Facultad de Ciencias y Tecnologías Computacionales

Sistema para la gestión de certificación de estudios terminados en la dirección municipal de educación de Alquízar

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

Autor: Alexander Rizo Sánchez

Tutores: Inst. Ing. Laritza Asán Caballero Ms.C.
Ing. Arnaldo Robaina Blanco

La Habana, noviembre de 2023

“Año 65 de la Revolución”

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título **“Sistema para la gestión de certificación de estudios terminados en la dirección municipal de educación de Alquízar”** 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 firma la presente a los 6 días del mes de noviembre del año 2023.

Alexander Rizo Sánchez

_____
Firma del Autor

Ing. Arnaldo Robaina Blanco

_____
Firma del Tutor

Inst. Ing. Laritza Asán Caballero Ms.C.

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Arnaldo Robaina Blanco

E-mail: arobaina92@vivaldi.net

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas (Habana) en febrero 2023.

Administrador de redes y sistemas informáticos en la Empresa SOFTEL, MICOM

Especialista principal en sistemas de Firma Digital y PKI. Consultor en implantación de software y configuraciones de red para utilizar Firma Digital en empresas clientes.

Tutor: Inst. Ing. Laritza Asán Caballero Ms.C.

E-mail: lasan@uci.cu

Ingeniera en Ciencias Informáticas, graduada en la Universidad de las Ciencias Informáticas en 2013.

Por 6 años fue desarrollador en el Centro de Geoinformática y Señales Digitales.

Actualmente pertenece al Departamento de Bioinformática de la Facultad de Ciencias y Tecnologías Computacionales como profesora.

Graduada de Máster en Informática Avanzada en 2022.

AGRADECIMIENTOS

La realización de este trabajo de diploma, nunca hubiese sido posible, sin el apoyo incondicional de las personas a las cuales quiero agradecer profundamente:

A mis tutores Arnaldo y Laritza.

A mis amigos Pedro Pablo, Juan Carlos y Manuel.

A mis compañeros de grupo en especial al “Equipo Z”.

A todos los profesores del CITEC que nos transmitieron sus conocimientos.

Por último, pero no menos importante, a nuestras secretarias docentes.

DEDICATORIA

A mi esposa e hijo por ser la inspiración de todo mi esfuerzo.

A mi mamá por estar ahí siempre y ser mi razón de ser.

A mi padrastro por apoyarnos siempre a mi madre y a mí.

A mi papá que está en el cielo.

A mis hermanas que las amo con la vida.

A toda mi familia y amigos que siempre me apoyaron de una forma u otra.

RESUMEN

El Ministerio de Educación (MINED) a lo largo de los años ha desarrollado un programa de informatización de la gestión académica, con el fin de humanizar el trabajo en los diferentes centros educativos. En este sentido podemos decir que el proceso de emisión de certificación de estudios terminados del MINED posee bajos niveles de informatización, lo cual no contribuye al proceso de estandarización y seguridad de estos documentos. Por tanto, esta investigación se enfoca en el proceso de gestión académica en las direcciones municipales de educación, y se propone desarrollar un sistema para la gestión del proceso de emisión de certificación de estudios terminados, que cumpla con las necesidades del personal de la institución para dar respuesta a las solicitudes de los egresados. Para dar solución a la propuesta se usa la metodología de desarrollo de software Proceso Unificado Ágil en su variación para la Universidad de las Ciencias Informáticas (AUP-UCI). Durante la investigación se realiza un estudio del estado del arte referente a los sistemas automatizados vinculados al objeto de estudio. En el desarrollo del sistema se utiliza como gestor de base de datos a MongoDB, así como el lenguaje de programación JavaScript con los frameworks React y NestJS. Con el fin de satisfacer las necesidades del cliente, se obtuvo como resultado, un sistema que apoya la gestión del proceso de emisión de certificación de estudios terminados.

Palabras claves: certificación de estudios terminados, gestión académica, sistemas automatizados.

ABSTRACT

Over the years, the Ministry of Education (MINED) has developed a program for the computerization of academic management, with the aim of humanizing the work in the different educational centers. In this sense, we can say that the process of issuing certification of completed studies of the Ministry of Education has low levels of computerization, which does not contribute to the process of standardization and security of these documents. Therefore, this research focuses on the process of academic management in the municipal directorates of education, and proposes to develop a system for the management of the process of issuance of certification of completed studies, which meets the needs of the institution's staff to respond to the requests of graduates. To provide a solution to the proposal, the Agile Unified Process software development methodology is used in its variation for the University of Informatics Sciences (AUP-UCI). During the research, a state-of-the-art study is carried out regarding the automated systems linked to the object of study. In the development of the system, MongoDB is used as a database manager, as well as the JavaScript programming language with the React and NestJS frameworks. In order to meet the client's needs, a system was obtained that supports the management of the process of issuing certification of completed studies.

Keywords: certification of completed studies, academic management, automated systems.

TABLA DE CONTENIDOS

INTRODUCCIÓN 1
CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA 6
Introducción 6
1.1 Conceptos fundamentales asociados a la investigación 6
1.2 Sistemas homólogos 7
1.3 Metodologías de desarrollo de software 10
1.3.1 Extreme Programming (XP): 11
1.3.2 SCRUM 11
1.3.3 AUP-UCI 12
1.4 Metodología de desarrollo a utilizar 13
1.5 Tecnologías y herramientas utilizadas 13
Conclusiones parciales 16
CAPÍTULO II: DISEÑO DE LA PROPUESTA DE SOLUCIÓN 17
Introducción 17
2.1 Descripción de la propuesta de solución 17
2.2 Modelo conceptual 17
2.2.1 Descripción de los elementos del modelo conceptual 17
2.3 Requisitos funcionales 19
2.4 Requisitos no funcionales 20
2.5 Definición de los casos de uso 22
2.5.1 Definición de los actores 23
2.5.2 Diagrama de caso de uso del sistema 23
2.5.3 Descripción de caso de uso del sistema 24
2.6 Técnicas de trazabilidad de requisitos 28
2.7 Modelo de datos 29
2.8 Patrones arquitectónicos 29
2.8.1 Patrones de diseño 31
2.8.2 Patrones GRASP 33
Conclusiones parciales 35
CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN 36
Introducción 36
3.1 Diagrama de componentes 36
3.2 Estándares de codificación 37
3.3 Implementación del sistema 38
3.4 Diagrama de despliegue 39
3.5 Pruebas de software 40
3.5.1 Pruebas internas 40
3.5.1.1 Prueba de caja blanca 40
3.5.1.2 Pruebas de caja negra 43

3.5.1.3 Pruebas unitarias 44
3.5.2 Pruebas de aceptación 46
Conclusiones parciales 48
CONCLUSIONES 49
RECOMENDACIONES 50
REFERENCIAS BIBLIOGRÁFICAS 51
40. ANEXOS 54
Anexo 1: Casos de uso del sistema 54
Anexo 2: Acta de aceptación del cliente 69
Anexo 3: Aval del cliente 70

ÍNDICE DE TABLAS

Tabla 1. Requisitos funcionales. Elaboración propia. 19
Tabla 2. Descripción del caso de uso Gestionar Usuario. Elaboración propia. 24
Tabla 3. Descripción del camino #1. Elaboración propia. 43
Tabla 4. Descripción del camino #2. Elaboración propia. 43
Tabla 5. Descripción del escenario de Prueba del Caso de Uso "Gestionar Usuario". Elaboración Propia. ..44
Tabla 6. Descripción de caso de uso del sistema Autenticar Usuario. Elaboración propia. 54
Tabla 7. Descripción de caso de uso del sistema Gestionar Estudiante. Elaboración propia. 55
Tabla 8. Descripción de caso de uso del sistema Gestionar Asignatura. Elaboración propia. 61
Tabla 9. Descripción de caso de uso del sistema Gestionar Resolución. Elaboración propia. 63

ÍNDICE DE FIGURAS

Figura 1. Modelo conceptual. Elaboración propia. 18
Figura 2. Diagrama de caso de uso del sistema. Elaboración propia. 24
Figura 3. Matriz de trazabilidad Requisito Funcional-Caso de Uso. Elaboración propia. 29
Figura 4. Modelo de Datos. Elaboración propia. 29
Figura 5. Arquitectura MVC. Elaboración propia. 31
Figura 6. Diagrama de componentes NestJs API. Elaboración propia. 36
Figura 7. Diagrama de componentes ReactApp. Elaboración propia. 37
Figura 8. Diagrama de despliegue. Elaboración propia. 40
Figura 9. Grafo resultante. Elaboración propia. 42
Figura 10. Resultados de las pruebas de aceptación. Elaboración propia. 47

INTRODUCCIÓN

En los últimos años en Cuba se ha observado un desarrollo vertiginoso en cuanto al ámbito tecnológico se refiere, desarrollo que influye directamente en el mejoramiento de las condiciones de vida del ciudadano cubano. En este ámbito una de las principales ramas de desarrollo es el “Proceso de Informatización de la Sociedad cubana”, a través del cual se han alcanzado grandes logros, con el fin de humanizar el trabajo en los diferentes centros laborales del país, así como viabilizar y aligerar la carga burocrática en los trámites que se realizan para cualquier proceso legal.

En el marco de este trascendental esfuerzo, sin precedentes en el país, es notable observar el nacimiento de disímiles portales, aplicaciones y sistemas informáticos, entre los cuales se incluyen, la aplicación de la banca móvil Transfermóvil, la herramienta multiplataforma XAUCE AKADEMOS, así como los diferentes portales web que van dirigidos a los gobiernos de las diferentes provincias, con el fin de mejorar su gestión.

En este sentido el Ministerio de Educación (MINED), por su parte, acorde con su misión, la cual entre otros puntos se basa en: “Diseñar, orientar y controlar la política relacionada con la formación vocacional y la orientación profesional de las nuevas generaciones, en correspondencia con las necesidades del desarrollo socioeconómico del país” (Informatización - Ministerio de Educación de la República de Cuba, 2023), ha desarrollado un programa de Informatización de la gestión educativa, que posee las siguientes funciones:

1. Desarrollar y desplegar recursos informáticos para la Red informática del MINED (RIMED).

2. Seguimiento a los indicadores de conectividad y a los servicios telemáticos del Sistema Nacional de Educación.

3. Gestión y monitoreo de la Red, dirigiendo metodológicamente la política de Seguridad Informática en el Sistema Nacional de Educación.

4. Trazar la política de informatización del Sistema Nacional de Educación (Informatización - Ministerio de Educación de la República de Cuba, 2023).

Con este fin se ha dado grandes pasos en su proceso de informatización, entre los cuales se destaca, la creación del Canal Educativo y de la Empresa cubana de Informática y Medios Audiovisuales para

la Educación (Cinesoft), así como el lanzamiento del portal web Cubaeduca, experiencias que van dirigidas a complementar la formación de los niños y jóvenes.

El Ministerio de Educación (MINED) en Cuba es el organismo encargado de dirigir, proponer, ejecutar y controlar la política del Estado y el gobierno en cuanto a la educación. A lo largo del país están distribuidas en cada una de las provincias y municipios, las diferentes direcciones provinciales y municipales de educación. Para cumplir la misión y las funciones asignadas a la dirección municipal de educación, esta tiene aprobada una plantilla de cargos, integrada por un subdirector general, dos cargos de subdirectores y un total de hasta 11 departamentos, entre los que se encuentra el departamento de títulos y diplomas, el cual tiene entre sus funciones custodiar toda la documentación relacionada con títulos, diplomas y certificados.

Según las normas y procedimientos del MINED cuando un centro escolar cierra permanentemente, toda la documentación se traslada al departamento municipal de títulos y diplomas. Además, debe cumplir con la atención a clientes que asisten a dicha entidad para trámites, verificaciones y legalizaciones debido a pérdida de documentos originales que fueron entregados en el momento de su egreso.

En el municipio Alquízar de la provincia Artemisa, existen veintiocho centros educacionales que han sufrido este proceso de cierre permanente, lo que ha provocado que se reubique la documentación de estos centros educativos en este departamento.

Cuando un egresado en uno de estos centros necesita su certificación de estudios terminados puede dirigirse a dicho departamento para hacer la solicitud por escrito de la misma. Para dar respuesta a esta solicitud el asesor del departamento se dispone realizar una búsqueda manual en el archivo pasivo del departamento, que posee una gran cantidad de documentación, esto dificulta la búsqueda que además demora en gran medida la respuesta a dicha solicitud. Además, esta documentación se encuentra en riesgo de deterioro o desaparición debido al paso de los años y las malas condiciones en que se encuentra almacenada.

Luego de obtener toda la información deseada de los diferentes registros, el asesor deberá transcribir de forma manual toda esta información en el modelo de certificación de estudios terminados que luego será acuñado y firmado por las partes pertinentes. Esta etapa del proceso también esta acarreada por posibles errores y constantes demoras.

Las razones previamente expuestas, avalan la existencia de actividades a informatizar en el proceso de emisión de certificación de estudios terminados en el departamento de títulos y diplomas, lo que se

evidencia con demoras innecesarias y errores debido al alto nivel de trabajo manual, además se puede observar la necesidad de humanizar en mayor medida el trabajo en esta área. En conclusión, es fácil apreciar que el proceso de emisión de certificación de estudios terminados del MINED posee bajos niveles de informatización, lo cual no contribuye al proceso de estandarización y seguridad de dichos documentos. Teniendo en cuanta lo anteriormente planteado, el **problema científico** radica en: ¿Cómo contribuir con la informatización del proceso para la gestión de certificación de estudios terminados del MINED?

Como **objeto de estudio** de esta investigación se define: El proceso de gestión académica en las direcciones municipales del MINED, cuyo **campo de acción** está enmarcado en el proceso de emisión de certificación de estudios terminados del departamento de títulos y diplomas.

Para dar solución a la problemática planteada, la presente investigación tiene como **Objetivo general:** Desarrollar un sistema para la gestión del proceso de emisión de certificación de estudios terminados, que cumpla con las necesidades del personal de la institución.

Para dar solución al problema planteado se proponen las siguientes **tareas de investigación**:

1. Elaboración de los referentes teórico-metodológicos de la investigación, asociada a las deficiencias que presentan los servicios de trámites, verificaciones y legalizaciones en el departamento de títulos y diplomas.

2. Selección de la metodología de software, herramientas informáticas y tecnologías para su posterior utilización en el desarrollo del sistema.

3. Especificación de los requisitos funcionales y no funcionales, descripción de casos de uso del sistema y definición de los patrones de diseño y arquitectura utilizados en la implementación del sistema.

4. Implementación de un sistema donde se prioricen las funciones más importantes dentro del proceso de gestión de certificación de estudios terminados.

5. Validación de los resultados obtenidos con la implementación del nuevo sistema para la gestión de certificación de estudios terminados.

Métodos de la investigación

"Los métodos científicos de investigación son la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones" (Hernández Sampieri, 2010). Ellos se clasifican en teóricos y empíricos, en esta investigación se utilizaron:

Métodos teóricos

Analítico-Sintético: para descomponer el problema de investigación en elementos por separado, de esta forma se puede profundizar en el estudio de cada elemento, para luego sintetizarlos en la solución de la propuesta.

Análisis Histórico-lógico: Caracteriza al objeto en sus aspectos más externos, a través de la evolución y desarrollo histórico del mismo. Reproduce en el plano teórico la esencia del objeto de estudio. El empleo de este método posibilita realizar el estudio del estado del arte, investigar sobre los sistemas de gestión académica obteniendo así, sus características y elementos fundamentales, para un mejor entendimiento sobre el desarrollo del módulo que se quiere actualizar.

Métodos empíricos

Observación científica: Se utilizó para determinar el problema por el cual se realiza esta investigación y para comprender el proceso que se está investigando.

Para lograr una mejor organización y lectura el trabajo de diploma se estructuró de la siguiente manera: Resumen, Introducción, Capítulo I, Capítulo II, Capítulo III, Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos, donde se abarca todo lo relacionado con la investigación realizada.

