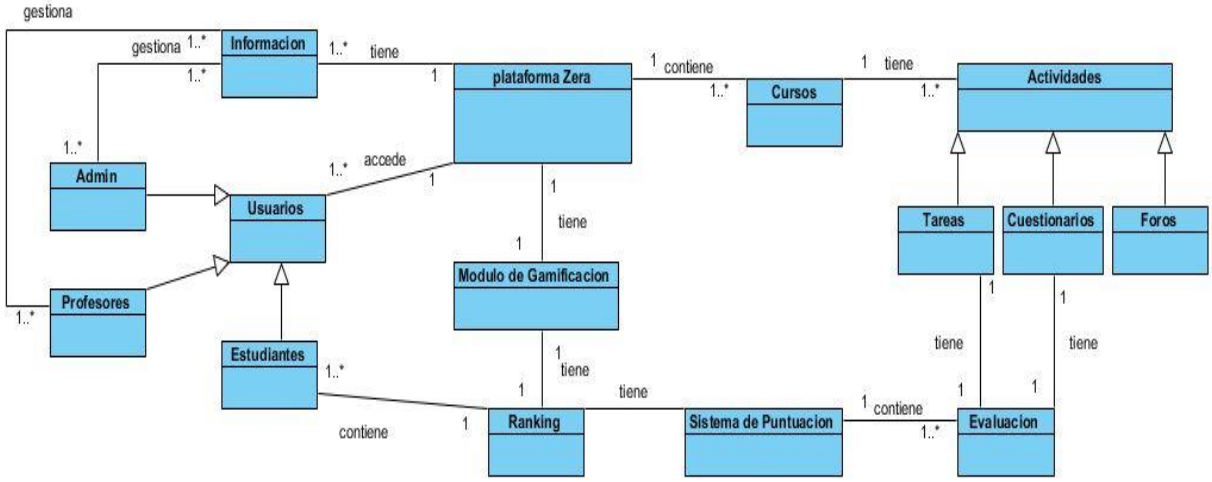


Ilustración 6. Diagrama Modelo de dominio (fuente: Elaboración propia)

Definición de los conceptos del Diagrama de Modelo de Dominio.

Plataforma ZERA: Es la Plataforma destinada a apoyar el PEA en la UCI, permite a los distintos usuarios del sistema intervenir en un ambiente dinámico e interactivo.

Usuario: Hace referencia a los distintos roles con que se puede acceder a la Plataforma Educativa XAUCE ZERA. Estos roles pueden ser estudiante, profesor o administrador.

Estudiante: Persona que recibe los cursos diseñados por el profesor en la Plataforma Educativa XAUCE ZERA.

Profesor: Persona encargada de diseñar y publicar los cursos en la Plataforma Educativa XAUCE ZERA.

Administrador: Persona encargada de administrar los cursos en la Plataforma Educativa XAUCE ZERA.

Evaluación: Es la forma que se utilizará para llevar a cabo la evaluación de las actividades.

Módulo de Gamificación: Bundle de Gamificación perteneciente a la plataforma ZERA.

Ranking: Elemento mecánico de gamificación.

2.2. Descripción de la propuesta de solución

En la plataforma ZERA del Centro FORTES no cuenta con un sistema de gamificación basado en ranking. Este sistema basado en ranking se va a generar por las puntuaciones de las evaluaciones de los estudiantes, en el cual se pondrá de forma escalonada a los estudiantes de lo que sean capaz de realizar en el sistema de la plataforma ZERA. Este sistema de puntos se define en base a puntos ganados respecto al turno anterior, puntos ganados por mantener posición. También que a partir de este sistema de punto se podrá ver escalonadamente a los estudiantes de acuerdo a un curso seleccionado por el usuario, sea estudiante o profesor, lo cual permite realizar una comparación de acuerdo al ranking que se genera después de seleccionar un curso uno de estos cursos.

2.2.1. Obtención del ranking de estudiantes para la plataforma ZERA

El módulo a realizar para la plataforma ZERA es de gamificación basada en ranking y en todo ranking a realizar se debe tener en cuenta criterios de comparación; el criterio de comparación en este módulo será por las evaluaciones de los estudiantes.

Primeramente, el ranking se va a generar partir de evaluaciones de estudiantes (**Ee**) en el cual van a estar asociados los estudiantes (E) y cada vez que un estudiante se evalúa se debe actualizar el sistema.

Cada estudiante(**E**) tiene un vector de cursos (**Cu**).

$E \rightarrow (Cu_1, Cu_2, \ldots Cu_n)$ donde Cu_i, es el curso i

Cada curso (**Cu**) tendrá asociado un vector de actividades (**Act**).

$Cu \rightarrow (A_1, A_2, \ldots A_n)$ donde A_i es la actividad i

Cada actividad tendrá asociado la nota (**N**) y su complejidad (**C**), en el cual la nota tomará valor -1 si el estudiante no ha realizado la actividad, en cualquier otro caso tomará valores 2, 3, 4, y 5. Las actividades serán tipo tareas y cuestionarios, definiendo que este último tipo de ejercicio presenta 4 niveles de complejidad. La complejidad de este tipo de actividad está definida por las categorías: very easy, easy, hard, very hard. Estos valores están en un intervalo de entre 0 y 1, por lo que tomarán valores de 0.25, 0.5, 0.75, 1. En caso de las actividades de tipo tareas, no presentan complejidad definida en el sistema de la plataforma ZERA, por lo que se le define como complejidad 1.
Con las actividades se realiza un promedio de evaluaciones por curso en el cual se calcula a partir de la siguiente ecuación 1:

$$Cu = \sum_{i=1}^{n} A_i$$

Ecuación 1: Sumatoria de las actividades de un curso (Fuente: Elaboración propia).

Las actividades se calcularían de la forma (ecuación 2):

$$A_i = N_i * C_i$$

Ecuación 2: Producto de la actividad i por la complejidad i. (Fuente: Elaboración propia).

, donde N_i es Nota de la actividad i y C_i es la complejidad de la actividad i

De forma general como se explica anteriormente, el ranking o la tabla de posiciones se genera a partir de los estudiantes y cada estudiante tendrá la suma promedio de los cursos, la ecuación 3 sería la siguiente:

$$E = \sum_{i=1}^{n} Cu_i$$

Ecuación 3: Sumatoria del total de evaluaciones de los cursos. (Fuente: Elaboración propia).

Para conseguir que el ranking sea los más equilibrado posible en cuanto a la nota de las actividades y a la complejidad, la tabla 3 muestra los valores de las notas contra las complejidades de las actividades realizadas por los estudiantes. Esto se realiza con la necesidad de que el sistema pueda posicionar a los usuarios, debido a que, si un usuario realiza una actividad con una complejidad muy difícil, pero obtuvo una nota de 3 y existe otro usuario que realizo una actividad con una complejidad muy fácil y obtuvo una nota de 5, el sistema debe tener en cuenta estos aspectos a la hora de posicionar y así sucesivamente para todo los demás casos y variaciones existentes.

Tabla 3: Valores de Dificultad de las actividades contra las notas de las actividades. (Fuente: Elaboración propia).

	Notas			
Dificultad	2	3	4	5
Muy fácil (0.25)	0	0.75	1	1.25
Fácil (0.5)	0	1.5	2	2.5
Medio (0.75)	0	2.25	3	3,75
Difícil (1)	0	3	4	5

2.3. Especificación de requisitos del software

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. Los requisitos tratan de establecer lo que el sistema debe de hacer, sus propiedades emergentes deseadas, esenciales y las restricciones en el funcionamiento del sistema y los procesos de desarrollo de software. Estos se clasifican en dos grupos: requisitos funcionales y no funcionales .

2.3.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe de proporcionar el sistema, de la manera en que éste debe de reaccionar a entradas particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe de hacer. En otras palabras, los requisitos funcionales describen con detalles las funcionalidades del software, sus entradas, salidas y excepciones (33).

Especificación de los Requisitos Funcionales:

A continuación, se describen los requisitos funcionales del módulo a implementar:

RF1: Mostrar ranking general de usuarios. El sistema debe permitir visualizar una tabla de posiciones en la vista principal de la plataforma ZERA. Esta vista va a estar definida en un bloque que se llamará ranking de usuarios que solo mostrará los 5 usuarios con más puntuación.

RF2: Mostrar ranking por curso. El sistema debe permitir visualizar una tabla de posiciones referente a cada curso creado en la plataforma.

RF3: Comparar usuarios en el curso. El sistema debe permitir que se pueda comparar un usuario del ranking de curso con otro usuario. El usuario que está realizando la comparación solo permite realizar la comparación entre el usuario que está haciendo la comparación con otro usuario.

RF4: Comparar usuarios en el curso por el profesor. El sistema debe permitir que un profesor pueda comparar a dos usuarios. El profesor debe seleccionar a dos estudiantes cualquiera que se encuentren en el ranking.

RF5: Filtrar estudiante en el ranking general. El sistema debe permitir encontrar un estudiante deseado por en nombre o por la puntuación que se encuentre en el ranking.

Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funcionalidades ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y normas o estándares. Además, no se aplican regularmente a rasgos o servicios individuales del sistema (39). Para el posicionamiento de los requisitos no funcionales se utilizaron las dadas por diferentes autores, obteniendo como clasificaciones las siguientes (33):

RNF 1. Requisitos de Seguridad:

RNF 1.1. Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo a los roles que posean.

RNF 2. Requisitos de Usabilidad:

RNF 2.1. Cada nombre de los botones existentes en el sistema debe estar relacionado con la función que se realiza al oprimirlo.

RNF 2.2. El módulo debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios.

RFN 3. Requisitos de Implementación:

RNF 3.1. SGBD: PostgreSQL v9.5.12.

RNF 3.2. Servidor web: Nginx v1.10.7 o Superior.

RNF 3.3. Sistema de Control de Versiones Git v2.7.4.

RNF 3.4. PHP v7.0, CSS v3.0, HTML v5.0 y JavaScript v2.1.

2.4. Historias de Usuario

La metodología de desarrollo seleccionada define cuatro escenarios para llevar a cabo la modelación del sistema en los proyectos. De acuerdo a las características del proyecto de desarrollo a quien tributa la propuesta de solución, se selecciona el escenario #4. En este escenario, el proyecto evaluó previamente el negocio a informatizar y como resultado obtuvo un negocio muy bien definido. Además, el cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. El artefacto que se genera a través de este escenario es la historia de usuario (34).

La historia de usuario es una técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (34).

A continuación, se presenta las historias de usuario pertenecientes al módulo.

Tabla 4: Historia de Usuario del Requisito funcional Mostrar ranking general de estudiantes [Fuente: Elaboración propia]

Número: 1	**Nombre del requisito:** Mostrar ranking general de estudiantes
Programador: Bernardo Sarría Fuentes	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alto	**Tiempo Real:**

1- Objetivo:

-Permitir mostrar un ranking general en el sistema donde solo se visualizarán los 5 usuarios con más puntuación.

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

Para mostrar el ranking general hay que:

-Estar autenticado en el sistema.

2.1- Flujo de la acción a realizar:

-El sistema debe permitir mostrar en un bloque en la página principal del sistema los 5 mejores estudiantes con más puntuación.

-El sistema debe permitir que en el bloque la funcionalidad del botón Ver Todos, muestre en un modal o modelo el ranking general con todos los usuarios existentes.

Prototipo de interfaz:

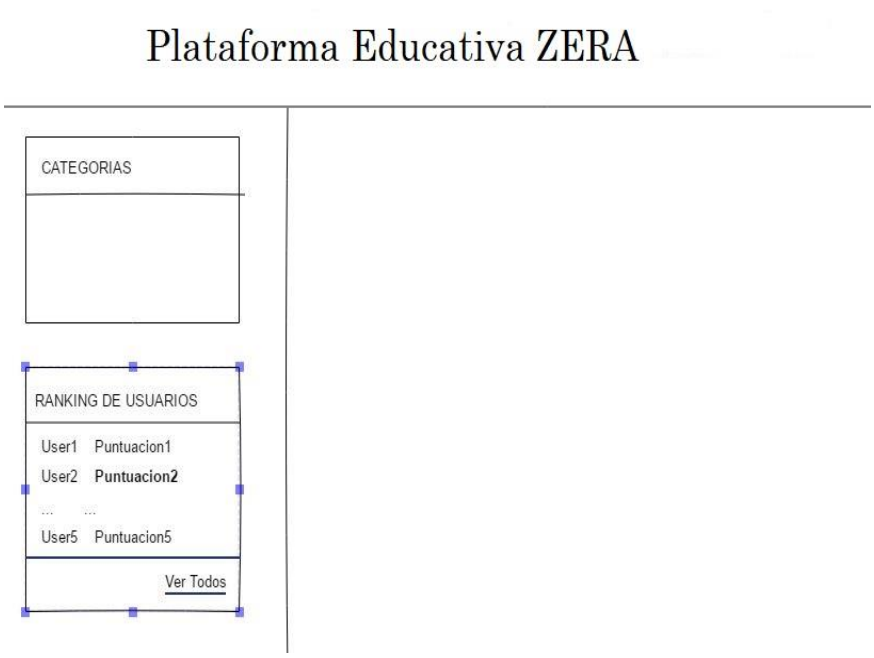


Tabla 5: Mostrar ranking de estudiantes por curso. (Fuente: Elaboración propia).

Número: 2	**Nombre del requisito:** Mostrar ranking de estudiantes por curso
Programador: Bernardo Sarría Fuentes	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: Alto	**Tiempo Real:**

1- Objetivo:

-Permitir mostrar un ranking de estudiantes en cada curso que se encuentre en el sistema donde solo se visualizarán los estudiantes que estén matriculados en cada curso.

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

-Para mostrar el ranking de estudiantes por curso hay que:

Acceder a un curso, en la configuración del curso se muestra la funcionalidad Ranking del Curso

2.1- Flujo de la acción a realizar:

-El sistema debe permitir mostrar un ranking con el nombre y la puntuación de los usuarios del curso.

Prototipo de interfaz:

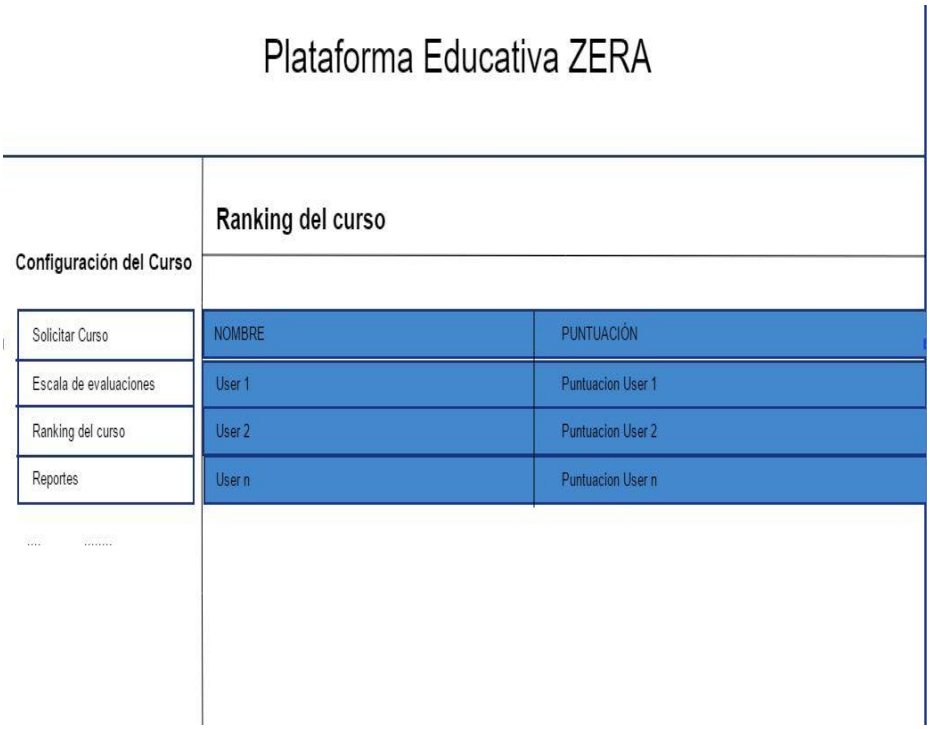


Tabla 6: Comparar Estudiante. (Fuente: Elaboración propia).

Número: 3	**Nombre del requisito:** Comparar Estudiante
Programador: Bernardo Sarría Fuentes	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1 días

1- Objetivo:

-Permitir comparar usuarios que participen en el ranking.

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

-Para comparar usuarios hay que:

Acceder a un curso.

2.1- Flujo de la acción a realizar:

-El sistema debe permitir realizar una comparación en la que el estudiante se compare con otro estudiante.

