UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS-PARQUE CIENTÍFICO TECNOLÓGICO DE LA HABANA

Facultad de Ciencias y Tecnologías Computacionales

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

Título: Componente de control de acceso del Sistema para la gestión administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación (GAPID)

Autores: Josué Rivas Calderón

Cristian Guerra Luaces

Tutor: Dr.C. Arturo Orellana García, P.A.

La Habana, noviembre de 2023

Año 65 de la Revolución

DELARACIÓN DE AUTORÍA

Declaramos por este medio que Josué Rivas Calderón y Cristian Guerra Luaces, somos los autores del Trabajo de Diploma Componente de ¨Control de acceso del Sistema de Gestión Administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación¨ y que autorizamos al Proyecto de Incubación del PCT: ¨GAPID: Sistema para la gestión administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación¨ los derechos patrimoniales con carácter exclusivo.

Y para que así conste, se firmó la presente declaración de autoría en La Habana a los __ noviembre de 2023.

Josué Rivas Calderón

_____
Firma del Autor

Cristian Guerra Luaces

_____
Firma del Autor

Dr.C. Arturo Orellana García, P.A.

_____
Firma del Tutor

DATOS DE CONTACTO

Dr.C. Arturo Orellana García: graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2012. Se desempeña como líder del Grupo de Investigación de Minería de procesos y Asesor de Capacitación, Desarrollo e Investigación del Centro de Soluciones de Informática Médica. Ha liderado proyectos I+D+i de desarrollo de componentes de software a partir de minería de procesos para el análisis de procesos de negocio del entorno hospitalario. Investiga la Ingeniería de comportamiento, la medicina de precisión y el procesamiento de imágenes médicas. Tutora varias tesis de grado, maestrías y doctorados enfocados al análisis de procesos de negocio, la informática médica y otras áreas del conocimiento. Obtuvo el grado de Máster en Informática Aplicada en 2015 desarrollando una herramienta informática basada en técnicas de minería de procesos para identificar problemas en la ejecución de procesos de negocio. Doctor en Ciencias Técnicas desde 2016 presentando un modelo computacional para la detección de variabilidad en procesos de negocio del entorno sanitario aplicando minería de procesos. Correo electrónico: aorellana@uci.cu

«“. ¿Y qué juventud queremos? ¿Queremos, acaso, una juventud

que simplemente se concrete a oír y a repetir?

¡No! Queremos una juventud que piense(...) una juventud

que aprende por si misma a ser revolucionaria,

una juventud que desarrolle

plenamente su pensamiento».

Fidel Castro Ruz

DEDICATORIA

Por parte de Josué:

Quiero dedicar este trabajo de diploma a mi familia en general, por estar siempre presentes y brindarme su aliento y su confianza.

Por parte de Cristian:

Quiero dedicar este trabajo de diploma especialmente a mi madre, por estar siempre presente y brindarme su aliento y su confianza.

AGRADECIMIENTOS

Por parte de Josué:

Quiero aprovechar esta oportunidad para expresar mi más profundo agradecimiento a mi familia, especialmente a mi madre y a mis dos hermanas, por su apoyo incondicional, su comprensión y su amor durante todo el proceso de elaboración de esta tesis. A mis compañeros de la universidad con quienes compartí momentos de aprendizaje, de debate y de amistad, a Arturo por su excelente colaboración como tutor. Sin ellos, nada de esto hubiera sido posible.

Por parte de Cristian:

Quiero aprovechar esta oportunidad para expresar mi más profundo agradecimiento a mi familia, especialmente a mi madre, por su apoyo incondicional, su comprensión, su apoyo y su amor durante todo el proceso de elaboración de esta tesis. A mis compañeros de la universidad con quienes compartí momentos de aprendizaje, de debate y de amistad, a Arturo por su excelente colaboración como tutor. Sin ellos, nada de esto hubiera sido posible.

RESUMEN

En el Parque Científico Tecnológico de la Habana se desarrolla el Sistema de Gestión Administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación (GAPID) con el propósito de informatizar la gestión de la información en el CITMA. Este sistema consta de diversos módulos, como Gestión de Programas, Proyectos y Recursos Humanos, Certificación de Actividades/Resultados, Gestión Documental, entre otros. Sin embargo, el sistema está expuesto a vulnerabilidades, que comprometen la seguridad de la información y permiten a usuarios acceder a datos que no les corresponden. El objetivo general del proyecto es desarrollar un componente de control de acceso en GAPID para mejorar la gestión de usuarios, roles y permisos. Los métodos teóricos aplicados incluyen el enfoque histórico-lógico para establecer antecedentes y la actualidad de proyectos de investigación, el método analítico-sintético para comparar sistemas existentes, el inductivo-deductivo para definir la problemática y la modelación para crear artefactos ingenieriles. En cuanto a los métodos empíricos, se realizó un análisis de antecedentes para estudiar plataformas relacionadas con la gestión de información de proyectos de investigación. Se realizaron entrevistas a participantes clave en el sistema GAPID para recopilar información sobre gestión de usuarios, roles y permisos, y se ejecutó un análisis de documentos para revisar manuales, políticas y registros de usuarios. El resultado del proyecto es un componente de control de acceso que contribuye a una correcta gestión de usuarios, roles y permisos en GAPID, mejorando la seguridad y eliminando las vulnerabilidades del sistema.

Palabras clave: sistema GAPID, componente de control de acceso, gestión, seguridad, vulnerabilidad.

ABSTRACT

In the Scientific and Technological Park of Havana, the Administrative Management System for Science, Technology and Innovation Programs and Projects (GAPID) is developed with the purpose of computerizing information management at CITMA. This system consists of various modules, such as Program Management, Projects and Human Resources, Certification of Activities/Results, Document Management, among others. However, the system is exposed to vulnerabilities, which compromise the security of the information and allow users to access data that does not belong to them. The general objective of the project is to develop an access control component in GAPID to improve the management of users, roles and permissions. The theoretical methods applied include the historical-logical approach to establish the background and current nature of research projects, the analytical-synthetic method to compare existing systems, the inductive-deductive method to define the problem and modeling to create engineering artifacts. Regarding empirical methods, a background analysis was carried out to study platforms related to information management of research projects. Interviews were conducted with key participants in the GAPID system to collect information on user management, roles and permissions, and a document analysis was executed to review manuals, policies and user records. The result of the project is an access control component that contributes to the correct management of users, roles and permissions in GAPID, improving security and eliminating system vulnerabilities.

Keywords: GAPID system, access control component, management, security, vulnerabilities

ÍNDICE DE CONTENIDOS

Introducción 1
CAPÍTULO 1: Fundamentos teóricos metodológicos de la investigación 6
1.1 Marco Conceptual de la investigación 6
1.2 Análisis de las indicaciones metodológicas del sistema de programas y proyectos 8
1.3 Sistema GAPID 10
1.4 Análisis de soluciones similares 12
1.4.1 Trello 12
1.4.2 Asana 14
1.4.3 Akademos 15
1.4.4 GESPRO 16
1.4.5 Análisis comparativos de los sistemas homólogos 17
1.5 Ambiente de desarrollo 18
1.5.1 Herramientas 18
1.5.2 Sistemas gestores de bases de datos (SGBD) 20
1.5.3 Lenguaje de programación 21
1.5.4 Metodología de desarrollo de software 22
1.6 Conclusiones del capítulo 24
CAPÍTULO 2: Características del componente de control de accesos 25
2.1 Características de la propuesta de solución 25
2.2 Modelado del dominio 26
2.3 Requisitos del Sistema 28
2.3.1 Requisitos Funcionales 28
2.3.2 Requisitos No Funcionales 29
2.4 Historia de Usuario 30
2.5 Arquitectura de software 33
2.6 Patrones de diseño 35

2.7 Modelo de datos 38
2.8 Conclusiones del capítulo 41
CAPÍTULO 3: Implementación y pruebas.....41
3.1 Seguridad informática 42
3.2 Estándares de codificación 42
3.3 Pruebas de Software.....44
3.3.1 Tipos de prueba 44
3.3.2 Estrategia de Evaluación de Software.....44
3.3.3 Método de caja blanca. Técnica de camino básico 45
3.3.4 Método de caja negra. Técnica de partición equivalente.....48
3.4 Conclusiones del capítulo 50
Conclusiones generales.....52
Recomendaciones 53
Referencias bibliográficas 54
Anexos.....58
Anexo 1. Interfaz de autenticación del sistema GAPID 58
Anexo 2. Interfaz de los participantes con sus usuarios. 58
Anexo 3. Listado de programas dentro del sistema.....59
Anexo 4. Gestión administrativa de un programa.....59
Anexo 5. Listado de proyectos dentro un programa. 60
Anexo 6. Gestión administrativa de un proyecto. 60
Anexo 7. Interfaz del listado general de participantes. 61

ÍNDICE DE TABLAS

Tabla 1. Análisis comparativo de sistemas homólogos. Fuente: Elaboración propia. 17
Tabla 2. Fases de la variación AUP UCI. Fuente:(31) 22
Tabla 3. Roles de los usuarios. Fuente: Elaboración propia.....27
Tabla 4. Requisitos funcionales. Fuente: Elaboración propia. 28
Tabla 5. Historia de usuario No.1. Fuente: Elaboración propia.....31
Tabla 6. Historia de usuario No.2. Fuente: elaboración propia. 32
Tabla 7. Casos de prueba. Fuente: Elaboración propia.....47
Tabla 8. Resultados método caja negra mediante la técnica de partición equivalente en las pruebas de integración. Fuente: Elaboración propia 49
Tabla 9. Resultados del método caja negra mediante la técnica partición equivalente en las pruebas de aceptación. Fuente: Elaboración propia. 50

ÍNDICE DE FIGURAS

Figura 1. Interfaz de Trello. Fuente:(16) 13
Figura 2. Interfaz de Asana. Fuente:(16) 14
Figura 3. Interfaz de Akademos. Fuente:(17) 15
Figura 4. Interfaz de GESPRO. Fuente:(18) 16
Figura 5. Modelo de dominio. Fuente: Elaboración propia. 26
Figura 6. Modelo-Vista-Plantilla. Fuente:(37) 34
Figura 7. Código de la clase registro. Fuente: Elaboración propia 38
Figura 8. Modelo de datos. Fuente: Elaboración propia. 39
Figura 9. Relación entre la tabla authentication y participantes. Fuente: Elaboración propia 40
Figura 10. Relación entre la tabla la tabla rol y auth_permission. Fuente: Elaboración propia. 40
Figura 11. Método caja blanca. Fuente:(47) 45
Figura 12. Fragmento de código de la clase registro. Fuente: Elaboración propia. 46
Figura 13. Grafo de flujo. Fuente: Elaboración propia 46
Figura 14. Método de caja negra. Fuente:(47) 48
Figura 15. Autenticación de usuario. Fuente: Elaboración propia. 58
Figura 16. Listado de participantes con usuarios. Fuente: Elaboración propia. 58
Figura 17. Listado de programas. Fuente: Elaboración propia. 59
Figura 18. Gestión Administrativa de un programa. Fuente: Elaboración propia. 59
Figura 19. Listado de proyectos dentro de un programa. Fuente: Elaboración propia. 60
Figura 20. Gestión administrativa de un proyecto. Fuente: Elaboración propia. 60
Figura 21. Listado general de participantes. Fuente: Elaboración propia. 61

Introducción

El desarrollo económico, científico y técnico impone nuevas metas a la sociedad. Avances vertiginosos de la ciencia, en especial los que responden a las Tecnologías de la Información y las Comunicaciones (TIC), definen la actual era tecnológica. El auge que ha tomado la informatización de los procesos involucrados en las distintas esferas de la sociedad apuesta por la inclusión de mejores métodos con el uso de la informática y las comunicaciones para responder las necesidades existentes. (1)

Como resultado de este auge en Estados Unidos se lleva a cabo El *Project Management Institute* (PMI), una organización sin fines de lucro que asocia a profesionales relacionados con la Gestión de Proyectos, y es la más grande del mundo en su área. Esta manifiesta que: un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. Temporario porque tiene un comienzo y un fin definido; y único porque el producto o servicio es diferente de alguna manera que lo distingue de otros productos o servicios. (2)

En América Latina el desarrollo de la ciencia, la tecnología y la innovación se han dado fundamentalmente de la mano de los Estados, por lo que las iniciativas gubernamentales han tenido un papel protagónico en la región. Las políticas de ciencia y tecnología han sido impulsadas desde iniciativas gubernamentales y la academia ha jugado un papel protagónico. Esto le otorga una carga escolarizada a las regulaciones que siguen los investigadores que generalmente comparten la docencia con la actividad científica.

América Latina y el Caribe, como espacio de interacción entre sujetos, está estructurada por un conjunto de prácticas determinadas desde las políticas y las agendas simbólicas de investigación. Las inversiones en I+D en la región se concentran en Brasil, México y Argentina, y se enfocan fundamentalmente en las llamadas ciencias duras. Las ciencias sociales y humanísticas ocupan un segundo plano dentro de la asignación de recursos en la investigación. Consecuentemente, los estudios sociales acerca del desarrollo comunitario son relegados e integrados a temáticas donde se abordan, generalmente, de forma transversal. (3-5)

Cuba desarrolla una política basada en la ciencia y luego en la transformación digital, lo que ha provocado que se creen programas para la ejecución de proyectos y en este sentido las universidades cubanas juegan un papel fundamental. Por parte del Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA) se realizan convocatorias a proyectos Nacionales, Sectoriales e Institucionales para favorecer el desarrollo científico del país. (6)

El Sistema de Programas y Proyectos de Ciencia, Tecnología e Innovación (SPP) constituye la forma organizativa fundamental para la ejecución de las Actividades de Ciencia, Tecnología e Innovación (ACTI), y se expresa a través de los programas y proyectos incluidos en el Plan de Ciencia, Tecnología e Innovación del país, integrado al Plan de la Economía Nacional en todos sus niveles de organización.

Un Programa de Ciencia, Tecnología e Innovación, es un conjunto de actividades diversas de ciencia, tecnología e innovación, organizadas en proyectos que se relacionan entre sí, cuyo objetivo es resolver de forma integral, un problema identificado en las prioridades a su nivel, dirigido a lograr resultados de impactos específicos en un período determinado. Las relaciones entre las entidades que participan en la ejecución de los programas y proyectos, así como con los clientes y usuarios de sus resultados, quedan refrendadas en contratos económicos firmados por las partes a tenor de las normas establecidas en la legislación vigente en el país. (7)

Los Proyectos de Ciencia, Tecnología e Innovación, constituyen la forma organizativa fundamental, con carácter temporal, para la planificación, ejecución, financiamiento, evaluación y control de las actividades y tareas de investigación, desarrollo e innovación con la finalidad de materializar objetivos concretos, obtener resultados de impacto y contribuir a la solución del problema que determine su puesta en ejecución, sea propio o del programa en el que están insertados. (6)

Las indicaciones metodológicas para los proyectos de I+D+i requiere de la elaboración de varios documentos y anexos para hacer efectiva la certificación de resultados en determinado periodo y el control de la ejecución de los proyectos. Para confeccionar el expediente se utilizan los documentos Word y Excel, los cuales generan un cúmulo de información dispersa y extensa, difícil de gestionar. El Jefe de Proyecto asume actividades de gestor, económico, planificador, revisor y documentador lo que provoca que su actividad científico técnica y guía dentro del proyecto se afecte por el tiempo a dedicar a otras tareas administrativas.

