Facultad 2

Sistema para la gestión de la evaluación y ubicación de la práctica profesional de los estudiantes de la Facultad 2

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

Autor:

Daidelín Ríos Guillén

Tutor:

MSc. Darling Darias Pérez

La Habana, noviembre de 2023

"Año 64 de la Revolución"

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título "**Sistema para la gestión de la evaluación y ubicación en la PID de los estudiantes de la Facultad 2**" concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firman la presente a los 20 días del mes de octubre del año 2023.

_____	_____
Firma del Autor	Firma del Tutor
Daidelín Ríos Guillén	MSc. Darling Darias Pérez

AGRADECIMIENTOS

RESUMEN

El presente estudio de investigación se centra en la creación de un sistema web para el proceso de gestión de la ubicación y evaluación de la PID en la facultad 2. Este sistema en línea fue dirigido por la metodología AUP versión UCI En cuanto a las tecnologías del lado del usuario, se emplearon: HTML5, JavaScript, CSS3, respaldadas por el marco de diseño Bootstrap. El lenguaje del lado del servidor utilizado fue JavaScript mediante su framework NodeJS, se empleó el Visual Studio Code como entorno de desarrollo, PgAdmin 4 y PostgreSQL para la administración de bases de datos. Se incorporaron funcionalidades en la gestión de áreas, problemas, estudiantes, evaluaciones y proyectos. Asimismo, se introdujo la funcionalidad de exportar el registro de las evaluaciones existentes en el sistema. Se llevaron a cabo pruebas de software a la propuesta de solución, las cuales proporcionan niveles de confirmación de las funcionalidades y la satisfacción por parte del usuario.

PALABRAS CLAVE: evaluación, registro, ubicación.

ABSTRACT

The present research study focuses on the creation of a web system for the management process of the placement and evaluation of the PID in faculty 2. This online system was directed by the AUP methodology UCI version Regarding the technologies of the On the user side, HTML5, JavaScript, CSS3 were used, supported by the Bootstrap design framework. The server-side language used was JavaScript through its NodeJS framework, Visual Studio Code was used as the development environment, PgAdmin 4 and PostgreSQL for database administration. Functionalities were incorporated in the management of areas, problems, students, evaluations and projects. Likewise, the functionality of exporting the record of existing evaluations in the system was introduced. Software tests were carried out on the solution proposal, which provide levels of confirmation of the functionalities and user satisfaction.

KEYWORDS: evaluation, location, registration.

ÍNDICE

Índice de tablas

Índice de figuras

INTRODUCCIÓN

En la era actual caracterizada por el avanzado desarrollo en el campo de la informática, es innegable que la digitalización ha tomado una posición importante en la mayoría de las organizaciones empresariales. Los sistemas informáticos se han convertido en un recurso ineludible que abarca todas las esferas de la vida, proporcionando una infraestructura esencial para la gestión de datos y eliminando la laboriosa tarea de realizar tareas manuales. Esta automatización no solo previene que los trabajadores cometan errores, sino que también permite el acceso instantáneo a información actualizada en todos los procesos digitalizados, ofreciendo un control y seguimiento riguroso de los mismos (Carlos Valdivia, 2020).

La Universidad de las Ciencias Informáticas (UCI) es una universidad insertada en la Industria Cubana del Software, que basa su formación en un modelo de integración formación-producción-investigación, a partir de la vinculación estudio-trabajo. Es por ello considerada una “universidad productiva”, concepto definido por el comandante en Jefe Fidel Castro en el 2004 para señalar las características de la UCI y diferenciarla de las clásicas relación universidad-empresa en este campo (Fidel Castro, 2004).

El plan del proceso docente de la distribución por semestre y años de estudio y las indicaciones metodológicas y de organización de la carrera va orientado a la práctica laboral, trabajo independiente del estudiante, estrategias curriculares y evaluación para la culminación de estudios. Todo ello en conjunto implica la necesidad de integración de sus procesos fundamentales: la formación, la producción y la investigación. Fidel puntualizó el carácter de ciencia de la carrera, el cual se concreta a partir de la afirmación de que lo investigativo es esencia de lo académico y de lo laboral al definir como disciplina principal integradora la práctica profesional (Moreira Pérez, Dayyanis, Yanes Watson, Berta Irailis, 2021).

La actividad de desarrollo-producción de la UCI se soporta sobre la red de centros, integrada por centros de desarrollo, un centro de soporte, dirección de calidad de software, y diferentes áreas y líneas de investigación que brindan servicios transversales a la producción. En la actualidad el proceso para la inserción de estudiantes en la práctica

profesional se realiza a través de una encuesta. En dicha encuesta expresan el interés de pertenecer a cualquiera de estos centros de desarrollo, áreas, líneas de investigación y dirección de investigación. A partir del índice académico y las necesidades de cada área, se realiza la distribución de todas las facultades. Esto evidencia que existen muchos estudiantes de diferentes facultades distribuidos en varias áreas.

Actualmente este proceso ocurre de manera atropellada, sin analizar o tener en cuenta los procedimientos previstos, las informaciones son muy tardías, tienden a ser incongruentes. A los jefes de departamento de informática de cada facultad, les es engorroso saber el área en que se ubica, las necesidades, así como los roles que poseen los estudiantes en los centros y áreas productivas. Además, de que se omite por la premura y atropello de la ubicación, las habilidades y aptitudes que posee el estudiante. Posteriormente le es difícil conocer las tareas que realiza este estudiante, así como las evaluaciones que se le emite según su desempeño y resultados. Pues debe asignar a una persona o él buscar por las diferentes áreas la evaluación cuantitativa y cualitativa del estudiante.

En muchas ocasiones, las áreas no otorgan un posible tema investigativo y de desarrollo en víspera de trabajo de culminación de estudios, como tampoco se conoce con un banco de problemas visibles en cada área para poder apoyar como docentes a la producción.

Teniendo en cuenta la situación problemática anterior, surge como **problema a resolver**:

¿Cómo facilitar el proceso de evaluación y ubicación en la PID de los estudiantes de la facultad 2? Siendo el **objeto de estudio**: el proceso de gestión de la información en a PID, acotándose en el **campo de acción:** el proceso de gestión de la ubicación y evaluación de la PID en la facultad 2.

Se propone como **objetivo general** desarrollar un sistema informático que facilite el proceso de evaluación y ubicación de la PID de los estudiantes de la Facultad 2.

A partir del objetivo general y en víspera de darle solución se derivan los siguientes **objetivos específicos:**

1. Establecer los fundametos y referentes teóricos-metodológicos que sustentan la investigación para el desarrollo del sistema.

2. Diseñar una propuesta de solución que permita la ubicación y evaluación en la PID de los estudiantes de la facultad 2.

3. Implementar una propuesta de solución que permita la ubicación y evaluación en la PID de los estudiantes de la facultad 2.

4. Validar la propuesta a partir de la aplicación implementada para la gestión la ubicación y evaluación de la PID.

Para lograr los objetivos trazados se llevan a cabo las siguentes **tareas de investigación**:

Tareas de la investigación

1. Fundamentación de la investigación mediante búsqueda y análisis bibliográfico, teniendo en cuenta el diagnóstico del estado actual de los procesos de gestión de ubicación y evaluación de la PID en la facultad 2.

2. Análisis valorativo de los sistemas informáticos referentes al proceso de gestión de prácticas laborales, existentes a nivel nacional e internacional, estableciendo similitudes con la investigación.

3. Implementación de una aplicación informática aplicando las pautas de diseño definidas, siguiendo una metodología y lo establecido en la especificación de requisitos de software.

4. Validación de la propuesta para verificar el cumplimiento y satisfacción del cliente.

Métodos de la investigación

Los métodos de la investigación que se analizaron son los métodos teóricos, empíricos y matemáticos-estadísticos.

Métodos teóricos

- **Análisis histórico-lógico:** para el estudio crítico de los trabajos anteriores sobre el proceso de ubicación y evaluación de la PID, para utilizar estos como punto de referencia y comparación con los resultados alcanzados.

- **Analítico-sintético:** para descomponer el problema de investigación en elementos separados y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

- **Inductivo-deductivo:** para la identificación de la problemática y de las soluciones en el proceso de evaluación y asignación de la PID.

- **Modelación:** para modelar teóricamente la aplicación informática propuesta para el proceso de ubicación y evaluación de los estudiantes en la PID.

Métodos empíricos

- **Observación:** para obtener la información necesaria para constatar la problemática y el problema de investigación sobre el proceso de asignación y evaluación de la PID.

- **Entrevista:** para obtener la información necesaria de los estudiantes y profesores antes de diseñar la solución informática propuesta.

Posibles resultados y beneficios:

- ✓ Un sistema informático para la gestión de la ubicación y evaluación de la PID de los estudiantes de la facultad 2.

- ✓ Un trabajo investigativo, que sustenta y fundamenta la propuesta de solución.

- ✓ Agilidad del proceso de evaluación de los estudiantes en la asignatura de PID en la facultad 2.

Para lograr una sustentación más sólida de la investigación el documento cuenta con la siguiente estructura:

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

Se hace una revisión de la bibliografía donde se analizan los principales conceptos. Se analiza el estado del arte de la investigación. Se definen las tecnologías y herramientas a utilizar en la solución propuesta. Se propone la metodología de desarrollo de software para la construcción de la propuesta de solución de la investigación.

CAPÍTULO 2. DISEÑO DE LA PROPUESTA DE SOLUCIÓN

En este capítulo se presenta la documentación del ciclo de vida de la propuesta de solución hasta la etapa de implementación utilizando la modelación, la descripción y la aplicación de buenas prácticas de desarrollo de software.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS DE LA PROPUESTA.

Se presenta la validación del sistema desarrollado mediante varios criterios y tipos de pruebas para determinar que la solución propuesta responde satisfactoriamente a los objetivos que se trazaron.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se lleva a cabo el estudio del estado del arte realizándose un estudio comparativo de varios sistemas referentes al proceso de gestión de trabajos de culminación de estudios que sirven de base a la investigación del problema planteado. En este capítulo se analizarán los principales conceptos relacionados con el objeto de estudio, se realiza una descripción de los sistemas existentes de gestión académica; y una explicación de herramientas y tecnologías potenciales a utilizar en el desarrollo de la propuesta de solución.

1.1 Referentes asociados al dominio del problema

1.1.1 Práctica Laboral

La práctica laboral en la educación técnica y profesional, es la vinculación de los estudiantes a la actividad laboral de una empresa o entidad de la producción, o los servicios durante un tiempo determinado donde el alumno consolida las habilidades propias del año que cursa. La práctica laboral se programa de manera curricular en los diseños actuales de planes de estudio (Alarcón & Gómez, 2006).

1.1.2 Evaluación educativa

Proceso sistemático, sistémico, participativo y reflexivo que permite emitir una valoración sobre el desarrollo de las potencialidades del y la estudiante, para una toma de decisiones que garantice el logro de los objetivos establecidos (Pasek & Mejía, 2017).

Objetivos

- Desarrollar y consolidar habilidades y capacidades profesionales, propias del año que cursa o años precedentes. Aprender haciendo.
- Integrar los conocimientos adquiridos al proceso productivo o de los servicios, identificándose con las características del sector en el que se forman.
- Consolidar la formación vocacional y profesional

- Formar conciencia de productores, con cultura económica.
- Desarrollar el componente instructivo y laboral.
- Consolidar valores relacionados con su actividad (SNE, 2009).

