Facultad 5

Centro de Entornos Interactivos 3D (VERTEX)

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

Título:

Generación de productos de trabajo mediante Sistema de Apoyo en la disciplina de Requisitos.

Autor (a): Eylin Campillo Santos

Tutor: Ing. Andy Hernández Paez

Co-Tutor (res): Ing. Alberto Enrique Ruiz Romero

Ing. Yanet Liliana Garbey Gainza

"Año 58 de la Revolución"
La Habana, Cuba
Junio 2016

"La función de la educación es enseñar a pensar intensamente y pensar críticamente. Inteligencia más carácter, ese es el objetivo de la verdadera educación."

Martin Luther King

Declaro por este medio que yo Eylin Campillo Santos, con carné de identidad 93052908017, soy el autor principal de este trabajo “Sistema de apoyo para la generación de productos de trabajo en la disciplina de Requisitos” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____	_____
Eylin Campillo Santos	Ing. Andy Hernández Páez
Autora	**Tutor**
_____	_____
Ing. Alberto Enrique Ruiz Romero	Ing. Yanet Liliana Garbey Gainza
Co-Tutor	**Co-Tutora**

Tutor: Ing. Andy Hernández Paez.

Edad: 27.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero en Ciencias Informáticas.

Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en junio del 2012, en la Universidad de Ciencias Informáticas. Profesor Instructor de la disciplina Ingeniería y Gestión de Software. Actualmente profesor del centro VERTEX Entornos Interactivos 3D y Asesor de Calidad en Software. Posee 5 años de experiencia como Analista de sistema en proyectos de software de Informática Industrial y Realidad Virtual. Especialista en el marco de trabajo de la Ingeniería de Requisitos y Arquitectura de la Información. Presenta varias publicaciones y participación en eventos.

E-mail: andyhp@uci.cu

Tutor: Ing. Alberto Enrique Ruiz Romero.

Edad: 31.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero en Ciencias Informáticas.

Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en junio del 2008, en la Universidad de Ciencias Informáticas. Profesor Asistente, Jefe de colectivo de asignatura y Jefe de la disciplina de Programación. Actualmente es Jefe del Departamento de Ingeniería y Gestión de Software.

Posee 8 años de experiencia como Desarrollador en diferentes centros de la UCI y del MININT.

Presenta varias publicaciones y participación en eventos.

E-mail: aeruiz@uci.cu

Tutor: Ing. Yanet Liliana Garbey Gainza.

Edad: 24.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero en Ciencias Informáticas.

Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en junio del 2015, en la Universidad de Ciencias Informáticas. Recién graduada en adiestramiento prestando servicios como profesora de Idioma Extranjero. Se encuentra en primer año de experiencia como Analista de sistema en proyectos informáticos de Realidad Virtual. Se desempeña como Administradora de la Configuración del Centro Vertex y forma parte del grupo de Ingeniería y Calidad de Software de dicho centro. Presenta publicaciones en eventos científicos.

E-mail: liliana@uci.cu

A mis padres por ser mi ejemplo a seguir.

A mis familiares y amigos.

A Alberto, por ser una de las cosas más importantes en mi vida.

A mis dos tesoros, mis padres, por ser ejemplo a seguir y motivo de impulso para mi vida profesional, por seguir adelante a pesar de todas las dificultades que se presentan en el camino.

A mis tíos Charo y Tony que están siempre pendientes de mis notas, mis problemas y dándome el impulso para salir adelante en esto cinco años de carrera.

A Alberto por ser pareja, amigo, profesor, por apoyarme y ayudarme en todo momento.

A mis abuelos, en especial a mi abuelo Tito que hace menos de un año no está, pero me hubiera gustado mucho que me viera graduarme.

A mis familiares por estar siempre a mi lado y ayudarme a cumplir mis sueños.

A mis tutores Andy y Yanet por el apoyo y ayuda que me han dado para salir adelante en este trabajo.

A mis amigos Mayra y Vega por compartir momentos lindos y ayudarme en mi carrera.

A mis amigos Yei, Dianne, Anelis, David y Ernesto por estar siempre soportándonos y ayudándonos cuando lo hemos necesitado durante estos cinco años de la carrera.

Resumen

En la Universidad de las Ciencias Informáticas específicamente en el Centro de Entornos Interactivos 3D (VERTEX) surge la necesidad de crear una herramienta capaz de administrar los productos de trabajo de la disciplina de ingeniería de requisitos (IR) para cada una de las líneas de desarrollo del centro: juegos serios (videojuegos), entrenadores 3D, paseos virtuales y simulación computacional 3D. Para la gestión de estos productos de trabajo se tiene como base el Expediente de Proyectos de Desarrollo definido a nivel institucional. Para esto fue necesario el estudio y migración del Expediente de Proyectos de Desarrollo (EPD) 3.4 a la versión 4.0 del Proceso de Desarrollo de Software que se utiliza actualmente a nivel de universidad. A través del estudio del estado del arte se realiza un análisis de las herramientas con estos fines, específicamente en la disciplina de requisitos y el EPD, con el propósito de identificar sus mejoras y un conjunto de productos de trabajo a generar. Además, se realiza un análisis conciso de las tecnologías y lenguajes para el desarrollo de la solución informática, así como las nuevas tendencias. Las pruebas al software representan un punto crucial para determinar que el mismo cumple con el nivel de expectativas del cliente y se encuentra libre de defectos. En esta disciplina del proceso de desarrollo de la solución se aplicaron varios tipos de pruebas, tales como, de funcionalidad, rendimiento y aceptación. Dichos tipos de pruebas permitieron validar que el sistema está listo para ser usado por los usuarios finales y que cumple con las normas, estándares y parámetros de calidad lo mejor posible.

Palabras clave: Expediente de Proyectos de Desarrollo 4.0, ingeniería de requisitos, líneas de desarrollo, VERTEX.

Índice de contenidos

Índice de tablas

Índice de ilustraciones

Introducción

La informática tiene como propósito convertirse en una de las ramas más productivas para Cuba. Para lograr esta meta es trascendental el lugar que ocupa la Universidad de las Ciencias Informáticas (UCI). La UCI, además de su perfil académico, encaminado a preparar profesionales revolucionarios en el campo de la informática, tiene como segundo perfil la investigación y producción de Software donde los estudiantes están vinculados a proyectos reales desde tercer año de la carrera. Por la magnitud que han alcanzado estos proyectos y el peso que tienen dentro de la economía del país es preciso contar con procesos bien definidos que ayuden a lograr una alta calidad en los productos desarrollados. Uno de estos procesos es la gestión de requisitos que debe contar con una especificación clara y completa desde las fases iniciales para no tener problemas posteriores que implicarían un retraso en el cronograma, un presupuesto erróneo, o hasta la posible cancelación del proyecto.

Estudios actuales realizados por el Instituto de Administración de Proyectos (conocido por sus siglas en inglés como PMI) sobre el éxito del desarrollo de software reflejan que esta es una de las disciplinas que más incidencia tiene en el fracaso de los proyectos, una muestra de estos factores se puede ver en el libro de Gestión de Requisitos, “Una competencia esencial para el éxito de proyectos y programas” (1).

Para facilitar este proceso en Cuba y el mundo, el uso de herramientas para auxiliar la gestión de requisitos se ha convertido en un aspecto importante en la disciplina de requisitos. Por tal motivo, estas herramientas usadas por los administradores para automatizar los procesos de la disciplina, han disminuido el trabajo duro en el mantenimiento de requisitos y por consiguiente han traído beneficios significativos en la reducción de errores. En la práctica, para gestionar los requisitos el personal del equipo de desarrollo confía en muchas ocasiones en herramientas ofimáticas tradicionales como Word, Excel y Access.

La UCI dedicada al desarrollo de software no está exenta de ellos y consta con herramientas que sustentan la disciplina de requisitos. En función de la línea de desarrollo y el EPD 4.0 se hace necesario tener presente las nuevas especificaciones

definidas por el programa de mejora de la institución y por ende desarrollar herramientas que le den soporte. Actualmente esta universidad está compuesta por varios centros que trabajan con este EPD y lo ajustan a sus necesidades.

El Centro de Entornos Interactivos 3D (VERTEX) tiene como principal misión desarrollar servicios y productos de software en 3D partiendo de un proceso investigativo mediante el uso de la creatividad y novedad científica, por lo que sus líneas de desarrollo están guiadas por I+D+i (investigación + desarrollo + innovación). Las líneas de desarrollo fundamentales de VERTEX son: juegos serios (videojuegos), entrenadores 3D, paseos virtuales y simulación computacional 3D.

En este centro para generar los productos de trabajo del Proceso de Desarrollo de Software (PDS) es necesario tener definido de forma correcta el escenario a aplicar en el producto de software que se desarrolla, según propone la metodología de PDS AUP definida a nivel de universidad para la disciplina de requisitos. Existen 4 escenarios bases para dicha disciplina, los cuales son: Casos de Uso del Negocio (CUN) + Modelo Conceptual (MC) = Casos Uso del Sistema (CUS), Modelo Conceptual (MC) = Casos de Uso del Sistema (CUS), Descripción de Procesos de Negocio (DPN) + Modelo Conceptual (MC) = Descripción de Requisitos por Procesos (DRP) e Historias de Usuarios (HU).

En el centro existe una aplicación web de apoyo a la reutilización de requisitos en el desarrollo de laboratorios virtuales con fines educativos. Esta aplicación sólo permite generar los productos de trabajo asociados al marco de trabajo común de la IR. Estos productos de trabajo se encuentran en la versión 3.4 del expediente de proyectos de desarrollo, lo que no resulta factible debido a que la versión que se emplea en la actualidad es la 4.0. Por otra parte, la aplicación está enfocada principalmente a una línea de productos en específico: entrenadores virtuales.

Los escenarios más apropiados a utilizar en los proyectos del centro VERTEX son MC=CUS para sus líneas de desarrollo: entrenadores virtuales, paseos virtuales, simulación computacional y para videojuegos se realizó una adecuación del cuarto

escenario que es HU que por las características de los videojuegos se definen otros productos de trabajo.

Algunos productos de trabajo que se hacen imprescindibles generar en la disciplina de requisitos y que responden a los escenarios más apropiados a utilizar y características de los proyectos del centro VERTEX son: criterios para validar requisitos del cliente y del producto, registro de proveedores de requisitos, evaluación de requisitos y casos de uso del sistema, especificación de requisitos de software y casos de uso del sistema, requisitos rechazados, reporte de trazabilidad, modelo conceptual, modelo de diseño, glosario de términos, salidas del sistema, diseño del videojuego y especificación de mecanismos. Donde los nueve primeros productos de trabajo son gestionados por el sistema web que implementa el marco de trabajo común de la IR para el desarrollo de laboratorios virtuales con fines educativos, sin embargo:

- Existen productos de trabajo que no cumplen con las normas y estándares establecidos en la versión 4.0 del EPD por el programa de mejora de la universidad, lo que provoca incumplimiento de las políticas y prácticas definidas a nivel institucional.
- No se documentan adecuadamente los productos de trabajo necesarios para cumplir con los escenarios dos y la adecuación del cuatro para videojuegos de la disciplina de requisitos aplicables a proyectos de software del centro VERTEX.
- En ocasiones la disponibilidad, centralización y uniformidad de los productos de trabajo entorpecen el PDS para realizar su gestión.
- No existe una correlación o identidad en las especificaciones realizadas durante la disciplina de requisitos y el resto de las disciplinas del PDS (análisis, diseño, pruebas y entre otras).

Una vez presentada la situación problemática se llegó al siguiente **problema de la investigación**: ¿Cómo contribuir a la generación de los productos de trabajo en proyectos de software del centro VERTEX aplicables a los escenarios de la disciplina de requisitos?

Obteniendo como **objeto de estudio**: Disciplina de requisitos en el proceso de desarrollo de software. Según el objeto de estudio anteriormente expuesto se especifica el **campo de acción**: Gestión de la información en la disciplina de requisitos para escenarios de desarrollo en Entornos Interactivos 3D.

En vista a la solución de esta interrogante se plantea el **objetivo general**: Desarrollar una aplicación web de apoyo a la generación de productos de trabajo del expediente de proyectos de desarrollo 4.0 en la disciplina de requisitos.

Para dar cumplimiento al objetivo general se trazaron las siguientes **tareas de investigación**:

1. Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema.
2. Análisis de los productos de trabajo de los expedientes de proyectos de desarrollo 3.4 y 4.0 para el área de proceso de Administración de Requisitos (REQM) del nivel 2 de CMMI para seleccionar los artefactos a generar.
3. Análisis de los escenarios de la disciplina de Requisitos aplicables a los proyectos de software del centro VERTEX según metodología de desarrollo de software AUP en su versión UCI.
4. Selección de la metodología, herramientas y tecnologías para el desarrollo del sistema como propuesta de solución.
5. Generación de los artefactos que corresponden a los flujos de trabajo definidos por la metodología seleccionada como pruebas del trabajo realizado.
6. Implementación de un sistema que facilite la generación de los productos de trabajo en escenarios de desarrollo de proyectos del centro VERTEX.
7. Validación del sistema mediante pruebas de funcionalidad, rendimiento y aceptación, para mitigar errores en la propuesta de solución.

Para el esclarecimiento de la propuesta se manejaron métodos de investigación científica, entre ellos se encontraron:

Métodos teóricos:

- **Histórico-Lógico:** Se reflejó en el estudio de los sistemas de gestión de requisitos y proyectos ya existentes en el mercado y el análisis del nuevo expediente de proyecto de desarrollo 4.0 de la administración de requisitos y en particular cada uno de sus productos de trabajo.
- **Analítico-Sintético:** Se utilizó para entender y aprender los conceptos asociados a la disciplina de administración de requisitos y permitió de manera simplificada el análisis a documentos, productos de trabajo y escenarios de la disciplina que tributen a la confección de un sistema eficiente y actualizado.
- **Modelación:** Se empleó para representar los diagramas de cada uno de los flujos de trabajo y de esta forma explicar de una manera más clara la realidad.

Métodos empíricos:

- **Consulta de la información en todo tipo de fuente:** Lo cual permite la elaboración del marco teórico de la Investigación y su correcta investigación metodológica.
- **Observación:** Facilitó observar las características, el funcionamiento y la evolución de las herramientas de administración de requisitos y el uso del expediente de proyecto 4.0 por parte de los especialistas.
- **Entrevista:** Logró detectar las incongruencias y deficiencias a las herramientas seleccionadas en dicho estudio.
- **Encuesta:** Apoyó al entendimiento de la situación actual de dicho tema y a la recopilación de ideas y criterios de especialistas para la confección de nuevos requerimientos.

Estructura:

Capítulo 1. Fundamentación teórica: Contendrá el estudio del estado del arte de las herramientas de administración de requisitos, abarcando la esfera internacional y de la

universidad. Se tendrán en cuenta las tecnologías, las metodologías, las herramientas de software y el lenguaje de programación que se propone para dar solución al problema planteado, junto con la justificación de su uso, buscando obtener el cumplimiento de los objetivos propuestos.

Capítulo 2. Características y diseño del sistema: Se definen los actores, trabajadores, casos de uso de la herramienta y se realiza la descripción del modelo de dominio. También se determinan los requisitos funcionales y no funcionales. Se crean los productos de trabajo concernientes a la disciplina de requisitos, los diagramas de clases del diseño, los diagramas de secuencia, la descripción de las clases y de la arquitectura a utilizar en la aplicación.

Capítulo 3. Implementación y prueba del sistema: Se basa en la implementación de la aplicación. Aquí es donde se representa el modelo de implementación, el diagrama de componentes y de despliegue. Además, se realizan las pruebas pertinentes para validar y verificar el estado y la calidad del sistema.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

La creación de una herramienta para la gestión de productos de trabajo del EPD 4.0 y el desarrollo de videojuegos tiene vital importancia hoy día en centros de desarrollo como lo es el Centro de Entornos Interactivos 3D VERTEX de la UCI. Por tal motivo se hace necesario abordar principios y conceptos que sirven para el entendimiento de toda esta investigación y para el desarrollo de un sistema informático. Durante este proceso se realiza un estudio a los escenarios para modelar el sistema en los proyectos, obteniendo como escenarios a emplear el dos (Modelo Conceptual = Caso de Uso del Sistema) y una adecuación al cuatro (Historia de Usuario) solamente para la línea de videojuegos que a su vez propone nuevos productos de trabajo explicados más adelante. Por otro lado, la nueva versión del EPD posee cambios en algunos de los productos a generar, ellos son:

Producto de trabajo	Transformación
Criterios para validar requisitos del cliente	Eliminar la columna: "¿El resultado de la evaluación de impacto es positivo?".
Evaluación de requisitos	Varía el rango de selección de los parámetros: "Alta, Media, Baja".
Especificación de requisitos de software	Varía la taxonomía de clasificación de los requisitos no funcionales. La descripción de los requisitos no funcionales se realiza a partir de una tabla.

Tabla 1: Transformación en los productos de trabajo de la v4.0 del EPD.

Además, se comparan las principales herramientas empleadas para la gestión de requisitos con el objetivo de resumir las funcionalidades y características que presentan. Por último, se hace un bosquejo de las posibles herramientas, tecnologías, metodologías y lenguajes a utilizar para la confección de la propuesta de solución.

