Universidad de las Ciencias Informáticas

Facultad 4

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

Sistema de Gestión para el cumplimiento de los objetivos estratégicos de la Facultad 4

Autor: Adrián Céspedes Calzado

Tutor(es): Ing. Manuel Alejandro Oliva Labaut

MSc. Graciela Falcón Pi

La Habana, junio de 2019

"Año 61 de la Revolución"

"...no basta saber, se debe también aplicar. No es suficiente querer, se debe también hacer".

Johann Wolfgang von Goethe

Declaración de autoría

Declaramos ser los autores del presente trabajo de diploma y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Tutora
Adrián Céspedes Calzado	MSc. Graciela Falcón Pi

_____

Firma del Tutor

Ing. Manuel A. Oliva Labaut

Dedicatoria

Esta tesis va dedicada a mis padres por haberme forjado como la persona que hoy soy, por todo su amor, preocupación, por su entendimiento y creer en que lo podría lograr.

A mi hermano por ayudarme y ser un ejemplo de lucha.

A mi hermana que a pesar de que somos polos opuestos en ciertas cuestiones tu preocupación valió de mucho.

A mi novia por tanta preocupación, dedicación y entrega.

Agradecimientos

Primero que todo agradecer a mis padres, quiero que sepan que ustedes son los mejores del mundo, gracias a ustedes por su paciencia amor y cariño, por que se que en un momento de mi vida me caí y ustedes estuvieron ahí a pesar de las circunstancias, pero ya ven este es el mejor regalo que les puedo hacer y todo lo que soy se lo debo a ustedes.

A mi hermana y mi hermano por tanta preocupación, por su ayuda incondicional, y ser una gran persona, gracias.

A mi novia por siempre estar ahí, por tanta preocupación, de verdad gracias por tu paciencia, cariño y amor.

A mis amigos quiero que sepan que ustedes con sus imperfecciones hacían estos cinco años perfectos, gracias por cada chiste por cada sonrisa, ustedes son los mejores.

A todos los que me falto por mencionar y estuvieron en esta trayectoria, en general a todos porque cada momento compartido ayudo a formar la persona que hoy soy.

Resumen

La eficiente y eficaz gestión de organizaciones, es un proceso que tiene como finalidad la optimización de los recursos y la maximización de los beneficios sean estos sociales, económicos y/o medio ambientales. Para alcanzar esas metas han surgido diferentes técnicas o herramientas que constituyen el cuerpo académico-científico de la gestión empresarial. La administración por objetivos constituye una de estas técnicas y permite la evaluación, medición y seguimiento de los resultados del trabajo en las organizaciones. La Universidad de las Ciencias Informáticas en su misión de formar profesionales altamente calificados y producir aplicaciones y servicios informáticos, define anualmente sus objetivos estratégicos agrupados en áreas de resultados claves, para cada una de las facultades que componen su estructura organizacional. El objetivo de este trabajo es desarrollar un sistema de gestión que contribuya al cumplimiento de los objetivos estratégicos de la Facultad 4 adjunta al mencionado centro de altos estudios. El sistema informático obtenido como resultado de este trabajo permite planificar, controlar y evaluar el cumplimiento de los mencionados objetivos de trabajo y está basado totalmente en tecnologías de software libre. Otra contribución importante es el apoyo a la toma de decisiones, pues permite identificar a tiempo posibles atrasos en el cumplimiento de los mismos.

Palabras clave: Sistema de gestión, administración por objetivos, objetivos estratégicos.

Índice de Contenidos

Introducción 11
Capítulo 1 Fundamentación Teórica 16
1.1 Conceptos fundamentales 16
1.2 Análisis de sistemas homólogos 17
1.2.1 Resumen del estudio de los sistemas 20
1.3 Tecnologías y herramientas 22
1.3.1 Front-End 22
1.3.2 Back-End 27
1.4 Entorno de Desarrollo Integrado 37
1.5 Herramientas Case 37
1.5.1 Visual Paradigm 38
1.5.2 Lenguaje Unificado de Modelado 38
1.6 Metodología de desarrollo de software 39
1.6.1 Variación de AUP para la UCI 41
1.7 Descripción de las fases 41
1.7.1 Escenarios para la disciplina Requisitos 42
1.8 Conclusiones parciales 43
Capítulo 2. Análisis y Diseño de la propuesta de solución 45
2.1 Descripción de la propuesta de solución 45
2.2 Requisitos de software 46
2.2.1 Requisitos funcionales 46
2.2.2 Requisitos no funcionales 48
2.3 Historias de Usuario 49
2.4 Plan de iteraciones 52
2.5 Descripción del Patrón arquitectónico 53
2.5.1 Elementos del patrón MVVM 53
2.6 Patrones de diseño 55
2.6.1 Los patrones GRASP 55
2.6.2 Los patrones GOF 56
2.7 Diseño de la base de datos 57
2.8 Conclusiones parciales 57
Capítulo 3 Implementación y pruebas 59
3.1 Diagrama de despliegue 59

3.2 Diagrama de Componentes 60
3.3 Pruebas del sistema 60
3.3.1 Pruebas Funcionales 60
3.3.2 Resultados de las pruebas funcionales 62
3.3.3 Pruebas Unitarias 63
3.4 Pruebas de carga y estrés 67
3.5 Conclusiones Parciales 68
Conclusiones generales 70
Referencia Bibliográfica 72
Anexos 77
Anexo 1: Historias de usuario 77
Anexo 2: Casos de prueba 103
Anexo 3: Diagrama de componentes 124

Índice de Figuras

Figura 1 Modelo conceptual del negocio [Fuente: Elaboración Propia] 45
Figura 2. Arquitectura del sistema [Fuente: Elaboración propia] 54
Figura 3. Diagrama entidad relación. [Fuente: Elaboración propia] 57
Figura 4. Diagrama de despliegue. [Fuente: Elaboración propia] 59
Figura 5. Método modicarCampoCriteriodemedida($id) [Fuente: Elaboración propia]..... 64
Figura 6. Grafo del flujo asociado al método modificar criterio de medida [Fuente: Elaboración propia] 65
Figura 7. Reporte de resumen [Fuente: Herramienta JMeter] 68
Figura 8. Diagrama de componentes [Fuente: Elaboración propia] 124

Índice de Tablas

Tabla 1. Angular [Fuente: Elaboración propia] 24
Tabla 2. Vue [Fuente: Elaboración propia] 25
Tabla 3. PHP (Hypertext Pre-processor) [Fuente: Elaboración propia] 27
Tabla 4. JAVA [Fuente: Elaboración propia] 28
Tabla 5. Laravel [Fuente: Elaboración propia] 30
Tabla 6. Symfony [Fuente: Elaboración propia] 31
Tabla 7. PostgreSQL [Fuente: Elaboración propia] 33
Tabla 8. MySQL [Fuente: Elaboración propia] 34
Tabla 9. Fases de AUP y AUP-UCI [Fuente: Documento Oficial] 41
Tabla 10. Historia de usuario Insertar Áreas [Fuente: Elaboración Propia] 50
Tabla 11. Historia de Usuario listar Áreas [Fuente: Elaboración Propia] 51
Tabla 12. Plan de iteraciones. [Fuente: Elaboración propia] 52
Tabla 13. Caso de prueba: Visualizar estado del cumplimiento de los objetivos por ARC en por cientos [Fuente: Elaboración Propia] 61
Tabla 14. Camino básico del flujo asociado al método modificar campo estado criterio de medida [Fuente: Elaboración Propia] 66
Tabla 15. Caso de prueba unitaria [Fuente: Elaboración Propia] 66
Tabla 16. Caso de prueba unitaria [Fuente: Elaboración Propia] 66
Tabla 17. Caso de prueba unitaria [Fuente: Elaboración Propia] 67
Tabla 18. Caso de prueba unitaria [Fuente: Elaboración Propia] 67
Tabla 19. Historia de Usuario Autenticar Usuario. [Fuente: Elaboración Propia] 77
Tabla 20. Historia de Usuario Insertar Objetivo. [Fuente: Elaboración Propia] 78
Tabla 21. Historia de Usuario Eliminar Objetivo. [Fuente: Elaboración Propia] 79
Tabla 22. Historia de Usuario Editar Objetivo. [Fuente: Elaboración Propia] 80
Tabla 23. Historia de Usuario Listar Objetivo. [Fuente: Elaboración Propia] 81
Tabla 24. Historia de Usuario Eliminar Área. [Fuente: Elaboración Propia] 82
Tabla 25. Historia de Usuario Editar Área. [Fuente: Elaboración Propia] 83
Tabla 26. Historia de Usuario Insertar criterios de medidas. [Fuente: Elaboración Propia] 84
Tabla 27. Historia de Usuario Listar criterios de medidas. [Fuente: Elaboración Propia] 85
Tabla 28. Historia de Usuario Eliminar criterios de medidas. [Fuente: Elaboración Propia] 86
Tabla 29. Historia de Usuario Listar criterios de medidas. [Fuente: Elaboración Propia] 87

Tabla 30. Historia de Usuario Crear actividades. [Fuente: Elaboración Propia]..... 88
Tabla 31. Historia de Usuario Crear actividades. [Fuente: Elaboración Propia]..... 89
Tabla 32. Historia de Usuario Editar actividades. [Fuente: Elaboración Propia] 90
Tabla 33. Historia de Usuario Listar actividades. [Fuente: Elaboración Propia]..... 91
Tabla 34. Historia de Usuario Visualizar cumplimiento de objetivos. [Fuente: Elaboración Propia] 92
Tabla 35. Historia de Usuario Filtrar objetivos por estado de cumplimiento. [Fuente: Elaboración Propia] 93
Tabla 36. Historia de Usuario Filtrar objetivos por ARC. [Fuente: Elaboración Propia]..... 94
Tabla 37. Historia de Usuario Asignar roles. [Fuente: Elaboración Propia] 95
Tabla 38. Historia de Usuario Insertar usuario. [Fuente: Elaboración Propia] 96
Tabla 39. Historia de Usuario Listar usuario. [Fuente: Elaboración Propia]..... 97
Tabla 40. Historia de Usuario Editar usuario. [Fuente: Elaboración Propia] 98
Tabla 41. Historia de Usuario Eliminar usuario. [Fuente: Elaboración Propia]..... 99
Tabla 42. Historia de Usuario Generar informe. [Fuente: Elaboración Propia] 100
Tabla 43. Historia de Usuario Subir evidencia. [Fuente: Elaboración Propia]..... 101
Tabla 44. Historia de Usuario Descargar evidencia. [Fuente: Elaboración Propia]..... 102
Tabla 45. Caso de prueba: Insertar Usuario [Fuente: Elaboración Propia]..... 103
Tabla 46. Caso de prueba: Listar Usuario [Fuente: Elaboración Propia] 104
Tabla 47. Caso de prueba: Editar Usuario [Fuente: Elaboración Propia]..... 104
Tabla 48. Caso de prueba: Eliminar Usuario [Fuente: Elaboración Propia]..... 105
Tabla 49. Caso de prueba: Autenticar Usuario [Fuente: Elaboración Propia]..... 106
Tabla 50. Descripción de las variables [Fuente: Elaboración Propia]..... 106
Tabla 51. Caso de prueba: Asignar rol [Fuente: Elaboración Propia]..... 107
Tabla 52. Caso de prueba: Insertar ARC [Fuente: Elaboración Propia] 108
Tabla 53. Caso de prueba: Eliminar ARC [Fuente: Elaboración Propia]..... 109
Tabla 54. Caso de prueba: Listar ARC [Fuente: Elaboración Propia]..... 109
Tabla 55. Caso de prueba: Editar ARC [Fuente: Elaboración Propia] 109
Tabla 56. Caso de prueba: Insertar objetivos [Fuente: Elaboración Propia]..... 110
Tabla 57. Caso de prueba: Listar objetivos [Fuente: Elaboración Propia] 111
Tabla 58. Caso de prueba: Editar objetivo [Fuente: Elaboración Propia]..... 111
Tabla 59. Caso de prueba: Eliminar objetivo [Fuente: Elaboración Propia]..... 112
Tabla 60. Caso de prueba: Insertar criterios de medida [Fuente: Elaboración Propia] ... 113
Tabla 61. Caso de prueba: Editar criterios de medida [Fuente: Elaboración Propia]..... 114

Tabla 62. Caso de prueba: Eliminar criterios de medida [Fuente: Elaboración Propia] .. 115
Tabla 63. Caso de prueba: Listar criterios de medida [Fuente: Elaboración Propia] 115
Tabla 64. Caso de prueba: Insertar actividades [Fuente: Elaboración Propia] 116
Tabla 65. Caso de prueba: Listar actividades [Fuente: Elaboración Propia]..... 118
Tabla 66. Caso de prueba: Editar actividades [Fuente: Elaboración Propia] 119
Tabla 67. Caso de prueba: Eliminar actividades [Fuente: Elaboración Propia] 120
Tabla 68. Caso de prueba: Subir evidencia [Fuente: Elaboración Propia]..... 120
Tabla 69. Caso de prueba: Descargar evidencia [Fuente: Elaboración Propia]..... 120
Tabla 70. Caso de prueba: Filtrar Objetivos [Fuente: Elaboración Propia] 121
Tabla 71. Caso de prueba: Filtrar objetivos por ARC [Fuente: Elaboración Propia] 123
Tabla 72. Caso de prueba: Generar informe de objetivos de trabajo [Fuente: Elaboración Propia] 123

Introducción

El desarrollo de las Tecnologías ha transformado las formas de dirigir las organizaciones. El avance de las tecnologías de la información y comunicación (TIC) ha permitido que los negocios tradicionales puedan convertirse en negocios digitales mediante la informatización de sus procesos a escala nacional e internacional.

Hoy en día muchas organizaciones trabajan con una cantidad de información en continuo crecimiento y renovación, por lo que la gestión de sus procesos cada vez se hace más engorroso. La informatización de procesos es un aspecto que brinda soluciones automatizadas a flujos de trabajos manuales por lo que debería de ser tenido en cuenta por las organizaciones cubanas. Este factor no solo es necesario para cumplir con las normas que regulan cada sector, sino que permite lograr una mayor productividad y competitividad; además de ayudar en las tareas gestión por su aporte de información clave para la toma de decisiones estratégicas.

En la actualidad existen muchas técnicas de gestión empresarial, los cuales ayudan al mejoramiento de procesos dentro de las empresas. Una de estas técnicas son los sistemas de apoyo a la toma de decisiones, los cuales constituyen una forma de modelar y decidir, basadas en datos, el destino de la organización. Tomar la decisión correcta en los negocios se suele basar en la calidad de sus datos y su capacidad para filtrar y analizar la información y encontrar las tendencias en las cuales se puede crear soluciones y estrategias.

Los sistemas informatizados también permiten abarcar los procesos dentro de una organización ya que los aseguran, optimizan y realizan una mejor toma de decisiones, analizando la información de forma efectiva logrando un mayor control de los procesos y tareas. Por ello, es de suma importancia para las organizaciones sobrecargadas de tareas y siempre en busca de vías más eficientes, inteligentes y seguras la informatización de sus procesos.

Es válido resaltar que en este mundo cada vez más competitivo, el desarrollo de las organizaciones depende también del uso de sistemas de gestión (SG), los cuales son herramientas que tienen como finalidad la mejor y más eficiente asignación de los recursos con el objetivo de obtener los mejores resultados posibles.

Hoy en día existen diversos SG desde los más simples hasta los más sofisticados, estos últimos no solo manejan documentación, sino que permiten informatizar procesos internos

dentro de las entidades u organizaciones. Estos sistemas se deben aplicar en los procesos de planificación para complementar, mejorar el análisis y las decisiones con el fin de lograr los objetivos estratégicos trazados y a largo plazo la posición de la organización, es decir, son los resultados que se esperan alcanzar en un año.

Estos objetivos estratégicos son metas a corto y largo plazo por lo que son de gran importancia para evaluar el desempeño de la organización. La definición clara de objetivos son pasos a seguir realizando estrategias para enfrentar de la forma más conveniente los problemas que puedan surgir en un futuro, por lo que estos objetivos deben ser principalmente, claros, coherentes, medibles y alcanzables.

Los objetivos estratégicos se controlan mediante un proceso administrativo donde los trabajadores con un alto nivel jerárquico definen las decisiones correspondientes a la manera en que la organización funciona. Según lo planeado y organizado se podrá identificar problemas para corregirlos.

En la Universidad de las Ciencias Informáticas (UCI) en la Facultad 4, ubicada en el Docente 5 “José Antonio Echeverría”, se definen y aprueban los objetivos estratégicos del año en el Consejo de Dirección de la Facultad. Estos objetivos se aplican en las distintas áreas de trabajo de la organización y a distintos niveles. Deben ser planificados, realizados y controlados con el fin de conseguir los resultados que se proponen. El control del cumplimiento de los mismos se lleva mediante herramientas ofimáticas y a partir de la información extraída de otros sistemas con objetivos diferentes como el Sistema de Gestión Académica (AKADEMOS).

Cada área de trabajo está compuesta por profesores y especialistas, los cuales realizan una serie de actividades, tareas e investigaciones que se encuentran planificadas en el documento Objetivos de Trabajo de la Facultad 4. El mismo se encuentra estructurado en áreas de resultados clave (ARC) las cuales varían en función de la planeación estratégica de la Universidad para un año natural.

Una vez realizadas las actividades por cada área se levanta un documento a partir de los objetivos que fueron alcanzados. Este documento lo supervisa el jefe de cada área, el cual se entrega a la decana de la facultad. Una vez recibido el documento, se almacena en una computadora, por tanto, todo aquel que desee consultar o actualizar el cumplimiento de los objetivos no tiene acceso en cualquier momento al mismo.

La evaluación de los objetivos estratégicos por cada área requiere una previa consulta a las partes involucradas y estos se lleva a cabo a través de reuniones, despachos y en ocasiones a través de llamadas telefónicas. Los datos recopilados durante el proceso suelen estar duplicados al no estar centralizados, desactualizados y en ocasiones no brinda la información oportuna en el momento indicado. Esto conduce a que el control y el monitoreo continuo del proceso se dificulte, situación que propicia a que no se tomen las medidas correctivas a tiempo y abra las puertas al incumplimiento de algunos objetivos planeados.

A partir de la situación problemática descrita anteriormente se plantea como **problema de investigación** ¿Cómo planificar, monitorear, controlar y evaluar el cumplimiento de los objetivos estratégicos de la Facultad 4?

A partir del planteamiento del problema de investigación se tiene como **objeto de estudio** de la investigación: la administración por objetivos en las organizaciones; enmarcado en el **campo de acción**: La Gestión de los objetivos estratégicos de la Facultad 4.

El **objetivo general** de este trabajo de diploma es desarrollar un sistema de gestión que contribuya al cumplimiento de los objetivos estratégicos de la Facultad 4, empleando tecnologías y herramientas de software libre.

Para dar solución al objetivo planteado se desglosan las siguientes tareas de la investigación:

- Elaboración del marco teórico de la investigación mediante el estudio del estado actual de las herramientas existentes en el mercado, dirigidas a los sistemas de gestión por objetivo de organizaciones.
- Selección de las herramientas a utilizar para el desarrollo de la propuesta de solución.
- Realización de los artefactos ingenieriles a partir de la metodología de desarrollo de software seleccionada.
- Implementación del sistema propuesto que permita gestionar el cumplimiento de los objetivos estratégicos de la Facultad.
- Ejecución de pruebas para garantizar la calidad del software.

Durante la investigación se emplearon los siguientes métodos científicos:

Métodos teóricos:

Histórico-Lógico: se utilizó en el análisis de la base teórica, el estudio de las tecnologías a emplear y la descripción de los sistemas de gestión para elevar el control en las organizaciones.

Analítico-Sintético: con la utilización del método se han buscado, investigado y analizado documentos para extraer los elementos a nivel nacional e internacional, que están relacionados con el objeto de estudio; realizando consultas a diversas fuentes bibliográficas.

Modelación: se utilizó durante la elaboración del sistema, donde fue necesario explicarle al cliente mediante modelos cómo se previó la realización del sistema y si cumplía con sus necesidades. Este modelo constituye una representación basada en las Ciencias de la Computación de la situación real.

Métodos Empíricos:

Observación: se empleó para realizar el control y gestión de la información para de esa forma poder entender la situación existente, permitiendo adoptar nuevos conocimientos que llevaron a la solución del problema planteado.

Entrevista: se utilizó con el objetivo de entrevistar a los directivos y personal de la institución para así obtener información rápida y precisa de sus principales problemas, necesidades y posibles resultados esperados.

La estructura del trabajo de diploma está compuesta por tres capítulos, resumen, introducción, conclusiones, recomendaciones, referencias bibliográficas, anexos y un glosario de términos. A continuación, se describen los principales aspectos abordados en cada uno de los capítulos:

Capítulo 1: Fundamentación teórica.

En este capítulo se puntualizan los principales conceptos relacionados con el tema, se realiza el estudio del estado del arte a nivel nacional e internacional sobre los sistemas de gestión. Además, se definen las herramientas, metodología y tecnologías a utilizar durante el desarrollo del sistema.

Capítulo 2: Análisis y Diseño de la propuesta de solución

En este capítulo se expone una descripción general del sistema de gestión a desarrollar como propuesta de solución, se capturan los requisitos funcionales y no funcionales y otros

elementos importantes dentro del proceso de desarrollo, tales como: arquitectura, diseño del sistema, patrones empleados y los artefactos derivados de la metodología de desarrollo de software seleccionada.

Capítulo 3: Implementación y pruebas

En este capítulo se abordan aspectos relacionados con la implementación del sistema en base a la arquitectura de desarrollo de software. Se documentan las pruebas de software al sistema de gestión para el cumplimiento de los objetivos estratégicos de la Facultad 4, realizándose el nivel de prueba unitaria, pruebas funcionales y pruebas de carga y estrés para verificar que responda a un correcto funcionamiento, garantizar la confidencialidad y detectar fallas en el sistema.

Capítulo 1 Fundamentación Teórica

En este capítulo se exponen los elementos teóricos y conceptuales necesarios para la implementación de un sistema de gestión que permita el control de los objetivos estratégicos de la Facultad 4. Se realiza un estudio de sistemas homólogos para así establecer las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema de gestión lo que conlleva a identificar las principales fuentes bibliográficas y áreas del conocimiento que engloba al objeto de estudio y campo de acción.

1.1 Conceptos fundamentales

Para el desarrollo y mayor entendimiento de la presente investigación, es necesario puntualizar algunos conceptos fundamentales que son referidos al marco teórico:

Según Fúquene, la Gestión o Administración por objetivos es una técnica participativa de planeación y evaluación, a través de la cual, superiores y subordinados conjuntamente, definen aspectos prioritarios y establecen objetivos (resultados), a ser alcanzados en un determinado período de tiempo y en términos cuantitativos, dimensionando las respectivas contribuciones (metas), haciendo seguimiento sistemático del desempeño (control) y procediendo a las correcciones necesarias. (Fúquene, 2014), y cuenta con las siguientes características:

1. La estructura básica de la corporación es la forma organizativa que a menudo se llama jerarquía.

2. La administración por objetivos se ocupa del mantenimiento y el crecimiento ordenado de la organización mediante especificaciones de lo que se espera de cada uno de sus miembros y la medición de lo que realmente se realiza.