Prototipo de interfaz:

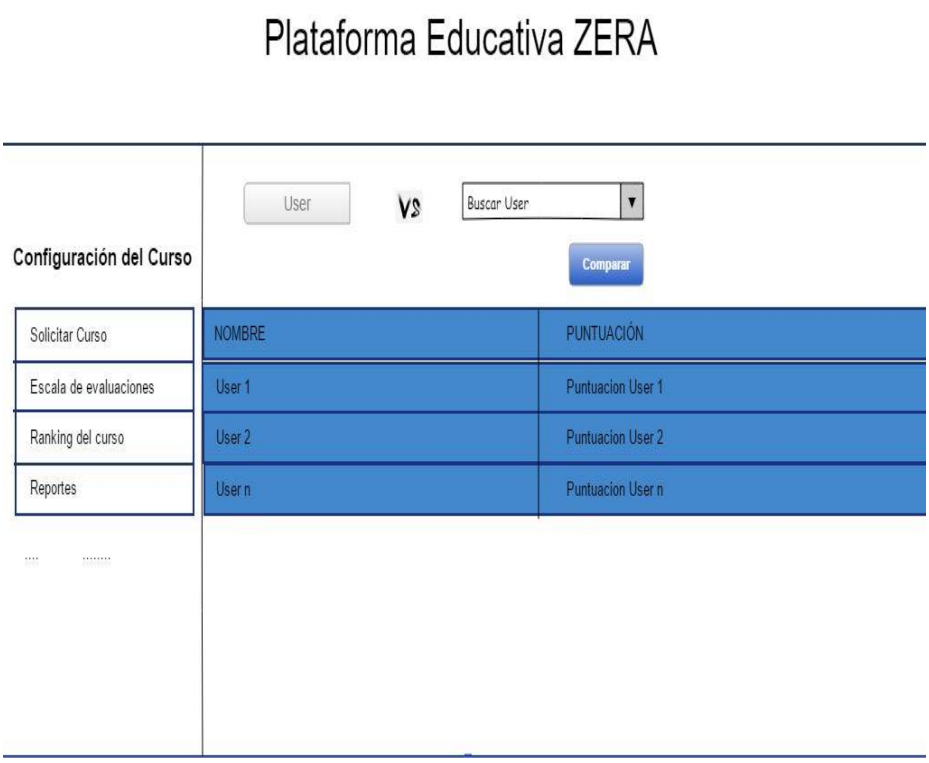


Tabla 7: Comparar estudiantes por el profesor. (Elaboración propia).

Número: 4	**Nombre del requisito:** Comparar estudiantes por el profesor
Programador: Bernardo Sarría Fuentes	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Alto	**Tiempo Real:**

1- Objetivo:

-Permitir comparar usuarios que participen en el ranking.

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

-Para mostrar el ranking general hay que:

Acceder a un curso.

2.1- Flujo de la acción a realizar:

-El sistema debe permitir realizar una comparación en la que el profesor pueda seleccionar dos estudiantes y compararlos.

Prototipo de interfaz:

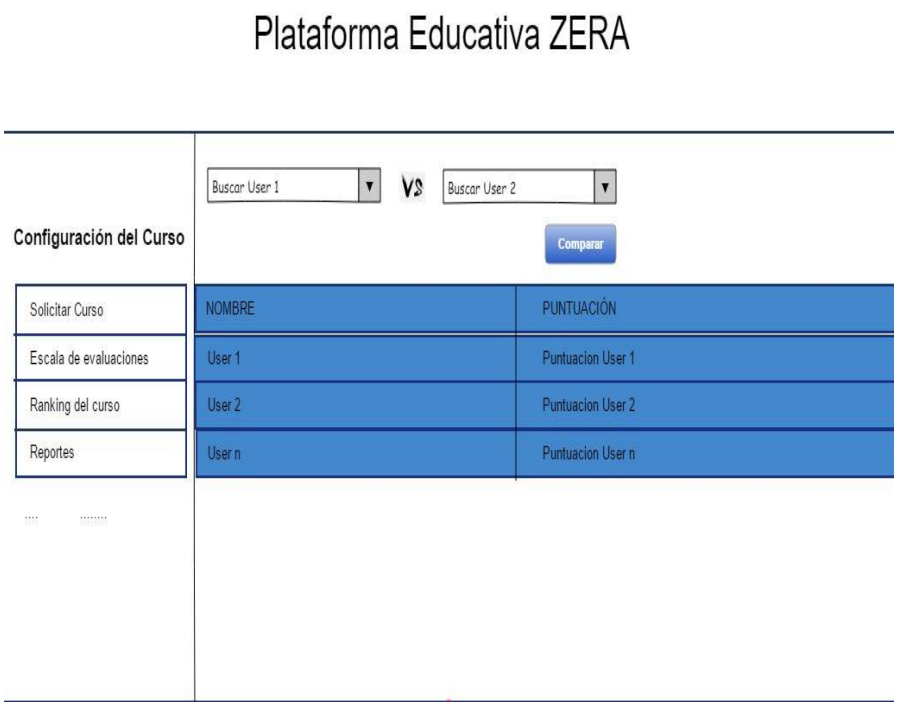


Tabla 8: Filtrar Estudiante. (Fuente: Elaboración propia).

Número: 5	**Nombre del requisito:** Filtrar Estudiante
Programador: Bernardo Sarría Fuentes	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: Alto	**Tiempo Real:**

1- Objetivo:

-Permitir buscar en el sistema los usuarios por el nombre o por la puntuación.

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

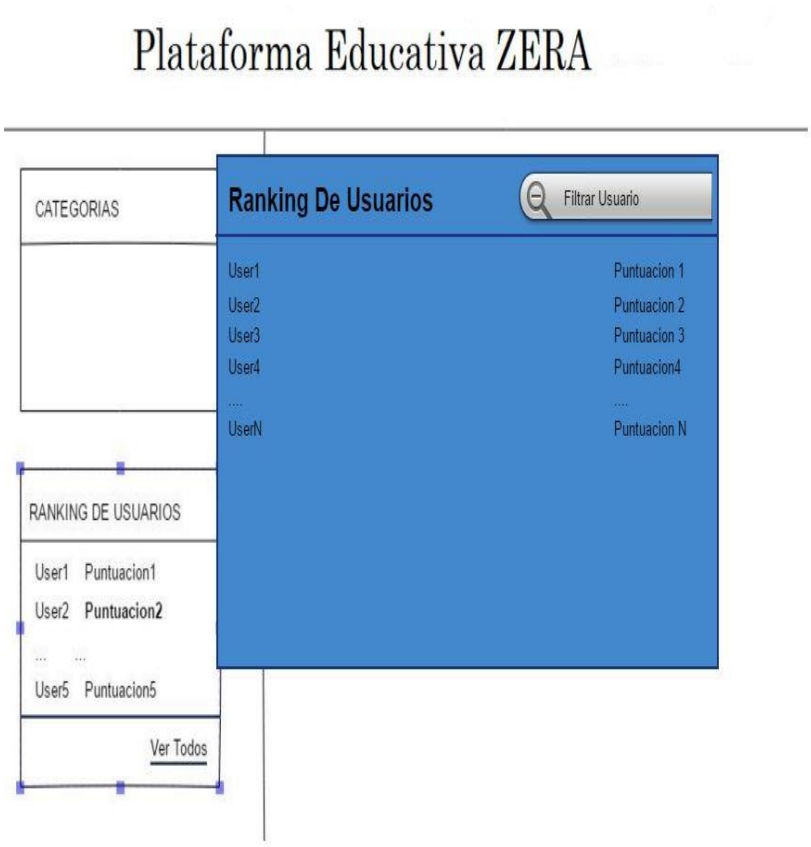
-Para filtrar el usuario hay que:

Acceder a la funcionalidad del botón en el bloque Ranking de usuarios (Ver Todos).

2.1- Flujo de la acción a realizar:

-El sistema debe permitir buscar el/los usuarios por el nombre o por la puntuación.

Prototipo de interfaz:



2.4.1. Diseño

La esencia del diseño de software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa la organización lógica como un modelo en un lenguaje definido de modelado tal como UML y otras veces simplemente utiliza notaciones informales y esbozos para representar el diseño.

2.5. Arquitectura de software

La Arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (39).

Patrón Modelo-Vista-Controlador

El patrón arquitectónico *Model-View-Controller* (MVC) divide una aplicación interactiva en tres componentes. El Modelo contiene la funcionalidad central y los datos, las Vistas despliegan la información al usuario y los Controladores manejan la entrada del usuario. Estos dos últimos componentes, en conjunto, forman la interfaz del usuario. Un mecanismo de propagación de cambio asegura la consistencia entre la interfaz del usuario y el modelo (37).

Cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente (40):

1. El sistema de enrutamiento determina qué Controlador está asociado con la página de la portada.
2. Symfony ejecuta el Controlador asociado a la portada. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.
3. El Controlador solicita al Modelo los datos necesarios. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
4. Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
5. El Controlador entrega al servidor la página creada por la Vista.

Para el desarrollo del módulo deseado se utilizó Symfony 2, el cual es un framework de tipo MVC escrito en PHP 5 que ha sido diseñado para obtener lo mejor de dicho patrón y una mayor facilidad de uso (ilustración 7).

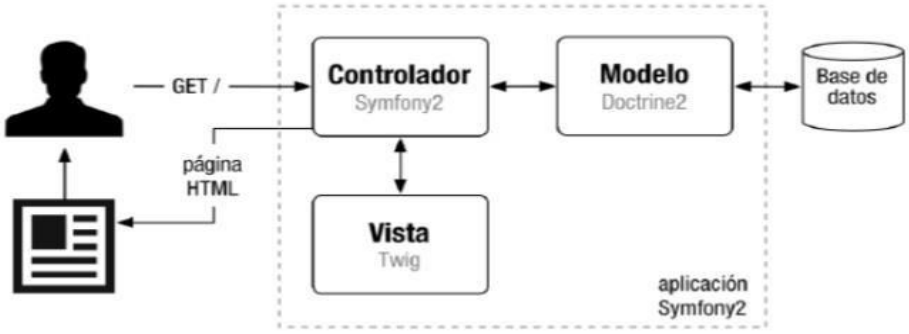


Ilustración 7. Patrón Modelo-Vista-Controlador

2.6. Diagrama de clases del diseño

El diagrama de clases del diseño (DCD) describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real (36).