1.2 Términos relacionados con la investigación

Para el entendimiento y desarrollo de la propuesta de solución se hace necesario conocer un grupo de conceptos que van a formar parte del proceso de la disciplina de requisitos. Como primer concepto y punto de partida se tiene que “La IR ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio…” (2).

Definiendo así a la IR como la comunicación entre las necesidades del cliente y del equipo de trabajo que desarrolla la aplicación informática. Donde estos últimos se encargarán de manera iterativa del diseño y modelado durante el ciclo de vida del sistema, en este caso la herramienta para la gestión de requisitos.

Todas estas tareas o procesos van a estar guiadas por un marco de trabajo que establece la base para un proceso de software completo, identificando un número de tareas del marco teórico aplicables a todos los procesos de software, sin importar su tamaño y complejidad. Según Pressman, Sommerville, Wiegers y la IEEE existen varias definiciones para referirse a un marco de trabajo como parte de la ingeniería de requisitos y como elemento base de todo proceso de desarrollo de software. Este se define como un elemento teórico-práctico que se usa para encaminar el trabajo enfocado a la calidad del producto y erradicar las pérdidas tanto de tiempo como de recursos.

Sommerville	Wiegers	IEEE	Pressman
1. Estudio de Viabilidad	1. Elicitación	1. Elicitación	1. Inicio
2. Obtención	2. Análisis	2. Análisis	2. Obtención
3. Análisis	3. Especificación	3. Especificación	3. Elaboración
4. Especificación	4. Validación	4. Validación	4. Negociación
5. Validación	5. Gestión de requisitos	5. Gestión de requisitos	5. Especificación
			6. Validación
			7. Gestión de requisitos

Tabla 2: Listado de procesos del Marco de Trabajo según autores.

Luego de un análisis de las definiciones y procesos por los cuales se rige el proceso de desarrollo de software por dichos autores, se definió para la herramienta un marco de trabajo que cuenta con cinco procesos, ellos son:

- **Obtención de requisitos:** Proceso en el cual se analiza con el cliente para identificar el problema que el sistema debe resolver, los diferentes servicios que debe prestar y las restricciones que se pueden presentar.
- **Análisis de requisitos:** Proceso en el cual se realiza un estudio del dominio de aplicación. Se analizan los requisitos extraídos en busca de irregularidades, se proponen soluciones a los problemas encontrados, se analizan las necesidades del cliente, sus limitaciones y expectativas, se estiman costos contra beneficios que brindará el software, así como la factibilidad y soporte del mismo.
- **Especificación de requisitos:** Proceso en el cual el analista del sistema debe llevar un documento formal con una descripción completa del comportamiento del sistema a desarrollar.
- **Validación de requisitos:** Proceso en el cual se verifica que los requisitos especificados representen una descripción clara del sistema. Es la validación final de que los requisitos cubren las necesidades de los usuarios.
- **Administración de requisitos:** Proceso en el cual se debe comprender y controlar los cambios en los requisitos. Es la administración de todos los requisitos recibidos o generados por el proyecto, incluyendo tanto requisitos técnicos como no técnicos, así como aquellos requisitos impuestos en el proyecto por la organización.

Estos cinco procesos van a estar enfocados al desarrollo de los productos de trabajo que se encuentran en el EPD definido por la universidad. El expediente de proyecto es definido como una “Colección de proyectos o programas y otros trabajos que se agrupan juntos para facilitar la gestión efectiva de ese trabajo para alcanzar los objetivos estratégicos del negocio. Los componentes de este portafolio son cuantificables; eso es, que pueden medirse, clasificarse y priorizarse. Los proyectos o programas pueden no ser interdependientes o estar directamente relacionados (3).

En nuestra universidad el expediente de proyecto es un factor esencial para el desarrollo de los distintos proyectos, pues este contiene los productos de trabajo generados en el proceso de desarrollo de software. Este expediente de proyecto se divide en 4 grupos en los que se encuentran distribuidos disímiles productos de trabajo, ellos son: ingeniería, gestión de proyecto, soporte y líneas bases. En el caso del centro Entornos Interactivos 3D se les incluyen productos de trabajo para la línea de desarrollo de videojuegos. Actualmente los proyectos desarrollados en la UCI están guiados bajo la versión 4.0 de este expediente.

Una vez identificada el área del conocimiento, el marco de trabajo y EPD definido por la institución se da paso a los acciones o métodos para la gestión de la información, el cual se define como un proceso que incluye acceso, extracción, manipulación, depuración, conservación, análisis y entrega de la misma a través de diferentes fuentes o simplemente para la toma de decisiones con el objetivo de garantizar su integridad, disponibilidad y confidencialidad de dicha información.

1.3 Herramientas de gestión de requisitos

En la actualidad existen diversas herramientas de gestión de requisitos que son la base de dicho proceso con la finalidad de mejorar la productividad y calidad en el desarrollo de los proyectos de software. Una necesidad a cubrir por esta herramienta es la disponibilidad de la información de los distintos proyectos en un repositorio al cual se pueda acceder por los distintos usuarios con el fin de ser incorporada, consultada o reutilizada.

Para la propuesta de solución se realizó un estudio de las herramientas internacionales y nacionales de gestión de requisitos que se verán a continuación:

1.3.1 OSRMT (Open Source Requeriments Management Tool)

Es una herramienta diseñada para dar cobertura a todo el ciclo de vida de desarrollo del software. Dispone de control de versiones, permite definir requerimientos derivados, es posible definir tanto casos de uso como casos de prueba y brinda la posibilidad de definir atributos para los requisitos como son el riesgo y esfuerzo entre otros.

1.3.2 REM (Requisite Management)

Aunque es una herramienta de uso libre puede ser utilizada únicamente sobre Windows, ha sido utilizada con frecuencia para fines educacionales. REM permite generar un documento normalizado en el que se pueden incluir los requisitos necesarios para el desarrollo de sistemas de información. La documentación es generada en formato HTML.

1.3.3 Rational Requisite Pro

Es una herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos (aunque también permite guardarlos directamente en la base de datos), mientras que las otras herramientas están orientadas a requisitos. Auxilia especialmente en el control de cambio de requisitos, con trazabilidad para especificaciones de software y pruebas. Está muy unido a MS Word ya que es partner de Microsoft Development. La herramienta permite el uso de Oracle sobre Unix o Windows como "*back-end database*" y también soporta SQL Server sobre Windows.

1.3.4 CaliberRM

Es para sistemas grandes y complejos y proporciona una base de datos de requisitos con trazabilidad. La compañía ve a los requisitos como parte del proceso de gestión de la calidad del software, el cual es considerado también, las pruebas (*testing*) y el trazado de defectos (*defect tracking*). Caliber está basado en Internet y maneja referencia de documentos, responsabilidad de usuario, trazabilidad, prioridad y estado entre otras características.

1.3.5 IRqA (Integral Requisite Analizer)

Es una de las herramientas de gestión de requisitos más completas del mercado. Los requisitos que se capturan se almacenan en documentos Word y las descripciones de los mismos pueden referenciar a documentos externos como son tablas, gráficos y hojas de cálculo de Microsoft Excel. Permite establecer relaciones entre requisitos, además se puede integrar con Rational Rose.

1.3.6 Telelogic Doors

Es un sistema multiplataforma diseñado para la gestión de requisitos mediante la captura, trazabilidad, enlazado, análisis y manejo de los cambios que en ellos se realicen. Mediante el uso de Doors se puede realizar un análisis de trazabilidad para identificar las áreas de riesgo, y resulta fácil manejar los cambios que tengan lugar en los requisitos. Además, permite gestionar un gran número de requisitos de forma eficiente mediante el uso de una base de datos sencilla, lo que se conoce como característica de gran escalabilidad.

1.3.7 VirtualLabRDM

Herramienta diseñada para la gestión de requisitos de laboratorios virtuales que se encarga de la captura, trazabilidad, análisis y modificaciones de los mismos. Se pueden establecer dependencias, especificaciones de requisitos y de casos de uso. Implementada en lenguaje JAVA y su licencia es libre. Genera reportes en formato PDF. No tiene definido mediante un algoritmo un mecanismo que asigne permisos de las funcionalidades a cada tipo de usuario. Los productos de trabajo y documentación que se generan están sujeto a la versión 3.4 del EPD.

1.3.8 GR_LP_SCADA

Herramienta específicamente para líneas de producto de software SCADA. La herramienta en cuestión está desarrollada como una aplicación de escritorio implementada en Java como lenguaje de programación, con acceso por niveles usando roles, garantizando la integridad de la información utilizada y generada. En su versión 1.0 permitía la gestión de requisitos pero no incluía en su programación la trazabilidad bidireccional con productos de trabajo que forman parte indisoluble del PDS como lo son los casos de uso, diagramas de clases, casos de pruebas, los que garantizarían un mayor alcance de la aplicación en el control de la gestión y el control de los cambios, mientras que en la versión 2.0 se mejoraría este aspecto y además se podrían generar reportes de los cambios realizados.

1.3.9 Análisis de las herramientas

A partir del estudio realizado de estas herramientas se determinó que ninguna comprendía en su totalidad las características descritas a continuación:

- Tipo de licencia
- Usabilidad (navegación, notificación de acciones y sencillez)
- Control de acceso
- Plataforma
- Facilidad para importar/exportar requisitos

Por ello, se puede decir que a pesar de haber aportado una serie de ideas del funcionamiento de las mismas, de cómo gestionar los distintos conceptos dentro de la disciplina de requisitos y dar una panorámica de las posibles funcionalidades que tendría la propuesta de solución se determinó que no cumplen con los elementos que contiene la versión 4.0 del EPD y no existe una correlación en las especificaciones realizadas durante la disciplina de requisitos y el resto de las disciplinas del PDS (análisis, diseño, prueba). Además, se hace necesario incorporar a este proceso de gestión de requisitos una línea de videojuegos que va a contemplar procesos de diseño y productos de trabajo específicos que los antes mencionados no poseen.

1.4 Productos de trabajo de Requisitos del Expediente de Proyectos de Desarrollo UCI

El EPD UCI en su versión 4.0 es un factor esencial que influye en la propuesta de solución, pues este contiene los productos de trabajo generados durante la gestión de requisitos en el PDS. Este expediente se divide en 4 grupos en los que se encuentran distribuidos disímiles productos de trabajo, ellos son: ingeniería, gestión de proyecto, soporte y líneas bases.

Para la propuesta de solución se estarán realizando 12 productos de trabajo, específicamente del grupo de ingeniería concerniente a requisitos, casos de uso y trazabilidad mientras que también se trabajarán con 2 productos de trabajo de una variante de este EPD enfocada a la línea de videojuegos.

Dichos productos de trabajo serán los generados de manera semiautomática por el sistema a desarrollar. Estos productos de trabajo o productos son:

- **Criterios para validar requisitos del producto:** Se evalúan los requisitos seleccionados con el fin de aceptarlos o rechazarlos. En caso de que sean aceptados se describen los riesgos, fases y tipos de selección.
- **Criterios para validar requisitos del cliente:** Se evalúan los requisitos seleccionados con el fin de aceptarlos o rechazarlos. En caso de que sean aceptados se describen los riesgos.
- **Especificación de Requisitos de Software:** Se describen las características del sistema y los actores que van a estar interactuando con el mismo. De manera general se caracterizan los requisitos funcionales y no funcionales.
- **Especificación de Casos de Uso del Sistema:** En caso de existir división de paquetes se analizan y describen de manera independiente. Se describen los casos de uso especificados, el modelo conceptual, así como los formatos de entrada y salida del sistema.
- **Evaluación de Requisitos:** De manera resumida se evalúan los requisitos por independiente en cuanto a una serie de aspectos como interfaces, comportamientos, formas de inicialización, consultas a fuentes de datos y otros para luego llegar a establecerles a cada uno una complejidad.
- **Evaluación de Casos de Uso:** Se analizan los casos de uso de manera independiente con una serie de aspectos como transacciones, entidades candidatas, reutilización, criticidad, dependencia, estabilidad para poder analizar de manera satisfactoria la complejidad y prioridad de cada uno.
- **Modelo Conceptual:** Se analizará el modelo conceptual y en caso de llevarlo se desglosarán en paquetes. Además, se describirán las entidades, las propiedades de esta y de sus atributos y el estado de validez de las mismas.
- **Glosario de Términos:** Se especificarán todos los conceptos y términos asociados al producto y a su gestión.
- **Salida del Sistema:** Se describen de manera general todas las propiedades que tendrá el formato del reporte, las tablas y los gráficos.

- **Requisitos Rechazados:** Una vez detectados los requisitos rechazados se procede a describir estos requisitos por independiente analizando los criterios de fallos, las causas y las propuestas.
- **Registro de Proveedores:** Se verifican y evalúan los proveedores con una serie de requisitos y de datos personales para clasificarlos en aptos o no aptos. En caso de ser aceptados como aptos se analiza la selección y el horario de disponibilidad.
- **Reporte de Trazabilidad:** Se analizan aspectos relacionados con la trazabilidad como los datos de los reportes, elementos y el análisis de impacto en el tiempo de desarrollo que incluye costos, recursos y calidad del producto.
- **Especificación de Mecanismos:** Se generan los mecanismos y el diagrama que los engloba, así como los requisitos no funcionales del producto y sus clasificaciones.
- **Modelo de diseño:** Se describe el diagrama de paquete del proyecto, así como las distintas clases que van a interactuar en el sistema, sus diagramas, su estructura y el propósito de las mismas. Contará con un acápite en caso de alguna observación adicional sobre las clases.

- **Diseño de Videojuegos:** De manera general se resume el juego, el diseño del mismo, las metas para el jugador, la estructura, los jugadores, los objetivos, procedimientos o reglas y demás. También cuenta con una breve descripción de las pantallas gráficas elementales.

1.5 Tendencias y tecnologías

Para lograr un producto con el máximo de calidad posible es necesario el uso de metodologías, tecnologías y herramientas para el análisis, diseño, implementación y prueba del mismo, que permitan mantener una línea de trabajo óptima para lograr mayor confiabilidad, integridad y fortaleza para el software.

1.5.1 Metodología de desarrollo

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito (4).

Existen dos clasificaciones: robustas y ágiles. Las robustas son necesarias y eficaces en proyectos mayores por estar orientadas al control de procesos, listado de tareas a realizar y herramientas a emplear, generando así una extensa planificación y documentación del producto. Las ágiles son recomendadas para proyectos más chicos porque tienen como objetivo la interrelación con el cliente. En períodos de tiempo corto se les va enseñando a los clientes los avances de las versiones o de funcionalidades, para que de esta forma pueda evaluar y sugerir cambios al software.

En la actualidad se usan disimiles metodologías que pertenecen a cada uno de estos dos grupos y se ha comprobado que muy pocos proyectos las aplican en su totalidad. Las diferencias entre estas metodologías no radican en los productos de trabajo que proponen o en sus roles definidos, sino en su forma de planificar y administrar las estimaciones del tiempo.

Según un estudio realizado de disímiles metodologías se ha decidido usar la metodología AUP - Agile Unified Process o Proceso Unificado Ágil. AUP es la metodología que se ajusta a la necesidad del proyecto porque combina características de la metodología ágil *eXtremme Programming* (XP)[1] con los artefactos de *Rational Unified Process* (RUP)[2]. Dentro de las características particulares de AUP, se tiene que es una versión simplificada de la metodología RUP.

AUP - Agile Unified Process

Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo dirigido por pruebas (*test driven development - TDD*).

[1] **XP** (en español, Programación Extrema) Metodología de desarrollo de software ágil formulada por Kent Beck.

[2] **RUP** (en español, Proceso Unificado de Rational) Metodología de desarrollo de software robusta desarrollado por la empresa de Rational Software, actualmente propiedad de IBM.

- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

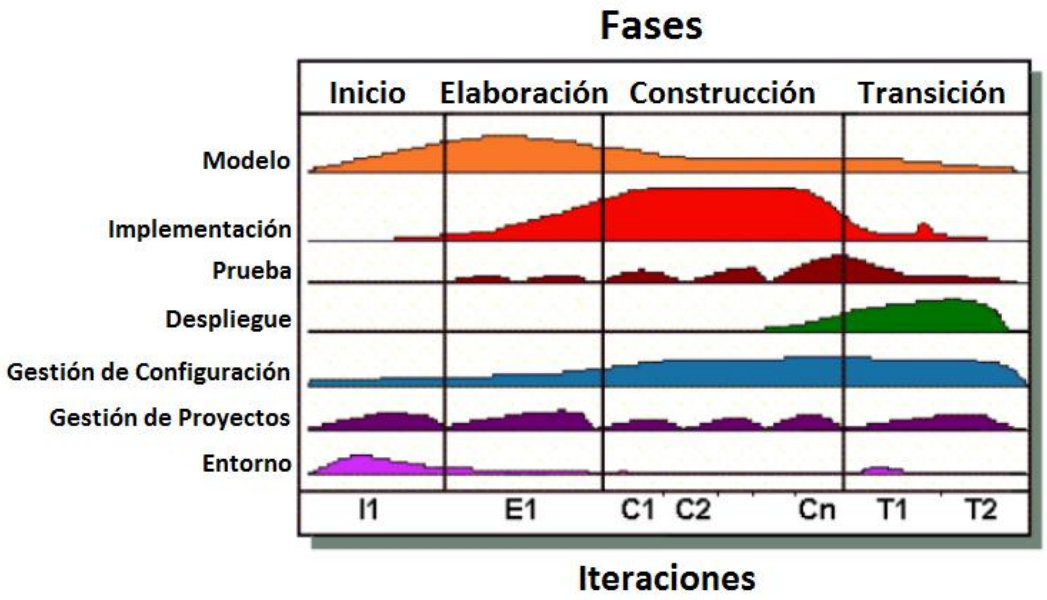


Ilustración 1: Esquema de las disciplinas y fases de trabajo de AUP.