- **Capítulo I Fundamentación teórica:** Se definen los conceptos más relevantes relacionados con los sistemas de gestión académica y se abordan los conceptos más importantes para la investigación. Por otro lado, se realiza una descripción de las tecnologías a emplear como lenguaje de programación y se analizan las principales herramientas para el diseño, modelado e implementación de la solución.
- **Capítulo II Diseño de la propuesta de solución:** Se especifican los requisitos funcionales que debe cumplir la propuesta, así como la descripción de estos. También se muestran los

modelos y diagramas diseñados, la explicación de la arquitectura del sistema y los patrones de diseño utilizados.

- **Capítulo III Implementación y validación de la propuesta de solución:** Este capítulo abarca todo lo relacionado con la implementación de la solución y las pruebas realizadas a la propuesta de solución, en función de verificar la calidad y efectividad de acuerdo con las necesidades del cliente.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se detallan los elementos teóricos que apoyan la investigación, el desarrollo del tema, así como el planteamiento de los conceptos y aspectos más importantes. El autor realiza un estudio detallado de los sistemas de gestión académica existentes a nivel internacional y nacional. Describe la metodología, tecnologías y herramientas de trabajo a utilizar en el desarrollo de la solución propuesta, sentándose las bases teóricas para sustentar la investigación.

1.1 Conceptos fundamentales asociados a la investigación

Gestión académica

La gestión académica es vital para el logro de la misión de una institución, así como sus metas de formación, por eso los profesores y los estudiantes son actores centrales en el diseño, implantación y desarrollo de los procesos claves en la gestión académica, como son: la elaboración y revisión de planes de estudio; las normas y los procedimientos de evaluación del desempeño de los estudiantes; la planificación de las actividades para la generación y aplicación del conocimiento; la selección de estudiantes y administración de procesos relacionados con ellos y la selección de los medios instrumentales y bibliográficos apropiados para la enseñanza, el aprendizaje y la investigación.

Otro punto de vista apunta a que dicha gestión académica es un proceso con características fundamentales en el orden administrativo. Una institución contribuye al proceso de formación integral del estudiante, tiene relación directa con la enseñanza, el aprendizaje, y los procesos de admisión de estudiantes, matrículas, pensiones y servicios educativos integrales.

Algunos de los componentes de la Gestión Académica son:

- Procesos de preinscripción.
- Selección y admisión.
- Planes de estudio.
- Modelos pedagógicos.
- Ofertas de programas.
- Información académica y administrativa de los estudiantes y docentes (Mastrapa, 2014).

Sistemas de Gestión académica

Realizar los procesos docentes de forma manual puede resultar una tarea compleja y tediosa debido al gran volumen de información involucrado. Afortunadamente la incorporación de las tecnologías de la información ha dado la posibilidad de informatizar estos procesos mediante el empleo de sistemas automatizados para la gestión académica. Estos sistemas constituyen una herramienta de trabajo que permite hacer más eficientes los procesos docentes y optimizar sus recursos (Mastrapa, 2014).

Por todo lo planteado, se hace necesario identificar las experiencias efectivas de las instituciones que estén inmersas en el desarrollo de soluciones informáticas y así obtener una solución al problema existente, por lo que se requiere realizar un estudio sobre los sistemas similares de gestión de información académica.

1.2 Sistemas homólogos

A continuación, se detalla sobre la búsqueda y estudio de sistemas informáticos similares a la solución propuesta en la investigación, tanto en el ámbito internacional como nacional con el fin de utilizarlo como guía para comprender los procesos en los sistemas de gestión de la información académica en cada uno de ellos.

Ámbito Internacional:

Sistema de Censo Académico en Línea (**CENACAD**) para la automatización de la evaluación a los docentes, fue realizado por parte del personal de la Escuela Superior Politécnica del Litoral (ESPOL), (Guayaquil, Ecuador). Nace de la necesidad de brindar una solución informática sobre las evaluaciones docentes en la ESPOL. Cuenta con un módulo que les permite a los usuarios la obtención de diferentes tipos de reportes, a continuación, se muestran algunos de ellos:

- Listado de mejores docentes en base a los mejores promedios obtenidos en la ESPOL por el período de un año.
- Promedio de la(s) facultad(es) o instituto(s).
- Promedios obtenidos en cada área que conforman cada una de las encuestas realizadas a cada materia. Estos promedios deberán ser generales, de toda la facultad o unidad. Además, son presentadas de manera gráfica (CENACAD Censo Académico en Línea, 2023).

Sistema de Información de Gestión Académica (**SIGA**) de la Universidad Técnica Federico Santa María (Valparaíso, Chile), con el fin de propiciar un crecimiento continuo de la Universidad y agregar valor a

la Organización, su misión principal es contribuir al desarrollo de las tecnologías de la información con el fin de responder a las necesidades de la comunidad universitaria, a través del desarrollo de sistemas que permitan mejorar la calidad de los servicios que brinda dicha institución. Dentro de sus indicadores más importantes se encuentran:

- Rendimiento Académico Ponderado.
- Índice Académico.
- Avance Académico.
- Prioridad Académica.

Objetivos generales:

- Desarrollar los Sistemas Institucionales de la Universidad.
- Fortalecer la Tecnología de Información en la organización como agente de cambio.
- Dar acceso a la comunidad universitaria, a los recursos de información local y globalmente disponible.
- Aumentar la interacción de la comunidad para el enriquecimiento del conocimiento, la información, las habilidades y la comunicación.
- Generar nuevos servicios (Portal de Sistemas de Gestión Académica, 2023).

Sistema de Información Universitario (**SIU**) de la Universidad de San Juan (UNSJ), Argentina, es un sistema que colabora con las universidades en el procesamiento de datos para facilitar el análisis de la información. Es utilizado por la UNSJ principalmente para la administración de la gestión referida a aspirantes, alumnos, graduados, docentes, becas y estadísticas. Su objetivo es contribuir a mejorar la gestión de las instituciones, permitiéndoles contar con información segura, íntegra y disponible, optimizar sus recursos y lograr que el software sea aprovechado en toda su potencialidad.

Tiene como objetivos:

- Promover un sistema de información integral que involucre a las universidades y a la Secretaría de Políticas Universitarias.
- Desarrollar los acuerdos necesarios para lograr el intercambio de conocimientos y llegar a soluciones informáticas comunes a los organismos involucrados.

- Garantizar la disponibilidad, integridad, seguridad y calidad de la información.
- Colaborar en el análisis de la información producida y la consecuente definición de estrategias a fin de lograr un adecuado aprovechamiento de los recursos del Estado.
- Contribuir con la transparencia de la gestión y la rendición de cuentas, asegurando el derecho a la información veraz (FACSO, 2023).

Tras el estudio de las soluciones informáticas expuestas en el ámbito internacional, se arroja como resultado que, aunque todas llevan a cabo la informatización de los procesos de gestión de información en las distintas instituciones, estas carecen de funcionalidades importantes que sirvan de beneficio en el sistema a desarrollar, además, presentan como inconveniente que son softwares privativos, lo cual impide su modificación y utilización en Cuba.

Ámbito nacional:

Se realizó un estudio de los diferentes sistemas que realizaban la gestión académica en el país, destacándose que, en la Facultad de Informática de la Universidad de Matanzas Camilo Cienfuegos, se cuenta con un Sistema para la Gestión académica (GESTACAD), el cual brinda un grupo de funcionalidades que permiten mantener una información actualizada de estudiantes y profesores del centro, así como la obtención de reportes.

Se investigó además en la provincia de La Habana el software Sistema de Gestión de la Nueva Universidad (SIGENU), el cual tiene como fin, permitir la gestión de toda la información académica vinculada con la educación superior en Cuba. En correspondencia con su carácter nacional y la gran diversidad de sistemas de enseñanza superior con que cuenta la universidad cubana, este sistema ha sido concebido de manera tal que sea capaz de brindar gran seguridad e integridad de la información.

Otra de las soluciones estudiadas es el Sistema Docente Integral (SDI) creado en el Instituto Superior Politécnico “José Antonio Echeverría” (ISPJAE), el mismo surge debido a la importancia que tiene para este centro el control, planificación y resultado de los procesos docentes. El SDI permite registrar, procesar y crear mecanismos de recuperación de la información relacionada con el proceso docente, tanto de pregrado como de postgrado de forma automatizada. Esto constituye una ventaja ya que con su uso se pueden realizar estos procesos de forma eficiente permitiendo optimizar los recursos del centro de secretaría de las facultades.

El Sistema de Gestión Universitaria (UCI) (**Xauge Akademos**), es un sistema con el que cuenta la universidad para manejar la información en las distintas áreas, está dividido por varios módulos uno de ellos es Pregrado el cual se encarga de gestionar todo lo relacionado con los temas de la docencia de los estudiantes. Este módulo gestiona evaluaciones del estudiante, exámenes de premio, bonificaciones y promedio. El usuario es capaz de obtener la información mediante el filtrado de los diferentes componentes y esta puede ser exportada en formato PDF (Mastrapa, 2014).

A partir del estudio de los sistemas los sistemas homólogos en el ámbito nacional, se detecta que brindan una serie de funciones que serán tomadas como base para la propuesta de solución, incorporándoles además las que se describen anteriormente que no poseen.

Todas estas soluciones mencionadas anteriormente permitieron tener un conocimiento acerca del funcionamiento de la gestión académica en otras universidades del país. Cabe destacar que estos sistemas han permitido en las instituciones en las que se han utilizado, un mejor desarrollo del proceso de gestión académica de manera general, pero carecen de funciones específicas que son importantes para dar solución a la situación problemática planteada.

1.3 Metodologías de desarrollo de software

El desarrollo de software no es una tarea fácil. Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, establece rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros (Paez & Fontela, 2014), ejemplo de ellas son *Rational Unified Process* (RUP) y *Microsoft Solution Framework* (MSF).

Existen metodologías que proporcionan el apoyo necesario para crear un software adaptado a las necesidades del usuario, incluye la gestión de riesgos y problemas; sin embargo, en ocasiones tal gestión no se aplica (Fernández Sanz & Bernad Silva, 2014), arriesga la integridad del producto. Es de vital importancia establecer parámetros que guíen a los desarrolladores y arquitectos a no cometer o repetir acciones equivocadas, y como resultado mejorar los procesos para la creación del software.

Por otra parte, al considerar su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y en el modelado, reciben el apelativo de Metodologías Tradicionales (o también denominadas Metodologías Pesadas o Peso Pesado). Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación

de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso (Pressman, 2010). A continuación, se analizan algunas metodologías ágiles tras cumplir con la situación existente.

1.3.1 Extreme Programming (XP):

A fin de ilustrar un proceso ágil con más detalle, ofrece un panorama de la programación extrema (XP, por sus siglas en inglés), el enfoque más utilizado del desarrollo de software ágil. Aunque las primeras actividades con las ideas y los métodos asociados a XP ocurrieron al final de la década de 1980, el trabajo fundamental sobre la materia había sido escrito por Kent Beck. Una variante de esta metodología llamada XP industrial (IXP) se propuso en una época más reciente. IXP mejora la XP y tiene como objetivo el proceso ágil para ser usado específicamente en organizaciones grandes (Pressman, 2010).

Por ejemplo, en la XP muchas versiones actuales de un sistema pueden desarrollarse mediante diferentes programadores, integrarse y ponerse a prueba en un solo día. En la programación extrema, los requerimientos se expresan como escenarios (llamados historias de usuario), que se implementan directamente como una serie de tareas. Los programadores trabajan en pares y antes de escribir el código desarrollan pruebas para cada tarea. Todas las pruebas deben ejecutarse con éxito una vez que el nuevo código se integre en el sistema (Sommerville, 2016).

1.3.2 SCRUM

Es un método de desarrollo ágil de software concebido por *Jeff Sutherland* y su equipo de desarrollo a principios de la década de 1990. En años recientes, *Schwaber* y *Beedle* han desarrollado más los métodos *Scrum*. Sus principios son congruentes con el manifiesto ágil y se utilizan para guiar actividades de desarrollo dentro de un proceso de análisis que incorpora las siguientes actividades estructurales: requerimientos, análisis, diseño, evolución y entrega. Dentro de cada actividad estructural, las tareas del trabajo ocurren con un patrón del proceso llamado *sprint*. El trabajo realizado dentro de un *sprint* se adapta al problema en cuestión y se define y con frecuencia se modifica en tiempo real por parte del equipo *Scrum*. Acentúa el uso de un conjunto de patrones de proceso del software que han demostrado ser eficaces para proyectos con plazos de entrega muy apretados, requerimientos cambiantes y negocios críticos (Pressman, 2010).

1.3.3 AUP-UCI

AUP es una versión simplificada del Proceso Unificado de Software (RUP). Describe de una forma fácil de entender una aproximación al desarrollo de software empresarial mediante técnicas ágiles, se mantiene fiel a las técnicas y conceptos definidos por RUP. Las técnicas ágiles empleadas por AUP incluyen un desarrollo dirigido por pruebas, modelado ágil, gestión de cambios ágil y refactorización de base de datos para mejorar la productividad. La metodología AUP-UCI responde a una variación que se realiza a la metodología ágil AUP. Esta metodología de tipo ágil divide el ciclo de vida de la producción del software en tres fases (Sánchez, 2019):

Inicio: fase donde se realiza un estudio inicial de la organización cliente que permite obtener una información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: fase donde se ejecutan las actividades requeridas para desarrollar el software, incluye el ajuste de los planes del proyecto, considera 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: fase donde se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP-UCI propone para el ciclo de vida de los proyectos siete disciplinas, donde los flujos de trabajo son los siguientes: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Además, está compuesta por cuatro escenarios (Sánchez, 2019):

Escenario No 1: proyectos que modelen el negocio con casos de uso del negocio solo pueden modelar el sistema con casos de uso del sistema.

Escenario No 2: proyectos que modelen el negocio con modelo conceptual solo pueden modelar el sistema con casos de uso del sistema.

Escenario No 3: proyectos que modelen el negocio con descripción de procesos de negocio solo pueden modelar el sistema con descripción de requisitos por procesos.

Escenario No 4: proyectos que no modelen negocio solo pueden modelar el sistema con historias de usuario.

1.4 Metodología de desarrollo a utilizar

Se selecciona para esta investigación la metodología AUP-UCI en su escenario número dos porque este 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 modelan exclusivamente los conceptos fundamentales del negocio. Este escenario es ideal para proyectos donde el objetivo primario es la gestión y presentación de la información.

1.5 Tecnologías y herramientas utilizadas

Para cumplir el objetivo de esta investigación es imprescindible definir algunas tecnologías y herramientas que posibiliten el desarrollo exitoso del sistema.

HTML

HTML (Lenguaje de Marcas de Hipertexto, del inglés *HyperText Markup Language*) es el componente más básico de la *web*. Define el significado y la estructura del contenido *web*. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página *web* (CSS) o la funcionalidad/comportamiento (JavaScript)(HTML, 2023).

HTML5

La última versión estable de HTML, HTML5 convierte a HTML de un simple formato de marcado para estructurar documentos en una plataforma completa de desarrollo de aplicaciones. Entre otras características, HTML5 incluye nuevos elementos y API de JavaScript para mejorar el almacenamiento, la multimedia y el acceso al hardware (HTML5 - Glosario de MDN Web Docs, 2022).

CSS3

Hojas de Estilo en Cascada (del inglés *Cascading Style Sheets*) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (incluyendo varios lenguajes basados en XML como SVG, MathML o XHTML). CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.

CSS es uno de los lenguajes base de la *Open Web* y posee una especificación estandarizada por parte del W3C. Desde CSS3, el alcance de las especificaciones se incrementó de forma significativa y el progreso de los diferentes módulos de CSS comenzó a mostrar varias diferencias, lo que hizo más efectivo desarrollar y publicar recomendaciones separadas por módulos. En vez de versionar las

especificaciones de CSS, la W3C actualmente realiza una captura de las ultimas especificaciones estables de CSS (CSS | MDN, 2023).