Según George S. Odiorne, el sistema de administración por objetivos se puede describir como un proceso mediante el cual el superior y subordinado identifican conjuntamente sus objetivos comunes, definen las principales áreas de cada individuo de la responsabilidad en cuanto a los resultados que se esperan de él, y el uso de estas medidas como guías para la operación de la unidad y evaluar la contribución de cada uno de sus miembros (Odiorne, 1977).

Por otra parte, un sistema de gestión (SG) según José F. Naranjo es considerado como una serie de procesos, acciones y tareas que se llevan a cabo sobre un conjunto de elementos (personas, procedimientos, estrategias, planes, recursos, productos, etc.) para

lograr el éxito sostenido de una organización, es decir, disponer de capacidad para satisfacer las necesidades y las expectativas de sus clientes o beneficiarios, trabajadores y de otras partes interesadas a largo plazo y de un modo equilibrado y sostenible (Naranjo, 2015).

Según la Dra. Marcia Almeida Guzmán, un sistema de gestión es conjunto de elementos de una organización interrelacionados o que interactúan para establecer políticas, objetivos y procesos para lograr estos objetivos. Estos elementos establecen la estructura de la organización, los roles y las responsabilidades, la planificación, la operación, las políticas, las practicas, las reglas, las creencias, los objetivos y los procesos para lograr esos objetivos. Por otra parte, el alcance puede incluir la totalidad de la organización, funciones específicas e identificadas de la organización, secciones especificas e identificadas de la organización, o una o más funciones dentro de un grupo de organizaciones (Guzmán, 2015).

El autor de la investigación define que un SG es una herramienta de apoyo que permite controlar los procesos de una empresa, con el propósito mejorar la productividad y a su vez tomar mejores decisiones. Además, realiza una reducción de costos, hace que las actividades se realicen una forma más eficiente, elimina tareas repetitivas permitiendo un mejor trabajo en la organización.

1.2 Análisis de sistemas homólogos

Con el objetivo de verificar si existen algunos sistemas que satisfagan las necesidades de la Facultad 4, se realizó un estudio de las tendencias actuales de los SG a nivel nacional e internacional. La investigación a SG permitió la obtención de indicadores que son de importancia para el sistema, para así ir identificando posibles requisitos funcionales. A partir de la búsqueda efectuada se realizó un análisis de los siguientes sistemas:

Sistema de Gestión Académica (AKADEMOS)

AKADEMOS es un sistema informático cuya principal misión es el control de todos los procesos que intervienen en la gestión académica de un centro de altos estudios. Surgió como respuesta a la necesidad de sustentar y dar soporte en la Universidad de las Ciencias Informáticas (UCI) a toda la labor del personal de la secretaría docente, con la visión de obtener un producto genérico, capaz de ser aplicable y adaptable en cualquier centro que implemente el control docente universitario. La primera versión fue diseñada y desarrollada por un equipo de estudiantes del Plan CUJAE y la UCI, rectorados por la Dirección de

Informatización de la UCI en el año 2004. En sus inicios su desarrollo se basó en el *framework* **.NET** de **Microsoft** (**C#** como lenguaje de programación, **ASP.NET** como *framework* de desarrollo y **Microsoft SQL Server** como Sistema de Gestión de Base de Datos (SGBD), lo que influyó en que posteriormente, se decidiera migrar el mismo hacia plataformas de software libre. En el 2008 fue lanzado la versión 2 de AKADEMOS, usando PHP como lenguaje de programación y PostgreSQL como SGBD (Suárez Hernández & Cepero Rojas, 2008).

Los visión, objetivos y funcionalidades de este sistema ha ido evolucionando para adaptarse a los cada vez mejor estructurados procesos de la Universidad. En el 2015 fue liberado una nueva versión la que se basa en el marco de trabajo **GUUD 2.0**, el cual constituye un híbrido entre el marco de trabajo de **PHP CodeIgniter** y la biblioteca de **JavaScript jQuery**. También es de relevancia señalar que este sistema permite la interoperabilidad a través de servicios web basados en el protocolo de comunicación **SOAP** (Pérez Pérez & Segura Tristá, 2015).

AKADEMOS actualmente gestiona un grupo de información de las áreas docentes, productivas e investigativas de la Universidad referente a Pregrado, Postgrado, Investigación, Extensión, Ingreso y Egreso.

En particular el módulo de Investigación soporta los procesos de esta área en la Universidad. El módulo está estructurado en los siguientes subsistemas: *Eventos*, *Publicaciones*, *Premios*, *Estructura investigativa* y *Evidencias*. Este último subsistema permite el registro de evidencias para Eventos, Premios y Publicaciones lo que le permite al usuario el reconocimiento de su labor científica por parte de la Universidad. Este reconocimiento es tenido en cuenta para la evaluación de los trabajadores del Centro y constituye el primer paso para el cálculo de los indicadores de CTI (Ciencia, Tecnología e Investigación) en las diferentes áreas de trabajo en las cuales está organizada la Institución.

El flujo del proceso es como se describe a continuación:

1. El usuario decide crear una evidencia en la que detalla toda la información necesaria: Evento y edición en el que se presentó, Datos del trabajo (Nombre del trabajo, Resultado, URL de la memoria del evento, fecha del evento, imagen de captura del certificado de participación o de resultado) y Autores de la misma.

2. Este registro es utilizado por la Dirección de Información Científico-Técnica que, de conjunto con la Dirección de Investigaciones, emite un certificado para el reconocimiento de la evidencia.

FlowDesk ERP

Es un sistema de gestión empresarial, que permite integrar todos los procesos internos de la organización, haciéndolas más eficientes. Esta permite el control del negocio en poco tiempo, contando con información exacta, lo que facilita una mejor toma de decisiones. ***FlowDesk*** contiene extensos módulos de análisis y optimización para ofrecer soluciones para una amplia lista de necesidades. Estos módulos se pueden usar para automatizar e integrar las tareas realizadas por todas las áreas de una empresa, incluyendo Marketing, Programación, Planificación y Operaciones. El aumento de la confiabilidad, la eficiencia y la efectividad conduce a mayores ganancias y a mayores ingresos. Ayuda a dirigir un negocio con una perspectiva de "panorama general" a la vez que agiliza las tareas del departamento en una sola solución (Flowdesk ERP, 2019).

Sistema de gestión de evaluación del desempeño (*Hrider*)

Es una herramienta que permite personalizar evaluaciones y contar con funcionalidades flexibles que permiten digitalizar procesos. Evalúa competencias y objetivos por separado o juntos en la misma evaluación. Incluye plan de acciones donde los empleados y supervisores podrán colaborar definiendo objetivos y acciones necesarias para lograrlos, registra sus progresos y conoce fácilmente las acciones pendientes, completadas o atrasadas. Realiza informes de resultados mediante gráficos los cuales ayudan a la toma de decisiones y a definir las estrategias de la organización (Hrider, 2019).

XEDRO GESPRO

Es un sistema orientado a la web que permite la planificación, seguimiento y control de productos en forma de proyectos. Cuenta con herramientas para el apoyo a la toma de decisiones a nivel de proyecto, nivel de entidad ejecutora y nivel gerencial. Se presenta en un modelo de negocios basado en servicios que combinan el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de proyectos. Esta combinación posibilita no sólo la informatización de las organizaciones sino también la mejora integral de los procesos de planificación, control y seguimiento de proyectos. El modelo de negocio es basado en servicios y el precio del producto varía de acuerdo a los servicios definidos, incluyendo

dentro de estos, consultorías sobre gestión de proyectos. Actualmente es utilizado en la Universidad de las Ciencias Informáticas (UCI) (Universidad de las Ciencias Informáticas , 2019).

Cuadro de Mando Integral (CMI)

El cuadro de mando integral es una herramienta del ámbito empresarial ampliamente utilizada para evaluar el desempeño de una organización, e incluso ha llegado a convertirse en un sistema de gestión para implementar las estrategias empresariales. Su aplicación al ámbito de las tecnologías de la información (TI) también se ha extendido ampliamente como instrumento de alineación estratégica y evaluación del desempeño (Patricia Pérez Lorences, 2013).

El CMI integra cuatro perspectivas o áreas clave y las relaciona con la misión, visión y objetivos. Las cuatro perspectivas son la financiera, la de formación, los procesos internos y la relación con los clientes / usuarios.

Existen diferentes tipos de CMI:

1) CMI operativos: útiles para la gestión del cambio (innovaciones en la organización) en periodos breves de tiempo.

2) CMI estratégicos: definen los objetivos básicos de la organización en relación en su misión y visión a largo plazo.

3) CMI departamentales: específicos para un área de la organización: financiera, dirección, recursos humanos, etc.

4) CMI organizativos: definidos según los niveles de responsabilidad.

La herramienta básica del CMI son los indicadores, es decir, índices que describen el comportamiento de diversas variables ya sea por cuantificación directa de una variable (primarios) o por comparación entre variables (secundarios). Es preferible que los indicadores sean de naturaleza numérica, ya que así permitirán el establecimiento de niveles de tolerancia o de desviaciones aceptables dentro del CMI. En la definición de los indicadores hay que escoger aquellos que sean: relevantes, pertinentes, unívocos, objetivos, precisos y accesibles (que el coste de obtenerlos no sea desorbitado).

1.2.1 Resumen del estudio de los sistemas

El estudio realizado a los sistemas anteriores se pudo llegar a la siguiente conclusión:

El módulo de Investigación del sistema **AKADEMOS**, permite la validación y el reconocimiento oficial de la actividad científica – investigativa de profesores y especialistas en la UCI. El subsistema de Evidencia es la base para el mencionado reconocimiento y se tiene en cuenta la información antes detallada para la consecución de sus objetivos.

Los sistemas ***FlowDesk ERP*** y (***Hrider***) permiten la gestión de objetivos y evaluaciones de los mismos. Estos realizan informes y el avance de sus objetivos se puede observar mediante gráficos, optimizan procesos dentro de las organizaciones, pero no son soluciones factibles ya que estos son sistemas privados.

El sistema **XEDRO GESPRO** esta permite la planificación, seguimiento y control de productos en forma de proyectos. Este permite la evaluación de actividades a través de indicadores y a partir de un análisis estadístico de la información referente al cumplimiento de proyectos visualizándolos en gráficos. También permite la exportación de información para realizar comparaciones con el fin de ver el avance de estas tareas o actividades. El sistema es una buena opción para satisfacer las necesidades de la Facultad, pero está orientado a proyectos y no a objetivos estratégicos por lo que no es una solución factible a nuestro problema. Para el estudio de este sistema se tuvo en cuenta la exportación de información, las evaluaciones de las actividades y los gráficos donde se muestra el avance de las actividades.

El **CMI** es un sistema de gestión para la toma de decisiones compuesta por objetivos estratégicos, indicadores y metas, este no es una solución al problema, pero si se tuvo en cuenta su principal herramienta: los indicadores, los cuales describen el comportamiento de diversas variables ya sea por cuantificación directa de una variable (primarios) o por comparación entre variables.

Al no existir un sistema que se adapte o cumpla todos los requisitos necesarios es preciso desarrollar una nueva solución basada en la web. Se toma la decisión de hacer una aplicación web debido a las ventajas que este ofrece, pues se podrá usar desde cualquier ordenador de la institución aumentando la movilidad empresarial, solo haciendo uso del navegador, por lo que se obtiene una solución multiplataforma. Solo basta con tener conexión a la red, para ello la institución posee la infraestructura necesaria. A continuación, se procede a estudiar las metodologías de desarrollo de software, herramientas y tecnologías actuales para identificar las más apropiadas para el desarrollo de la aplicación que se propone.

1.3 Tecnologías y herramientas

Es importante establecer que para el desarrollo de un SG web en correspondencia con los estándares y las tendencias actuales es necesario hacerlo desde dos perspectivas diferentes: del lado del ***Front-End*** y del lado del ***Back-End***. Básicamente serán abstracciones que ayudarán a mantener las diferentes partes del sistema en forma separada.

1.3.1 Front-End

Son todas aquellas tecnologías que se ejecutan del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, HTML, CSS y JavaScript (serprogramador.es, 2019).

HTML

HTML es un conjunto de normas que debemos de seguir, cuyo objetivo es desplegar un sitio web de forma correcta e indicada para que los navegadores web puedan leer el lenguaje HTML e interpretarlo, para así finalmente mostrar al usuario un sitio web (Gauchat J. D., 2012). Básicamente el lenguaje de marcado HTML se puede definir como un estándar que sirve para describir la estructura básica de una página y organizar la forma en que se mostrará su contenido, además de que HTML permite incluir enlaces (***links***) hacia otras páginas o documentos.

CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (Gauchat J. D., 2012).

Al crear una página web, se utiliza en primer lugar el lenguaje ***HTML/XHTML*** para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página (Gauchat J. D., 2012). Dentro del CSS existen ***frameworks*** muy usados para mejorar y facilitar el diseño del sistema. El que se utilizará en el proyecto es Bootstrap en su versión 4.1 por las siguientes razones:

Bootstrap v4.1

Bootstrap, es un ***framework*** originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como ***responsive design*** o diseño adaptativo (Mestras, Aplicaciones Web/Sistemas Web, 2014).

El beneficio de usar ***responsive design*** en un sitio web, es principalmente que el sitio web se adapta automáticamente al dispositivo desde donde se acceda. Lo que se usa con más frecuencia, es el uso de media queries, que es un módulo de CSS3 que permite la representación de contenido para adaptarse a condiciones como la resolución de la pantalla (Codina, 2017). Algunas de sus características son:

- Fácil e intuitivo: En vez de estar modificando grandes cantidades de código CSS se maqueta directamente.
- Optimizado para dispositivos móviles: Provee las reglas de CSS3 necesarias para que la aplicación se adapte a cualquier resolución de pantalla.
- Se integra perfectamente con las principales librerías JavaScript.
- Ofrece un diseño sólido usando estándares como CSS3/HTML5.

JavaScript

JavaScript es un lenguaje de programación de alto nivel que se utiliza en las aplicaciones web para proveer dinamismo a las páginas web estáticas diseñadas solo con HTML. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web que permite mejoras en la interfaz de usuario y páginas web dinámicas. Los navegadores proporcionan una implementación Modelo de Objetos de Documento (DOM, por sus siglas en inglés) que permite manipular las interfaces mediante lenguaje JavaScript una vez que estas ya están cargadas en el navegador del cliente.

Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Gauchat J. D., 2012).

Node.js es un entorno de ejecución de E/S dirigida por eventos y por lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google V8. Esta es importante al ejecutar ***frameworks*** *JavaScript*. Este entorno JavaScript no se ejecuta en el navegador,

sino en el lado del servidor. Se basa en la máquina virtual V8 de Google, el mismo entorno de ejecución para JavaScript que Chrome utiliza. Su fin es construir fácilmente aplicaciones escalables (Manuel Kiessling, 2015).

Node.js:

- Puede utilizarse en la gran mayoría de servidores, incluyendo los más conocidos como Unix, Microsoft o Mac.
- Al estar inspirado en JavaScript cuenta con una curva de aprendizaje baja lo que permite que cualquier programador lo pueda aplicar.
- Permite trabajar tanto desde el servidor como desde el cliente, es posible generar una transferencia de información mucho más rápida e inmediata.
- Tiene una comunidad bastante amplia que permite establecer lazos de colaboración muy interesantes desde diferentes ópticas lo cual resulta muy enriquecedor para los desarrolladores.

Frameworks JavaScript

Los *frameworks JavaScript* o marcos de trabajo no son más que bibliotecas de código que sirven para el desarrollo de aplicaciones. En la actualidad existen gran cantidad de *frameworks* dentro de los más usados se encuentran Angular y Vue. Estos frameworks son de gran utilidad por lo que se analizará las ventajas y desventajas para escoger cual será utilizado.

Tabla 1. Angular [Fuente: Elaboración propia]

Este es el framework MVW (***Model View WhatEver***) de Google que se ha encargado de traer orden a las aplicaciones JavaScript y potenciar las Arquitecturas SPA. Es un framework de código abierto, realizado en JavaScript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso sobre el lenguaje JavaScript con programación del lado del cliente (Alberto Basalo, 2016).	
Ventajas	**Desventajas**

• Sugiere muy buenas convenciones para nombrar los archivos, componentes, módulos y servicios. • Documentación detallada que permite obtener toda la información necesaria para el desarrollador individual. • Enlace de datos bidireccional que permite un comportamiento singular para la aplicación que minimiza los riesgos a posibles errores. • MVVM (Model-View-ViewModel) que permite a los desarrolladores trabajar por separado en la misma sección de la aplicación utilizando el mismo conjunto de datos.	• La sintaxis compleja que proviene de la primera versión de Angular. Sin embargo, Angular 5 usa TypeScript 2.4, que es el menos difícil de aprender en comparación. • Problemas de migración que pueden aparecer al pasar de la versión anterior a la última.

Tabla 2. Vue [Fuente: Elaboración propia]

Es un framework progresivo, es decir, es un framework que sirve para consumir interfaz del usuario. Vue está diseñado desde el inicio para ser adoptado incrementalmente. La biblioteca principal se enfoca solo en la capa de la vista, y es muy simple de utilizar e integrar con otros proyectos o bibliotecas existentes. Por otro lado, Vue también es perfectamente capaz de soportar aplicaciones sofisticadas de una sola página cuando se utiliza en combinación con herramientas modernas y librerías compatibles (Sánchez J. A., 2018).	
Ventajas	**Desventajas**
• Gran escala. Vue.js puede ayudar a desarrollar plantillas reutilizables bastante grandes que se pueden crear sin tiempo adicional asignado	• Riesgo de exceso de flexibilidad. A veces, Vue.js puede tener problemas al integrarse en grandes proyectos y todavía no hay

para eso según su estructura simple. • Adaptabilidad. Proporciona un período de cambio rápido de otros frameworks a Vue.js debido a la similitud con Angular en términos de diseño y arquitectura. • Impresionante integración. Vue.js se puede utilizar tanto para crear aplicaciones de una sola página como para interfaces web de aplicaciones más difíciles. Lo principal es que las partes interactivas más pequeñas se pueden integrar fácilmente en la infraestructura existente sin ningún efecto negativo en todo el sistema. • Documentación detallada Vue.js tiene una documentación muy circunstancial que puede ajustar la curva de aprendizaje para los desarrolladores y ahorrar mucho tiempo para desarrollar una aplicación utilizando solo los conocimientos básicos de HTML y JavaScript.	experiencia con posibles soluciones, pero definitivamente vendrán pronto. • Fondo chino En cuanto a Vue.js tiene un poco de origen chino, muchos elementos y descripciones todavía están disponibles en chino. Esto conduce a una complejidad parcial en algunas etapas de desarrollo, sin embargo, cada vez se traducen más materiales al inglés. • Falta de recursos. Vue.js todavía tiene una cuota de mercado bastante pequeña en comparación Angular, lo que significa que el intercambio de conocimientos en este framework todavía está en la fase inicial.

Una vez realizada la investigación de los framework JavaScript se llegó a la siguiente selección: Se elige Angular.js en su versión 6, el cual será utilizado para el desarrollo del SG por las siguientes razones:

- Es un framework gratuito y de código abierto.
- Usa la Arquitectura de diseño MVVM.

- Comunidad muy grande con multitud de tutoriales y librerías.
- Programación reactiva, la vista se actualiza automáticamente a los cambios.
- Basado en componentes, es decir, podemos escribir componentes web con vista y lógica para después reutilizarlos en otras páginas.
- Es el framework más usado y más popular de todos.

1.3.2 Back-End

El Back-End es el trabajo del lado del servidor: es decir, el programador backend se encarga de los lenguajes PHP, Python, .Net, Java, entre otros. También es el responsable de interactuar con bases de datos, comprobar y asignar roles y sesiones a los usuarios, y es el que acaba de asegurarse que el ***Front-End*** se vea de forma correcta gracias al desarrollo de la página en el servidor. Actualmente los lenguajes de programación más usados en la UCI se encuentran Java y PHP por lo que se realizará una comparación para así seleccionar el lenguaje a usar.

Tabla 3. PHP (Hypertext Pre-processor) [Fuente: Elaboración propia]

<table>
<tr><td colspan="2">Es un lenguaje de programación utilizado para la creación de sitios web dinámicos. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”. Surgió en 1995, desarrollado por PHP Group. Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, Open Database Connectivity (ODBC), Microsoft SQL Server, Firebird y SQLite, lo cual permite la creación de aplicaciones web robustas (Sansón, 2019).

PHP tiene también la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como Linux, Windows y Mac OS X. Interactúa con los servidores web más populares que existen en versión Common Gateway Interface (CGI), módulo para Apache e ISAPI (Mestras, PHP, Aplicaciones Web/Sistemas Web, 2013).</td></tr>
<tr><td>Ventajas</td><td>Desventajas</td></tr>
<tr><td>• Muy fácil de aprender.
• Se caracteriza por ser un lenguaje muy rápido.</td><td>• Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto, puede ser más ineficiente a medida que las solicitudes aumenten de número.</td></tr>
</table>

• Soporta en cierta medida la orientación a objeto. Clases y herencia. • Es un lenguaje multiplataforma: Linux, Windows, entre otros. • Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras. • Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones. • Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. • Incluye gran cantidad de funciones.	• La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP. • La programación orientada a objetos es aún muy deficiente para aplicaciones grandes. • Dificulta la organización por capas de la aplicación.

Tabla 4. JAVA [Fuente: Elaboración propia]

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Java nacía como un lenguaje de programación que pudiese ser multiplataforma y multidispositivo, bajo el paradigma "***Write Once Run Anywhere***" (WORA). De esta forma un programa Java escrito una vez podemos ejecutarle sobre diferentes plataformas, siendo soportados los sistemas operativos Windows, MacOs y UNIX. Y a su vez en diferentes tipos de dispositivos. Su sintaxis es la de C++ "simplificada". Los creadores de Java partieron de la sintaxis de C++ y trataron de eliminar de este todo lo que resultase complicado o fuente de errores en este lenguaje (Flask, 2016).

Ventajas	Desventajas
• Lenguaje totalmente orientado a Objetos • Disponibilidad de un amplio conjunto de bibliotecas. • Lenguaje simple • Distribuido • Interpretado y compilado a la vez • Robusto • Seguro • Indiferente a la arquitectura	• No es tan simple: Adquirir un estilo de programación orientada a objetos es difícil. También saber sobre la filosofía y los componentes en las clases de librerías es difícil y consume mucho tiempo. • Para poder ejecutar un programa de Java en múltiples dispositivos, cada uno requiere un intérprete de Java. • La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.

Selección del lenguaje de programación

Una vez estudiadas e interpretadas las principales características de los lenguajes de programación, se decidió que para este trabajo se debe seleccionar PHP en su versión 7.2.4 por las siguientes razones:

- PHP usa tanto programación estructurada (funciones) como Programación Orientada a Objetos (clases y objetos).
- PHP se basa en que permiteembeber pequeños fragmentos de código dentro de lo que sería una página común creada con HTML. Esos scripts PHP nos permiten realizar determinadas acciones de una forma fácil y eficaz, pudiendo realizar todo tipo de tareas, de las más simples a las más complejas.
- Sencillo, directo y multiplataforma.
- PHP es un lenguaje totalmente libre y abierto. Tiene una curva de aprendizaje muy baja, su sintaxis es simple y cumple estándares básicos de la programación orientada a objetos.
- Existen multitud de herramientas, librerías, frameworks gratuitos.