A continuación, se presenta el DCD (ilustración 8) perteneciente al requisito funcional Mostrar ranking general; el resto de los artefactos de este tipo se encuentran en el directorio Diagramas de Clases de Diseño.

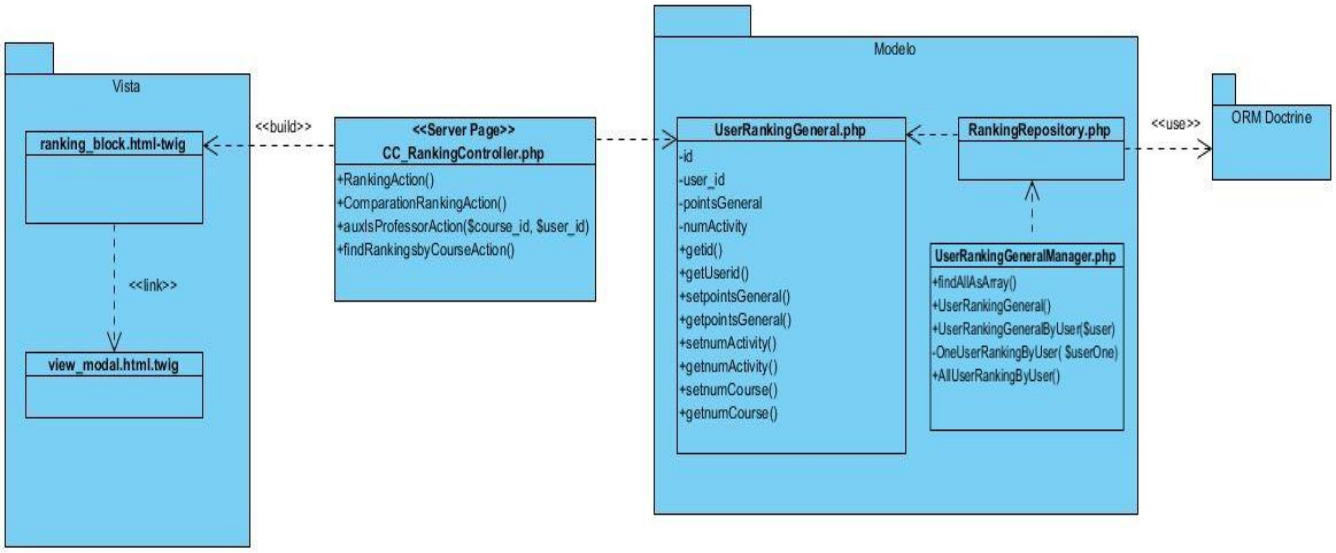


Ilustración 8. Diagrama de CD correspondiente al ranking general de estudiantes (Fuente: Elaboración propia).

2.7. Patrones del diseño

Un patrón de diseño describe una estructura que resuelve un problema de diseño en particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón (33).

Los patrones se evidencian en dos grupos como se muestra en la ilustración 9:

Ilustración 9. Grupos de patrones. (Fuente: Elaboración propia).

Patrones GRASP

Los patrones generales de software para asignar responsabilidad (GRASP por sus siglas en inglés *General Responsibility Asignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (33).

Dentro de los patrones GRASP utilizados para el diseño del módulo y que incluye por defecto en su arquitectura Symfony, situándolos en las capas Modelo y Control que plantea el patrón arquitectónico MVC, se destacan los siguientes:

Experto: Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan (33).

Creador: Este patrón permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias a formularios y entidades, para la vista del usuario (33). En la propuesta de solución, este patrón se evidencia en las clases RankingController.

Controlador: Este se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye un único punto de entrada para cada evento (33). En la solución propuesta este patrón se evidencia en las clases controladoras RankingController.

Alta cohesión: En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (33). En la solución propuesta este patrón está evidenciado en las clases que extienden de la clase Repository y Manager.

Bajo Acoplamiento: El acoplamiento mide el grado en que una clase está conectada, tiene conocimiento o de alguna manera depende de otra. Este patrón consiste en asignar la responsabilidad de manera que el acoplamiento permanezca bajo. El bajo acoplamiento permite crear clases más independientes, más reutilizables, lo que implica mayor productividad (33). En la solución propuesta este patrón se evidencia en las clases que extienden de Repository y Manager.

Patrones GOF

Los patrones GOF (por sus siglas en inglés *The Gang of Four*) son aquellos que describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos (41).

Singleton (Instancia Única): Permite el manejo de objetos únicos y que sean accesibles a otros objetos. Además, permite el acceso controlado a una única instancia. Symfony usa por defecto este patrón al crear las instancias de los servicios (33). Un ejemplo específico en la solución propuesta es el servicio fortes_ranking.manager, que tiene como función devolver una instancia de la clase RankingManager.

Patrón Decorador: El motor de plantillas Twig, está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas se realiza de una manera flexible y versátil (33). Este patrón se observa en todos los archivos de tipo .html.twig que se encuentran en el módulo y que son los que contienen el código HTML que es común para todas las páginas, por lo que cada página que se crea heredará de estos.

Patrón Inyección de Dependencias: Se puede apreciar en el objeto especial entity manager (em) a través del cual Doctrine2 realiza la manipulación de la información (buscar, insertar, modificar y eliminar registros en las tablas) sin que el programador tenga que escribir sentencias SQL (33).

Complejo: Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, debido que al poseer todos ellos una interfaz común, se tratan todos de la misma manera. Este se evidencia en la clase UserRankingGeneralManager (33).

2.8. Diagrama de despliegue

Un diagrama de despliegue es un artefacto generado en la disciplina Análisis y Diseño que muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución *run-time* en las instancias de los nodos de proceso (36).

El diagrama de despliegue que se muestra en la ilustración 10 representa la distribución física del sistema a través de nodos. Está compuesto por una PC Cliente que deberá tener instalado un navegador web, donde la comunicación entre ella, el Servidor de Aplicaciones XAUCE ZERA y el Servidor Media se llevará a cabo a través del protocolo HTTP[7]. Además, el Servidor de aplicaciones XAUCE ZERA se comunicará vía TCP[8] (Protocolo de Control de Transmisión), con el Servidor de Base de Datos XERA ZERA.

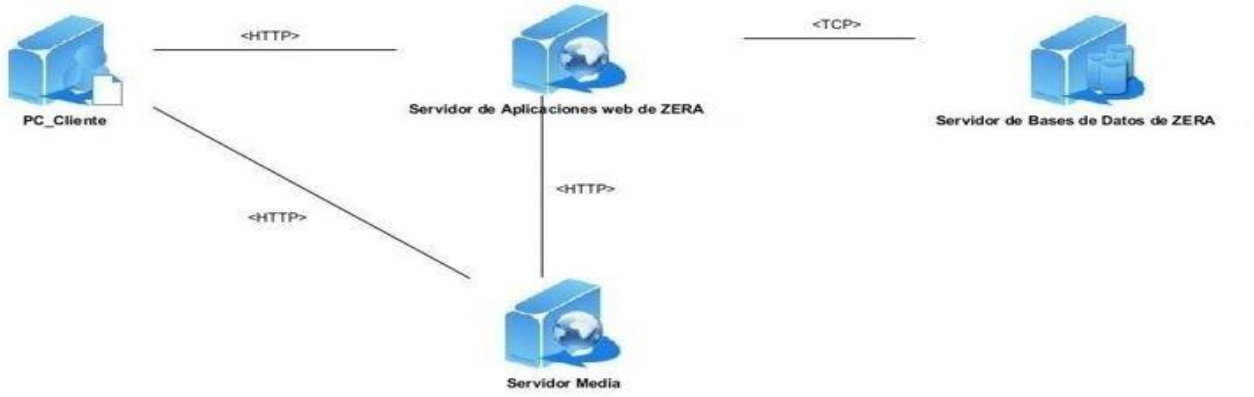


Ilustración 10. Diagrama Modelo de despliegue (fuente: Elaboración propia)

[7] protocolo de transferencia de hipertextos', que se utiliza en algunas direcciones de internet.

[8] Protocolo de control de transmisión. Da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, cliente etc.) y protocolos de aplicación

2.9. Diseño de la Base de Datos

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas, entre otros (42).

El modelo entidad-relación (MER) como se muestra en la ilustración 11, es un modelo de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones, incorporando una representación visual conocida como diagrama entidad-relación (42).

