Universidad de las Ciencias Informáticas
Dirección de Posgrado, Facultad de Tecnologías Interactivas

Panel de control de apoyo a la toma de decisiones en la formación doctoral de la UCI

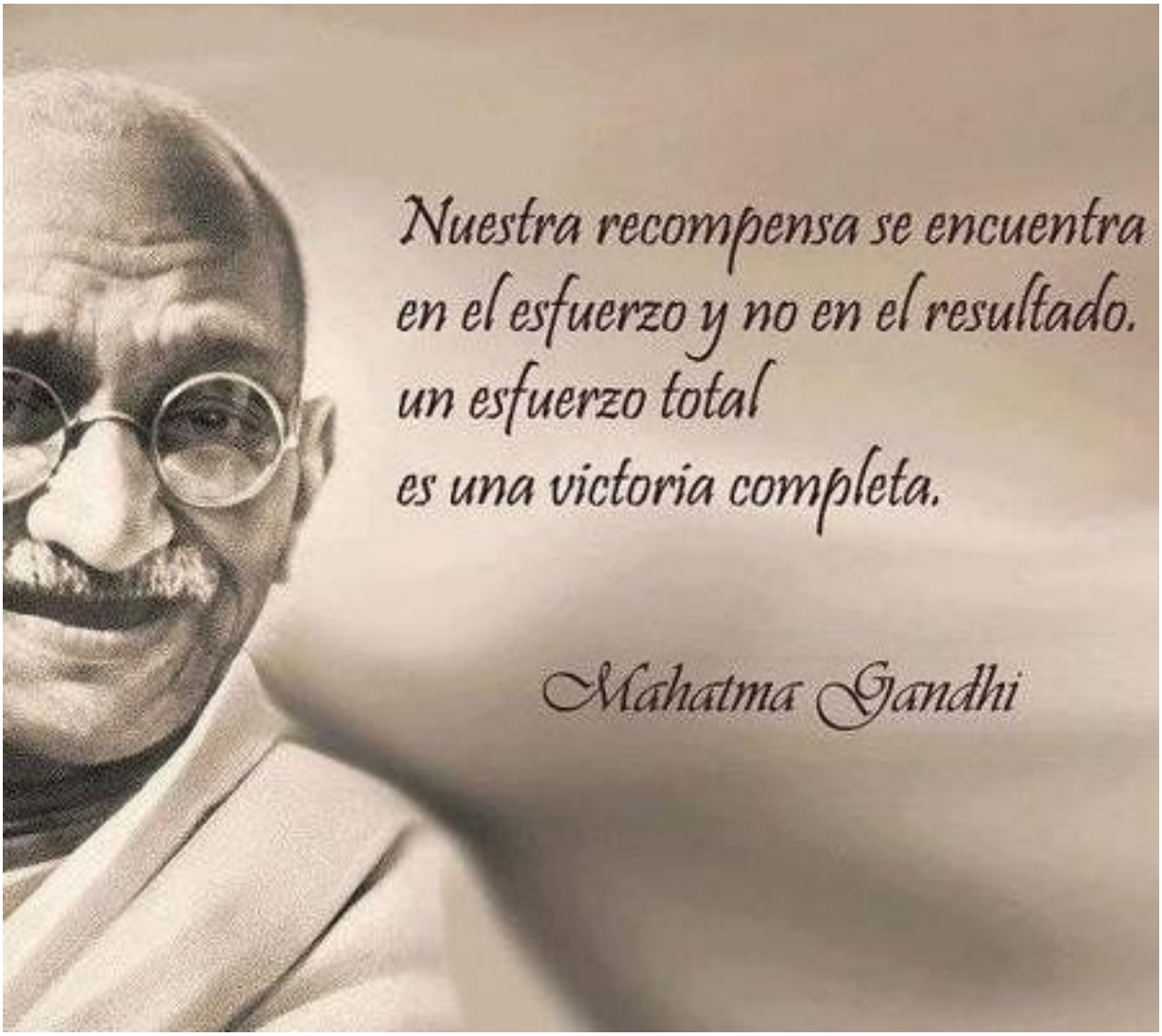
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Orlián Mesa Cáceres

Tutores: Dr. C. Yamilis Fernández Pérez

M. Sc. Sailyn Salas Hechavarria

La Habana, 2024


Nuestra recompensa se encuentra
en el esfuerzo y no en el resultado.
un esfuerzo total
es una victoria completa.
Mahatma Gandhi

Dedicatoria

Quiero dedicar esta tesis a las personas que han sido mi luz en los días más oscuros y mi motivación constante para superarme. A mis padres Milagro y Orlando, quienes han sido mi guía y sostén incondicional a lo largo de mi trayectoria como estudiante. Gracias por el sacrificio, gracias por anteponer siempre mis necesidades a las suyas y por ser los padres más maravillosos del mundo.