- PHP es unos de los mejores lenguajes para trabajar en la web.

Framework de desarrollo de PHP

El desarrollo de aplicaciones web como herramientas facilitadoras para el desarrollador, brindan una base sólida para la construcción de la misma. Los llamados frameworks para PHP, son un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. Son un conjunto de archivos, en este caso PHP, que vienen preparados con toda la estructura necesaria para desarrollar varios tipos de proyectos (F. Sierra, 2015). En la actualidad existen una gran diversidad de frameworks desarrollados en PHP por esto se realizará una comparación entre el Frameworks más usado en la UCI Symfony con Laravel este último relativamente reciente:

Tabla 5. Laravel [Fuente: Elaboración propia]

<table>
<tr><td colspan="2">Es uno de los frameworks de código abierto más fáciles de asimilar para PHP. Es simple, muy potente y tiene una interfaz elegante. Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC. El objetivo de Laravel es el de ser un framework que permita el uso de una sintaxis refinada y expresiva para crear código de forma sencilla, evitando el “código espagueti” y permitiendo multitud de funcionalidades. Aprovecha todo lo bueno de otros frameworks y utiliza las características de las últimas versiones de PHP. La mayor parte de su estructura está formada por dependencias, especialmente de Symfony, lo que implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias (Taylor Otwell, 2019).</td></tr>
<tr><td>Ventajas</td><td>Desventajas</td></tr>
<tr><td>Laravel Framework facilitará el proceso de desarrollo, ya que lleva mucho menos tiempo completar el proyecto web.
Este Marco aumentará el tráfico en línea para su sitio web, ya que es aplicable en cualquier navegador y dispositivo.
Este Marco simplifica el sitio web ya que incorpora ciertas funciones</td><td>Está pensado para las últimas versiones de PHP y solo soporta a partir de PHP 5.4.</td></tr>
</table>

flexibles en su aplicación web. Se compone de sistemas de embalaje modular. • Este Marco proporciona URL únicas que pueden crear diferentes rutas con el nombre de ruta similar. • Cuenta con facilidad de carga automática no es necesario incluir rutas y mantenimiento. • Proteger los sitios web de ataques de seguridad dirigidos.	

Tabla 6. Symfony [Fuente: Elaboración propia]

Según los autores Fabien Potencier, François Zaninotto, Symfony es un framework para construir aplicaciones web con PHP. En otras palabras, Symfony es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web (Fabien Potencier, 2019). Este framework consiste en el conjunto de componentes y códigos PHP reutilizables que se utilizan principalmente para aplicaciones web a gran escala. Puede ayudar a superar tareas como plantillas, autenticación, enrutamiento, configuración de objetos y mucho más. Permite el desarrollo de aplicaciones web empresariales a gran escala.	
Ventajas	**Desventajas**
• Como este marco es un código abierto, le permite desarrollarse sin ninguna restricción. • El marco de Symfony es más potente y flexible. • Symfony combina medidas de seguridad preventivas para	• En Symfony, el mecanismo de seguridad es un poco difícil. • El análisis de archivos es muy difícil de manejar. • Se utiliza principalmente para el desarrollo de aplicaciones web a gran escala y no a pequeña escala.

superar ataques, inyección SQL. Integra estos procesos de seguridad como medidas integradas. • Mantiene documentación completa y actualizada. • Fiabilidad del código del componente.	

Está claro que cada framework cumple ciertos requisitos mejor que el otro. Ambos ofrecen ayudas muy útiles para el desarrollo. Como todo, decidir si utilizar un framework en lugar de otros siempre es decisión del desarrollador. Hay que recalcar que cada framework se adapta diferente, con base a los requerimientos de cada aplicación. La selección del frameworks será Laravel porque tiene una curva de aprendizaje bastante cómoda, además de una comunidad muy grande dispuesta a ayudar. No es necesario ser un experto para programar con él, este da la libertad de crear una estructura con base a las necesidades del desarrollador. Es muy útil en aplicaciones web sencillas. Su gran cantidad de componentes y paquetes hace que te enfoques en lo que es más importante: la lógica de negocio del proyecto.

Una vez escogido el lenguaje de programación y el framework de trabajo es importante saber que en el Back-End se encarga más que nada de la manipulación de los datos, por lo que el programador de Backend debe de conocer de Bases de Datos ya que él se va a encargar de que todos los datos que llegan desde el Front-End lleguen a una base de datos para poder gestionarlos.

Sistemas Gestores de Bases de Datos (SGBD)

Un sistema gestor de bases de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener una base de datos, y proporciona acceso controlado a la misma. En general, un SGBD proporciona los siguientes servicios (Tutorialspoint, 2015):

- Permite la definición de la base de datos mediante el lenguaje de definición de datos (DDL–Data Description Language). Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos.

- Permite la inserción, actualización, eliminación y consulta de datos mediante el lenguaje de manejo o manipulación de datos (DML-Data Manipulation Language).
- Proporciona un acceso controlado a la base de datos mediante sistemas de seguridad, integridad, control de concurrencia, control de recuperación, un diccionario de datos o catálogo accesible y por el usuario que contiene la descripción de los datos de la base de datos.

La principal herramienta de un SGBD es la interfaz de programación con el usuario. Esta interfaz consiste en un lenguaje muy sencillo mediante el cual el usuario interactúa con el servidor. Este lenguaje comúnmente se denomina SQL, ***Structure Query Language***, está estandarizado por la ISO 1, es decir, todas las BD que soporten SQL deben tener la misma sintaxis a la hora de aplicar el lenguaje (Guevara, 2018).

Tabla 7. PostgreSQL [Fuente: Elaboración propia]

<table>
<tr><td colspan="2">Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Es un gestor de bases de datos de código abierto, brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación (Marín, 2019).</td></tr>
<tr><td>Ventajas</td><td>Desventajas</td></tr>
<tr><td>• Sólida implementación de las especificaciones SQL estándar.
• Proporciona compatibilidad con capacidades avanzadas de SQL, como expresiones de tabla comunes y funciones de Windows</td><td>• Cuando se trata de popularidad, de ninguna manera está cerca de MySQL. Como tal, esto hace que sea difícil de aprender y dar soporte, dada su base de soporte de la comunidad baja PostgreSQL vs MySQL.
• Su implementación de replicación no se refina como la de MySQL.</td></tr>
</table>

• PL / pgSQL, que es su lenguaje de programación de procedimientos, interactúa muy bien con SQL. • Su soporte para tipos de datos avanzados, como matrices multidimensionales, tipos definidos por el usuario, etc. le da una ventaja sobre MySQL. • Su capacidad para manejar tareas complejas a través de procedimientos almacenados es definitivamente una ventaja cuando se compara con MySQL • PostgreSQL tiene una ventaja cuando se trata de mantener la integridad de los datos.	• Carece PostgreSQL cuando se trata de opciones de lectura pesadas en comparación con MySQL. • PostgreSQL puede ser un poco difícil de configurar debido a la falta de una gran comunidad de soporte como MySQL.

Tabla 8. MySQL [Fuente: Elaboración propia]

MySQL es un sistema gestor de bases de datos relacionales rápido, sólido y flexible. Es idóneo para la creación de bases de datos con acceso desde páginas web dinámicas, así como para la creación de cualquier otra solución que implique el almacenamiento de datos, posibilitando realizar múltiples y rápidas consultas. Está desarrollado en C y C++, facilitando su integración en otras aplicaciones desarrolladas también en esos lenguajes (Marín, 2019).

Ventajas	**Desventajas**
• Es rico en características y soporta muchas especificaciones SQL • Definitivamente es más barato y está fácilmente disponible debido al amplio soporte entre los	• A diferencia de PostgreSQL, MySQL no admite capacidades avanzadas de SQL como las funciones de Windows • Algunas funcionalidades, como referencias, transacciones,

proveedores de servicios y los hosts web. • MySQL es rápido y funciona de manera eficiente, por lo que puede manejar una gran cantidad de datos y escalar adecuadamente cuando sea necesario. No es de extrañar por qué grandes sitios web y aplicaciones como Facebook lo utilizan. • Viene incorporado con una gran cantidad de características de seguridad para garantizar la protección de datos • Otra gran ventaja para MySQL es su facilidad de configuración. Ya sea que lo esté ejecutando de forma independiente o en clústeres, proporciona un mejor soporte y documentación.	auditorías, etc., se vuelven un poco ineficientes como las que se obtienen de otros sistemas de administración de bases de datos PostgreSQL vs MySQL. • MySQL tiene una capacidad limitada para manejar la transformación de datos complejos. • Tiene un mayor riesgo de pérdida de datos y corrupción, pero esto no es importante y puede mitigarse manteniendo copias de seguridad y / o replicaciones regulares.

Selección del SGBD

Se elige el ***SGBD MySQL*** en su versión 5.0.12 debido a que es un sistema libre y gratuito. MySQL es más simple debido a que se usa para poder trabajar de forma sencilla y rápida. Las consultas suelen ser pequeñas y si es necesario hacer modificaciones la idea es que no se demore mucho en ello. En otras palabras, es ideal para bases de datos pequeñas y medianas. MySQL es confiable e implementa seguridad basada en Listas de Control de Acceso (ACLs, por sus siglas en inglés) para todas las conexiones, consultas y otras operaciones que un usuario pudiera intentar realizar. También hay algo de soporte para conexiones cifradas con SSL entre clientes y servidores MySQL. Tiene una gran comunidad de contribuyentes que, particularmente desde la adquisición por parte de

Oracle, se centra principalmente en mantener características existentes con algunas características nuevas surgiendo ocasionalmente.

phpMyadmin

Es una herramienta de Software Libre escrita en PHP creada para la manipulación de la base de datos MySQL a través de los clientes web. Soporta una amplia gama de operaciones. Éstas son efectuadas a través de una interfaz gráfica. Las más frecuentes son el manejo de bases de datos, tablas, campos, relaciones, índices, usuarios y permisos. También proporciona la capacidad de ejecutar directamente sentencias SQL. Es muy utilizada en aplicaciones con Symfony2 y PHP (phpMyAdmin, 2019).

Servidor web

Según Esteban Borges un Servidor web no es más que un equipo informático que satisface peticiones de otras máquinas (clientes) de forma remota (Borges, 2018). Los servidores web son los pilares del Hosting, permiten a los clientes alquilar almacenamiento en servidores para alojar sus archivos. El hosting es la prestación en donde una empresa ofrece a su cliente el espacio físico (dentro de un ordenador conocido como “servidor”) para que almacene los datos del sitio web de modo que esté siempre en línea. Este es el encargado de despachar archivos por los protocolos HTTP y HTTPS hacia un cliente que solicita dicha información.

Un servidor web Apache puede ser una excelente opción para ejecutar tu sitio web en una plataforma estable y versátil. Unos de los servidores más usados en la Universidad de Ciencias Informáticas es el servidor Apache el cual es que se utilizara debido a que:

- Entre sus ventajas encontramos que es código abierto, es además software gratuito, y multiplataforma (Windows, Linux y Unix).
- Software confiable y estable.
- Parches de seguridad regulares y actualizados con frecuencia.
- Fácil de configurar para principiantes.
- Enorme comunidad y soporte fácilmente disponible en caso de cualquier problema.
- Apache soporta HTTPS, haciendo de esta manera más seguro el servidor Web.

1.4 Entorno de Desarrollo Integrado

En el desarrollo del Sistema es importante utilizar un ambiente de desarrollo integrado o entorno de desarrollo interactivo (IDE, por sus siglas en inglés) el cual proporciona herramientas para agilizar el proceso de desarrollo del programador. Un IDE consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Estos pueden constituir aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Además, proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. Para la implementación del sistema propuesto en esta investigación se seleccionó como entorno de desarrollo integrado NetBeans IDE 8.1 (Rouse, 2018).

NetBeans IDE 8.1 es un entorno de desarrollo integrado libre, de código abierto y sin restricciones para su uso; pensado principalmente para el desarrollo de aplicaciones con el lenguaje Java. Permite escribir, compilar, depurar y ejecutar programas. Soporta además otros lenguajes tales como PHP, C/C++, XML, HTML. Todas sus funciones de este entorno son provistas por módulos. Tiene versiones para Windows, Linux, Mac OS. Para el desarrollo de la solución se utilizará la versión 8.1. Se requiere utilizar NetBeans debido a que facilita el desarrollo del sistema, al proveer un entorno de desarrollo profesional para desarrollar sistemas (Alyona Stashkova, 2016).

1.5 Herramientas Case

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación (Patrick Thibodeau, 2016). CASE se define también como:

- Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.
- La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.
- Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

1.5.1 Visual Paradigm

Es una herramienta para la Ingeniería de Software Asistida por Computación (CASE, por sus siglas en inglés) disponible para los sistemas operativos Linux y Windows. Permite realizar ingeniería directa o inversa sobre el software y es capaz de, a partir de un modelo relacional en diferentes sistemas gestores de base de datos, desplegar todas las clases asociadas a las tablas. Permite la generación de diagramas, diseño centrado en casos de uso, soporta aplicaciones web, genera reportes, genera bases de datos, entre otras características. Soporta el ciclo de vida completo de desarrollo de un software, desde la fase de análisis hasta el despliegue del mismo (Burrows, Visual Parading, 2019).

Esta herramienta es integrable con varios entornos de desarrollo como Eclipse, NetBeans, Visual Studio. Permite aumentar la velocidad en el análisis, captura, plan, desarrollo, comprobación y despliegue de los requisitos; es multiplataforma y cuenta con una versión libre para la comunidad (***Community Edition***). Genera la documentación en los formatos PDF, HTML y en formato de documento de Microsoft Word y permite importar proyectos de otras herramientas de modelado como Rational Rose, Erwin y Microsoft Vision (Burrows, Visual Parading, 2019).

La herramienta CASE a utilizar es Visual Paradigm 8.0, la cual es multiplataforma y posee una excelente integración con los IDE. Brinda grandes facilidades para la creación de diagramas y cuenta con la posibilidad de realizar control de versiones durante todo el ciclo de trabajo.

1.5.2 Lenguaje Unificado de Modelado

“El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas” (Larman, 2019).

UML centra el desarrollo en tres modelos diferentes:

- Modelo funcional: diagramas de casos de uso, describen el sistema desde la perspectiva del usuario.
- Modelo objeto: diagrama de clases, describen la estructura de un sistema en términos de objetos, atributos, asociaciones y operaciones.

- Modelo dinámico: diagramas de secuencia y de estados, describen el comportamiento del sistema.

Dentro de sus características se pueden mencionar algunas tales como:

- Tecnología orientada a objetos.
- Viabilidad en la corrección de errores.
- Desarrollo iterativo e incremental.
- Participación del cliente en todas las etapas del proyecto.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

Para él modelado del SG se utilizará la versión 2.5.1. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

1.6 Metodología de desarrollo de software

Cada proceso de desarrollo de una solución informática, debe estar regido por alguna metodología de desarrollo de software que guíe los procesos y permita tener un registro detallado del avance de la investigación y la creación del SG.

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. La metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo (Maida, 2015). Estas se dividen en dos tipos: las robustas y las ágiles. Las metodologías robustas o

pesadas están concebidas para aplicaciones grandes que requieran gran nivel de detalles y que posean un tiempo largo de creación. Estas metodologías generan gran cantidad de documentación debido a que es posible que un proceso de desarrollo de software pase por varios equipos de trabajo. Su funcionamiento está basado en iteraciones que cubren todo el ciclo de vida del proyecto, lo que permite que se minimicen los riesgos, y se desarrolle el software en corto tiempo.

Las metodologías ágiles han ganado bastante popularidad desde hace algunos años. Si bien son una muy buena solución para proyectos a corto plazo, en especial, aquellos proyectos en donde los requerimientos están cambiando constantemente. El diseño de la arquitectura de software es una práctica muy importante para el desarrollo de software. Tener una buena arquitectura implica que el sistema tiene atributos de calidad que nos van a dar un valor muy importante en el software. Si se definen actividades que fomenten el uso de métodos para el desarrollo de la arquitectura, en un proceso de desarrollo de software, se puede obtener muchos beneficios con respecto al producto que se desarrolló (Maida, 2015).

La metodología (AUP por sus siglas en inglés ***Agile Unified Process)*** de Scott Ambler la cual es una versión simplificada del Proceso Racional Unificado (RUP, por sus siglas en inglés) describe de una manera simple y fácil como entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (Sánchez T. R., 2014).

El AUP aplica técnicas ágiles incluyendo (Sánchez T. R., 2014):

- Desarrollo Dirigido por Pruebas del inglés ***Test Driven Development*** (TDD).
- Modelado Ágil.
- Gestión de Cambios Ágil.
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva.

- Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

- Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

1.6.1 Variación de AUP para la UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se contó con el Modelo CMMI-DEV v1.3. el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora, dichas prácticas se centran en el desarrollo de productos y servicios de calidad (Sánchez T. R., 2014).

1.7 Descripción de las fases

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

Tabla 9. Fases de AUP y AUP-UCI [Fuente: Documento Oficial]

Fases AUP	**Fases Variación AUP-UCI**	**Objetivos de las fases (Variación AUP-UCI)**
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite

		obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.7.1 Escenarios para la disciplina Requisitos

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN), Descripción de Procesos del Negocio (DPN) o Modelo Conceptual (MC)) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma (Sánchez T. R., 2014):

Escenario No 1:

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS

Escenario No 2:

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

Escenario No 3:

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

Escenario No 4:

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

Durante la investigación se utilizará el escenario número cuatro porque aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

1.8 Conclusiones parciales

Luego del estudio y análisis realizado del objeto de investigación del presente trabajo de diploma, apoyado en los métodos de la investigación científicos identificados, se concluye lo siguiente:

- El análisis de las características y funcionamiento de los Sistemas de Gestión en el ámbito nacional e internacional, permitió comprobar que los sistemas estudiados no satisfacen en su totalidad a las necesidades de la Facultad 4, lo que ratificó la necesidad de crear una solución para el problema planteado e incluir y adaptar algunas de las funcionalidades estudiadas de dichos sistemas.
    - El sistema a desarrollar debe de estar permitir medir el avance del cumplimiento de los objetivos de trabajo, a través de estructuras visuales que viabilicen la rápida retroalimentación de la información para determinados usuarios.
    - La subida de evidencia anteriormente validada y reconocida por la Institución constituiría el primer paso para la cuantificación de las actividades que permitirían el avance del cumplimiento de los objetivos de trabajo.

- El filtrado de la información según el estado del cumplimiento sería una valiosísima herramienta para el apoyo a la toma de decisiones.

- El estudio de las diferentes metodologías de desarrollo de software permitió elegir AUP-UCI por lo que a partir de la misma se logra acelerar el desarrollo del software y permite documentar dicho proceso a partir de las HU.
- El estudio y selección de las herramientas seleccionadas permite obtener el entorno de desarrollo adecuado para satisfacer las necesidades del cliente garantizando el cumplimiento de los objetivos planteados.

Capítulo 2. Análisis y Diseño de la propuesta de solución

El presente capítulo aborda las principales características del SG. Se definen los requisitos funcionales y no funcionales del sistema. La metodología AUP, en su variante UCI, define en el escenario escogido el artefacto que tiene como fin facilitar la comprensión del negocio y el sistema a desarrollar. El artefacto que se presenta son las historias de usuario. En correspondencia con los principios ágiles. Esta metodología permite a los desarrolladores generar la documentación que consideren necesaria para mantener una comunicación fluida con los clientes. Se hace referencia además a los patrones de arquitectura y diseño que se tuvieron en cuenta para el desarrollo del sistema.

2.1 Descripción de la propuesta de solución

Como propuesta que fundamenta esta investigación se plantea, la creación de un Sistema de Gestión desarrollado con el framework **PHP** Laravel, que permita elevar el proceso de planificación, ejecución y control de los objetivos estratégicos de la Facultad 4. Se lleva a cabo la informatización del proceso actual de manera que la información se encuentre disponible, centralizada, segura y sea fácil de manipular, esto se facilita ofreciéndoles a los usuarios el estado de cumplimiento de los objetivos en tanto por cientos. Además, el acceso al sistema se realiza haciendo uso de su usuario y contraseña por lo que solo tendrá acceso a los datos de su correspondiente ARC. Se garantiza que el sistema desarrollado exporte información contenida dentro de él a formato PDF y brinde una interfaz con claridad, coherencia e interactividad. Todo esto permite a las áreas de resultados clave poder llevar a cabo el proceso de una manera más organizada y menos propensa a errores.

Para la mejor compresión del sistema se realizó el modelo de clases conceptuales que se presenta en la figura 1.

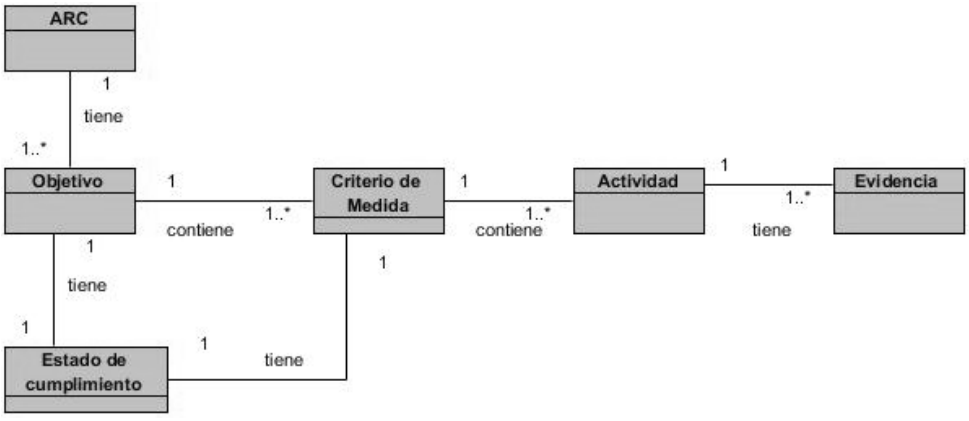


Figura 1 Modelo conceptual del negocio [Fuente: Elaboración Propia]

Con el objetivo de restringir el acceso a las opciones de gestión que presenta el sistema y en cumplimiento de las reglas del negocio, se definen los siguientes roles:

- Administrador: Posee control para asignar roles dentro el sistema y tiene acceso al panel de administración.
- Decano: Gestiona las áreas de resultados clave.
- Jefe de ARC: Gestiona los objetivos, criterios de medidas y redacta las actividades pertenecientes a su área.
- Activista de ARC: Encargado de subir al sistema las evidencias de los trabajadores para cada una de las actividades.

2.2 Requisitos de software

Los requisitos cumplen un papel primordial en el proceso de producción de software, ya que se enfocan en la definición de lo que se desea producir. El flujo de trabajo de requisitos es uno de los más importantes para el desarrollo de un software, ya que su propósito fundamental es guiar el desarrollo hacia el sistema correcto (Almira, 2012). Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes; minimizando problemas relacionados con la mala gestión de los requerimientos. Según el autor Michael Arias, un requisito no es más que la descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso (Chaves, 2017).

Los requisitos de software pueden dividirse en dos categorías: requisitos funcionales y requisitos no funcionales.