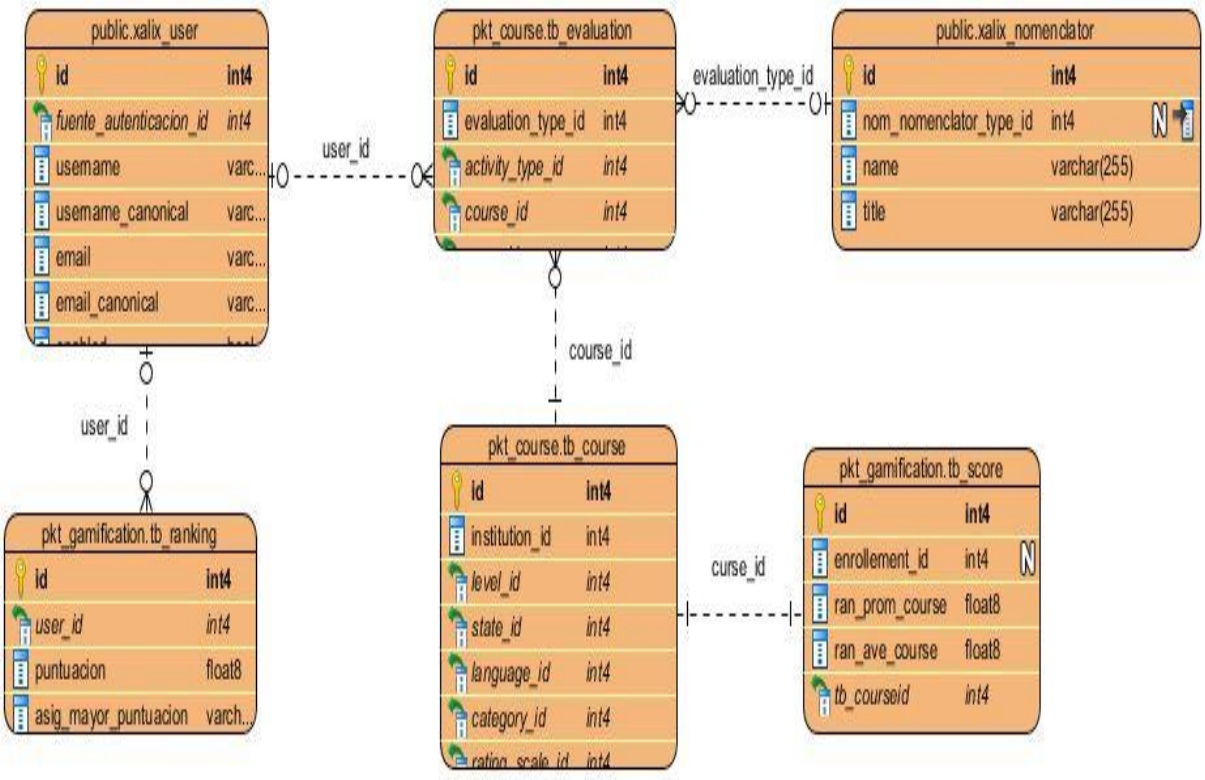


Ilustración 11. Modelo Entidad-Relación. (Fuente: Elaboración propia)**.**

Descripción de las tablas de la base de datos

En esta sección se presenta una breve descripción de cada uno de los atributos de la tabla 9 *"pkt*_gamification. tb_ranking*"*, el resto de las descripciones se encuentran en los Anexos (Anexo 1).

Tabla 9: Descripción de la entidad ranking de la base de datos de la Plataforma Educativa XAUCE ZERA. (Fuente: Elaboración propia)

pkt_gamification.tb_ranking		
Descripción: En esta tabla se agrupa la información correspondiente a ranking		
Atributo	Tipo	Descripción
Id	integer	Etiqueta única que identifica el ranking de usuarios
user_id	integer	Etiqueta única que identifica al usuario
puntuación	Float	Almacena la puntuación de cada usuario que va a estar definido en el ranking

Conclusiones parciales

La especificación de requisitos sustentada por 5 requisitos funcionales y 4 no funcionales brindó un enfoque del módulo a implementar. Igualmente, la confección de los artefactos definidos en la disciplina Análisis y Diseño de la metodología AUP en su variante UCI, permitió definir las bases necesarias para la implementación del módulo y posibilitó que se obtuvieran de una forma concreta y detallada las relaciones que se establecen entre las entidades del sistema. Además, la investigación y definición de la arquitecta de software y los patrones de diseño a utilizar, contribuyeron a lograr una implementación centrada en el diseño realizado.

Capítulo III: Implementación y Pruebas

Establecido el alcance del módulo, definida la arquitectura y concretados los patrones a utilizar, quedan creadas las condiciones para iniciar el proceso de implementación y pruebas al módulo. Este capítulo está enmarcado en la fase de implementación del sistema; además, se muestra el diagrama de componentes, así como los estándares de codificación que se deben seguir para generar el código fuente. Se describen y realizan las pruebas de software al módulo implementado haciendo uso de los diseños de casos de prueba.

3.1. Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema (33).

A continuación, la ilustración 12 muestra el diagrama de componentes del sistema:

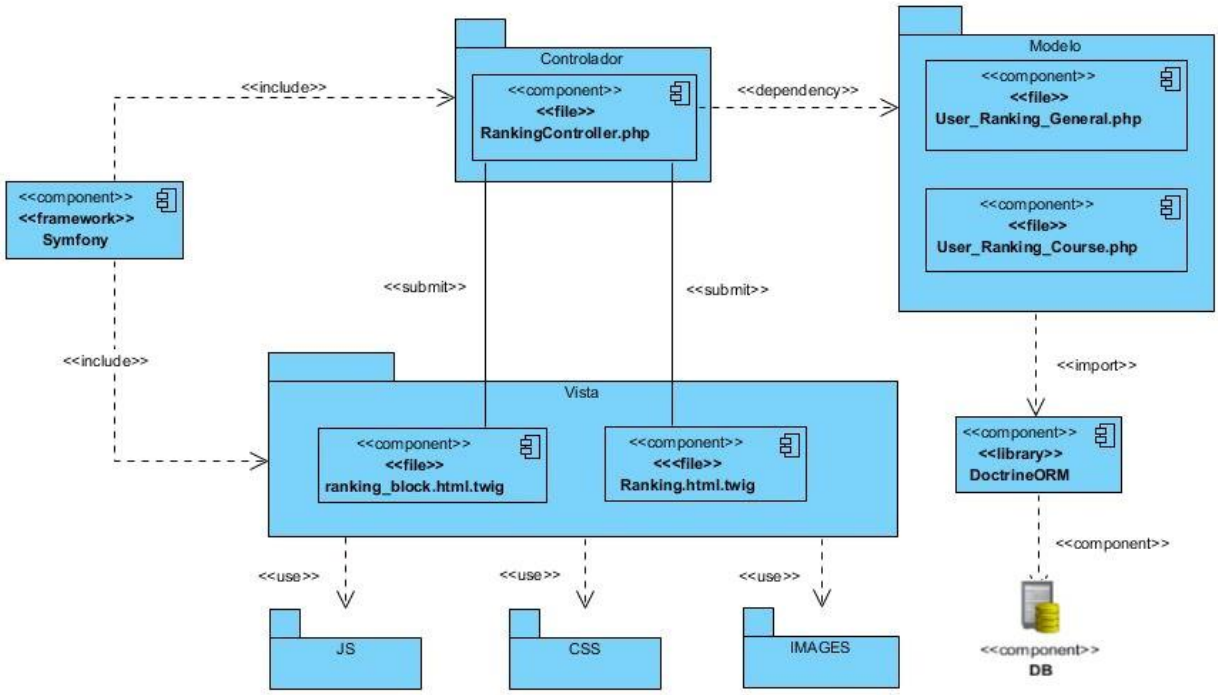


Ilustración 12. Diagrama de componentes. (Fuente: Elaboración propia).

3.2. Implementación

La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Los procesos de diseño y programación de software se encuentran implicados en esta etapa; si se utiliza un enfoque evolutivo de desarrollo, también puede incluir un refinamiento de la especificación del software.

3.2.1. Estándares de codificación

Un estándar de codificación (tabla 10) son reglas que se siguen para la escritura del código fuente. De tal manera que otros programadores puedan identificar las variables, las funciones o métodos (43).

Tabla 10: Estándares de codificación empleados en el módulo. (Fuente: Elaboración propia).

Descripción	Ejemplo
JavaScript	

Se utiliza la notación camelCase para los nombres de variables y funciones.
Todos los nombres comienzan con una letra. Se utiliza la palabra reservada "var" para declarar variables.
	var refreshRankingBlock = function(_paintSelector){
	
	var paintSelector = _paintSelector != undefined ? _paintSelector : '#ranking .matric_container div:last-child';
	
	$('#ranking .block-header-actions span#refreshOnlineRankingBtn').on('click', function(e){
	
	//var route = $(selector).parents('a').attr('href');
	//location.href = route;
	e.preventDefault();
	var route = xalixGenerateRoute('refresh_ranking_users');
	$.ajax({
	url: route,
	dataType: "html"
	}).done(function(respuesta){
	$(paintSelector).html(respuesta);
	refreshRankingBlock();
	showAllRankingUsers();
	initSlimScroll();
	initSearch_onlineUsers('#rankingUsersModal #online-user-searchbox');
	}).error(function(){
	
	});
	});
	
	};

	html.find('.plantilla_activity').append(editSpanActivity); html.find('.plantilla_activity > .templateactions').prepend(deleteSpanActivity); }

HTML	

No se debe colocar espacios entre la relación atributo-valor.
Se debe utilizar lowercase* para el nombre de los atributos de las etiquetas.
 	