A mi hermano Lian Carlos, mi fuente de inspiración diaria, quien con su presencia me impulsa a superarme y ser un mejor yo. A mis abuelos, especialmente a Carlos y Edenia, por su amor incondicional, apoyo inquebrantable y por darme todo lo que soy.

A mi novia Meliza, quien ha entregado todo su amor y apoyo incondicional en las buenas y en las malas. A mi tía Nereida, quien ha sido como una segunda madre para mí, brindándome su cariño y apoyo incondicional.

A mis compañeros de estudio a lo largo de mis años de estudiante, quienes han hecho más grato cada día de aprendizaje y han compartido conmigo momentos inolvidables.

A todos ustedes, quienes han sido mis pilares, mi fortaleza y mi mayor fuente de amor y apoyo, les dedico esta tesis. Son ustedes quienes verdaderamente se merecen todo el crédito y han hecho posible que haya llegado hasta este punto. ¡Los amo!

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han apoyado en la realización de esta tesis.

En primer lugar, a mi familia, por su amor incondicional y por ser mi mayor fuente de motivación. Su aliento me ha proporcionado la fuerza necesaria para afrontar los desafíos que se presentaron en este camino.

Agradezco especialmente a mis tutoras, Sailyn, quien confió en mí para el desarrollo de esta investigación y me acompañó a lo largo de toda mi carrera universitaria, y Yamilis, por su invaluable orientación, paciencia y conocimientos. El apoyo de ambas ha sido fundamental en este proceso.

A mis profesores y compañeros de la universidad, les agradezco por su colaboración y por compartir sus ideas y experiencias, lo que ha enriquecido significativamente mi trabajo.

Finalmente, a todos aquellos que, de alguna forma, han influido en mi vida académica y personal, gracias por su apoyo y confianza.

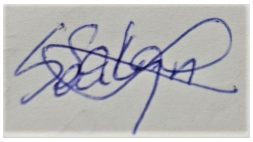
Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los 2 días del mes de diciembre del año 2024.

Orlián Mesa Cáceres
Autor

Dr. C. Yamilis Fernández Pérez
Tutora



M. Sc. Sailyn Salas Hechavarria
Tutora

Resumen

El desarrollo de un panel de control para el apoyo en la toma de decisiones en la formación doctoral en la Universidad de las Ciencias Informáticas (UCI) surge de la necesidad de gestionar de manera efectiva los indicadores clave de desempeño (KPI) en este proceso académico. La investigación tiene como objetivo desarrollar una herramienta informática que facilite el monitoreo y análisis de datos relevantes, superando las limitaciones del uso actual de software como Microsoft Excel, que presenta desafíos en la actualización y seguridad de la información. El estudio se fundamenta en un marco teórico que abarca los conceptos esenciales en la formación doctoral que facilitan la comprensión y diseño de la solución propuesta. El desarrollo estuvo guiado por la metodología XP y se seleccionó como principales tecnologías el marco de trabajo Django, React, Python y JavaScript como lenguaje de programación, el entorno integrado de desarrollo Visual Studio Code y como gestor de base de datos se utilizó MySQL. Para verificar la calidad y funcionalidad del sistema, así como el ajuste a las expectativas del cliente se realizaron pruebas unitarias y de aceptación. El panel de control implementado facilita la toma de decisiones y permite la monitorización efectiva de la formación doctoral a través de la visualización de los indicadores claves de manera gráfica y sencilla.

Palabras clave: formación doctoral, indicadores clave, panel de control, toma de decisiones.

Índice general

Índice de figuras

Índice de tablas

Introducción

La excelencia universitaria tiene como uno de sus indicadores fundamentales la cantidad de doctores en ciencia que conforman el claustro y la comunidad académica, pues la obtención del grado científico acredita la capacidad para enriquecer una rama de la ciencia mediante aportes teóricos y prácticos que hayan sido introducidos en la práctica social, o que demuestren las potencialidades de ser introducidos y generalizados sobre la base de una profunda argumentación y dominio del objeto de investigación (Educación de Posgrado, 2023).

Es esencial contar con una sólida formación de profesores y especialistas que garanticen la calidad de los procesos fundamentales que se desarrollan en la universidad. La formación doctoral contribuye directamente a estos propósitos.