JavaScript (ECMAScript 2020)

JavaScript es un poderoso lenguaje de programación construido para el navegador Netscape en 1995. Además, es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo *web*, pensado para agregar potencial de interacción y dinamismo a las páginas *web*. Es compatible con todos los navegadores modernos y se ejecuta del lado del cliente, lo que significa que se ejecuta en el navegador *web* del usuario final. Además de su uso en el desarrollo *web*, JavaScript también se utiliza en aplicaciones de servidor (con tecnologías como Node.js) y en el desarrollo de aplicaciones móviles y de escritorio (Coppola, 2023).

JavaScript proporciona una amplia gama de funcionalidades, como manipulación del DOM (*Document Object Model*) para interactuar con elementos de una página *web*, manejo de eventos, comunicación con servidores a través de AJAX, creación de animaciones, validación de formularios y mucho más. Es un lenguaje flexible y dinámico que permite a los desarrolladores crear experiencias interactivas y ricas en contenido (Coppola, 2023).

React (v18.2.0)

React es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes.

Es importante señalar que React no es un *framework* de JavaScript. Esto porque sólo es responsable de renderizar los componentes de la capa de vista de una aplicación. React es una alternativa a *frameworks* como Angular y Vue, que permiten crear funciones complejas (Qué es React: definición, características y funcionamiento, 2023).

NestJS (v10.1.11)

NestJS es un *framework* progresivo de Node.js para la creación de aplicaciones eficientes, confiables y escalables del lado del servidor, el cual está construido y es completamente compatible con TypeScript (no obstante aún permite la codificación en JS puro), combinando elementos de la programación orientada a objetos (POO en español; OOP, según sus siglas en inglés, *Object-oriented programming*), programación funcional (FP, según sus siglas en inglés, *functional programming*) y

programación reactiva funcional (FRP, según sus siglas en inglés, *functional reactive programming*) (NestJS: qué es y por qué empezar a usarlo – GFourmis, 2023).

TypeScript (v4.7.4)

TypeScript es un lenguaje de programación moderno que permite crear aplicaciones *web* robustas en JavaScript utilizando herramientas de POO (Programación Orientada a Objetos). TypeScript no requiere de ningún tipo de *plugin*, puesto que lo que hace es generar código JavaScript que se ejecuta en cualquier navegador, plataforma o sistema operativo. Es un superset de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases (¿Que Es TypeScript?, 2023).

Node.js (v18.17.1)

Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.

Node.js fue creado por los desarrolladores originales de JavaScript. Lo transformaron de algo que solo podía ejecutarse en el navegador en algo que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara. Gracias a Node.js se puede ir un paso más allá en la programación con JavaScript no solo creando sitios *web* interactivos, sino teniendo la capacidad de hacer cosas que otros lenguajes de secuencia de comandos como Python pueden crear (Qué es NodeJS y para qué sirve | OpenWebinars, 2023).

MongoDB (1.39.1)

MongoDB es un sistema de gestión de bases de datos (DBMS, por sus siglas en inglés) no relacionales y de código abierto, que utiliza documentos flexibles en lugar de tablas y filas para procesar y almacenar varias formas de datos. Al ser una solución de base de datos NoSQL, MongoDB no requiere un sistema de gestión de bases de datos relacionales (RDBMS), por lo que proporciona un modelo de almacenamiento de datos elástico que permite a los usuarios almacenar y consultar fácilmente tipos de datos multivariados. Esto no solo simplifica la gestión de la base de datos para los desarrolladores, sino que también crea un entorno altamente escalable para aplicaciones y servicios multiplataforma.

Los documentos o colecciones de documentos de MongoDB son las unidades básicas de datos. Estos documentos, con formato JSON (Java Script Object Notation) binario, pueden almacenar varios tipos de datos y distribuirse en varios sistemas. El diseño de esquema dinámico de MongoDB brinda a los

usuarios una flexibilidad sin igual para crear registros de datos, consultar colecciones de documentos a través de la agregación de MongoDB y analizar grandes cantidades de información (¿Qué es MongoDB?, 2023).

VS Code (v1.76.2)

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación (Qué es Visual Studio Code y qué ventajas ofrece | OpenWebinars, 2023).

Lenguaje de Modelado Unificado (UML)

Es un lenguaje estándar para escribir diseños de software. Puede usarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo (Pressman, 2010). En la investigación se empleó el UML para la construcción del modelo conceptual y se trabajó con una herramienta CASE. La ingeniería de software asistida por computadora (CASE, por sus siglas en inglés) consiste en diversas aplicaciones que pueden ayudar en el ciclo de vida de desarrollo de software.

Visual Paradigm (v15.1)

Es una herramienta de diseño y administración poderosa, multiplataforma y fácil de usar para los sistemas de tecnología de información. Ofrece a los desarrolladores de software una plataforma de desarrollo innovadora para crear aplicaciones de calidad de manera más rápida. Facilita una excelente interoperabilidad la mayoría de los entornos de desarrollo integrado (Sommerville, 2016).

Conclusiones parciales

Luego de desarrollar el capítulo se ha podido arribar a las siguientes conclusiones parciales:

- El análisis de los principales conceptos permitió una mejor comprensión de la presente investigación.
- La investigación de sistemas homólogos permitió identificar las características principales de los sistemas de gestión de información académica tanto internacionales como nacionales.
- El estudio de las tecnologías, herramientas, lenguajes y metodologías proporcionó los conocimientos necesarios para su utilización en el desarrollo de la solución propuesta.

CAPÍTULO II: DISEÑO DE LA PROPUESTA DE SOLUCIÓN

Introducción

El presente capítulo tiene como objetivo describir la propuesta de solución, precisar las características con que contarán las aplicaciones e implementarlas. El autor evidencia las disciplinas Modelado de negocio, Requisitos y Análisis, Diseño e Implementación que propone la metodología AUP-UCI para el ciclo de vida del proyecto. Realiza una representación del modelo conceptual, define los requisitos funcionales y no funcionales del módulo. Presenta los diagramas de casos de uso; así como su descripción para una mayor comprensión.

2.1 Descripción de la propuesta de solución

La solución propuesta tiene como objetivo desarrollar un sistema para la gestión del proceso de emisión de certificación de estudios terminados. Dentro del mismo los usuarios pueden desempeñar dos roles diferentes: administrador y asesor. Inicialmente el acceso al sistema se realiza mediante un usuario con rol administrador que contiene el mismo en su configuración predeterminada.

El usuario con rol administrador es el encargado de la gestión completa del sitio y es el usuario de mayor privilegio en el sistema, además es el encargado de gestionar los usuarios y los roles que estos desempeñan. Una vez creados los usuarios se le asignan los permisos para diferenciar los recursos a los que van a tener acceso.

El acceso con rol asesor le permite al usuario gestionar todo lo referente a las asignaturas, resoluciones y registros de cursos de estudiantes. De esta manera se conforma una base de datos con toda la información referente a los egresados, que el sistema utiliza para la emisión de la certificación de estudios terminados de manera automatizada al introducir los datos del solicitante.

2.2 Modelo conceptual

El modelo conceptual es una representación visual de los conceptos u objetos que se manejan en el dominio del sistema. Los objetos o conceptos incluidos en el modelo conceptual no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de construcción del sistema (Báster, 2011).

2.2.1 Descripción de los elementos del modelo conceptual

- **Asesor:** encargado de gestionar toda la información en el departamento de títulos y diplomas.

- **Resolución:** legislación por la que se rige el plan de estudio.
- **Solicitud:** documento que contiene los datos del egresado que realiza la solicitud.
- **Plan de Estudio:** se compone de una serie de objetivos, metodologías y criterios de evaluación, que permiten establecer los conocimientos y habilidades que se deben adquirir durante el proceso educativo y asegurarse de que los alumnos cumplan con los requisitos para la obtención de la titulación.
- **Registro de Estudiante:** documentos y otros materiales que contienen información directamente relacionada con el estudiante.
- **Egresado:** estudiante que después de haber acabado su preparación académica, abandona el centro de estudios donde fue preparado.
- **Certificación de estudios terminados:** documento que desglosa las asignaturas a lo largo de su programa, con sus respectivas evaluaciones finales, promedios por periodo, promedio acumulado e intensidad horaria total.
- **Registro de matrícula y graduados:** registra toda la matrícula de graduados en un determinado centro escolar.
- **Relación nominal de graduados:** contiene la información de los cursos de cada estudiante que se graduó en un determinado centro escolar.

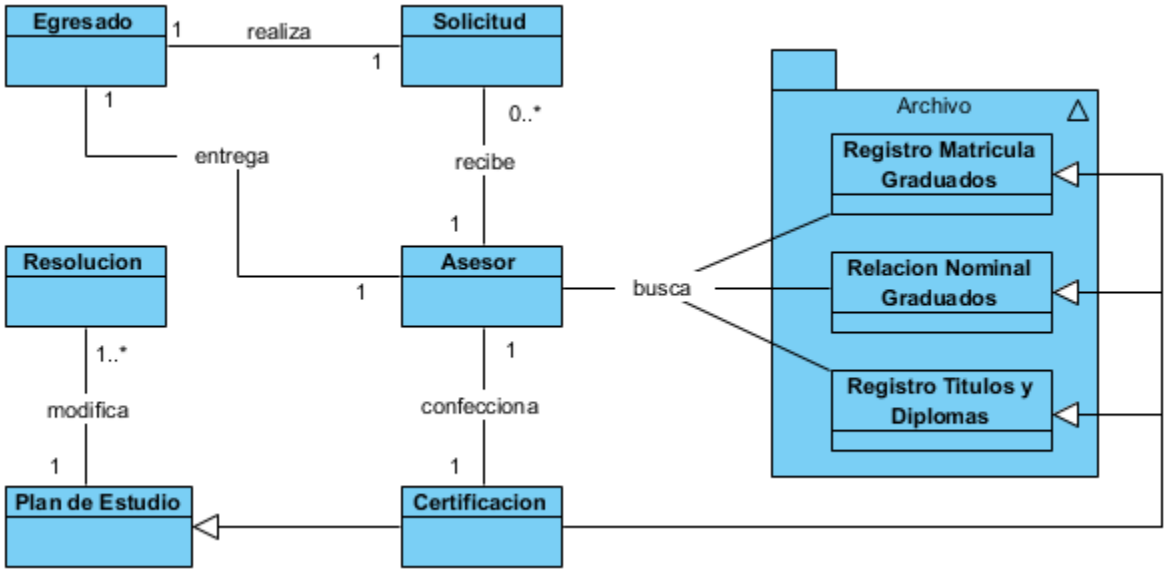


Figura 1. Modelo conceptual. Elaboración propia.

2.3 Requisitos funcionales

Los requisitos funcionales definen los componentes de un sistema de software, describen un conjunto de entradas, comportamientos, salidas y funcionalidades específicas que un sistema debe cumplir. Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas y las salidas que debe devolver según situaciones particulares (Sommerville, 2016). A continuación, se plasman los requisitos funcionales identificados para el desarrollo de la solución propuesta:

Tabla 1. Requisitos funcionales. Elaboración propia.

No	Requisito	Prioridad	Descripción	Complejidad
RF1	Autenticar usuario	Alta	El sistema deberá verificar la identidad del usuario mediante su contraseña. Datos de entrada (usuario, contraseña).	Alta
RF2	Crear usuario	Alta	El sistema debe permitir al administrador crear un nuevo usuario. Datos de entrada (usuario, contraseña, email).	Media
RF3	Modificar usuario	Alta	El sistema debe permitir al administrador modificar usuario. Datos de entrada (contraseña, role).	Media
RF4	Eliminar usuario	Alta	El sistema debe permitir al administrador eliminar usuario.	Media
RF5	Crear estudiante	Alta	El sistema debe permitir al administrador y al asesor Crear estudiante. Datos de entrada (nombre, apellidos, CI, numero_expediente, año_final).	Media
RF6	Modificar estudiante	Alta	El sistema debe permitir al administrador y al asesor modificar estudiante. Datos de entrada (nombre, apellidos, CI, numero_expediente, año_final).	Media
RF7	Eliminar estudiante	Alta	El sistema debe permitir al administrador y al asesor eliminar estudiante.	Media
RF8	Listar estudiante	Alta	El sistema debe mostrar una lista ordenada de estudiantes.	Media

RF9	Buscar estudiante	Alta	El sistema debe permitir al administrador y al asesor buscar estudiante. Datos de entrada (nombre, CI, numero_expediente).	Alta
RF10	Crear asignatura	Alta	El sistema debe permitir al administrador y al asesor crear asignatura. Datos de entrada (nombre, descripción).	Media
RF11	Eliminar asignatura	Alta	El sistema debe permitir al administrador y al asesor eliminar asignatura.	Media
RF12	Crear resolución	Alta	El sistema debe permitir al administrador y al asesor crear resolución. Datos de entrada (nombre, descripción, año_inicio, año_final, grados, enseñanza).	Media
RF13	Modificar resolución	Alta	El sistema debe permitir al administrador y al asesor modificar resolución. Datos de entrada (nombre, descripción, año_inicio, año_final, grados, enseñanza).	Baja
RF14	Eliminar resolución	Alta	El sistema debe permitir al administrador y al asesor eliminar resolución.	Media
RF15	Listar resolución	Alta	El sistema debe permitir al administrador y al asesor listar resolución.	Media
RF16	Buscar resolución	Alta	El sistema debe permitir al administrador y al asesor buscar resolución. Datos de entrada (nombre).	Alta
RF17	Exportar certificación	Alta	El sistema debe permitir al asesor exportar certificación.	Alta

2.4 Requisitos no funcionales

Los Requisitos no Funcionales (RNF) se refieren a funciones específicas que proporciona el sistema. Son restricciones de los servicios o funciones ofrecidas por el sistema (fiabilidad, tiempo de respuestas, capacidad de almacenamiento). Generalmente se aplican al sistema en su totalidad. Surgen de las necesidades del usuario (restricciones de presupuesto, políticas de la organización, necesidad de interoperabilidad). En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el

sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (Pressman, 2010).

A continuación, se describen los requisitos no funcionales que debe cumplir el sistema para la gestión de la certificación de estudios terminados en la dirección municipal de educación de Alquízar:

Usabilidad

RNF1: El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo para usuarios sin experiencia y el acceso de manera rápida y efectiva a la información buscada.

RNF2: El sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a mayor comprensión por parte del cliente de la herramienta de trabajo.

RNF3: El sistema debe presentar un menú lateral que permita el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

Rendimiento

RNF4: Los tiempos de respuesta deben ser generalmente cortos al igual que la velocidad de procesamiento de la información, para lograr una respuesta rápida debe garantizarse una buena velocidad de conexión, específicamente una Red de Área Local (LAN).

Seguridad

RNF5: Restringir el acceso a la información, permitiendo su gestión solo al personal autorizado para ello, garantizando así su debido uso.

RNF6: Identificar el personal que se autoriza y se le concede permisos para determinadas tareas (procesos de identificación, autorización y asignación de roles).

RNF7: Mantener la integridad de la información, cumpliendo correctamente con los requisitos de restricción de acceso y asignación de roles.

Requisitos de software

RNF8: Para el uso del sistema la PC cliente debe contar con un navegador web. Se recomienda utilizar Mozilla Firefox v111.0.1 o Google Chrome v111.0.5563.65 o superior.

RNF9: El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos:

- Sistema Operativo (SO): Windows server 2012 o Linux Debian 9.
- Usuario con privilegios de administrador al SO.

- MongoDB a partir de su versión 4.4.
- Apache Tomcat a partir de su version 10.

Requisitos de hardware

RNF10: El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos de hardware:

Requisitos mínimos

- Puede ser servidor virtual o físico.
- Memoria RAM 4GB.
- Un CPU i3 de cuarta generación.
- 25GB de capacidad de disco duro.
- Conexión a la red mediante LAN con velocidad de 100 MB/s.

Requisitos recomendados

- Servidor Virtual.
- Memoria RAM 8GB.
- Un CPU i5 de cuarta generación.
- 50GB de capacidad de disco duro.
- Conexión a la red mediante LAN con velocidad de 1GB/s.