* Se refiere a que las palabras deben estar escritas en minúscula
	{% extends
	'RankingBundle:Block:ranking_block.html.twig' %}
	{% trans_default_domain "RankingBundle"
	%}
	{% import '::macros.html.twig' as macros %}
	
	{% block stylesheets %}
	{{ parent() }} <link href="{{
	asset('bundles/ranking/css/ranking_all.css')
	}}" rel="stylesheet">
	////
	{% endblock stylesheets %}
	
	{% block header_menu %}
	
	{% endblock header_menu %}
	
	{% block pageContent %}
	
	<a class="btn btn-primary" href="{{

CSS

Una declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves. Para hacer un código CSS legible, ponga una declaración en cada línea. Coloque la llave que cierra en una línea nueva.	activity_fronted.css .option { margin- bottom: 0; padding: 0 1.0em; } .option label { border- bottom: 1px solid #fff; display: block; padding: 0.3em 0 0.1em; } .panel-container{ border: 1px solid #e5e5e5; height: auto; } #sources-container{ border- right: 1px solid #e5e5e5; height: auto; }
PHP	
Adicionar una línea en blanco antes de cada sentencia return, a menos que se encuentre como única sentencia (como en un if).	$rankingManager = $this->container- >get('fortes_ranking.manager'); $rankingList = $rankingManager- >findAllAsArray(); return $this->render('RankingBundle:Block:ranking_block.html.twig', array('rankingList' => $rankingList,

	)); }

Declarar los atributos de las clases antes de los métodos.	Class Ranking {
	
	/**
	* @var integer $id
	* @ORM\Column(name="id", type="integer")
	* @ORM\Id
	* @ORM\GeneratedValue(strategy="AUTO")
	*/
	private $id;
	
	public function getId()
	{
	return $this->id;
	}

Convención de nombres	

La declaración de funciones o métodos siempre comenzará con letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa camelCase.*
 	
*Se refiere al estilo de escritura donde la segunda palabra de una función con nombre compuesto será escrita en mayúscula.
	public function findRankingsAction(){
	
	cuerpo del método
	}

Se utiliza namespaces* para todas las clases.	namespace FORTES\RankingBundle\Entity;

*Representa la ruta de donde se encuentra cada clase en el módulo	
Para definir cada uno de los servicios hay que tener presente los siguientes indicadores: - Los nombres de los servicios contienen grupos separados por puntos. - El alias de Inyección de Dependencias del bundle es el primer grupo. - Se utiliza minúsculas para los nombres de servicios y sus parámetros. - Un nombre de grupo utiliza la notación guión bajo.	fortes_ranking.manager
Comentarios en el código	

Los comentarios utilizan las anotaciones siguientes:
@param: esta etiqueta provee el nombre, el tipo y la descripción de los parámetros de una función.
@var: esta etiqueta define qué tipo de dato se representa mediante la propiedad. @return: esta etiqueta es utilizada para documentar el valor que retorna una función.
	/**
	*@var text $description
	*
	*@ORM\Column(name="
	description",type="text",nullable="true"
	*/
	private $description;
	/**
	*Set name
	*@param mixed $name Nombre de la institución
	*@return void
	*/
	public function Setname($name){
	$this name->$name;
	}

3.3. Pruebas de software

Las pruebas de software son un conjunto de actividades que implican ejecutar una implementación del software con datos de prueba. Se examinan las salidas del software y su entorno operacional para comprobar que funciona tal y como se requiere (39). Tienen como objetivo descubrir y corregir el máximo de errores posibles antes de su entrega al cliente, asegurando así el correcto cumplimiento de las funcionalidades del producto (33).

Siguiendo los pasos de la metodología que guía el proceso de desarrollo de software en la presente investigación, así como las características del módulo y el sistema a quien tributa el mismo, las pruebas que se utilizaron se llevan a cabo en dos disciplinas: Pruebas internas y Pruebas de Aceptación. A continuación, se describen estas disciplinas a través de los principales elementos y los resultados obtenidos en cada una de estas.

3.3.1. Pruebas Internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.

Para evaluar las funcionalidades internas del módulo, se utilizó el método de caja blanca, el mismo es un tipo de prueba de software que se realiza sobre las funciones internas de un módulo para derivar los casos de prueba. Al emplear lo métodos de caja blanca, el ingeniero de software podrá derivar casos de prueba que (33):

- Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
- Se ejerciten los lados verdaderos y falsos de todas las decisiones lógicas.
- Se ejecuten todos los bucles dentro de sus límites operacionales.
- Se ejerciten estructuras de datos internos para asegurar su validez.

Para aplicar este método de prueba se empleó la técnica de la Ruta Básica, utilizando como ejemplo la función ***UserRankingGeneral***, perteneciente a la clase ***UserRankingGeneralManager***. La selección de la función se realizó teniendo en cuenta que la misma responde a una de las principales funcionalidades dentro del módulo, ya que es la que da inicio al proceso del ranking general y verificar la complejidad de las actividades.

La técnica de la Ruta Básica permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un

conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejecutar el conjunto básico deben garantizar que se ejecutan cada instrucción del programa por lo menos una vez durante la prueba (38). A continuación, en la ilustración 13, se muestra la numeración de los nodos definidos en cada porción del código.

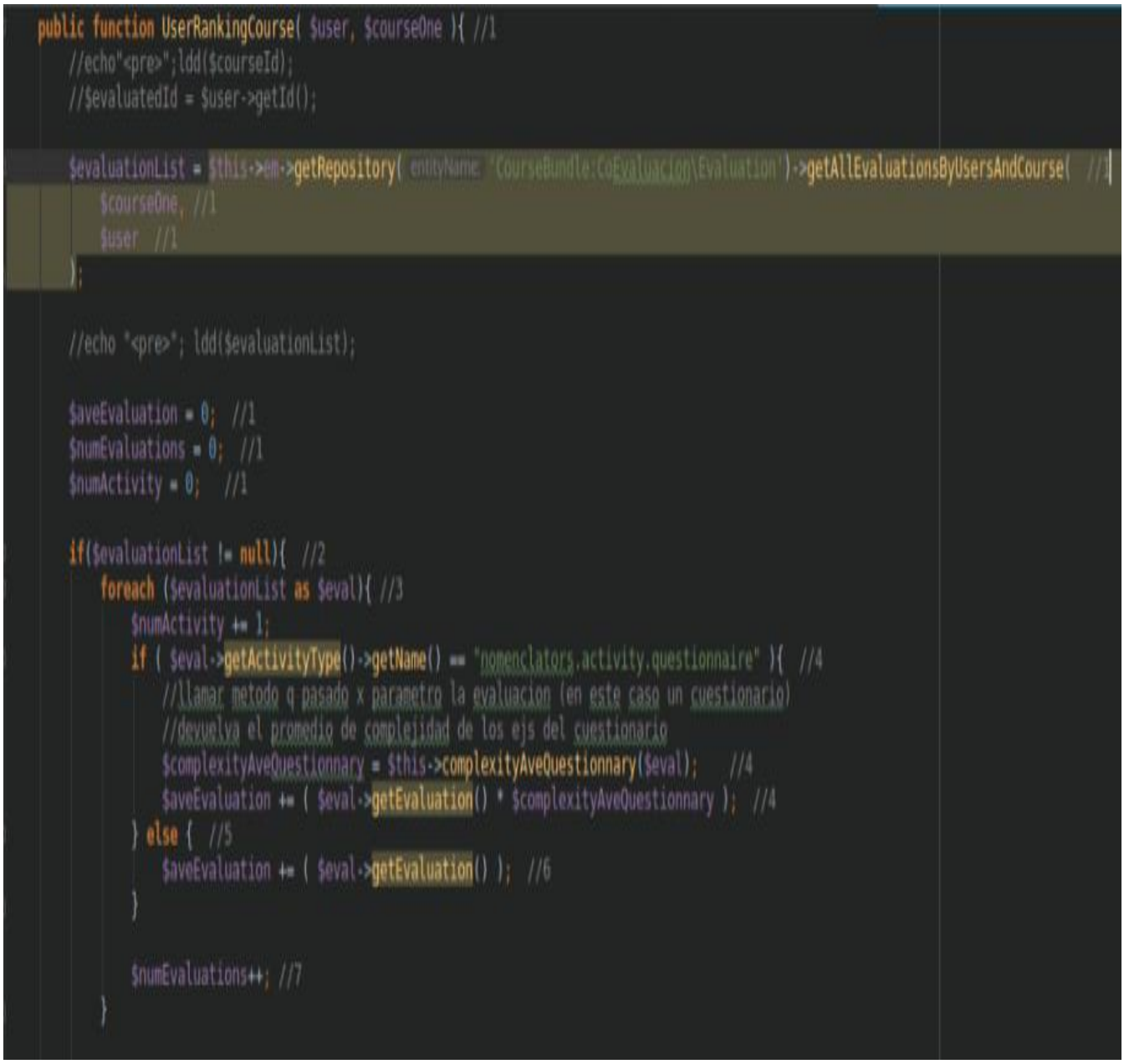


Ilustración 13. Numeración de los nodos en cada porción del código. (Fuente: Elaboración propia).