La formación doctoral en Informática, inició en 2009 con la aprobación de la UCI como institución autorizada para el desarrollo de Doctorado en Informática. En enero de 2011 comienza el Programa Especial de Formación Científica en Informática (PEFCI), con el objetivo fundamental de organizar, conducir y orientar un grupo inicial de jóvenes investigadores con resultados relevantes de innovación y desarrollo y con posibilidades de lograr doctorarse en Informática a corto plazo. En enero de 2014 el consejo de dirección de la UCI aprueba el Programa de Doctorado en Informática, el cual fue ratificado por la Comisión Nacional de Grados Científicos (CNGC) del MES. En el 2018 es acreditado Programa de Excelencia por la JAN y en 2020 recibe la Mención de Honor a la Calidad del Posgrado y el Doctorado en Iberoamérica, otorgada por la Asociación Universitaria Iberoamericana de Posgrado (AUIP). En el 2021 se ratificó como institución autorizada para la formación doctoral y se aprobó la actualización del programa doctoral. Este programa ha graduado hasta diciembre de 2023 un total de 53 doctores en Ciencias Técnicas, de ellos 6 extranjeros y 4 nacionales externos a la Universidad (ibíd.).

La UCI continúa en el camino del fortalecimiento de las capacidades para la formación doctoral, como vía para incrementar el número de profesores que ostentan este grado científico y como muestra de que su capital humano está en condiciones de asumir los retos de la actual época, condicionada por el impetuoso desarrollo de la ciencia y la tecnología. Para ello se ha definido la Estrategia de Formación Doctoral de la UCI y un conjunto de indicadores para evaluar el cumplimiento de la estrategia y sus resultados en diferentes estructuras universitarias. Un control, monitoreo y gestión adecuado de estos indicadores facilitaría la toma de decisiones en este proceso. Sin embargo, a pesar de estar definidos los indicadores se dificulta el control de los mismos debido a:

- La falta de capacidad para manejar grandes conjuntos de datos.

- No hay una herramienta colaborativa de gestión de los indicadores.
- La información puede ser fácilmente perdida o compartida sin autorización.
- Dificultad para mantener una copia de seguridad confiable.

En la UCI no se cuenta con una herramienta de apoyo a la decisión en el proceso de formación doctoral que permita monitorizar, analizar y mostrar de manera visual los indicadores clave de desempeño (KPI), las métricas y datos fundamentales para hacer un seguimiento del estado del proceso de formación doctoral.
A partir de esta problemática se enuncia como **problema científico:** ¿Cómo mejorar la toma de decisiones en la formación doctoral en la UCI?
Se define como **objeto de estudio:** El proceso de toma de decisiones en la formación doctoral, precisando como **campo de acción:** Herramientas informáticas de apoyo a la toma de decisiones en el proceso de formación doctoral de la UCI.
Para resolver la situación problemática establecida se plantea como **objetivo general:** Desarrollar un panel de control de información (*dashboard*) que apoye la toma de decisiones en el proceso de formación doctoral de la UCI.
Objetivos específicos:

- Desarrollar el marco teórico referencial de la investigación.
- Definir la arquitectura, herramientas, tecnologías y metodología a emplear para el desarrollo del sistema.
- Desarrollar el módulo de gestión de la información de los doctorandos e indicadores clave para la formación doctoral.
- Desarrollar un panel de información o panel de gestión (dashboard) para monitorizar la formación doctoral, teniendo en cuenta los indicadores definidos.
- Validar las funcionalidades del panel de control a través de pruebas de software.

Métodos de investigación científica
Métodos teóricos:

- Analítico-sintético: Analizar los documentos y la información referente a la forma de gestión de la información sobre el proceso de doctorados.

Métodos empíricos:

- Entrevista: Se realizan entrevistas a los profesores implicados en el proceso, con el objetivo de precisar el problema a resolver. En el apéndice *A* de los anexos se muestran las preguntas realizadas.
- Análisis documental: Facilitó la indagación de la bibliografía relacionada con el proceso de formación doctoral en la UCI.

Estructura del documento:

- **Capítulo 1: Fundamentos y referentes teóricos sobre el objeto de estudio.** Este capítulo analiza los conceptos clave relacionados con el tema de investigación. Además, se revisa el estado actual de sistemas de gestión similares, junto con las herramientas, técnicas, tecnologías y metodologías que se emplearán para alcanzar el objetivo general.
- **Capítulo 2: Diseño de la solución propuesta al problema científico.** Se detallan las características del sistema propuesto, lo que facilitará una mejor comprensión del mismo. También se presentan los artefactos generados por la metodología de desarrollo, que se crean en las diferentes etapas del proceso de desarrollo de software seleccionado.
- **Capítulo 3: Validación de la solución propuesta.** En este capítulo, se implementan las funcionalidades solicitadas por el cliente y se realizan pruebas de aceptación para comprobar el correcto funcionamiento del sistema y la satisfacción del cliente.