2.2.1 Requisitos funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas (Chaves, 2017). Fueron levantados 28 requisitos funcionales, que posteriormente fueron agrupados identificando funcionalidades similares sobre diferentes entidades (CRUD, siglas en inglés de *Create, Replace, Update, Delete*). A continuación, se relatan los requisitos funcionales identificados en conjunto con el cliente:

RF 1: Autenticar usuario: El Sistema permite al autenticar usuario.

RF 2: Gestionar usuario

RF 2.1: Insertar usuario: El Sistema insertar usuarios.

RF 2.2: Listar usuario: El Sistema permite visualizar una lista de usuarios.

RF 2.3: Editar usuario: El Sistema permite editar los usuarios.

RF 2.4: Eliminar usuario: EL Sistema permite eliminar los usuarios.

RF 3: Asignar roles: El sistema permite al administrador la asignación de roles.

RF 4: Gestionar Áreas

RF 4.1: Insertar Áreas: El Sistema permite al usuario insertar Áreas.

RF 4.2: Listar Áreas: El Sistema permite al usuario listar las Áreas.

RF 4.3: Editar Áreas: El Sistema permite al usuario modificar las Áreas.

RF 4.4: Eliminar Áreas: EL Sistema permite eliminar el objetivo que se haya Áreas.

RF 5: Gestionar Objetivo

RF 5.1: Insertar objetivo: El Sistema permite al usuario insertar objetivos.

RF 5.2: Listar objetivo: El Sistema permite al usuario listar los objetivos.

RF 5.3: Editar objetivo: El Sistema permite al usuario modificar los objetivos.

RF 5.4: Eliminar objetivo: El Sistema permite eliminar el objetivo que se haya alcanzado.

RF 6: Gestionar criterios de medidas

RF 6.1: Insertar criterios de medidas: El Sistema permite la creación de criterios de medidas asociado a un objetivo.

RF 6.2: Editar criterios de medidas: El Sistema permite la modificación de los criterios de medidas.

RF 6.3: Eliminar criterios de medidas: El Sistema permite la eliminación del criterio de medida.

RF 6.4: Listar criterios de medidas: El Sistema permite visualizar una lista de los criterios de medidas creados.

RF 7: Gestionar actividades

RF 7.1: Insertar actividades: El Sistema permite la creación de actividades asociado a un objetivo.

RF 7.2: Listar actividades: El Sistema permite listar las actividades.

RF 7.3: Editar actividades: El Sistema permite la modificación de las actividades.

RF 7.4: Eliminar actividades: El Sistema permite la eliminación de las actividades una vez ya realizadas.

RF 8: Subir evidencia: se sube una evidencia al sistema y la misma es indexada a la actividad.

RF 9: Descargar evidencia: una evidencia subida e indexada en el sistema se puede descargar.

RF 10: Visualizar estado del cumplimiento de los objetivos por ARC en por cientos: El Sistema permite visualizar gráficamente el estado de los objetivos actuales.

RF 11: Filtrar Objetivos

RF 11.1: Filtrar objetivos por estado de cumplimiento: El Sistema permite la filtrar los objetivos mostrando las opciones cumplidos, no cumplidos, cumplidos con señalamientos y sobre cumplido.

RF 11.2: Filtrar objetivos por ARC: El Sistema permite la filtrar los objetivos por ARC mostrando los cumplidos, no cumplidos, cumplidos con señalamientos y sobre cumplido del área perteneciente.

RF 12: Generar informe de objetivos de trabajo: El sistema debe generar el informe de cierre de los objetivos de trabajo. El sistema permite exportar los objetivos y métricas en formato PDF.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales se asocian con características que de una u otra forma puedan limitar el sistema, no describen información a guardar, ni funciones a realizar, sino características de funcionamiento (Chaves, 2017).

Usabilidad

RNF1. La usabilidad representa facilidad de uso por parte de los usuarios: El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de

manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.

Fiabilidad

RNF2. Políticas de seguridad por usuarios y roles: El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

Portabilidad

RNF3. El sistema debe funcionar en los sistemas operativos (Windows, Linux).

Software

RNF4. Las computadoras de los clientes sólo requieren de un navegador Mozilla Firefox o Google Chrome

Apariencia o interfaz externa

RNF5. El diseño de las interfaces será sencillo, con pocas imágenes y colores y la información aparecerá correctamente organizada de forma tal que el usuario, pueda encontrar lo que busca rápidamente.

Accesibilidad:

RNF6. Las opciones deben ser de fácil acceso, se utilizarán iconos conocidos y conceptos de navegación sencillos además de utilizar encabezados breves y descriptivos para estructurar la información.

Hardware

RNF7. Para la PC cliente se utilizó una PC Intel Pentium 4 o superior, CPU 1.2GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.

RNF8. Para la PC del servidor se utilizó una PC Intel Pentium 4 o superior, CPU 1.2GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.

2.3 Historias de Usuario

Las historias de usuario son requerimientos que expresan el problema que el sistema o producto software debe resolver. Se utilizan para especificar los requisitos de las aplicaciones software en las metodologías ágiles. Las historias de usuario son tarjetas en dónde el interesado describe brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que se

pueda implementar en unas tres semanas aproximadamente (para no superar el tamaño de una iteración, que es el tiempo estimado para una entrega de un componente de desarrollo de manera incremental (Suaza, 2013). Poseen las siguientes características: una descripción escrita que será utilizada para planificar y posteriormente disgregar los detalles con el dueño del producto, las conversaciones propiamente dichas con el dueño del producto y las pruebas que han de determinar si las historias están finalizadas o no. El resto de las HU generadas que representan funcionalidades que serán implementadas se pueden consultar en los Anexos.

En el proceso de desarrollo del SG se definió para mantener la organización, el nivel de prioridad para el negocio de cada uno de los requisitos obtenidos en la entrevista con el cliente. Los niveles se dividen en:

- **Alta:** Funcionalidades que son sumamente primordiales en la aplicación, que por su función no deberían faltar en el editor, es decir, la aplicación depende de estas para su correcto funcionamiento.
- **Media:** Funcionalidades que no son muy primordiales, pero otras dependen de estas para poder ejecutarse.
- **Baja:** Funcionalidades que su existencia en la aplicación no es significativa, es decir, la aplicación puede funcionar correctamente sin estas.

Tabla 10. Historia de usuario Insertar Áreas [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Insertar Áreas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	
Descripción: El usuario de debe dirigirse al link Áreas, una vez dado click, el sistema muestra un botón para agregar las áreas de resultados claves que se encuentra en la parte superior derecha.	

Observaciones: N/A
Prototipo de interfaz:

Tabla 11. Historia de Usuario listar Áreas [Fuente: Elaboración Propia]

Número: 9	**Nombre del requisito:** Listar Áreas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	
Descripción: El usuario puede observar una lista de las áreas insertadas.	
Observaciones: N/A	
Prototipo de interfaz:	

2.4 Plan de iteraciones

El Plan de Iteraciones (PI) es un artefacto obtenido dentro del proceso Planeación de Proyecto definido en el modelo CMMI-DEV v1.3. En concreto se elaboró el PI para la fase de Ejecución establecida por AUP UCI. Este artefacto permite organizar, escalar y/o planificar como van a ser entregadas las Unidades de desarrollo dentro de la mencionada fase. Para la obtención del mismo se realizaron las siguientes acciones:

1. Priorización de las HU.
2. Estimación del tiempo de desarrollo para HU basado en el riesgo y la experiencia previa en la implementación de funcionalidades similares.
3. Distribución de las HU por iteraciones teniendo en cuenta la prioridad que definió el cliente para cada HU.

A partir de los elementos mencionados anteriormente se obtuvo la siguiente planificación:

Tabla 12. Plan de iteraciones. [Fuente: Elaboración propia]

Iteración	Unidades de desarrollo	Justificación
1	Gestionar usuarios, Asignar roles, Gestionar ARC, Gestionar Objetivo, Gestionar Criterio de medidas, Gestionar actividad	Constituyen las HU con una prioridad Alta. Son la base para la funcionalidad del sistema.

2	Visualizar Estado, Generar informe	Constituyen las HU con una prioridad Media. Son funcionalidades no tan primordiales del sistema.
3	Autenticar, Filtrar objetivos, Subir evidencia, Descargar evidencia	Constituyen las HU con una prioridad Baja al ser las de menor importancia. Igualmente son fáciles de implementar o poseen un bajo riesgo.

2.5 Descripción del Patrón arquitectónico

En el desarrollo de la solución propuesta se utilizó el patrón arquitectónico Modelo Vista Vista Modelo (**MVVM**), este patrón es una variación del Modelo Vista Controlador que está diseñado para plataformas de desarrollo de interfaz de usuario moderna donde las vista es la responsabilidad de un diseñador en lugar de un desarrollador clásico, además **MVVM** también se basa en un mecanismo general de enlace de datos que facilita el desarrollo de la capa de separación de vista desde el resto del patrón mediante la eliminación del todo el código subyacente de la capa de la vista. El concepto fundamental de **MVVM** es separar el Modelo (Model) y la Vista (View) introduciendo una capa abstracta entre ellos, un "Modelo de la Vista" o "**ViewModel**". La vista y el modelo de la Vista son instanciados normalmente por la aplicación contenedora. MVVM tiene como objetivo que los datos trasladados a la vista se puedan presentar y gestionar de la manera más sencilla. Es el **ViewModel** quien expone los datos desde el modelo y, en este sentido, el **ViewModel** es más un modelo que una vista. Pero además gestiona la lógica de visualización de la vista por lo que esta, desde este punto de vista, es más una vista que un modelo (Bermeo, 2019).

2.5.1 Elementos del patrón MVVM

La finalidad principal del patrón MVVM (Modelo Vista Vista-Modelo) es tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación. Dentro de sus principales características se encuentra:

El modelo

Representa la capa de datos y/o la lógica de negocio, también denominado como el objeto del dominio. El modelo contiene la información, pero nunca las acciones o servicios que la manipulan. En ningún caso tiene dependencia alguna con la vista (Bermeo, 2019).

La Vista

Encapsula la interfaz de usuario y la lógica de la interfaz de usuario. Las vistas definen la interfaz de usuario específica para una parte de la aplicación; las vistas son normalmente sólo los controles que se han diseñado para funcionar bien cuando se une a un **ViewModel**. Su función específica es presentar al usuario una interfaz que representa los datos subyacentes de la aplicación y la lógica de la forma más adecuada, y para interpretar las acciones del usuario y las tramitará al **ViewModel** que define la forma en que afectan a los datos subyacentes y el estado de aplicación en el modelo. En algunos casos, la vista puede contener la lógica de la interfaz de usuario en la vista, el código subyacente. Sin embargo, tales códigos específicos de interfaz de usuario son difícil de probar y se debe evitar tanto como sea posible (Bermeo, 2019).

El View Model

Encapsula la lógica de presentación y estado. La lógica de presentación se define aquí como la lógica de la aplicación que se ocupa de casos de uso de la aplicación (o casos de usuario, tareas de usuario, flujo de trabajo, etc.) y define el comportamiento lógico y la estructura de la aplicación. El modelo de vista no es el código de un control de atrás. La lógica de presentación no debe tener idea de lo que la vista será y debe ser independiente de cualquier implementación específica de la interfaz de usuario. Para maximizar las oportunidades de reutilización, el ***ViewModel*** no debe tener ninguna referencia a las clases específicas de interfaz de usuario, elementos, controles o comportamiento (como animaciones). No se debe derivar de cualquier clase base de interfaz de usuario (lo ideal es que sea un poco) (Bermeo, 2019).

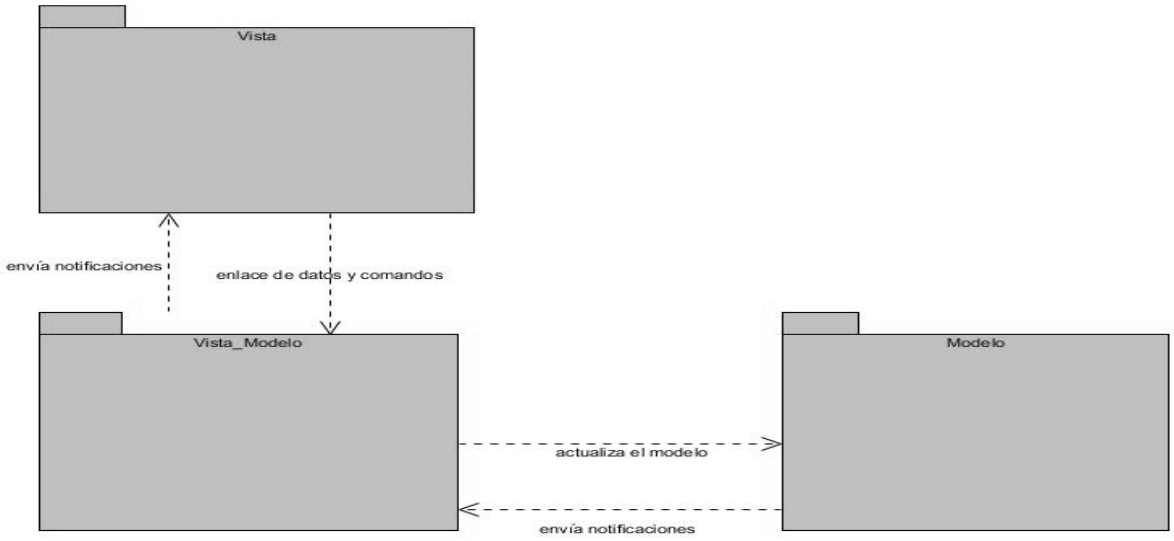


Figura 2. Arquitectura del sistema [Fuente: Elaboración propia]

2.6 Patrones de diseño

En el desarrollo de software el uso de patrones de diseño constituye una buena práctica debido a que proporciona una organización estructurada entendible por cualquier programador. El uso de patrones permite construir software que sean más fáciles de mantener y extender, representando un ahorro de tiempo. Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En lugar de reinventar soluciones se pueden utilizar patrones que representan soluciones ya probadas y documentadas a problemas de desarrollo de software que están sujetos a contextos similares tales como acceso a datos, creación de objetos y operaciones entre sistemas. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (Leiva, 2015).

2.6.1 Los patrones GRASP

Los patrones GRASP son un acrónimo que significa **General Responsibility Asignment Software Patterns** (patrones generales de software para asignar responsabilidades) (Leiva, 2015).

Patrón Experto: es unos de los patrones que más se utilizan cuando se trabaja con Laravel, para mapear la base de datos, Laravel cuando realiza su capa de abstracción en el modelo encapsula toda la lógica de los datos y genera las clases con todas las funcionalidades comunes de las entidades. Las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla correspondiente. Este patrón se puede observar en la clase *ObjetiveController* donde se asigna la responsabilidad de salvar todos los datos de los objetivos. Este patrón hace posible que se mantenga el encapsulamiento de la información, lo que conlleva a un bajo acoplamiento.

Patrón Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. Este patrón se puede observar en la clase *ActivityController* al ser esta la clase que crea todas las instancias de los objetos que se manejan la clase *Activity*, para luego realizar operaciones con dichas instancias

Patrón Bajo Acoplamiento: El Bajo Acoplamiento soporta el diseño de clases más independiente, reduce el impacto de los cambios más reutilizables y acrecienta la oportunidad de una mayor productividad. Este patrón se puede observar en la clase *GeneratePdfController* al encapsular la información, haciendo posible que esta clase sea la experta en la información, dependiendo lo menos posible de otras clases.

Patrón Alta Cohesión: Una clase cohesionada posee un número relativamente pequeño de responsabilidades, con una importante funcionalidad relacionada y poco trabajo por hacer. Este patrón se puede observar en la clase *GeneratePdfController* quien, para llevar a cabo la tarea de generar el Pdf, carga los datos a través de una instancia de la clase *ObjectiveController.*

Patrón Controlador: Representa al sistema completo y es el manejador artificial de los eventos. Consiste en la asignación de responsabilidades para controlar el flujo de los eventos que genera el sistema a clases específicas. Su uso se evidencia en la clase ObjectiveController.php pues es esta la clase encargada de instanciar todos los objetos de otras clases y además la de salvar todos los datos de los objetivos.

2.6.2 Los patrones GOF

Los patrones GOF (Pandilla de los Cuatro del inglés *Gang of Four*) fueron creados por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados por expertos diseñadores de software orientado a objetos. Estos patrones se clasifican en 3 categorías basadas en su propósito (Angel, 2013).

- Creacionales: Inicialización y configuración de objetos, ayudan a encapsular y abstraer dichos objetos.
- Estructurales: Separan la interfaz de la implementación, se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes.
- De comportamiento: Describen la comunicación entre objetos o clases.

En el desarrollo de la aplicación se utilizaron algunos patrones GOF, estos son:

Iterador: Este patrón pertenece a la categoría de comportamiento y permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. Este patrón se evidencia su uso en la clase *AreaController* en él método *destroy()* el cual elimina la(s) relación(es) entre el área y el usuario.

Singleton: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. La clase *AreaController* obtiene mediante el componente DB (*DataBase*) una única instancia de la clase para acceder a sus atributos y funcionalidades.

2.7 Diseño de la base de datos

El modelo de datos describe de forma abstracta las entidades y sus características, además de las relaciones entre estas dentro de la base de datos. Para este trabajo fue construido el Modelo Físico de Datos, cual proporciona una flexibilidad óptima para el soporte de la informatización.

A continuación, se muestra el diseño de la base de datos del sistema:

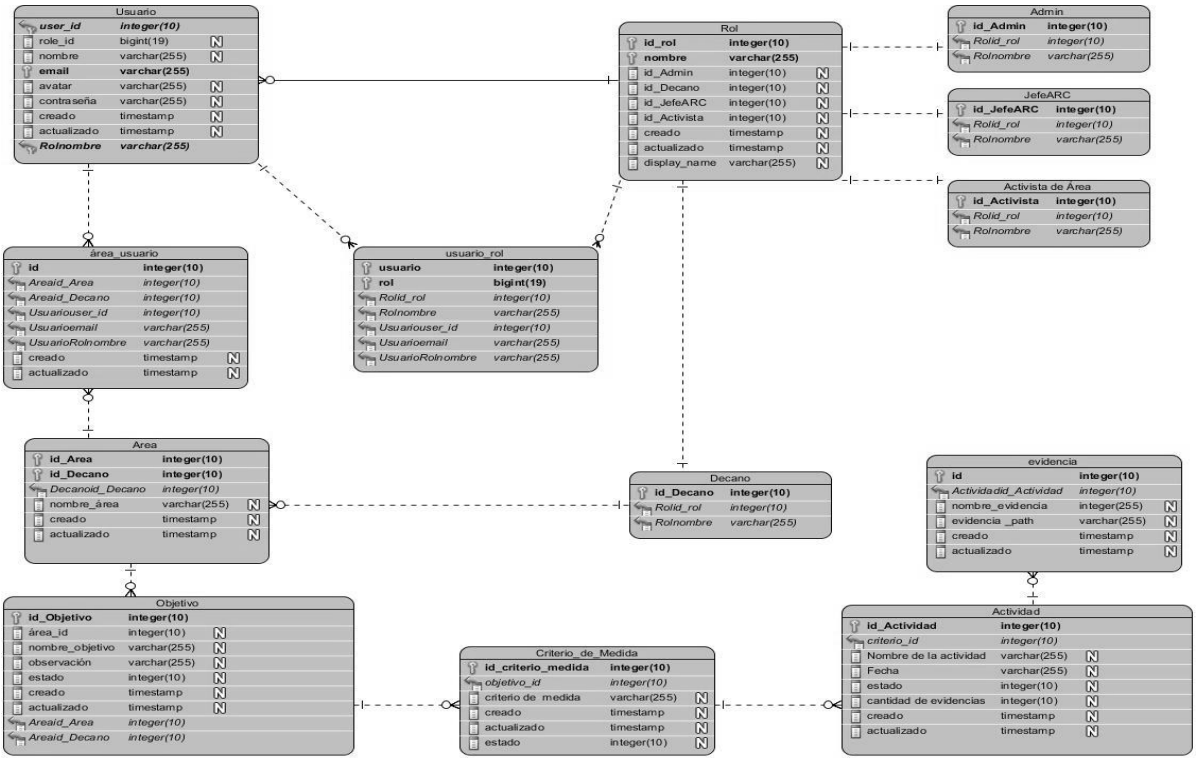


Figura 3. Diagrama entidad relación. [Fuente: Elaboración propia]

2.8 Conclusiones parciales

Luego de haber concluido el presente capítulo se arribó a las siguientes conclusiones:

- Se definió la propuesta de solución por lo cual se obtuvieron una mejor comprensión de la estructura y comportamiento del sistema a implementar.

- A partir de la elaboración de las historias de usuarios se definen las funcionalidades que brindará el sistema, participando en este proceso el cliente, por lo que se garantiza que dichas funcionalidades respondan a las necesidades reales.

Capítulo 3 Implementación y pruebas

En el presente capítulo se presentan los resultados de la validación del sistema, es decir, se describe cómo realizar una correcta implementación a partir del análisis y diseño, además se discuten los resultados alcanzados al aplicar las pruebas de caja blanca y de caja negra. Para ello se crea el diagrama de despliegue representando los recursos físicos necesarios para el despliegue de la aplicación, diagrama de componentes, se aplica las pruebas de carga y estrés donde se muestra los resultados del rendimiento.

3.1 Diagrama de despliegue

El diagrama de despliegue representa la arquitectura de tiempo de ejecución de los procesadores, dispositivos y los componentes de software que se ejecutan en esa arquitectura. Es una descripción física de la topología del sistema y describe la estructura de las unidades de hardware. Además, representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación (Burrows, Visual Parading, 2019). Un nodo es un recurso de ejecución tal como un procesador, un dispositivo o memoria. En los procesadores es donde se encuentran alojados los componentes. La Figura que se muestra a continuación representa el diagrama de despliegue del Sistema de Gestión para el cumplimiento de los objetivos estratégicos de la Facultad 4 y está compuesto por:

PC_Cliente: Representa una computadora desde la cual el usuario podrá acceder a la aplicación.

Servidor web (Apache): Establece comunicación con la PC_Cliente utilizando el protocolo HTTPS y realiza peticiones mediante TCP/IP al sevidor.

Servidor de base de datos: Representa el servidor donde estará el SGBD que da repuesta a las peticiones realizadas por la aplicación.

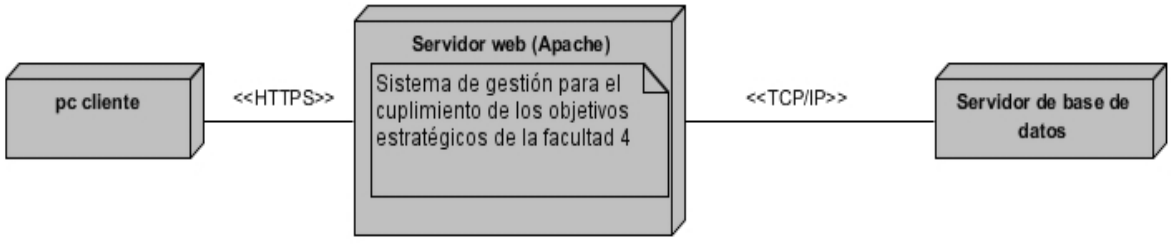


Figura 4. Diagrama de despliegue. [Fuente: Elaboración propia]

3.2 Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema (Pressman, 2005). El diagrama de componentes se puede consultar en los anexos.