Esta metodología posee cuatro fases (inicio, elaboración, construcción y transición) las cuales tienen el objetivo de establecer un equilibrio entre las necesidades del cliente y la opinión del equipo de desarrollo, validar arquitecturas, diseños y pruebas en el desarrollo y por último una serie de pruebas de validación, aceptación y el tan esperado despliegue. Además, cuenta con siete disciplinas (4 ingenieriles y 3 de gestión de proyectos):

1. Modelo: Entiende el negocio de la organización, el problema de dominio que se aborda en el proyecto y determina una solución viable para resolver el problema de dominio.
2. Implementación: Transforma su modelo en código ejecutable y realizar un nivel básico de las pruebas.
3. Prueba: Realiza una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.

4. Despliegue: Ejecución del sistema y que el mismo este a disposición de los usuarios finales.
5. Gestión de configuración: Gestiona el acceso a herramientas de su proyecto. Esto incluye seguimiento de versiones, y control y gestión de cambios.
6. Gestión de proyectos: Dirige las actividades que se lleva a cabo en el proyecto. (gestión de riesgos, gestión de recursos humanos, gestión de costo y tiempo).
7. Entorno: Garantiza que el proceso sea el adecuado y que la orientación (normas y directrices) y herramientas (hardware, software) estén disponibles para el equipo de desarrollo según sus necesidades.

Esta versión ágil de RUP es una metodología descriptiva que permite realizar todo el proceso con sus artefactos pertinentes sin llegar a realizar todos los diagramas a las funcionalidades, sólo las de mayor importancia respecto al negocio y que a su vez dificulten el proceso de creación de otras. Estas funcionalidades de mayor envergadura son gestionar proyecto, gestionar requisito funcional, gestionar requisito no funcional, gestionar mecanismo y gestionar productos de trabajo.

En resumen, AUP se basa en los siguientes principios:

- **Simplicidad:** Refiere a que todo se describe de forma concisa usando poca documentación, no muchas de ellas.
- **Agilidad:** Refiere al ajuste de los valores y principios de la Alianza Ágil. Es por ello que es necesario centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- **Herramienta de la independencia:** Refiere a que se puede usar cualquier conjunto de herramientas que desea con el AUP. Sugiere utilizar las herramientas idóneas para el trabajo, que normalmente son herramientas simples o incluso herramientas de código abierto.
- **Usted querrá adaptar este producto para satisfacer sus propias necesidades:** Refiere a que la metodología AUP es un producto de fácil uso, indiferentemente de la herramienta usada.

1.5.2 Lenguaje de modelado

Como lenguaje de modelado se utilizará UML – Unified Modeling Language, en español Lenguaje Unificado de Modelado. UML es un estándar OMG[3] diseñado para visualizar, especificar, construir y documentar software orientado a objetos. Es una simplicidad de la realidad (5). El trabajo de UML es esencial porque con él se logra establecer un listado de requerimientos y estructuras necesarias para idear un sistema antes del proceso de implementación. Además en este trayecto se logra comprender mejor lo que estamos construyendo y en algunos casos se describen las oportunidades de simplificación y reutilización.

1.5.3 Herramientas de modelado

Para el modelado de nuestra propuesta de solución se trabajará con la herramienta CASE Visual Paradigm (VP). Herramienta multiplataforma de gran alcance y fácil de utilizar, además propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Es de código abierto y forma parte de las políticas del país de utilizar herramientas libres, ya que no se deben pagar licencias comerciales.

Para la propuesta de solución VP aporta un gran apoyo en cuanto a la generación de diagramas UML como es el caso de los diagramas de clases, diagramas de casos de uso del sistema, diagramas de secuencias. Por otro lado en el proceso de captura de requisitos con el diagrama de requisitos, en el modelado de bases de datos con los diagramas de entidad-relación, diagramas ORM[4] y por último en las matrices de trazabilidad.

1.5.4 Servidores web

Los servidores se encargan de la gestión de cualquier aplicación en el lado del servidor realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con

[3] ***OMG, Object Management Group*** es un consorcio, formado en 1989, dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.
[4] ***ORM, Object-Relational Mapping***, en español Mapeo Objeto-Relacional es un modelo de programación que consiste en la transformación de las tablas de una base de datos.

el cliente generando una respuesta en cualquier lenguaje o aplicación en el lado del cliente.

Apache

Es un servidor web HTTP de código abierto potente y flexible, que está disponible para las plataformas Windows, Linux y MacOS. Apache HTTP Server presenta muchas características y funciones que lo califican como un servidor robusto y rápido. Posee gran popularidad entre los usuarios siendo el más común y utilizado en todo el mundo, lo que va a facilitar conseguir la ayuda y soporte al presentarse un problema. Trabaja con gran cantidad de lenguajes: Perl, PHP y otros de script, Java y páginas JSP, teniendo todo el soporte que se necesita para tener páginas dinámicas. Por su diseño modular es muy sencillo ampliar las capacidades del servidor web Apache. En la actualidad hay casi infinitos módulos para Apache que los vamos instalando a medida que se necesitan.

Cherokee

Es un servidor web con una completa funcionalidad que según algunas pruebas de rendimiento está entre los más rápidos que existen. Es un servidor web rápido, flexible y fácil de configurar, compatible con las principales tecnologías existentes. Soporta *FastCGI*, SCGI, PHP, CGI, SSI, TLS y conexiones encriptadas SSL, *Virtual hosts,* codificación sobre la marcha, balanceo de carga, logs compatibles con Apache, balanceo de bases de datos, *Reverse HTTP Proxy, Traffic Shaper, Video Streaming, entre* otros. Es multiplataforma bajo la licencia GNU[5] (6).

Luego de analizados se escogió Apache por ser uno de los servidores HTTP más usados en más del 80% de los sitios web en el mundo debido a su modularidad, código abierto, diversidad de entornos (multiplataforma) y fácil soporte.

1.5.5 Lenguaje de programación para aplicaciones web

PHP - PHP Hypertext Pre-processor

[5] ***Licencia GNU (General Public License),*** en español Licencia Pública General es la licencia más ampliamente usada en el mundo del software.

Es un lenguaje de programación multiplataforma de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. La última versión estable es la 5.5.12 (30 de abril del 2014).

En la actualidad es ampliamente usado en entornos de desarrollo web por su facilidad de uso, su integración perfecta con HTML y su versatilidad de uso en diferentes Sistemas Operativos. Tanto es su expansión, que se calcula su uso en torno a más de 20 millones de sitios web y un millón de servidores en todo el mundo (7).

Java

Este lenguaje de programación multiplataforma fue originalmente desarrollado por Jame Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, es un lenguaje de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Java tiene un tipo de dato fuerte y estático. La última versión estable de este lenguaje es Java Standard Edition 8 (18 de marzo del 2014).

PHP y Java son dos tecnologías que desde su lanzamiento siempre han venido precedidas de debates acerca de las ventajas y desventajas. En el mundo de los desarrolladores, en la mayoría de los temas, especialmente en los lenguajes de programación existe diversidad de criterios. En esta investigación se ha llegado a la selección de PHP por su simplicidad, pues permite la generación de código de manera rápida ya que cuenta con una amplia gama de funciones predefinidas y seguridad porque tiene protección contra ataques, suministrando niveles de seguridad. Además, no necesita la generación de código intermedio como JAVA (máquina virtual) que relentiza el sistema requiriendo más recursos de hardware.

1.5.6 *Framework* de desarrollo

Framework de desarrollo es una estructura conceptual y tecnológica de soporte definido, normalmente con productos de trabajo o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Después de seleccionado el lenguaje de programación con que vamos a trabajar pasaremos al análisis del *framework* de trabajo. En PHP, los *frameworks* más importantes son: Zend, Symfony, Codeigniter, Yii y Cake PHP.

A continuación se describen algunas de las principales características que sustentan la selección del *framework*.

	Yii	CodeIgniter	Zend	Cake PHP	Symfony
Necesidad de técnicas	PHP5, OOP	PHP, OPP básico	PHP5, OOP, Diseño	PHP, OOP, Habilidad de ordenar el código fuente	PHP, OOP, ORM, Consola
Proyectos	Pequeños-Grandes	Pequeños-Grandes	Medios-Grandes	Pequeños-Medios	Grandes
Versión PHP	5.2	5.2	5.2	5.2	5.2
Estructura de catálogo rígida	No	Si	No	Si	Si
Soporte oficial	Si	Si	Si	Si	Si
Instalación y complejidad de ajuste	Media	Baja	Alta	Baja	Alta
Configuraciones adicionales	Poco	Poco	Notable	Poco	Notable
Apoyo completo de ORM	Disco activo	No	No	Si (no muy conveniente)	Si
Documentación y muestras	Excelente	Excelente	Bueno	Normal	En proceso
Pruebas de unidad para el código fuente	Si	No	Si	Si	Si
Comunidad	Si	Fórum, Wiki	Si	Si	Si

		tutoriales, Blogs			
Licencia	Nueva BSD	Codeigniter License Agreement	Nueva BSD	MIT	MIT

Tabla 3: Análisis y comparación entre frameworks de PHP.

Se optó por el *framework* Yii ya que:

- Posee una curva rápida de aprendizaje, tiene una excelente comunidad, foros y demás.
- Utiliza el patrón Modelo-Vista-Controlador (MVC) de una manera fácil de entender y muy organizado.
- Posee una excelente documentación.
- Cuenta con un módulo Gii para la autogeneración de código, no sólo del modelo, sino que tiene *scafolding* el cual te genera vistas y controladores (CRUD). Además, puedes generar o definir tus propias plantillas para esa autogeneración. En caso de no gustar las plantillas que trae por defecto, se les puede cambiar el estilo y la estructura de los directorios por lo que se convierte en un *framework* bastante flexible.

1.5.7 Gestores de bases de datos

Un gestor de base de datos es un software que permite introducir, organizar y recuperar la información de las bases de datos, así como administrarlas. Existen distintos tipos de gestores de bases de datos: relacional, jerárquico, entre otros.

De manera general se analizaron dos gestores de bases de datos y según sus características y potencialidades se seleccionó uno para la propuesta de solución:

MySQL

Es un SGBD relacional multiplataforma. Posee un sistema de privilegios y contraseñas que es muy flexible y seguro. Soporta grandes bases de datos. MySQL tiene soporte para comandos SQL para chequear, optimizar y reparar tablas. Es muy rápido, seguro y fácil de usar, esto significa que es un servidor bastante apropiado para acceder a bases

de datos en internet. Presenta características como son: conectividad segura, transacciones y claves foráneas, replicación, búsqueda de indexación de campos de texto y disponibilidad en gran cantidad de plataformas y sistemas (8).

PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional. Está liberado bajo licencia BSD, además es extensible, multiplataforma y presenta modelos de negocios rentables con instalaciones a gran escala. Tiene ahorros considerables en costos de operación. El software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características de rendimiento.

Una vez analizados y comparados los SGBD se seleccionó PostgreSQL por ser libre, robusto y para sistemas serios e importantes (bases de datos de bancos) garantiza la consistencia de dichas bases de datos. Además, aunque en ocasiones puede ser algo lento en su proceso, soporta transacciones, tiene mejor soporte para triggers y procedimientos en el servidor, es una tecnología multiplataforma y puede ser usado en diversos entornos de desarrollo.

1.5.8 Entornos de desarrollo integrado

Un *Integrated Drive Electronics (IDE)* es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. A continuación se analizaron dos de los IDEs acorde a las tecnologías anteriormente seleccionadas:

NetBeans

Programa que sirve como IDE que permite programar en distintos lenguajes, es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), además ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. La IDE de NetBeans es

perfecta y muy cómoda para los programadores. Tiene un excelente balance entre una interfaz con múltiples opciones y un aceptable completamiento de código.

Eclipse

Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). En Eclipse se pueden usar diferentes lenguajes de programación como: Java, ANCI C, C++, JSP, sh, perl, php.

Para el desarrollo de la propuesta de solución se seleccionó el IDE NetBeans por las siguientes razones:

- Carácter multiplataforma.
- Tiene una amplia gama de soporte de lenguajes de programación y está respaldado por una fuerte comunidad de desarrolladores. Mientras Eclipse está basado en *plugin* y gran parte de sus funcionalidades viene de estos.
- Viene con soporte incorporado en los drivers, SQL, MySQL y Oracle, además de que incluye otros, por lo que resulta más fácil para los principiantes.

1.6 Conclusiones parciales

Después de haber realizado un estudio del mercado internacional e institucional de la disciplina de requisitos y las herramientas de gestión de requisitos, los productos de trabajo del EPD, los principales framework para PHP, los SGBD, los IDE y metodologías de desarrollo se llegan a las siguientes conclusiones:

- El análisis de los principales términos asociados a la propuesta de solución permitió tener una mejor comprensión de la misma y a la vez evacuar las dudas de los procesos que los interrelacionan.
- La investigación y el estudio de las herramientas para la gestión de requisitos evidenció que eran herramientas desarrolladas para fines distintos y no cumplían

en su totalidad con las necesidades de la propuesta de solución aunque sirvieron para incorporar nuevas opiniones de negocio al desarrollo de la misma.

- El estudio de las principales herramientas, tecnologías y metodologías permitió hacer una selección adecuada de las mismas para la confección del sistema.

Capítulo 2: Características y Diseño del Sistema

2.1 Introducción

En este capítulo se diseña y modelan los artefactos que ayudan a la construcción de la aplicación web. Se expondrán los requisitos funcionales, no funcionales, mecanismos, sus casos de usos y se elaboran sus diagramas de clases e interacción (diagrama de secuencia). Por otro lado, se obtiene el diagrama de clases persistentes y a la vez el modelo de datos, este último como soporte a la base de datos futura de la propuesta de solución.

Además, una vez identificados los requisitos funcionales y no funcionales se realiza un diagrama de casos de uso, mediante el cual se podrá establecer la interacción y las relaciones entre los actores y el sistema.

2.2 Propuesta de Solución

Para el desarrollo de la propuesta de solución, el cual es un sistema web llamado AREXPRO, se incluirán algunos rasgos pertenecientes a la estrategia marcaria XEDRO debido a la vinculación temática de esta línea de diseño UCI con el negocio a tratar en esta investigación. La misma estará destinada a la generación de los productos de trabajo del expediente de proyecto 4.0 aplicables a los escenarios de la disciplina de requisitos de los proyectos del centro VERTEX, con el propósito de ayudar a los integrantes de los equipos de proyectos en el desempeño de sus tareas y proveer disponibilidad y centralización de la información en cualquier momento. Dentro de los productos de trabajo a generar se encuentran: criterios para validar requisitos del cliente y del producto, registro de proveedores de requisitos, evaluación de requisitos y casos de uso del sistema, especificación de requisitos de software y casos de uso del sistema, modelo conceptual, modelo de diseño, historias de usuario, diseño del videojuego, modelo de diseño y especificación de mecanismos. Incluye algunas de estas características:

- La gestión de los proyectos, la posibilidad de evaluación de los requisitos definidos tanto del cliente como del producto, la gestión de los casos de uso.
- Velar porque los productos de trabajo y documentación que se generan se encuentran regidos por la versión 4.0 del EPD que poseen cambios con la versión anterior. Estos cambios se evidencian a la hora de realizar la especificación de requisitos en cuanto a la taxonomía para evaluar los requisitos no funcionales y la manera de cómo describirlos, siendo en esta nueva versión por tablas con una serie de atributos a llenar.
- Incluye una línea de desarrollo específicamente para videojuegos con sus productos de trabajo correspondientes: diseño de videojuego y especificación de mecanismos.
- Permite la confección de matrices bidireccionales entre requisitos - requisitos, requisitos - casos de uso, requisitos - casos de prueba y requisitos - entregables.
- Cuenta con un sistema de control de acceso en el cual se le asignan roles y permisos a los usuarios que pueden o no ser del centro para el cumplimiento de sus tareas, evitando así la mala manipulación o modificación de la información ya guardada.
- Permite la generación de productos de trabajo en formato PDF en caso necesario. Reportes que fueron creados con la herramienta IReport manteniendo un estándar de diseño muy similar a las plantillas ofimáticas anteriormente usadas.
- Cuenta con un módulo para la disciplina de pruebas el cual toma los casos de uso registrados en la fase de análisis y los convierte en posibles casos o secciones de prueba conjuntamente con las variables descritas.
- Posee una página principal en la cual se brinda un resumen de estadísticas necesarias para los usuarios sobre los proyectos que se encuentran en el sistema.