Es por este motivo que, en los avances tecnológicos, es fundamental que los sistemas informáticos se mantengan actualizados y se adapten a las necesidades cambiantes de las organizaciones. Para ello en el Parque Científico de la Habana se desarrolla un software llamado Sistema de Gestión Administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación(GAPID) el cual pretende informatizar o facilitar la gestión de la información de programas y proyectos del CITMA, donde se han desarrollado módulos de Gestión de Programas, Proyectos y Recursos humanos, Certificación de actividades/resultados, Gestión

documental de programas y proyectos, Gestión del Conocimiento, Planificación de Resultados, Gestión económica y Control.

Los distintos módulos del sistema presentan una variedad de niveles de acceso, privilegios y permisos para los usuarios. En particular, el jefe de programa cuenta con la capacidad de modificar la información dentro de su programa y revisar los detalles de los proyectos asociados a dicho programa. Por otro lado, el jefe de proyecto tiene la autorización para gestionar y modificar la información específica de su proyecto, pero no tiene acceso a otros proyectos ni al programa en general.

Al no contar con una gestión más detallada de estos accesos plantea un riesgo significativo para la seguridad del sistema. La falta de restricciones precisas podría llevar a situaciones de vulnerabilidad, donde la información confidencial podría estar expuesta. Además, existe el riesgo de que otros usuarios accedan a información que no les corresponde, lo cual compromete la integridad y privacidad de los datos. En resumen, la implementación de un sistema más robusto de control de accesos se vuelve crucial para salvaguardar la integridad y confidencialidad de la información en el sistema.

Problema de investigación: ¿Cómo garantizar el correcto nivel de acceso a la información, módulos y funcionalidades del sistema GAPID?

Objeto de estudio: El proceso de control de acceso en sistemas de gestión de la información.

Campo de acción: El control de acceso en sistemas informáticos de gestión de proyectos.

Objetivo general: Desarrollar un componente de control de acceso en el sistema GAPID que mejore la gestión de usuarios, roles y permisos.

Tareas de Investigación:

1. Elaboración del marco teórico-metodológico de la investigación relacionado a los programas y proyectos de ciencia, tecnología e innovación, así como el control de acceso a la información que se gestiona en los mismos.
2. Definición y caracterización de los elementos relacionados a la correcta gestión de usuarios, roles y permisos en un sistema de gestión.
3. Desarrollo de un componente de control acceso para una correcta gestión de usuarios, roles y permisos del sistema GAPID.

4. Integración del componente de control de acceso para la correcta gestión de usuarios, roles y permisos al sistema GAPID.
5. Validación de los resultados obtenidos a partir de pruebas de software.

Para dar cumplimiento a las tareas de investigación se utilizaron los siguientes métodos científicos:

Métodos teóricos:

- Histórico - lógico: Fue esencial para analizar los antecedentes y actualidad del control de acceso en sistemas similares en el ámbito científico y tecnológico. Se exploraron los antecedentes de soluciones implementadas previamente en el Parque y otras instituciones afines, lo que permite comprender la progresión temporal y lógica de los enfoques adoptados en la gestión de acceso.
- Analítico - Sintético: La aplicación de este método resultó crucial para realizar un análisis exhaustivo de los sistemas de control de acceso existentes en el entorno del Parque Científico Tecnológico de la Habana. Se desglosaron las características y funcionalidades de cada sistema, lo que facilita la identificación de elementos claves que podrían integrarse en el diseño del componente de control de acceso, así como la síntesis de un enfoque integral y eficiente.
- Inductivo - Deductivo: A partir del estudio detallado de los procesos de gestión de información y acceso en proyectos de investigación dentro del Parque, se adoptó un enfoque inductivo para identificar patrones y problemáticas específicas. Luego, mediante un enfoque deductivo, se derivaron soluciones específicas para abordar las necesidades particulares identificadas. Este método permitió una adaptación precisa a los requerimientos y desafíos únicos del entorno.

Métodos empíricos:

- Análisis de antecedentes: Este enfoque consistió en examinar detalladamente las plataformas y sistemas existentes que guardan relación con la gestión de información de proyectos de investigación. Se llevaron a cabo evaluaciones exhaustivas de soluciones previas implementadas en el mismo contexto, se identificaron sus fortalezas y limitaciones en cuanto al control de acceso.

- Entrevista: Consiste en seleccionar participantes clave relacionados con el sistema GAPID, formular preguntas específicas, ejecutar entrevistas para recopilar información

sobre la gestión de usuarios, roles y permisos, analizar los datos para identificar patrones y problemas, y finalmente, presentar conclusiones y recomendaciones basadas en los hallazgos.

- Análisis de documentos: Implica la revisión y examen detallado de documentos relevantes, como manuales, políticas, registros de usuarios, y otros materiales escritos relacionados con la configuración del sistema. A través de este proceso, se recopilan datos y se identifican tendencias, regulaciones y problemas existentes en la gestión de usuarios, roles y permisos. Estos hallazgos documentados pueden servir como base para la toma de decisiones y la mejora de las prácticas de gestión de usuarios, roles y permisos en el sistema.

La investigación estará estructurada en 3 capítulos como se muestra a continuación:

Capítulo 1: Fundamentos teóricos metodológicos de la investigación. En el desarrollo de este capítulo se abarca lo relacionado con la fundamentación teórica que sustenta la presente investigación, se expondrán los principales conceptos, tendencias, metodologías y herramientas del sistema y toda la información referente.

Capítulo 2: Características del componente de control de accesos. En este capítulo se seleccionaron los requerimientos del sistema que se desean desarrollar. Se realizó una descripción con sistema en la que se enmarca el problema para dar paso a la creación de un entorno conceptual asociado a la información.

Capítulo 3: Implementación y pruebas. Este capítulo presenta las estrategias de pruebas y se evidencian las pruebas realizadas al sistema para la comprobación de su correcto funcionamiento y que cumpla con el objetivo general de la investigación.

CAPÍTULO 1: Fundamentos teóricos metodológicos de la investigación

En el capítulo de Fundamentación Teórica se presenta una revisión exhaustiva de los conceptos, teorías y estudios previos relacionados con el tema de la investigación. Se exploran los fundamentos teóricos y las bases conceptuales que respaldan el desarrollo del componente de control de acceso para el sistema GAPID. Esta sección proporciona el marco teórico necesario para comprender el contexto y la relevancia del trabajo, y así establecer las bases para el desarrollo y la justificación de la propuesta.

1.1 Marco Conceptual de la investigación

En este apartado, se exploran los conceptos clave asociados al problema abordado en el proyecto del componente de control de acceso del Sistema GAPID. Estos conceptos proporcionan una comprensión del contexto y la naturaleza del proyecto. A continuación, se definen y explican brevemente algunos de estos conceptos:

Control de acceso

El control de acceso es un elemento esencial de la seguridad que determina quién tiene permiso para tener acceso a determinados datos, aplicaciones y recursos y en qué circunstancias. Permite que la persona adecuada entre y que la que no lo es se quede fuera. Las directivas de control de acceso dependen en gran medida de técnicas como la autenticación y la autorización, que permiten a las organizaciones verificar de forma explícita que los usuarios son quienes dicen ser y que cuentan con el nivel adecuado de acceso con base en elementos contextuales como el dispositivo, la ubicación, el rol y mucho más. (8)

Proyecto

Entonces, un proyecto es la ideación de una tarea determinada, para la cual establecemos el modo en el que se va a realizar, de esta forma en el proyecto se debe recoger una planificación del conjunto de actividades, así como la forma de llevarlas a cabo. Por último, el proyecto también debe incluir el detalle del conjunto de recursos y medios necesarios para llevarlo a cabo. Un proyecto puede ser de una infinidad de tipos. No debe mantener siempre una estructura concreta. (9)

Gestión de proyectos

Este enfoque metódico se orienta en la estimación, administración y cumplimiento de los objetivos específicos, medibles, alcanzables y realistas para la realización de tareas dentro de una organización. (10)

Sus objetivos son claros:

- ✓ Gestionar el arranque y evolución de los proyectos
- ✓ Administrar y resolver problemas que puedan suscitarse durante el proceso;
- ✓ Facilitar las tareas de finalización y aprobación del proyecto.

Gestión de usuarios

La gestión de usuarios es una forma de añadir y gestionar usuarios de su cuenta en un sistema determinado. Sirve también para determinar quién puede acceder a su cuenta y cuáles son las funciones que tienen a su disposición. (11,12)

Permisos por roles

Un rol es un conjunto de permisos. Normalmente, los permisos de un rol definen una actividad concreta que puede realizar un usuario. Un usuario puede tener asignado más de un rol. El número de roles y permisos depende del usuario. (13)

Configuración

En informática, la configuración es un conjunto de datos que determina el valor de algunas variables de un programa informático o de un sistema operativo. Estas opciones generalmente se cargan durante el inicio del programa y en algunos casos es necesario reiniciarlo para poder ver los cambios. Es habitual que los programas utilicen ficheros para guardar su configuración, pero también puede almacenarse en una base de datos. También se pueden encontrar programas que cifran su configuración para evitar que los usuarios la modifiquen. (14)

Administración de sistema

La administración de sistemas es el trabajo realizado por expertos en tecnología de la información (TI) para una organización. Su trabajo es garantizar que los sistemas informáticos y todos los servicios relacionados funcionen bien. (15)

1.2 Análisis de las indicaciones metodológicas del sistema de programas y proyectos

El organismo o entidad que gestiona el programa tiene como funciones el diseño, la planificación y control de este, tanto desde el punto de vista técnico como financiero; su jefe máximo evalúa al Jefe de Programa y tiene en cuenta el criterio del organismo o entidad que lo dirige; nombra al personal designado para ejercer la función de Expertos y contrata a los evaluadores, cuando sea necesario. (7)

El Equipo de Dirección del Programa es la estructura encargada de coordinar y controlar la ejecución del Programa y está constituido por un Jefe, un Secretario Ejecutivo y el Grupo de Expertos. A continuación, se identificaron las responsabilidades que influyen en el desarrollo de la propuesta de solución.

El Equipo de Dirección del Programa, a cualquier nivel, tiene las responsabilidades siguientes:

- ✓ Seleccionar los Proyectos que integran la carpeta del Programa, para ello tiene en cuenta los criterios de selección que se establezcan.
- ✓ Controlar y evaluar la ejecución de los Proyectos, calidad, novedad y pertinencia de los resultados.
- ✓ Determinar la detención, cancelación o modificación de los Proyectos.
- ✓ Participar en todo tipo de evaluación del Programa.
- ✓ Elaborar los informes sobre la ejecución del Programa.
- ✓ Participar en la actualización de los objetivos del Programa, según las necesidades y demandas para el desarrollo.

El Jefe y el Secretario Ejecutivo del Programa, tienen las siguientes funciones comunes:

- ✓ Seleccionar los evaluadores, pertenecientes al Grupo de Expertos o externos a este, que realizan la evaluación de los proyectos en cualquiera de sus etapas.
- ✓ Convocar al Grupo de Expertos, organizar sus actividades y evaluar su trabajo; elaborar las actas de las reuniones, los dictámenes y controlar el cumplimiento de sus acuerdos.
- ✓ Velar por el cumplimiento de los resultados, las tareas o actividades previstas y otras obligaciones establecidas en el contrato con respecto al financiamiento.
- ✓ Evaluar al jefe de cada Proyecto cuando se requiera.

- ✓ Elaborar un cronograma de actividades de gestión del Programa que tome en cuenta la interrelación entre los diferentes Proyectos y sus resultados; así como las acciones de evaluación y control de sus expertos.

El Jefe del Programa, además de sus responsabilidades como integrante del Equipo de Dirección, tiene las siguientes:

- ✓ Controlar el Expediente Único del Programa.

El Secretario Ejecutivo del Programa, además de sus responsabilidades como integrante del Equipo de Dirección, tiene las siguientes:

- ✓ Mantener actualizado el sistema de información establecido para los Programas;
- ✓ Convocar a la conciliación de los Proyectos con los ejecutores y orientar la conformación del contrato.
- ✓ Dirigir el proceso de evaluación de los Proyectos: ex ante, durante y final; con los expertos y evaluadores que participan en el mismo.
- ✓ Archivar las propuestas de Proyectos que no pasan el proceso de evaluación ex ante, con la documentación y una nota que da por cerrado el proceso.
- ✓ Elaborar los dictámenes de aprobación de las evaluaciones de los proyectos por el Grupo de Expertos en las diferentes etapas: ex ante, durante y final, y enviarlos a los Jefes de Proyectos.
- ✓ Recepcionar y tramitar los Informes Científico Técnicos parciales o finales de los resultados y otros reportes de investigación para su evaluación.
- ✓ Certificar, trimestral o semestralmente, el cumplimiento de los resultados, tareas o actividades aprobadas en el Grupo de Expertos y de las obligaciones establecidas en el contrato con respecto al financiamiento, y tener en cuenta la planificación prevista.
- ✓ Elaborar la documentación para la compatibilización de los proyectos que integran el Programa con los intereses de la Defensa.

El Grupo de Expertos, además de sus responsabilidades como integrante del Equipo de Dirección del Programa, tiene las funciones siguientes:

- ✓ Evaluar, durante la ejecución de los Proyectos, el comportamiento de las soluciones, métodos y vías adoptados para alcanzar los objetivos planteados.

- ✓ Participar en el control de la ejecución de los Proyectos, dictaminar los resultados obtenidos y proponer las medidas necesarias para aumentar su calidad y efectividad técnico económica.

1.3 Sistema GAPID

En el contexto de las crecientes políticas basadas en la ciencia y la transformación digital en Cuba, se ha generado la necesidad de establecer programas y proyectos para impulsar el desarrollo científico del país. La aprobación del Decreto Ley 7/2020 del Sistema de Ciencia, Tecnología e Innovación y la Resolución 287/2019 Reglamento para el Sistema de Programas y Proyectos de Ciencia, Tecnología e Innovación, estableció la base para la creación de las Indicaciones Metodológicas para la Actividad del Sistema de Programas y Proyectos de Ciencia, Tecnología e Innovación.

Bajo este marco normativo, surge la iniciativa de desarrollar el Sistema de Gestión Administrativa de Programas y Proyectos de Ciencia, Tecnología e Innovación (GAPID) en el Parque Científico Tecnológico de la Habana. Este sistema busca optimizar la ejecución de proyectos al proporcionar una plataforma electrónica que simplifica la gestión administrativa, financiera y documental. La complejidad de la gestión actual, basada en documentos Word y Excel, genera inconvenientes en la compilación de información, la certificación de actividades y la evaluación de resultados.

La falta de una solución integral específica para el control de programas y proyectos dentro del entorno científico y tecnológico motivó la búsqueda de sistemas existentes en el ámbito nacional. Sin embargo, los sistemas identificados, como GESPRO, no cumple completamente con los requisitos particulares del Sistema de Programas y Proyectos de Ciencia, Tecnología e Innovación.

La necesidad de una solución a medida que abarque la gestión de programas, proyectos, participantes y la generación automática de documentos asociados a certificaciones y evaluaciones, junto con la consideración de las actividades científico-técnicas y los presupuestos, justifica la creación del Sistema GAPID en el Parque Científico Tecnológico de la Habana. Este sistema busca superar las limitaciones actuales, mejorar la eficiencia en la gestión de la información y fortalecer la ejecución de proyectos de I+D+i en consonancia con las directrices establecidas por las autoridades competentes.