Caracterización de la práctica profesional

La actividad investigativa-laboral se fundamenta en el desarrollo de la disciplina de Práctica Profesional que constituye la Disciplina Principal Integradora. La misma está compuesta por 10 asignaturas en segundo, tercero, cuarto y quinto año, con 2048 horas en total, a ejecutarse de forma distribuida durante los semestres involucrados y con la garantía de la incorporación de los estudiantes a proyectos productivos reales. Además, en tercer año se desarrolla en el primer semestre la asignatura Metodología de la Investigación Científica, en el segundo semestre de cuarto año, Componente Profesional de Ingeniería y Gestión de Software (CPIGSW) y durante el segundo semestre de quinto año el Trabajo de Diploma. En cada una de las asignaturas de Proyecto de Investigación y Desarrollo (PID), se definen los objetivos educativos e instructivos que se deben alcanzar, así como el contenido y las habilidades correspondientes que deben ser desarrolladas. Los responsables en cada Centro de Desarrollo que existen en cada facultad, se subordinan metodológicamente al Jefe de Departamento de Ingeniería de Software (IGSW) en la Facultad. La Práctica Profesional de igual forma se integra y responde a los cortes C-1, a realizarse en las semanas 8 y 16 de cada semestre. Cada estudiante es atendido por un Supervisor Evaluador Tutor (SET), los cuales pueden ser: profesores o especialistas. El profesor de PID de conjunto con los tutores elaboran un plan de formación a los estudiantes que atienden, donde se incluyen actividades relacionadas con la docencia, investigación y producción. En el plan se incluyen las fechas de cumplimiento de las actividades asignadas.

El régimen de trabajo de los estudiantes se ajusta al régimen de cada centro de desarrollo o del grupo de investigación al cual se vinculen. Cada estudiante debe tener al menos una tarea asignada. Las tareas, en los casos que así procedan,

deben tener determinados aseguramientos instructivos como: teoría, bibliografía, ejemplos, entre otros. Deben tener además un resultado medible, los cuales pueden ser: artefactos, documentos, informes, aplicación, código fuente, etc. Los resultados de estas tareas deben ser presentados en talleres, seminarios o reuniones de proyectos. La evaluación final de las asignaturas se realiza a través de la discusión de un informe final, donde se presenten los resultados de las tareas realizadas durante la asignatura en todo el semestre. Los tribunales que evalúan la defensa de los informes finales se organizan por el Jefe de Departamento de la Facultad de conjunto con los centros. La nota final del estudiante la otorga el profesor de PID a partir de las evaluaciones, parciales y finales y el criterio de los tutores (Ministerio de Educación Superior, 2014).

1.2 Estudio del estado del arte

Para la selección de estos sistemas se valoró principalmente su funcionamiento y gestión total o parcial de los procesos de ejercicios de culminación de estudios similar al módulo que se desea implementar.

1.2.1 Sistemas internacionales

Aplicación web para la gestión de entornos virtuales. (Universidad Complutense de Madrid. Facultad de Informática)

Esta es una aplicación para la gestión de entornos virtuales, sobre las que se realizarán las prácticas de las asignaturas de una determinada titulación dada. Se distinguen tres tipos o niveles diferentes de usuarios: alumnos, profesores y administradores. Los cuales tendrán acceso a diferentes funcionalidades y recursos en función de su nivel de autenticación en el sistema. La aplicación consta principalmente de una base de datos desarrollada en lenguaje Ruby con SQLite, donde se almacena toda la información referente a la aplicación y cuatro interfaces o páginas web. El sistema usa el patrón Modelo-Vista- Controlador. Actualmente la aplicación es sencilla y funcional, pero quizás no demasiado interactiva ni atractiva de cara a un potencial usuario (Alamaraz Hernández et al., 2011).

Ventajas:

- Control de roles y permisos para la autenticación. Permite gestión de la información.

Limitaciones:

- No responde a las características de la práctica laboral realizada en la Universidad.

Aplicación web para la gestión de calificaciones de alumnos. (Universidad Carlos III de Madrid. Departamento de Ingeniería Telemática)

La aplicación Web permite a los docentes de un colegio, instituto o universidad gestionar las calificaciones de sus alumnos para cada curso académico. Teniendo la información de las asignaturas, los grupos y los alumnos organizada. Varios profesores pueden conectarse simultáneamente sin compartir la información. Se permite que un profesor comparta esta información asignando el control de forma parcial a otros profesores. La aplicación permite calificar a los alumnos de una asignatura manualmente o mediante el cálculo automático de las mismas, para el caso de calificaciones que dependen del cálculo de una fórmula. Los listados de las calificaciones, así como los datos de las asignaturas o alumnos se pueden importar y/o exportar a un formato reconocible por el usuario (Núñez Mayorga, 2009).

Ventajas:

- Permite la gestión de las evaluaciones de los estudiantes en las diferentes asignaturas.
- Control de roles y permisos de autenticación.

Limitaciones:

- No responde a las características de la práctica laboral realizada en la Universidad.

1.2.2 Sistemas nacionales

Web docente para la práctica profesional de la Ingeniería Informática, en la semipresencialidad. (Universidad Tecnológica de La Habana José Antonio Echeverría (CUJAE))

El sitio Web docente tiene un diseño estructural no lineal y en los diferentes menús se brinda información de la Disciplina Práctica Profesional y de la asignatura Práctica Profesional I como: programa analítico, secuencia de actividades, indicaciones, artículos del reglamento docente relacionados con la Práctica Profesional, bibliografías, entre otras. Su implementación se realizó utilizando el Sistema de Gestión de Contenidos (CMS) Drupal versión 6.27 y WAMP versión 2.2 (De la Torre Matamoro, 2013).

Ventajas:

- Informa sobre todo lo referente a la práctica profesional de la carrera, actividades, artículos, etc.

Limitaciones:

- No es multiplataforma.
- Es solamente informativo.
- No responde a las características de la práctica laboral realizada en la Universidad.

Sistema para la gestión de la Práctica Profesional en la Facultad 3 (UCI)

El sistema que se propuso el facultad 3 permite realizar el control y seguimiento de los estudiantes en los Proyectos de Investigación y Desarrollo, facilitando un conjunto de reportes que agilizan la entrega de la información que se solicite. Se utilizó la metodología de software Variación de AUP para la Universidad y en la implementación se emplearon tecnologías y herramientas de software libre. El mismo aporta de manera significativa a la informatización y mejora de los procesos

de control y seguimiento de los estudiantes pertenecientes a los centros de desarrollo, mientras cursan las asignaturas PID de la disciplina de Práctica Profesional en el ciclo profesional (Bagrotti Abreu, 2017).

Ventajas:

- Tecnología reciente y adecuada a los requisitos del centro de desarrollo.
- Realiza reportes sobre la información generada en el desarrollo de la práctica.
- Gestiona la información de los estudiantes en la práctica profesional como son las evaluaciones y tareas a realizar.

Limitaciones:

- No gestiona los proyectos desarrollados por los estudiantes en la asignatura PID.
- No responde a las características de la práctica laboral realizada en la Universidad en la actualidad, ya que fue realizada en un momento en el que el desarrollo de la práctica profesional se realizaba en los propios centros de la facultad y además existían departamentos de la práctica en las facultades.

Sistema de Gestión académica (Akademos) (UCI)

En la UCI para la gestión de las evaluaciones se utiliza el Sistema de Gestión Académica (Akademos), herramienta que mantiene informado a los estudiantes y profesores sobre el desempeño académico: matrícula de estudiantes, desarrollo del proceso docente, notas, asistencia, planes de estudio, asignaturas a cursar, disciplina y perfiles. Este sistema soluciona las dificultades para la gestión de información de las diferentes asignaturas que conforman el Plan de Estudio de la Universidad, no siendo de la misma manera para la asignatura PID debido a su concepción y cómo se maneja en cada facultad pues lo único que registra es la nota final de la asignatura (UCI, 2013).

Ventajas:

- Está organizado de manera que cada profesor tiene acceso al grupo que debe atender cualquiera sea su facultad, sus evaluaciones, asistencias, etc. Gestiona las evaluaciones y la asistencia de los estudiantes en cada una de las asignaturas cursadas.

Limitaciones:

- No lleva planificación de actividades productivas.
- No gestiona las tareas desarrollada por los estudiantes en la práctica profesional.
- No registra el plan de formación de los estudiantes en la práctica profesional.

A continuación, se presenta una tabla comparativa de las funcionalidades de los sistemas mencionados:

Tabla 1*Comparación entre sistemas homólogos*

Funcionalidad	**Aplicación web para entornos virtuales (UCM)**	**Aplicación web para la gestión de calificaciones (UC3M)**	**Sistema para la gestión de la Práctica Profesional (UCI)**	**Web docente para la práctica profesional (CUJAE)**	**Sistema de Gestión Académica (Akademos) (UCI)**
Control de roles y permisos para la autenticación	Sí	Sí	No	No	No
Gestión de evaluaciones de estudiantes	No	Sí	Sí	No	No
Gestión de proyectos desarrollados por	No	No	No	No	No

estudiantes en la asignatura PID					
Gestión de tareas desarrolladas por estudiantes en la práctica profesional	No	No	Sí	No	No
Información sobre la práctica profesional de la carrera, actividades, artículos, etc.	No	No	No	Sí	No

Luego de la investigación realizada sobre las soluciones existentes tanto nacionales como internacionales se llegó a la conclusión de realizar un sistema que permita el mejoramiento del flujo de información de la práctica profesional. Los sistemas encontrados no brindan todos los requisitos que se deben tener en cuenta para resolver los problemas existentes en la gestión de la práctica profesional. Las ventajas detectadas constituyen una fuente inicial para la definición de requisitos y las funcionalidades que no pueden faltar en el sistema para la gestión de la información de la práctica profesional. Además, a partir de la profunda indagación de las características de estos sistemas se observó que en las universidades analizadas no se realiza la práctica profesional de igual manera. La mayoría de dichos sistemas no se encuentran disponibles o no se llegaron a implementar y quedaron solo en la fase teórica de un proyecto o tesis, lo que indica la necesidad de un nuevo sistema que responda a las características de la práctica realizada en la Universidad de las Ciencias Informáticas.

1.3 Metodología de Desarrollo

Una metodología de desarrollo de software es una estructura de trabajo usada para planificar y controlar el Proceso de Desarrollo de Software (PDS) en sistemas de

información. Cada metodología de desarrollo de software tiene su propio enfoque para su desarrollo (Pressman, 2014).

Las metodologías se clasifican en robustas o tradicionales y ágiles.

Metodología tradicional: Están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo (Pressman, 2014).

Debido a que se requiere reducir drásticamente el tiempo de desarrollo de la herramienta propuesta, que el equipo de trabajo es pequeño y existe un constante intercambio con el cliente, siendo este parte del equipo, se decide utilizar una metodología ágil. Dentro de las metodologías ágiles se destacan: AUP (Proceso Unificado Ágil) y XP (Programación Extrema).