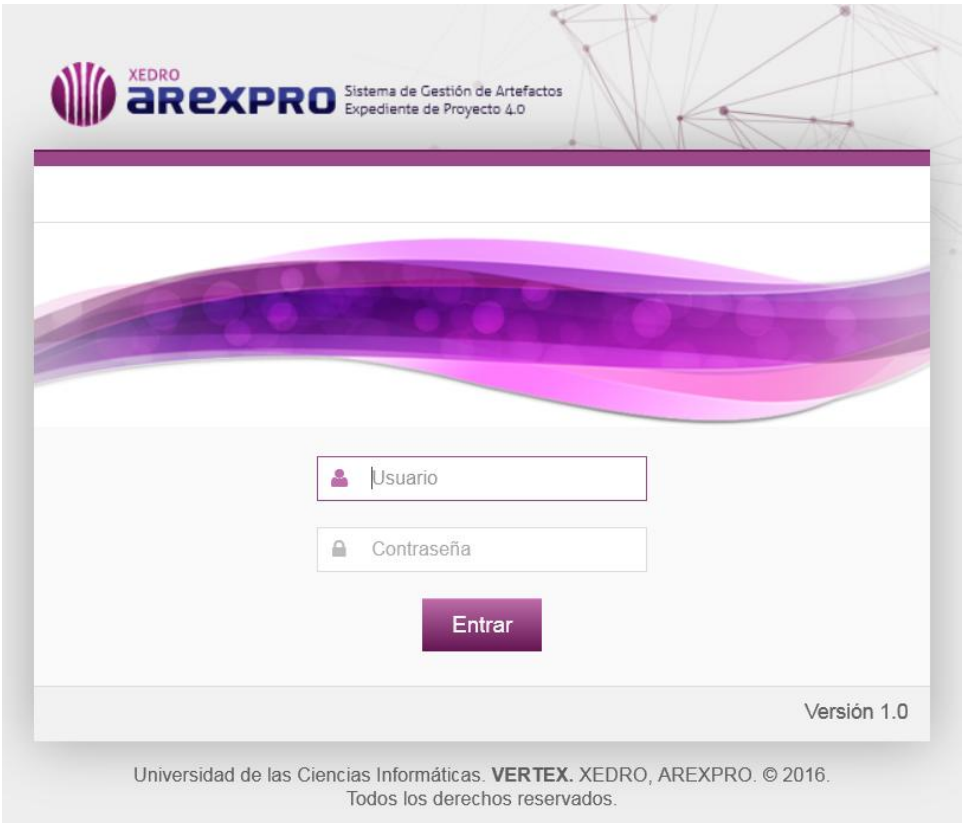


Ilustración 2: Interfaz de la página de Autenticación del Sistema.

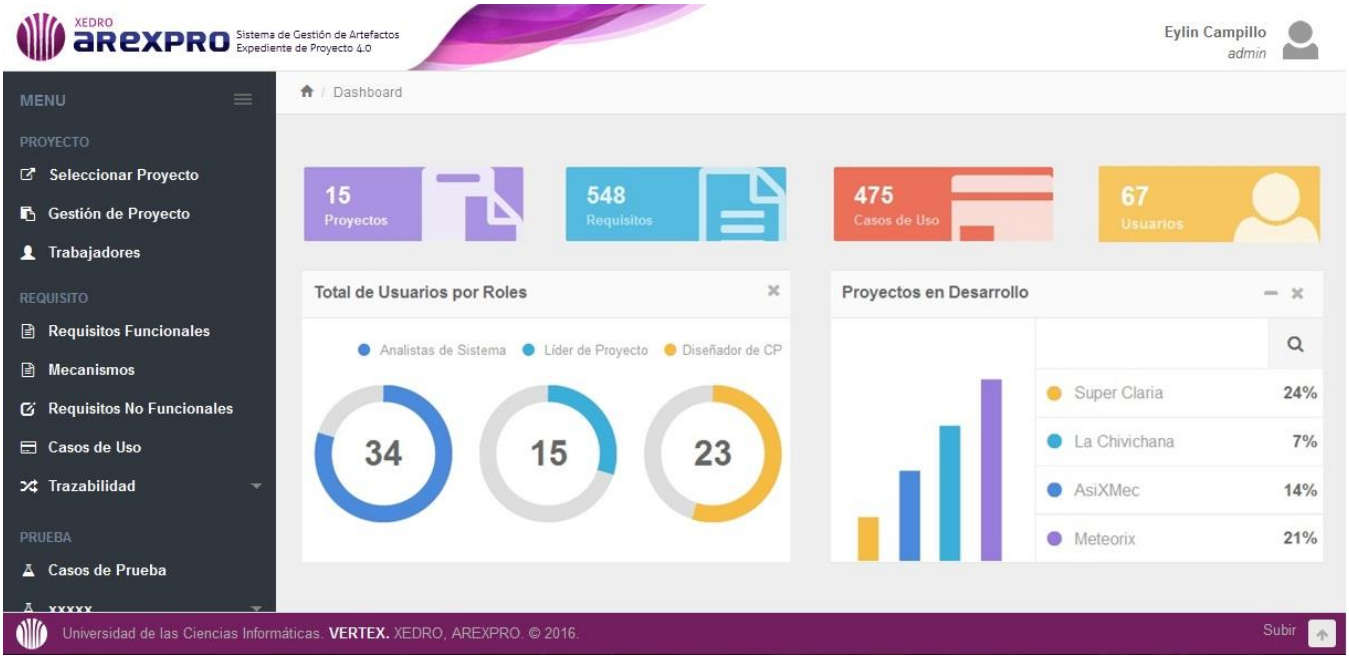


Ilustración 3: Interfaz Principal del Sistema.

2.3 Modelo de dominio

El modelo de dominio puede ser tomado como punto de partida para el diseño de una aplicación web, es por eso que por lo general para comprender el sector de negocios al cual pertenece el sistema es necesario realizarlo, así como su contribución a la identificación de actores, eventos, transacciones y objetos involucrados en el mismo. Este modelo ayuda a que el personal relacionado (clientes o equipo de desarrollo) con el sistema mantenga una correlación en la terminología a la hora del cumplimiento y desarrollo de la propuesta de solución.

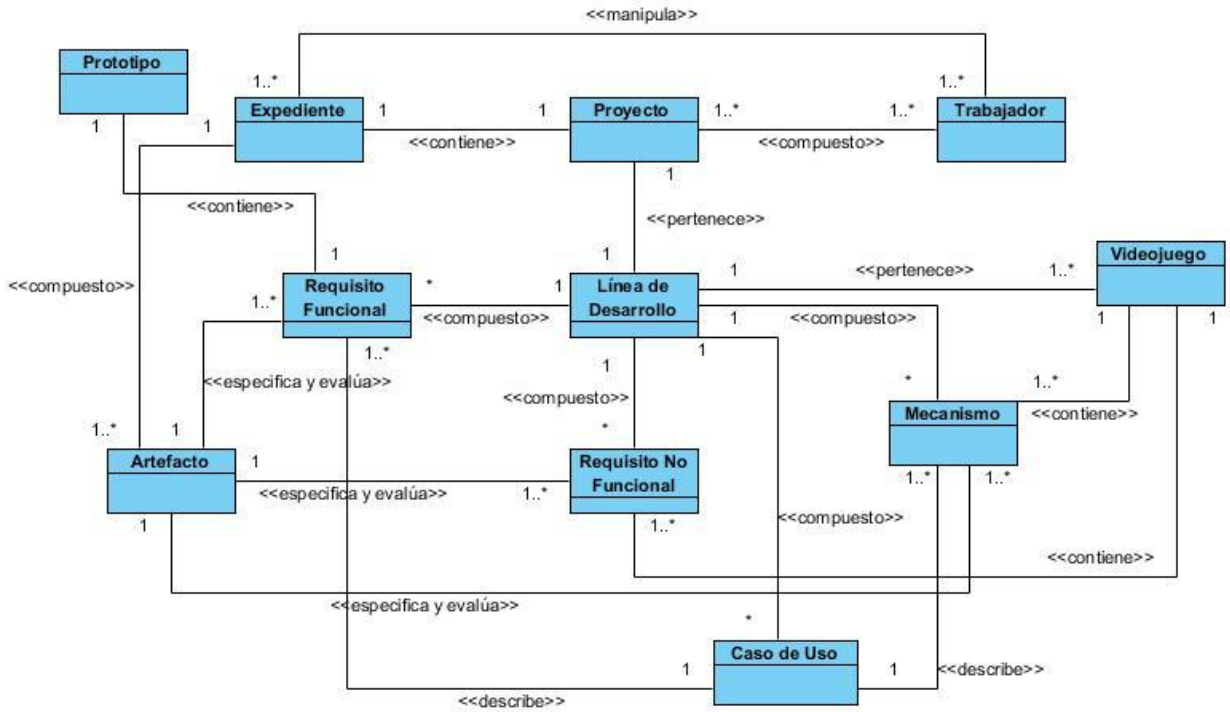


Ilustración 4: Modelo de Dominio.

Entidad	Descripción
Trabajador	Forma parte de un proyecto y manipula el expediente de trabajo.
Proyecto	Está compuesto por trabajadores y cuenta con un expediente de trabajo.
Expediente	Está compuesto por productos de trabajo y manipulado por los trabajadores.

Artefactos	Especifica y evalúa los requisitos funcionales, no funcionales y mecanismos y forma parte del expediente de trabajo.
Línea de Desarrollo	Está compuesto por los requisitos funcionales, no funcionales, mecanismos, casos de usos.
Videojuego	Pertenece a una línea de desarrollo, contiene mecanismos y requisitos no funcionales.
Requisito Funcional	Compone una línea de desarrollo, es especificado y evaluado por un producto de trabajo, contiene un prototipo y describe un caso de uso.
Requisito No Funcional	Compone una línea de desarrollo, es especificado y evaluado por un producto de trabajo y conforma un videojuego.
Mecanismo	Compone una línea de desarrollo, es especificado y evaluado por un producto de trabajo, conforma un videojuego y describe un caso de uso.
Caso de Uso	Es la descripción de mecanismos y de requisitos funcionales.
Prototipo	Contiene la representación de un requisito funcional.

Tabla 4: Descripción de las Entidades.

2.4 Descripción de los actores del sistema

Con el sistema resultante interactuarán actores como: el administrador del sistema, líder de proyecto, analista de sistema y diseñador de casos de pruebas. Cada uno de estos actores tendrá asociado un rol de desarrollo y estos a su vez obtendrán permisos específicos o generales. Entre los permisos generales se encuentran el autenticar usuario y consultar reportes.

Actores	Descripción
Usuario del Sistema	Persona (administrador del sistema y asesor de análisis y diseño) encargada de autenticarse, gestionar reportes, exportar productos de trabajo en .pdf.
Administrador del Sistema	Tendrá la posibilidad de gestionar usuarios, gestionar roles, gestionar nomencladores y asignar trabajadores a proyecto.
Jefe de Proyecto	Tendrá la posibilidad de realizar la funcionalidad gestionar proyecto, así como las generales del asesor de análisis y diseño, gestionar productos de trabajo y gestionar matriz.

Analista de Sistema	Tendrá la posibilidad de gestionar los casos de uso del sistema, gestionar los requisitos funcionales, no funcionales y los mecanismos, evaluar los requisitos funcionales y los casos de uso del sistema, así como las del asesor de análisis y diseño, generar productos de trabajo pertenecientes a la línea de desarrollo seleccionada y generar matriz deseada.
Asesor de Análisis y Diseño	Persona (jefe de proyecto y analista de sistema) encargada de gestionar todos los productos de trabajo y generar matriz (requisito x requisito, requisito x caso de uso, requisito x caso de prueba, requisito x entregable).

Tabla 5: Descripción de los actores del sistema.

2.5 Requisitos del sistema

"Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema" (9).

A continuación, se relacionan los requisitos funcionales (RF) y requisitos no funcionales (RNF) que cuenta la herramienta AREXPRO:

2.5.1 Requisitos Funcionales

Los requisitos funcionales son condiciones que el sistema debe cumplir. Estos se mantienen invariables sin importar con que propiedades o cualidades se relacionen. Para el desarrollo de la herramienta se identificaron los siguientes RF:

Identificador	Requisito
RF 1.	Autenticar usuario
RF 2.	Registrar usuario
RF 3.	Modificar usuario
RF 4.	Eliminar usuario
RF 5.	Listar usuarios
RF 6.	Registrar rol
RF 7.	Modificar rol
RF 8.	Eliminar rol
RF 9.	Listar roles
RF 10.	Registrar trabajador

RF 11.	Modificar trabajador
RF 12.	Eliminar trabajador
RF 13.	Listar trabajadores
RF 14.	Asignar trabajador a proyecto
RF 15.	Registrar proyecto
RF 16.	Modificar proyecto
RF 17.	Eliminar proyecto
RF 18.	Listar proyectos
RF 19.	Seleccionar proyecto
RF 20.	Registrar requisito funcional
RF 21.	Modificar requisito funcional
RF 22.	Eliminar requisito funcional
RF 23.	Listar requisitos funcionales
RF 24.	Registrar requisito no funcional
RF 25.	Modificar requisito no funcional
RF 26.	Eliminar requisito no funcional
RF 27.	Listar requisitos no funcionales
RF 28.	Registrar caso de uso
RF 29.	Modificar caso de uso
RF 30.	Eliminar caso de uso
RF 31.	Listar casos de uso
RF 32.	Registrar mecanismo
RF 33.	Modificar mecanismo
RF 34.	Eliminar mecanismo
RF 35.	Listar mecanismos
RF 36.	Establecer dependencia Requisito x Requisito
RF 37.	Establecer dependencia Requisito x Caso de Uso del Sistema
RF 38.	Establecer dependencia Requisito x Entregable
RF 39.	Establecer dependencia Requisito x Caso de Prueba
RF 40.	Visualizar matriz de trazabilidad Requisito x Requisito
RF 41.	Visualizar matriz de trazabilidad Requisito x Caso de Uso del Sistema
RF 42.	Visualizar matriz de trazabilidad Requisito x Entregable
RF 43.	Visualizar matriz de trazabilidad Requisito x Caso de Prueba
RF 44.	Crear productos de trabajo Criterio para Validar Requisito del Cliente
RF 45.	Modificar producto de trabajo Criterio para Validar Requisito del Producto
RF 46.	Crear producto de trabajo Criterio para Validar Requisito del Cliente
RF 47.	Modificar producto de trabajo Criterio para Validar Requisito del Producto
RF 48.	Crear producto de trabajo Evaluación de Requisitos
RF 49.	Modificar producto de trabajo Evaluación de Requisitos
RF 50.	Crear producto de trabajo Evaluación de Casos de Uso del Sistema
RF 51.	Modificar producto de trabajo Evaluación de Casos de Uso del Sistema
RF 52.	Crear producto de trabajo Especificación de Requisitos de Software
RF 53.	Modificar producto de trabajo Especificación de Requisitos de Software
RF 54.	Crear producto de trabajo Especificación de Caso de Uso del Software
RF 55.	Modificar producto de trabajo Especificación de Caso de Uso del Software
RF 56.	Crear producto de trabajo Registro de Proveedores
RF 57.	Modificar producto de trabajo Registro de Proveedores
RF 58.	Crear producto de trabajo Glosario de Términos
RF 59.	Modificar producto de trabajo Glosario de Términos

RF 60.	Crear producto de trabajo Requisitos Rechazados
RF 61.	Modificar producto de trabajo Requisitos Rechazados
RF 62.	Crear producto de trabajo Salida del Sistema
RF 63.	Modificar producto de trabajo Salida del Sistema
RF 64.	Crear producto de trabajo Modelo Conceptual
RF 65.	Modificar producto de trabajo Modelo Conceptual
RF 66.	Crear producto de trabajo Reporte de trazabilidad
RF 67.	Modificar producto de trabajo Reporte de trazabilidad
RF 68.	Crear producto de trabajo Diseño de Videojuegos
RF 69.	Modificar producto de trabajo Diseño de Videojuegos
RF 70.	Crear producto de trabajo Especificación de Mecanismos
RF 71.	Modificar producto de trabajo Especificación de Mecanismos
RF 72.	Crear productos de trabajo Modelo de diseño
RF 73.	Modificar productos de trabajo Modelo de diseño
RF 74.	Exportar los productos de trabajo en .pdf
RF 75.	Registrar nomenclador
RF 76.	Modificar nomenclador
RF 77.	Eliminar nomenclador
RF 78.	Listar nomencladores
RF 79.	Consultar Perfil de Usuario
RF 80.	Generar Reportes
RF 81.	Salir del Sistema

Tabla 6: Selección de Requisitos Funcionales del Sistema.

2.5.2 Requisitos No Funcionales

- **Usabilidad**

RnF 1. El sistema deberá presentar una navegación intuitiva, ya que el nombre de las funciones que tendrá para trabajar será acorde a la finalidad de la misma. Además, contará con migas de pan y títulos para identificar las páginas en donde se encuentre el usuario.

RnF 2. Contará con información o ayuda sobre las funciones a realizar por el usuario, destacando si son correctas o no.

RnF 3. El sistema comprenderá una fácil navegación por los diferentes contenidos que incluye. Esto se tiene en cuenta para que el usuario tenga el conocimiento y el dominio requerido.

- **Eficiencia**

RnF 4. El consumo de memoria RAM deberá ser menor de 512 Mb.