Objetivos específicos:

- ✓ Diseñar e implementar los flujos asociados a la gestión de Programas, Proyectos y Recursos humanos.
- ✓ Desarrollar el módulo de certificación de actividades/resultados.
- ✓ Desarrollar el módulo de gestión documental de programas y proyectos.
- ✓ Desarrollar el módulo de gestión del conocimiento.
- ✓ Desarrollar el módulo de Planificación de Resultados.
- ✓ Desarrollar el módulo Gestión económica.
- ✓ Desarrollar el Módulo de Control.
- ✓ Desarrollar funcionalidades para la evaluación de proyectos por los expertos.
- ✓ Desplegar la solución para su explotación por programas y proyectos.
- ✓ Brindar soporte técnico a los programas y proyectos.
- ✓ Propiciar la formación de recursos humanos en las carreras de Ingeniería en Ciencias Informáticas y Ciberseguridad.

Alcances de la propuesta

La propuesta comprende el desarrollo general de un sistema para la gestión administrativa

de Programas y Proyectos de Ciencia, Tecnología e Innovación.

1. Gestión integral de los Programas, Proyectos y Recursos humanos a partir de las indicaciones metodológicas y resoluciones del CITMA vigentes en la Gaceta Oficial de la República de Cuba.

2. Certificación de Actividades/Resultados desde el sistema, con la posibilidad de generar Evaluaciones, Actas de Conformidad, Remuneración de Participantes, Remuneración del Jefe de Proyecto, Remuneración del Grupo de Expertos, Secretario y Jefe de Programa.

3. Gestión de la documentación general de programas y proyectos, además de la que se genera en cada período de certificación de Actividades/Resultados, para propiciar un repositorio centralizado que facilita el control del expediente único de los proyectos.

4. Gestión del conocimiento de los resultados obtenidos a partir de las indicaciones metodológicas del CITMA.

Resultados e impactos esperados

La propuesta se resume en 7 Resultados principales, para lograrlo se identifican estudiantes de la Carrera de Ingeniería en Ciencias Informáticas y expertos del Sistema de programas y proyectos de Ciencia, Tecnología e Innovación.

R1. Módulo de Gestión de Programas, Proyectos y Recursos humanos.

R2. Módulo de certificación de actividades/resultados.

R3. Módulo de Gestión documental de programas y proyectos.

R4. Módulo de Gestión del Conocimiento.

R5 Módulo de Planificación de Resultados.

R6. Módulo de Gestión económica.

R7. Módulo de Control.

La puesta en práctica de este tipo de solución propicia que los usuarios lleguen a ser más productivos; su eficiencia operativa también aumenta. Mejora la experiencia del cliente al digitalizar el proceso de gestión integral del sistema de Programas y Proyectos.

La solución disminuye los tiempos necesarios para la elaboración de la documentación y gestión de resultados asociados a un proyecto o programa. Disminuye la probabilidad de ocurrencia de errores en el traspaso de información entre documentos Word y Excel. Propicia la centralización de expedientes de proyecto para su gestión directa por los programas. Propicia el ahorro por concepto de gastos de TI y contratación de soluciones complejas. Propicia obtener una solución a la medida de las necesidades de los usuarios, con la posibilidad de ajustar o ampliar servicios toda vez que sean requeridos. Se obtiene una solución extensible y escalable, que permite la incorporación de nuevas funcionalidades a partir de resoluciones y leyes que surjan.

1.4 Análisis de soluciones similares

1.4.1 Trello

Trello es un software de gestión de proyectos online, intuitivo y fácil de usar, basado en la metodología Kanban. Su enfoque visual permite organizar y hacer seguimiento de tareas y proyectos de manera eficiente, lo que lo convierte en una herramienta popular para equipos de cualquier tamaño. (16)

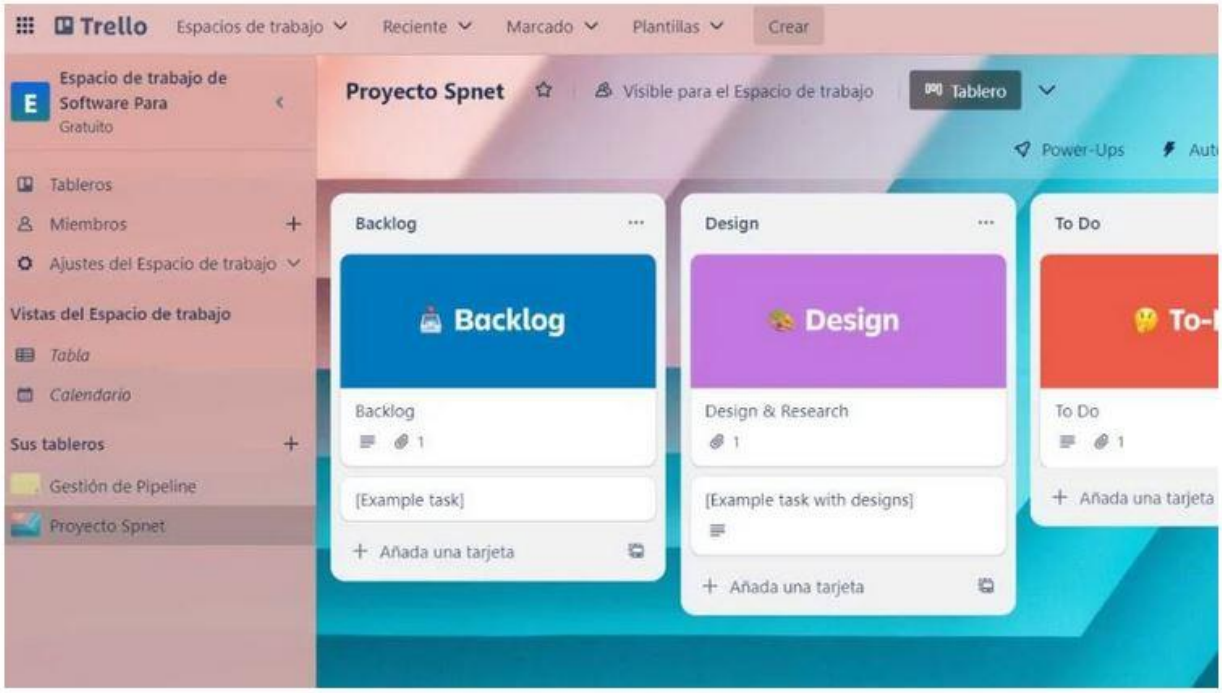


Figura 1. Interfaz de Trello. Fuente:(16)

1. **Organización visual para el gerente de proyecto y su equipo**: Con los tableros y tarjetas personalizables, podéis visualizar las tareas de manera clara y rápida, al entender las responsabilidades y prioridades en cada proyecto.
2. **Mantén un seguimiento del avance**: Gracias a los flujos de trabajo personalizados y las listas de etapas específicas, se identifica fácilmente el progreso de cada tarea y los cuellos de botella que puedan surgir.
3. **Automatización para ahorrar tiempo**: Al integrar Trello con otras herramientas y utilizar sus automatizaciones y reglas, se puede optimizar los procesos y mejorar la comunicación.
4. **Función cronograma**: Te permite visualizar las fechas de inicio y finalización de las tareas en una línea de tiempo, lo que facilita la planificación y el seguimiento del progreso del proyecto de manera eficiente.

Trello es ideal para planificar y gestionar recursos en pequeñas y medianas empresas. Pero cuando los proyectos son más grandes y complicados, con muchos niveles y dependencias, es donde enfrentamos un problema: necesitamos aplicaciones de terceros, llamadas *Power-Ups*, para mejorar sus funciones, y eso puede incrementar los costes.

1.4.2 Asana

Asana es un software de gestión de proyectos versátil, diseñado para organizar flujos de trabajo en empresas de todos los tamaños. Ofrece soluciones tanto para tareas rutinarias como para grandes iniciativas que requieran una planificación más profunda. Asana permite visualizar los proyectos de diferentes maneras, como listas, cronogramas o tableros, y cuenta con un completo conjunto de informes y gráficos para realizar seguimientos y detectar posibles errores. (16)

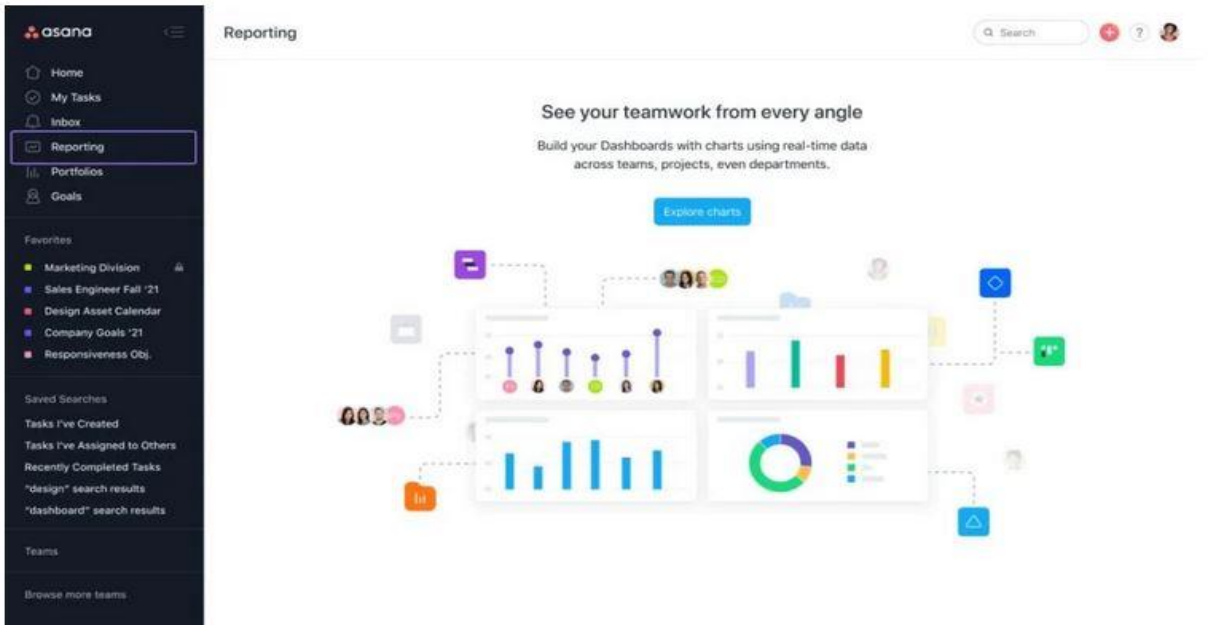


Figura 2. Interfaz de Asana. Fuente:(16)

1. **Flexibilidad en la visualización**: Asana te permite elegir entre diferentes vistas para adaptarse a las necesidades de cada proyecto, incluye listas, tableros Kanban o cronogramas. De esta forma, puedes organizarte mejor y visualizar las prioridades y plazos de entrega.
2. **Informes y seguimiento**: El apartado de informes de Asana cuenta con cuadros de mando y gráficos para analizar el estado de las actividades y detectar posibles problemas en el desarrollo de una línea de negocio.
3. **Integraciones con herramientas populares**: Su producto se integra con las principales herramientas de Google y Microsoft (Calendarios, Gmail, *Microsoft Teams*), así como con aplicaciones de finanzas, ventas y recursos humanos.
4. **Gestión eficiente de tareas y responsabilidades**: Esta herramienta te ayudará a simplificar la asignación de tareas, al establecer responsabilidades y fechas límite claras.

Asana es una de las herramientas de gestión de proyectos más completas en el mercado. Sin embargo, esta característica puede convertirse en una desventaja para algunos usuarios nuevos, quienes podrían encontrar complicado el aprendizaje debido a la gran cantidad de funcionalidades y preferir soluciones más sencillas y accesibles.

1.4.3 Akademos

Figura 3. Interfaz de Akademos. Fuente:(17)

Herramienta multiplataforma que contribuye al perfeccionamiento de los procesos académicos de una institución. Su uso permite el desarrollo coherente de una estrategia organizacional que articule todos los niveles de decisión presentes en los procesos universitarios. Todos los roles del proceso educativo están involucrados en la solución, por lo que se permitirá el acceso a la información de forma segura a todos los niveles, y facilita la toma de decisiones. (17)

Es un sistema Web distribuido, desarrollado en la plataforma .NET, que utiliza SQL Server 2000 para el almacenamiento de los datos, IIS (*Internet Information Services*) como servidor Web y Servicios WEB XML para el intercambio con otras aplicaciones. Sus funcionalidades se agrupan en siete módulos, de los cuales el Plan de Estudio es la entidad fundamental, pues rige todos los subprocesos.

Módulo de plan de estudio: Permite la gestión de diferentes especialidades o carreras al definir los planes de estudio.

Módulo de matrícula: Se encarga del control de los datos de los estudiantes y la gestión de los movimientos a que son sometidos.

Módulo de expediente: Actúa como un repositorio digital de los documentos y la información histórica de los estudiantes, disponible en todo momento.

Módulo de registro: Permite el control del desarrollo de un período académico con el registro de las evaluaciones y la asistencia.

Módulo de profesor: Mantiene un control de la plantilla de profesores.

Módulo de reportes: Obtiene reportes ordenados de la información almacenada en el sistema.

Módulo de estudiantes: Ofrece el historial académico de los años cursados por los estudiantes.

Con respecto a la seguridad, Akademos implementa varios niveles de acceso para restringir las acciones que puede realizar un usuario determinado. Lleva un control sobre las acciones que realizan los usuarios en el sistema y cuenta con una aplicación de monitoreo de las incidencias en tiempo real.

1.4.4 GESPRO

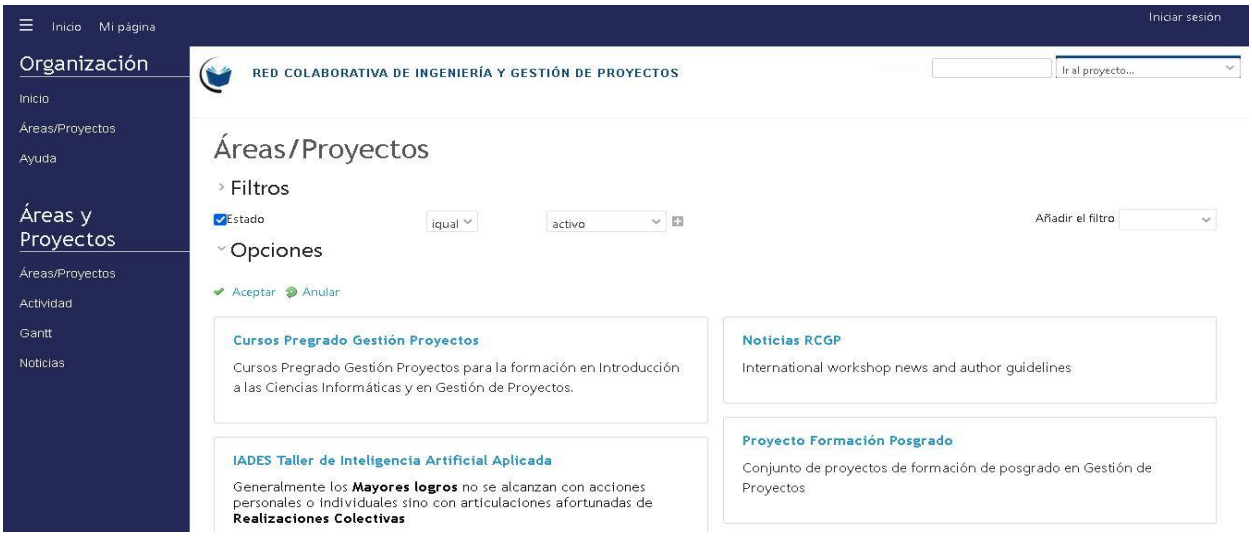


Figura 4. Interfaz de GESPRO. Fuente:(18)

El Sistema de Gestión de Proyectos (GESPRO) es una Suite orientada a la Web que permite planificar, monitorizar y controlar productos como proyectos informáticos. Cuenta con herramientas para apoyar la toma de decisiones en diferentes niveles como proyecto, entidad ejecutora o gerencia. Su modelo de negocio se basa en servicios que combinan el uso de una solución de software integrada para la gestión de proyectos y un sistema de formación especializada en gestión de proyectos. Esta combinación permite no solo la informatización de las organizaciones, sino también la mejora general de la planificación, control y seguimiento de los proyectos. El modelo de negocio se basa en los servicios y el precio del producto varía según

los servicios definidos, incluida la consultoría de gestión de proyectos. Actualmente, se utiliza en la Universidad de Ciencias Informáticas (UCI). (18)

Este repositorio de activos está orientado a interesados que deseen agilidad, integración y organización del proceso de toma de decisiones empresariales, en particular en organizaciones orientadas a proyectos. Incluye un cuadro de mando integral y tableros de controles especializados para la ayuda a la toma de decisiones: financieras, contables, gestión de ventas, recursos humanos, logística, gestión de proyectos, gestión de la producción, gestión de tareas, sistemas conversacionales y donde los procesos de análisis y visualización de la información están basados en técnicas de inteligencia artificial. Los activos del repositorio están desarrollados en Ruby, aunque incluye facilidades para la integración con Python y varios gestores de bases de datos. El gestor de base de datos principal sobre el que se basa la familia de soluciones es Postgres. Los activos incluyen facilidades con sistemas ERP, en particular para la gestión contable y financiera. Todos los activos están basados en licencia GNU GPL 2.0 y cumplen las libertades del software libre.