Descripción de los componentes más relevantes:

Template: Contiene los archivos de la estructura del sistema.

Component: Contiene todos los archivos con código JavaScript del sistema.

Css: Contiene todo el código css del tema, código utilizado para las vistas del sistema.

Service: Contiene los módulos de servicio del sistema.

A continuación, se muestra el diagrama de componentes del sistema:

3.3 Pruebas del sistema

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software antes de su puesta en marcha. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. Existen varios tipos de pruebas que suelen ser utilizadas por los equipos de probadores. Las pruebas de caja negra y de caja blanca se encuentran en función de lo que se conoce. Por otra parte, las de carga y estrés, las unitarias y funcionales se encuentran en dependencia de lo que se prueba (Javier J. Gutiérrez, 2016).

Pruebas de Caja Negra: Son aquellas orientadas a los requisitos funcionales, buscan que se haya ingresado toda clase de entrada y que la salida obtenida es igual a la esperada.

Pruebas de Caja Blanca: Consisten en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido.

3.3.1 Pruebas Funcionales

Se denominan pruebas funcionales a las pruebas de software que tienen como objetivo fundamental asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados (Milton Eduardo Escobar Sánchez, 2015).

Las metas de estas pruebas son:

- Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio.

- Verificar la apropiada aceptación de datos.

Para la realización de estas pruebas se utilizó el método de caja negra haciendo uso de la técnica partición de equivalencia que se encarga de examinar los valores válidos e inválidos de las entradas existentes en el sistema (Sommerville, 2011). A continuación, se muestran algunos casos de pruebas basados en requisitos que son de importancia para el cliente.

Tabla 13. Caso de prueba: Visualizar estado del cumplimiento de los objetivos por ARC en por cientos [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Total de criterios de medidas**	**Criterios de medidas cumplidos**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Objetivos sobre cumplidos.	El Sistema permite visualizar gráficamente el por ciento de cumplimiento de los objetivos actuales por área.	4	4	El sistema muestra una gráfica del estado de cumplimiento de los objetivos sobre cumplidos al 100%.	1. Actividades cumplidas. 2. Criterios de medidas cumplidos. 3. Estado de cumplimiento del objetivo por ARC.
EC 1.2 Objetivos cumplidos.	El Sistema permite visualizar gráficamente el por ciento de cumplimiento de los	7	6	El sistema muestra una gráfica del estado de cumplimiento de los objetivos cumplidos	1. Actividades cumplidas. 2. Criterios de medidas cumplidos. 3. Estado de cumplimiento

	objetivos actuales por área.			del >95% <100%.	del objetivo por ARC.
EC 1.3 Objetivos cumplidos con señalamiento.	El Sistema permite visualizar gráficamente el por ciento de cumplimiento de los objetivos actuales por área.	4	3	El sistema muestra una gráfica del estado de cumplimiento de los objetivos con señalamiento 75%> al <85%.	1. Actividades cumplidas. 2. Criterios de medidas cumplidos. 3. Estado de cumplimiento del objetivo por ARC.
EC 1.4 Objetivo incumplido.	El Sistema permite visualizar gráficamente el por ciento de cumplimiento de los objetivos actuales por área.	4	2	El sistema muestra una gráfica del estado de cumplimiento de los objetivos incumplidos <75%.	1. Actividades cumplidas. 2. Criterios de medidas cumplidos. 3. Estado de cumplimiento del objetivo por ARC.

3.3.2 Resultados de las pruebas funcionales

Se aplicaron los casos de pruebas correspondientes a las funcionalidades del sistema de Gestión durante tres iteraciones de pruebas, se detectaron en cada una de estas 10, 6 y 3 no conformidades respectivamente, llegando al total de 19 no conformidades, cabe añadir que de la primera iteración quedaron dos no conformidades sin resolver las cuales se adicionaron a la segunda iteración y en esta iteración quedó una, la cual se adicionó a la tercera. Las pruebas realizadas revelaban errores en la documentación, codificación y en

el diseño de las interfaces de usuario que en la mayoría de los casos no incidían significativamente en las respuestas esperadas. Todas estas dificultades fueron solucionadas en el menor tiempo posible, siempre antes de pasar a la iteración siguiente. Una vez llegada a la solución final, se le demostró al cliente el cumplimiento de todos los requisitos solicitados.

3.3.3 Pruebas Unitarias

Las pruebas unitarias se utilizan para detectar errores en el código, pues se centran en la menor unidad de diseño del software (módulos, método de una clase, clases). Tiene como objetivo principal detectar errores en los principales procesos al ser ejecutado de manera independiente. Estas son diseñadas por los programadores y realizadas ante cualquier modificación del sistema para detectar fallas que pudieran ocurrir (Herrmann, 2011).

Las pruebas de unitarias proporcionan algunas ventajas como que impulsa al desarrollador a crear una estructura de código más simple (lo que se conoce como refactorización). Simplifica la integración, pues permite llegar a la fase de integración con un código que funcione correctamente y facilita un mejor entendimiento del código ya que lo documenta a través de las pruebas. Estas siempre están orientadas a las técnicas de caja blanca (Pressman, 2005).

Método de Caja blanca

El método de caja blanca, también conocido como pruebas de caja transparente o pruebas estructurales, se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente (Peño, 2015). La prueba de Caja Blanca es uno de los tipos de pruebas más importantes que se le aplican al software logrando como resultado que disminuya en un gran porciento el número de errores existentes en el sistema. Para lograr una mayor calidad y confiabilidad se utilizó la técnica de camino básico el cual se centra exclusivamente en la validez de las construcciones de bucles. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico (Pressman, 2005). Para ello es necesario conocer el número de caminos independientes, por lo que se construirá un grafo asociado y se calculará de la complejidad ciclomática.

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y brinda información para corroborar que el trabajo se está haciendo adecuadamente. Los componentes son:

- **Nodo**: Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.
- **Flechas**: Las flechas del grafo se denominan aristas y representan el flujo de control. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.
- **Regiones**: Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

A continuación, se analiza el código y se enumeran las sentencias de código del procedimiento realizado sobre la función *modicarCampoCriteriodemedida($id)* el cual es el encargado de modificar el campo de criterios de medida.

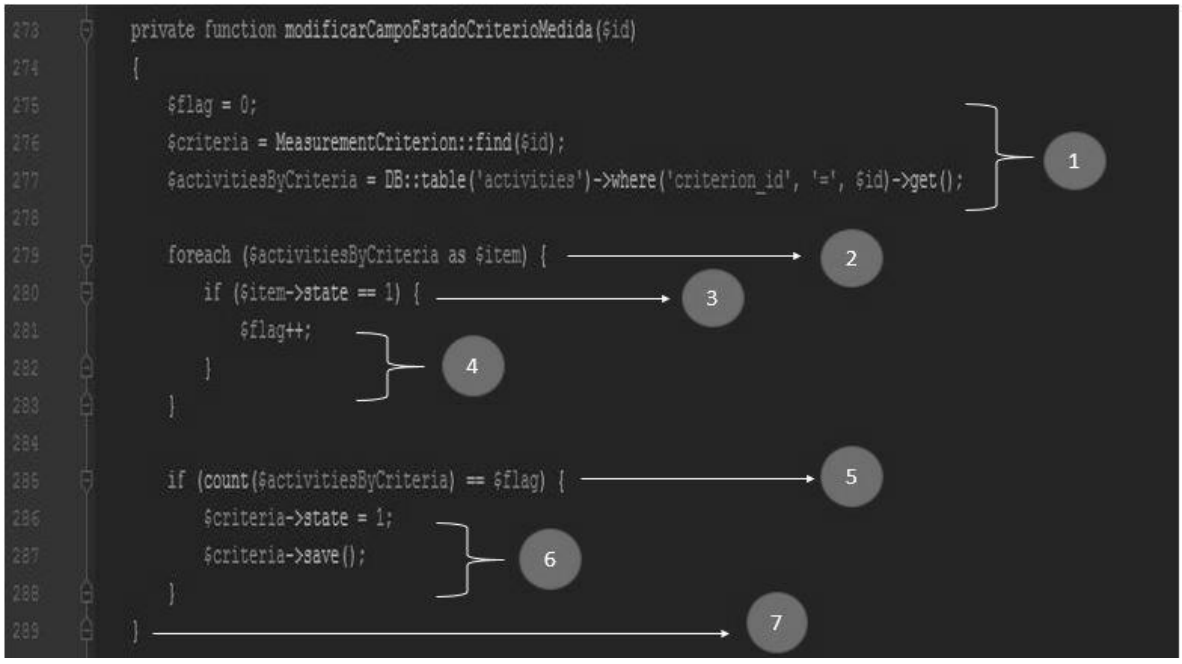


Figura 5. Método modicarCampoCriteriodemedida($id) [Fuente: Elaboración propia]

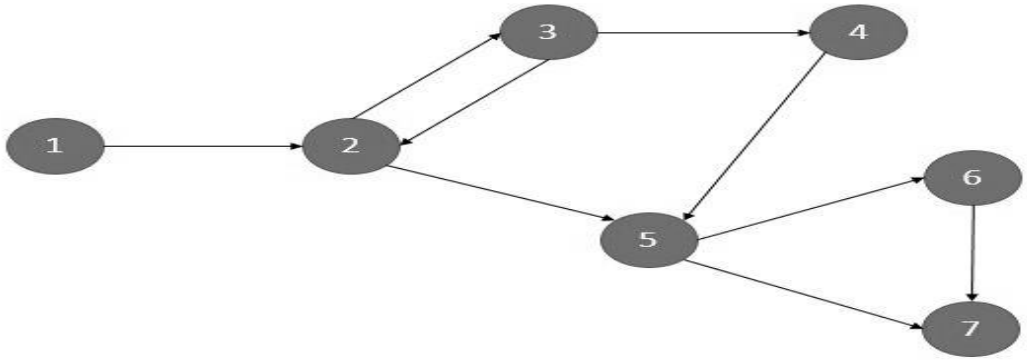


Figura 6. Grafo del flujo asociado al método modificar criterio de medida [Fuente: Elaboración propia]

Fórmulas para calcular la complejidad ciclomática:

1. V(G) = (A - N) + 2

   Donde “A” es la cantidad de aristas y “N” es la cantidad de nodos.

   V(G) = (9 - 7) + 2

   V(G) = 4

2. V(G) = P + 1

   Siendo “P” la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

   V(G) = 3 + 1

   V(G) = 4

3. V(G) = R

   Donde “R” representa la cantidad de regiones en el grafo.

   V(G) = 4

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es 4, lo que significa que existen cuatro

posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 14. Camino básico del flujo asociado al método modificar campo estado criterio de medida [Fuente: Elaboración Propia]

Números	Caminos básicos
1	1-2-3-4-5-7
2	1-2-3-4-5-6-7
3	1-2-5-7
4	1-2-5-6-7

Tabla 15. Caso de prueba unitaria [Fuente: Elaboración Propia]

Caso de prueba
Camino: 1-2-3-4-5-6-7
Entrada: recibe id de criterio de medida.
Resultado: se actualiza satisfactoriamente el campo estado de criterio de medida.
Condiciones: debe existir el criterio de medida pasado por parámetro.

Tabla 16. Caso de prueba unitaria [Fuente: Elaboración Propia]

Caso de prueba
Camino: 1-2-3-4-5-7
Entrada: recibe id de criterio de medida
Resultado: no se actualiza el campo estado de criterio de medida.

Condiciones: no debe de haber la misma cantidad de criterios que de criterios con campo estado en 1.

Tabla 17. Caso de prueba unitaria [Fuente: Elaboración Propia]

Caso de prueba
Camino: 1-2-5-7
Entrada: recibe id de criterio de medida.
Resultado: no se actualiza el campo estado de criterio de medida.
Condiciones: no debe existir el criterio de medida pasado por parámetro.

Tabla 18. Caso de prueba unitaria [Fuente: Elaboración Propia]

Caso de prueba
Camino: 1-2-5-6-7
Entrada: recibe id de criterio de medida.
Resultado: se actualiza el campo estado de criterio de medida.
Condiciones: el criterio de medida no tiene actividades asociadas.

3.4 Pruebas de carga y estrés

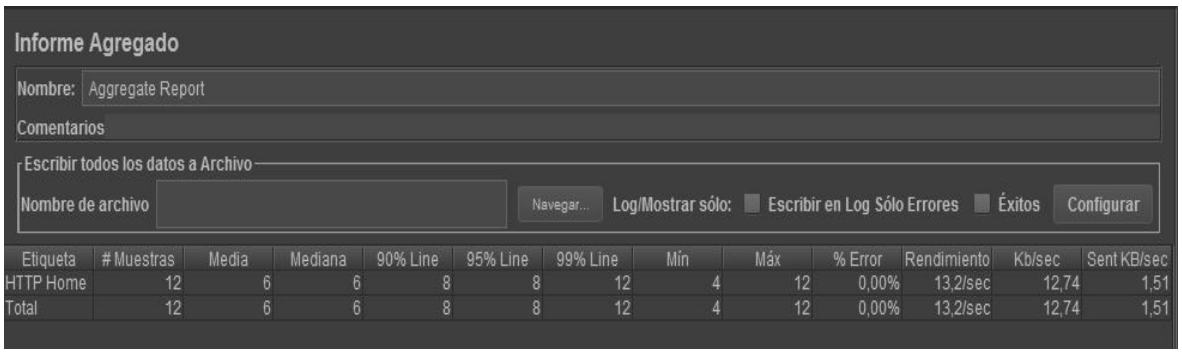
Estas pruebas se diseñan para enfrentar a los sistemas a situaciones anormales, es decir, ejecutar el sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Igualmente busca validar el correcto funcionamiento del sistema bajo las condiciones de carga normales para la operación.

Para la ejecución de las pruebas se utilizó una computadora con las características:

- Microprocesador Intel Corei3.

- Memoria RAM de 4 GB.
- Disco duro de 500 GB.
- Sistema Operativo Xubuntu 14.10.

La prueba de carga y estrés realizada a través de la herramienta Apache Jmeter, brinda los resultados del rendimiento del sistema. Para ello se realizó una prueba donde para 12 usuarios conectados concurrentemente en un intervalo de 1 segundo, se obtuvo lo mostrado a continuación en el reporte resumen generado por la herramienta utilizada.



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
HTTP Home	12	6	6	8	8	12	4	12	0,00%	13,2/sec	12,74	1,51
Total	12	6	6	8	8	12	4	12	0,00%	13,2/sec	12,74	1,51

Figura 7. Reporte de resumen [Fuente: Herramienta JMeter]

Los resultados de las pruebas de carga y estrés se consideran satisfactorios, debido a que los tiempos de respuesta del servidor ante la interacción de 12 usuarios concurrentes se encuentra en el rango de tiempo de 1 a 3 segundos. La propuesta de solución generó un total de 12,74 kb/seg de transferencia de datos para 12 usuarios, con un rendimiento de 13,2 seg. Con ello queda demostrado que la propuesta de solución es estable, ya que se mantuvo prestando servicios todo el tiempo.

3.5 Conclusiones Parciales

Al concluir este capítulo se termina la realización de la aplicación, obteniendo los siguientes resultados:

- La elaboración del diagrama de componentes representó la vista de implementación del sistema y obtuvo al finalizar la fase de implementación del producto final con las funcionalidades requeridas.
- El diagrama de despliegue mostró los nodos procesadores la distribución de los procesos y de los componentes.

- La realización de las pruebas al sistema desarrollado, permitió detectar y corregir las no conformidades presentes en el mismo por tanto los resultados obtenidos son satisfactorios.

Conclusiones generales

Luego de culminada la presente investigación se puede arribar a las siguientes conclusiones:

- El estudio realizado como parte de la fundamentación sirvió de apoyo en la toma de decisiones con vista al desarrollo de la propuesta de solución, destacando los aspectos positivos identificados en los sistemas homólogos. Estos aspectos fueron traducidos en requisitos del sistema propuesto.
- El empleo de la metodología, las herramientas, tecnologías y lenguajes de desarrollo seleccionados soportaron todo el proceso de desarrollo de la propuesta de solución. Además, los artefactos generados en correspondencia con la metodología de desarrollo AUP en su variante UCI, facilitaron la comprensión e implementación del sistema desarrollado.
- Los requisitos analizados, el diseño propuesto y la implementación lograda permitieron obtener un sistema que centraliza los datos del proceso, logra la disponibilidad de la información para los usuarios responsables de las ARC, permite la toma de decisiones y la aplicación de medidas correctivas en el caso de atrasos en las actividades planificadas, cumpliendo de esta forma el objetivo definido en la presente investigación.
- La ejecución de las pruebas unitarias y funcionales para la verificación del adecuado funcionamiento del sistema, permitió corregir los errores detectados tras la ejecución de 28 casos de prueba validando así la propuesta de solución.

Recomendaciones

Para el desarrollo de un trabajo de diploma se precisa de un tiempo determinado. Considerando ese tiempo breve para darle completitud a la presente investigación y considerando a la UCI como albacea de los resultados de la misma, se sugieren los siguientes elementos para continuar mejorando la solución propuesta:

- Desarrollar mecanismos para la integración del SG con sistemas de la Universidad como AKADEMOS, que constituye una valiosísima fuente de información al tener informatizado varios de los procesos de la Universidad, dígase Investigación, Formación y el módulo de Personal.
- Extender y enriquecer el SG para que se pueda gestionar desde el mismo el Plan Anual Individual de Resultados de los trabajadores de la UCI y este se convierta en la principal fuente de información para medir el cumplimiento de los objetivos estratégicos de la organización.
- Se considera que el software propuesto tiene un alto impacto en el proceso de Planeación Estratégica de una organización, por lo que conducir la solución hacia un sistema genérico y adaptable a otros entornos o escenarios podría constituir una fuente de ingresos para la Universidad y estaría contribuyendo a la gestión de organizaciones basadas en las tecnologías e indirectamente a la Informatización de la sociedad cubana.

Referencia Bibliográfica

- ✓ Alberto Basalo, M. A. (2016). *Manual de AngularJS.* Canada: https://programacion.net.
- ✓ Almira, L. t. (2012). *Intranet del Ministerio de Informática y las Telecomunicaciones.Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* La Habana.
- ✓ Alyona Stashkova, C. P. (2016). *Developing Applications with NetBeans IDE.* EE.UU: https://docs.oracle.com.
- ✓ Angel. (15 de Abril de 2013). *Geek the planet*. Obtenido de Geek the planet: http://geektheplanet.net/5462/patrones-gof.xhtml
- ✓ Astudillo, M. V. (2017). *Fundamentos de Ingeniería de Software.* España.
- ✓ AULBACH, S. S. (2015). *Manual de PHP.* Grupo de documentación de PHP, [Disponible en: www.php.net] .
- ✓ Bermeo, J. A. (13 de Abril de 2019). *espoch*. Obtenido de espoch: http://dspace.espoch.edu.ec/handle/123456789/3559
- ✓ Borges, S. (30 de Marzo de 2018). *infranetworking*. Obtenido de infranetworking: https://blog.infranetworking.com/servidor-web/
- ✓ Burrows, W. (3 de Abril de 2019). *Visual Parading*. Obtenido de Visual Parading: https://www.visual-paradigm.com/tutorials/
- ✓ Burrows, W. (20 de Abril de 2019). *Visual Parading*. Obtenido de Visual Parading: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/;WWWSESSIONID=5AACBB92F3E9FD2509D3C2FB7934142E.www1
- ✓ Caballero, J. J. (2016). *Modelo de procesos para el desarrollo del front-end.* Lima, Perú.
- ✓ Chaves, M. A. (10 de Abril de 2017). *redalyc*. Obtenido de redalyc: http://www.redalyc.org/articulo.oa?id=66612870011
- ✓ Christopher Pitt, T. O. (2014). *Laravel4Cookbook(ES).*
- ✓ Codina, J. (2017). *¿Qué es Bootstrap? principios básicos.* Barcelona: https://www.drauta.com/que-es-bootstrap-principios-basicos.
- ✓ Costa, C., Domínguez, J., Hernández, J., Leiva, A. M., & Verdú, F. (2012). *Cuadro de mando integral (Balanced Scorecard).* España: https://www.ucipfg.com.
- ✓ F. Sierra, J. A. (2015). *Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web.* Venenzuela: http://publicaciones.unisimonbolivar.edu.co/rdigital/inovacioning/index.php/identic/article/view/73/91.

Bibliografía

- ✓ Fabien Potencier, F. Z. (26 de Marzo de 2019). *Uniwebsidad*. Obtenido de Uniwebsidad: https://uniwebsidad.com/libros/symfony-1-4
- ✓ Flask, R. (2016). *Java for Beginners .* Malta: http://staff.um.edu.mt.
- ✓ Flowdesk ERP. (10 de febrero de 2019). *Flowdesk ERP*. Obtenido de Flowdesk ERP: http://flowdesk.pe
- ✓ Fúquene, M. E. (2014). *Administración por objetivos, una estrategia que le permite a la organización alcanzar sus metas de forma conjunta.* Bogotá: https://repository.unimilitar.edu.co.
- ✓ Gauchat, J. D. (2012). *El gran libro de HTML5, CSS y javaScript.* Barcelona: https://gutl.jovenclub.cu.
- ✓ Gauchat, J. D. (2012). *HTML,CSS3 y JavaScript.* (España),Barcelona.
- ✓ Girardi, T. (2009). *Netbeans.*
- ✓ González, U. G. (2017). *JavaScript,¡Inspírate!*
- ✓ Guevara, L. V. (2018). *Gestión de Bases de Datos Versión 1.0.* Sevilla: http://buildmedia.readthedoc.org.
- ✓ Guzmán, D. M. (2015). *Ámbitos de la integración de sistemas de gestión.* Ecuador: http://portal.uasb.edu.ec.
- ✓ Haverbeke, M. (2018). *Eloquent JavaScript .*
- ✓ Herrmann, E. (2011). *Pruebas de Software: Herramientas: Pruebas Unitarias.*
- ✓ Hrider. (2019). *Software de Evaluación del Desempeño y Feedback 360º.* España: https://www.hrider.net.
- ✓ International, V. P. (2014). *Visual Paradigm Searchfor Visual Paradigm.*
- ✓ Javier J. Gutiérrez, M. J. (2016). *Implementación de pruebas del sistema. Un caso práctico .* Sevilla: http://www.lsi.us.es.
- ✓ John Freddy Vega, C. V. (2011). *Guía HTML5 El presente de la web.*
- ✓ Juan Manuel Gimeno, J. L. (2011). *Introducción a Netbean .*
- ✓ kinderman, H. (2011). *PostgresSQL Ventajas y Desventajas.*
- ✓ Kioskea. (2015). *«Patrones de diseño».*
- ✓ Larman, C. (9 de Abril de 2019). *ResearchGate*. Obtenido de ResearchGate: https://www.researchgate.net/publication/44470322_UML_y_patrones_introduccion_al_analisis_y_diseno_orientado_a_objetos_Craig_Larman
- ✓ Leiva, A. (12 de Abril de 2015). *devexperto*. Obtenido de devexperto: https://devexperto.com/patrones-de-diseno-software/
- ✓ Maida, E. G. (2015). *Metodología de desarrollo de software.* Argentina: Argentina: Universidad Católica Argentina, 2015.