Metodologías ágiles: son usadas en proyectos cuyo objetivo fundamental es tener el software funcionando lo antes posible, de esta manera el cliente tendrá las primeras versiones donde podrá comprobar y aportar su idea de negocio. Con este objetivo, se trabaja sobre las versiones previas, siendo el desarrollo de la siguiente iteración una mejora de la anterior. Las metodologías ágiles proponen una forma de trabajo flexible cuya planificación se actualiza continuamente. Esto contrasta con las metodologías tradicionales, las cuales siguen una planificación precisa desde el principio.

Metodología AUP–UCI

El Proceso Unificado Ágil (AUP) consiste en una versión simplificada de la metodología de desarrollo tradicional Proceso Racional Unificado (RUP, por sus siglas en inglés, Rational Unified Process). AUP describe la forma de desarrollar aplicaciones de software de manera fácil de entender, usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP

La metodología AUP propone organizar el proceso de desarrollo de software en cuatro fases (Inicio, Elaboración, Construcción, Transición). En el caso de la adaptación de esta metodología para los proyectos de la UCI se decide mantener la fase de inicio, pero modificando su objetivo, las tres restantes fases se unifican, quedando una sola denominada ejecución y se agrega una fase de cierre.

Descripción de las fases:

- **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.

- **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.

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP propone siete disciplinas, las cuales son: Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno. Se decide para el ciclo de vida de los proyectos de la UCI mantener las mismas disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajo: Modelado de negocio, Requisitos, Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagregan en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con la áreas de procesos que define CMMI-DEV v1.3 para que pueda emplearse en

los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Rodríguez Sánchez, 2015).

Descripción de los escenarios:

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o 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:

- **Escenario 1:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema.

- **Escenario 2:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.

- **Escenario 3:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.

- **Escenario 4:** 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 (Rodríguez Sánchez, 2015).

Metodología XP

Es una metodología ágil cuyo pilar principal es el trabajo en equipo y la capacitación de sus miembros con el objetivo de obtener el éxito en el desarrollo de software. Se basa en una interacción constante entre el cliente y el equipo de desarrollo, haciendo flexible el proceso de asimilar nuevos cambios y generar implementaciones simples.

Una de sus características principales es la historia de usuario, es una técnica que se utiliza para obtener los requisitos funcionales y no funcionales que deberán tenerse en cuenta durante el desarrollo, el propio cliente es el que lo crea describiendo sus necesidades. Estas historias se convierten en tareas de programación las cuales son asignadas a los programadores en cada iteración.

Las características esenciales de esta metodología son las siguientes:

- **Comunicación:** Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

- **Simplicidad:** Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se requieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.

- **Realimentación (Feedback):** Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto (Ferreira Escutia, 2014).

Justificación de selección de la metodología

Luego de haberse estudiado en profundidad las características y particularidades, tanto de los enfoques ágiles como de los tradicionales, se decidió que para el desarrollo de la presente solución se adoptará una metodología ágil, basándose en la necesidad de realizar entregas parciales de la solución en el menor tiempo posible, dándole prioridad a las funcionalidades más esenciales y aprovechando la flexibilidad de las tareas asignadas. Como metodología de desarrollo, fue seleccionada AUP-UCI en su escenario 4. Para ello, se tuvo en cuenta que:

- El equipo de desarrollo es pequeño (una persona).
- Se logra una constante y perfecta comunicación con el cliente, integrándose este al equipo de desarrollo.
- Los requisitos de software pueden cambiar durante el proceso de implementación.
- El software debe ser implementado en un tiempo relativamente corto, haciendo iteraciones con el cliente, logrando un diseño simple y claro.

Después de realizar el estudio de la metodología y determinar que es factible el uso de AUP-UCI para guiar el desarrollo del portal web y de los componentes propuestos a partir de las fases definidas en ella; se determinó las tecnologías a utilizar.

1.4 Ambiente de Desarrollo

1.4.1 Entornos de Desarrollo

Los entornos de Desarrollo integrado son programas informáticos que contienen un conjunto de herramientas que son muy útiles por los desarrolladores de software, ya que estos optimizan la tarea de escribir programas, corregirlos y ejecutarlos. Ha sido empaquetado como un programa de aplicación, es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (XETID, 2020)

Visual Studio Code v1.31.0

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un idioma dado (Microsoft, 2016).

VS Code es un editor de código fuente sofisticado que admite muchas funcionalidades practicas al momento de trabajar con el código. Estas son algunas de ellas:

Lenguajes de programación: La edición de código no está limitada para C# y VB (lenguajes propietarios de Microsoft) si no que de nueva cuenta el Open Source está en el paquete: Java, Go, C, C++, Ruby, Python, PHP, Perl, JavaScript, Groovy, Swift, PowerShell, Rust, DockerFile, CSS, HTML, XML, JSON, Lua, F#, Batch, SQL.

Multiplataforma: Fue creado y diseñado para que funcione en los tres sistemas operativos mayormente utilizados: Windows, Linux y Mac OS.

Plugins: VS Code es una herramienta que se actualiza constantemente, tiene la posibilidad de adaptar plugins para trabajar con el cómputo en la nube de Microsoft Azure y desplegar proyectos directamente.

Acerca de los Proyectos: Las plantillas que se encuentran en Visual Studio ayudan a construir la estructura base de los proyectos, con VS Code también podemos crearlos, pero siempre desde cero (Cloud, 2016).

Por lo anteriormente expuesto se escoge el Visual Studio Code v1.31.0 como editor de código por la flexibilidad que brinda para el desarrollo de aplicaciones, además que su mayor potencial se encuentra en las extensiones, y es que desde el mismo editor se acceder a una gran variedad de extensiones y complementos que permitirá dotarle de las funciones y características necesarias para el desarrollo de la herramienta.

Visual Paradigm

Es una herramienta, para modelado en UML. Esta herramienta tiene características gráficas muy cómodas, que facilitan la realización de los diagramas de modelado que sigue el estándar de UML; los mismos son: diagramas de clases, casos de uso, comunicación, secuencia, estado, actividad, componentes. Entre sus principales ventajas se encuentran: que presenta entorno de creación de diagramas para UML 2.0; su diseño es centrado en casos de uso y enfocado al negocio, lo cual genera un software de mayor calidad; posee uso de un lenguaje estándar común a todo el equipo de desarrollo, que facilita la comunicación, las capacidades de ingeniería directa e inversa, modelo y código, que permanece sincronizado en todo el ciclo de desarrollo.

Las ventajas que proporciona Visual Paradigm para UML son:

Dibujo: Facilita el modelado de UML, ya que proporciona herramientas específicas para ello.

Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.

Generación de código: Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.

Generación de informes: Permite generar diversos informes a partir de la información introducida en la herramienta.

Para el modelado de los artefactos obtenidos durante el proceso de desarrollo se utilizará Visual Paradigm, ya que está disponible en múltiples plataformas, soporta aplicaciones web, está disponible en varios idiomas, además de disponer de fácil instalación y uso, además de presentar compatibilidad entre versiones (PARADIGM, 2018).

1.4.2 Lenguaje de Desarrollo

HTML5

HTML5 es la última evolución de la norma que define el lenguaje de marcas de hipertexto (HTML por sus siglas en inglés). Se trata de una nueva versión del lenguaje, con nuevos elementos, atributos y comportamientos, y un conjunto amplio de tecnologías que permite crear sitios web y las aplicaciones más diversas y de gran alcance. Este nuevo estándar de HTML permite que el formato de las webs formado por elementos como cabecera, pie y navegadores se agrupe en nuevas etiquetas que representan cada una de las partes típicas de una página. Se utilizará el lenguaje HTML5 para el diseño de las interfaces del subsistema, aprovechando las características del soporte para CSS3 y el manejo mejorado de formularios en el navegador (Alvarez, 2001).

Hojas de estilo en cascada (CSS 3)

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. Además de que separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

El lenguaje CSS se utiliza 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, entre otros (Eguiluz Pérez, 2008).

NodeJS v18.18.0:

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque

de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla (Finley, 2014).

NodeJS permite que tanto la base de datos, junto a los métodos que se encargan de gestionar su información, y la parte frontal del sistema compartan un único lenguaje y así facilitar la velocidad de las consultas y los tiempos de respuesta.

Lenguaje Unificado de Modelado (UML) 2.5.1

UML, abreviatura de Unified Modeling Language, es un lenguaje de modelado estandarizado que consiste en un conjunto integrado de diagramas, desarrollado para ayudar a los desarrolladores de sistemas y software a especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como para el modelado de negocios y otros sistemas que no son de software. La UML representa una colección de las mejores prácticas de ingeniería que han demostrado tener éxito en el modelado de sistemas grandes y complejos. La UML es una parte muy importante del desarrollo de software orientado a objetos y del proceso de desarrollo de software. La UML utiliza principalmente notaciones gráficas para expresar el diseño de proyectos de software. El uso de la UML ayuda a los equipos de proyecto a comunicarse, explorar posibles diseños y validar el diseño arquitectónico del software. En este artículo, le daremos ideas detalladas sobre lo que es UML, la historia de UML y una descripción de cada tipo de diagrama UML, junto con ejemplos de UML (Visual Paradigm, 2022).

1.3.3 Sistema Gestor de Base de Datos

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales orientadas a objetos y de código abierto que incluye diversas características tales como la herencia múltiple, funciones, restricciones, etc. El desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de

desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales. Dicha comunidad es denominada el P

pgAdmin 4 v4.30

Es la principal herramienta de gestión de código abierto para Postgres, está diseñada para satisfacer las necesidades sus usuarios, tanto principiantes como experimentados, proporcionando una potente interfaz gráfica que simplifica la creación, el mantenimiento y el uso de los objetos de la base de da- tos. Consiste en un servicio web desarrollado en Python que permite acceder a todas las funcionalidades de la base de datos, consulta, manipulación y gestión de datos. Además, puede ser ejecutas en múltiples plataformas entre ellas Linux, Mac OS X y Windows (pgAdmin, 2021).

Conclusiones Parciales

El análisis del estado del arte de los principales conceptos asociados al dominio del problema permitió una mejor comprensión del proceso que siguen los sistemas de gestión de la información académica y cómo se realiza la práctica profesional. Los sistemas de gestión de la información académica analizados permitieron conocer que ciertas funcionalidades no pueden faltar en la solución a desarrollar. Las ventajas presentadas en el análisis de los sistemas homólogos constituyen una fuente inicial para la definición de requisitos. En el desarrollo de la aplicación se emplea la metodología AUP-UCI, la herramienta case Visual Paradigm for UML para modelar el sistema y los lenguajes de programación Pyhton para el lado del servidor y como lenguajes del lado del cliente HTML5, CSS3 y JavaScript. Se utilizará el framework de desarrollo Django.

CAPÍTULO 2. DISEÑO DE LA PROPUESTA DE SOLUCIÓN

El presente capítulo aborda los principales aspectos relacionados con las características de la propuesta de solución. Se identifican los requisitos funcionales y no funcionales con los que debe cumplir la solución propuesta, así como el estilo arquitectónico y los patrones de diseño para lograr buenas prácticas en el diseño y posterior implementación del sistema. Igualmente se muestran los principales artefactos de ingeniería de software propuestos por la metodología utilizada.