Restricciones de diseño e implementación

RNF11: El sistema debe cumplir con la arquitectura de información definida para el Sistema de Gestión académica.

RNF12: El sistema debe ser desarrollado con herramientas y tecnologías de código abierto.

2.5 Definición de los casos de uso

Un caso de uso es un artefacto que define una secuencia de acciones que da lugar a un resultado de valor observable. Los casos de uso proporcionan una estructura para expresar requisitos funcionales en el contexto de procesos empresariales y de sistema. Los casos de uso pueden representarse como un elemento gráfico en un diagrama y como una especificación de caso de uso en un documento textual (IBM Documentation, 2021).

- CU1: Autenticar usuario

- CU2: Gestionar usuario
- CU3: Gestionar estudiante
- CU4: Gestionar asignatura
- CU5: Gestionar resolución

2.5.1 Definición de los actores

Los actores son un tipo de clasificador UML similar a las clases y pueden participar en secuencias como instancias. Sin embargo, a los actores se les imponen las siguientes restricciones (IBM Documentation, 2021):

- No pueden agregar ni componer ningún elemento.
- Se generalizan sólo de otros actores.
- No se pueden convertir a clases, o viceversa.

El sistema cuenta con un solo tipo de actor:

Asesor: Este usuario es el encargado de ejecutar el proceso para la gestión de la certificación de estudios terminados.

2.5.2 Diagrama de caso de uso del sistema

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un Caso de Uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Un Caso de Uso es una unidad de trabajo significativo; por ejemplo, crear una solicitud y modificar una solicitud son todos Casos de Uso (Sparks & Pinciroli, 2003).

Cada Caso de Uso tiene una descripción que especifica la funcionalidad que se incorporará al sistema propuesto. Un Caso de Uso puede 'incluir' la funcionalidad de otro Caso de Uso o puede 'extender' otro Caso de Uso con su propio comportamiento. Los casos de uso típicamente se relacionan con 'actores'. Un actor es un humano o una máquina que interactúa con el sistema para realizar un trabajo significativo (Sparks & Pinciroli, 2003).

A continuación, se muestra el diagrama de caso de uso del sistema (Figura 2).

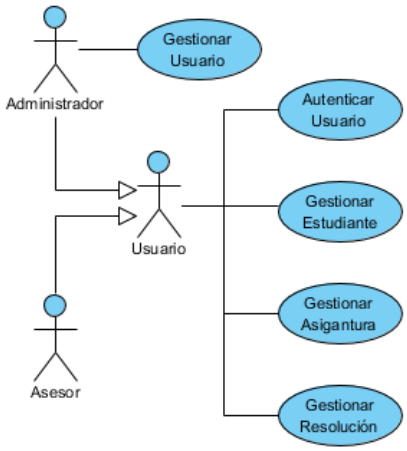


Figura 2. Diagrama de caso de uso del sistema. Elaboración propia.

2.5.3 Descripción de caso de uso del sistema

En la descripción de los casos de uso del sistema se muestra el caso de uso, los actores involucrados y un breve resumen de este. La tabla 2 muestra la descripción del caso de uso del sistema Gestionar Usuario. El resto de los casos de uso se pueden consultar en el Anexo 1 Casos de uso del sistema.

Objetivo: Define el objetivo principal del caso de uso.

Actores: Define los actores del caso de uso.

Resumen: Un breve resumen del caso de uso.

Complejidad: Define un nivel de complejidad para el desarrollo del caso de uso.

Prioridad: Nivel de prioridad del caso de uso definido por el cliente.

Precondiciones: Condiciones que debe cumplir antes de iniciar el caso de uso.

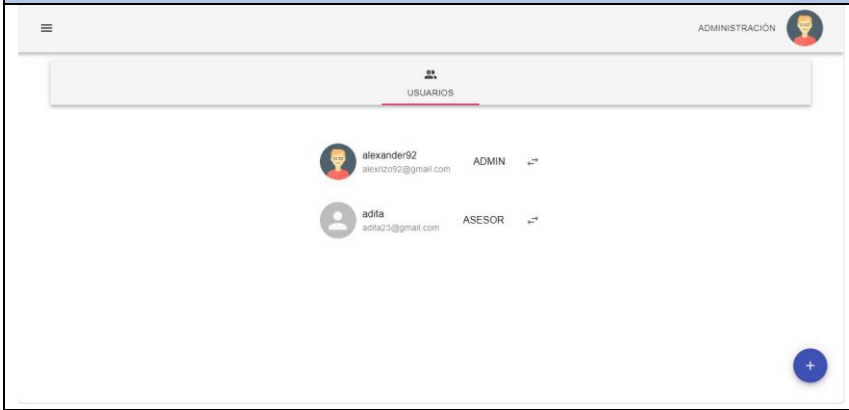
Postcondiciones: Condiciones que debe cumplir una vez termine el caso de uso.

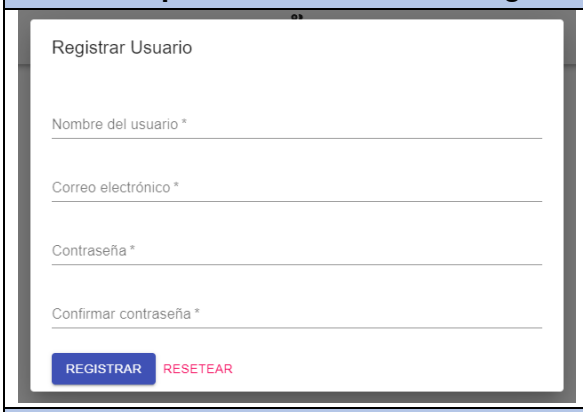
Flujo básico: Secuencia de actividades básicas entre el actor y el sistema.

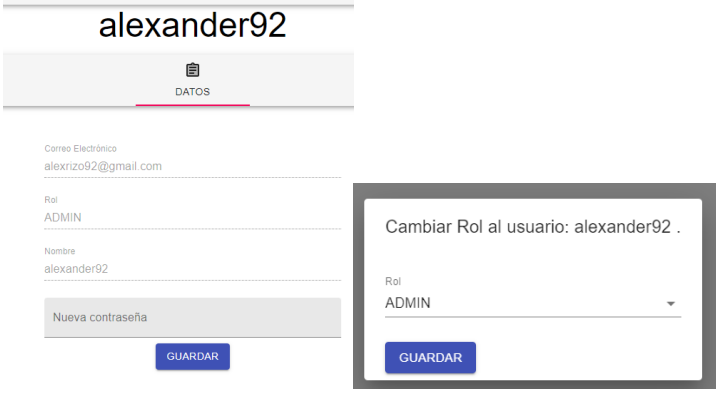
Flujo alterno: Secuencia de actividades específicas entre el actor y el sistema.

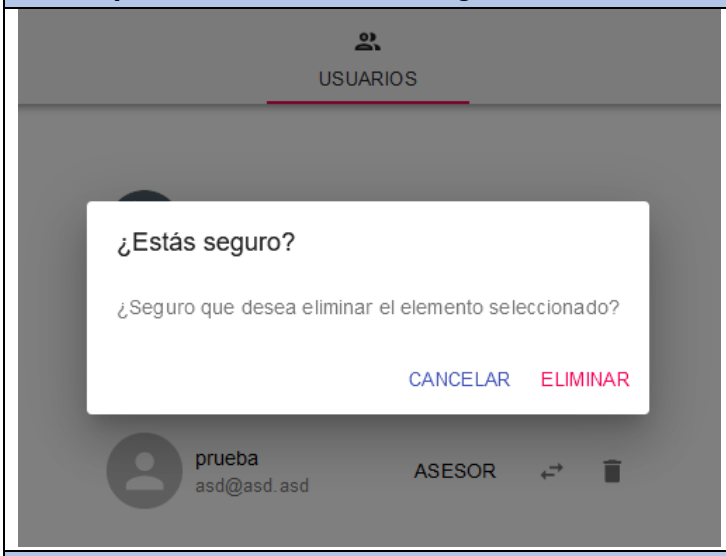
Tabla 2. Descripción del caso de uso Gestionar Usuario. Elaboración propia.

Objetivo	Permitir crear, modificar y eliminar datos acerca de un usuario.
Actores	Administrador

Resumen		El caso de uso se inicia cuando el Administrador decide registrar, modificar o eliminar datos acerca de un usuario.
Complejidad		Alta.
Prioridad		Alta.
Precondiciones		El administrador debe estar autenticado.
Postcondiciones		Se creó, modificó o se eliminó usuario(s).
Flujo de eventos		
Flujo básico "Gestionar usuarios"		
Actor		**Sistema**
1.	Selecciona de la página principal la opción "gestionar usuarios".	
2.		Muestra una pantalla con un listado de usuarios y permite Crear, Modificar o Eliminar usuario(s).
3.	Desea crear, modificar o eliminar usuario(s).	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: 1. Si decide crear un usuario, ir a la sección "Crear usuario". 2. Si decide modificar los datos de un usuario, ir a la sección "Modificar usuario". 3. Si decide eliminar un(os) usuario(s), ir a la sección "Eliminar usuario".
Prototipo elemental de interfaz gráfica		
		
Flujos alternos		
2a. Listado de usuarios vacío		
Actor		**Sistema**
1.		Carga una pantalla en blanco sin usuarios registrados.

Sección 1: “Crear usuario”		
Flujo básico Crear usuario		
Actor		**Sistema**
1.	Presiona el botón “Crear usuario”.	
2.		Muestra una ventana con los siguientes campos a introducir: • usuario, • contraseña, • email Y el botón “Registrar”.
3.	Introduce los datos y presiona el botón “Registrar”.	
4.		Verifica que todos los campos estén llenos.
5.		Verifica que los datos introducidos estén correctos.
6.		Verifica que este usuario no exista.
7.		Almacena los datos del usuario y muestra el mensaje “Usuario creado satisfactoriamente”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujos alternos		
4a. Campos vacíos		
Actor		**Sistema**
1.		Muestra el mensaje “Campos vacíos”.
5a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
6a. Usuario existente		
Actor		**Sistema**
1.		Muestra el mensaje “Usuario existente”.

Sección 2: “Modificar Usuario”		
Flujo básico Modificar Usuario		
	Actor	**Sistema**
1.	Selecciona uno de los usuarios mostrados, escoge la opción “Modificar usuario”.	
2.		Muestra una ventana con los datos a modificar: • contraseña, • role Y el botón “Guardar”.
3.	Realiza las actualizaciones deseadas y presiona el botón “Guardar”.	
4.		Verifica que los datos introducidos estén correctos.
5.		Actualiza la información incorporada al usuario y se emite un mensaje “Modificación satisfactoria”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujos Alternos		
4a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
Sección 3: “Eliminar Usuario”		
Flujo básico Eliminar Usuario		
Actor		**Sistema**
1.	Marca el(los) usuario(s) a eliminar y presiona la opción “Eliminar”.	
2.		Muestra el mensaje de confirmación “¿Está seguro que desea eliminar el(los) usuario(s) seleccionado(s)?”. Y los botones “Eliminar” y “Cancelar”.

3.	Presiona el botón “Aceptar”.	
4.		Elimina él(los) usuario(s) seleccionado(s) y emite el mensaje “Eliminación satisfactoria”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujo Alterno		
2a. Cancelar eliminación de usuario		
Actor		**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve al paso 2 del flujo básico “Gestionar usuario”.
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

2.6 Técnicas de trazabilidad de requisitos

La matriz de trazabilidad de requisitos actúa como un apoyo para mantener al día el estado de cada uno de los requisitos y especificaciones del proyecto. Este estado puede cambiar con el tiempo y en la matriz de trazabilidad se lleva el registro de estos cambios desde su inicio hasta su consecución. La matriz de trazabilidad mantiene dos versiones generales, un antes y un después, lo que es de gran utilidad (Blanco, 2023).

	Autenticar Usuario	Gestionar Asigantura	Gestionar Estudiante	Gestionar Resolución	Gestionar Usuario
Autenticar Usuario	✓				
Buscar Estudiante			✓		
Buscar Resolucion				✓	
Crear Asignatura		✓			
Crear Estudiante			✓		
Crear Resolucion				✓	
Crear Usuario					✓
Eliminar Asignatura		✓			
Eliminar Estudiante			✓		
Eliminar Resolucion				✓	
Eliminar Usuario					✓
Exporta Certificacion			✓		
Listar Estudiante			✓		
Listar Resolucion				✓	
Modificar Estudiante			✓		
Modificar Resolucion				✓	
Modificar Usuario					✓

Figura 3. Matriz de trazabilidad Requisito Funcional-Caso de Uso. Elaboración propia.

2.7 Modelo de datos

Un modelo de datos es una serie de conceptos que se utilizan para describir un conjunto de datos y las operaciones para manipularlos (Pressman, 2010). A continuación, se presenta el modelo de datos de la investigación:

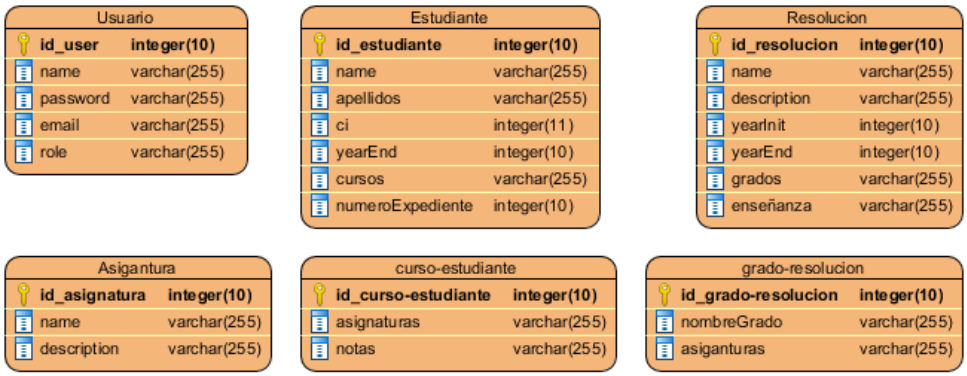


Figura 4. Modelo de Datos. Elaboración propia.

2.8 Patrones arquitectónicos

Un estilo arquitectónico es una transformación que se impone al diseño de todo el sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En el caso en el que ha de hacerse la reingeniería de una arquitectura ya existente, la imposición de un estilo arquitectónico dará

como resultado cambios fundamentales en la estructura del software, incluida la reasignación de las funciones de los componentes.

Un patrón arquitectónico, como un estilo de arquitectura, impone la transformación del diseño de una arquitectura. Sin embargo, un patrón difiere de un estilo en varias formas fundamentales:

1. el alcance del patrón es menos amplio y se centra en un aspecto de la arquitectura más que en el total de ésta.
2. un patrón impone una regla a la arquitectura, describe la manera en la que el software manejará ciertos aspectos de su funcionalidad en el nivel de la infraestructura (por ejemplo, la concurrencia).
3. los patrones arquitectónicos tienden a abocarse a aspectos específicos del comportamiento en el contexto de la arquitectura (por ejemplo, cómo manejarán la sincronización o las interrupciones las aplicaciones en tiempo real). Los patrones se utilizan junto con un estilo arquitectónico para dar forma a la estructura general de un sistema (Pressman, 2010).

MVC (Modelo-Vista-Controlador) es un patrón de diseño que se estructura mediante tres componentes: modelo, vista y controlador. Este patrón tiene como principio que cada uno de los componentes esté separado en diferentes objetos, esto significa que los componentes no se pueden combinar dentro de una misma clase. Sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario.

- **Modelo**: este componente se encarga de manipular, gestionar y actualizar los datos de una base de datos. No contiene ninguna lógica que describa como presentar los datos a un usuario.
- **Vista**: este componente presenta los datos del modelo al usuario. La vista sabe cómo acceder a los datos del modelo, pero no sabe que significa esta información o que puede hacer el usuario para manipularla.
- **Controlador**: este componente se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. Por medio del controlador se comunican el modelo y la vista: solicitando los datos necesarios, manipularlos para obtener los resultados y entregarlos a la vista para que pueda mostrarlos.