1.4.5 Análisis comparativos de los sistemas homólogos

Tabla 1. Análisis comparativo de sistemas homólogos. Fuente: Elaboración propia.

	Sistemas			
Características	**Trello**	**Asana**	**Akademos**	**GESPRO**
Gestión de usuarios	Sí	Sí	Sí	No
Gestión de roles	No	Sí	Sí	Sí
Gestión de permisos	No	Sí	No	Sí
Configuración de funcionalidades	Sí	No	Sí	Sí

Estas comparaciones señalan que ninguno de estos sistemas, tal como se ha calificado, proporciona una solución completa y adecuada para desarrollar un componente de control de acceso para la correcta gestión de usuarios, roles y permisos en el sistema GAPID. Puede ser más efectivo buscar una solución específica de gestión de usuarios, roles y permisos o considerar el desarrollo de un componente de control de acceso que se ajuste exactamente a las necesidades del sistema.

1.5 Ambiente de desarrollo

1.5.1 Herramientas

Las herramientas de desarrollo de software son diversos productos informáticos que dan soporte a una tarea concreta dentro de las actividades de desarrollo de software, que facilitan y aseguran entregar un sistema con calidad.

Django 4

Django es un marco de trabajo (*framework*) web de Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Es gratis y de código abierto. Django incluye docenas de extras que puede usar para manejar tareas comunes de desarrollo web. Se encarga de la autenticación de usuarios, la administración de contenido, los mapas del sitio, las fuentes RSS y muchas más tareas, desde el primer momento. Basado en un patrón de diseño conocido como Modelo-Vista-Plantilla, que brinda una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página desarrollada con este, y que puede administrar varias páginas hechas con Django a partir de una misma instalación. La aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, para llevar un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios. (19)

El objetivo principal de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio: “No te repitas”. (20)

Boostrap 5

Bootstrap es un marco *frontend* gratuito para un desarrollo web más rápido y fácil. Incluye plantillas de diseño basadas en HTML y CSS para tipografía, formularios, botones, tablas, navegación, modales, carruseles de imágenes y muchos otros, así como complementos de JavaScript opcionales. Bootstrap también le brinda la capacidad de crear fácilmente diseños receptivos (21).

El framework bootstrap 5 es una de las versiones recientemente lanzadas del Framework Bootstrap para el desarrollo web. Esta es una de las librerías más conocidas, pues se pueden construir aplicaciones web adaptables para móvil con el CDN de open source jsDelivr y una página con una plantilla de inicio. (22)

En términos de la versión 5, además de algunas mejoras sobre muchas herramientas, se tomaron decisiones relacionadas al uso y a la aplicación del framework. Por ejemplo, esta versión no es compatible con Internet Explorer ni con Jquery. Asimismo, se integraron con el generador de sitios web estáticos Hugo, para dejar de lado a Jekyll.

Por otro lado, se agregaron variables CSS, perfeccionaron el Grid CSS y se permitió añadir una nueva API de interfaz con Sass, donde puedes incluir tus propias herramientas.

Elementos de Bootstrap

El bootsrapt 5 está compuesta de varios elementos que la convierten en un *framework* mucho más completo e interesante para los desarrolladores web. Algunos de estos elementos son:

- ✓ El bootstrap5 precompilado que aparece una vez descargado y descomprimido el archivo de descarga de la librería.
- ✓ Todos los archivos CSS que puedas imaginar: *utilities, grid, reboot*.
- ✓ Todos los archivos JS compilados.
- ✓ El Bootstrap *source code*.

En términos programables de la librería, encontrarás:

- ✓ Elementos con cambios específicos de la hoja de estilos CSS en forma de *Reboot.*
- ✓ Documentación y ejemplificación de las tipografías nativas de este *framework*.
- ✓ Documentación y ejemplo para un uso de imágenes óptimo y con un comportamiento receptivo.
- ✓ Documentación y ejemplos para el uso opcional de tablas y su estilo.
- ✓ Documentación y ejemplos para el uso de figuras sobre el *display*.

Visual Paradigm

Visual Paradigm es una herramienta de modelado visual y gestión de requisitos ampliamente utilizada en el desarrollo de software. Permite crear modelos y diagramas que representan la estructura y el comportamiento de un sistema, así como gestionar los requisitos del proyecto. Con una interfaz intuitiva y funciones de colaboración en tiempo real, Visual Paradigm facilita la creación de documentación y el trabajo en equipo. Es una herramienta esencial para los desarrolladores de software en la planificación, diseño y documentación de proyectos. (23)

Visual Studio Code

Visual Studio Code es un entorno de desarrollo integrado (IDE) ligero y altamente popular utilizado por desarrolladores de software en todo el mundo. Proporciona una interfaz intuitiva y personalizable con una amplia gama de características y extensiones que facilitan la escritura de código, la depuración, el control de versiones y la colaboración en proyectos. Visual Studio Code admite múltiples lenguajes de programación y ofrece herramientas avanzadas como resaltado de sintaxis, autocompletado, navegación de código y depuración integrada. Además, su integración con Git permite un flujo de trabajo fluido para el control de versiones. Con su enfoque en la productividad y la eficiencia, Visual Studio Code se ha convertido en una herramienta esencial para los desarrolladores en su trabajo diario. (24)

1.5.2 Sistemas gestores de bases de datos (SGBD)

Sistema Gestor de Base de Datos. Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: *DataBase Management System*) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos. (25)

Los SGBD relacionales son una herramienta efectiva que permite a varios usuarios acceder a los datos al mismo tiempo. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos. (Sistemas de bases de datos orientadas a objetos.)

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto ampliamente utilizado en el desarrollo de aplicaciones. Proporciona una plataforma robusta y confiable para almacenar, organizar y recuperar datos de manera eficiente. PostgreSQL ofrece una amplia gama de características avanzadas, como soporte para consultas complejas, integridad referencial, transacciones ACID y funciones de escalabilidad. Su arquitectura modular y flexible permite personalizar y adaptar la base de datos según las necesidades específicas del proyecto. PostgreSQL es considerado uno de los sistemas de gestión de bases de datos más

potentes y confiables disponibles en la actualidad, y es una opción popular tanto para aplicaciones empresariales como para proyectos de código abierto. (26)

1.5.3 Lenguaje de programación

Python 3.12

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. Es multiplataforma, está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, *Windows*, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios (27).

JavaScript

El JavaScript es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado. Orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Proviene del Java y se utiliza principalmente para la creación de páginas web. El JavaScript es una mezcla entre el Java y el HTML, es un lenguaje que se incorpora dentro de la página web, y forma parte del código HTML (28).

HTML 5

Por sus siglas en inglés de (*HyperText Markup Language*) es el lenguaje básico de la web para crear documentos y aplicaciones para que los desarrolladores lo utilicen en cualquier lugar. HTML en su versión 5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Es un lenguaje simple que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos (29).

CSS 3

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Cabe agregar que el lenguaje CSS 3 se puede aplicar en la misma hoja en la que se desarrolla un documento HTML, pero por motivos de productividad se suele realizar en un documento aparte

con la extensión .CSS. Este documento se puede vincular a cada página HTML que conforme el sitio web, es por ello que es más útil realizar los estilos por separado. CSS 3 funciona mediante módulos son sólo categorías en las que se pueden dividir las modificaciones que hacemos al aspecto de nuestro sitio web. (30)

1.5.4 Metodología de desarrollo de software

La metodología de desarrollo de software AUP versión UCI La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP (Proceso Ágil Unificado), con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modifica el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agrega también una nueva fase llamada Cierre. (31)

A continuación, se muestra una tabla con las fases de la metodología AUP-UCI:

Tabla 2. Fases de la variación AUP UCI. Fuente:(31)

Fase AUP	**Fases Variación AUP-UCI**	**Objetivos de las fases (Variación AUP-UCI)**
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración, Construcción y Transición	Ejecución	En esta fase 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	En esta fase se analizan tanto los resultados del proyecto como la ejecución y se realizan las actividades formales de cierre del proyecto.

La metodología de software AUP-UCI a partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos, como son: CUN (Casos de uso del negocio), DPN (Descripción de proceso de negocio) o MC (Modelo conceptual) y existen tres formas de encapsular los requisitos, los cuales son: CUS (Casos de uso del sistema), HU (Historias de usuario), DRP (Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos, los cuales son:

• Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

• Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

• Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

• Escenario No 4: Proyectos que no modelen negocio, solo pueden modelar el sistema con HU

Se decide hacer una variación de la metodología AUP-UCI, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

Escenario No 4.

Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañado del equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una Historia de usuarios (HU) no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen Iteraciones y se obtengan resultados incrementales. En resumen, la metodología AUP-UCI Escenario 4 Historias de Usuario se seleccionó debido a su enfoque ágil, su capacidad de adaptación a los cambios y su enfoque en la entrega de valor temprana. Su énfasis en la

colaboración con los usuarios finales a través de historias de usuarios contribuirá a garantizar un desarrollo efectivo y satisfactorio del proyecto de tesis.

1.6 Conclusiones del capítulo

- ✓ EL marco conceptual de la investigación permitió mostrar la novedad y actualidad que tiene la necesidad de desarrollar una solución propia para el sistema GAPID.
- ✓ Los sistemas identificados a pesar de que son buenos y gestionan correctamente para su entorno la configuración del sistema, no cuentan con los elementos que se necesitan para esta investigación por lo que es necesario buscar una variante.
- ✓ El ambiente de desarrollo definido es el idóneo porque permitirá integrar de forma natural los resultados obtenidos al sistema GAPID.

CAPÍTULO 2: Características del componente de control de acceso

En el capítulo se describe el procedimiento y los elementos que se tuvieron en cuenta para el desarrollo del componente de control de acceso del sistema GAPID. Se presentan los conceptos asociados al modelo conceptual y la representación formal del mismo. Se identifican los requisitos no funcionales y funcionales. Además, se emplea la técnica de descripción de requisitos, acorde con la metodología empleada.

2.1 Características de la propuesta de solución

El objetivo principal de esta propuesta es propiciar mayor funcionalidad al sistema GAPID mediante la implementación de un componente de control de acceso que aborde las limitaciones actuales y brinde a los usuarios una mayor flexibilidad y funcionalidad. A continuación, se describen las características clave de esta solución:

1. **Gestión de usuarios**: La gestión de usuarios se refiere al conjunto de procesos, políticas y procedimientos utilizados para administrar las identidades y los accesos de las personas a sistemas informáticos, aplicaciones, redes y recursos dentro de una organización.
2. **Gestión de Roles y Permisos:** Desarrollar una interfaz de administración de roles y permisos que permita a los administradores definir roles personalizados y asignar permisos específicos a cada usuario. Esto garantizará que cada usuario tenga acceso solo a las funciones pertinentes a su rol en la institución.
3. **Control de Acceso Dinámico:** Implementar un sistema de control de acceso dinámico que se adapte a las necesidades cambiantes de la institución. Esto permitirá a los administradores ajustar rápidamente los permisos de los usuarios en función de los cambios en la estructura organizativa o los roles.
4. **Interfaz de Usuario Intuitiva:** Diseñar una interfaz de usuario intuitiva y amigable que permita a los usuarios administrar configuraciones y permisos de manera eficiente. Se debe prestar especial atención a la usabilidad y la accesibilidad.
5. **Integración con el Sistema Existente:** Asegurar que el componente de control de acceso se integre de manera fluida con el sistema GAPID existente sin interrupciones en el funcionamiento actual.

Descripción del negocio

El sistema GAPID presenta limitaciones en la gestión de usuarios, roles y permisos, lo que exige la creación de un componente de control de acceso. Además, es esencial introducir nuevas funcionalidades que permitan la personalización y control de las configuraciones del sistema.

2.2 Modelado del dominio

El modelado de dominio es una herramienta importante para comprender las entidades, relaciones y conceptos clave relacionados con el sistema y la solución propuesta. Su utilidad radica en ser una forma de “inspiración” para el diseño de los objetos software, es entrada para muchos de los artefactos que se construyen en un proceso software. Un modelo de dominio muestra las clases conceptuales significativas en un dominio del problema, se centra en las abstracciones relevantes, vocabulario del dominio e información del dominio y es el artefacto clave del análisis orientado a objetos. En UML se utilizan los diagramas de clases para representar los modelos de dominio. (32)

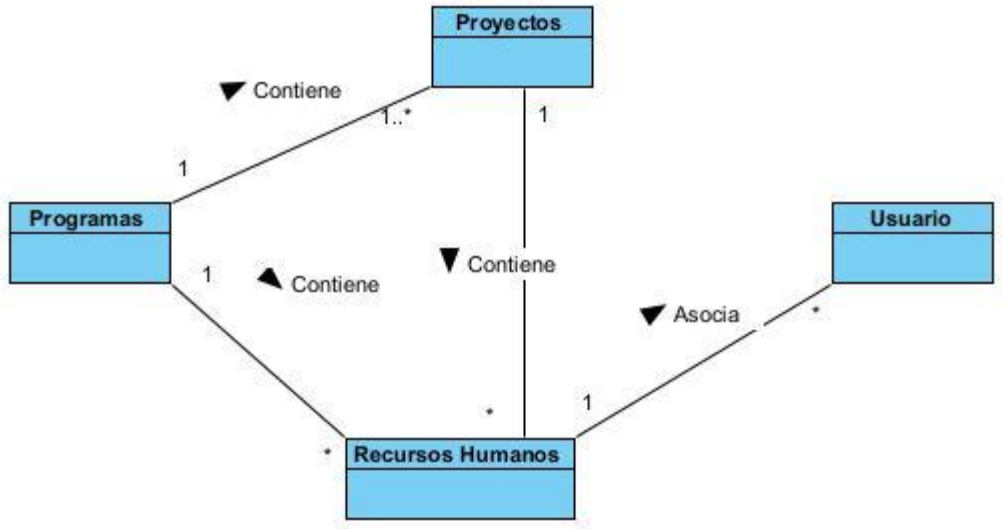


Figura 5. Modelo de dominio. Fuente: Elaboración propia.