Bibliografía

- ✓ Manuel Kiessling, H. A. (2015). *The Node Beginner Book.* United States: https://www.nodebeginner.org/.
- ✓ Marín, R. (28 de Marzo de 2019). *RevistaDigital*. Obtenido de RevistaDigital: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/
- ✓ Medrano, J. a. (2016). *Arquitectura de Datos.*
- ✓ Mestras, J. P. (2013). *PHP, Aplicaciones Web/Sistemas Web.* Madrid: https://www.fdi.ucm.es.
- ✓ Mestras, J. P. (2014). *Aplicaciones Web/Sistemas Web.* España: https://www.fdi.ucm.es/profesor/jpavon/web/26-Bootstrap.pdf.
- ✓ Microsoft. (2017). *Diagramas de componentes de UML.* https://msdn.microsoft.com/es-es/library/dd409390.aspx.
- ✓ Milton Eduardo Escobar Sánchez, W. M. (2015). *Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2.* Ecuador: http://dialnet.unirioja.es.
- ✓ Mozilla. (12 de Mayo de 2017). *Mozilla*. Obtenido de Mozilla: https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Introducci%C3%B3n#%C2%BFQu%C3%A9_es_JavaScript
- ✓ Naranjo, F. J. (2015). *Sistemas de Gestión: Valor Estratégico de las Organizaciones.* España: http://blog.seidor.com/infraestructura/sistemas-de-gestion-valor-estrategico-de-las-organizaciones/.
- ✓ Nodejs. (7 de Mayo de 2019). *nodejs*. Obtenido de nodejs: https://nodejs.org
- ✓ Odiorne, G. S. (1977). *Administración por objetivos. Nuevo Sistema para la dirección.* Mexico: https://www.worldcat.org/title/administracion-por-objetivos-nuevo-sistema-para-la-direccion/oclc/880987278?referer=di&ht=edition.
- ✓ Patricia Pérez Lorences, L. G. (2013). *La construcción de un cuadro de mando integral de tecnologías de la información en una empresa.* Santa Clara, Cuba: https://dialnet.unirioja.es.
- ✓ Patrick Thibodeau, B. M. (2 de Abril de 2016). *SearchERP*. Obtenido de SearchERP: https://searcherp.techtarget.com/definition/CASE-computer-aided-software-engineering
- ✓ Peño, J. M. (2015). *Pruebas de Software. Fundamentos y Técnicas .* Madrid, España.
- ✓ Pérez Pérez, M. S., & Segura Tristá, A. (2015). *Componente Configuración en la versión 2.0 del Sistema de Gestión Universitaria.* Habana. Obtenido de http:/repositorio.uci.cu/jspui/handle/123456789/7129
- ✓ PÉREZ, E. E. (2012). *Aplicación de administración por objetivos como técnica de cambio organizacional en una empresa.*
- ✓ Pérez, J. E. (2009). *Introducción a CSS .*

- ✓ phpMyAdmin. (29 de Marzo de 2019). *phpMyAdmin*. Obtenido de phpMyAdmin: https://www.phpmyadmin.net/
- ✓ Pressman, R. S. (2005). *Ingeniería de software. Un enfoque práctico Séptima Edición.* University of Connecticut.
- ✓ Reddin, W. J. (2012). *Administracion por objetivos.*
- ✓ RIVERA, K. S. (2017). *"La administración por objetivos y su relación con la productividad en la empresa Clariant SA, Callao, año 2017" .*
- ✓ Rouse, M. (2018). *Integrated development environment (IDE) .* Beijing: https://searchsoftwarequality.techtarget.com/definition/integrated-development-environment.
- ✓ Sánchez, J. A. (2018). *Desarrolla aplicaciones con Vuejs.* España: https://www.pdf-manual.es.
- ✓ Sánchez, M. Á. (4 de Mayo de 2009). *Medium*. Obtenido de Medium: https://medium.com/all-you-need-is-clean-code/patrones-de-dise%C3%B1o-b7a99b8525e
- ✓ Sánchez, T. R. (2014). *Metodología de desarrollo para la Actividad de la UCI.* La Habana.
- ✓ Sansón, C. R. (2019). *Aplicación Web ONG.* Barcelona: https://upcommons.upc.edu.
- ✓ serprogramador.es. (26 de enero de 2019). *serprogramador.es*. Obtenido de serprogramador.es: https://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/
- ✓ Siigo, C. d. (2018). *Sistema de Gestión.*
- ✓ Sommerville, I. (2011). *INGENIERÍA DE SOFTWARE Novena edición .* México: Pearson Educación de México, S.A. de C.V.
- ✓ Suárez Hernández, A., & Cepero Rojas, D. (2008). *Arquitectura para Akademos 2.0.* Habana: https://repositorio.uci.cu/handle/ident/TD_1643_08. Obtenido de https://repositorio.uci.cu/handle/ident/TD_1643_08
- ✓ Suaza, K. V. (2013). *Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software .* Medellín, Colombia .
- ✓ Taylor Otwell. (25 de Marzo de 2019). *laravel.com*. Obtenido de laravel.com: https://laravel.com/
- ✓ Torres, J. A. (2018). *ES6. El remozado JavaScript.* 17 Mayo.
- ✓ Tutorialspoint. (2015). *Database management system.* India: http://www.kciti.edu/wp-content/uploads/2017/07/dbms_tutorial.pdf.
- ✓ Universidad de las Ciencias Informáticas . (2019). *GESPRO 13.05.* La Habana: https://www.uci.cu/investigacion-y-desarrollo/productos/xedro/gespro-1305.

✓ Ynfante, R. (2009). *Administración por Objetivos(APO).*

Anexos

Anexo 1: Historias de usuario

Tabla 19. Historia de Usuario Autenticar Usuario. [Fuente: Elaboración Propia]

Número: 1	**Nombre del requisito:** Autenticar usuario
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 3ra
Prioridad: Baja	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	

Descripción:

El usuario inserta su usuario y contraseña se dirige al botón aceptar y el Sistema valida que los datos insertados son correctos para poder entrar al sistema

Observaciones:

- El usuario debe autenticarse con usuario y contraseña.
- Si el usuario introduce la información de forma incorrecta, el sistema emite un mensaje notificando el error.
- Si el usuario deja algún campo vacío el sistema emite un mensaje notificando el error

Prototipo de interfaz:

Tabla 20. Historia de Usuario Insertar Objetivo. [Fuente: Elaboración Propia]

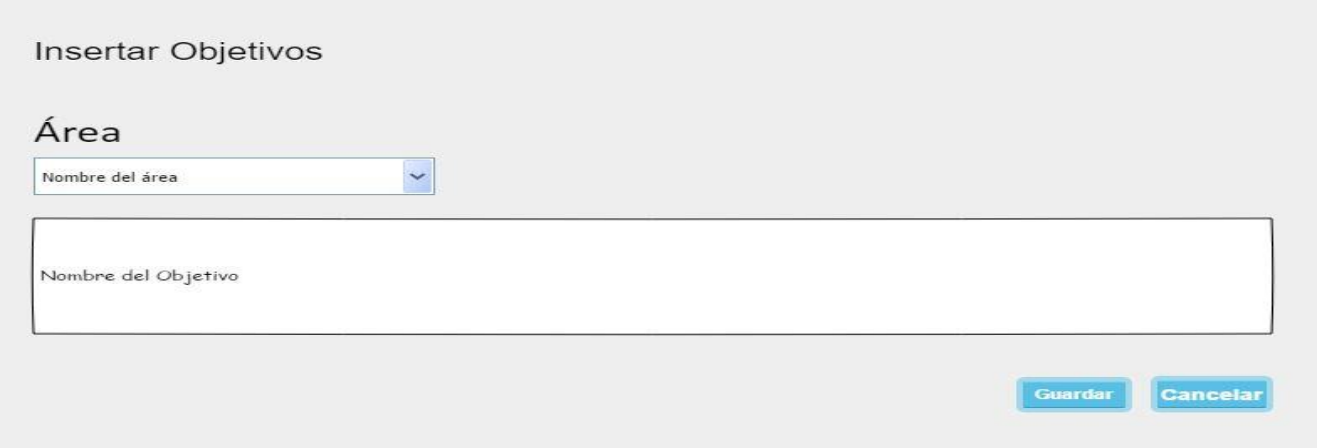
Número: 2	**Nombre del requisito:** Insertar objetivo
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario se dirige al link objetivos que se encuentra en la cabeza de la página una vez dado un click se muestra un botón con un signo de más el cual nos muestra donde insertar los objetivos por área.	
Observaciones: El usuario debe introducir el objetivo y los criterios de medidas asociados al objetivo. Si el usuario deja un campo vacío el sistema emite un error de formulario.	
Prototipo de interfaz: 	

Tabla 21. Historia de Usuario Eliminar Objetivo. [Fuente: Elaboración Propia]

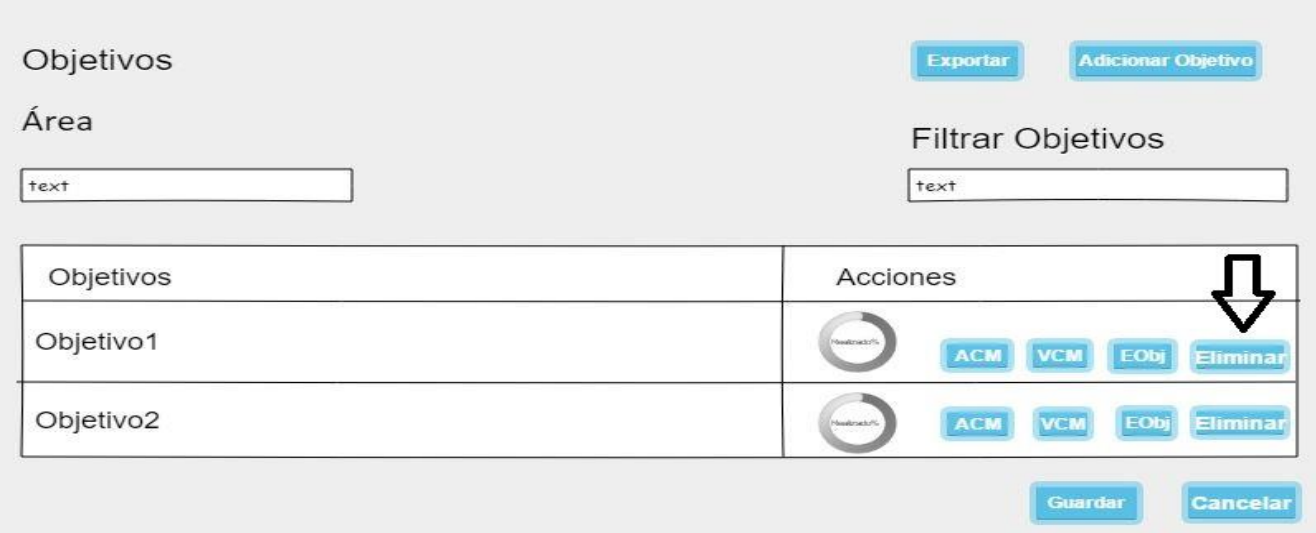
Número: 3	**Nombre del requisito:** Eliminar Objetivo
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario puede visualizar el botón eliminar, al darle click, el objetivo se elimina.	
Observaciones: El sistema no permite eliminar los objetivos si tiene criterios de medidas asociados, este emite un error de criterios de medidas asociadas.	
Prototipo de interfaz: 	

Tabla 22. Historia de Usuario Editar Objetivo. [Fuente: Elaboración Propia]

Número: 4	**Nombre del requisito:** Editar objetivos
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario puede visualizar el botón editar el cual al darle click le mostrara la interfaz de editar los objetivos.	
Observaciones: N/A	
Prototipo de interfaz:	

Tabla 23. Historia de Usuario Listar Objetivo. [Fuente: Elaboración Propia]

Número: 5	**Nombre del requisito:** Listar objetivos
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario puede visualizar una lista de objetivos que pertenecen al área.	
Observaciones: N/A	
Prototipo de interfaz:	

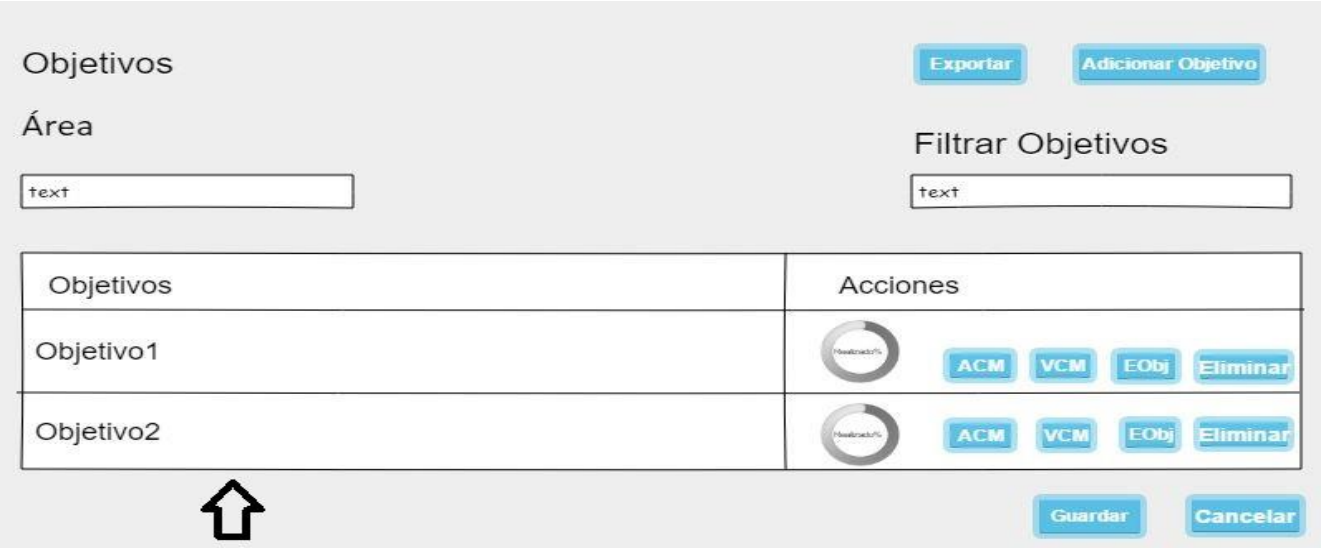


Tabla 24. Historia de Usuario Eliminar Área. [Fuente: Elaboración Propia]

Número: 7	**Nombre del requisito:** Eliminar Área
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	
Descripción: El usuario puede eliminar las áreas dando click al botón que se encuentra debajo del campo acciones.	
Observaciones: Las áreas no se pueden eliminar si tienen objetivos asociados. Este emite un error de que tiene objetivos asociados.	
Prototipo de interfaz:	

Tabla 25. Historia de Usuario Editar Área. [Fuente: Elaboración Propia]

Número: 8	**Nombre del requisito:** Editar Áreas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Medio	
Descripción: El usuario puede editar las áreas dando click al botón que se encuentra debajo del campo acciones.	
Observaciones: N/A	
Prototipo de interfaz:	

Tabla 26. Historia de Usuario Insertar criterios de medidas. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Insertar criterios de medidas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario al insertar un objetivo crea los criterios de medidas.	
Observaciones: Es obligatorio insertar el campo de criterios de medidas. El sistema emite un error de formulario.	
Prototipo de interfaz:	

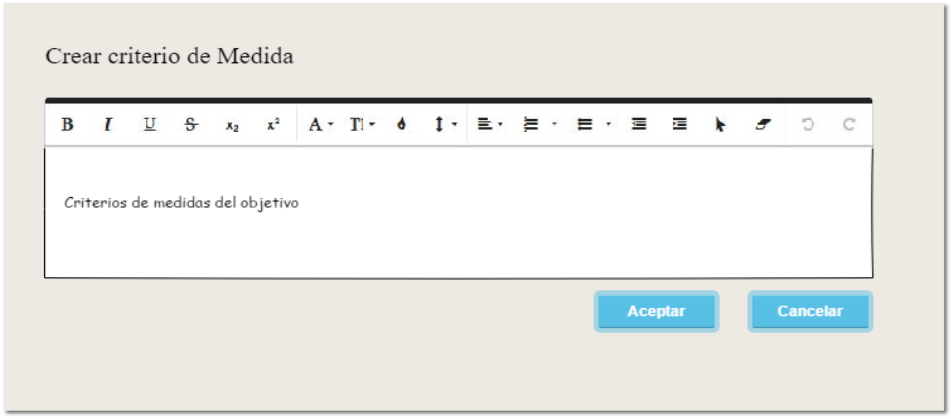


Tabla 27. Historia de Usuario Listar criterios de medidas. [Fuente: Elaboración Propia]

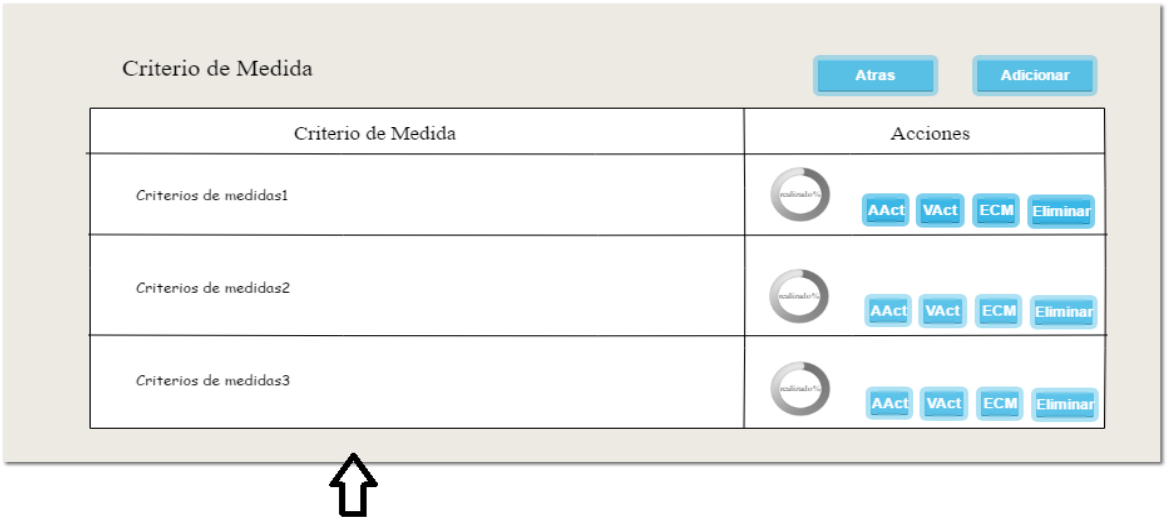
Número: 6	**Nombre del requisito:** Listar criterios de medidas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario puede visualizar un listado de los criterios de medidas	
Observaciones: Es obligatorio dar click al botón de estado para poder visualizar la lista de los criterios de medidas.	
Prototipo de interfaz: 	

Tabla 28. Historia de Usuario Eliminar criterios de medidas. [Fuente: Elaboración Propia]

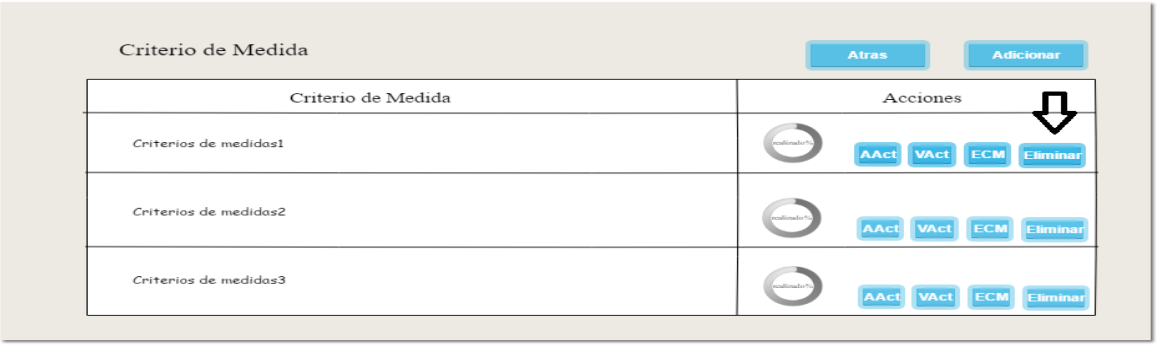
Número: 6	**Nombre del requisito:** Eliminar criterios de medidas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario cuando edita los objetivos también puede editar los criterios de medidas.	
Observaciones: Es obligatorio insertar el campo de criterios de medidas. El sistema emite un error de formulario en caso de que no se llene.	
Prototipo de interfaz: 	

Tabla 29. Historia de Usuario Listar criterios de medidas. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Listar criterios de medidas
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	
Descripción: El usuario cuando crea los criterios de medidas. Se muestra el listado y el avance de los criterios con los botones para realizar las acciones adicionar, editar, eliminar, adicionar actividad.	
Observaciones: N/A	
Prototipo de interfaz:	

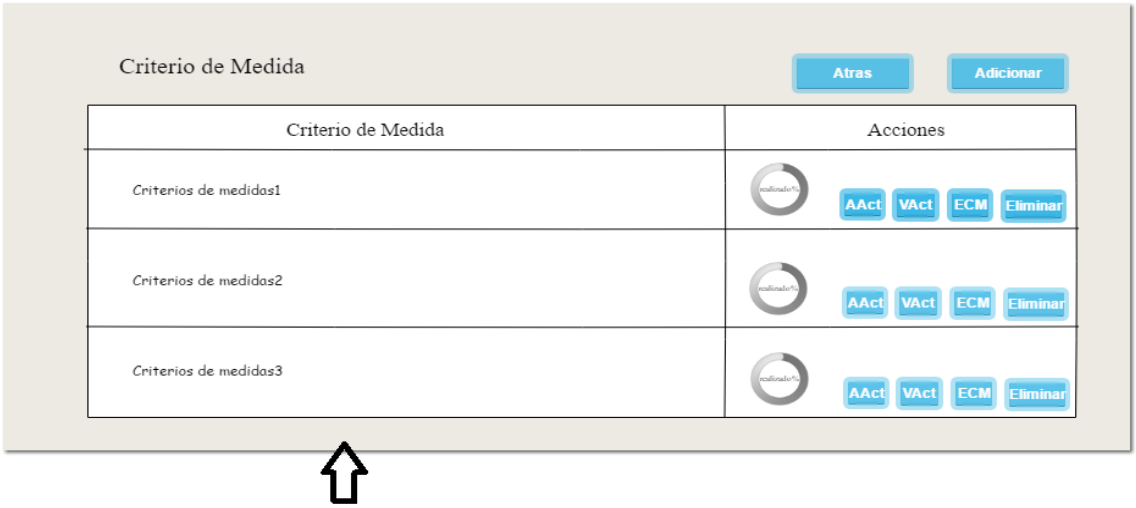


Tabla 30. Historia de Usuario Crear actividades. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Crear actividades
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	

Descripción:

El usuario debe dirigirse al link actividades que se encuentra en la cabeza de la página, al darle click se mostrara un botón para agregar las actividades una vez pulsado el botón se muestra la interfaz para crearlas.