Los tres componentes de **MVC** están interconectados. La vista muestra el modelo para el usuario, después el controlador acepta la entrada del usuario y actualiza el modelo y debido a esta acción la vista vuelve a tener un cambio con los datos actualizados (Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo? | Easy App CODE, 2023).

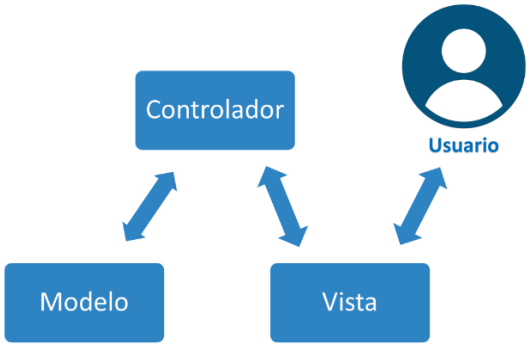


Figura 5. Arquitectura MVC. Elaboración propia.

2.8.1 Patrones de diseño

Los patrones de diseño son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Patrones de diseño con TypeScript en el mundo real, 2023).

Singleton es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia (Refactorización y patrones de diseño, 2023), lo que lo hace excelente para la administración de un estado global.

```
export class AsignaturaService {
    constructor(@InjectModel('Asignatura') private asignaturaModel: Model<Asigna-
turaDocument>) {}

    async create(asignaturaDto: CreateAsignaturaDto) {
        try {
            const newAsignatura = new this.asignaturaModel(asignaturaDto);
            const asignatura = await newAsignatura.save();
            return {
                err: false,
                asignatura,
                msg: 'Asignatura creada con éxito'
            };
        } catch (err) {
            console.log(err);
            throw new InternalServerErrorException({
```

```
                err,
                msg: 'Error al crear la asignatura'
            });
        }
    }
```

Composite es un patrón de diseño estructural que te permite componer objetos en estructuras de árbol y trabajar con esas estructuras como si fueran objetos individuales (Refactorización y patrones de diseño, 2023).

```
import {
    Grid,
    Card,
    CardContent,
    CardActions,
    Button,
    Typography
} from "@material-ui/core";

const AsignaturaCard = ({ asignatura, goToAsignatura }) => (
    <Grid item md={3}>
        <Card onClick={() => goToAsignatura(asignatura)} className="pointer link">
            <CardContent>
                <Typography variant="h5" component="h2">
                    {asignatura.name}
                </Typography>
                <Typography variant="body2" color="textSecondary" component="p">
                    {asignatura.description && typeof asignatura.description ===
'string'
                        ? asignatura.description.length <= 60
                            ? asignatura.description
                            : asignatura.description.substring(0, 60) + '...'
                        : ''}
```

Facade es un patrón de diseño estructural que proporciona una interfaz simplificada a una biblioteca, un *framework* o cualquier otro grupo complejo de clases. Una fachada es una clase que proporciona una interfaz simple a un subsistema complejo que contiene muchas partes móviles. Una fachada puede proporcionar una funcionalidad limitada en comparación con trabajar directamente con el subsistema. Sin embargo, tan solo incluye las funciones realmente importantes para los clientes (Refactorización y patrones de diseño, 2023).

```
const NavBar = () => {
  const [user, setUser] = useState({});
  const [open, setOpen] = useState(false);
  const [token, setToken] = useState(null);

  useEffect(() => {
    let user = localStorage.getItem('user');
    let token = localStorage.getItem('token');
    if (user && user !== 'undefined' && token && token !== 'undefined') {
      let userJson = JSON.parse(user);
      setUser(userJson);
      setToken(token);
    }
  }, []);
```

2.8.2 Patrones GRASP

El uso de los patrones GRASP - *General Responsibility Asigment Software Patterns* (patrones generales de software para asignar responsabilidades) ayuda al entendimiento y la aplicación de los conceptos sobre la POO (Programación Orientada a Objetos) y a diseñar software con mayor calidad aplicando buenas prácticas que provienen de la experiencia de años de desarrollo. Los patrones GRASP, se basan en la asignación de responsabilidades que es uno de los objetivos primordiales de la POO (Vargas Ortega, 2021).

Experto

- **Problema:** Cuales son las clases y que nivel de responsabilidad se les puede llegar a asignar dentro del sistema.
- **Solución:** Asignar responsabilidades a las clases cuyos objetos poseen la información necesaria para cumplirlas. Para poder cumplir con esto es posible que requiera información distribuida en otros objetos, es decir, otros expertos. Para asignar las responsabilidades es indispensable apoyarse en los diagramas de colaboración (UML), quienes son la base para la toma de decisiones en la elaboración del diseño orientado a objetos bajo los Patrones GRASP (Vargas Ortega, 2021).

```
export interface EstudianteDocument extends mongoose.Document {
    name: string;
    apellidos: string;
    ci: number;
    yearEnd: number;
```

```
    cursos: mongoose.Types.ObjectId[];
    numeroExpediente: number;
}

export const EstudianteSchema = new mongoose.Schema({
    name: {
        type: String,
        required: "Campo obligatorio"
    },
```

Bajo acoplamiento

- **Problema:** Como evitar la gran dependencia generada entre las clases que forman parte del sistema sin que esto conlleve a traumatismo y la generación de código excesivo para compensar en algún momento la ruptura en la relación de clases muy dependientes.
- **Solución:** Se debe tener poca dependencia entre las clases, el tener una alta dependencia entre clases creará una gran cantidad de software, el objetivo de este patrón es invitar a la creación de clases con la suficiente información para que su relación con otras se reduzca y puedan en algún momento subsistir si una de las clases a las cuales se relaciona deja de existir (Vargas Ortega, 2021).

```
@Module({
    imports: [
        MongooseModule.forFeature([
            { name: 'Resolucion', schema: ResolucionSchema },
            { name: 'GradoResolucion', schema: GradoResolucionSchema }
        ]),
    ],
    controllers: [ResolucionController],
    providers: [ResolucionService, GradoResolucionService],
    exports: [ResolucionService],
})
export class ResolucionModule {}
```

Alta Cohesión

- **Problema:** Como crear clases que contengan la suficiente información, sin que se llegue al exceso de contenido, para que puedan funcionar de una manera óptima y así evitar la alta dependencia con otras que le suministren datos necesarios para su existencia.

- **Solución:** Una clase con alta cohesión posee una importante funcionalidad y poco trabajo, colabora con otros objetos para compartir tareas muy grandes. Su alto grado de funcionalidad permite simplificar el mantenimiento y los mejoramientos (Vargas Ortega, 2021).

```
@Module({
  imports: [ ConfigModule.forRoot({
    envFilePath:  '.env' ,
  }),
    MongooseModule.forRoot(process.env.DB),
    UserModule,
    AuthModule,
    AsignaturaModule,
    ResolucionModule,
    EstudianteModule],
  controllers: [AppController],
  providers: [
    AppService,
    {
      provide: APP_INTERCEPTOR,
      useClass: LoggingInterceptor,
    },
  ],
})
export class AppModule {}
```

Conclusiones parciales

A partir del modelado del proceso actual se logra un mejor entendimiento de las actividades que se realizan durante el proceso de emisión de certificación de estudios terminados en el departamento de títulos y diplomas, lo que establece las bases para ofrecer una mejora de los mismos. La obtención de los diagramas de proceso mejorado de cada uno de estos, permitió establecer las actividades, los actores, sus entradas, salidas y las reglas que intervienen. También fueron identificados los requisitos funcionales y no funcionales del sistema propuesto, lo que permite determinar las capacidades y propiedades con que debe contar el proceso mejorado e identificar las funcionalidades a desarrollar. La arquitectura de software basada en MVC con sus patrones de diseño *Singleton, Composite* y *Facade* sentaron las bases para la construcción de la solución.

CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

El presente capítulo está enfocado a la implementación de la aplicación para dar solución a los requisitos especificados. Se elaboran los diagramas de componentes que forman parte del modelo de implementación. Se define el estándar de codificación a utilizar en la implementación de la solución y se realiza el diagrama de despliegue. Además, se realizan las Pruebas Internas y Pruebas de Aceptación que propone la metodología AUP-UCI.

3.1 Diagrama de componentes

Los diagramas de componentes UML representan las relaciones entre los componentes individuales del sistema mediante una vista de diseño estática. Pueden ilustrar aspectos de modelado lógico y físico.

En el contexto del UML, los componentes son partes modulares de un sistema independientes entre sí, que pueden reemplazarse con componentes equivalentes. Son autocontenidos y encapsulan estructuras de cualquier grado de complejidad. Los elementos encapsulados solo se comunican con los otros a través de interfaces. Los componentes no solo pueden proporcionar sus propias interfaces, sino que también pueden utilizar las interfaces de otros componentes, por ejemplo, para acceder a sus funciones y servicios. A su vez, las interfaces de un diagrama de componentes documentan las relaciones y dependencias en una arquitectura de software (Diagrama de componentes, 2020).

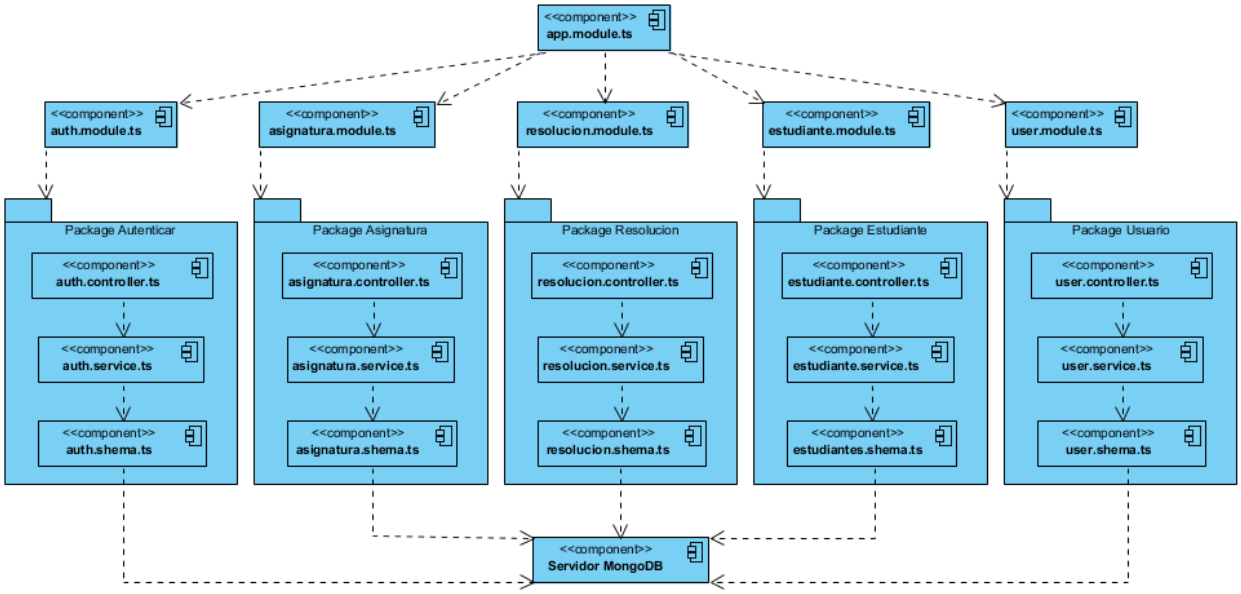


Figura 6. Diagrama de componentes NestJs API. Elaboración propia.

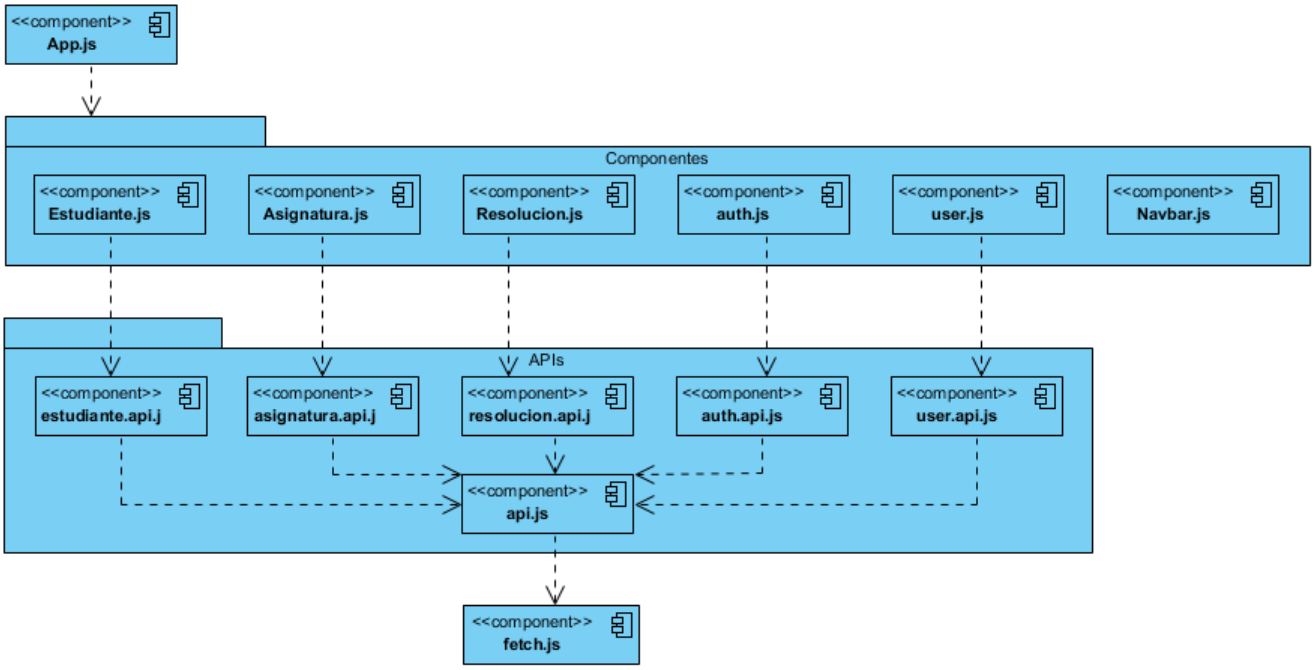


Figura 7. Diagrama de componentes ReactApp. Elaboración propia.

3.2 Estándares de codificación

- Los nombres de variables deben ser escritos con notación camelCase.
- Los espacios entre el nombre y el valor de una variable son muy importantes, puesto que mejoran la legibilidad del código.
- Cuando se declaran múltiples variables deben declararse todas con la misma sentencia *var*, una variable por línea y con coma (,) al final a menos que sea la última variable, en cuyo caso se usará punto y coma (;) al final de la línea. A partir de la segunda línea debe usarse 4 espacios antes del nombre de variable para mejorar la legibilidad del código fuente. El orden de las variables debe ser alfabético de ser posible.
- Cuando la variable a declarar contiene un elemento de tipo colección (arreglo y objeto) y su contenido excede los 70 caracteres o tiene más de 2 elementos debe tener cada ítem de la colección en una línea diferente para mejorar la legibilidad del código.
- Los nombres de funciones deben ser escritos con notación camelCase.
- Las llaves de las funciones deben empezar en la misma línea que se declara la función y debe haber un espacio entre el paréntesis de cierre de argumentos y la llave de inicio de cuerpo de función.

- En javascript las funciones pueden ser funciones como tal o funciones como variables. En caso de que sean funciones como variables es necesario usar punto y coma (;) después del cuerpo de la función para finalizar la sentencia.
- Para mejorar la legibilidad del código javascript los argumentos de funciones deben ser nombres con notación camelCase. Si son varios argumentos deben estar separados por coma (`,`) y un espacio.
- Deben ser usadas comillas simples (') puesto de ésta manera se reducen los problemas a la hora de escapar las comillas dobles (") usadas cuando se genera HTML y mejora la legibilidad.
- Todas las operaciones deben tener espacios entre los operadores y los operandos para mejorar la legibilidad del código. Si se están usando paréntesis no es necesario usar espacio entre los paréntesis y los operandos.
- Deben usarse los comparadores estrictos (aquellos que comparan valor y tipo `===`, `!==`) para todas las operaciones lógicas de comparación.