A continuación, se detallan los pasos que se realizaron para aplicar la técnica Ruta Básica:

1. **Confeccionar el grafo de flujo**: Utilizando el código de la ilustración 13 se realizó la representación del grafo de flujo, el cual describe un flujo de control lógico y está compuesto por los siguientes elementos (33):

- **Nodos de gráfica de flujo:** Son círculos que representan una o más instrucciones procedimentales.
- **Aristas o enlaces:** Son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- **Regiones:** Son las áreas delimitadas por aristas y nodos.

En la ilustración 14 se presenta el grafo de flujo obtenido:

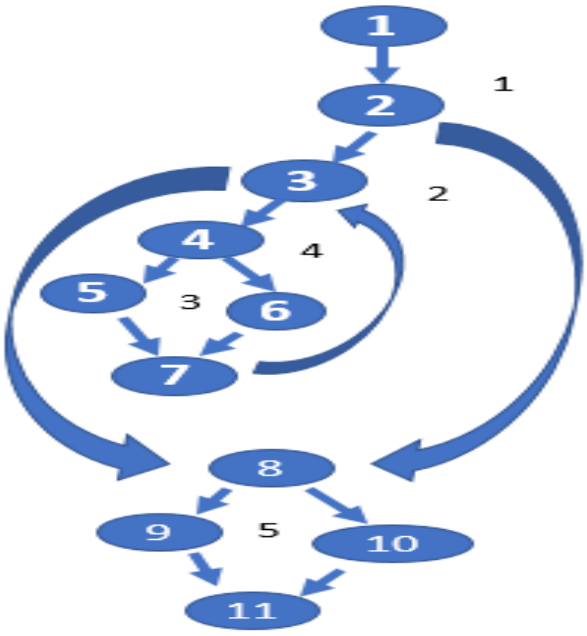


Ilustración 14. Grafo de flujo. (Fuente: Elaboración propia).

1. **Calcular la complejidad ciclomática**: Proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado al menos una vez.

La complejidad ciclomática se calcula mediante las tres formas siguientes (33):

- **V(G) = cantidad de regiones**

  V(G) = 5
- **V(G) = A – N + 2**, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

  V(G) = A – N + 2 = (14 – 11) + 2 = 5
- **V (G) = P + 1,** donde P es el número de nodos predicados contenidos en el gráfico de flujo, es decir, nodos de donde emergen 2 o más aristas.

3. **Determinar un conjunto básico de rutas linealmente independientes**: El valor de V(G) indica el número de rutas linealmente independientes de la estructura de control del programa, por lo que se definen las 2 rutas independientes obtenidas:

Ruta básica 1: 1-2-8-9-11

Ruta básica 2: 1-2-3-4-7-8-10-11

4. **Obtención de casos de pruebas**: Cada ruta independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino (33). En este caso se obtuvieron 2 rutas independientes, que dan lugar a la confección de igual número de casos de pruebas. A continuación, se muestran los casos de pruebas en las tablas 11 y 12:

Tabla 11: Caso de prueba. Ruta independiente #1. (Fuente: Elaboración propia).

Caso de prueba: Ruta independiente #1	
Descripción: **El usuario no tiene evaluaciones en el sistema**	
Entrada	Un usuario en el sistema.
Resultados esperados	El sistema inicializa por defecto en cero la cantidad de puntos, el promedio y la cantidad de evaluaciones del usuario.
Condiciones	El usuario no ha realizado ninguna evaluación.

Tabla 12: Caso de prueba: Ruta independiente #2. (Fuente: Elaboración propia).

Caso de prueba: Ruta independiente **#2**	
Descripción: **El usuario tiene evaluaciones en el sistema**	
Entrada	Un usuario en el sistema.
Resultados esperados	El usuario ha realizado una evaluación y se le actualiza la cantidad de puntos, su promedio y la cantidad de evaluaciones.
Condiciones	El usuario haya realizado al menos una evaluación en el sistema.

Descripción de la ejecución de los casos de prueba

Para ejecutar cada caso de prueba se realizaron pruebas manuales ejecutándose el módulo en tiempo real:

- Caso de prueba # 1: Se comprobó que el usuario que no presenta evaluaciones, el sistema le crea inicializando por defecto en cero, la cantidad de evaluaciones, una cantidad de puntos y un promedio asociado al usuario de manera correcta, lo que evidencia que el caso de prueba se ejecutó favorablemente.
- Caso de prueba # 2: Se comprobó, que el usuario al realizar las evaluaciones, se le actualiza la cantidad de evaluaciones, la cantidad de puntos y el promedio de evaluaciones de cursos, siendo esta la respuesta esperada en este escenario.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica Ruta Básica, se concluye que los mismos fueron probados satisfactoriamente.

Pruebas funcionales:

Como pruebas para evaluar las funcionalidades externas del módulo, se utilizaron los Métodos de Caja Negra, los cuales se concentran en los requisitos funcionales del software. Estas permiten al ingeniero de software derivar un conjunto de condiciones de entrada que ejercitarán por completo todos los

requisitos funcionales de un programa. Son un enfoque complementario que tiene probabilidades de descubrir errores de clases diferentes de los que se descubrirán en los métodos de caja blanca (38).
Con el objetivo de aplicar los métodos de caja negra, es necesario apoyarse en el Diseño de Casos de Prueba (DCP) propuesto por la metodología de desarrollo de software seleccionada. Un DCP es un artefacto generado en la disciplina Análisis y Diseño y utilizado en la Implementación para llevar a cabo las pruebas funcionales. Tiene como objetivo comprobar el correcto funcionamiento del sistema, en estos se incluyen las entradas, resultados y condiciones con la que se ha de verificar, constituyendo la guía principal para el probador (44).
Para el diseño de estos casos de pruebas se tuvo en cuenta la técnica de **Partición equivalente**. Esta técnica divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Es por esto que por cada requisito se obtuvo la Descripción de las variables (DV) para luego dividir el campo de entrada en clases de datos válidos e inválidos. Una vez definida la DV, se procede a diseñar los escenarios de pruebas para cada caso de prueba.
A continuación, la tabla 13 presenta el DCP perteneciente al requisito Mostrar ranking de usuarios:

Tabla 13: DCP correspondiente al requisito funcional Mostrar ranking general. (Fuente: Elaboración propia).

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción de visualizar el ranking de usuarios.	Se listarán los cincos usuarios con mejor puntuación en el sistema y que tengan al menos una evaluación realizada.	Se muestra un listado con los 5 mejores usuarios del sistema. Se muestran los siguientes datos de los usuarios: - Nombre - Posición - Puntuación	Ranking de usuarios
EC 1.2 Opción Ver Todos	Se muestran los usuarios registrados en el sistema que tengan al menos un curso	Se muestra un listado con todos los usuarios que existen en el sistema. Se muestran los siguientes datos de los usuarios: - Nombre - Posición - Puntuación	Ranking de usuarios / Ver Todos.

Seguidamente se realizaron pruebas funcionales al módulo para comprobar que las funcionalidades descritas en el documento de requisitos del sistema se cumplen con la implementación realizada, utilizando los Diseños de Casos de Pruebas generados anteriormente. De igual forma, se aplicaron

pruebas de regresión para asegurar que fueran corregidas todas las No Conformidades (NC) detectadas antes de pasar a la próxima iteración.

En cada una de las iteraciones, se detectaron diferentes NC. Las NC medias y bajas, se centraron en errores ortográficos como: omisiones de tildes y cambio de mayúscula por minúscula y mensajes sin traducir al inglés. Las altas consistieron en errores en las funcionalidades.

Al realizarse la primera iteración se identificaron 7 NC. Para darle solución a las mismas se realizaron las pruebas de regresión antes de pasar a la próxima iteración. Una vez corregidas las NC, se procedió a hacer una segunda iteración y fueron encontradas 3 NC que también fueron corregidas luego de aplicarles pruebas de regresión a las mismas. Una vez concluidas estas iteraciones, quedan solucionadas todas las NC detectadas, obteniéndose un módulo libre de errores.

En la siguiente tabla 14, se muestran las NC clasificadas.

Tabla 14: Tabla de No Conformidades clasificadas. (Fuente: Elaboración propia).

Cantidad de NC/Posicionamiento	**Validación**	**Funcionalidad**	**Ortografía**	**Internacionalización**
	0	1	5	1

En la siguiente tabla, se muestran la cantidad de NC por casos de prueba por iteración.