Observaciones:

Es obligatorio insertar los campos de la creación de la actividad y la fecha de la misma. El sistema emite un error de formulario en caso de que no se llenen. Las actividades contables deben marcarse para que el sistema de la opción de cantidad de actividades.

Prototipo de interfaz:

Tabla 31. Historia de Usuario Crear actividades. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Eliminar actividades
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	

Descripción:

El usuario debe dirigirse al botón eliminar que se encuentra en la derecha de la página, al darle click este elimina la actividad seleccionada.

Observaciones:

Las actividades contables si no se ha subido la evidencia no se puede eliminar hasta que exista la misma. El sistema emite un error actividad no evidenciada.

Prototipo de interfaz:

Tabla 32. Historia de Usuario Editar actividades. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Editar actividades
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	

Descripción:

El usuario debe dirigirse al botón editar que se encuentra en la derecha de la página, al darle click este muestra una interfaz para editar las actividades. Esta muestra la cantidad de evidencias y el progreso de esas evidencias.

Observaciones:

N/A

Prototipo de interfaz:

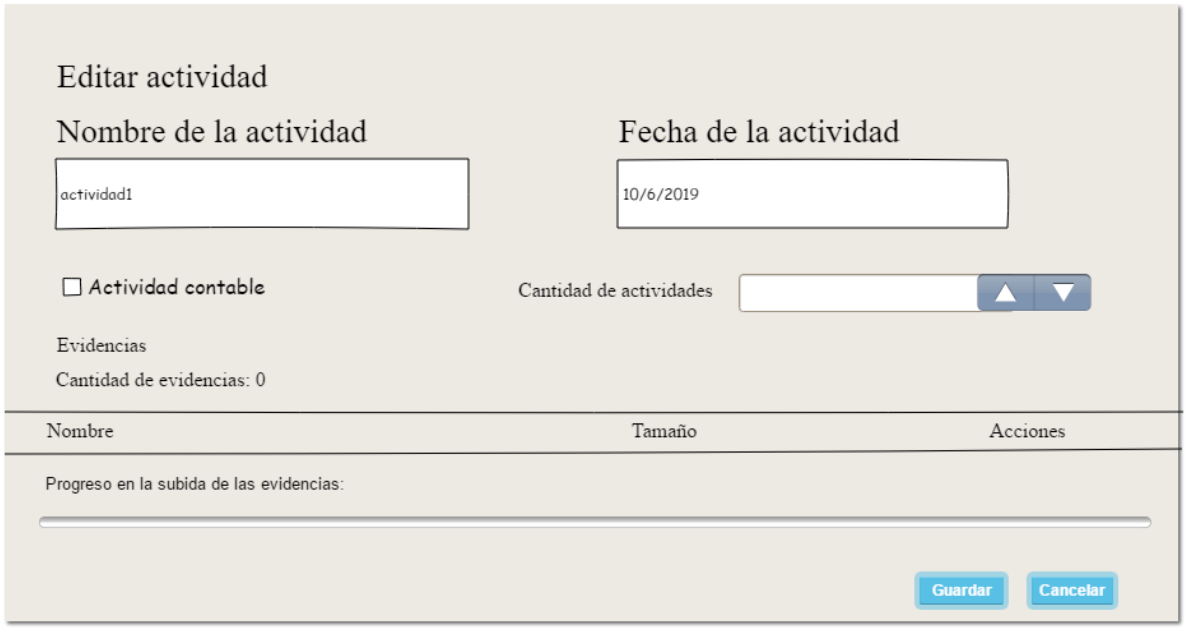


Tabla 33. Historia de Usuario Listar actividades. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Listar actividades
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	

Descripción:

El sistema una vez creadas las actividades muestra un listado de las mismas con las opciones eliminar, crear, editar.

Observaciones:

N/A

Prototipo de interfaz:

Tabla 34. Historia de Usuario Visualizar cumplimiento de objetivos. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Visualizar cumplimiento de objetivos
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 2da
Prioridad: Media	**Tiempo Estimado:** 2
Riesgo en Desarrollo: Medio	
Descripción: El sistema brinda el estado del cumplimiento de los objetivos en por cientos, en la parte derecha de la página.	
Observaciones: EL objetivo debe tener criterios de medidas para que se pueda ver el avance en por cientos del mismo.	
Prototipo de interfaz:	

Tabla 35. Historia de Usuario Filtrar objetivos por estado de cumplimiento. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Filtrar objetivos por estado de cumplimiento
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 3ra
Prioridad: Baja	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Bajo	

Descripción:

El Sistema permite la filtrar los objetivos para mostrar los cumplidos, no cumplidos, cumplidos con señalamientos y sobre cumplido en dependencia cual se escoja.

Observaciones:

Si no se escoge ninguno de las opciones de filtrado el sistema muestra todos los objetivos.

Prototipo de interfaz:

Tabla 36. Historia de Usuario Filtrar objetivos por ARC. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Filtrar objetivos por ARC
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 3ra
Prioridad: Baja	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Bajo	

Descripción:

El Sistema permite la filtrar los objetivos por ARC mostrando un campo desplegable que se encuentra en la izquierda de la página, donde se puede escoger las áreas a la que quiera gestionar los objetivos.

Observaciones:

Si el usuario es jefe de más de un área de resultados claves tendrá que escoger cual área quisiera ver los objetivos, en caso contrario por defecto mostrara los objetivos a su área perteneciente.

Prototipo de interfaz:

Tabla 37. Historia de Usuario Asignar roles. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Asignar roles
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	

Descripción:

El usuario debe dirigirse a la parte derecha de la página y dar click en el link usuario, donde una vez realizada la acción mostrara un botón adicionar usuario, al darle click el sistema muestra los campos que debe llenar como: nombre, e-mail, contraseña, rol, rol adicional. El campo rol muestra las opciones de rol que existen y se escoge el rol que sea el usuario.

Observaciones:

Los campos deben llenarse en caso que no se llenen el sistema emite un error.

El campo rol adicional es en caso que el usuario debe realizar dos roles distintos este campo no es obligatorio.

Prototipo de interfaz:

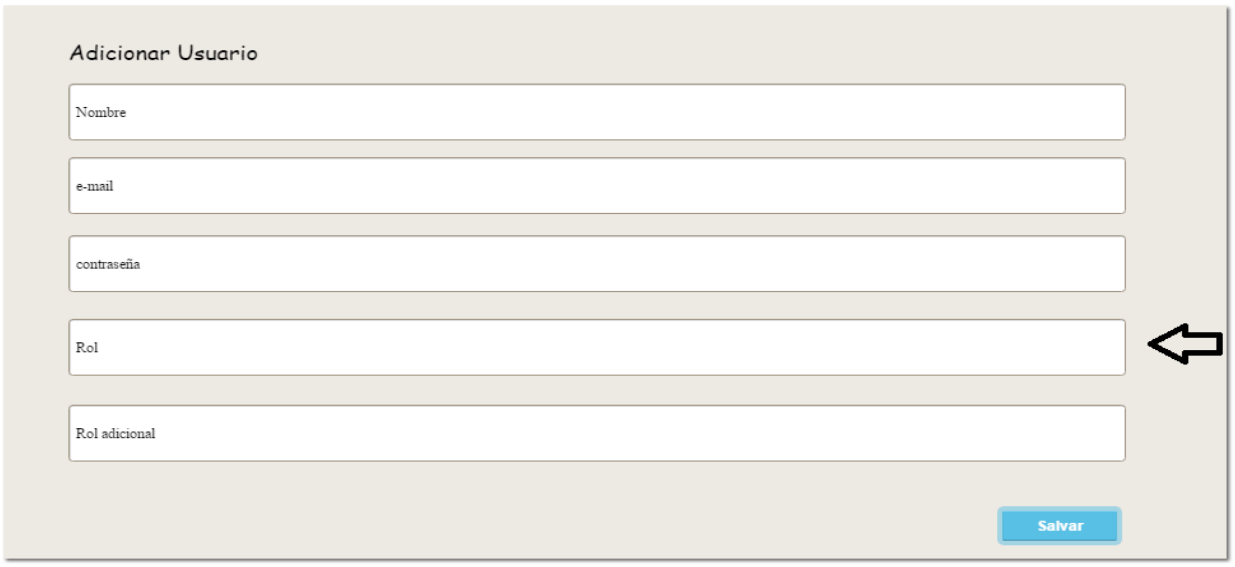


Tabla 38. Historia de Usuario Insertar usuario. [Fuente: Elaboración Propia]

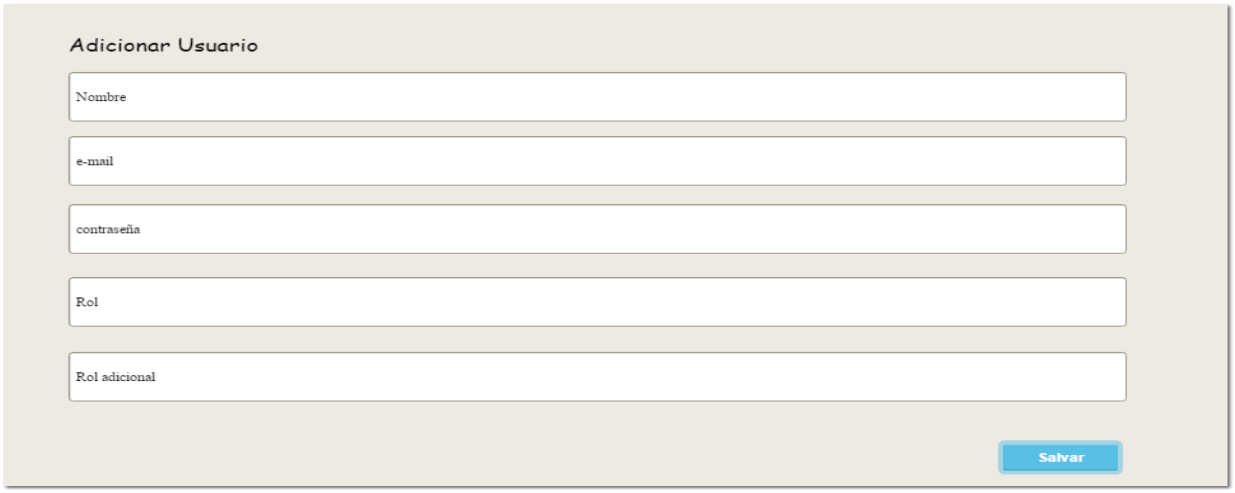
Número: 6	**Nombre del requisito:** Insertar usuario
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	
Descripción: El usuario debe dirigirse a la parte derecha de la página y dar click en el link usuario, donde una vez realizada la acción mostrara un botón adicionar usuario, al darle click el sistema muestra los campos que debe llenar como: nombre, e-mail, contraseña, rol, rol adicional	
Observaciones: Los campos deben llenarse en caso que no se llenen el sistema emite un error. El campo rol adicional es en caso que el usuario debe realizar dos roles distintos este campo no es obligatorio.	
Prototipo de interfaz: 	

Tabla 39. Historia de Usuario Listar usuario. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Listar usuario
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	

Descripción:

El usuario debe dirigirse a la parte derecha de la página y dar click en el link usuario, donde una vez realizada la acción mostrara un listado de los usuarios con sus datos pertenecientes (email, fecha de creación, avatar, rol, rol adicional) además de las acciones que se puede realizar (adicionar, editar, eliminar)

Observaciones:

N/A

Prototipo de interfaz:

Tabla 40. Historia de Usuario Editar usuario. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Editar usuario
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	
Descripción: El usuario debe dirigirse a las acciones que se encuentran en la parte derecha de la página y debe darle click al botón editar el cual mostrara los campos para realizar la acción de edición.	
Observaciones: N/A	
Prototipo de interfaz:	

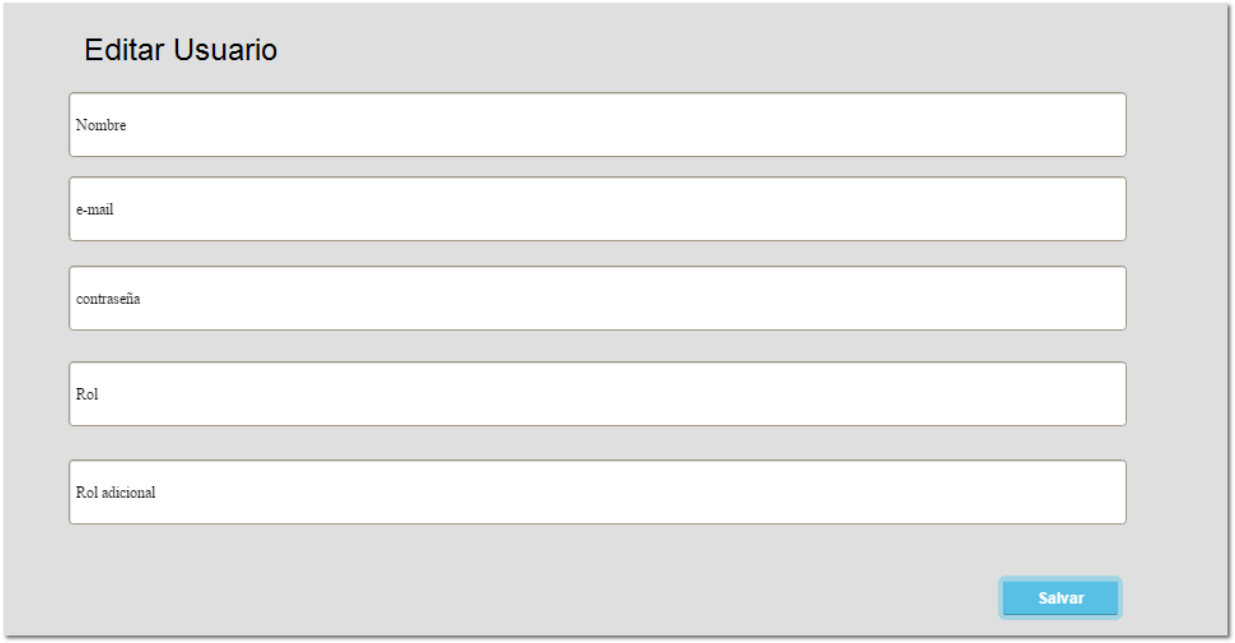


Tabla 41. Historia de Usuario Eliminar usuario. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Eliminar usuario
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Bajo	
Descripción: El usuario debe dirigirse a las acciones que se encuentran en la parte derecha de la página y debe darle click al botón eliminar.	
Observaciones: N/A	
Prototipo de interfaz:	

Tabla 42. Historia de Usuario Generar informe. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito: Generar informe de objetivos de trabajo**
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 2da
Prioridad: Media	**Tiempo Estimado:** 2
Riesgo en Desarrollo: Medio	

Descripción:

El usuario debe dirigirse a la parte superior derecha de la página donde se puede visualizar el botón exportar, al darle click el sistema permite exportar los objetivos, métricas y actividades en formato pdf.

Observaciones:

N/A

Prototipo de interfaz:

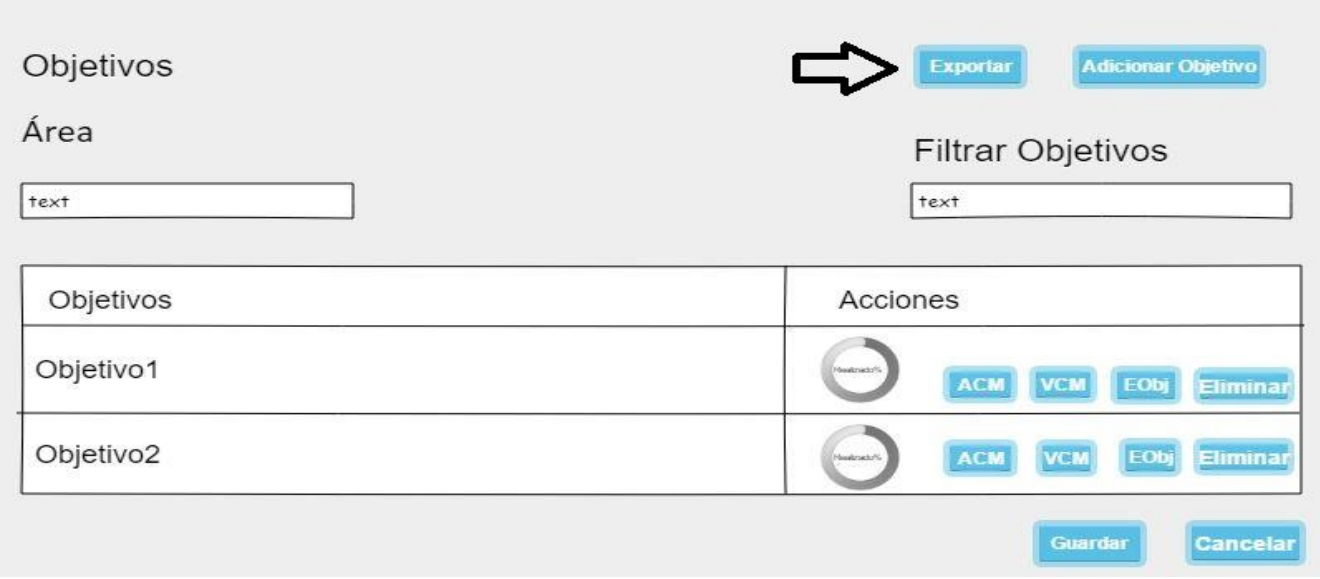


Tabla 43. Historia de Usuario Subir evidencia. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Subir evidencia
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 3ra
Prioridad: Baja	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Bajo	

Descripción:

El usuario debe dirigirse al botón editar actividades donde podrá visualizar el nombre de la actividad y la cantidad de evidencias que debe subir en caso de que la actividad lo lleve.

Observaciones:

Solo se puede subir la cantidad de evidencias que esta lleve.

Prototipo de interfaz:

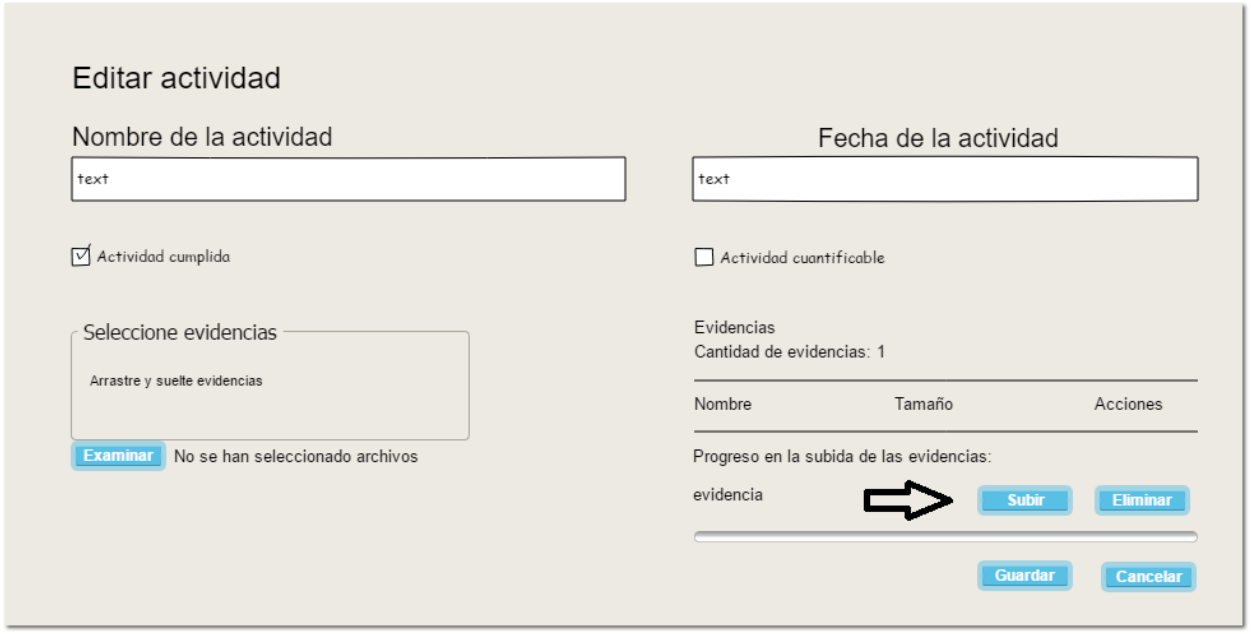


Tabla 44. Historia de Usuario Descargar evidencia. [Fuente: Elaboración Propia]

Número: 6	**Nombre del requisito:** Descargar evidencia
Programador: Adrián Céspedes Calzado	**Iteración Asignada:** 3ra
Prioridad: Baja	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Bajo	

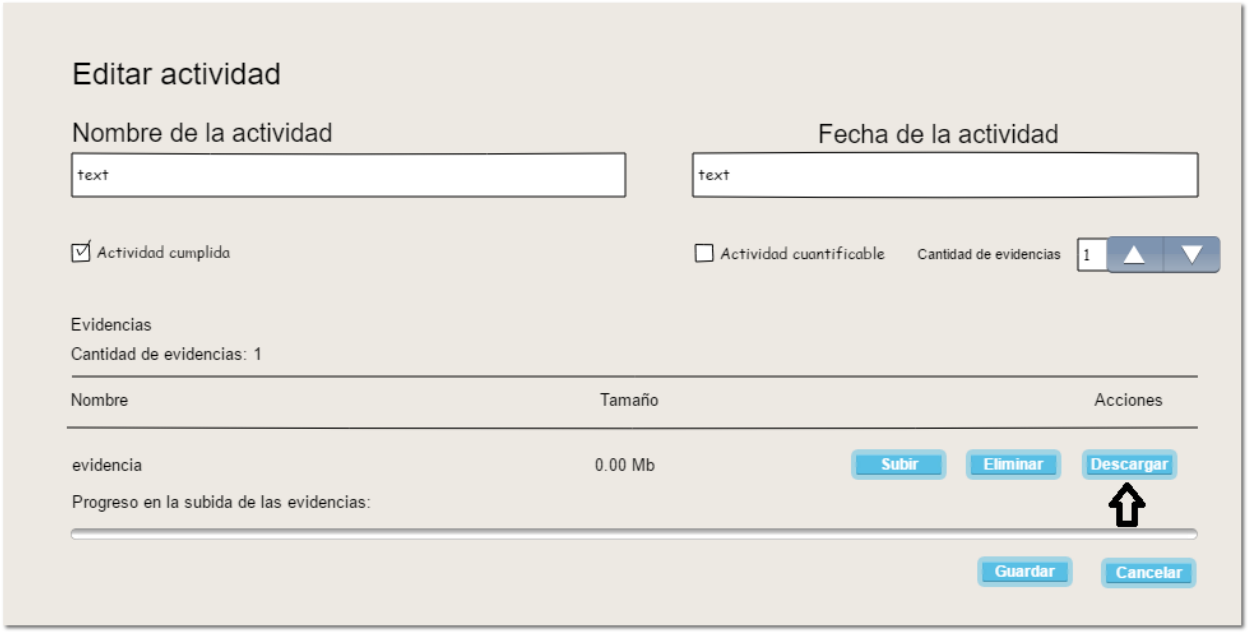
Descripción:

El usuario debe dirigirse al botón editar actividades donde podrá visualizar las evidencias de la actividad correspondiente con las acciones (subir, eliminar, descargar) que se encuentran en la parte derecha inferior de la página. El usuario le da click al botón descargar y se inicia la descarga de la evidencia.