3.3 Implementación del sistema

A continuación, se muestra la implementación del sistema para la gestión del proceso de emisión de certificación de estudios terminados. Este fue implementado en VS Code con una API desarrollada en NestJs y una aplicación en React. Se procederá por pasos lógicos:

1. Preparar el ambiente de desarrollo.
2. Crear un nuevo proyecto en NestJs e instalar las dependencias.
3. Crear los esquemas de acceso a datos utilizando ‘mongoose’.
4. Implementar en los servicios la lógica de negocio para cada funcionalidad.
5. Configurar los *endpoints* necesarios.
6. Implementar las pruebas de la API.
7. Crear una nueva app en React e instalar las dependencias.
8. Crear los APIs para las peticiones http.
9. Crear los componentes de las vistas.
10. Implementar las pruebas de la aplicación.

Después de realizar estos pasos queda creada la API en NestJs y la aplicación en React para la gestión del proceso de emisión de certificación de estudios terminados.

NestJS

```
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors();
  app.setGlobalPrefix('api');
  const config = new DocumentBuilder()
    .setTitle('Ejemplo de API')
    .setDescription('Descripción de la API')
    .setVersion('1.0')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('docs', app, document);

  await app.listen(3000);
}
bootstrap();
```

React

```
function App() {
    return (
        <Router>
            <div className="App">
                <NavBar />
                <MyRouter />
            </div>
        </Router>
    );
}

export default App;
```

3.4 Diagrama de despliegue

Los diagramas de despliegue muestran la configuración de los elementos del proceso de tiempo de ejecución y las instancias de componente de software que residen en ellos. Son utilizados para especificar la arquitectura física de tiempo de ejecución de un sistema. En concreto, son gráficos de nodos conectados por asociaciones de comunicación. Las instancias de componente se asignan para ejecutarse en nodos específicos durante la ejecución del programa. Las líneas de relación representan vías de acceso de comunicación entre nodos (IBM Documentation, 2021).

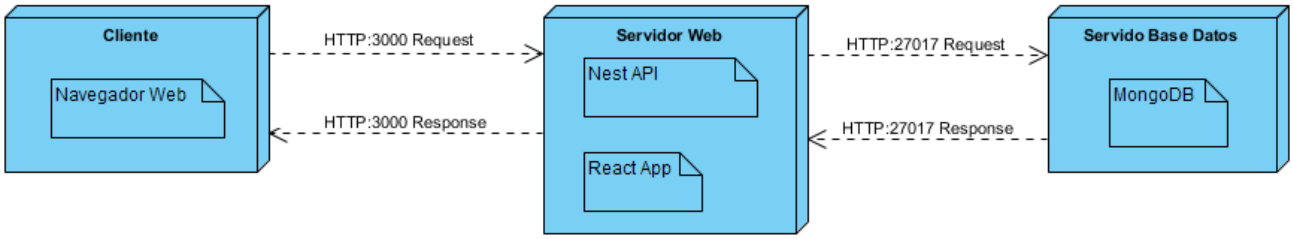


Figura 8. Diagrama de despliegue. Elaboración propia.

3.5 Pruebas de software

Las pruebas de software son un método para verificar si el producto real cumple con los requisitos esperados y garantizar que el producto esté libre de defectos. Implica la ejecución de componentes de software/sistema, utiliza herramientas manuales o automatizadas para evaluar una o más propiedades de interés. El propósito de las pruebas de software es identificar errores, brechas o requisitos faltantes en contraste con los requisitos reales.

Algunos prefieren definir una prueba de software como una prueba de caja blanca y caja negra. En términos simples, pruebas de software significa la verificación de la aplicación bajo prueba (*Application Under Test*, AUT, por sus siglas en inglés) (Hamilton, 2023).

3.5.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación, prueba cada construcción, incluye tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componente de prueba ejecutable para automatizar las pruebas (Pressman, 2010).

En este contexto se realizan pruebas a la aplicación mediante el empleo de diferentes métodos:

- Prueba de caja blanca, con su técnica del camino básico.
- Prueba de caja negra, con su técnica de partición equivalente.
- Pruebas unitarias, con el aprovechamiento de la herramienta *Jest.*

3.5.1.1 Prueba de caja blanca

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El ingeniero de pruebas escoge distintos valores de entrada para

examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados. Al estar basadas en una implementación concreta, si esta se modifica, por regla general las pruebas también deberán rediseñarse.

Aunque las pruebas de caja blanca son aplicables a varios niveles, habitualmente se aplican a las unidades de software. Su cometido es comprobar los flujos de ejecución dentro de cada unidad, pero también pueden probar los flujos entre unidades durante la integración, e incluso entre subsistemas, durante las pruebas (*Wikiwand - Pruebas de caja blanca*, 2023).

Prueba de camino básico

La prueba del camino básico es una técnica de prueba de la Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo lo enunciado en el programa, al menos una vez durante la prueba (Pressman, 2010).

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, obtiene el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo obtenido.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico (Pérez Pincay, 2017).

Para la realización de la Prueba de caja blanca se escogió el método *validateUser* por todos los criterios.

Método *validateUser* de la clase *AuthService.*

```
    async validateUser(payload: any) {
        // Buscar al usuario por su correo electrónico
        const user = await this.userService.getUserByEmail(payload.email);
        // Verificar si el usuario existe
        if (!user) {
            throw new UnauthorizedException();
```

```
        }
        // Devolver el objeto que representa al usuario
        return user;
    }
```

Descripción simplificada del método paso a paso:

1. **Inicio**: Se llama a la función *validateUser*.
2. **Buscar usuario**: Se busca al usuario por su correo electrónico.
3. **Verificación**: Se verifica si el usuario existe.
    4. Si el usuario existe, se devuelve el objeto del usuario.
    5. Si el usuario no existe, se lanza una excepción *UnauthorizedException*.
6. **Fin**: Termina el proceso.

Grafo resultante:

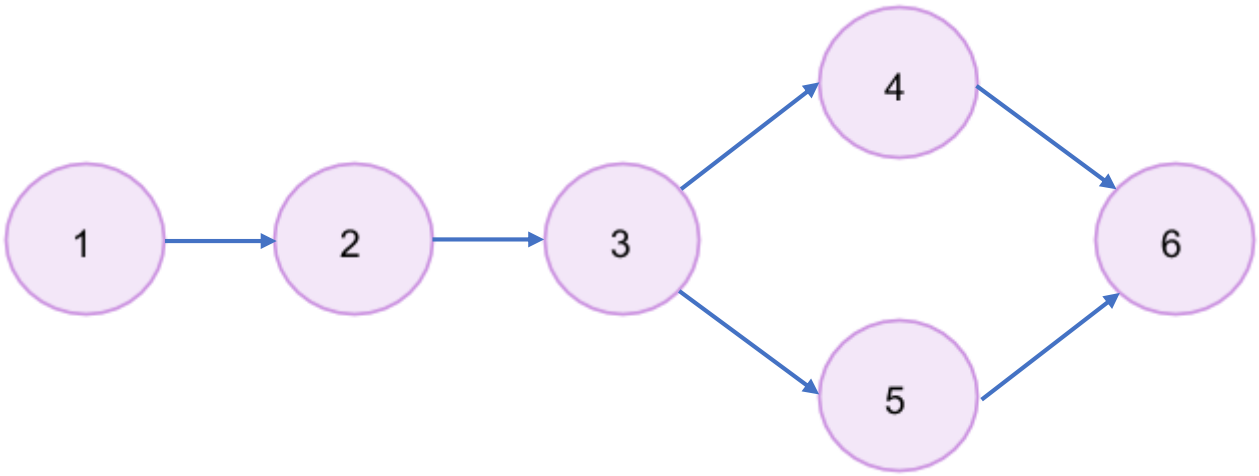


Figura 9. Grafo resultante. Elaboración propia.

Leyenda	Complejidad Ciclomática:	Caminos lógicos:
• E (# de aristas) • N (# de nodos) • P (regiones cerradas)	V(G) = # de regiones = 2 V(G) = E-N+2 = 6-6+2 = 2 V(G) = P+1 = 1+1 = 2	Camino #1: 1 2 3 4 6 Camino #2: 1 2 3 5 6

Descripción de los caminos obtenidos

Tabla 3. Descripción del camino #1. Elaboración propia.

Caso de prueba #1 para el camino básico:	1 2 3 4 6
Descripción:	Si el usuario existe, se devuelve el objeto del usuario.
Condición de Ejecución	Ninguna
Procedimiento prueba automatizada	
Datos de entrada	*payload*
Tipo de datos esperados	*any*
Evaluación del caso de prueba	Satisfactoria.

Tabla 4. Descripción del camino #2. Elaboración propia.

Caso de prueba #1 para el camino básico:	1 2 3 5 6
Descripción:	Si el usuario no existe, se lanza una excepción *UnauthorizedException*.
Condición de Ejecución	Ninguna
Procedimiento prueba automatizada	
Datos de entrada	*payload*
Tipo de datos esperados	*any*
Evaluación del caso de prueba	Satisfactoria.

3.5.1.2 Pruebas de caja negra

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obvia el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Técnica de prueba para el método de caja negra: Partición Equivalente.

Esta técnica intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia y definir el menor número de casos de prueba que descubran clases de errores (Sommerville, 2016).

El diseño de casos de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia.
2. Identificar los casos de prueba.

Una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución, resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final (Jacobson, 1998).

A continuación, se muestra la descripción del caso de prueba Gestionar Usuario (Tabla 5), descrito por escenarios.

Tabla 5. Descripción del escenario de Prueba del Caso de Uso "Gestionar Usuario". Elaboración Propia.

Caso de Uso	Entrada	Partición	Resultado Esperado
Crear usuario	Datos de entrada correctos (nombre de usuario, correo electrónico, contraseña).	Partición válida	Usuario creado exitosamente.
Crear usuario	Datos de entrada incorrectos (nombre de usuario ya existente, correo electrónico no válido, contraseña débil).	Partición inválida	Error al crear usuario.
Modificar usuario	Cambios permitidos (actualización de correo electrónico, cambio de contraseña).	Partición válida	Usuario modificado exitosamente.
Modificar usuario	Cambios no permitidos (modificación del nombre de usuario a uno que ya existe, correo electrónico no válido, contraseña débil).	Partición inválida	Error al modificar usuario.
Eliminar usuario	Usuario existente.	Partición válida	Usuario eliminado exitosamente.
Eliminar usuario	Usuario no existente.	Partición inválida	Error al eliminar usuario.

3.5.1.3 Pruebas unitarias

Se llaman pruebas unitarias porque descomponen las funciones del programa en comportamientos comprobables discretos que se pueden probar como unidades individuales. Las pruebas unitarias tienen el mayor efecto en la calidad del código cuando son parte integral del flujo de trabajo de desarrollo de software. En el desarrollo controlado por pruebas, las pruebas unitarias se crean antes de escribir el código, de modo que se usan como documentación de diseño y como especificaciones de las funciones (Mikejo5000, 2023).

Estas son las razones clave para realizar pruebas unitarias en ingeniería de software:

1. Las pruebas unitarias ayudan a corregir errores al principio del ciclo de desarrollo y ahorran costos.
2. Ayuda a los desarrolladores a comprender la base del código de prueba y les permite realizar cambios rápidamente.
3. Las buenas pruebas unitarias sirven como documentación del proyecto.
4. Las pruebas unitarias ayudan con la reutilización del código. Migrar tanto su código como sus pruebas a un nuevo proyecto. Modificar el código hasta que las pruebas se ejecuten nuevamente (Hamilton, 2023).

Jest v29.5.0:

Jest es un *framework* de prueba para el entorno de React que fue desarrollado por Facebook (ahora Meta) en 2014. La principal ventaja de Jest es que puede testear React de forma directa y nativa, gracias a que tienen el mismo desarrollador.

Algunas de las características principales de las pruebas unitarias con Jest son las siguientes:

- La herramienta de Jest tiene una sintaxis clara.
- Jest tiene contextos únicos.
- Con Jest podemos realizar pruebas descriptivas, lo que ayudará a conformar repositorios y mejores resultados finales (KeepCoding, 2023).

Dependencias utilizadas

```
  "devDependencies": {
    "@nestjs/testing": "^10.0.0",
    "@types/jest": "^29.5.2",
    "@types/supertest": "^2.0.12",
    "jest": "^29.5.0",
    "supertest": "^6.3.3",
    "ts-jest": "^29.1.0",
  },
```

Instancia de Jest para probar la clase *AsignaturaService*

```
describe('AsignaturaService', () => {
    let service: AsignaturaService;
    // @ts-ignore
    let model: Model;
```

```
    beforeEach(async () => {
        const module: TestingModule = await Test.createTestingModule({
            providers: [
                AsignaturaService,
                {
                    provide: getModelToken('Asignatura'),
                    useValue: Model,
                },
            ],
        }).compile();

        service = module.get<AsignaturaService>(AsignaturaService);
        // @ts-ignore
        model = module.get<Model>(getModelToken('Asignatura'));
    });

    it('should be defined', () => {
        expect(service).toBeDefined();
    });

    it('should create asignatura', async () => {
        const result  = { name: 'Asignatura 1' };
        jest.spyOn(model, 'save').mockImplementation(() => result);

        if (result instanceof CreateAsignaturaDto) {
            expect(await service.create(result)).toBe(result);
        }
    });
```

3.5.2 Pruebas de aceptación

Las pruebas de aceptación son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas. Estas pruebas generalmente son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción.

Las pruebas de aceptación son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software. Estas pruebas se realizan una vez que ya se ha probado que cada módulo funciona bien por separado, que el software realice las funciones esperadas y que todos los módulos se integran correctamente. Además, terminan de definir el nivel de calidad de la aplicación y le permite conocer al equipo qué tan bien supo interpretar correctamente los requerimientos del usuario. Es el cliente quien tendrá la decisión final de aprobar o no el producto, como también de solicitar modificaciones (Que son las pruebas de aceptación?, 2020).

Resultados de las pruebas de aceptación

Durante la ejecución de las pruebas de aceptación fueron detectadas un total de cinco no conformidades (NC) de funcionalidad en tres iteraciones, en la primera iteración fueron detectadas tres no conformidades, quedaron resueltas dos y pendiente una, en la segunda iteración se detectaron dos no conformidades más, quedaron resueltas dos no conformidades y pendiente una, que fue resuelta en la tercera iteración, en esta última no quedó pendiente ninguna no conformidad. De forma general se excluyeron todos los errores culminando las pruebas con resultados satisfactorios.

Las no conformidades fueron:

1. Al guardar un estudiante, una asignatura o una resolución modificada el sistema no redirecciona a la vista correspondiente.
2. No esta implementado el requisito funcional “Eliminar usuario”.
3. El botón “Limpiar” no realiza la función esperada.
4. No esta implementado el requisito funcional “Exportar certificación”.
5. Al crear un usuario aparece con el rol “USER” por defecto.