2.1 Descripción de la propuesta

La propuesta de solución tiene como objetivo facilitar el proceso de evaluación y ubicación en la PID de los estudiantes de la facultad 2. Los usuarios que posean el rol de administrador serán los encargados del mantenimiento y correcto funcionamiento del sistema. Además, entre sus tareas se encuentran la gestión de usuarios y de las áreas. Por otra parte, los usuarios con el rol de Jefe de área serán los encargados de registrar a los estudiantes en el sistema, además de gestionar los proyectos, el banco de problemas y de asignar los estudiantes a proyectos. También, el usuario con el rol Encargado, serán los que gestionen las evaluaciones de los estudiantes en el sistema.

El sistema contará con una interfaz con los colores representativo de la Universidad de las Ciencias Informáticas además del logo. Presentará interfaz intuitiva, fácil de recordar para los usuarios y sin mucha carga de información.

2.1.1 Roles y permisos

Para garantizar la seguridad del sistema y la protección de la información, se le asignan roles y permisos a los diferentes usuarios, para poder controlar las acciones que se puedan realizar y las funcionalidades que ofrece el sistema en dependencia del usuario. A continuación, en la tabla 2 se muestran los roles y permisos otorgados para acceder al mismo:

Tabla 2 Roles y permisos en el sistema

Rol	Permisos
Administrador	Usuario con permiso para gestionar los usuarios y las áreas
Jefe de área	Usuario con permisos para gestionar banco de problemas, los estudiantes, los proyectos y la asignación de los estudiantes a proyectos.
Encargado	Usuario con permisos para gestionar las evaluaciones de los estudiantes.

2.2 Identificación de funcionalidades

El propósito de la definición de requisitos es especificar las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar.

2.2.1 Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville, 2016).

Tabla 3 *Requisitos Funcionales*

No.	Requisitos	Descripción	Prioridad
RF 1	Autenticar usuario	El sistema permite al usuario iniciar sesión en el sistema.	Baa
RF 2	Insertar área	El sistema permite al usuario de rol Administrador insertar un área en el sistema.	Alta

RF 3	Modificar área	El sistema permite al usuario de rol Administrador modificar un área del sistema.	Baja
RF 4	Eliminar área	El sistema permite al usuario de rol Administrador eliminar un área del sistema.	Baja
RF 5	Listar áreas	El sistema permite al usuario de rol Administrador listar las áreas existentes en el sistema.	Media
RF 6	Insertar estudiante	El sistema permite al usuario de rol Jefe de área registrar un estudiante en el sistema.	Alta
RF 7	Modificar estudiante	El sistema permite al usuario de rol jefe de área modificar un estudiante del sistema.	Media
RF 8	Eliminar estudiante	El sistema permite al usuario de rol jefe de área eliminar un estudiante del sistema.	Baja
RF 9	Listar estudiantes	El sistema permite al usuario de rol jefe de área listar los estudiantes existentes en el sistema.	Alta
RF 10	Insertar usuario	El sistema permite al usuario de rol Administrador registrar un usuario en el sistema.	Baja
RF 11	Modificar usuario	El sistema permite al usuario de rol Administrador modificar un usuario del sistema.	Baja
RF 12	Eliminar usuario	El sistema permite al usuario de rol Administrador eliminar un usuario del sistema.	Baja

RF 13	Listar usuarios	El sistema permite al usuario de rol Administrador listar los usuarios existentes en el sistema.	Media
RF 14	Insertar proyecto	El sistema permite al usuario de rol jefe de área registrar un proyecto en el sistema.	Alta
RF 15	Modificar proyecto	El sistema permite al usuario de rol jefe de área modificar un proyecto del sistema.	Alta
RF 16	Eliminar proyecto	El sistema permite al usuario de rol jefe de área eliminar un proyecto del sistema.	Media
RF 17	Listar proyectos	El sistema permite al usuario de rol jefe de área listar los proyectos existentes en el sistema.	Alta
RF 18	Asignar estudiantes a proyectos	El sistema permite a los usuarios de rol jefe de área asignar estudiantes a los proyectos existentes en el sistema.	Alta
RF 20	Registrar problema en el banco de problemas	El sistema permite al usuario de rol jefe de área registrar problema en el sistema.	Alta
RF 21	Modificar problema en el banco de problemas	El sistema permite al usuario de rol jefe de área modificar problema del sistema.	Media
RF 22	Eliminar problema en el banco de problemas	El sistema permite al usuario de rol jefe de área eliminar un problema del sistema.	Baja

RF 23	Mostrar banco de problemas	El sistema permite al usuario de rol jefe de área mostrar el banco de problemas.	Alta
RF 24	Insertar evaluación	El sistema permite al usuario de rol Encargado insertar una evaluación en el sistema.	Alta
RF 25	Modificar evaluación	El sistema permite al usuario de rol Encargado modificar una evaluación del sistema.	Media
RF 26	Visualizar registro de evaluaciones	El sistema permite al usuario de rol Encargado visualizar el registro de evaluaciones.	Alta
RF 27	Exportar registro de evaluaciones	El sistema permite al usuario de rol Encargado imprimir el registro de evaluaciones.	Alta

2.2.2 Requisitos No Funcionales

Son restricciones de los servicios o funciones que debe cumplir el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2016)

Tabla 4 *Requisitos No Funcionales*

No.	Descripción	Tipo
RnF 1	Los términos utilizados se establecen acorde al negocio correspondiente para facilitar la comprensión del usuario en el sistema.	Usabilidad
RnF 2	El sistema se detiene si se ve afectado por ausencia de conexión a través de la red.	Confiabilidad

RnF 3	Si el funcionamiento del sistema se ve interrumpido por el fluido eléctrico, una vez que vuelva este, el sistema se reanuda y continúa realizando las operaciones normalmente.	
RnF 4	La velocidad de procesamiento de la información será relativamente rápida, no mayor de 20 segundos en cada inserción.	Rendimiento
RnF 5	Se asignan los permisos de acceso, escritura, lectura en dependencia del rol que desempeñe cada usuario del sistema.	Seguridad

2.3 Historias de usuario

Las Historias de Usuarios sirven para escribir lo que el usuario desea ser capaz de hacer. Además, se centran en el valor que viene de usar el sistema en lugar de una especificación detallada de lo que el sistema debe hacer. Están concebidos como un medio para fomentar la colaboración (Angel Luis Lozano Sánchez, 2016). A continuación se detallan las HU, el resto se encuentra en el anexo 1.

Tabla 5 *HU1 Gestionar áreas*

Historia de Usuario	
Número: 1	**Nombre:** Gestionar áreas
Usuario: Administrador	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Estimación: 0.6	
Descripción Se debe tener la posibilidad de gestionar la información de las áreas. La funcionalidad Gestionar áreas permite listar, modificar, insertar y eliminar área. **Acciones para lograr el objetivo (precondiciones y datos):**	

- El usuario debe estar autenticado como rol Administrador. **Flujo de la acción a realizar:** El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar las áreas. Al seleccionar esta opción se muestra una tabla con la información de las áreas permitiendo modificar, eliminar e insertar una nueva área.
Prototipo de interfaz de usuario:

Tabla 6 *HU2 Gestionar estudiantes*

Historia de Usuario	
Número: 2	**Nombre:** Gestionar estudiantes
Usuario: Jefe de área	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Estimación: 0.6	
Descripción Se debe tener la posibilidad de gestionar la información de los estudiantes. La funcionalidad Gestionar estudiantes permite listar, modificar, insertar y eliminar estudiante. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe estar autenticado como rol jefe de área. **Flujo de la acción a realizar:** El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar los estudiantes. Al seleccionar esta opción se muestra una tabla con la información de los estudiantes permitiendo modificar, eliminar e insertar un nuevo estudiante.	
Prototipo de interfaz de usuario:	

Tabla 7 *HU9 Exportar registro de evaluaciones*

Historia de Usuario	
Número: 9	**Nombre:** Exportar registro de evaluaciones
Usuario: Encargado	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Estimación: 0.6	
Descripción: **Objetivo:** permite que el usuario exporte el registro de evaluaciones. **Condiciones para lograr el objetivo:** El usuario debe estar autenticado como rol Encargado. **Flujo de la acción a realizar:** Luego de mostrar el registro de evaluaciones, se activa un botón exportar, que permite exportar el modelo de este registro.	
Observaciones:	

2.4 Patrón Arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2014).

El patrón arquitectónico seleccionado es Modelo Vista Controlador (MVC). El modelo MVC (Modelo, Vista, Controlador) es un esquema de arquitectura por capas muy utilizado en el desarrollo de software basado en aplicaciones web. Además,

este patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos de forma que las modificaciones del componente de la vista puedan ser hechas con un mínimo de impacto en el componente del modelo de datos (Ávila del Campo, 2019).

Modelo: Representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.

Vista: Convierte el modelo en una página web que facilita al usuario interactuar con ella.

Controlador: Es el encargado de procesar las interacciones del usuario y ejecutar los cabios adecuados en el modelo o en la vista. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para peticiones (HTTP, consola de comandos, email, etc.).

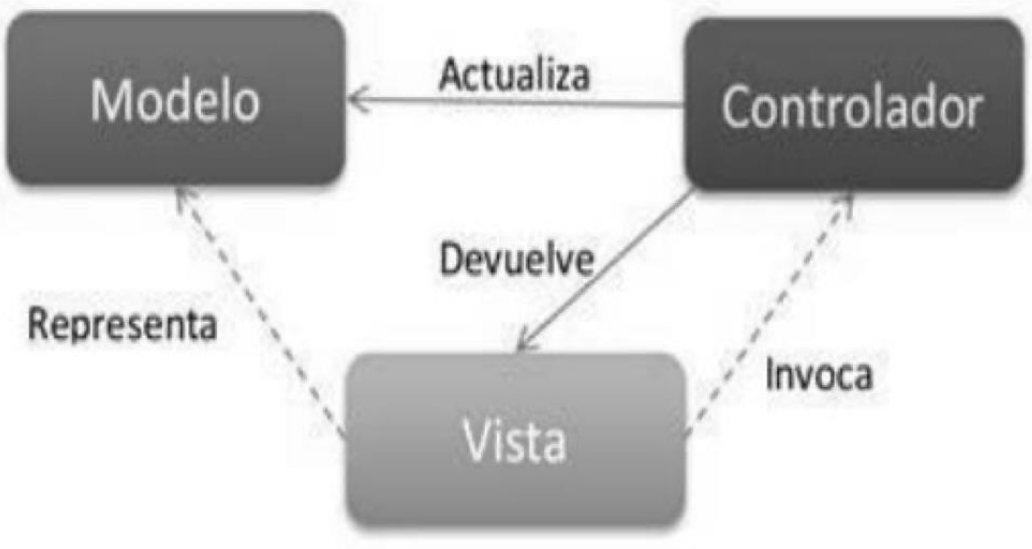


Figura 1 *Representación del Modelo Vista Controlador.*

2.5 Patrones de Diseño

Los patrones de diseño constituyen una descripción de un problema y la solución, la misma recibe un nombre y se puede aplicar a varios contextos. Muchos patrones

proporcionan guías en el modo en que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (Larman, 2003).

Patrones de Principios Generales para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. A continuación, se detalla el uso de estos patrones en la solución:

- **Experto (Expert):** consiste en asignar una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. Se evidencia en clases como: LayerModel y Layer, debido que cada una de ellas es responsable de manejar su información.