Observaciones:

Las opciones subir y eliminar evidencia solo está disponible para el rol de Activista.

Prototipo de interfaz:



Anexo 2: Casos de prueba

Casos de prueba HU. Gestionar usuario

Tabla 45. Caso de prueba: Insertar Usuario [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**email**	**Contraseña**	**Rol**	**Rol adicional**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Insertar usuario	Se inserta correctamente un nuevo usuario	V	V	V	V	V	Se inserta un nuevo usuario en el sistema	1. Se selecciona la opción “adicionar” en la página de usuarios. 2. Se llenan y seleccionan todos los campos. 3. Se selecciona la opción “salvar”
EC 1.2 Dejar campos vacíos	Se intenta insertar un usuario dejando campos vacíos.	I	N/A	N/A	N/A	N/A	Muestra un mensaje sobre los campos que están vacíos	1. Se selecciona la opción “adicionar” en la página de usuarios. 2. Se llenan los campos. 3. Se selecciona

								la opción “Salvar”. 4. Se muestran mensajes de error.

Tabla 46. Caso de prueba: Listar Usuario [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 2.1 Listar usuario	Se adiciona un usuario	Muestra una lista con los usuarios insertados con sus datos.	Se selecciona la opción salvar en la página insertar usuario

Tabla 47. Caso de prueba: Editar Usuario [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**email**	**Contraseña**	**Rol**	**Rol adicional**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Insertar datos del usuario correctamente.	Se guardan correctamente los cambios realizados.	V	V	V	V	V	Se guardan el usuario con los nuevos datos insertados.	1. Se selecciona la opción “Editar” en la página principal de usuario. 2. Se editan los campos. 3. Se selecciona la opción “Salvar”.

EC 3.2 Dejar campos vacíos	Se intenta editar una evidencia dejando campos vacíos.	I	N/A	N/A	N/A	N/A	Muestra un mensaje de error sobre los campos que se dejaron vacíos.	1. Se selecciona la opción “Editar” en la página principal de usuario. 2. Se editan los campos. 3. Se selecciona la opción “Salvar”. 4. Se muestran mensajes de error.

Tabla 48. Caso de prueba: Eliminar Usuario [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 4.1 Eliminar correctamente un usuario.	Se elimina un usuario del sistema.	Elimina el usuario de la base de datos.	1. Se selecciona la opción “Eliminar” que se encuentra en la página principal de usuarios.
EC 4.2 Eliminar un usuario con área asociada.	El administrador intenta eliminar un usuario con un área asociada.	Redirige hacia una página de error.	1. Se selecciona la opción “Eliminar” que se encuentra en la página principal de usuarios.

			2. Se redirige hacia una página de error.

Caso de prueba HU. "Autenticar usuario"

Tabla 49. Caso de prueba: Autenticar Usuario [Fuente: Elaboración Propia]

Escenario	Descripción	Usuario	Contraseña	Respuesta del Sistema	Flujo Central
Autenticar usuario correctamente	El usuario introduce los datos (usuario y contraseña) correctos	Válido (V)	Válido (V)	El sistema verifica que los datos estén correctos de ser correctos de ser así, mostrará la sesión definida para este rol.	El usuario selecciona el botón "Iniciar" que se encuentra en a la derecha de la pantalla principal, luego selecciona el campo Usuario y Contraseña , introduce los datos y selecciona el botón "iniciar sesión"
		prueba	prueba		
Autenticar usuario incorrectamente	El usuario introduce el dato usuario incorrectamente	Incorrecto (I) pepe	V	El sistema verifica que los datos estén correctos de no ser así, se muestra un mensaje de error	El usuario selecciona el campo Usuario y Contraseña introduce los datos y selecciona el botón "iniciar sesión"

Tabla 50. Descripción de las variables [Fuente: Elaboración Propia]

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	usuario	texto	no	En este campo se le permite al usuario poner su nombre de usuario
2	contraseña	texto	no	En este campo se le permite al usuario poner su contraseña

Caso de prueba HU. "Asignar rol"

Tabla 51. Caso de prueba: Asignar rol [Fuente: Elaboración Propia]

Escenario	Descripción	Nombre	email	Contraseña	Rol	Rol adicional	Respuesta del Sistema	Flujo Central
EC 3.1 Asignar rol	Se insertan correctamente los datos.	V	V	V	V	V	Se guardan el usuario con los datos insertados.	1. Se selecciona la opción "adicionar" en la página principal de adicionar usuario. 2. Se selecciona el campo rol. 3. Se selecciona la opción "Salvar".

EC 3.2 Dejar campos vacíos	Se intenta insertar un usuario dejando campos vacíos.	I	N/A	N/A	N/A	N/A	Muestra un mensaje de error sobre los campos que se dejaron vacíos.	1. Se selecciona la opción “adicionar” en la página principal de usuario. 2. 2. Se selecciona el campo rol. 3. Se selecciona la opción “Salvar”. 4. Se muestran mensajes de error.

Casos de prueba HU. “Gestionar Área”

Tabla 52. Caso de prueba: Insertar ARC [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Insertar ARC.	Se adiciona correctamente el ARC.	V	Se inserta una nueva ARC en el sistema.	1. Se selecciona la opción “Insertar” en la página principal de Áreas. 2. Se selecciona la

				opción “Guardar”.
EC 1.2 dejar campo vacío.	Se intenta adicionar un área dejando el campo vacío.	N/A	Muestra una notificación “El campo nombre del área es obligatorio”.	1. Se selecciona la opción “Insertar” en la página principal de Áreas. 2. Se selecciona la opción “Guardar”.

Tabla 53. Caso de prueba: Eliminar ARC [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 2.1 Eliminar un ARC.	Se elimina un ARC del sistema.	Elimina el ARC de la base de datos.	1. se selecciona la opción “Eliminar” de la página principal de Áreas.

Tabla 54. Caso de prueba: Listar ARC [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Listar área.	Se adiciona un área.	Muestra una lista con las ARC insertados.	1. Se selecciona la opción “adicionar” en la página de Áreas. 2. Se inserta el nombre del ARC. 3. Se selecciona la opción “Guardar”.

Tabla 55. Caso de prueba: Editar ARC [Fuente: Elaboración Propia]

Escenario	Descripción	Nombre	Respuesta del Sistema	Flujo Central
EC 4.1 Editar un ARC	Se guardan correctamente los cambios realizados.	V	Se guardan ARC seleccionada con los nuevos datos.	1. Se selecciona la opción “editar” en la página de Áreas. 2. Editar las ARC. 2. Se selecciona la opción “Guardar”.
EC 4.2 dejar campo vacío.	Se intenta editar un ARC dejando el campo vacío.	N/A	Muestra una notificación “El campo nombre del área es obligatorio”.	1. Se selecciona la opción “Editar” en la página principal de Áreas. 2. Se selecciona la opción “Guardar”.

Caso de prueba HU. “Gestionar objetivos”

Tabla 56. Caso de prueba: Insertar objetivos [Fuente: Elaboración Propia]

Escenario	Descripción	Nombre	Respuesta del Sistema	Flujo Central
EC 1.1 Insertar objetivos.	Se adiciona correctamente el objetivo.	V	Se inserta un nuevo objetivo en el sistema.	1. Se selecciona la opción “Insertar” en la

				página principal de Objetivos. 2. Se llenan todos los campos 3. Se selecciona la opción “Guardar”.
EC 1.2 dejar campo vacío.	Se intenta adicionar un objetivo dejando el campo vacío.	N/A	Muestra una notificación “El campo nombre del objetivo es obligatorio”.	1. Se selecciona la opción “Insertar” en la página principal de Objetivos. 2. Se selecciona la opción “Guardar”.

Tabla 57. Caso de prueba: Listar objetivos [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Listar objetivos.	Se adiciona un objetivo.	Muestra una lista con los objetivos insertados.	1. Se selecciona la opción “adicionar” en la página de Objetivos. 2. Se inserta el nombre del objetivo. 3. Se selecciona la opción “Guardar”.

Tabla 58. Caso de prueba: Editar objetivo [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**Observaciones**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Editar un objetivo	Se guardan correctamente los cambios realizados.	V	V	Se guardan el objetivo seleccionado con los nuevos datos.	1. Se selecciona la opción “editar” en la página de Objetivos. 2. Editar el nombre del objetivo. 3. Se selecciona la opción “Guardar”.
EC 3.2 dejar campo vacío.	Se intenta editar un objetivo dejando el campo vacío.	N/A	N/A	Muestra una notificación “El campo nombre del objetivo es obligatorio”.	1. Se selecciona la opción “editar” en la página de Objetivos. 2. Editar el nombre del objetivo. 3. Se selecciona la opción “Guardar”.

Tabla 59. Caso de prueba: Eliminar objetivo [Fuente: Elaboración Propia]

Escenario	Descripción	Respuesta del Sistema	Flujo Central
EC 4.1 Eliminar un objetivo.	Se elimina un objetivo del sistema.	Elimina el objetivo de la base de datos.	1. Se selecciona la opción “Eliminar” de la página principal de Objetivos.
EC 4.2 Eliminar objetivo con criterio de medida asociado.	Se elimina un objetivo del sistema con criterio de medida asociado.	Muestra el mensaje informativo comunicándole al usuario que el objetivo contiene criterios de medida y no puede ser eliminado.	1. Se selecciona la opción “Eliminar” de la página principal de Objetivos. 2. Se muestra un mensaje de error.

Casos de prueba HU. “Gestionar criterios de medidas”

Tabla 60. Caso de prueba: Insertar criterios de medida [Fuente: Elaboración Propia]

Escenario	Descripción	Nombre	Respuesta del Sistema	Flujo Central
EC 1.1 Insertar criterios de medida.	Se adiciona correctamente el criterio de medida.	V	Se inserta un nuevo criterio de medida en el sistema.	1. Se selecciona la opción “Ver criterios de medida” en la página principal de Objetivos. 2. Se llenan todos los campos 3. Se selecciona la opción “Guardar”.

EC 1.2 dejar campo vacío.	Se intenta adicionar un criterio de medida dejando el campo vacío.	N/A	Muestra una notificación “El campo criterio de medida del objetivo es obligatorio”.	1. Se selecciona la opción “Ver criterios de medida” en la página principal de Objetivos. 2. Se selecciona la opción “Guardar”.

Tabla 61. Caso de prueba: Editar criterios de medida [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**Respuesta del Sistema**	**Flujo Central**
EC 2.1 Editar un criterio de medida	Se guardan correctamente los cambios realizados.	V	Se guardan el objetivo seleccionado con los nuevos datos.	1. Se selecciona la opción “editar” en la página de Criterios de medida. 2. Editar el nombre del criterio de medida. 3. Se selecciona la opción “Guardar”.
EC 2.2 dejar campo vacío.	Se intenta editar un criterio de medida	N/A	Muestra una notificación “El campo criterio de medida es obligatorio”.	1. Se selecciona la opción “editar” en la página de

	dejando el campo vacío.			Criterios de medida. 2. Se selecciona la opción “Guardar”.

Tabla 62. Caso de prueba: Eliminar criterios de medida [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Eliminar un criterio de medida.	Se elimina un criterio de medida del sistema.	Elimina el criterio de medida de la base de datos.	1. Se selecciona la opción “Eliminar” de la página principal de Criterios de medidas.
EC 3.2 Eliminar criterio de medida con actividades asociadas.	Se elimina un criterio de medida del sistema con actividades asociadas.	Muestra el mensaje informativo comunicándole al usuario que el criterio de medida contiene actividades y no puede ser eliminado.	1. Se selecciona la opción “Eliminar” de la página principal de Criterios de medidas. 2. Se muestra un mensaje de error.

Tabla 63. Caso de prueba: Listar criterios de medida [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Listar criterios de medida.	Se adiciona un criterio de medida.	Muestra una lista con los criterios de medida.	1. Se selecciona la opción “adicionar” en la página de Criterios de medidas.

			2. Se inserta el nombre del criterio de medida. 3. Se selecciona la opción “Guardar”.

Casos de pruebas HU. “Gestionar actividades”

Tabla 64. Caso de prueba: Insertar actividades [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**Fecha**	**Actividad cuantificable**	**Cantidad de evidencias**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Insertar actividad no cuantificable.	Se adiciona correctamente la actividad	V	V	N/A	N/A	Se inserta una nueva actividad en el sistema.	1. Se selecciona la opción “Ver actividades” en la página principal de Criterios de medidas. 2. Se selecciona la opción “Insertar actividad” en la página Actividades planificadas. 3. Se llenan todos los campos

							4. Se selecciona la opción “Guardar”.
EC 1.2 Insertar actividad cuantificable.	Se adiciona correctamente la actividad	V	V	V	V	Se inserta una nueva actividad en el sistema.	1. Se selecciona la opción “Ver actividades” en la página principal de Criterios de medidas. 2. Se selecciona la opción “Insertar actividad” en la página Actividades planificadas. 3. Se llenan todos los campos 4. Se selecciona la opción “Guardar”.
EC 1.3 dejar campo vacío.	Se intenta adicionar una actividad dejando el campo vacío.	N/A	N/A	N/A	N/A	Muestra una notificación “Campos vacíos”.	1. Se selecciona la opción “Ver actividades” en la página

							principal de Criterios de medidas. 2. Se selecciona la opción “Insertar actividad” en la página Actividades planificadas. 2. Se selecciona la opción “Guardar”.

Tabla 65. Caso de prueba: Listar actividades [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 2.1 Listar actividades.	Se adiciona una actividad.	Muestra una lista con las actividades adicionadas.	1. Se selecciona la opción “Ver actividades” en la página principal de Criterios de medidas. 2. Se selecciona la opción “Insertar actividad” en la página Actividades planificadas. 3. Se llenan todos los campos 4. Se selecciona la opción “Guardar”.

Tabla 66. Caso de prueba: Editar actividades [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Nombre**	**Fecha**	**Actividad cuantificable**	**Actividad cumplida**	**Respuesta del Sistema**	**Flujo Central**
EC 3.1 Editar una actividad no cuantificable.	Se guardan correctamente los cambios realizados.	V	V	V	V	Se guardan la actividad seleccionada con los nuevos datos.	1. Se selecciona la opción “Editar actividad” en la página Actividades planificadas. 2. Se editan los campos 3. Se selecciona la opción “Guardar”.
EC 3.2 dejar campo vacío.	Se intenta editar una actividad dejando el campo vacío.	N/A	N/A	V	V	Muestra una notificación “Campos vacíos”.	1. Se selecciona la opción “Editar actividad” en la página Actividades planificadas. 2. Se selecciona la opción “Guardar”.

Tabla 67. Caso de prueba: Eliminar actividades [Fuente: Elaboración Propia]

Escenario	Descripción	Respuesta del Sistema	Flujo Central
EC 4.1 Eliminar una actividad.	Se elimina una actividad del sistema.	Elimina la actividad de la base de datos.	1. Se selecciona la opción “Eliminar” de la página principal de Actividades planificadas.
EC 4.2 Eliminar actividad con evidencia.	Se elimina una actividad con evidencia.	Muestra el mensaje informativo comunicándole al usuario que la actividad contiene evidencias.	1. Se selecciona la opción “Eliminar” de la página principal de Actividades planificadas. 2. Se muestra un mensaje informativo.

Caso de prueba HU. “Subir evidencia”

Tabla 68. Caso de prueba: Subir evidencia [Fuente: Elaboración Propia]

Escenario	Descripción	Respuesta del Sistema	Flujo Central
EC 1.1 Subir evidencias	Se sube una evidencia al sistema y la misma es indexada a la actividad.	Muestra la cantidad de evidencias y el progreso de subida de las evidencias.	1. Se escoge la opción “Examinar” en la página principal de Editar evidencias. 2. Se selecciona la opción “Subir”. 3.Se selecciona la opción guardar.

Caso de prueba HU. “Descargar evidencia”

Tabla 69. Caso de prueba: Descargar evidencia [Fuente: Elaboración Propia]

Escenario	Descripción	Respuesta del Sistema	Flujo Central

EC 1.1 Descargar evidencia	Una evidencia subida e indexada en el sistema se puede descargar.	Muestra los datos de la actividad y la cantidad de evidencias de la misma.	1. Se selecciona la opción “Editar actividad” en la página Actividades planificadas. 2. Se selecciona la opción descargar. 3. Se selecciona la opción “Aceptar”.

Casos de pruebas HU. “Filtrar Objetivos”

Tabla 70. Caso de prueba: Filtrar Objetivos [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Sobre cumplido**	**Cumplido**	**Cumplido con señalamiento**	**Incumplido**	**Todos los objetivos**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Filtrar todos los objetivos.	Se selecciona las opciones para filtrar los objetivos y muestra los objetivos.	N/A	N/A	N/A	N/A	V	El sistema muestra todos los objetivos.	1. Se selecciona la opción “Filtrar objetivos” de la página principal de Objetivos.
EC 1.2 Filtrar objetivos sobre cumplidos	Se selecciona las opciones para filtrar los objetivos y muestra	V	N/A	N/A	N/A	N/A	El sistema muestra todos los objetivos sobre cumplidos.	1. Se selecciona la opción “Filtrar objetivos” de la

	los objetivos.							página principal de Objetivos.
EC 1.3 Filtrar objetivos cumplidos	Se selecciona las opciones para filtrar los objetivos y muestra los objetivos.	N/A	V	N/A	N/A	N/A	El sistema muestra todos los objetivos cumplidos.	1. Se selecciona la opción “Filtrar objetivos” de la página principal de Objetivos.
EC 1.4 Filtrar objetivos cumplidos con señalamientos	Se selecciona las opciones para filtrar los objetivos y muestra los objetivos.	N/A	N/A	V	N/A	N/A	El sistema muestra todos los objetivos cumplidos con señalamiento	1. Se selecciona la opción “Filtrar objetivos” de la página principal de Objetivos.
EC 1.5 Filtrar objetivos incumplidos.	Se selecciona las opciones para filtrar los objetivos y muestra los objetivos.	N/A	N/A	N/A	V	N/A	El sistema muestra todos los objetivos incumplidos.	1. Se selecciona la opción “Filtrar objetivos” de la página principal de Objetivos.

Tabla 71. Caso de prueba: Filtrar objetivos por ARC [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Escoger ARC disponible**	**Respuesta del Sistema**	**Flujo Central**
EC 2.1 Escoger ARC.	Se selecciona el ARC y muestra el estado de los objetivos.	V	El sistema muestra un filtro con las ARC que pertenece el usuario visualizando el estado de los objetivos del ARC que selecciono.	1. Se selecciona la opción “Área” de la página principal de Objetivos.
EC 2.2 No escoger ARC.	No se escoge el ARC y se muestra el estado de los objetivos	N/A	El sistema muestra el ARC que pertenece el usuario visualizando el estado de los objetivos.	1. Se visualiza el nombre del ARC en el campo “Área” de la página principal de Objetivos.

Caso de prueba HU. “Generar informe de objetivos de trabajo”

Tabla 72. Caso de prueba: Generar informe de objetivos de trabajo [Fuente: Elaboración Propia]

Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Generar informe.	Se exporta los objetivos y métricas de las ARC.	El sistema muestra una ventana para exportar los objetivos y métricas en formato pdf.	1. Se selecciona la opción “Exportar” de la página principal de Objetivos.

Anexo 3: Diagrama de componentes

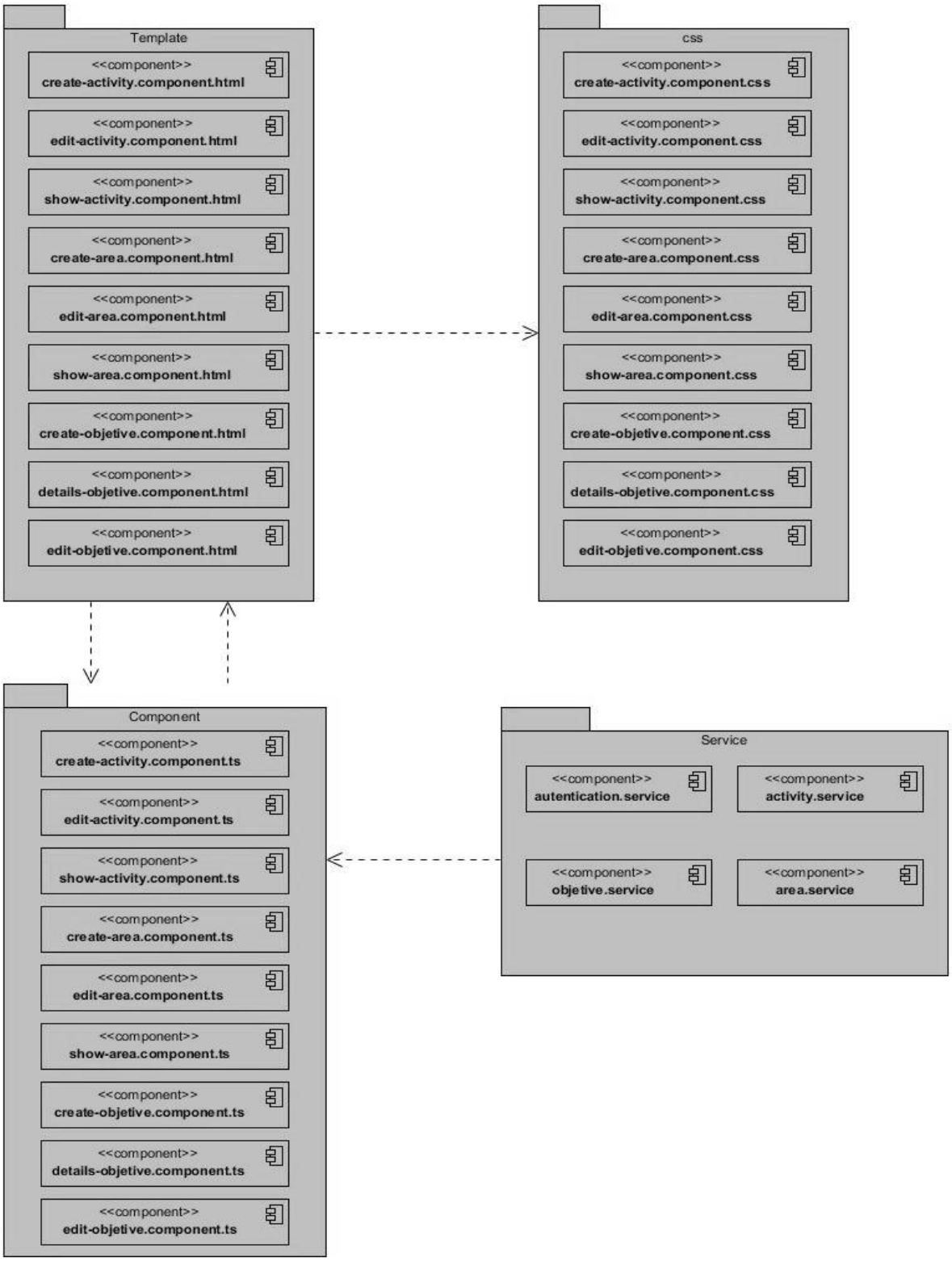


Figura 8. Diagrama de componentes [Fuente: Elaboración propia]