Entidades Principales:

1. **Usuario:** Representa a una persona que utiliza el sistema GAPID. Cada usuario tiene un nombre de usuario y una contraseña, y se le asigna uno o varios roles.

2. **Rol:** Define un conjunto de permisos y privilegios en el sistema. Los roles incluyen roles predefinidos (por ejemplo, administrador, usuario estándar) y roles personalizados creados por los administradores.
3. **Permiso:** Cada rol tiene asociados uno o varios permisos que determinan las acciones que un usuario con ese rol puede realizar en el sistema.
4. **Recursos Humanos:** Representa todos los participantes involucrados en el sistema, los que son partes de los programas y proyectos con su respectivo rol.

Relaciones Clave:

- **Usuario tiene Rol:** Una relación que conecta a cada usuario con uno o varios roles. Un usuario puede tener múltiples roles, lo que permite una configuración personalizada de sus permisos.
- **Rol tiene Permiso:** Cada rol está relacionado con uno o varios permisos que definen las acciones permitidas para ese rol.

Tabla 3. Roles de los usuarios. Fuente: Elaboración propia

Rol	Descripción
Administrador	Encargado de gestionar los usuarios, roles, permisos y cerrar los programas y proyectos.
Jefe de proyecto	Encargado de gestionar la información de su programa y visualizar los proyectos dentro de este.
Secretario de proyecto	Encargado de gestionar la información de su programa y visualizar los proyectos dentro de este.
Jefe de proyecto	Encargado de gestionar la información de los proyectos y sus recursos humanos.

Gestor de proyecto	Encargado de certificar, evaluar, subir y descargar documentos asociados a un proyecto

2.3 Requisitos del Sistema

Los requerimientos son la esencia misma de un sistema, ya que definen las expectativas y necesidades de sus usuarios. Los requerimientos son la descripción de los servicios ofrecidos por el sistema y las restricciones que deben cumplirse. En el contexto del sistema GAPID, los requerimientos son la columna vertebral que guiará el diseño y la implementación de funcionalidades esenciales. (33)

2.3.1 Requisitos Funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la forma en que debe reaccionar ante ciertas entradas y cómo se debe comportar en situaciones particulares (34). A continuación, los requisitos identificados:

Tabla 4. Requisitos funcionales. Fuente: Elaboración propia.

No.	Requisito	Descripción	Prioridad	Complejidad
RF1	Crear usuario	Permite crear un nuevo usuario en el sistema	Alta	Alta
RF2	Modificar usuario	Permite modificar la información de un usuario registrado en el sistema	Alta	Alta
RF3	Deshabilitar usuario	Permite deshabilitar un usuario registrado en el sistema	Media	Media
RF4	Habilitar usuario	Permite habilitar un usuario registrado en el sistema en caso de estar deshabilitado	Media	Media

RF5	Listar usuarios	Permite mostrar una lista de usuarios registrados	Alta	Media
RF6	Autenticar usuario	Permite iniciar sesión a los usuarios	Alta	Alta
RF7	Cerrar sesión	Permite cerrar sesión a los usuarios	Alta	Media
RF8	Asignar permisos al rol	Permite asignar permisos según el rol	Alto	Alto
RF9	Quitar permisos	Permite quitar los permisos asignados que tenga un rol	Alto	Alto
RF10	Buscar usuario	Permite buscar un usuario registrado en el sistema	Media	Baja
RF11	Buscar rol	Permite buscar un rol asignado	Media	Baja
RF12	Buscar permiso	Permite buscar permisos según rol	Media	Baja

2.3.2 Requisitos No Funcionales

Los requerimientos no funcionales, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. Los requisitos no funcionales, en el contexto de esta tesis sobre el modelado del negocio y la propuesta de solución para el sistema GAPID se centran en sus características y capacidades generales. Estos requisitos determinan las propiedades y restricciones clave del sistema, como su rendimiento, seguridad, usabilidad y capacidad de escalabilidad. (34)

Estos requisitos forman una parte significativa de la especificación de requisitos y en algunos casos estos son críticos para el éxito del producto. Con frecuencia estos requisitos son ignorados o subestimados debido a que para muchos proyectos estos implican una cantidad considerable

de trabajo y esfuerzo; resultan ser más complejos y requieren un mayor nivel de conocimiento. (35)

Usabilidad:

RNF 1: Los mensajes para interactuar con los usuarios y los de error deben ser lo suficientemente informativos, en idioma español y no deben revelar información interna.

Rendimiento:

RNF 2: El tiempo de respuesta del sistema no debe exceder de 5 segundos.

Interfaz:

RNF 3: El diseño será sencillo, amigable e intuitivo; que combine correctamente los colores, tipo y tamaño de letra.

Seguridad:

RNF 4: El sistema requiere la autenticación como primera acción, con un nombre de usuario único y una contraseña, que deben ser de conocimiento exclusivo de la persona que se autentica.

RNF 5: La información solo puede ser modificados por aquellos usuarios autorizados.

RNF 6: La seguridad estará regida por un mecanismo de control de acceso basado en roles, convenientemente asignados a los usuarios del sistema al momento de su creación.

2.4 Historia de Usuario

Las historias de usuario son una técnica utilizada en el desarrollo de software, particularmente en metodologías ágiles como Scrum, para definir requisitos y funcionalidades desde la perspectiva del usuario. Estas historias son una forma de capturar los requisitos del sistema en un lenguaje simple y comprensible para todas las partes involucradas, incluyen desarrolladores, diseñadores y usuarios finales. (36) Generalmente, una historia de usuario consta de los siguientes elementos:

1. **Nombre o Título de la Historia**: Un nombre breve pero descriptivo que resume la funcionalidad que se describe.
2. **Número de Identificación**: Un identificador único para la historia de usuario, que a menudo incluye una referencia al proyecto o iteración.

3. **Usuario**: La persona o el rol del usuario que necesita la funcionalidad descrita en la historia.
4. **Descripción**: Una narrativa o explicación detallada de la funcionalidad que se requiere. Debe ser lo suficientemente clara como para que el equipo de desarrollo comprenda qué se debe lograr.
5. **Prioridad en el Negocio**: Una indicación de la importancia de la historia para el negocio o el proyecto. Por lo general, se clasifican en categorías como "Alta", "Media" o "Baja" para ayudar a priorizar el trabajo.
6. **Riesgo en el Desarrollo**: Una evaluación del riesgo que implica implementar la funcionalidad descrita en la historia, a menudo clasificada como "Alto", "Medio" o "Bajo".
7. **Tiempo Estimado**: Una estimación aproximada del tiempo que llevará implementar la historia, generalmente en días o semanas.
8. **Iteración Asignada**: En qué iteración o ciclo de desarrollo se abordará esta historia.
9. **Programador Responsable**: El miembro del equipo de desarrollo que se encargará de llevar a cabo la implementación.
10. **Prototipo de Interfaz Gráfica de Usuario**: Si es relevante, se pueden incluir bocetos, diseños o enlaces a prototipos de la interfaz de usuario relacionados con la historia.

Las historias de usuario se utilizan para garantizar que el desarrollo de software se enfoque en las necesidades y expectativas del usuario final. Además, facilitan la colaboración y la comunicación efectiva entre los miembros del equipo, lo que es esencial en las metodologías ágiles. Cada historia de usuario representa una pequeña parte de la funcionalidad total del sistema y se aborda de manera incremental a lo largo del tiempo.

Tabla 5. Historia de usuario No.1. Fuente: Elaboración propia.

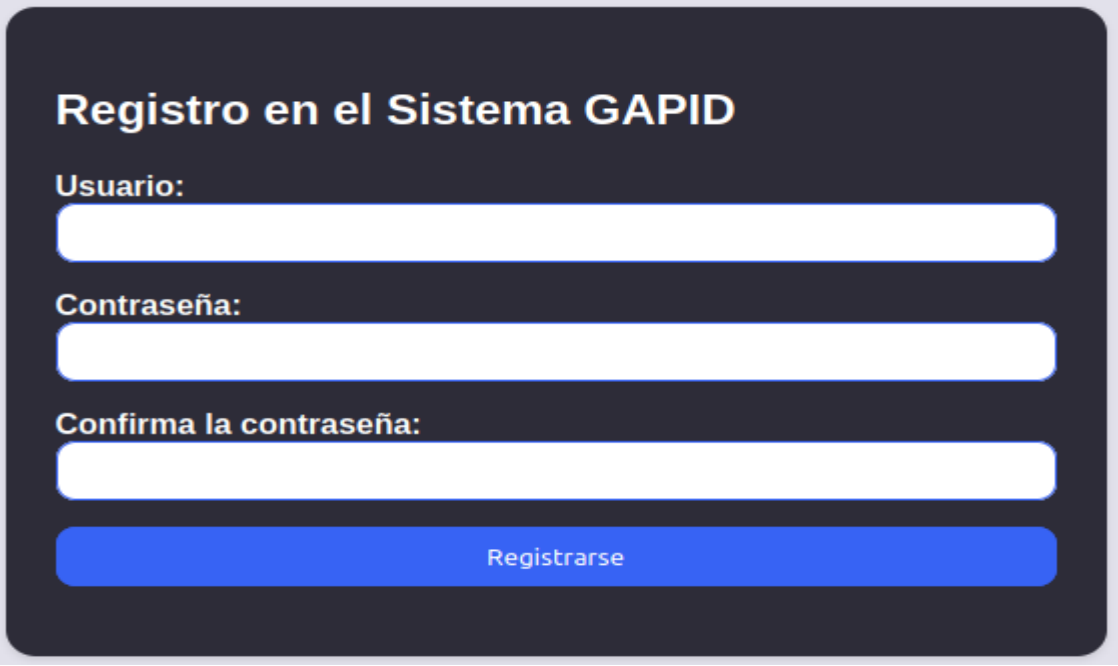
Historia de Usuario	
Número: RF 3	**Usuario:** Administrador
Nombre de Historia: Deshabilitar usuario	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo

Tiempo estimado: 5 días	**Iteración Asignada:**2
Programador Responsable: Josué Rivas Calderón	
Descripción: En caso de tener ya un usuario asociado a un participante en el sistema el administrador tiene la opción de deshabilitarle el usuario a ese participante**.**	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 6. Historia de usuario No.2. Fuente: elaboración propia.

Historia de Usuario	
Número: RF 1	**Usuario:** Administrador
Nombre de Historia: Crear usuario	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Tiempo estimado: 7 días	**Iteración Asignada:**1
Programador Responsable: Cristian Guerra Luaces	
Descripción: Se accederá a una interfaz de registro de usuario, donde se llenaran los datos del formulario para crear el usuario en el sistema.	

Prototipo elemental de interfaz gráfica de usuario:



2.5 Arquitectura de software

La arquitectura del sistema existente con el que se relaciona la propuesta de solución es de vital importancia para comprender su estructura y funcionamiento. En este sentido, se empleará la modelo vista plantilla (MVT) como enfoque arquitectónico para el desarrollo del sistema GAPID.

El patrón Modelo-Vista-Controlador (MVC) se utiliza en el desarrollo web para separar las preocupaciones de una aplicación en tres componentes principales: el Modelo (que maneja los datos y la lógica de negocio), la Vista (que se encarga de la presentación) y el Controlador (que actúa como intermediario entre el Modelo y la Vista). En Django, un *framework* de desarrollo web de Python, la implementación del patrón MVC se denomina Modelo-Vista-Plantilla (MVT). (37)

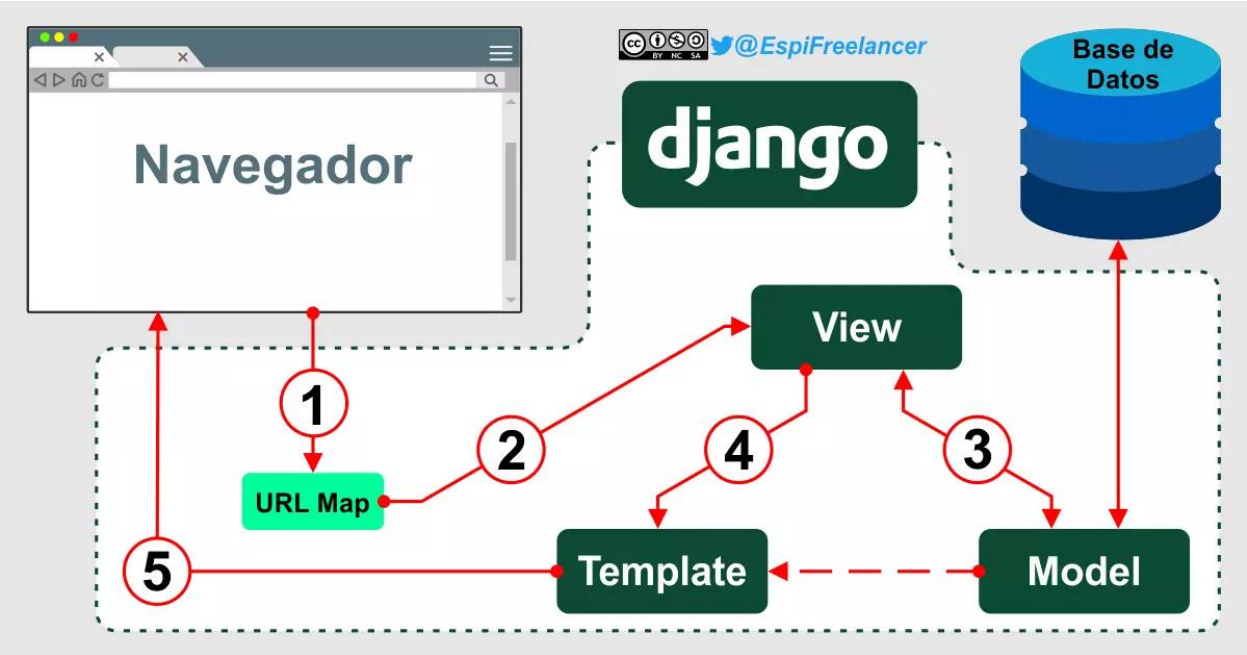


Figura 6. Modelo-Vista-Plantilla. Fuente:(37)

Explicación de los componentes:

1. **Modelo (*Model)***:
    - ✓ Representa los datos y la lógica de negocio de la aplicación.
    - ✓ Comunica la aplicación con la base de datos.
    - ✓ En Django, los modelos se definen como clases de Python en el archivo models.py.
2. **Vista (*View)***:
    - ✓ Se encarga de la lógica relacionada con la presentación y la interacción del usuario.
    - ✓ En Django, las vistas son funciones o clases de Python definidas en el archivo views.py.
    - ✓ Se comunican con el Modelo para acceder a los datos y renderizan plantillas para generar la interfaz de usuario.
3. **Plantillas (*Template*)**:
    - ✓ Representa la parte visual de la aplicación web.

- ✓ Las plantillas en Django se crean con la utilización del motor de plantillas de Django y se almacenan en archivos separados.
- ✓ Las vistas utilizan plantillas para generar contenido dinámico basado en datos del Modelo.