- **Creador (Creator):** permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades, para la vista del usuario. Se evidencia en la clase LayersController, la cual utiliza la información de la clase LayerModel para crear objetos de ella.

- **Controlador (Controller):** se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. Se evidencia en la clase DesignerController, la cual controla el flujo de eventos del módulo.

- **Bajo Acoplamiento (Low Coupling):** el acoplamiento mide el grado en que una clase está conectada, tiene conocimiento o de alguna manera depende de otra. Este patrón consiste en asignar la responsabilidad de manera que el acoplamiento permanezca bajo. El bajo acoplamiento permite crear clases más independientes, más reutilizables, lo que implica mayor productividad. Se evidencia en todo el diseño del sistema ya que cada clase solo se relaciona con otras que necesite, para evitar que al realizar un cambio en alguna de ellas se afecten lo menos posible las otras clases.

- **Alta Cohesión (High Cohesion):** en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se evidencia en las clases Layer y LayerModel ya que cada clase contiene la información y los métodos correspondientes específicos de ellas, logrando que no se sobrecarguen con información de otras clases.

Patrones de la banda de los cuatro (GoF)

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la banda de los cuatro (del inglés, Gang of Four). Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además introducir más notación UML. Los patrones de diseño del grupo GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Demestre & González, 2021).

- **Patrones creacionales:** Son los que facilitan la tarea de creación de nuevos objetos, de tal forma que el proceso de creación pueda ser desacoplado de la implementación del resto del sistema. En este grupo están los patrones que sirven de guía para construir objetos. Además, independizan el ¿Qué?, ¿Quién?, ¿Cómo? y ¿Cuándo? en la creación del objeto (Astorga et al., 2019).

    - **Prototipo:** El patrón de prototipo se usa cuando la creación de objetos es costosa, requiere mucho tiempo y recursos y ya poseemos un objeto similar. Este patrón proporciona un mecanismo para copiar el objeto original a un nuevo objeto y luego modificarlo de acuerdo a nuestras necesidades. Es utilizado por la clase DesignerController al disponer del Árbol de Capas como un prototipo para la elaboración de la leyenda y permitir su posterior modificación en dependencia de las capas habilitadas.

- **Singleton:** El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Se evidencia en la clase DesignerController, la cual define varias variables globales de instancia única que permiten la comunicación con el modelo y demás clases.

- **Patrones estructurales:** Un patrón estructural de clases utiliza la herencia para componer interfaces o implementaciones. Un patrón estructural de objetos describe la forma en que se componen objetos para obtener nueva funcionalidad, además se añade la flexibilidad de cambiar la composición en tiempo de ejecución. En este grupo encontramos los patrones que organizan las clases y objetos de características similares y así formar estructuras más grandes (Astorga et al., 2019).

    - **Proxy:** Es una clase que funciona como interfaz hacia cualquier otra cosa: una conexión a Internet, un archivo en disco o cualquier otro recurso que sea costoso o imposible de duplicar. Este patrón es empleado por la clase DesignerController.

    - **Decorator:** Permite añadir funcionalidad extra a un objeto (de forma dinámica o estática) sin modificar el comportamiento del resto de objetos del mismo tipo. Se ve reflejado en las clases cliente de la vista al heredar el código que es común en todas las páginas del sistema, para no tener que repetirlo en cada una de estas.

- **Patrones de comportamiento:** Buscan proporcionar al usuario del código objetos que aporten una funcionalidad determinada. De este modo los objetos presentan una firma cuyos métodos resuelven un problema específico sin exponer su funcionamiento (Astorga et al., 2019).

    - **Observer:** Los objetos son capaces de suscribirse a una serie de eventos que otro objetivo va a emitir, y serán avisados cuando esto

ocurra. Este patrón se utiliza en la clase LayersController para controlar las llamadas sincrónicas a la base de datos; así como en las clases del paquete vista para realizar las acciones deseadas luego de concluir una interacción del mapa.

2.6 Modelo de datos

Un modelo de datos describe los datos que apoyan los procesos de una organización y refleja con exactitud cómo serán guardados los datos en la base de datos. Contiene cada una de las tablas de la aplicación, así como sus atributos y relaciones. El modelo se encuentra normalizado en Tercera Forma Normal, por lo que todos los atributos de las tablas del modelo, dependen únicamente de la llave primaria.

Figura 2 *Modelo de datos*

2.8 Estándares de codificación

Se entiende como estándar de código a un conjunto de convenciones establecidas de ante mano (denominaciones, formatos, etc.) para la escritura de código. Estos

estándares varían dependiendo del lenguaje de programación elegido y además varían en cobertura, algunos son más extensos que otros (Vera Pasabare, 2019). Para la implementación del módulo se hacen uso de los siguientes estándares de codificación definidos por Jairo Rojas Delgado para el desarrollo de la plataforma ULTRON:

- Añadir comentarios para explicar determinadas partes de tu código.
- Al elegir nombres de las variables, se recomienda el utilizar variables descriptivas.
- Las funciones tienen que seguir el siguiente esquema de nombrado: modulo_sub- modulo_nombre_funcion ().
- Las funciones se escriben en minúsculas y separados con guiones de suelo, y los argumentos que reciben son descriptivos.
- Todo el código desarrollado tendrá una indentación de 4 espacios.
- La indentación se realizará siempre con tabuladores y no con cuatro puntos cuidando de no mezclar un estilo con el otro.
- Se dejará una línea en blanco inmediatamente después del inicio de un bloque de código, dígase bloques de funciones, bloques condicionales o bucles.
- La llave de apertura de un bloque de código se coloca después de la instrucción anterior y no en una línea en blanco independiente.
- La llave de clausura de un bloque de código tendrá el mismo nivel de indentación que la línea de la llave de apertura.
- Cada bloque de código anidado tendrá un nivel de indentación más que el bloque padre exceptuando las llaves de apertura y cierre.
- Todas las variables de un bloque de código se declararán al inicio del método en cuestión.
- Las variables se declaran una por línea y empleando la palabra reservada <var> que nunca podrá omitirse, aunque el lenguaje lo permita.

2.9 Diagrama de despliegue

El modelo de despliegue se realiza como parte de la implementación para describir la distribución física del sistema. Establece la correspondencia entre la arquitectura lógica, los procesos y nodos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre ellos (Rumbaugh et al., 2000). Se refleja en este artefacto los protocolos de comunicación mediante los cuales se comunican los nodos respectivos. Para la realización de este diagrama se tiene en cuenta la distribución cliente-servidor de la aplicación.

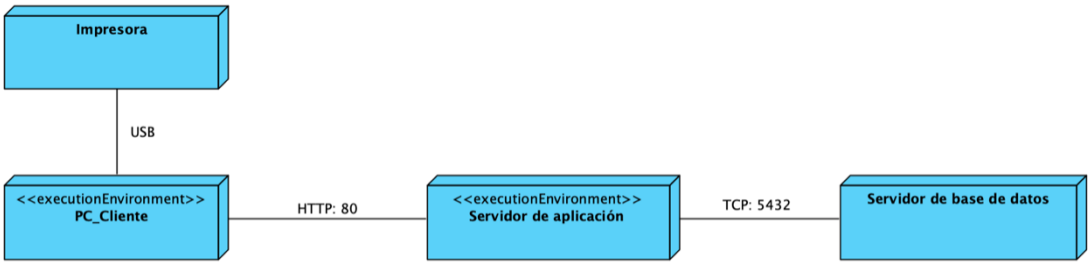


Figura 3 *Diagrama de despliegue*

PC cliente: es desde donde se accede al sistema a través del navegador web Mozilla Firefox v30.0 o superior, Chrome v35.0 o superior, Opera v12.0 o superior sin importar el sistema operativo.

Servidor de aplicación: es donde radica la lógica de negocio de la aplicación, compuesto por el Servidor Web Express y como lenguaje NodeJS.

Servidor de base de datos: utilizando PostgreSQL en su versión 11 como sistema gestor de base de datos es el encargado de almacenar los datos del sistema.

Conclusiones parciales

En este capítulo se abordaron los contenidos de las fases de exploración, planificación y diseño correspondiente a la metodología AUP-UCI con lo cual se

desarrolla el Sistema para la gestión de los procesos de evaluación y asignación de la PID. Lo anterior sirvió como guía para la implementación de dicho sistema. El estudio de los sistemas de gestión de información académica y las entrevistas realizadas permitieron definir 27 requisitos funcionales y 5 requisitos no funcionales. Al definir el modelo de datos se entendió y organizó mejor la información. Al aplicarse los patrones del diseño y arquitectónicos se creó una estructura común y conocida.

CAPÍTULO 3 VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

En este capítulo se realiza la validación de la propuesta con el fin de evaluar la calidad y fiabilidad del sistema. Según la metodología utilizada serán ejecutadas pruebas al producto desarrollado para verificar su correcto funcionamiento, con el objetivo de detectar y corregir las posibles no conformidades en los futuros usuarios.

3.1 Fase de pruebas de software

El proceso se realiza de forma contínua a lo largo del proceso de desarrollo con el fin de asegurar el éxito en el producto final. Las pruebas se llevaran a cabo con la finalidad de demostrar que el programa hace lo que se pide que realice, así como descubrir defectos en el programa antes de usarlo. Para ello, se define una estrategia a seguir, teniendo en cuenta la propuesta de la metodología en dicha fase. Se define el método de prueba de caja blanca, con el objetivo de verificar el código, empleando el tipo de pruebas unitarias con el uso de la técnica de camino básico. Para lograr las pruebas de aceptación implícitas en el método de pruebas de caja negra, se aplican los casos de pruebas para verificar el funcionamiento del sistema. Posteriormente se analista el comportamiento de los resultados de las pruebas en las distintas iteraciones realizadas.

3.2 Pruebas unitarias

Las pruebas unitarias se realizan sobre las funcionalidades internas de un módulo y se encargan de comprobar los caminos lógicos, ciclos (bucles) y condiciones que debe cumplir el programa.

Para aplicar las pruebas unitarias, el autor de la presente investigación decide utilizar el método de Caja blanca. Con el empleo de este método es posible desarrollar casos de prueba que garanticen la ejecución, al menos una vez, de los caminos independientes (Pressman, 2014). Para aplicar este método se define la técnica de camino básico.

CAPÍTULO 3

Técnica de camino básico

La prueba de caja blanca que se realizará es el método del camino básico, a partir del cálculo de la complejidad ciclomática del algoritmo a ser analizado. Para realizar la prueba se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad.

1. **Notación del grafo de flujo:** Usando el código como base se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
    - Nodo: a cada círculo denominado nodo, representa una o más sentencias procedimentales.
    - Arista: las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
    - Región: las áreas delimitadas por aristas y nodos se denominan regiones.

2. **Complejidad Ciclomática:** Es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática se representa por CC (G) y se puede calcular de varias formas, la utilizada es:

    CC (G)= Aristas –Vértices + 2

La generación de los casos en los que se divide la prueba se puede realizar siguiendo unos sencillos pasos que se describen a continuación:

- Partiendo del código fuente se representa el grafo de flujo.
- Se determina la complejidad ciclomática.
- Se genera un conjunto básico de caminos, o sea, tantos caminos independientes como indica la complejidad ciclomática.
- Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico (Pressman, 2014).