- **Restricciones de diseño e implementación**

RnF 5. Herramienta de modelado a emplear Visual Paradigm 8.0 y para el modelo de datos ER/Studio 8.0.

RnF 6. Lenguaje de programación a utilizar PHP bajo el IDE NetBeans 8.0.

RnF 7. Utilizar las bibliotecas JDBC para PostgreSQL 9.2.

- **Software (cliente)**

RnF 8. Se recomienda usar Mozilla Firefox y como alternativa Opera o Chrome porque son los que más se entienden con el estándar de *css* y *javascript*.

- **Software (servidor)**

RnF 9. Como IDE se utilizará NetBeans en su versión 8.0 o superior.

RnF 10. Como *framework* de desarrollo se empleará Yii en su versión 1.1.14.

RnF 11. Como gestor de bases de datos se utilizará PostgreSQL en su versión 9.2.

RnF 12. Se necesitará la máquina virtual de java en su versión 8 para generar reportes con IReport y para realizar las pruebas de rendimiento con JMeter.

RnF 13. Se recomienda la utilización del servidor WAMP, ya que es multiplataforma e incluye Apache y PHP.

- **Hardware (servidor)**

RnF 14. La capacidad requerida de la memoria RAM tiene que ser mayor o igual a 1Gb.

RnF 15. El disco duro que puede soportar a la aplicación debe tener una capacidad mayor o igual a 80 Gigabyte.

- **Interfaz gráfica o apariencia externa**

RnF 16. Las interfaces estarán respaldadas por los colores que pertenecen a la estrategia marcaria XEDRO Empresa-Industria que son el gris, blanco, morado y negro.

Contará con un menú superior en el que se recogen las funcionalidades generales del sistema: seleccionar proyecto, perfil de usuario y salir del sistema. Por otra parte, contará con un menú lateral que recoge las funcionalidades específicas del sistema como: gestionar proyectos, gestionar requisitos, gestionar casos de uso, entre otras y en el área del centro irá el contenido correspondiente a cada una de ellas.

- **Requisitos de licencia**

RnF 17. Licencia GPL[6]: Netbeans versión 2 (lDE de desarrollo).

RnF 18. Licencia CDDL[7]: Netbeans versión 2 (lDE de desarrollo).

RnF 19. Licencia PostgreSQL *Licence*: PostgreSQL 9.2 (Sistema de Gestión de Bases de Datos).

2.6 Diagrama de Casos de Uso del Sistema

Los casos de uso son una técnica para especificar el comportamiento de un sistema: “Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios.” Todo sistema de software ofrece a su entorno una serie de servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa (10).

Un diagrama de casos de uso del sistema tiene entre sus funciones la representación gráfica a los procesos y su interacción con los actores.

[6] **GPL** *GNU General Public License* (en español, Licencia Pública General de GNU).

[7] **CDDL** *Common Development and Distribution License* (en español, Licencia Común de Desarrollo y Distribución).

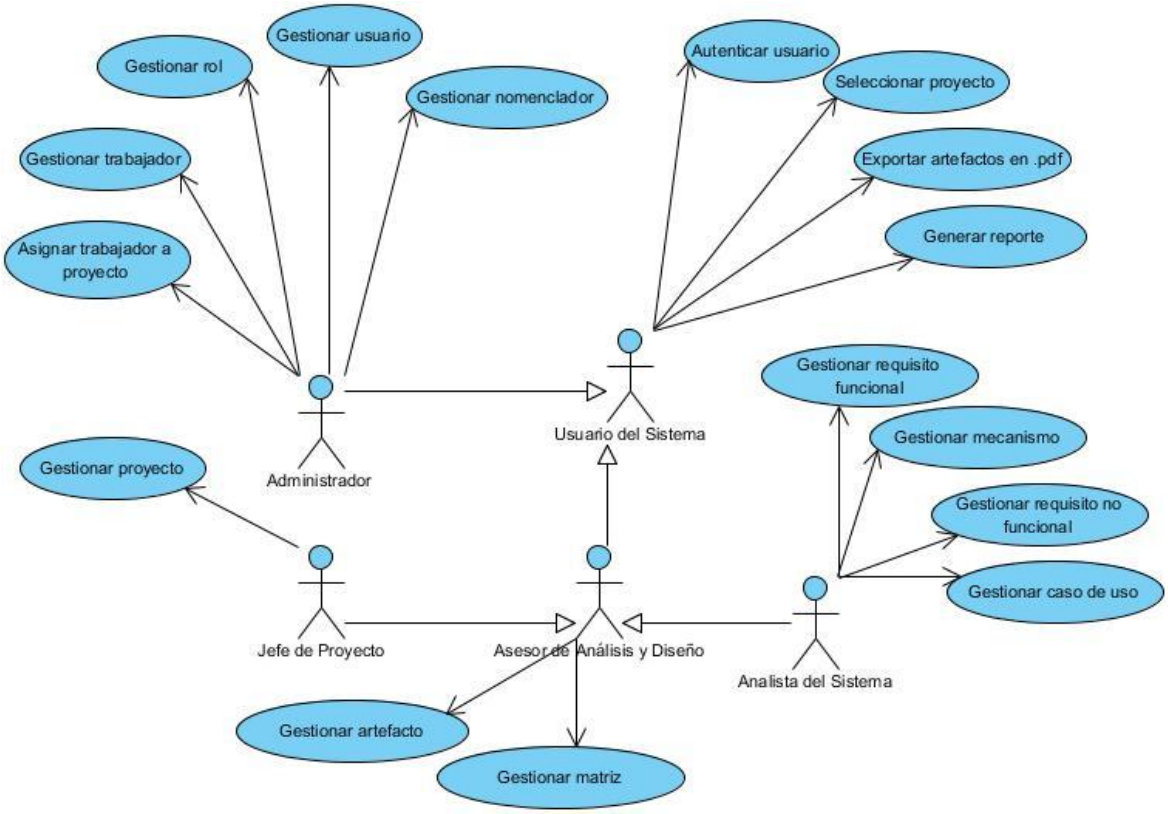


Ilustración 5: Diagrama de Casos de Uso del Sistema (DCUS).

2.6.1 Patrones de Caso de Uso del Sistema

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos (11). Se basan en la experiencia de desarrollo en ese problema planteado.

Los patrones de casos de uso utilizados para la propuesta del sistema son:

- Múltiples actores – roles comunes
- CRUD completo

2.6.2 Descripción de los Casos de Uso del Sistema

Objetivo	Permitir registrar, modificar, listar y eliminar datos acerca de un mecanismo.
Actores	Administrador del sistema, Asesor de Análisis y Diseño y el Analista de Sistema.
Resumen	El caso de uso se inicia cuando el actor decide registrar, modificar o eliminar datos acerca de un mecanismo.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Actor ya autenticado.
Postcondiciones	Se registra un mecanismo, se modifican los datos o se elimina el mismo.

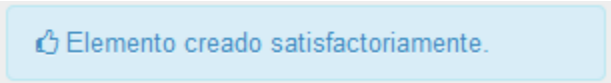
Flujo de eventos

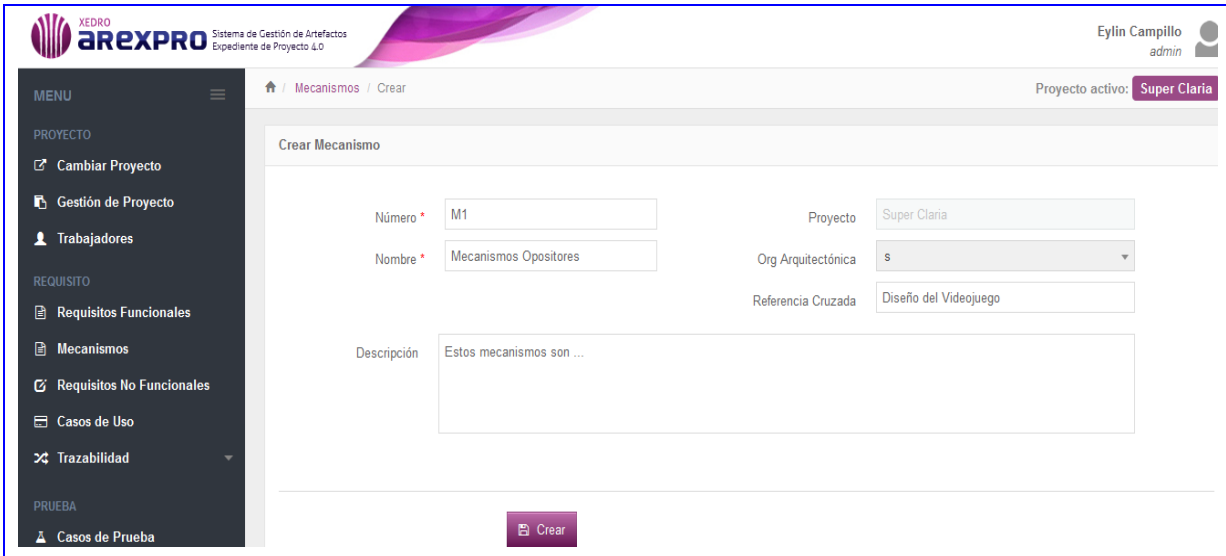
Flujo básico.

	Actor	**Sistema**
1.	Desea registrar, modificar o eliminar los datos de un mecanismo.	Muestra un menú llamado Mecanismo con las siguientes opciones: -Registrar mecanismo -Modificar mecanismo -Eliminar mecanismo
2.	Escoge una de las opciones: registrar, modificar o eliminar los datos de un mecanismo.	Ejecuta algunas de las siguientes acciones: a) Si decide registrar un mecanismo, ir a la sección “Registrar mecanismo” b) Si decide modificar los datos de un mecanismo, ir a la sección “Modificar mecanismo”. c) Si decide eliminar un mecanismo, ir a la sección “Eliminar mecanismo”.

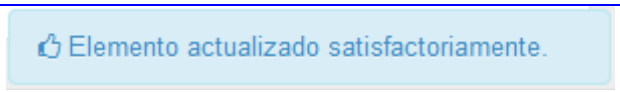
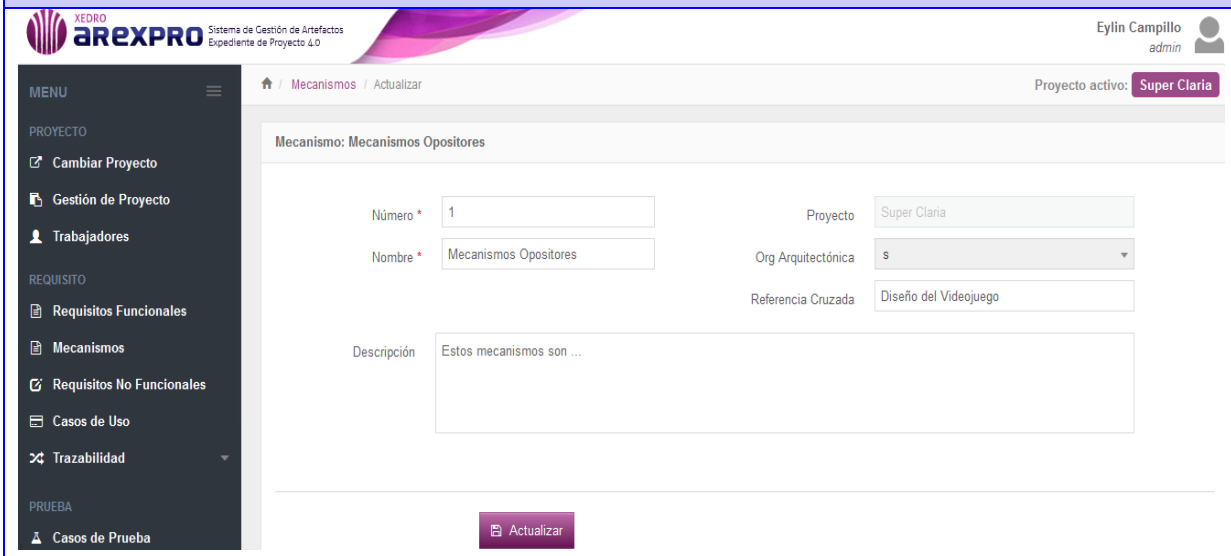
Sección “Registrar mecanismo”

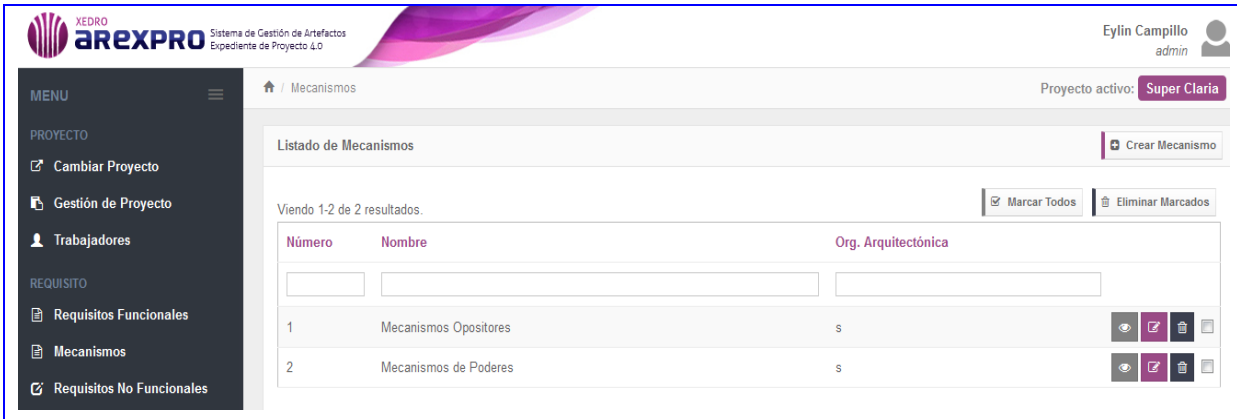
Actor	**Sistema**
Flujo básico Registrar mecanismo.	
1	Muestra un formulario con los siguientes campos a introducir: -Índice -Nombre -Prioridad

		-Descripción -Prototipo no funcional -Fichero VPP -Referencia Cruzada Y el botón Crear.
2	Introduce los datos y presiona el botón Crear.	Verifica que todos los campos estén llenos.
3		Verifica que este mecanismo no exista.
4		Almacena los datos del mecanismo y muestra el mensaje:  Finalizando así el caso de uso.
Flujo Alterno de Eventos "Campos vacíos"		
Actor		**Sistema**
Flujo básico Campos vacíos		
1		Muestra el mensaje: 
Flujo Alterno de Eventos "Mecanismo existente"		
Actor		**Sistema**
Flujo básico.		
1		Muestra el mensaje ¨Mecanismo existente¨
Prototipo elemental de interfaz gráfica de mecanismo "Registrar Mecanismo".		



Sección "Modificar Mecanismo"		
Actor		**Sistema**
Flujo básico Modificar Mecanismo.		
1		Muestra un listado con todos los requisitos funcionales y el botón Actualizar.
2	Selecciona el mecanismo a modificar y presiona el botón Actualizar.	Muestra un formulario con los datos a modificar: -Índice -Nombre -Prioridad -Descripción -Prototipo no funcional -Fichero VPP -Referencia Cruzada Y el botón Actualizar.
3	Realiza las actualizaciones deseadas en los datos y presiona el botón Actualizar.	Verifica que todos los campos estén llenos.
4		Actualiza la información incorporada al mecanismo y se emite un mensaje:

		 Finalizando así el caso de uso.
Flujo Alterno de Eventos "Campos vacíos"		
Actor		**Sistema**
Flujo básico Campos vacíos.		
1		Muestra el mensaje: 
Prototipo elemental de interfaz gráfica de mecanismo "Actualizar Mecanismo".		
		
Sección "Listar Mecanismo"		
Actor		**Sistema**
Flujo básico Listar Mecanismo.		
1		Muestra un listado con todos los requisitos funcionales.
Prototipo elemental de interfaz gráfica de mecanismo "Listar Mecanismo".		



Sección "Eliminar Mecanismo"		
Actor		**Sistema**
Flujo básico Eliminar Mecanismo.		
1		Muestra un listado con todos los requisitos funcionales y el botón Eliminar.
2	Selecciona el mecanismo a eliminar y presiona el botón Eliminar.	Muestra el mensaje de confirmación: ¿Está seguro que desea borrar este elemento? Aceptar Cancelar Y los botones aceptar y cancelar.
3	Presiona el botón aceptar.	Elimina al mecanismo seleccionado. Finalizando así el caso de uso.
Flujo Alterno de Eventos "Cancelar eliminación de mecanismo"		
Actor		**Sistema**
Flujo básico. Cancelar eliminación de mecanismo.		
1	Presiona el botón cancelar.	Vuelve al paso 4 del flujo básico de eliminar mecanismo.
-Relaciones	**CU Incluidos**	No aplica.
	CU Extendidos	No aplica.

Requisitos no funcionales	No aplica.
Asuntos Pendientes	No aplica.

Tabla 7: Descripción de Caso de Uso Gestionar Mecanismo.

2.7 Arquitectura del sistema

Para lograr una arquitectura robusta en la confección de un sistema, es una buena práctica el uso mecanismos que validen de forma teórico-práctico el buen desempeño del mismo.