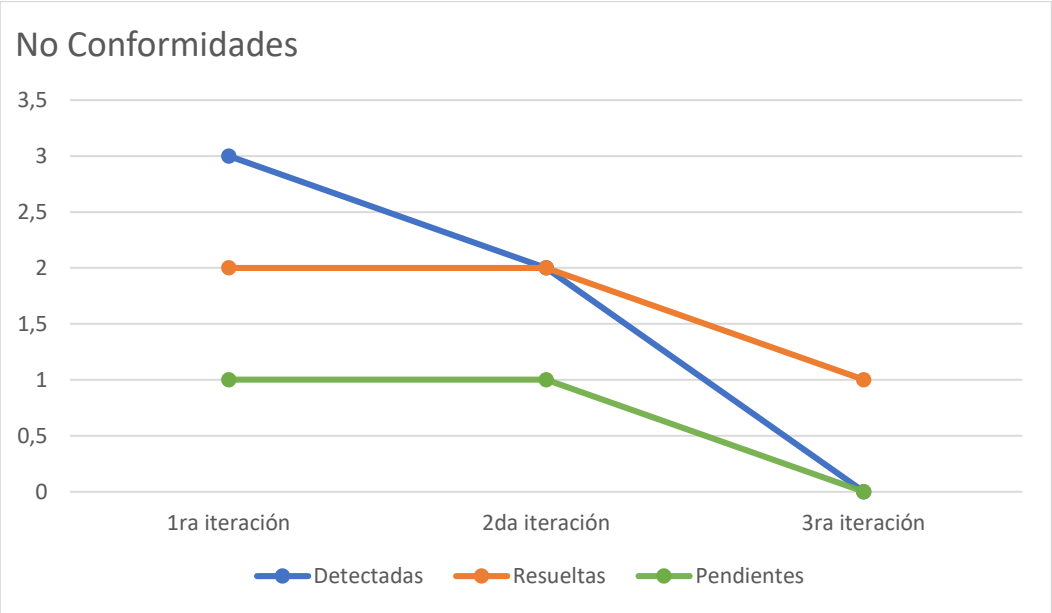


Figura 10. Resultados de las pruebas de aceptación. Elaboración propia.

Conclusiones parciales

En este capítulo como parte de la implementación y pruebas de las aplicaciones se realizaron los siguientes pasos:

- Se estructuró el modelo de implementación a través de los diagramas de componentes, los cuales permitieron mostrar los componentes y sus relaciones sobre la base de la arquitectura definida para el desarrollo de las aplicaciones.
- Se realizó la implementación de las aplicaciones con la aplicación de los estándares de codificación definidos.
- Como parte de la validación de la aplicación y su correcto funcionamiento se realizaron pruebas con el uso del método de caja blanca con su técnica de camino básico y las pruebas unitarias a nivel de desarrollador con la herramienta *Jest*; las pruebas funcionales a nivel de sistema basada en la técnica de partición de equivalencia del método de caja negra, guiados por los diseños de casos de prueba basados en casos de uso. Además, se realizaron las pruebas de aceptación con la obtención de resultados satisfactorios.

CONCLUSIONES

1. Mediante la investigación de los referentes teórico-metodológicos, se ha mejorado la comprensión de las deficiencias presentes y se obtuvo información útil para la elaboración de la propuesta de solución.
2. La selección adecuada de la metodología de software, herramientas informáticas y tecnologías ha sido fundamental para garantizar el desarrollo exitoso del sistema de gestión de certificación de estudios terminados.
3. La especificación detallada de los requisitos funcionales y no funcionales, así como la definición de casos de uso y la implementación de patrones de diseño y arquitectura, ha permitido una adecuada planificación y desarrollo del sistema.
4. La implementación del sistema, priorizando las funciones más importantes en el proceso de gestión de certificación de estudios terminados, mejora el proceso de emisión de dichas certificaciones en el departamento de títulos y diplomas.
5. La validación de los resultados obtenidos con la implementación del nuevo sistema ha permitido la eliminación de errores y no conformidades, así como garantizar el correcto funcionamiento de la aplicación desarrollada.

RECOMENDACIONES

Como parte del proceso investigativo llevado a cabo se recomienda:

1. Continuar con la investigación para aumentar las funcionalidades del sistema, con el objetivo de obtener mejoras en futuras versiones del mismo.
2. Extender el sistema de manera que pueda ser utilizado en todas las direcciones municipales de educación del país.

REFERENCIAS BIBLIOGRÁFICAS

1. Báster, D. R. L. (2011). *Título: Infraestructura de firma y validación digital de los documentos clínicos electrónicos generados por el sistema alas HIS*.
2. Blanco, A. Y. R. (2023). *Arquitectura de Microservicios aplicada a servicios de validación de identidad digital en la plataforma INTEGRO.*
3. *CENACAD Censo Académico en Línea*. (2023). https://cenacad.espol.edu.ec/
4. Coppola, M. (2023). *Qué es JavaScript, para qué sirve y cómo funciona*. https://blog.hubspot.es/website/que-es-javascript
5. *CSS | MDN*. (2023, marzo 13). https://developer.mozilla.org/es/docs/Web/CSS
6. *Diagrama de componentes: Modelado eficiente de sistemas con módulos de software*. (2020, septiembre 23). IONOS Digital Guide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-componentes/
7. *FACSO*. (2023). https://www.facso.unsj.edu.ar/
8. Fernández Sanz, L., & Bernad Silva, P. (2014). Risk management in software development projects in Spain: A state of art. *Revista Facultad de Ingeniería Universidad de Antioquia*, *70*, 233-243. https://doi.org/10.17533/udea.redin.13376
9. Hamilton, T. (2023, septiembre 19). *What is Software Testing? Definition*. https://www.guru99.com/software-testing-introduction-importance.html
10. Hernández Sampieri, R. (2010). *METODOLOGÍA de la investigación*.
11. *HTML: Lenguaje de etiquetas de hipertexto | MDN*. (2023, julio 24). https://developer.mozilla.org/es/docs/Web/HTML
12. *HTML5 - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN*. (2022, noviembre 30). https://developer.mozilla.org/es/docs/Glossary/HTML5
13. *IBM Documentation*. (2021, marzo 9). https://www.ibm.com/docs/es/engineering-lifecycle-management-suite/lifecycle-management/6.0.3?topic=requirements-defining-use-cases
14. *Informatización—Ministerio de Educación de la República de Cuba*. (2023). https://www.mined.gob.cu/tecnologia-educativa/informatizacion/

15. Jacobson, I. (1998). *Applying UML in The Unified Process*.
16. KeepCoding, R. (2023, abril 25). *Realizar el unit testing con Jest | KeepCoding Bootcamps*. https://keepcoding.io/blog/unit-testing-con-jest/
17. Mastrapa, E. (2014). *Sistema de apoyo a la gestión de la información académica*.
18. Mikejo5000. (2023, mayo 8). *Datos fundamentales de las pruebas unitarias—Visual Studio (Windows)*. https://learn.microsoft.com/es-es/visualstudio/test/unit-test-basics?view=vs-2022
19. Montero, Y. M., Padrón, I. S. N., Rodríguez, C. Y. R., & Darias, I. D. H. (2020). *Título: Módulo Trámites Docentes 1.1 del Sistema de Gestión Académica XAUCE AKADEMOS para el MINED*.
20. *NestJS: qué es y por qué empezar a usarlo – GFourmis*. (2023). https://gfourmis.co/nestjs-que-es-y-porque-empezar-a-usarlo/
21. Osorio, R. C. P. (2020). *Módulo titulación para el sistema de gestión académica del Ministerio de Educación*.
22. Paez, N., & Fontela, C. (2014). *Construcción de Software: Una mirada ágil*.
23. *Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo? | Easy App CODE*. (2023). https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo
24. *Patrones de diseño con TypeScript en el mundo real: Creacionales y estructurales | Software Crafters*. (2023). Patrones de diseño con TypeScript en el mundo real: creacionales y estructurales | Software Crafters. http://softwarecrafters.iopatrones-diseno-creacionales-estructurales
25. Pérez Pincay, C. J. (2017, mayo 28). Ingenieria del software: Prueba de la caja blanca y camino básico. *Monografias.com*. https://www.monografias.com/docs113/ingenieria-software-prueba-caja-blanca-y-camino-basico/ingenieria-software-prueba-caja-blanca-y-camino-basico
26. *Portal de Sistemas de Gestión Académica*. (2023). https://siga.usm.cl/pag/home.jsp
27. Pressman, R. S. (2010). *Ingeniería del software.* (7.a ed.).
28. *¿Qué es MongoDB? | IBM*. (2023). https://www.ibm.com/es-es/topics/mongodb
29. *Qué es NodeJS y para qué sirve | OpenWebinars*. (2023). https://openwebinars.net/blog/que-es-nodejs/
30. *Qué es React: Definición, características y funcionamiento*. (2023). https://www.hostinger.es/tutoriales/que-es-react

31. *¿Que es TypeScript?* (2023). Https://Www.Victorvr.Com/Tutorial/Que-Es-Typescript. https://www.victorvr.com/tutorial/que-es-typescript
32. *Qué es Visual Studio Code y qué ventajas ofrece | OpenWebinars*. (2023). https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/
33. *Que son las pruebas de aceptación?* (2020). https://es.linkedin.com/pulse/que-son-las-pruebas-de-aceptaci%C3%B3n-los-andes-training
34. *Refactorización y patrones de diseño*. (2023). https://refactoring.guru/es
35. Sánchez, T. (2019). *Metodología de desarrollo AUP UCI*. *12*.
36. Sommerville, I. (2016). *Software engineering* (Tenth edition, global edition). Pearson.
37. Sparks, G., & Pinciroli, F. (2003). *El Modelo de Casos de Uso*.
38. Vargas Ortega, G. A. (2021). *Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP*. https://doi.org/10.5281/ZENODO.5716304
39. *Wikiwand—Pruebas de caja blanca*. (2023). Wikiwand. https://www.wikiwand.com/es/Pruebas_de_caja_blanca

40. ANEXOS

Anexo 1: Casos de uso del sistema

Tabla 6. Descripción de caso de uso del sistema Autenticar Usuario. Elaboración propia.

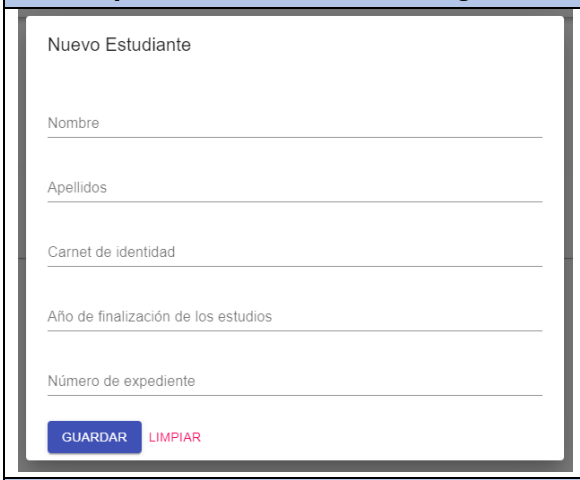
Objetivo	Permitir al usuario del sistema poder acceder a las funcionalidades según le correspondan.	
Actores	Usuario del Sistema.	
Resumen	El caso de uso se inicia cuando el usuario del sistema decide autenticarse para acceder a las operaciones que le permite realizar el sistema.	
Complejidad	Baja.	
Prioridad	Alta.	
Precondiciones	El sistema debe estar activo.	
Postcondiciones	Se autenticó un usuario del sistema.	
Flujo de eventos		
Flujo básico "Autenticar usuario"		
Actor		**Sistema**
1.	Introduce la URL para acceder.	
2.		Muestra una pantalla con los siguientes datos a introducir: • usuario, • contraseña Y el botón "Iniciar Sesión".
3.	Introduce los datos y presiona el botón "Iniciar Sesión".	
4.		Valida que los campos estén correctos.
5.		Verifica existencia del usuario.
6.		Encripta la contraseña.
7.		Permite acceder a las funcionalidades según los privilegios. *Finalizando así el Caso de Uso.*
Prototipo elemental de interfaz gráfica		

Flujos Alternos		
4a. Campos vacíos y/o incorrectos		
Actor		**Sistema**
1.		Muestra un mensaje indicado: “Datos incorrectos”.
5a. Usuario inexistente		
Actor		**Sistema**
1.		Muestra un mensaje indicado: “Acceso denegado”.
Relaciones	**CU Incluidos**	No Aplica.
	CU Extendidos	No Aplica.
Requisitos no funcionales	No Aplica.	

Tabla 7. Descripción de caso de uso del sistema Gestionar Estudiante. Elaboración propia.

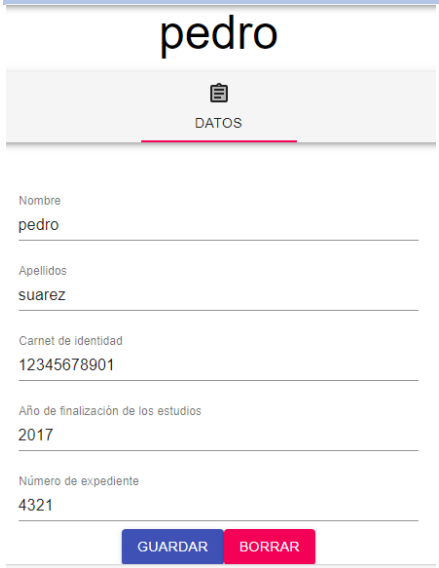
Objetivo	Permitir crear, modificar, eliminar, listar, buscar un estudiante o exportar certificación.	
Actores	Administrador o asesor.	
Resumen	El caso de uso se inicia cuando el administrador o el asesor decide crear, modificar, eliminar, buscar un estudiante o exportar certificación.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	El administrador o asesor debe estar autenticado.	
Postcondiciones	Se creó, modificó, eliminó, buscó un estudiante o exportó certificación de estudiante.	
Flujo de eventos		
Flujo básico Gestionar estudiante		
Actor		**Sistema**
1.	Selecciona de la página principal la opción “Gestionar estudiante”.	

2.		Muestra una pantalla con los estudiantes existentes y las opciones Crear, Modificar, Eliminar, Buscar estudiante(s) y Exportar certificación.
3.	Desea crear, modificar, eliminar, buscar estudiante(s) o exportar certificación.	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: 1. Si decide crear un estudiante, ir a la sección “Crear estudiante”. 2. Si decide modificar un estudiante, ir a la sección “Modificar estudiante”. 3. Si decide eliminar el estudiante, ir a la sección “Eliminar estudiante”. 4. Si decide buscar un estudiante, ir a la sección “Buscar estudiante”. 5. Si decide exportar certificación, ir a la sección “Exportar certificación”
Prototipo elemental de interfaz gráfica		
Flujos alternos		
2a. Listado de estudiante vacío		
Actor		**Sistema**
1.		Carga una pantalla en blanco sin estudiantes creados.
Sección 1: “Crear estudiante”		
Flujo básico Crear estudiante		
Actor		**Sistema**
1.	Presiona el botón “Crear estudiante”.	
2.		Muestra una ventana con los siguientes campos a introducir: • nombre, • apellidos,

		• CI, • numero_expediente, • año_final Y el botón “Guardar”.
3.	Introduce los datos y presiona el botón “Guardar”.	
4.		Verifica que todos los campos estén llenos y que los datos introducidos estén correctos.
5.		Almacena los datos de los estudiantes y muestra el mensaje “Estudiante creado satisfactoriamente”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujos alternos		
4a. Campos vacíos y/o Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
Sección 2: “Modificar Estudiante”		
Flujo básico Modificar Estudiante		
Actor		**Sistema**
1.	Selecciona uno de los estudiantes mostrados, escoge la opción “Modificar estudiante”.	
2.		Muestra una ventana con los datos a modificar: • nombre, • apellidos, • CI, • numero_expediente, • año_final Y el botón “Guardar”.