A continuación, se muestra en la tabla la puesta en práctica del método del camino básico, donde se le aplica al método visible:

Tabla 8 *Técnica de camino básico para la funcionalidad editarEstudiantePorId()*

1	app.post("/editarestudiante/:id", async(req,res)=>{
	const id =req.params.id;
	const nombre = req.body.nombre;
	const apellidos = req.body.apellidos;
	const facultad = req.body.facultad;
	const area = req.body.area;

2	try{ const cant = await pool.query('SELECT * FROM estudiante WHERE id = $1', [id]);
3	if (cant.rowCount > 0) {
4	const result = await pool.query(`UPDATE estudiante SET nombre = $1, apellidos = $2, facultad = $3, areaid = $4 WHERE id = $5`, [nombre,apellidos,facultad,area,id]) console.log(result);
5	} catch (error) {
6	console.log(error); }
7	res.redirect("/estudiantes"); })

A continuación, se muestra un grafo que representa el flujo del código para el método editarEstudiantePorId()(ver Figura 4).

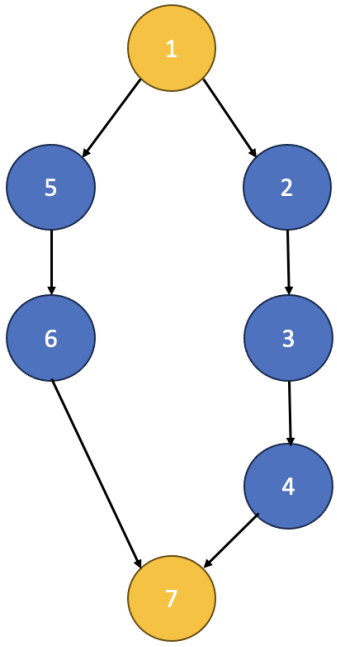


Figura 4 *Grafo de flujo de código del método editarTrazaPorId()*

La complejidad ciclomática puede ser calculada de 3 formas:

- V (G) = a- n + 2 = 7 - 7 + 2 = 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
- V (G) = r = 2, siendo r el número de regiones cerradas del grafo.
- V (G) = c + 1 = 1 +1 = 2, siendo c el número de nodos de condición.

  Por lo que el conjunto de caminos básico sería:

- **Camino Básico 1:** 1-2-3-4-7
- **Camino Básico 2:** 1-5-6-7

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación, se muestra el resultado de las pruebas aplicadas a los caminos básicos 1 y 2.

Tabla 9 *Caso de Prueba para el camino básico 1*

Caso de Prueba de Unidad	
Número de ruta: 1	**Ruta:** 1-2-3-4-7
Nombre de la persona que realiza la prueba:	
Descripción de la prueba: El usuario selecciona la opción editar estudiante.	
Entrada: Se recibe el id del estudiante que se desea editar.	
Resultado esperado: El sistema muestra la información del estudiante editada.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 10 *Caso de Prueba para el camino básico 2*

Caso de Prueba de Unidad	
Número de ruta: 2	**Ruta:** 1-5-6-7
Nombre de la persona que realiza la prueba:	
Descripción de la prueba: El usuario selecciona la opción editar estudiante.	
Entrada: Se recibe el id del estudiante que se desea editar.	

Resultado esperado: El sistema muestra un mensaje de error.
Evaluación de la Prueba: Satisfactoria.

3.3 Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema (lo que se va a testear, el software o una parte de él). Se realizan para comprobar si el software cumple las funciones esperadas (Pressman, 2014). Para aplicar este tipo de prueba se define aplicar el método de **Caja negra**. Este método permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La misma no es una alternativa a las técnicas de método de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir otros tipos de errores. Estas pruebas permiten encontrar: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a las bases de datos externas, errores de rendimiento, errores de inicialización y terminación (Pressman, 2014).

Para desarrollar el método de Caja negra se utilizan las técnicas:

- **Partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Análisis de valor de frontera:** prueba la habilidad del programa para manejar datos que se encuentran en los límites o fronteras aceptables.

Para aplicar este método, según las técnicas descritas anteriormente, se tienen en cuenta los Diseños de casos de pruebas (DCP). El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por su parte, el análisis de valor de frontera es una técnica de diseño de casos de prueba que complementan la partición de equivalencia. En lugar de seleccionar algún elemento de una clase de equivalencia, éste conduce a la selección de casos de prueba

en los “bordes” de la clase. En lugar de enfocarse exclusivamente en las condiciones de entrada, también deriva casos de prueba a partir del dominio de salida (Pressman, 2014).

Se realiza un DCP por cada funcionalidad evaluándose sus resultados. A continuación, se muestra un ejemplo de DCP para el requisito Adicionar Medida Disciplinarias a nivel de AC. Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato.

Descripción de las variables:

Tabla 11 *Descripción de las variables para el caso de prueba Insertar área () de la HU1*

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	ID	Campo de texto	No	Permite letras y números.
2	Nombre	Campo de texto	No	Permite letras, números y espacios.
3	Cantidad de trabajadores	Campo de texto	Si	Permite solo números.

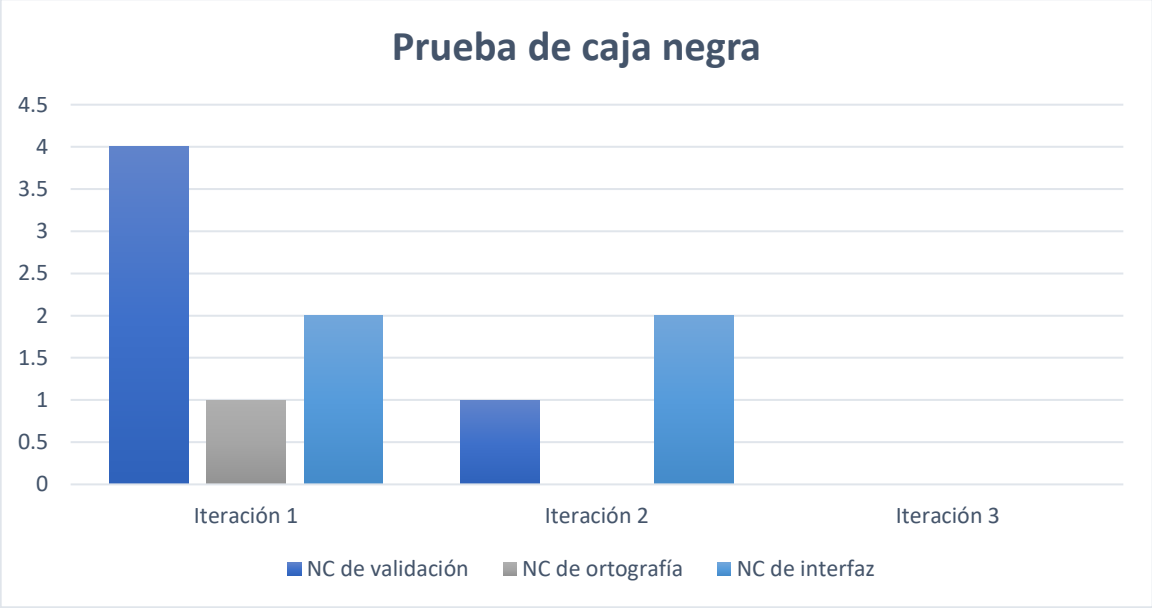
Tabla 12 *Caso de prueba para la funcionalidad Insertar área () de la HU1*

Escenario	ID	Nombre	Cantidad de trabajadores	Respuesta del sistema	Resultado
Introducir datos correctos	V	V	V	El sistema muestra un mensaje indicando la inserción exitosa.	Satisfactoria
	23FD4	CIGED	120		
Introducir datos incorrectos	F	V	V	El sistema muestra un mensaje indicando el campo incorrecto.	Satisfactoria
	22145$%	CIGED	120		
	V	F	V		

	23FD4	CIGED$	120		
	V	V	F		
	23FD4	CIGED	120!!!		
Dejar todos los campos vacíos	-	-	-	El sistema muestra un mensaje indicando llenar los campos.	Satisfactoria

Con el objetivo de comprobar el funcionamiento del componente se realizaron un total de tres iteraciones de pruebas, para poder alcanzar resultados satisfactorios, atendiendo al correcto comportamiento del sistema ante diferentes situaciones, obteniéndose los resultados que se muestran en la siguiente figura:

Figura 5 *Representación de la cantidad de NC por iteraciones*



Fuente: elaboración propia.

En una primera iteración se detectaron un total de siete NC (cuatro de validación, una de ortografía y dos de interfaz). En la segunda iteración se manifestaron tres NC (un error de validación y dos errores de interfaz). Finalmente, en una tercera iteración se obtuvieron resultados satisfactorios al no mostrar NC.

3.4 Pruebas de Aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tarea para las cuales fue construido. Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. Para el caso de la solución propuesta se diseñaron 6 casos de prueba (Ver Anexo 2: Diseño de Casos de Prueba de Aceptación). Para una primera iteración 14 casos de prueba fueron satisfactorios y 2 no satisfactorias. Para una segunda iteración 9 casos de prueba fueron satisfactoria y 1 no satisfactoria. Para una tercera iteración se obtuvo 1 caso de prueba no satisfactorio y 4 satisfactorios, resultados mostrados en la gráfica siguiente:

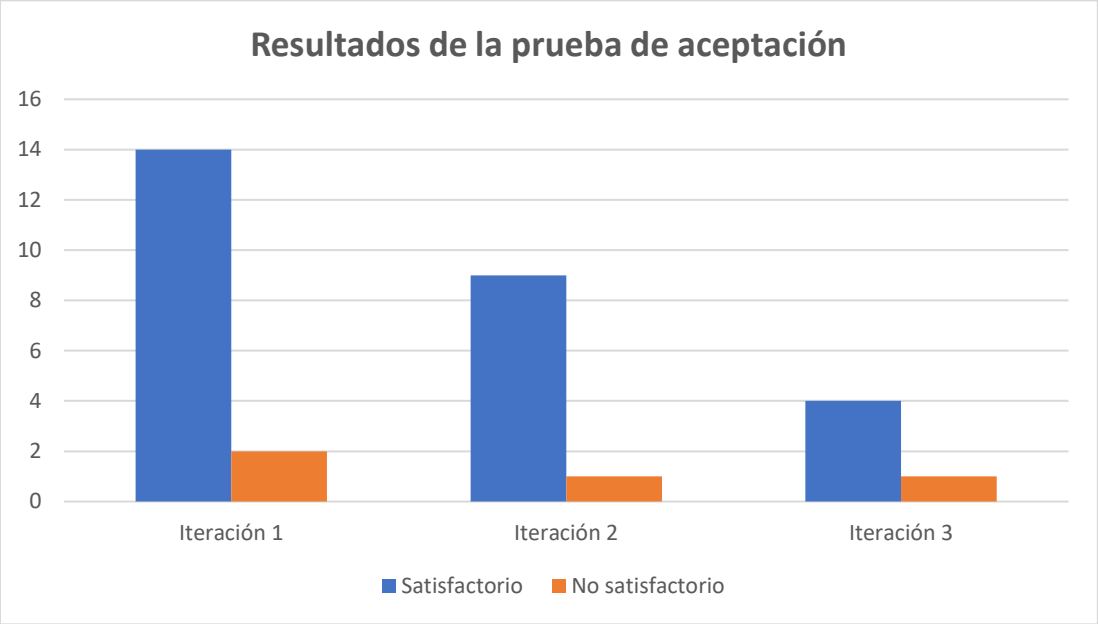


Figura 6 *Resultados de la Prueba de aceptación Fuente: Elaboración propia*

Conclusiones parciales

En el presente capítulo se logró ratificar que los requisitos de software definen el sistema que el cliente desea al aplicar las técnicas de validación definidas para los mismos, por lo que se puede concluir que:

- La ejecución de pruebas unitarias mediante el método caja blanca demostró que las funciones internas son operativas, facilitando la detección de no conformidades para su corrección.