Los patrones de arquitectura, los cuales expresan el esquema fundamental de organización para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos (12).

El *framework* seleccionado implementa el patrón de arquitectura Modelo Vista Controlador (MVC), el cual separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos definidos por sus nombres modelo, vista y control. Este se ve frecuentemente en aplicaciones web, donde la vista es la página HTML, la cual maneja la visualización de la información y el código que provee de datos dinámicos a la página. El modelo es el sistema de gestión de base de datos (administra el comportamiento y los datos del dominio de aplicación) y la lógica de negocio o controlador es el responsable de recibir los eventos de entrada desde la vista. Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Entre las ventajas del estilo están: soporte de vistas múltiples, adaptación al cambio, facilita la evolución por separado de ambos aspectos e incrementa la reutilización y flexibilidad. Como desventajas se tiene: la complejidad y costo de actualizaciones frecuentes.

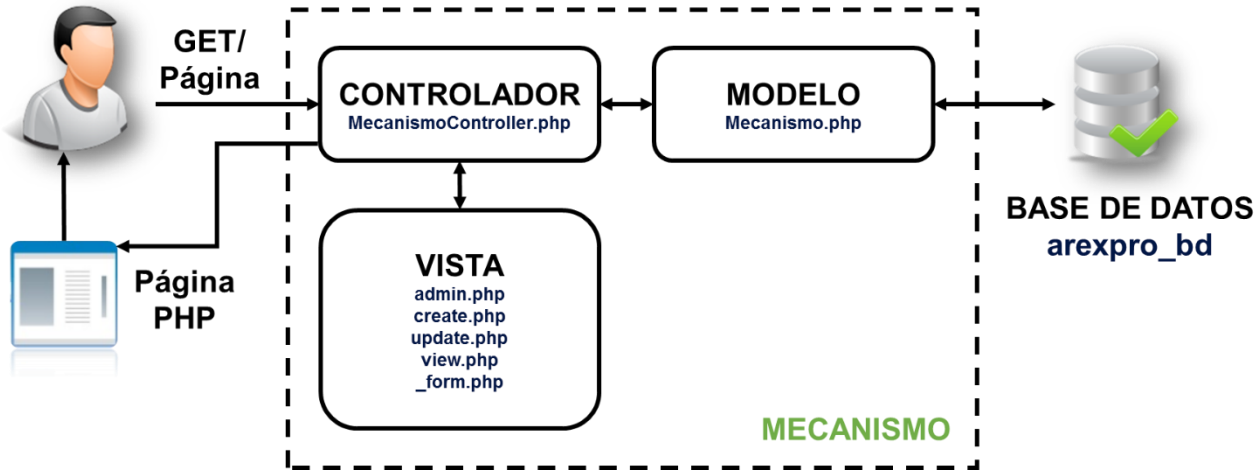


Ilustración 6: Patrón arquitectónico Modelo Vista Controlador (MVC).

Vista: Esta capa contiene las páginas HTML o vistas que son las que se encargan de visualizar la información. (Ejemplo: admin.php y create.php)

Modelo: Esta capa se encarga de recibir los eventos de entrada desde la vista o capa de presentación. (Ejemplo: Mecanismo.php)

Controlador: Esta capa trabaja mediante un sistema de gestión de base de datos, administrando, controlando el comportamiento y dominio de aplicación, así como la lógica de negocio. (Ejemplo: MecanismoController.php)

2.8 Diseño del sistema

Un patrón de diseño es una solución que surge de la experiencia práctica con varios proyectos y los equipos de desarrollo han encontrado que se puede aplicar en diversos contextos. Cada patrón de diseño describe a un conjunto de objetos y clases comunicadas.

En el diseño de la aplicación se emplearon los patrones GRASP[8], patrones para asignación de responsabilidades, fundamentalmente el patrón Experto, que plantea que se debe de asignar una responsabilidad al experto en información, o sea, la clase con

[8] **GRASP** *General Responsability Asignment Software Patterns* (en español, Patrones de Software para la Asignación General de Responsabilidades).

toda la información necesaria para llevarla a cabo (ejemplo: para crear un requisito funcional sólo lo puede crear el Modelo Requisito Funcional debido a que es el que posee todos los atributos para su creación), el de Controlador, que es el que controla la lógica del sistema (ejemplo: RequisitoFuncionalController se encarga de realizar todos los eventos asociados a la Gestión de Requisitos Funcionales), así como el Creador que se evidencia en la clase controladora RequisitoFuncionalController. Esta clase se encarga de crear instancias del objeto que maneja, o sea, la clase RequisitoFuncional. Este patrón brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento, mejores oportunidades de reutilización y propicia una alta cohesión.

Además se utilizaron algunos de los patrones GOF[9] como son: *Singleton* (Creación) que garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (ejemplo, a la hora de hacer una consulta a la Base de Datos a través del ORM *CActiveRecord* que tiene el *framework* Yii se hace una sola instancia a esa conexión) y *Decorator* (Estructural) que permite añadir funcionalidades dinámicamente a una clase con la creación de otra clase, sin necesidad de la herencia. En el *framework* Yii, los *layouts* decoran a las vistas, pues en un momento determinado les confieren propiedades dinámicas que no poseían. En esta aplicación se implementan dos *layouts (login_layout y main_layout, el primero enriquece la vista de la página de autenticación y el segundo al resto de las páginas del sistema).*

2.8.1 Diagramas de Clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y

[9] **GoF** *Pattern Gang of Four* (en español, Pandilla de los Cuatro). Clasificación de patrones en tres categorías: creacionales, estructurales y de comportamiento.

relaciones. A continuación, se muestran el diagrama de clases del diseño del CU Gestionar Mecanismo.

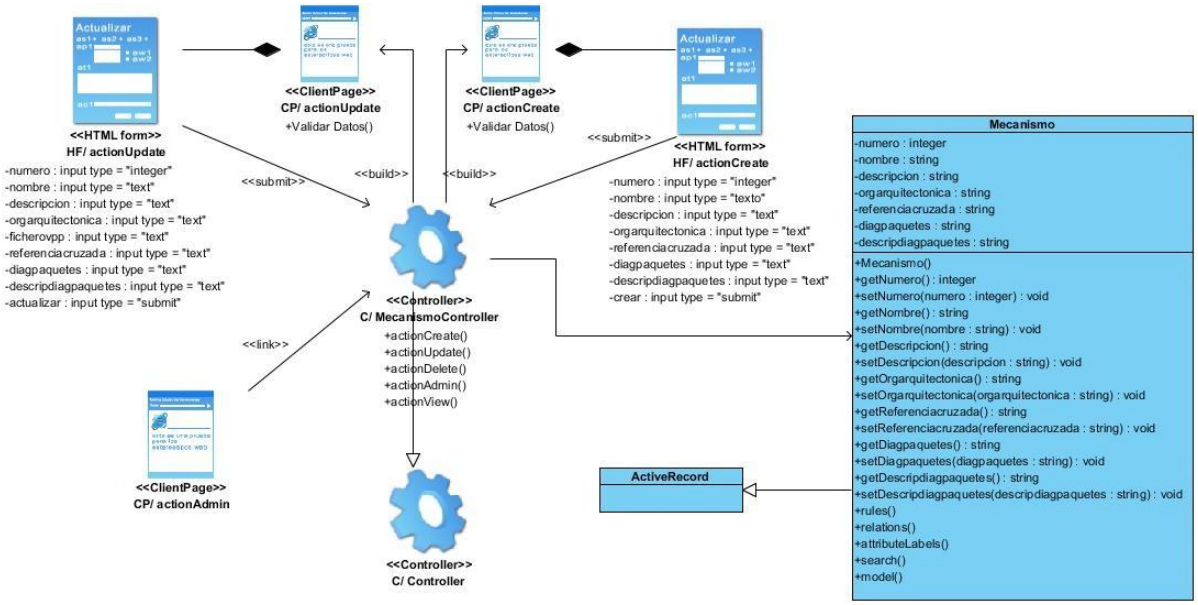


Ilustración 7: Diagrama de Clases del CU Gestionar Mecanismo.

Además, de los métodos internos de validar que tienen las páginas clientes, el sistema trabaja con:

- jQuery, una biblioteca de JavaScript que permite en el sistema manejar eventos, desarrollar animaciones, trabajo con AJAX [10], obtención de información del navegador y operar con objetos y vectores.
- CSS, hoja de estilo en cascada utilizada para describir el aspecto y formato de un documento escrito en lenguaje de marcas, dando la posibilidad de eliminar el uso de tablas para el diseño.

Es por ello que se define el diagrama de paquetes del sistema para evidenciar el uso de estas dos bibliotecas usadas por todas las páginas clientes o vistas.

[10] **AJAX** (en español, JavaScript asincrónico y XML), es una técnica de desarrollo web para crear aplicaciones interactivas.

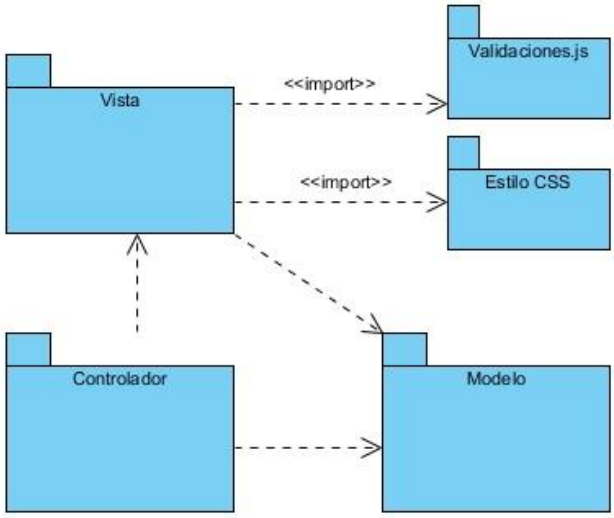


Ilustración 8: Diagrama de paquetes.

2.8.2 Diagramas de Interacción

Un diagrama de interacción, representa la forma en como un cliente (actor) u objetos (clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente (13).

En este trabajo se presentará específicamente el diagrama de secuencia que es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML. A continuación, se muestran los diagramas que describen un escenario del CU Gestionar Mecanismo del sistema:

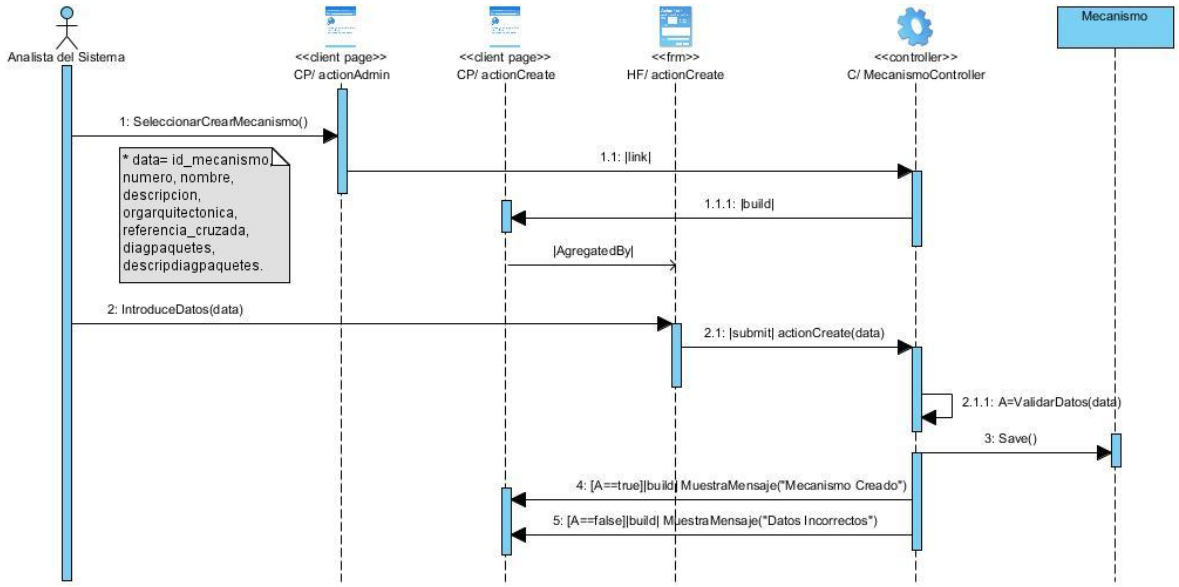


Ilustración 9: Diagrama de Secuencia CU Crear Mecanismo.

2.9 Modelo de Datos

El modelo de datos fue realizado por la herramienta ER/Studio 8.0. Se usa para el diseño, la construcción lógica y la física de base de datos. Su ambiente es de gran alcance, de varios niveles del diseño. Se diseña para hacer más fácil de entender el estado actual de los datos de la empresa. Simple y fácil al usuario, ayuda a organizaciones para tomar decisiones en cómo resolver embotellamientos de los datos, elimina redundancia y alcanza en última instancia usos de más alta calidad que entreguen datos más eficientes y exactos a la empresa.

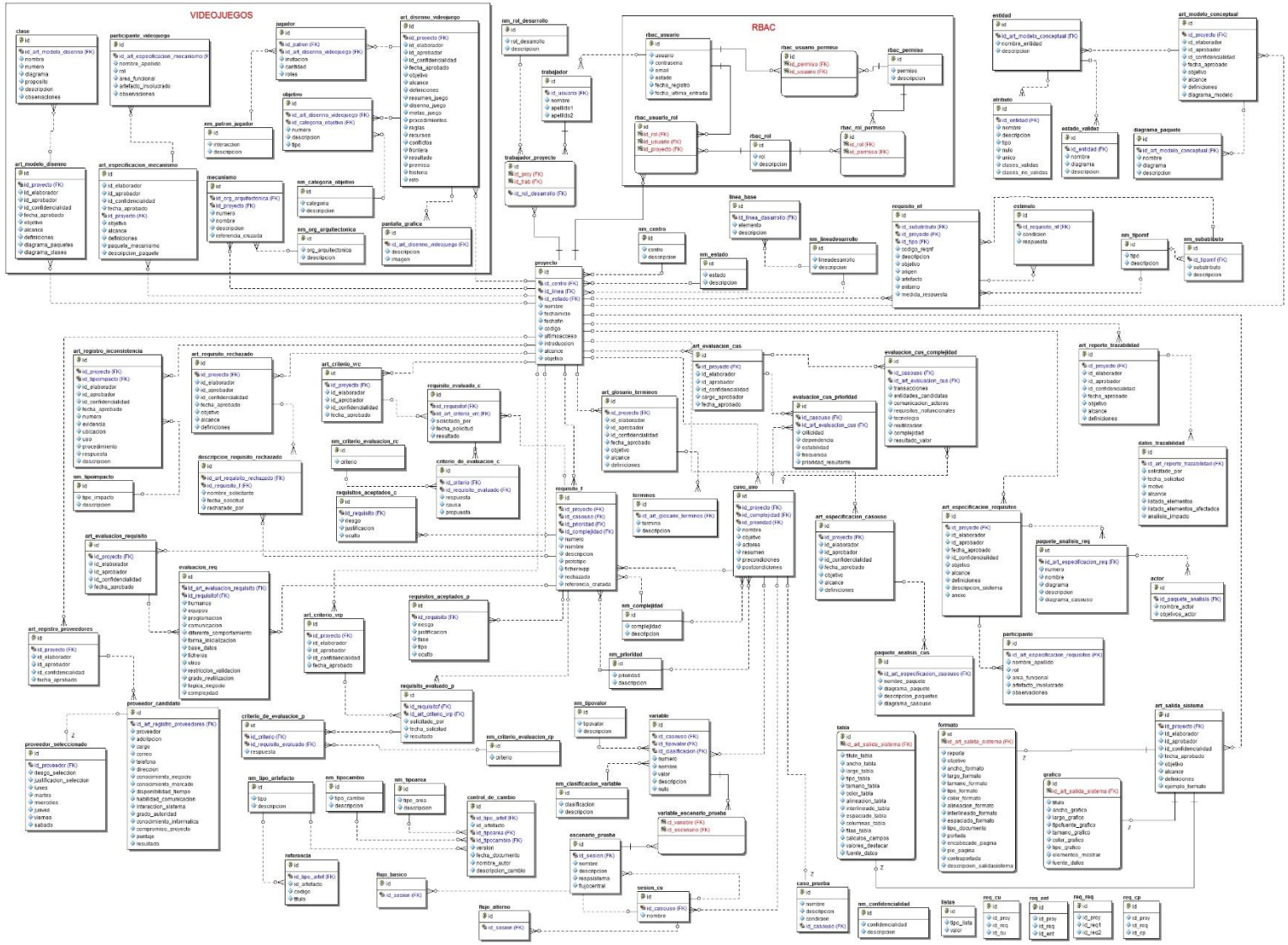


Ilustración 10: Modelo de Datos del Sistema.