2.6 Patrones de diseño

Los patrones de diseño son soluciones probadas y documentadas para problemas recurrentes en el diseño de software. Estas soluciones, desarrolladas y perfeccionadas por expertos en el campo, ofrecen un enfoque efectivo y eficiente para abordar desafíos comunes en el desarrollo de software. Utilizar patrones de diseño no solo aumenta la calidad y la claridad del diseño, sino que también promueve la reutilización de soluciones exitosas, aceleran el proceso de desarrollo y mejoran la mantenibilidad del software a lo largo del tiempo. En resumen, los patrones de diseño son un recurso valioso para los ingenieros de software en la creación de sistemas sólidos y flexibles. (38)

Los Patrones GRASP *(General Responsibility Assignment Software Patterns)* ofrecen un conjunto de prácticas sólidas para guiar el diseño de software. Su objetivo principal es definir principios fundamentales relacionados con la asignación de responsabilidades y el diseño de objetos. Entre los Patrones GRASP más destacados se incluyen Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador**.** (39)

Experto

El patrón experto, en el contexto de los patrones GRASP, se basa en el principio de asignar responsabilidades a las clases que tienen la información necesaria para cumplir con esas responsabilidades de manera efectiva. Este principio se refiere a que una clase debe ser considerada como un "experto" en una tarea específica si contiene la información y el conocimiento necesarios para llevar a cabo esa tarea de manera adecuada.

En términos simples, el patrón Experto se utiliza para determinar cuál es la clase más adecuada para realizar una tarea o llevar a cabo una operación dentro de un sistema orientado a objetos. La idea principal es evitar la duplicación de información y promover un diseño más claro y eficiente. (35)

Creador

El patrón Creador es un patrón de diseño de software que se utiliza en el ámbito de la programación orientada a objetos. Su objetivo principal es proporcionar una forma de construir

objetos complejos de manera que el proceso de creación se mantenga separado de su representación. Esto implica que se pueda crear un objeto sin necesidad de conocer los detalles exactos de cómo se construye o se ensambla.

El patrón Creador se implementa mediante una clase (o conjunto de clases) que se encarga de crear instancias de otras clases, normalmente objetos complejos, mediante la llamada a un constructor específico. Esto permite encapsular la lógica de creación y configuración de objetos, lo que hace que el código sea más flexible y mantenible.

Alta cohesión

La alta cohesión se refiere a la medida en que los elementos dentro de un módulo, clase o componente están estrechamente relacionados y trabajan juntos para lograr una función o tarea específica. Esto implica que un módulo o componente debe tener una única responsabilidad o función claramente definida, y todas sus partes deben contribuir a esa función sin desviarse hacia otras tareas no relacionadas. En otras palabras, un elemento con alta cohesión debe estar altamente enfocado en una tarea particular y no debe realizar funciones adicionales que no estén directamente relacionadas con esa tarea.

Bajo acoplamiento

El patrón de "bajo acoplamiento" en el diseño de software se refiere a la independencia y la reducción de dependencias entre módulos, componentes o clases en un sistema. Este concepto se relaciona con la organización de un sistema de manera que los componentes sean autónomos y no estén fuertemente vinculados entre sí. El objetivo es minimizar la influencia que un componente tiene sobre otros, lo que facilita la modificación, el mantenimiento y la reutilización de partes del sistema sin afectar a todo el sistema.

El bajo acoplamiento se logra con buenas prácticas de diseño, como la separación de preocupaciones, la encapsulación y la aplicación de patrones de diseño que promuevan la independencia de los componentes. Esto mejora la flexibilidad y la escalabilidad del sistema, ya que los cambios en un componente tienen un impacto limitado en otros componentes. En resumen, el bajo acoplamiento es un principio de diseño que la independencia de los componentes en un sistema de software.

Este principio se utiliza para mejorar la claridad, la reutilización y la mantenibilidad del código, ya que un diseño con alta cohesión tiende a ser más fácil de entender y de modificar, y sus partes son más autónomas y pueden ser reutilizadas en otros contextos con mayor eficacia.

Controlador

El patrón de diseño Controlador, dentro del contexto del patrón de arquitectura Modelo-Vista-Controlador (MVC), se refiere a la parte del sistema que gestiona la lógica de la aplicación y actúa como intermediario entre el modelo de datos y la vista de la interfaz de usuario. Su objetivo principal es mantener la separación de responsabilidades, lo que facilita la escalabilidad, y el mantenimiento de las aplicaciones.

En este patrón, el Controlador recibe las solicitudes del usuario, procesa la información y decide cómo interactuar con el Modelo para realizar operaciones en los datos. Luego, actualiza la Vista para mostrar los resultados al usuario. Esto permite que tanto el Modelo como la Vista operen de manera independiente, sin tener conocimiento directo entre sí.

Creador (*Creator*): Este patrón sugiere que una clase debe ser responsable de crear instancias de otras clases si hay una relación de composición entre ellas. En el código, por ejemplo, la clase Registro crea instancias de FormularioRegistroPersonalizado en los métodos get y post. Esto sigue el principio del Creador, ya que la clase Registro tiene la responsabilidad de crear instancias del formulario. Ver figura 7

Experto (*Expert*): Este principio establece que una clase debe tener la responsabilidad de la información que posee. En el código, la clase Registro actúa como un experto al manipular instancias del formulario FormularioRegistroPersonalizado y el modelo Participante. La clase que tiene la información necesaria debe ser la responsable de manipularla. Ver figura 7

Alta Cohesión (*High Cohesion*): Este principio se refiere a la medida en que las responsabilidades de una clase están relacionadas entre sí. En el código, parece haber una alta cohesión en las funciones de las vistas. Por ejemplo, la clase Registro se encarga de la lógica relacionada con el registro de usuarios y su asociación con participantes. Ver figura 7

Bajo Acoplamiento (*Low Coupling*): Este principio busca reducir las dependencias entre las clases. En el código, se observa un bajo acoplamiento, ya que las clases no están fuertemente acopladas entre sí. Por ejemplo, la clase Registro no tiene un conocimiento directo de las implementaciones internas de FormularioRegistroPersonalizado, simplemente interactúa con él a través de una interfaz. Ver figura 7

Controlador (*Controller*): El patrón Controlador sugiere que una clase debe ser designada para manejar todas las solicitudes de entrada y coordinar las operaciones del sistema. En el código, varias clases actúan como controladores, como la clase Registro que maneja la lógica de registro y la clase Participantes_admin que maneja la presentación de participantes. Ver figura 7

```python
class Registro(View):

    # Esta funcion renderiza el html registrar usuario
    def get(self, request):
        form = FormularioRegistroPersonalizado()
        return render(request, "registro.html", {"form": form})

    # Esta funcion envia los datos al modelo y los guarda en la base de datos
    def post(self, request):
      form = FormularioRegistroPersonalizado(request.POST)
      if form.is_valid():
        participante_id = request.GET.get('participante_id')  # Obtén el ID del participante
        participante = Participante.objects.get(id=participante_id)  # Obtén el participante
        user = form.save()  # Crea el usuario
        participante.user = user  # Asocia el usuario al participante
        participante.save()  # Guarda los cambios en la base de datos
        return redirect('lista-participantes')
      else:
        return render(request, "registro.html", {"form": form})
```

Figura 7. Código de la clase registro. Fuente: Elaboración propia.

2.7 Modelo de datos

El modelado de datos es el proceso de diagramación de los flujos de datos. Al crear la estructura de una base de datos nueva o alternativa, el diseñador comienza con un diagrama del flujo de los datos por dentro y fuera de la base de datos. Este diagrama se usa para definir los formatos y estructuras de los datos y las funciones de gestión de la base de datos, a fin de dar un soporte eficiente al flujo de datos. Una vez creada e implementada la base de datos, el modelo de datos es la documentación y justificación de por qué existe la base de datos y cómo se diseñaron los flujos. (40)

Para darle solución a la problemática planteada se obtuvo un modelo relacional de datos, que se desarrolló en IDE *Jetbrains Pycharm* para el diseño y análisis lógico de los datos. Se representa en la figura donde se aprecia la relación de datos general del sistema:

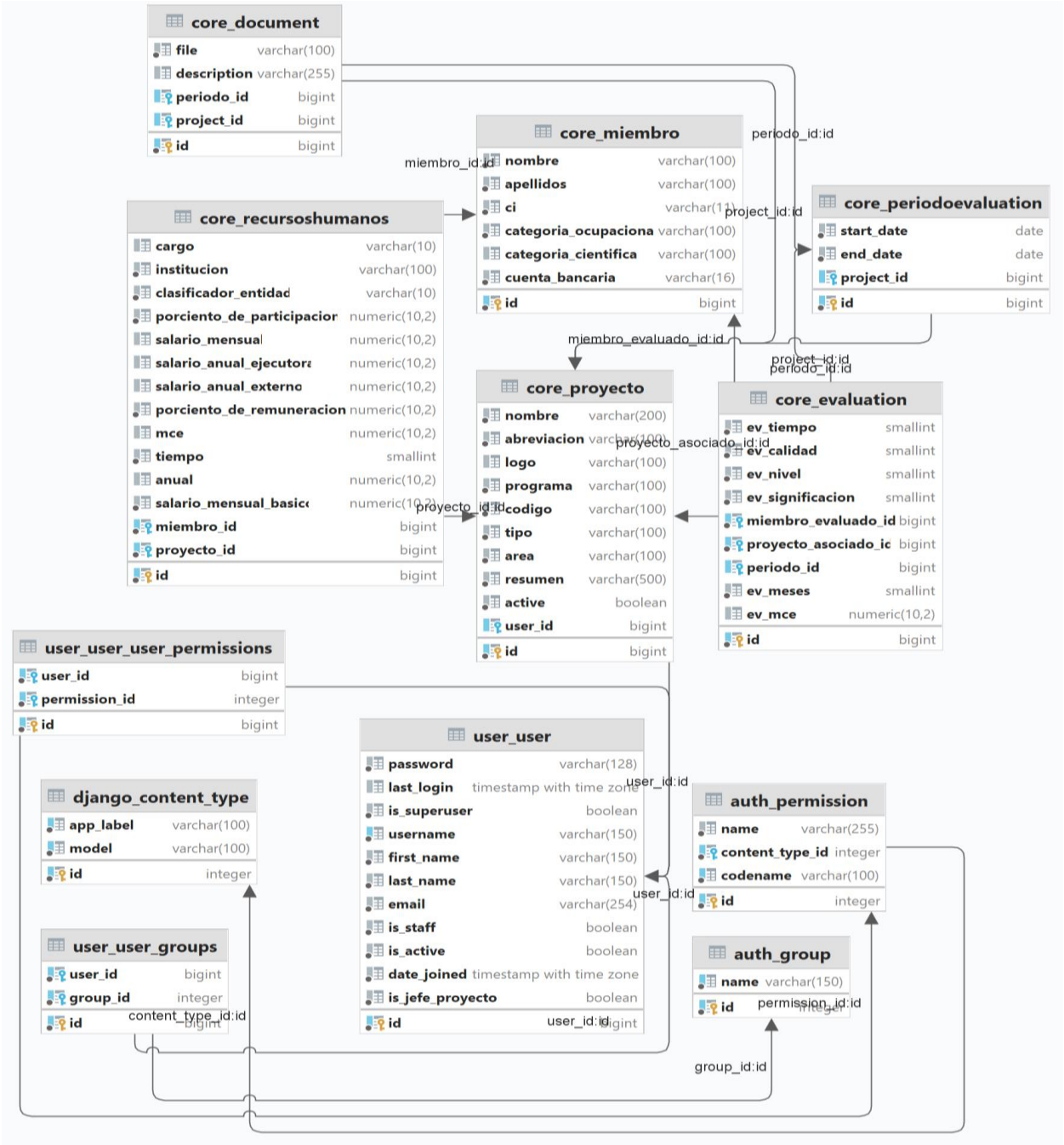


Figura 8. Modelo de datos. Fuente: Elaboración propia.

Para el desarrollo del componente de control de acceso se hicieron cambios sobre la tabla participantes a la cual se le agregó el atributo *user_id* que es la llave foránea de la tabla *authentication* lo que permite asociar un usuario a un participante donde el participante solo puede tener un usuario. También se trabajó sobre la tabla rol a la cual se le agregó el atributo *permission_id* que es la llave foránea de la tabla *auth_permission* para asociar los permisos de cada rol donde un rol puede tener varios permisos.

Figura 9. Relación entre la tabla *authentication* y participantes. Fuente: Elaboración propia.

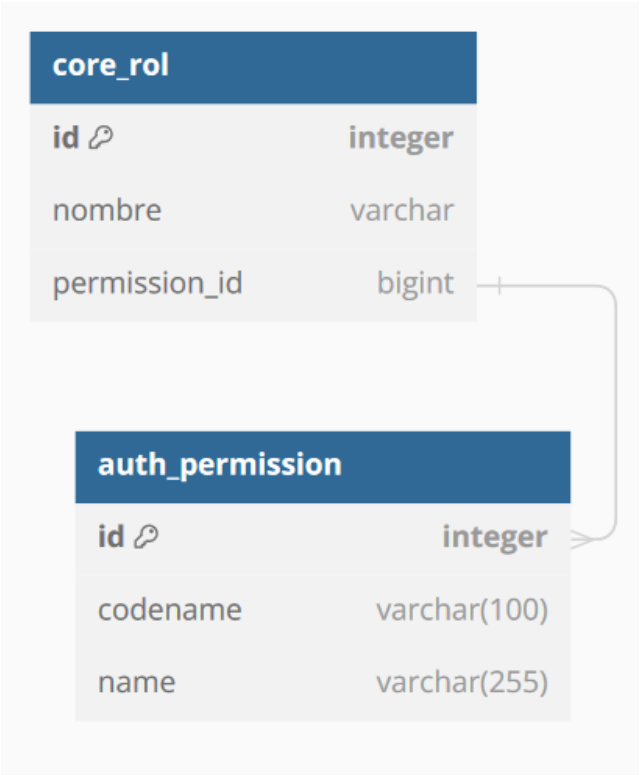


Figura 10. Relación entre la tabla la tabla rol y *auth_permission*. Fuente: Elaboración propia.

2.8 Conclusiones del capítulo

- ✓ La aplicación de las técnicas de adquisición de información permitió obtener los requisitos necesarios para poder desarrollar el componente de control de acceso a partir de la correcta gestión de los usuarios, roles y permisos.
- ✓ El uso de patrones de diseño propició obtener un código limpio, organizado y entendible además que permite ser escalable por otros desarrolladores.

CAPÍTULO 3: Implementación y pruebas

En este capítulo muestra los estándares de codificación usados durante la implementación de la solución y se evalúa el nivel de calidad y fiabilidad de los resultados obtenidos en el desarrollo de la propuesta de solución. Dicha valoración se lleva a cabo a partir del establecimiento de una estrategia de pruebas, que aplican las disciplinas de pruebas internas y de aceptación que define la metodología que guía el desarrollo de la solución propuesta, con el fin de verificar y revelar la calidad del producto antes de su entrega al cliente.

3.1 Seguridad informática

Todos los componentes de un sistema informático están expuestos a un ataque (hardware, software y datos) estos son los datos y la información los sujetos principales de protección de las técnicas de seguridad. La seguridad informática se dedica principalmente a proteger la confidencialidad, la integridad y disponibilidad de la información (46).

La seguridad es un tema de gran impacto en el sistema GAPID, pues es de vital importancia el control de la información que se almacena y se visualiza para garantizar la confidencialidad de la información. Es por ello, que todo usuario que interactúe con la solución propuesta deberá autenticarse para realizar alguna acción sobre la misma.

A continuación, se describen detalladamente las funcionalidades que ofrece el componente de control de acceso, el cual se encargará de garantizar la seguridad en el sistema a desarrollar:

- ✓ En el sistema se dan los permisos de acuerdo con la función que ocupa el usuario en el mismo, lo que permite solo tener acceso a funcionalidades y servicios que respondan directamente a su rol.
- ✓ El registro de trazas permite archivar las acciones que realiza el usuario, que pueden ser: inicio o cierre de sesión, acceso a un módulo o modificación de un atributo de una entidad. Para cada acción el sistema registra una traza en la base de datos.
- ✓ El sistema brinda a través de la funcionalidad administrar seguridad, la posibilidad de asignar o denegar permiso a roles y usuarios en las funcionalidades de los módulos.