- La ejecución de pruebas funcionales mediante el método caja negra demostró que las funciones son operativas a través de la interfaz del software, manteniendo así la integridad de la información externa.

- Las pruebas de aceptación evidenciaron que la solución propuesta cumple con las necesidades requeridas para ser entregado al cliente.

Conclusiones generales

Con la realización del presente trabajo de diploma se logra el objetivo general propuesto, llegando a las siguientes conclusiones:

- El establecimiento de los fundamentos teóricos-metodológicos del proceso de gestión de la asignación y evaluación de la PID en la facultad 2 permitió obtener la comprensión e identificación de los elementos esenciales para la propuesta de solución.

- Con la guía de la metodología definida y el empleo de las tecnologías y herramientas elegidas, se desarrolló el Sistema Informático para la gestión de la ubicación y evaluación en la PID de los estudiantes de la facultad 2.

- La aplicación de las pruebas trazadas en la estrategia, validó la propuesta de solución cumpliendo con la calidad y funcionalidades requeridas por el cliente.

Recomendaciones

En correspondencia con las conclusiones a las que se ha arribado, se recomienda:

- Investigar, Mejorar e incluir nuevas funcionalidades que puedan ser necesitadas por los clientes del sistema adecuándolo a las nuevas tecnologías.

Referencias Bibliográficas:

Alamaraz Hernández, J. M., Campos Cantero, P., & Castelo Delgado, T. (2011). *Desarrollo de una aplicación web para la gestión de Entornos Virtuales*.

Alarcón, M., & Gómez, A. (2006). *La formación laboral como cualidad de la personalidad*. Instituto Superior Pedagógico «José de la Luz y Caballero».

Alvarez, M. A. (2001). *Novedades de HTML 5—¿ Qué es HTML 5? Novedades de HTML 5—¿ Qué es HTML 5?* Desarrolloweb.com. http://www.desarrolloweb.com/articulos/que-es-html5.html

Astorga, J. A. I., Osuna Tirado, J. L., & Pereza Garzón, Á. (2019). *CLASIFICACIÓN DE LOS PATRONES DE DISEÑO IDÓNEOS EN PROGRAMACIÓN ANDROID*.

Ávila del Campo, A. (2019). *Diseño de un SIG para la ubicación óptima de una instalación minera*.

Bagrotti Abreu, C. (2017). *Sistema para la gestión de los procesos administrativos del CEIGE. Módulo para la Gestión de los procesos del Departamento de Práctica Profesional*.

Cloud, E. (2016). *Visual Studio Code ¿Qué es? ¿Qué no es?*

De la Torre Matamoro, J. C. (2013). *Web docente para la Práctica Profesional I de la Ingeniería Informática, en la semipresencialidad*.

Demestre, D. H., & González, y. h. (2021). *Diseño de un Sistema de Gestión de Información de Recursos Humanos.* 31-40.

Eguiluz Pérez, J. (2008). *Introduccion a CSS*.

Ferreira Escutia, R. (2014). *"XP Extreme Programming"*. https://slideplayer.es/slide/84721/

Finley, K. (2014). *The Ensembl REST API: Ensembl Data for Any Language*.

Larman, C. (2003). *UML y Patrones*.

Microsoft. (2016). *Visual Studio Code*.

Ministerio de Educación Superior. (2014). *Plan de estudio «D» Ingenieria en Ciencias Informáticas*.

Núñez Mayorga, G. M. (2009). *Desarrollo de una aplicación web para la gestión de calificaciones de alumnos.*

PARADIGM, V. (2018). *Visual Paradigm Product Overview*. https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

Pérez Figueroa, L. (2017). *Gestión de la información*. https://instituciones.sld.cu/toximed/2017/04/16/que-es-gestion-de-la-informacion/.

pgAdmin. (2021). *Pg Admin Development*. https://www.pgadmin.org/docs/pgadmin4/development/

Pressman. (2014). *Ingeniería del software: Un enfoque práctico, 7ma Edición – Roger S. Pressman*. https://www.freelibros.net/ingenieria/ingenieria-del-software-un-enfoque-practico-7ma-edicion-roger-s-pressman

Rodríguez Sánchez, T. (2015). *Metodología de Desarrollo para la actividad productiva de la UCI*.

Rumbaugh, J., Booch, G., & Jacobson, I. (2000). *El lenguaje unificado de modelado. Manual de referencia*. Madrid: Addison Wesley, 2000. https://dialnet.unirioja.es/servlet/libro?codigo=156687

SNE. (2009). *Reglamento de Enseñanza Práctica para los centros de la Educación Técnica y Profesional* (327/1985).

Sommerville, I. (2016). *Ingeniería del Software Séptima Edición* (Vol. 7). PEARSON ADDISON WEASLEY, SA. Madrid, 2005.

UCI. (2013). *Investigación y desarrollo*. https://www.uci.cu/investigacion-y-desarrollo/productos/xedro/gespro-1305.

Vera Pasabare, J. (2019). *LAS MEJORES PRACTICAS EN CODIFICACION DE SOFTWARE*.

Visual Paradigm. (2022). *What is Unified Modeling Language (UML)?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

XETID. (2020). *Proceso de Desarollo de Software*.

Anexos

Anexo 1 Historias de usuario

Tabla 13 *HU3 Gestionar usuarios*

Historia de Usuario	
Número: 3	**Nombre:** Gestionar usuarios
Usuario: Administrador	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Estimación: 0.4	
Descripción Se debe tener la posibilidad de gestionar la información de las áreas. La funcionalidad Gestionar usuarios permite listar, modificar, insertar y eliminar usuario. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe estar autenticado como rol Administrador. **Flujo de la acción a realizar:** El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar los usuarios. Al seleccionar esta opción se muestra una tabla con la información de los usuarios permitiendo modificar, eliminar e insertar una nueva área.	

Tabla 14 *HU4 Gestionar proyectos*

Historia de Usuario	
Número: 4	**Nombre:** Gestionar proyectos
Usuario: Jefe de área	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Estimación: 0.4	
Descripción Se debe tener la posibilidad de gestionar la información de los proyectos. La funcionalidad Gestionar proyectos permite listar, modificar, insertar y eliminar proyecto. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe estar autenticado como rol jefe de área.	

Flujo de la acción a realizar:
El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar los proyectos. Al seleccionar esta opción se muestra una tabla con la información de los proyectos permitiendo modificar, eliminar e insertar un nuevo proyecto.

Tabla 15 *HU5 Gestionar banco de problemas*

Historia de Usuario	
Número: 5	**Nombre:** Gestionar banco de problemas
Usuario: Jefe de área	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Estimación: 0.4	
Descripción Se debe tener la posibilidad de gestionar la información del banco de problemas. La funcionalidad Gestionar banco de problemas permite listar, modificar, insertar y eliminar un problema. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe estar autenticado como rol jefe de área. **Flujo de la acción a realizar:** El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar banco de problemas. Al seleccionar esta opción se muestra una tabla con la información de los problemas permitiendo modificar, eliminar e insertar un nuevo problema.	

Tabla 16 *HU7 Asignar estudiante*

Historia de Usuario	
Número: 7	**Nombre:** Asignar estudiante a proyecto
Usuario: Jefe de área	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Estimación: 0.4	
Descripción: **Objetivo:** permite que el usuario asigne un estudiante a un proyecto. **Condiciones para lograr el objetivo:**	

El usuario debe estar autenticado como rol jefe de área. **Flujo de la acción a realizar:** Cuando el usuario lista los estudiantes, se activa, además de otras funcionalidades, un botón “Vincular”. El usuario selecciona esta opción y se muestra un formulario con campos a llenar (proyecto, rol que desempeña) y los botones de “Aceptar” y “Cancelar”, Una vez llenos selecciona en aceptar para guardar la información, en caso de cancelar se regresa a la vista previa.

Tabla 17 *HU8 Gestionar evaluaciones*

Historia de Usuario	
Número: 8	**Nombre:** Gestionar evaluaciones
Usuario: Encargado	
Programador: Daidelín Ríos Guillén	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Estimación: 0.4	
Descripción Se debe tener la posibilidad de gestionar las evaluaciones de los estudiantes. La funcionalidad Gestionar evaluaciones permite registrar, visualizar y eliminar las evaluaciones de los estudiantes. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe estar autenticado como rol Encargado. **Flujo de la acción a realizar:** El usuario accede al sistema, una vez en la página de inicio se muestra la opción de gestionar las evaluaciones. Al seleccionar esta opción se muestra una tabla con la información de las evaluaciones de los estudiantes permitiendo registrar, visualizar y eliminar las evaluaciones de los estudiantes.	

Tabla 18 *HU6 Autenticar usuario*

Historia de Usuario	
Número: 6	**Nombre:** Autenticar Usuario
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Estimación: 0.2	
Descripción: **Objetivo:** permite que el usuario se autentique en el sistema. **Condiciones para lograr el objetivo:** • Ingresar los datos de Usuario y Contraseña. **Flujo de la acción a realizar:** Cuando el usuario toca el botón "Autenticar" ubicado en la parte superior derecha del sistema, se muestra una interfaz donde debe ingresar sus datos (usuario y contraseña), luego selecciona la opción "Autenticar" y si son correctos inicia sesión en el sistema. Si los datos son incorrectos se señala el campo en cuestión dando la posibilidad de realizar esa acción nuevamente. Si selecciona la acción "cancelar" regresa a la vista previa.	

Anexo 2: Diseño de Casos de Prueba de Aceptación

Tabla 19 *Caso de prueba de aceptación #1*

Caso de Prueba de Aceptación	
Código: HU1-P1	**Historia de Usuario:** HU-1
Nombre: Insertar área	
Descripción: el sistema debe permitir insertar una nueva área en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol administrador debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Insertar área y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Guardar y recibe un mensaje de notificación indicando el registro exitoso.	