2.10 Conclusiones parciales

Quedando así definido de manera específica las pautas para el desarrollo de la solución del sistema, se arrojan las siguientes conclusiones:

- La definición de los requerimientos de la aplicación permitió identificar las funcionalidades que contará el sistema propuesto, los cuales serán capaces de satisfacer las necesidades del cliente y darán respuesta al problema planteado.
- Siguiendo la metodología empleada (AUP) se generaron una serie de productos de trabajo que ayudan al análisis y modelado de la aplicación como son: modelo de dominio, diagramas de clases, diagramas de secuencias y modelo de datos con el objetivo de describir desde una fase de inicio todas las entidades y atributos del negocio, cómo estas se interrelacionan y cómo se comunican a través de peticiones o respuestas.
- Así como además, su forma de gestión en las bases de datos, siempre tratando de no duplicar información, de que no se pierda y de que aparezca el menor conflicto posible dentro de ella.
- Se describen los patrones de arquitectura y diseño utilizados para garantizar la estructura correcta y el buen funcionamiento del sistema.

Capítulo 3: Implementación y Pruebas del Sistema

3.1 Introducción

Las pruebas del software consisten en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido. Llevar a cabo las pruebas es una técnica dinámica de la verificación y validación, ya que requiere disponer de un prototipo ejecutable del sistema. En este capítulo se mostrará todo el trabajo realizado en esta disciplina.

3.2 Estándar de programación

Los estándares de programación se enfocan en definir la estructura y apariencia física del código a programar, lo que facilita su entendimiento y lectura. En la implementación de la aplicación se utilizaron diferentes estándares que se describen a continuación:

3.2.1 Definición de clases

Las declaraciones de clases tienen su llave de apertura una línea más abajo de la declaración y el nombre de la clase comienza con mayúscula. A continuación, un ejemplo de la definición de una clase.

```
class Mecanismo extends CActiveRecord
{
        /**
         * @return string the associated database table name
         */
        public function tableName()
        {
                return 'mecanismo';
        }
```

Ilustración 11: Ejemplo de definición de una clase.

3.2.2 Definiciones de métodos

El nombre de los métodos comienza con una letra minúscula, en caso de ser un nombre compuesto por dos o más palabras, estas comenzarán con letra mayúscula. A continuación, un ejemplo:

```
public function actionEliminarMarcados($items){

    try {
        $ids = explode(',', $items);
        foreach ($ids as $id)
            $this->loadModel($id)->delete();
        echo json_encode('ok');
    } catch (Exception $ex) {
        echo json_encode('Uno o varios de los elementos
    }
}
```

Ilustración 12: Ejemplo de definición de métodos.

3.2.3 Llamadas a funciones y asignación de variables

Las llamadas a las funciones de una clase se realizarán sin espacios entre el nombre de la función y los paréntesis. En el caso de utilizar llamadas a funciones de una misma clase, se utilizará *($this->)* antes del nombre de la función. Las asignaciones se realizarán mediante el signo de igualdad (=). A continuación se muestra un ejemplo:

```
$model= ArtReporteTrazabilidad::model()->findByPk($id);
$model_referencia = new Referencia; //para agregar referencia
$model_cambio = new ControlDeCambio;
```

Ilustración 13: Ejemplo de llamadas a funciones y asignación de variables.

3.2.4 Estructuras de control

Con el objetivo de mejorar la legibilidad del código fuente se define que las estructuras de control deben estar identadas. Las estructuras de control incluyen *if, for, while,* entre otras como se aprecia a continuación:

```
if(isset($_POST['ArtEspecificacionMecanismo']))
{
        $model->attributes=$_POST['ArtEspecificacionMecanismo'];
        if($model->save()){
                Yii::app()->user->setFlash('msg','Artefacto actualizado
                $this->redirect(array('admin'));
        }
}
```

Ilustración 14: Ejemplo de estructura de control identada.

3.3 Diagrama de Componentes

Este tipo de diagrama tiene como objetivo mostrar cómo el sistema está dividido en componentes (archivos, bibliotecas, ejecutables, entre otros) y las dependencias entre ellos. Además, ayudan a los desarrolladores a visualizar de una manera más fácil el camino de la implementación y sirve para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. Seguidamente se expondrá un ejemplo de este diagrama perteneciente al CU Gestionar Mecanismo.

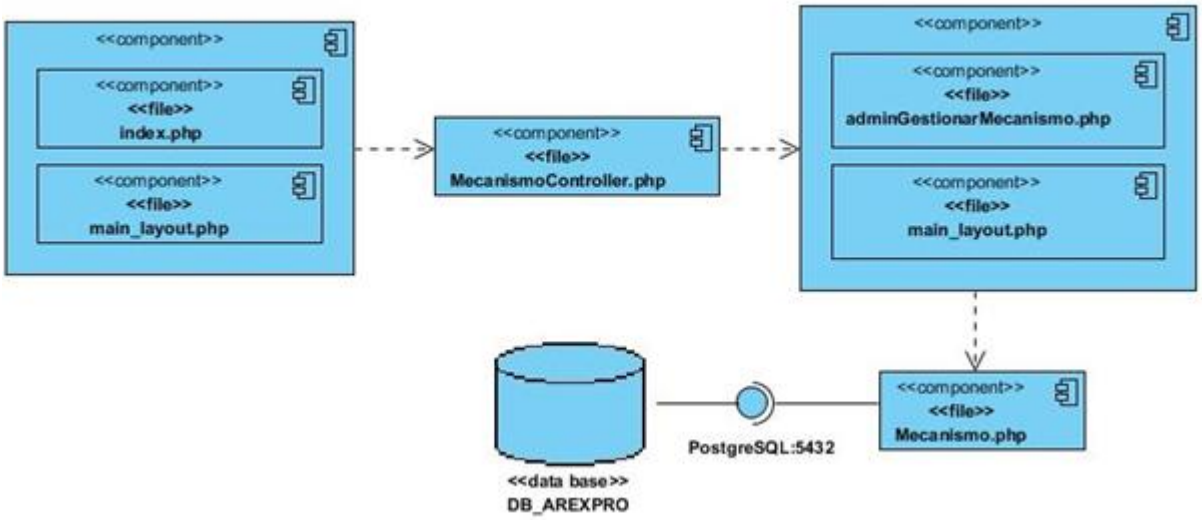


Ilustración 15: Diagrama de Componentes CU Gestionar Mecanismo.

3.4 Diagrama de Despliegue

El diagrama de despliegue permite modelar la disposición física o la topología de un sistema. Muestra el hardware usado, los componentes instalados en el mismo, las

conexiones físicas entre él y las relaciones entre componentes. Posteriormente se muestra el diagrama del sistema:

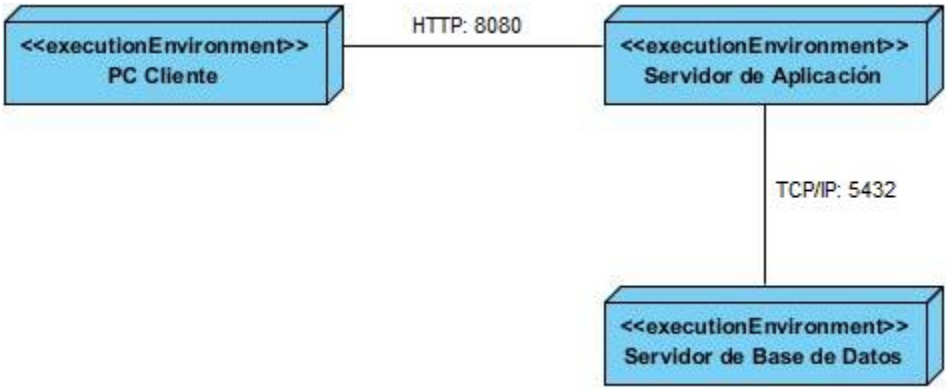


Ilustración 16: Diagrama de Despliegue del sistema.

PC Cliente: Es donde el usuario interactúa y usa la aplicación. El cliente es un ordenador que consume un servicio remoto en otro ordenador conocido como servidor normalmente a través de una red.

Servidor de Aplicaciones: Es donde se encuentra montado el servidor de la aplicación o sistema. Este servidor de aplicaciones generalmente gestiona la mayor parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación.

Servidor de Base de Datos: Es el nodo donde se encuentra el servidor de base de datos (BD) con toda la información persistente del sistema.

HTTP: Es un protocolo orientado a transacciones y sigue el esquema petición - respuesta entre un cliente y un servidor. Este protocolo emplea el puerto 8080 y su comienzo es ¨http://¨.

TCP/IP: Este protocolo es uno de los más importantes en Internet. TCP proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del puerto 5432 (usado por defecto por PostgreeSQL).

3.5 Pruebas

Las pruebas de software (también conocido como *testing*) son los procesos que permiten verificar y revelar la calidad de un producto. Ellas son empleadas para identificar cualquier fallo de implementación, calidad o usabilidad de un programa, por lo que se puede decir que una prueba ha sido exitosa si se ha encontrado un error o fallo que hasta ese período no se conocía, haciéndose efectiva la cita de Edsger Dijkstra[11]: "El *testing* puede probar la presencia de errores pero no la ausencia de ellos."

3.5.1 Diseño de Casos de Prueba (DCP)

Los Casos de Prueba (CP) son diseñados en dependencia de los distintos tipos de pruebas a utilizar. Cada CP va acompañado del resultado que ha de producir el software al ejecutarlo para detectar un posible fallo en el mismo. Se definen un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada método de prueba proporciona distintos criterios para generar estos casos o datos de prueba.

Para el desarrollo de estas pruebas existen dos métodos fundamentales: caja blanca y caja negra, las últimas se llevan a cabo sobre la interfaz del software. Su objetivo fundamental es demostrar que las funcionalidades son operativas, que las entradas se aceptan de forma adecuada y se produce el resultado correcto. La técnica aplicada fue la de partición de equivalencia para examinar los valores válidos e inválidos de las entradas existentes en la aplicación.

Para las pruebas realizadas a la propuesta se definen como posibles clasificaciones a las entradas a los campos como: válido (V), inválido (I) y no aplica (NA), teniendo en cuenta las variables definidas para el presente caso de prueba (CP). A continuación, se muestra el CP para el CU Gestionar Mecanismo:

[11] **Edsger Wybe Dijkstra,** Científico de la computación de origen holandés y Premio Turing en 1972.

Escenario	Descripción	Número	Nombre	Org. Arquitectónica	Referencia Cruzada	Descripción	Respuesta del Sistema	Flujo Central
EC 1.1 Entrada de datos válidos.	Se deberán introducir los campos correctos para crear un mecanismo exitoso.	V	V	V	V	V	Se guarda el mecanismo y se pasa a la vista Listar Mecanismos.	Se selecciona del menú lateral la opción “Mecanismo” y después en el área de trabajo el botón “Crear Mecanismo". El sistema muestra una vista donde debe llenar los campos y presionar el botón Crear.
		12 (Números enteros)	Detener enemigos (Números, Letras y Caracteres especiales)	Mecanismos alternos (Seleccionable)	Documento Diseño del Videojuego (Números, Letras y Caracteres especiales)	Objetos: Especies invasoras, casa nativa torre (Números, Letras y Caracteres especiales)		
EC 1.2 Ausencia de campos llenos de datos válidos.	Se deberá mantener el campo número o nombre de forma vacía.	I	V	V	V	V	Se mantiene abierta la vista Crear Mecanismo, se muestra un mensaje de error y no se crea el mecanismo.	Se selecciona del menú lateral la opción “Mecanismo” y después en el área de trabajo el botón “Crear Mecanismo". El sistema muestra una vista donde debe llenar los campos y presionar el botón Crear.
		Nulo	Detener enemigos (Números, Letras y Caracteres especiales)	Mecanismos alternos (Seleccionable)	Documento Diseño del Videojuego (Números, Letras y Caracteres especiales)	Objetos: Especies invasoras, casa nativa torre (Números, Letras y Caracteres especiales)		
		V	I	V	V	V		

		12 (Números enteros)	Nulo	Mecanismos alternos (Seleccionable)	Documento Diseño del Videojuego (Números, Letras y Caracteres especiales)	Objetos: Especies invasoras, casa nativa torre (Números, Letras y Caracteres especiales)		
EC 1.3 Entrada de datos inválidos.	Se deberá introducir el campo número o nombre con caracteres no válidos.	I	V	V	V	V	El sistema muestra mensajes de error y no se crea el mecanismo.	Se selecciona del menú lateral la opción “Mecanismo” y después en el área de trabajo el botón “Crear Mecanismo". El sistema muestra una vista donde debe llenar los campos y presionar el botón Crear.
		mecan /* \ (No acepta letras ni caracteres especiales)	Detener enemigos (Números, Letras y Caracteres especiales)	Mecanismos alternos (Seleccionable)	Documento Diseño del Videojuego (Números, Letras y Caracteres especiales)	Objetos: Especies invasoras, casa nativa torre (Números, Letras y Caracteres especiales)		
		V	I	V	V	V		
		12 (Números enteros)	23 * abc	Mecanismos alternos (Seleccionable)	Documento Diseño del Videojuego (Números, Letras y Caracteres especiales)	Objetos: Especies invasoras, casa nativa torre (Números, Letras y Caracteres especiales)		

EC 1.4 Cancelar la acción.	Se cancela la acción de crear mecanismo.	N/A	N/A	N/A	N/A	N/A	Se cierra la vista Crear Mecanismo y no se crea el mismo.	Se selecciona del menú lateral la opción “Mecanismo” y después en el área de trabajo el botón “Crear Mecanismo". El sistema muestra una vista donde debe llenar los campos y presionar nuevamente la opción “Mecanismo” por el menú lateral.
		No aplica	No aplica	No aplica	No aplica	No aplica		

Ilustración 17: Caso de Prueba del CU Crear Mecanismo.

No	Nombre de Campo	Clasificación	Valor Nulo	Descripción
1	Número	Campo de texto	No	El campo sólo acepta números enteros.
2	Nombre	Campo de texto	No	El campo acepta letras, números y caracteres especiales.
3	Org. Arquitectónica	Campo de selección	No	Campo para seleccionar la organización arquitectónica del mecanismo.
4	Referencia Cruzada	Campo de texto	Si	El campo acepta letras, números y caracteres especiales.
5	Descripción	Campo de texto	Si	El campo acepta letras, números y caracteres especiales.
6	Crear	Botón	No	Vinculo para crear el mecanismo.

Ilustración 18: Variables definidas para el CP del CU Crear Mecanismo.

3.4.2 Pruebas de Rendimiento

Las pruebas de rendimiento se realizan con el objetivo de determinar la rapidez con que se realiza una tarea en un sistema en condiciones particulares de trabajo. Además, sirven para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos.

A continuación, se evidencian las pruebas realizadas con la herramienta JMeter al sistema, en el cual se creó un grupo de 30 usuarios obteniendo un error de 0%.

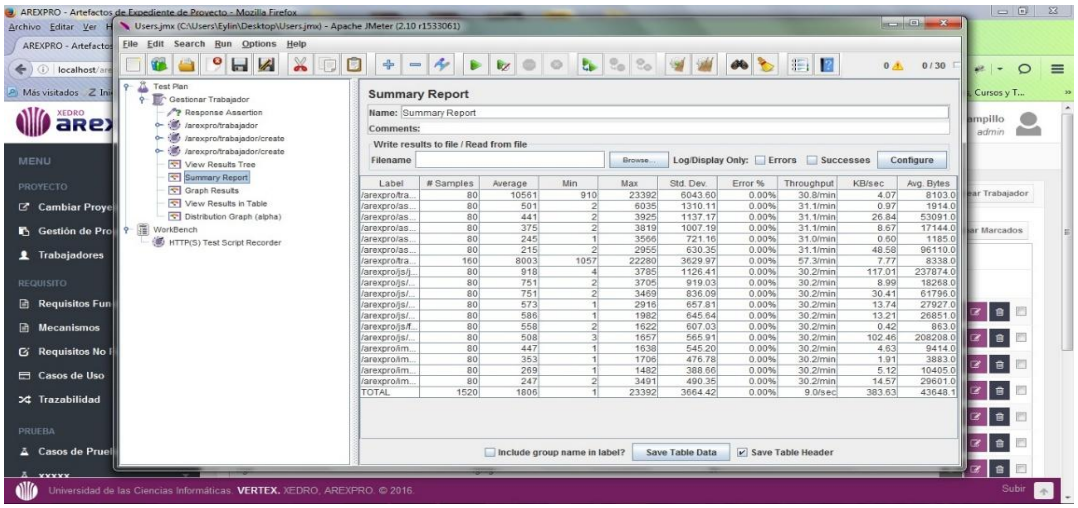


Ilustración 192: Prueba de rendimiento en JMeter (*Summary Report*).