3.2 Estándares de codificación

Los estándares de codificación, son una práctica altamente recomendada para desarrollar software de alta calidad, estos estándares establecen criterios únicos que los programadores deben implementar cuando escriben código, para que el código fuente pueda ser entendido por

cualquier miembro del equipo de desarrollo, y a su vez permite que el código pueda ser modificado por otro programador para evitar que tenga que escribir la totalidad del código, lo que ocasionaría costos extras y mayor tiempo del requerido. A continuación, se presentan algunos estándares de codificación que fueron utilizados durante la implementación del sistema. (41)

Grosor de línea:

Cada línea de código no debe exceder los 80 caracteres en la medida de lo posible (en circunstancias especiales, puede exceder ligeramente los 80, pero la más larga no puede exceder los 120).

Razones:

- ✓ Esto es útil al ver una diferencia de lado a lado.
- ✓ Conveniente para ver el código debajo de la consola.
- ✓ Demasiado tiempo puede ser un diseño defectuoso.

Líneas en blanco:

- ✓ Dos líneas en blanco entre las funciones de nivel de módulo y las definiciones de clase.
- ✓ Línea en blanco entre las funciones de los miembros de la clase.
- ✓ Las líneas en blanco se pueden utilizar en funciones para separar códigos relacionados lógicamente.

Declaración de importancia:

- ✓ Las declaraciones de importación deben organizarse en orden y deben ser separadas por una línea en blanco entre cada grupo.
- ✓ Deben colocarse al principio del archivo, después de la descripción del módulo y la cadena de documentos, y antes de las variables globales.

Espacio:

- ✓ Un espacio a cada lado del operador binario [=, -, +=, ==, >, in, is not, and].
- ✓ En la lista de parámetros de la función, debe haber un espacio después.
- ✓ No se puede agregar espacios adicionales después del paréntesis izquierdo y antes del paréntesis derecho.

- ✓ No puede haber espacios adicionales antes del paréntesis de apertura del objeto de diccionario.
- ✓ No se usa espacios adicionales para alinear las declaraciones de asignación.

Comentario de línea:

- ✓ Usa al menos dos espacios para separar la declaración, y los comentarios no pueden ser sin sentido.

3.3 Pruebas de Software

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información sobre la calidad del producto. Depende del tipo de pruebas, estas actividades pueden ser implementadas en cualquier momento del proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo (42).

3.3.1 Tipos de prueba

Para comprobar el correcto funcionamiento de la solución obtenida se realizan los siguientes tipos de pruebas:

- ✓ **Pruebas funcionales**: se basan en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas.
- ✓ **Pruebas estructurales**: se basan en medir la totalidad de las pruebas mediante la evaluación de tipo estructura.

En la presente investigación se utilizaron las técnicas de pruebas dinámicas, las cuales permitieron el uso de los métodos de caja negra para las pruebas funcionales y caja blanca para las pruebas estructurales.

3.3.2 Estrategia de Evaluación de Software

Una estrategia efectiva de evaluación de software es esencial para garantizar la calidad del producto final. Esta estrategia combina métodos de diseño de casos de prueba en una serie de pasos bien planificados que conducen a la construcción exitosa del software. Proporciona una hoja de ruta detallada que describe cuándo y cómo se realizarán las pruebas, los recursos necesarios y el tiempo estimado para su ejecución. (43)

Elementos clave de la estrategia de evaluación de software:

1. **Planificación de Pruebas:** En esta fase, se establecen los objetivos de las pruebas, se define el alcance y se asignan los recursos necesarios. También se identifican los niveles de pruebas que se llevarán a cabo a lo largo del ciclo de desarrollo.
2. **Diseño de Casos de Prueba:** Se crean casos de prueba específicos que abarcan una amplia variedad de escenarios. Estos casos deben cubrir todas las funcionalidades críticas y posibles situaciones de uso.
3. **Ejecución de Pruebas:** Se implementan los casos de prueba y se ejecutan en el software. Los resultados se registran y se comparan con los criterios de éxito definidos.
4. **Recolección y Evaluación de Datos:** Los datos de las pruebas se recopilan y analizan para determinar si el software cumple con los requisitos y para identificar posibles problemas o defectos.
5. **Niveles de Pruebas:** La estrategia incluye la definición de diferentes niveles de pruebas, como pruebas unitarias, pruebas de aceptación y pruebas de sistema. Cada nivel tiene su propio enfoque y objetivos específicos.
6. **Intensidad de las Pruebas:** Se especifica el grado de profundidad con el que se aplicarán las pruebas en cada nivel. Esto puede variar según la importancia de la funcionalidad evaluada y los recursos disponibles

3.3.3 Método de caja blanca. Técnica de camino básico

“Las pruebas de caja blanca están dirigidas a las funciones internas del módulo, caso contrario a las de caja negra, que examinan los requisitos funcionales desde el exterior del módulo” (44). Las pruebas de caja blanca pueden aplicarse a los métodos de la clase, aunque generalmente son usadas en funciones complejas de programación estructurada. Por lo expuesto anteriormente, se decide aplicar este tipo de pruebas a las funciones de mayor complejidad, para aprovechar así las potencialidades de las pruebas de caja blanca para detectar errores de codificación y flujo.

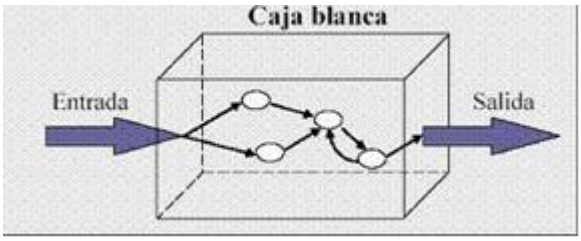


Figura 11. Método caja blanca. Fuente:(47)

Se realizaron pruebas unitarias a través del método de caja blanca en la clase registro y se utiliza la técnica de camino básico. A continuación, se muestra el código donde se realizaron las pruebas:

```
from django.contrib.auth import get_user_model
from apps.core.usuario.forms import FormularioRegistroPersonalizado

class Registro(View):
    def get(self, request):
        form = FormularioRegistroPersonalizado()
        return render(request, "registro.html", {"form": form})

    def post(self, request):

        if request.method == 'POST':
         form = FormularioRegistroPersonalizado(request.POST)
         if form.is_valid():
            User = form.save()
            return redirect('lista')
         else:
           return render(request,"registro.html", {"form": form})
```

Figura 12. Fragmento de código de la clase registro. Fuente: Elaboración propia.

Dentro de las pruebas de caja blanca, se emplea la técnica del camino básico, la cual permite conocer una medida de la complejidad lógica de una función procedural y usarla como guía para definir un conjunto básico de rutas de ejecución. Mediante esta técnica se garantiza que cada instrucción se ejecute al menos una vez durante el desarrollo de la prueba.

Se muestran las pruebas de caja blanca realizadas en la clase registro. Se analiza la complejidad ciclomática, que es la métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. “La complejidad ciclomática calcula la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez”. (45)

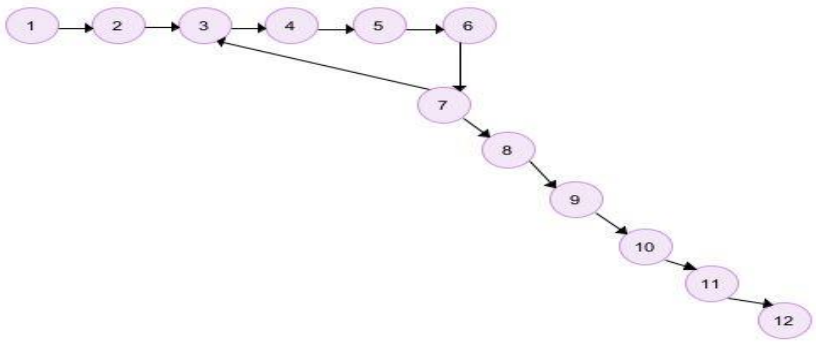


Figura 13. Grafo de flujo. Fuente: Elaboración propia.

Luego de elaborar el grafo de flujo, se calcula la complejidad ciclomática con sus tres diferentes fórmulas:

1. V(G) = R, donde R representa la cantidad de regiones en el grafo.

V(G) = 2

2. V(G) = A – N + 2, donde A es el número de aristas y N el número de nodos.

V(G) = 12 – 12 + 2 = 2

3. V(G) = P + 1, donde P es el número de nodos predicado contenidos en el grafo.

V(G) = 1 + 1 = 2

Si se tiene en cuenta que el cálculo de la complejidad ciclomática arrojó el mismo resultado para las 3 variantes y que este valor representa el número mínimo de pruebas para la función, se puede afirmar que la complejidad ciclomática de la función es 2, lo que significa que existen 2 posibles caminos, mostrados a continuación, por donde el flujo puede circular.

Camino #1: 1-2-3-4-5-6-7-8-9-10-11-12

Camino #2: 1-2-3-4-5-6-7-3-4-5-6-7-8-9-10-11-12

A continuación, se muestran los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo

Tabla 7. Casos de prueba. Fuente: Elaboración propia.

Camino	Valores de entrada	Resultado esperado
Camino básico #1	Validación de los valores del formulario si cumple con los parámetros de política de seguridad.	Se muestra un mensaje que el usuario se ha registrado en el sistema.

Camino básico #2	Validación de los valores del formulario si no cumple con los parámetros de política de seguridad.	Se muestra un mensaje de que los valores no cumplen con los requisitos

3.3.4 Método de caja negra. Técnica de partición equivalente

El método de caja negra examina el programa para que cuente con todas las funcionalidades y analiza los resultados que devuelve y prueba todas las entradas en sus valores válidos e inválidos. Con este método se intenta encontrar los errores: de inicialización y terminación, de interfaz y en las estructuras (47).

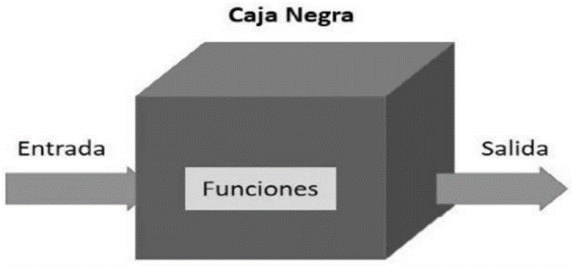


Figura 14. Método de caja negra. Fuente:(47)

Existen varias técnicas que se pueden aplicar al método de caja negra, pero en la presente investigación se empleará partición equivalente, el cual consiste en que los valores de entrada del sistema se dividen en grupos que vayan a tener un comportamiento similar para ser procesados de la misma forma.

Las pruebas integrales se tienen que aplicar justo después de haber llevado a cabo cada prueba unitaria con la intención de probar los métodos aplicados en el desarrollo. Si no existe ningún problema de código y las pruebas unitarias han terminado de forma exitosa se podrá pasar al test integral para asegurarse de que en este punto no se produce ningún tipo de problema en la combinación de elementos unitarios (47).

Una vez integrada el componente de control de acceso al sistema GAPID, se le realizaron pruebas de integración mediante el método de caja negra, y se utiliza la técnica de partición equivalente. A continuación, se muestran los resultados obtenidos una vez aplicadas las pruebas:

Tabla 8. Resultados método caja negra mediante la técnica de partición equivalente en las pruebas de integración. Fuente: Elaboración propia.

No. Iteración	NC	Funcionalidad	Interfaz	Correspondencia	Ortografía	Resueltas
1era	12	3	4	2	3	12
2da	5	0	2	1	2	5
3era	0	0	0	0	0	0
Total	17	3	6	3	5	17

Las causas de las no conformidades detectadas fueron:

- ✓ Errores de interfaz: Paneles que no cumplían con las pautas de diseño establecidas.
- ✓ Errores de ortografía: Errores ortográficos en la descripción de los casos de prueba.
- ✓ Errores de correspondencia: Las descripciones de varios escenarios no están en correspondencia con su nombre.
- ✓ Errores de funcionalidad: Algunos formularios no enviaban los datos correctamente y algunos botones no funcionaban.

Mediante el diseño de caso de prueba se comprobó el correcto y completo funcionamiento de las funcionalidades desarrolladas para el componente de control de acceso, lo que soluciona las no conformidades en la 3ra iteración.

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. (48)

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.

Se le realizó a la presente propuesta de solución el método de caja negra mediante la técnica de partición equivalente. Se tomó como base todos los requisitos funcionales, que generan un total de 6 Diseños de Casos de Prueba (DCP), los cuales se encuentran en el Expediente de Proyecto y Producto del CESIM. A continuación, se muestran los resultados obtenidos una vez aplicadas las pruebas:

Tabla 9. Resultados del método caja negra mediante la técnica partición equivalente en las pruebas de aceptación. Fuente: Elaboración propia.

No. Iteración	NC	Interfaz	Correspondencia	Resueltas
1	4	2	2	4
2	2	1	1	2
3	0	0	0	0
Total	6	3	3	6

Las causas de las no conformidades detectadas fueron:

- ✓ Errores de interfaz: paneles que no cumplían con las pautas de diseño establecidas.
- ✓ Errores de correspondencia: las descripciones de varios escenarios no están en correspondencia con su nombre.

3.4 Conclusiones del capítulo

- ✓ El uso de los estándares de codificación permitió organizar el código para un mejor entendimiento de todas las clases y métodos desarrollados.
- ✓ Las pruebas de software permitieron comprobar la calidad del resultado obtenido y la validez de su funcionamiento en el sistema de gestión implementado.

Conclusiones generales

Con el desarrollo del componente de control de acceso del Sistema para la gestión administrativa de programas y proyectos de ciencia, tecnología e innovación se concluye que:

- ✓ El marco teórico de la investigación, mediante el estudio y el análisis de los principales referentes teóricos en los que se sustenta la misma, da una idea de la estructura y los componentes de los sistemas homólogos, más que estos no pueden ser utilizados como propuesta de solución.
- ✓ Las técnicas de adquisición de información aplicadas permiten obtener los requisitos necesarios para poder desarrollar el componente de control de acceso a partir de la correcta gestión de los usuarios, roles y permisos.
- ✓ La definición de las herramientas, metodologías y tecnologías propician la implementación exitosa del componente de control de acceso e integración al sistema GAPID.
- ✓ Las pruebas unitarias y funcionales permiten validar el correcto funcionamiento del componente de control de acceso en el sistema GAPID.

Recomendaciones

- ✓ Creación de grupos de usuarios para la asignación de permisos, así se puede identificar permisos no solamente para cada usuario sino también para un grupo determinado de usuarios, agilizando el proceso.
- ✓ Creación de grupo de roles que tengan las mismas responsabilidades en el sistema.