3.	Realiza las actualizaciones deseadas y presiona el botón “Guardar”.	
4.		Verifica que los datos introducidos estén correctos.
5.		Actualiza la información incorporada al estudiante y se emite un mensaje “Modificación satisfactoria”. *Finalizando así el caso de uso.*

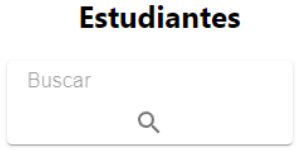
Prototipo elemental de interfaz gráfica



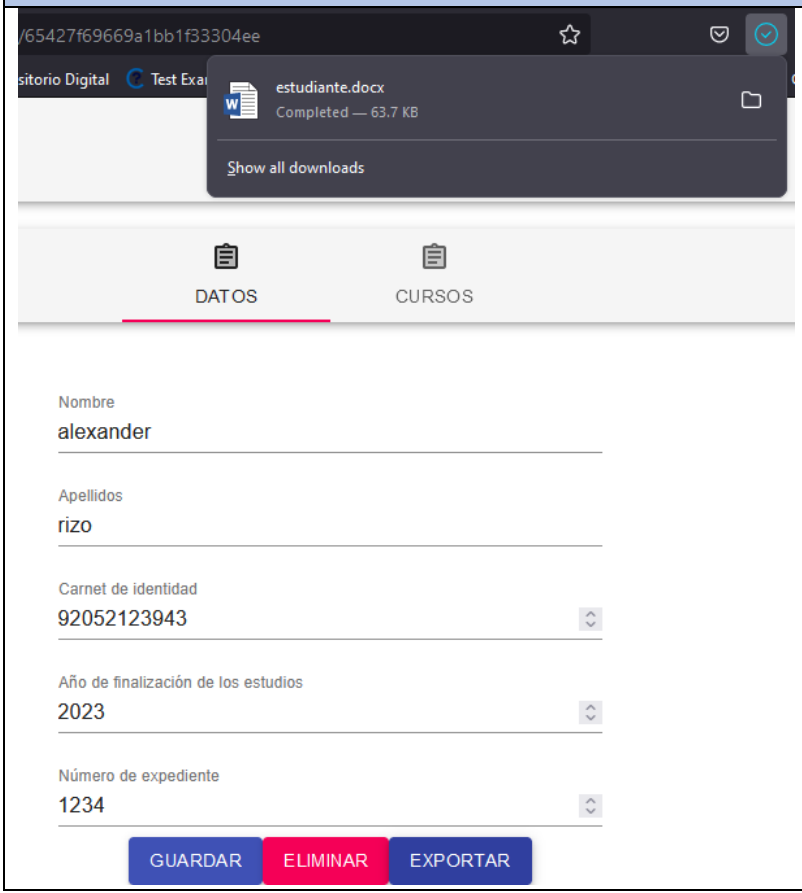
Sección 3: “Eliminar Estudiante”

Flujo básico Eliminar Estudiante

	Actor	Sistema
1.	Selecciona los estudiantes a eliminar y presiona la opción “Eliminar”.	
2.		Muestra el mensaje de confirmación “¿Está seguro que desea eliminar el(los) estudiante(s) seleccionado(s)?”. Y los botones “Eliminar” y “Cancelar”.
3.	Presiona el botón “Eliminar”.	
4.		Elimina el(los) estudiante(s) seleccionado(s) y emite el mensaje “Eliminación satisfactoria”. *Finalizando así el caso de uso.*

Prototipo elemental de interfaz gráfica		
Flujo Alterno		
2a. Cancelar eliminación de estudiante		
Actor		**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve al paso 2 del flujo básico “Gestionar estudiante”.
Sección 4: “Buscar Estudiante”		
Flujo básico Buscar Estudiante		
Actor		**Sistema**
1.	Inserta criterios de búsqueda: • ID, • nombre Y presiona el botón “Buscar”.	
2.		Verifica que los datos introducidos estén correctos.
3.		Muestra un listado con los estudiantes en el sistema que cumple(n) con el/los criterios(s) introducidos. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujo Alterno		
2a. Datos Incorrectos		
Actor		**Sistema**

1.		Muestra el mensaje “Debe introducir un criterio de búsqueda válido”.
3a. No hay estudiantes con el criterio de búsqueda especificado		
Actor		**Sistema**
1.		Muestra un listado en blanco.
Sección 5: “Exportar Certificación”		
Flujo básico Exportar Certificación		
Actor		**Sistema**
1.	Selecciona uno de los estudiantes mostrados, escoge la opción “Exportar certificación”.	
2.		Muestra el mensaje de confirmación “¿Desea exportar la certificación del estudiante seleccionado? Y los botones “Exportar” y “Cancelar”.
3.	Presiona el botón “Exportar”.	
4.		Exporta la certificación del estudiante seleccionado en formato PDF y emite el mensaje “Certificación exportada”.
Prototipo elemental de interfaz gráfica		

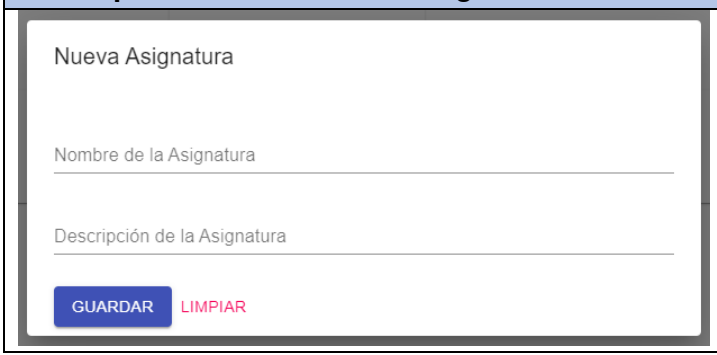


Flujo Alterno			
2a. Cancelar exportación de certificación			
Actor			**Sistema**
1.	Presiona el botón “Cancelar”.		
2.			Vuelve al paso 2 del flujo básico “Gestionar estudiante”.
Relaciones		**CU Incluidos**	No aplica
		CU Extendidos	No aplica
Requisitos no funcionales		No aplica	
Asuntos pendientes		No aplica	

Tabla 8. Descripción de caso de uso del sistema Gestionar Asignatura. Elaboración propia.

Objetivo		Permitir crear, eliminar o listar una asignatura.
Actores		Administrador o asesor
Resumen		El caso de uso se inicia cuando el administrador o el asesor decide crear, eliminar o listar una asignatura.
Complejidad		Media
Prioridad		Alta
Precondiciones		El administrador o asesor debe estar autenticado.
Postcondiciones		Se creó o eliminó una asignatura.
Flujo de eventos		
Flujo básico Gestionar asignatura		
Actor		**Sistema**
1.	Selecciona de la página principal la opción “Gestionar asignatura”.	
2.		Muestra una pantalla con las asignaturas existentes y las opciones de Crear o Eliminar asignatura(s).
3.	Desea crear o eliminar una asignatura.	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: 1. Si decide crear una asignatura, ir a la sección “Crear asignatura”. 2. Si decide eliminar asignatura(s), ir a la sección “Eliminar asignatura”.
Prototipo elemental de interfaz gráfica		

Flujos alternos		
2a. Listado de asignatura vacío		
Actor		**Sistema**
1.		Carga una pantalla en blanco sin asignaturas creadas.
Sección 1: "Crear asignatura"		
Flujo básico Crear asignatura		
Actor		**Sistema**
1.	Presiona el botón "Crear asignatura".	
2.		Muestra una ventana con los siguientes campos a introducir: • nombre, • descripcion Y el botón "Guardar".
3.	Introduce los datos y presiona el botón "Guardar".	
4.		Verifica que todos los campos estén llenos y que los datos introducidos estén correctos.
5.		Almacena los datos de los estudiantes y muestra el mensaje "Estudiante creado satisfactoria-mente". *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		



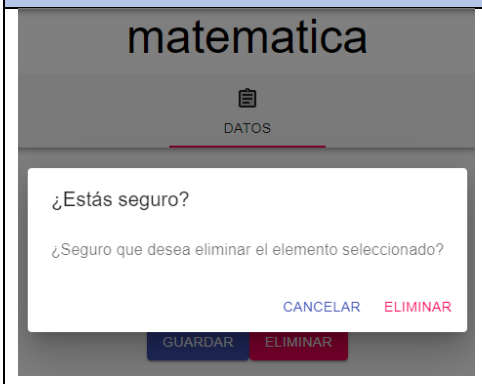
Flujos alternos		
4a. Campos vacíos y/o Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
Sección 2: “Eliminar Asignatura”		
Flujo básico Eliminar asignatura		
Actor		**Sistema**
1.	Selecciona la(s) asignatura(s) a eliminar y presiona la opción “Eliminar”.	
2.		Muestra el mensaje de confirmación “¿Está seguro que desea eliminar la(s) asignatura(s) seleccionada(s)?”. Y los botones “Eliminar” y “Cancelar”.
3.	Presiona el botón “Eliminar”.	
4.		Elimina la(s) asignatura(s) seleccionada(s) y emite el mensaje “Eliminación satisfactoria”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujo Alterno		
2a. Cancelar eliminación de asignatura		
Actor		**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve al paso 2 del flujo básico “Gestionar asignatura”.
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Tabla 9. Descripción de caso de uso del sistema Gestionar Resolución. Elaboración propia.

Objetivo	Permitir crear, modificar, eliminar, listar o buscar una resolución.

Actores	Administrador o asesor.
Resumen	El caso de uso se inicia cuando el administrador o el asesor decide crear, modificar, eliminar o buscar una resolución.
Complejidad	Alta
Prioridad	Alta
Precondiciones	El administrador o asesor debe estar autenticado.
Postcondiciones	Se creó, modificó, eliminó o buscó una resolución.
Flujo de eventos	
Flujo básico Gestionar resolución	

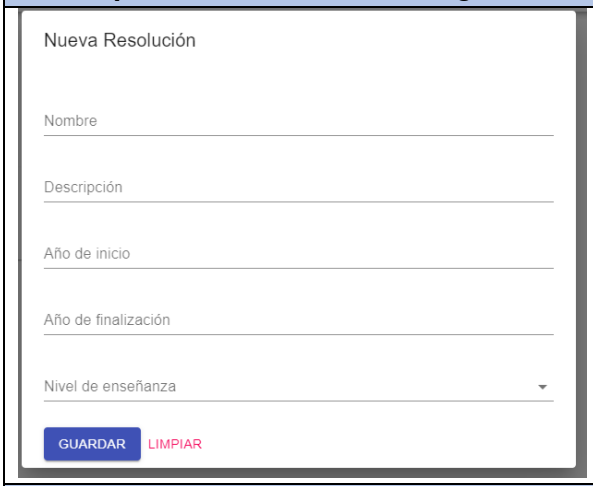
Actor		**Sistema**
1.	Selecciona de la página principal la opción “Gestionar resolución”.	
2.		Muestra una pantalla con las resoluciones existentes y las opciones de Crear, Modificar, Eliminar o Buscar una(s) resolución(es).
3.	Desea crear, modificar, eliminar o buscar una resolución.	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: 1. Si decide crear un estudiante, ir a la sección “Crear resolución”. 2. Si decide modificar un estudiante, ir a la sección “Modificar resolución”. 3. Si decide eliminar el estudiante, ir a la sección “Eliminar resolución”. 4. Si decide buscar un estudiante, ir a la sección “Buscar resolución”.

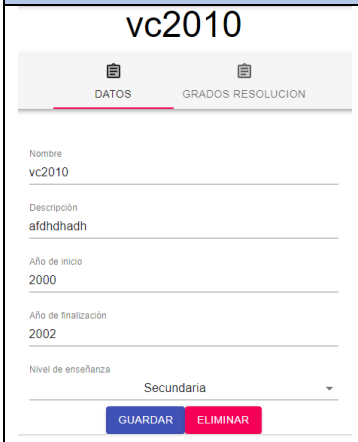
Prototipo elemental de interfaz gráfica

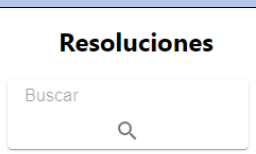
Flujos alternos

2a. Listado de resolución vacío

Actor	**Sistema**

1.		Carga una pantalla en blanco sin resoluciones creadas.
Sección 1: “Crear resolución”		
Flujo básico Crear resolución		
Actor		**Sistema**
1.	Presiona el botón “Crear resolución”.	
2.		Muestra una ventana con los siguientes campos a introducir: • nombre, • descripción, • año_inicio, • año_final, • enseñanza Y el botón “Guardar”.
3.	Introduce los datos y presiona el botón “Guardar”.	
4.		Verifica que todos los campos estén llenos y que los datos introducidos estén correctos.
5.		Almacena los datos de las resoluciones y muestra el mensaje “Resolución creada satisfactoriamente”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujos alternos		
4a. Campos vacíos y/o Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
Sección 2: “Modificar Resolución”		
Flujo básico Modificar Resolución		

Actor		Sistema
1.	Selecciona una de las resoluciones mostradas, escoge la opción “Modificar resolución”.	
2.		Muestra una ventana con los datos a modificar: • nombre, • descripción, • año_inicio, • año_final, • grados, • enseñanza Y el botón “Guardar”.
3.	Realiza las actualizaciones deseadas y presiona el botón “Guardar”.	
4.		Verifica que los datos introducidos estén correctos.
5.		Actualiza la información incorporada a la resolución y se emite un mensaje “Modificación satisfactoria”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Sección 3: “Eliminar Resolución”		
Flujo básico Eliminar Resolución		
Actor		**Sistema**
1.	Selecciona la(s) resoluciones a eliminar y presiona la opción “Eliminar”.	
2.		Muestra el mensaje de confirmación “¿Está seguro que desea eliminar la(s) resolución(es) seleccionada(s)?”. Y los botones “Cancelar” y “Eliminar”.
3.	Presiona el botón “Eliminar”.	

4.		Elimina la(s) resolución(es) seleccionada(s) y emite el mensaje “Eliminación satisfactoria”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
Flujo Alterno		
2a. Cancelar eliminación de resolución		
Actor		**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve al paso 2 del flujo básico “Gestionar resolución”.
Sección 4: “Buscar Resolución”		
Flujo básico Buscar Resolución		
Actor		**Sistema**
1.	Inserta criterios de búsqueda: • nombre Y presiona el botón “Buscar”.	
2.		Verifica que los datos introducidos estén correctos.
3.		Muestra un listado con las resoluciones en el sistema que cumple(n) con el/los criterios(s) introducidos. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
		
Flujo Alterno		
2a. Datos Incorrectos		
Actor		**Sistema**

1.		Muestra el mensaje “Debe introducir un criterio de búsqueda válido”.
3a. No hay resoluciones con el criterio de búsqueda especificado		
Actor		**Sistema**
1.		Muestra un listado en blanco.
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica
Requisitos no funcionales	No aplica	
Asuntos pendientes	No aplica	

Anexo 2: Acta de aceptación del cliente

Acta de Aceptación

En cumplimiento del convenio de colaboración con la dirección municipal de educación de Alquízar y en función de la ejecución del proyecto de tesis: Sistema para la gestión de certificación de estudios terminados en la dirección municipal de educación de Alquízar, se hace entrega del producto que se relaciona a continuación:

Sistema para la gestión de certificación de estudios terminados (GECET).

Entrega	Recibe
Nombre y Apellidos: Alexander Rizo Sánchez	Nombre y Apellidos: Lic. Ada María Sáinchez Mesa.
Cargo: Desarrollador	Cargo: Asesora Principal de Títulos y Diplomas
Firma:	Firma:

DIRECCION MUNICIPAL
DPTO. TITULOS Y DIPLOMAS
EDUCACION ALQUIZAR

Anexo 3: Aval del cliente

MINISTERIO
de EDUCACIÓN
REPÚBLICA DE CUBA

La Habana 3 de noviembre de 2023

A: *Decano de la Facultad de Ciencias y Tecnologías Computacionales de la Universidad de las Ciencias Informáticas.*

Por este medio comunicamos que el estudiante Alexander Rizo Sánchez, ha realizado su tesis de grado en la dirección municipal de educación de Alquízar. Su propuesta está basada en el "**Sistema para la gestión de certificación de estudios terminados (GECET)**", la cual ha cumplido con los requerimientos funcionales que fueron solicitados por la dirección, los resultados obtenidos en las pruebas realizadas, demuestran mejoría en los tiempos de respuesta a las solicitudes de certificación de estudios terminados; se evidencia, además, la posibilidad de resguardar en formato digital toda la información referente a los egresados de los diferentes centros educacionales del municipio que han cerrado definitivamente o han cambiado su nivel de enseñanza. Por lo tanto, la transformación ha cumplido con el objetivo principal de la dirección de agilizar el proceso de respuesta a las solicitudes de certificación de estudios terminados y el resguardo de la información relacionada con dicho proceso.

Sin otro asunto a tratar:

Lic. Ada María Sánchez Mesa
Asesora Municipal de Títulos y Diplomas

DIRECCION MUNICIPAL
DPTO. TITULOS Y DIPLOMAS
EDUCACION ALQUIZAR

Lic. Yanet Cruz Pulido
Directora Municipal de Educación