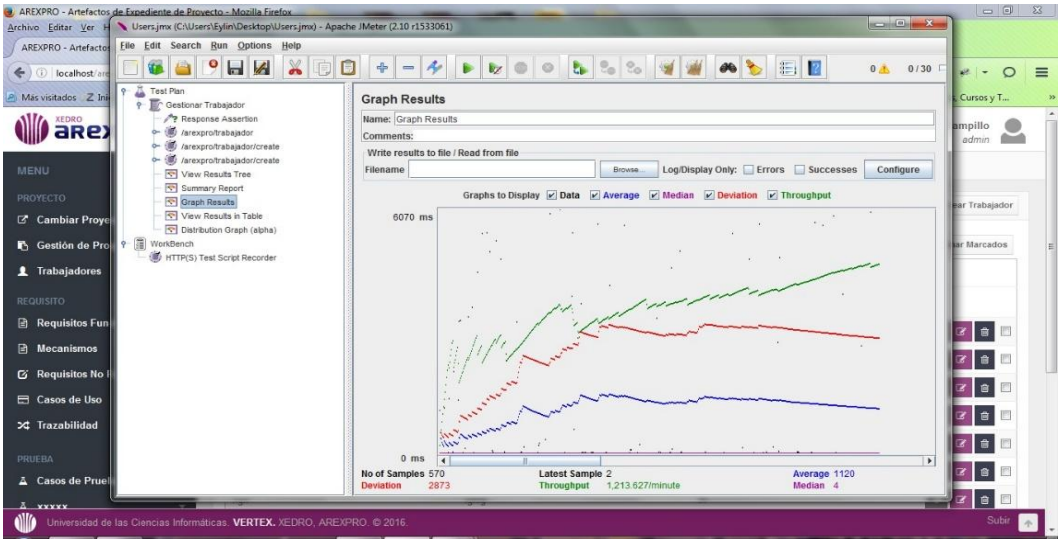


Ilustración 203: Prueba de rendimiento en JMeter (*Graph Results*).

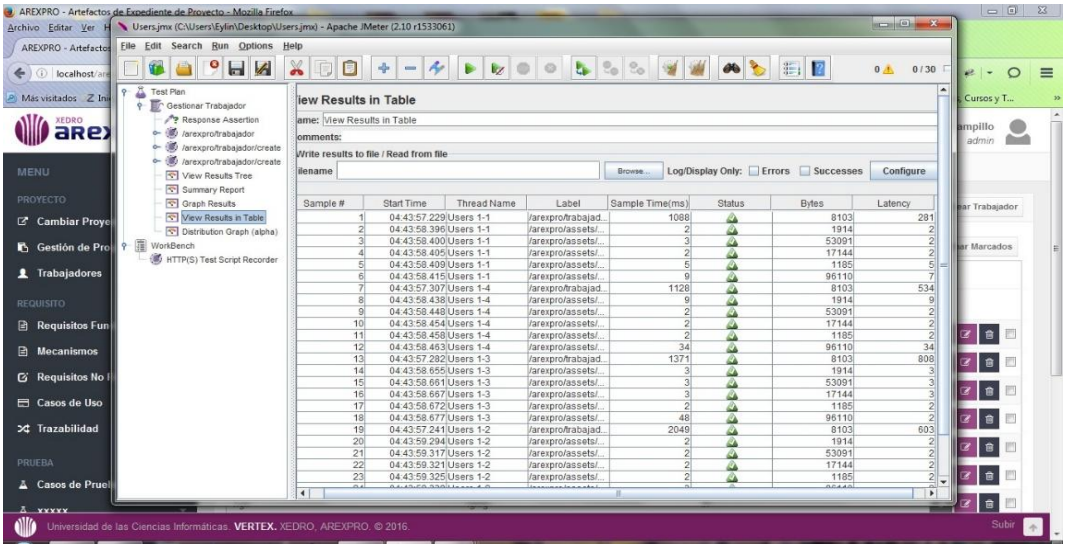


Ilustración 214: Prueba de rendimiento en JMeter (*View Results in Table*).

3.4.3 Validación de la propuesta

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Una vez concluida la etapa de desarrollo se le realizan una serie de pruebas para justificar su uso.

Por otro lado, es necesario validar las ventajas que realmente tiene el sistema. El mejor instrumento para esta determinación es la llamada prueba de aceptación. Las pruebas de aceptación de usuario validan el grado de satisfacción del cliente final, quien debe informar de todas las deficiencias o errores que encuentre antes de dar por aprobado el sistema definitivamente. Para la preparación, ejecución y evaluación de estas pruebas no se requiere de conocimientos informáticos. Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie bien planificada. Estas pruebas tienen vital importancia, pues los resultados proyectados deciden la aceptación o el rechazo del producto y definen el paso a nuevas fases como el despliegue y mantenimiento.

Para dar cumplimiento a esta prueba se realizaron encuestas a trabajadores (Saylin Salas, Yeili Ibarra, Lisleidy Mier, Lizandra Hernández, Linet Katiuska Remón y Sandy Guerra) de los centros de desarrollo Vertex, Entornos Interactivos 3D y Fortes donde se les facilitó la aplicación y se les explicó el funcionamiento de la misma. De una muestra

de 7 trabajadores, 3 fueron evaluados de aceptado, 2 de medio y solo 1 fue rechazado de acuerdo a las respuestas de las preguntas realizadas (ver encuesta en anexo 3.4.3.1). Estableciendo un análisis de los resultados, se considera que es factible la puesta en práctica del sistema informático de apoyo a la reutilización de productos de trabajo de la disciplina de requisitos.

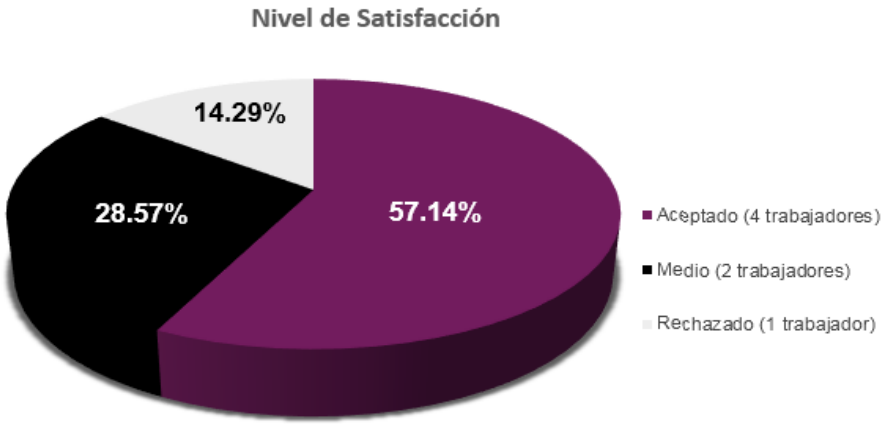


Ilustración 22: Nivel de Satisfacción.

3.5 Conclusiones parciales

Después definir las características de implementación y prueba de la solución se llegan a las siguientes conclusiones:

- Se explicó el estándar de codificación, declaración de métodos, variables y estructuras de control con el objetivo de mantener uniformidad en el código final y que futuros desarrolladores lo entiendan y puedan trabajar con él.
- Se realizaron los diagramas de componentes y de despliegue para que los que interactúen con el sistema sepan su estructura y las condiciones que debe cumplir para su ejecución.
- Se obtuvo una respuesta positiva luego de aplicadas las pruebas de aceptación y rendimiento en sus distintas fases.

Conclusiones Generales

En el presente trabajo de diploma se presentó la investigación y el posterior proceso de desarrollo del sistema AREXPRO y se arriba a las siguientes conclusiones:

- Con el estudio y la caracterización de las herramientas de gestión de requisitos y la definición del marco de trabajo para el proceso de desarrollo de software, fueron estudiadas las funcionalidades a incluir en la propuesta de solución, así como el proceso de generación de matrices de trazabilidad bidireccional.
- Mediante la implementación de una aplicación capaz de gestionar la disciplina de requisitos y generar reportes de los quince (criterios para validar requisitos del cliente y del producto, registro de proveedores de requisitos, evaluación de requisitos y casos de uso del sistema, especificación de requisitos de software y casos de uso del sistema, requisitos rechazados, reporte de trazabilidad, modelo conceptual, modelo de diseño, glosario de términos, salidas del sistema, diseño del videojuego y especificación de mecanismos) artefactos concernientes a la disciplina con toda la información necesaria y en el estilo deseado, se dio cumplimiento a la investigación presentada.
- El proceso de desarrollo guiado por la metodología AUP e implementado con el framework Yii de conjunto con PostgreSQL, facilitaron el desarrollo de la plataforma en el tiempo y la versión (v 4.0) deseada del EPD.

Recomendaciones

- Continuar con el estudio de las herramientas para la gestión de la información de productos de trabajos de la IR, con el objetivo de encontrar mejoras e incluirlas en futuras versiones.
- Incluir ya sea por definición o por integración, los elementos correspondientes a otras fases del EPD como son las pruebas.
- Incluir un módulo de trazas capaz de autogenerar el control de cambios de los artefactos del EPD y muestre estadísticas del comportamiento de variables asociadas a la información gestionada en el sistema a petición del usuario.

Referencias Bibliográficas

1. **Smith, Aaron.** *Gestión de Requisitos. Una competencia esencial para el éxito de proyectos y programas.* s.l. : Instituto de Administración de Proyectos, 2014.

2. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Sexta Edición : McGrawHill, 2007.

3. **Bucero, Alfonso.** AEDIP - Asociación Española de Dirección Integrada de Proyecto. *Project Management Asociation Asociacion Española de Direccion Integrada de Proyecto.* [En línea] [Citado el: 26 de octubre de 2015.] http://www.aedip.org/formacion/opinion-profesor/bucero.asp.

4. **Implementación y Debugging. *Ciclo de Vida del Software, Capítulo 1.* s.l. : Servicio de Atención al Lector.**

5. Grady Booch, Jim Rumbaugh e Ivar Jacobson. UML El Lenguaje Unificado de Modelado. [aut. libro] Fernando Berzal. *OOP - Introducción: Java .*

6. Guaita, Alvaro Martinez. Cherokee Web Server, el más rápido de los servidores web. *desarrollowe.com.* [En línea] [Citado el: 20 de 12 de 2015.] http://www.desarrolloweb.com/de_interes/.

7. Rubén Gómez López. PHP vs JAVA. *Adictos al trabajo. .* [En línea] [Citado el: 26 de octubre de 2015.] http://www.adictosaltrabajo.com/tutoriales/php-vs-java.

8. Martínez., Daniel Pecos. PostGreSQL vs MySQL. [En línea] 2005. [Citado el: 10 de noviembre de 2015.] https://danielpecos.com/documents/postgresql-vs-mysql/.

9. Sommerville, Ian. *Ingeniería del Software, Séptima Edición.* Madrid, España : Pearson Educación, 2005. 84-7829-074-5.

10. Ceria, Santiago. *Ingeniería de Software I, Caso de Uso - Un Método Práctico para Explorar Requerimientos.* Buenos Aires, Argentina : Cátedra de Ingeniería del Software I, Universidad de Buenos Aires.

11. Patrones de Diseño. [En línea] [Citado el: 26 de Febrero de 2016.] http://es.ccm.net/contents/224-patrones-de-diseno.

12. StudyLib. *Guía para la Arquitectura de Software.* [En línea] v1.0, 30 de Septiembre de 2008. [Citado el: 20 de Febrero de 2016.] http://studylib.es/doc/209467/gu%C3%ADa-para-la-arquitectura-de-software.

13. Tutorial de UML - Diagrama de Interacción. [En línea] [Citado el: 17 de febrero de 2016.] http://users.dcc.uchile.cl/~psalinas/uml/interaccion.html. 20160217022727.

14. Definición.de. *Definición de videojuego - Qué es, Significado y Concepto.* [En línea] [Citado el: 26 de octubre de 2015.] http://definicion.de/videojuego/.

15. *REASEM: Herramienta para la gestión de requisitos.* Jorge Eduardo Gómez Maldonado, German Urrego Giraldo, Liliana González Palacio. 2, Medellín : Revista Avances en Sistemas e Informática, 2009, Vol. 6. 1 657-7663.

16. *Heler: Una herramienta para la ingeniería de requisitos automatizada.* Mauro Callejas Cuervo, Luz Yadira Castillo Estupiñan, Ruby Mónica Fernández Álvarez. 2, Cali, Colombia : Sistemas de Computación, 2010, Vol. 6. 184-200.

17. Andrea Alarcón, Erika Sandoval. *Herramientas CASE para ingeniería de Requisitos.* s.l. : Cultura Científica, 2008.

18. *Aplicando el método de Boehm y Turner. Serie Científica de la Universidad de las Ciencias Informáticas.* Mairelys Boeras, Laritza Cabrera, Yaima Oval, Eylin Hernández, Ana María Sánchez, Eileén Llano. 6, La Habana, Cuba : Grupo Editorial ¨Ediciones Futuro¨, 2012, Vol. 5.

19. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Penadés., Patricio Letelier y María Carmen. Valencia, España. : Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación. Facultad de Informática. Universidad Politécnica de Valencia. 46022.

20. *Construcción de un modelo de predicción para el entendimiento de los diagramas de estados en UML. .* Marcela Genero, José Antonio Cruz-Lemus y Mario Piattini. España. : Grupo ALARCOS.

21. Ortega, Alvaro Lopez. Evolved Web Infrastructure Software. *Cherokee Web Server | Home.* [En línea] 2013. [Citado el: 13 de 12 de 2015.] http://cherokee-project.com/.

22. Digital Learning. *¿Qué hace un Servidor Web como Apache?. Configuración.* [En línea] 17 de marzo de 2012. [Citado el: 10 de enero de 2016.] http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/.

23. Electrónicos, Instituto de Ingenieros Eléctricos y. *Glosario de Terminología Estándar de Ingeniería de Software (IEEE: Standard Glossary of Software*

Engineering Terminology). **[IEEE Computer Dictionary] s.l. : Software Engineering Terms, 1990. IEEE 610-1990.**

24. Autores, Colectivo de. *Preparación Pedagógica Integral para Profesores Universitarios.* 2003. pág. 310. 9592585300.

25. Pérez, Antonio Blanco. *Componentes Didácticos del Proceso de Enseñanza-Aprendizaje.* 2007.

26. *ER/Studio.*

27. Sánchez, Tamara Rodríguez. *Programa de Mejora. Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : Universidad de las Ciencias Informáticas, 2014.

28. Gutierrez, Pedro. Frameworks, Desarrollo Web, PHP. *6 frameworks PHP más que te harán la vida más simple.* [En línea] 11 de junio de 2014. [Citado el: 20 de mayo de 2016.] http://www.genbetadev.com/frameworks/y-6-frameworks-php-mas-que-te-haran-la-vida-mas-simple.

29. Visure Solutions expertos en herramientas de gestión de requisitos. *Visure Requirements.* [En línea] [Citado el: 29 de 5 de 2016.] http://www.visuresolutions.es.

30. Guiadev.com - Tutoriales y Guías para Desarrolladores. *¿Qué framework PHP me conviene utilizar?* [En línea] 2014. [Citado el: 1 de junio de 2016.] https://guiadev.com/que-framework-php-me-conviene-utilizar/.

31. OpenWebinars. *Los 10 Frameworks PHP que solicitan las empresas.* [En línea] 28 de septiembre de 2015. [Citado el: 18 de mayo de 2016.] https://openwebinars.net/los-10-mejores-frameworks-php-que-solicitan-las-empresas/.

32. Rosique, Maria Francisca. *Evaluación de herramientas de gestión de requisitos.* España : Universidad Politécnica de Cartagena, 2014.

33. Pérez, Lisbán Torres. *Entorno de ingeniería de requisitos aplicado para producir software en una universidad.* La Habana, Cuba : s.n., 2014. ISSN 1815-5936.

34. Tutoriales de Programacion Java: Creación de Reportes con JasperRepots y iReports. *Parte 1: Reportes con Conexión a Base de Datos.* [En línea] 2014. [Citado el: 27 de abril de 2016.] http://www.javatutoriales.com/2009/02/creacion-de-reportes-con-jasperrepots-y.html.

35. Roberto J. Furutani. Mini-Tutorial. *Como criar relatórios Java para Web com JasperReports e iReport.* [En línea] 18 de mayo de 2016. [Citado el: 20 de julio de 2005.] http://www.furutani.eti.br.

36. *LABORATORIO 6. Manual Apache Jmeter. Establecer los niveles de servicio de acuerdo con estandares y requerimientos de la organización. .* s.l. : FAVA - Formación en Ambientes Virtuales de Aprendizaje, SENA - Servicio Nacional de Aprendizaje.

37. Cecilio Alvarez Caules. Arquitectura Java. *Introducción a JMeter y pruebas de carga - Arquitectura Java.* [En línea] 24 de abril de 2014. [Citado el: 20 de abril de 2016.] http://www.arquitecturajava.com/introduccion-jmeter-y-pruebas-de-carga/.

38. F. Javier Diaz. *Usando Jmeter para pruebas de rendimiento.* La Plata, Argentina : Fac. de Informática, Universidad Nacional de La Plata.

39. Garcia, Carlos Barreiro. *Usando IReport con JavaBeans.*

40. Javier Molina. JMeter pruebas de carga y estrés para tus aplicaciones Flex. [En línea] 24 de septiembre de 2010. [Citado el: 20 de mayo de 2016.] http://nosmoke.cycle-it.com/2010/09/24/jmeter-pruebas-de-carga-y-estres-para-tus-aplicaciones-flex/.

41. Pruebas de rendimiento con Apache JMeter. [En línea] 14 de diciembre de 2010. [Citado el: 20 de mayo de 2016.] http://www.aquihaydominios.com/blog/pruebas-de-rendimiento-con-apache-jmeter/.

42. tutorialspoint.com. JMeter Web Test Plan. [En línea] 2016. [Citado el: 15 de mayo de 2016.] http://www.tutorialspoint.com/jmeter/jmeter_web_test_plan.htm.