Resultados esperados: Se inserta una nueva área en el sistema.
Evaluación de la prueba: No satisfactoria

Tabla 20 *Caso de prueba de aceptación #2*

Caso de Prueba de Aceptación	
Código: HU1-P2	**Historia de Usuario:** HU-1
Nombre: Modificar área	
Descripción: el sistema debe permitir modificar una área previamente insertada.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Administrador.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.	
Resultados esperados: Se modifica los datos de una área.	
Evaluación de la prueba: Satisfactoria	

Tabla 21 *Caso de prueba de aceptación #3*

Caso de Prueba de Aceptación	
Código: HU1-P3	**Historia de Usuario:** HU-3
Nombre: Eliminar área	
Descripción: el sistema debe permitir eliminar una área previamente insertada.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Administrador.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Eliminar y se le muestra una ventana de notificación para confirmar la eliminación, selecciona el botón Eliminar y recibe un mensaje de notificación indicando la eliminación exitosa.	
Resultados esperados: Se elimina un área del sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 22 *Caso de prueba de aceptación #4*

Caso de Prueba de Aceptación

Código: HU1-P4	**Historia de Usuario:** HU-1
Nombre: Listar áreas	
Descripción: el sistema debe permitir listar las áreas	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Administrador.	
Entradas/Pasos de ejecución: El usuario selecciona el botón áreas y se le muestra una tabla con todas las áreas existentes.	
Resultados esperados: Se listan las áreas.	
Evaluación de la prueba: Satisfactoria	

Tabla 23 *Caso de prueba de aceptación #5*

Caso de Prueba de Aceptación	
Código: HU2-P5	**Historia de Usuario:** HU-2
Nombre: Insertar estudiante	
Descripción: el sistema debe permitir insertar un estudiante en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol Jefe de área debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Insertar estudiante y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Guardar y recibe un mensaje de notificación indicando el registro exitoso.	
Resultados esperados: Se inserta un nuevo estudiante en el sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 24 *Caso de prueba de aceptación #6*

Caso de Prueba de Aceptación	
Código: HU2-P6	**Historia de Usuario:** HU-6
Nombre: Modificar estudiante	
Descripción: el sistema debe permitir modificar un estudiante previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	

Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.
Resultados esperados: Se modifica los datos de un estudiante.
Evaluación de la prueba: Satisfactoria

Tabla 25 *Caso de prueba de aceptación #7*

Caso de Prueba de Aceptación	
Código: HU2-P7	**Historia de Usuario:** HU-2
Nombre: Eliminar estudiante	
Descripción: el sistema debe permitir eliminar un estudiante previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Eliminar y se le muestra una ventana de notificación para confirmar la eliminación, selecciona el botón Eliminar y recibe un mensaje de notificación indicando la eliminación exitosa.	
Resultados esperados: Se elimina un estudiante del sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 26 *Caso de prueba de aceptación #8*

Caso de Prueba de Aceptación	
Código: HU2-P8	**Historia de Usuario:** HU-2
Nombre: Listar estudiantes	
Descripción: el sistema debe permitir listar estudiantes	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona el botón Usuarios y se le muestra una tabla con todos los estudiantes existentes.	
Resultados esperados: Se listan los estudiantes.	
Evaluación de la prueba: Satisfactoria	

Tabla 27 *Caso de prueba de aceptación #9*

Caso de Prueba de Aceptación	
Código: HU3-P9	**Historia de Usuario:** HU-3
Nombre: Registrar usuario	
Descripción: el sistema debe permitir registrar un nuevo usuario en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol administrador debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Registrar usuario y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Registrar y recibe un mensaje de notificación indicando el registro exitoso.	
Resultados esperados: Se registra un nuevo usuario en el sistema.	
Evaluación de la prueba: No satisfactoria	

Tabla 28 *Caso de prueba de aceptación #10*

Caso de Prueba de Aceptación	
Código: HU3-P10	**Historia de Usuario:** HU-3
Nombre: Modificar usuario	
Descripción: el sistema debe permitir modificar un usuario previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Administrador.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.	
Resultados esperados: Se modifica los datos de un usuario.	
Evaluación de la prueba: Satisfactoria	

Tabla 29 *Caso de prueba de aceptación #11*

Caso de Prueba de Aceptación	
Código: HU3-P11	**Historia de Usuario:** HU-3

Nombre: Eliminar usuario
Descripción: el sistema debe permitir eliminar un usuario previamente insertado.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Administrador.
Entradas/Pasos de ejecución: El usuario selecciona la opción Eliminar y se le muestra una ventana de notificación para confirmar la eliminación, selecciona el botón Eliminar y recibe un mensaje de notificación indicando la eliminación exitosa.
Resultados esperados: Se elimina un usuario del sistema.
Evaluación de la prueba: Satisfactoria

Tabla 30 *Caso de prueba de aceptación #12*

Caso de Prueba de Aceptación	
Código: HU3-P12	**Historia de Usuario:** HU-3
Nombre: Listar usuarios	
Descripción: el sistema debe permitir listar usuarios	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como almacenero.	
Entradas/Pasos de ejecución: El usuario selecciona el botón Usuarios y se le muestra una tabla con todos los usuarios existentes.	
Resultados esperados: Se listan los usuarios.	
Evaluación de la prueba: Satisfactoria	

Tabla 31 *Caso de prueba de aceptación #13*

Caso de Prueba de Aceptación	
Código: HU4-P13	**Historia de Usuario:** HU-4
Nombre: Insertar proyecto	
Descripción: el sistema debe permitir insertar un proyecto en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol Jefe de área debe estar autenticado en el sistema.	

Entradas/Pasos de ejecución: El usuario selecciona la opción Insertar proyecto y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Guardar y recibe un mensaje de notificación indicando el registro exitoso.
Resultados esperados: Se inserta un nuevo proyecto en el sistema.
Evaluación de la prueba: Satisfactoria

Tabla 32 *Caso de prueba de aceptación #14*

Caso de Prueba de Aceptación	
Código: HU4-P14	**Historia de Usuario:** HU-4
Nombre: Modificar proyecto	
Descripción: el sistema debe permitir modificar un proyecto previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.	
Resultados esperados: Se modifica los datos de un proyecto.	
Evaluación de la prueba: Satisfactoria	

Tabla 33 *Caso de prueba de aceptación #15*

Caso de Prueba de Aceptación	
Código: HU4-P15	**Historia de Usuario:** HU-4
Nombre: Eliminar proyecto	
Descripción: el sistema debe permitir eliminar un proyecto previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Eliminar y se le muestra una ventana de notificación para confirmar la eliminación, selecciona el botón Eliminar y recibe un mensaje de notificación indicando la eliminación exitosa.	
Resultados esperados: Se elimina un proyecto del sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 34 *Caso de prueba de aceptación #16*

Caso de Prueba de Aceptación	
Código: HU4-P16	**Historia de Usuario:** HU-4
Nombre: Listar proyectos	
Descripción: el sistema debe permitir listar proyectos	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona el botón proyectos y se le muestra una tabla con todos los proyectos existentes.	
Resultados esperados: Se listan los proyectos.	
Evaluación de la prueba: Satisfactoria	

Tabla 35 *Caso de prueba de aceptación #17*

Caso de Prueba de Aceptación	
Código: HU5-P17	**Historia de Usuario:** HU-5
Nombre: Insertar problema	
Descripción: el sistema debe permitir insertar un problema en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol Jefe de área debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Insertar problema y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Guardar y recibe un mensaje de notificación indicando el registro exitoso.	
Resultados esperados: Se inserta un nuevo problema en el sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 36 *Caso de prueba de aceptación #18*

Caso de Prueba de Aceptación	
Código: HU5-P18	**Historia de Usuario:** HU-5
Nombre: Modificar problema	

Descripción: el sistema debe permitir modificar un problema previamente insertado.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.
Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.
Resultados esperados: Se modifica los datos de un problema.
Evaluación de la prueba: Satisfactoria

Tabla 37 *Caso de prueba de aceptación #19*

Caso de Prueba de Aceptación	
Código: HU5-P19	**Historia de Usuario:** HU-5
Nombre: Eliminar problema	
Descripción: el sistema debe permitir eliminar un problema previamente insertado.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Eliminar y se le muestra una ventana de notificación para confirmar la eliminación, selecciona el botón Eliminar y recibe un mensaje de notificación indicando la eliminación exitosa.	
Resultados esperados: Se elimina un problema del sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 38 *Caso de prueba de aceptación #20*

Caso de Prueba de Aceptación	
Código: HU5-P20	**Historia de Usuario:** HU-5
Nombre: Listar problemas	
Descripción: el sistema debe permitir listar problemas	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Jefe de área.	
Entradas/Pasos de ejecución: El usuario selecciona el botón problemas y se le muestra una tabla con todos los problemas existentes.	
Resultados esperados: Se listan los problemas.	

Evaluación de la prueba: Satisfactoria

Tabla 39 *Caso de prueba de aceptación #21*

Caso de Prueba de Aceptación	
Código: HU6-P21	**Historia de Usuario:** HU-21
Nombre: Autenticar Usuario	
Descripción: Prueba para la funcionalidad Autenticar Usuario	
Condiciones de ejecución: El usuario debe estar registrado en el sistema.	
Entradas/Pasos de ejecución: Se introducen el nombre de usuario y la contraseña correspondiente al usuario que desea iniciar sesión, en caso de haber introducido algún dato incorrecto se muestra un mensaje de error.	
Resultados esperados: El usuario accede al sistema.	
Evaluación de la prueba: Satisfactoria	

Tabla 40 *Caso de prueba de aceptación #22*

Caso de Prueba de Aceptación	
Código: HU8-P22	**Historia de Usuario:** HU-8
Nombre: Insertar evaluación	
Descripción: el sistema debe permitir insertar una nueva evaluación en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.	
Condiciones de ejecución: El usuario de rol Encargado debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario selecciona la opción Insertar evaluación y se le muestra un formulario con los campos a llenar, una vez llena los campos selecciona el botón Guardar y recibe un mensaje de notificación indicando el registro exitoso.	
Resultados esperados: Se inserta una nueva evaluación en el sistema.	
Evaluación de la prueba: No satisfactoria	

Tabla 41 *Caso de prueba de aceptación #23*

Caso de Prueba de Aceptación	
Código: HU8-P23	**Historia de Usuario:** HU-8
Nombre: Modificar evaluación	

Descripción: el sistema debe permitir modificar una evaluación previamente insertada.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Encargado.
Entradas/Pasos de ejecución: El usuario selecciona la opción Modificar y se le muestra un formulario con los campos que puede cambiar, una vez modifica los campos selecciona el botón Modificar y recibe un mensaje de notificación indicando la modificación exitosa.
Resultados esperados: Se modifica los datos de una evaluación.
Evaluación de la prueba: Satisfactoria

Tabla 42 *Caso de prueba de aceptación #24*

Caso de Prueba de Aceptación	
Código: HU8-P24	**Historia de Usuario:** HU-8
Nombre: Mostrar evaluaciones	
Descripción: el sistema debe permitir listar las evaluaciones	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema como Encargado.	
Entradas/Pasos de ejecución: El usuario selecciona el botón evaluaciones y se le muestra una tabla con todas las evaluaciones existentes.	
Resultados esperados: Se muestran las evaluaciones.	
Evaluación de la prueba: Satisfactoria	

Tabla 43 *Caso de prueba de aceptación #25*

Caso de Prueba de Aceptación	
Código: HU9-P25	**Historia de Usuario:** HU-9
Nombre: Exportar	
Descripción: el sistema debe permitir exportar el registro de evaluaciones.	
Condiciones de ejecución: El usuario de rol almacenero debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: Luego de mostrar el registro de evaluaciones, se activa un botón exportar, que permite exportar el modelo de este registro.	
Resultados esperados: Se exportar correctamente el registro de evaluaciones.	
Evaluación de la prueba: No satisfactoria	