A continuación, en la ilustración 15, se muestra un gráfico donde se puntualiza por iteraciones el total de No Conformidades

identificadas, las cuales fueron evaluadas en un rango (Alta, Media, Baja).

Ilustración 15. Cantidad de No Conformidades por iteración. (Fuente: Elaboración propia).

3.3.2. Pruebas de Aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Una vez realizadas las pruebas de aceptación, se generó el Acta de aceptación del producto como constancia de la conformidad por parte del cliente de la solución propuesta, dando de esta manera culminación al proceso de pruebas de software.

Conclusiones parciales

Los estándares de codificación definidos contribuyeron a obtener un estilo de programación homogéneo de manera tal que los participantes pudieran interpretar de manera eficiente la implementación de la propuesta de solución. Conjuntamente, la realización de las pruebas de software permitió evaluar la calidad del módulo con el objetivo de entregar al cliente un producto que satisfaga los requerimientos registrados inicialmente.

Conclusiones generales

- Los referentes teóricos demostraron la importancia que tiene contar con elementos de gamificación y las ventajas de los mismos.
- La realización del Análisis y Diseño al módulo guiado por la metodología AUP en su variante UCI permitió obtener una propuesta de solución más detallada sobre cómo incluir un ranking en la Plataforma Educativa XAUCE ZERA, la cual estuvo sustentada por 5 requisitos funcionales y 3 no funcionales.
- La implementación del módulo permitió que la Plataforma Educativa XAUCE ZERA contará con un espacio donde se logre visualizar el comportamiento de los usuarios en la plataforma.
- La ejecución de las pruebas de software al módulo implementado, permitió detectar a través de las pruebas funcionales un total de 7 No Conformidades, las cuales fueron resueltas en 2 iteraciones, arrojando resultados satisfactorios y un producto final libre de errores.

Recomendaciones

- Ampliar el módulo desarrollado empleando otras mecánicas de gamificación.
- Aplicar el método de average por decaimiento (*Decaying Average*) que utiliza la plataforma Canvas LMS al ranking.
- Aplicar al módulo otros criterios de evaluaciones.

Referencias Bibliográficas

1. **Bravo, Leonardo Emiro Contreras.** *Estrategias educativas para el uso de las TIC en la educación superior.* Colombia : s.n., 2013.

2. **Cervantes, María de los Ángeles Ahumada.** *El verdadero impacto de los LMS en el aprendizaje de los alumnos contra el uso de los Métodos tradicionales .* Mexico : s.n., 2019.

3. **Guillermo Mario Arturo Salazar Lugo, Armando Lozano Rodrígue.** *IDENTIFICACIÓN DE ESTILOS DE APRENDIZAJE EN PLATAFORMAS TECNOLOGICAS (LMS) MEDIANTE ARBOLES DE DESICION.* Mexico : s.n., 2019.

4. **Liuber, M. S., & Álvarez, G.** *CHARACTERIZATION OF THE EVALUATION OF INTERACTIVITY IN VIRTUAL ENVIRONMENTS OF TEACHING LEARNING AT THE UNIVERSITY OF HOLGUIN.* 2015.

5. **Carrión-Salinas, Gianella.** *Gamificación en educación primaria. Un estudio piloto desde la perspectiva de sus protagonistas.* Andalucia : s.n., 2017.

6. **[Internet], Universidad de las Ciencias Informáticas.** Universidad de las Ciencias Informáticas [Internet]. [En línea] 2018. http://www.uci.cu.

7. **[Internet], XAUCE ZERA: Plataforma Educativa.** [En línea] 2018. https://eva.uci.cu/es/aboutAs .

8. **Romero, Pablo, Torres, Frank y Aguaded, Marcos.** *Ludificación y educación para la ciudadanía.* 2017, Web learning, págs. 1-3.

9. **Miguel García Iruela, R. H.** *Análisis para la gamificación de un curso de Formación.* Madrid, España : s.n., 2017.

10. **Morales-Castany, M.** *Diseño de un ranking de estudiantes a partir de trofeos obtenidos en módulos profesionales.* Catalunya : Universidad Politecnica de Catalunya., 2017.

11. **A. ClUCCl, Leonardo .** *Gamificación: alcances y perspectivas en la ciudad de La Plata.* La Plata : s.n., 2016.

12. **Díaz Cruzado, Jesús.** *EL POTENCIAL DE LA GAMIFICACIÓN APLICADO AL AMBITO EDUCATIVO.* 2014.

13. **D, Macías Álvarez.** *Plataformas de enseñanza virtual libres y sus características de extensión: Desarrollo de un bloque para la gestión de tutorías en Moodle.* España : s.n., 2010.

14. **Moodle.** Moodle Community. *Moodle.* [En línea] Enero de 2019. https://moodle.org/plugins/block_ranking.

15. **Conde Vives JV, García Rodríguez J, García Luna D, Hermiz Ramírez A, Osorio Navarro A, Moreno López JJ.** *Manual Moodle 2.8 para profesores.* España : s.n., 2013.

16. **Schoology .** *Schoology Centro de Ayuda.* [En línea] Marzo de 2019. https://support.schoology.com/hc/es/articles/209970338-Cursos-Configuraci%C3%B3n-de-las-calificaciones.

17. **Canvas LMS.** Canvas Community. [En línea] 2019. http:\\Canvas.org.

18. **JA, Gallego Vázquez.** *Desarrollo web con php y mysql.* España : GRUPO ANAYA, S. A, 2003.

19. **Hernández Claro R, Greuguas Navarro D.** *Estándares de Diseño Web.* s.l. : Ciencias de la Información, 2010.

20. **J, Eguíluz Pérez.** Introducción a JavaScript . [En línea] 2009. www.librosweb.es.

21. **JM, Galindo Haro.** *Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE.* 2008.

22. **Symfony.org.** *CURSO BÁSICO DE SYMFONY 2.* 2012.

23. **Potencier F, Zaninotto F.** Symfony, la guía definitiva . [En línea] www.librosweb.es.

24. **C, Niska.** *Extending Bootstrap.* s.l. : UK: Packt Publishing Ltd, 2014.

25. **Ortiz Batista Y, López Reinoso Y, Medina León Y, Gonce Fernández S, Batard Lorenzo, D, Gulín González J.** *Propuesta de solución para la gestión de la información de la actividad de ciencia, tecnología e innovación en la Universidad de las Ciencias Informáticas.* 2010.

26. **DK, Ponce Briones.** *ANÁLISIS COMPARATIVO DE LOS ENTORNOS DE DESARROLLO INTEGRADOS (IDE): ECLIPSE, NETBEANS Y JDEVELOPER PARA EL DESARROLLO DE APLICACIONES JAVA ENTERPRISE EDITION.* Guayaquil, Ecuador : s.n., 2016.

27. **Cylwik López, L, Llanes Díaz A.** *Módulos para la simulación de modelos de propagación en la herramienta Andrómeda Revista Cubana de Ciencias Informáticas.* 2016.

28. **L, Hueso Ibáñez.** *Administración de sistemas gestores de bases de datos.* 2011.

29. **PostgreSQL database management system. [En línea] 2017. www.tutorialspoints.com .**

30**. N, Palma Pérez.** *Módulo para la administración de los servidores web en HMAST.* Cuba, UCI : s.n., 2013.

31. **S, Corona.** *Nginx: A Practical Guide to High Performance.* Estados Unidos : s.n., 2008.

32. **S, Chacon.** *Pro Git, el libro oficial de Git.* 2014.

33. **RS, Pressman.** *Ingeniería de software: Un enfoque práctico.* Nueva York, EUA : s.n., 2007**.**

34. **T, Rodríguez Sánchez.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana, Cuba : s.n., 2015.

35. **Canós JH, Letelier P, Penadés MC.** *Metodologías Ágiles en el Desarrollo de Software.* Valencia, España : s.n.

36. **C, Larman. UML y Patrones: Introducción al análisis y diseño orientado a objetos. Mexico : s.n., 1999.**

37. **Almeira AS, Perez Cavenago V.** *Arquitectura de Software: Estilos y Patrones.* Argentina: San Juan Bosco : s.n., 2007.

38. **uml, PARADIGM V.** Visual paradigm for. *Visual Paradigm for UML-UML tool for software application development.* 2013.

39. **CB, Reynoso.** *Introducción a la Arquitectura de Software.* Argentina: BuenosAires : s.n., 2004.

40. **J, Eguíluz Pérez.** *Desarrollo web ágil con Symfony2.* 2011.

41. **Guerrero CA, Suárez JM, Gutiérrez LE.** *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* Colombia : s.n., 2009.

42**. M, Blázquez Ochando.** *Modelo entidad-relación. Fundamentos y Diseño de Bases de Datos.* 2014.

43**. DJ, Salas Álvarez.** Estándares de codificacion Java . [En línea] 2016. http://www.aves.edu.co/ovaunicor/recursos/view/265..

44**. Sommervlle, lan.** ***INGENIERÍA DEL SOFTWARE.*** *Séptima edición.* Madrid : PEARSON EDUCACIÓN, S.A., 2005. ISBN: 84-7829-074-5.