Referencias bibliográficas

1. Salud, Departamento de Informática en. Hospital Italiano de Buenos Aires. [En línea] 2015. [Citado el: 20 de octubre de 2023.] http://www.hospitalitaliano.org.ar/infomed/index.php?contenido=ver_curso.php&id_curso=18084
2. Salcedo Quevedo, R. (05 de 09 de 2020). Web oficial de Universidad de San Martín de Porres. Obtenido de https://www.usmp.edu.pe/publicaciones/boletin/fia/info46/sistemas/articulo3.htm
3. PAZ-ENRIQUE, Luis Ernesto; NÚÑEZ-JOVER, Jorge Rafael; HERNÁNDEZ-ALFONSO, Eduardo Alejandro. Pensamiento latinoamericano en ciencia, tecnología e innovación: políticas, determinantes y prácticas. Desde el Sur, 2022, vol. 14, no 1.
4. CAMACHO MARÍN, Raúl; RIVAS VALLEJO, Carlos; GASPAR CASTRO, María. Innovación y tecnología educativa en el contexto actual latinoamericano. 2020.
5. MOLINA-MOLINA, Silvia, et al. Indicadores de ciencia, tecnología e innovación: hacia la configuración de un sistema de medición. Revista Interamericana de Bibliotecología, 2020, vol. 43, no 3
6. Ficha de Proyecto_GAPID Sistema para la gestión administrativa de Programas y Proyectos de Ciencia Tecnología e Innovación.pdf
7. Manual para la gestión del sistema de programas y proyectos de ciencia, tecnología e innovación.
8. SENOR, Inmaculada Carrión; ALEMÁN, José Luis Fernández; TOVAL, Ambrosio. Gestión del control de acceso en historiales clínicos electrónicos: revisión sistemática de la literatura. *Gaceta Sanitaria*, 2012, vol. 26, no 5, p. 463-468.
9. VARGAS, Christian. proyecto. *Revista de la Escuela de Arquitectura de la Universidad de Costa Rica–UCR–Número*, 2012, vol. 2, p. 1.
10. LLEDÓ, Pablo; RIVAROLA, Gustavo. *Gestión de proyectos*. Buenos Aires: Pearson Educación, 2007.
11. LÓPEZ SALAS, S. E. R. G. I. O. (2020). *Atención al cliente, consumidor y usuario*. Ediciones Paraninfo, SA.
12. González Fernández-Villavicencio, N. (2015). Qué entendemos por usuario como centro del servicio. Estrategia y táctica en marketing. *El profesional de la información, 24 (1), 5-13.*
13. ALLES, Martha Alicia. *El rol del jefe*. Ediciones Granica SA, 2008.

14. DE CONFIGURACION, VI ANÁLISIS; DE CONFIGURACIÓN, VI ANÁLISIS. INFORMÁTICA Y LINGÜÍSTICA (VI).
15. MURILLO, S.E.C., RODRÍGUEZ, J.C.R., OCHOA, I.M.V. y PINTO, A.B.M., 2019. LA SEGURIDAD INFORMÁTICA EN LOS PROCESOS ADMINISTRATIVOS. [en línea], Disponible en: http://www.cidepro.org/images/pdfs/si.pdf.
16. Javier Sánchez Pérez (2023). Los 10 mejores software de gestión de proyectos, Disponible en: https://softwarepara.net/gestion-de-proyectos/#tabla-comparativa-de-los-mejores-programas-de-gestion-de-proyectos
17. Akademos, G. d. (23 de mayo de 2019). Serie Científica. Obtenido de Serie Científica: https://publicaciones.uci.cu/index.php/serie/article/view/253
18. SOSA GONZÁLEZ, Rosel, et al. Ecosistema de Software GESPRO-16.05 para la Gestión de Proyectos. *Revista Cubana de Ciencias Informáticas*, 2016, vol. 10, p. 239-251.
19. MDN. (2022). Introducción a Django. Obtenido de Resources for Developers: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction
20. HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. *The definitive guide to Django: Web development done right*. Apress, 2009
21. w3schools.com. (2019). *www.w3schools.com*. Obtenido de www.w3schools.com:https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp.
22. HESTERBERG, Tim. Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2011, vol. 3, no 6, p. 497-526.
23. PEÑA, Dayana Mendoza; BAQUERO, L. Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso. *Univ. las Ciencias Informaticas*, 2016.
24. Kolade Chris. Visual Studio vs Visual Studio Code – What's The Difference Between These IDE Code Editors? January 31, 2023
25. Centro de Innovación para el Desarrollo y la Capacitación en Materiales Educativos. Arquitectura de los Sistemas de Administración de Bases de Datos. http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro14/index.html
26. PostgreSQL: el gestor de bases de datos a fondo https://www.ionos.es/digitalguide/servidores/know-how/postgresql/
27. ICTEA. (2022). Obtenido de ICTEA: https://www.ictea.com/cs/index.php?rp=/knowledgebase/8649/iQue-es-el-lenguaje-de-programacion-

PYTHON.html#:~:text=Python%20es%20un%20lenguaje%20de,en%20menor%20medida%2C%20programaci%C3%B3n%20funcional.

28. EGUÍLUZ PÉREZ, Javier. Introducción a JavaScript. 2012.
29. ENTERPRISE, Jubilee. *HTML 5 Manual Book*. Elex Media Komputindo, 2015.
30. GAUCHAT, Juan Diego. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012.
31. ALFONSO BENÍTEZ, Drymon. *Herramienta para generar productos de trabajos de la metodología variación AUP-UCI*. 2017. Tesis de Licenciatura. Universidad de las Ciencias Informáticas. Facultad 3.
32. GARCÍA-HOLGADO, Alicia; VÁZQUEZ-INGELMO, A.; GARCÍA-PEÑALVO, F. J. Modelo de dominio. 2022.
33. CASTILLO GUEVARA, Jorge del; TORRES PONJUÁN, Deborah. Metodología para especificar requisitos de gestión documental desde la ingeniería de requisitos. Investigación bibliotecológica, 2022, vol. 36, no 91, p. 33-48.
34. Sommerville, I. (2007). *Software Engineering.* Edtion ed.: Pearson Education.
35. MOLINA HERNÁNDEZ, Yenisel; GRANDA DIHIGO, Ailec; VELÁZQUEZ CINTRA, Alionuska. Los requisitos no funcionales de software. Una estrategia para su desarrollo en el Centro de Informática Médica. *Revista Cubana de Ciencias Informáticas*, 2019, vol. 13, no 2, p. 77-90.
36. IZAURRALDE, María Paula. Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario. *Trabajo de especialidad, febrero*, 2013.
37. Python Software Foundation. (2021). *Python*. Obtenido de https://www.python.org/
38. GAMMA, Erich, et al. *Patrones de diseño*. 2003.
39. BOTERO TABARES, Ricardo de Jesús. Patrones grasp y anti-patrones: un enfoque orientado a objetos desde lógica de programación. 2010.
40. PIÑEIRO GOMEZ, J. O. S. E. *Bases de datos relacionales y modelado de datos*. Ediciones Paraninfo, SA, 2013.
41. Arthur Hiken. Una onza de prevención: seguridad y protección a través de estándares de codificación de software, agosto de 2023
42. Pressman, R. (2007). Ingeniería de Software. Un enfoque práctico (ed.: Mc Graw Hill ed.).
43. MARIN DIAZ, Aymara; TRUJILLO CASAÑOLA, Yaimí; BUEDO HIDALGO, Denys. Estrategia de pruebas para organizaciones desarrolladoras de software. *Revista Cubana de Ciencias Informáticas*, 2020, vol. 14, no 3, p. 83-104.
44. Sommerville, Ian. 2009. Ingeniería de Software. Madrid: Pearson Education, 2009. 84-7829-074-5

45. Pressman, Roger S. 2010. Ingeniería de Software, un enfoque práctico. Madrid: Mc Graw Hill, 2010.
46. Bradanovic. Conceptos Básicos de Seguridad Informática. 2009. Disponible en: http://www.bradanovic.cl/pcasual/ayuda3.html
47. Sánchez Peño, José Manuel. 2015. Pruebas de Software. Fundamentos y Técnicas. Madrid: Universidad Politécnica de Madrid, 2015
48. GONZÁLEZ, José Félix Ponce, et al. Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. *Ibersid: revista de sistemas de información y documentación*, 2014, vol. 8, p. 73-80.

Anexos

Anexo 1. Interfaz de autenticación del sistema GAPID

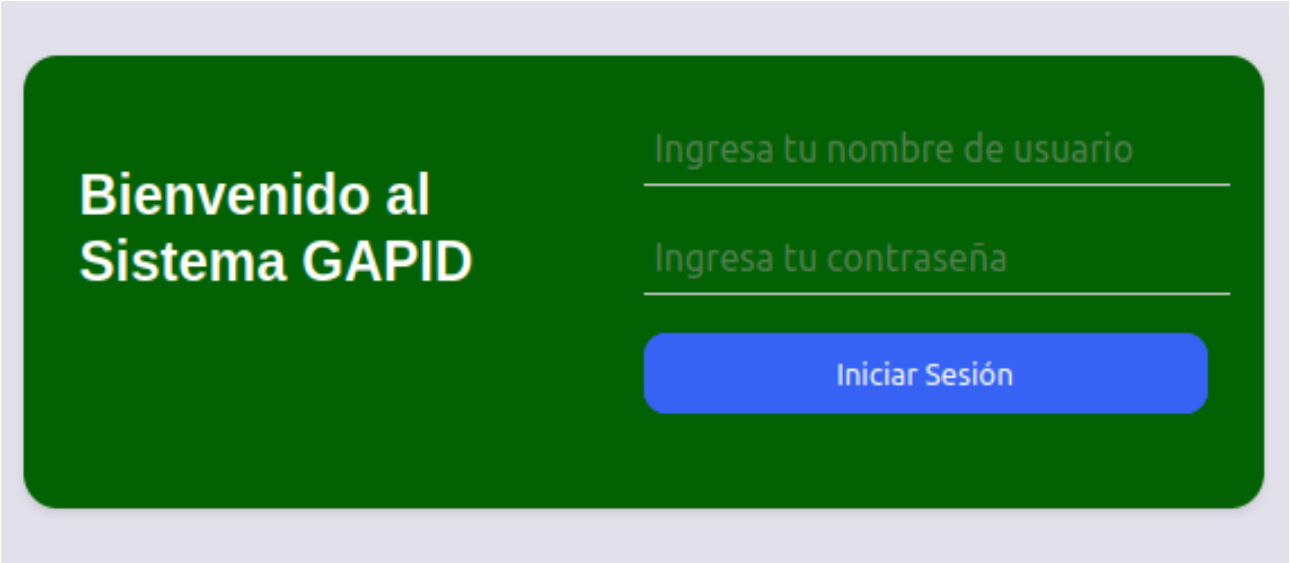


Figura 15. Autenticación de usuario. Fuente: Elaboración propia.

Anexo 2. Interfaz de los participantes con sus usuarios.

Figura 16. Listado de participantes con usuarios. Fuente: Elaboración propia.

Anexo 3. Listado de programas dentro del sistema.

Figura 17. Listado de programas. Fuente: Elaboración propia.

Anexo 4. Gestión administrativa de un programa.

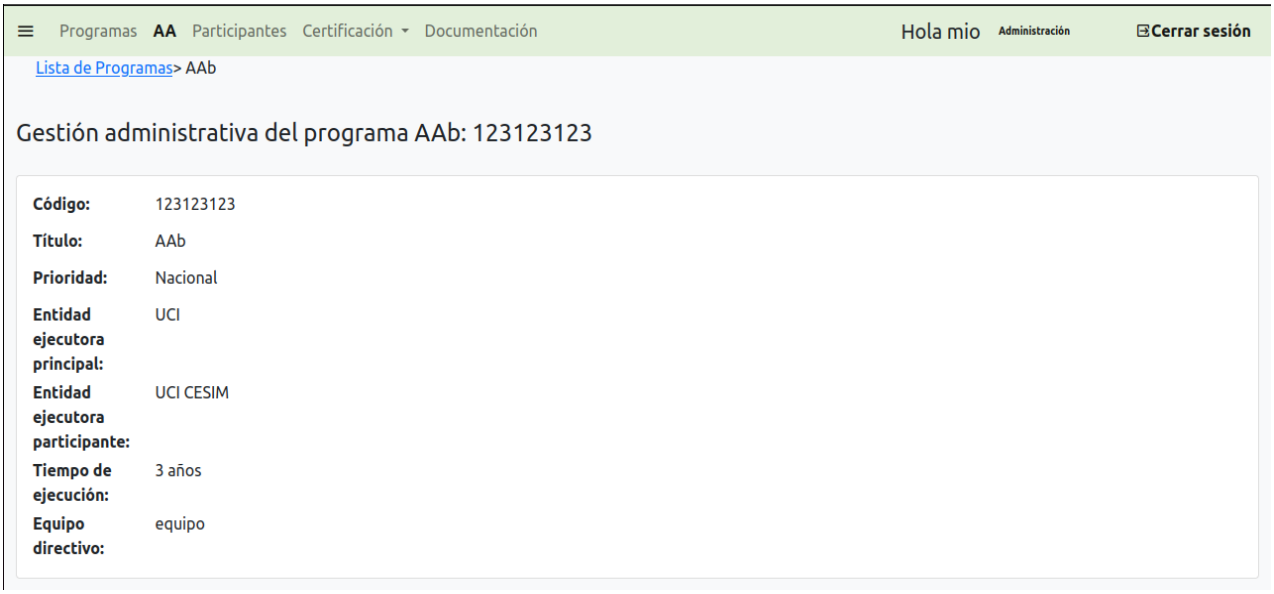


Figura 18. Gestión Administrativa de un programa. Fuente: Elaboración propia.

Anexo 5. Listado de proyectos dentro un programa.

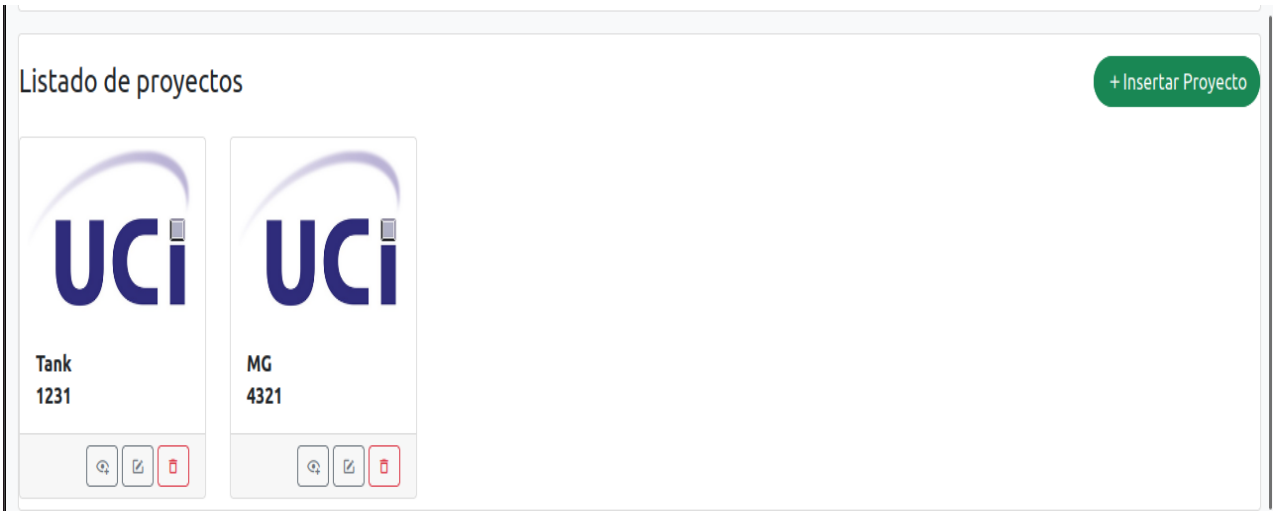


Figura 19. Listado de proyectos dentro de un programa. Fuente: Elaboración propia.

Anexo 6. Gestión administrativa de un proyecto.

Figura 20. Gestión administrativa de un proyecto. Fuente: Elaboración propia.

Anexo 7. Interfaz del listado general de participantes.

Figura 21. Listado general de participantes. Fuente: Elaboración propia.