CAPÍTULO 1

Fundamentos y referentes teóricos sobre el objeto de estudio

En este capítulo se establecen los fundamentos teóricos y tecnológicos que sustentan el desarrollo de la solución informática propuesta en la tesis. Se definen conceptos claves asociados al dominio del problema, para brindar un marco conceptual sólido. Además, se realiza un profundo análisis de sistemas homólogos existentes, con el fin de identificar características y funcionalidades relevantes.

1.1. Conceptos asociados al dominio del problema

Definir y comprender los conceptos asociados al dominio del problema es fundamental para establecer un marco teórico claro que guíe el análisis y la propuesta de solución. En el contexto de la excelencia universitaria, uno de los indicadores más relevantes es la presencia de doctores en ciencia dentro del claustro académico. La obtención de este grado no solo acredita la capacidad de un individuo para contribuir al avance de una disciplina, sino que también refleja el compromiso de la institución con el desarrollo científico y tecnológico.

La Universidad de Ciencias Informáticas (UCI) se ha enfocado en fortalecer su capacidad para formar doctores, reconociendo que este proceso es crucial para enfrentar los desafíos actuales en un entorno caracterizado por el acelerado avance de la ciencia y la tecnología. Según Saborido, 2018, la formación doctoral en Cuba es un componente esencial para la sustentación del potencial científico del país, particularmente en las universidades. La presencia de un número competitivo de doctores es determinante para asegurar la calidad de la educación superior y para la producción de resultados científicos que impacten en la economía y contribuyan a satisfacer necesidades sociales y culturales. Esto tiene, incluso, una trascendencia política y marca la imagen exterior de la educación superior cubana, que se intenta demeritar sin éxito.

El programa de formación doctoral se centra en el doctorando, definido como el graduado de nivel superior que, tras cumplir con los requisitos establecidos y tener un tema de doctorado aceptado en una de las líneas de investigación, es aprobado por la institución autorizada para formalizar su matrícula (Estado, 2019). Una vez culminado su proceso de formación, el doctorando alcanza finalmente el grado de doctor.

El doctorado es el último nivel del sistema educativo formal y marca el inicio de la carrera académica de

un investigador. Sin embargo, surge la pregunta sobre el propósito del doctorado si el tesista ya posee los conocimientos necesarios. Hoy por hoy, no hay dudas de que el doctorado es un escalón más en el proceso formativo de una persona (Fernández, 2018).

A partir de estos elementos, la UCI ha definido una estrategia de formación doctoral que busca consolidar la capacitación de doctores en ciencia. Esta estrategia tiene como objetivo elevar la pertinencia y el impacto social de la formación doctoral, enfocándose en la calidad de las investigaciones realizadas y en el desempeño investigativo de los egresados. Se espera que los graduados no solo aporten a su especialidad, sino que también contribuyan al desarrollo económico, social y cultural del territorio y del país. Asimismo, la estrategia se centra en asegurar la excelencia mediante el mejoramiento continuo de la calidad en el proceso de formación doctoral (Educación de Posgrado, 2023).

En este contexto, como parte de la estrategia se definieron un conjunto de indicadores claves para la evaluación del desempeño agrupados en ocho líneas: Proyección Institucional 2030, Programas de Doctorado, Doctores y Tutores, Ingreso y Doctorandos, Infraestructura, Alianzas interinstitucionales e internacionalización, Impacto social de los resultados e Información, comunicación, informatización (transformación digital) (ibíd.). La presente investigación abarca los indicadores asociados a Doctores, Programas y Doctorandos. En el anexo G.1 se muestra un resumen de los indicadores empleados.

El análisis de los indicadores es fundamental para la toma de decisiones en el proceso de formación doctoral. En este sentido, un *dashboard* o panel de control de información se convierte en una herramienta esencial para visualizar los indicadores establecidos. Este tipo de herramienta presenta información relevante de manera que permite un seguimiento efectivo en tiempo real. Para maximizar su utilidad y eficacia, un *dashboard* debe ofrecer abundante información mediante elementos visuales compactos, comunicando con claridad e inmediatez el comportamiento de los datos, lo que facilita un análisis posterior. Este objetivo exige un diseño que aproveche el poder de la percepción visual, permitiendo el procesamiento eficiente de grandes volúmenes de información (Few, 2024).

A continuación se muestra en la figura 1.1 la relación entre los diferentes conceptos asociados al dominio del problema. Esta figura ilustra cómo los diversos elementos interactúan y se interrelacionan, proporcionando una comprensión más profunda del contexto y las variables involucradas.

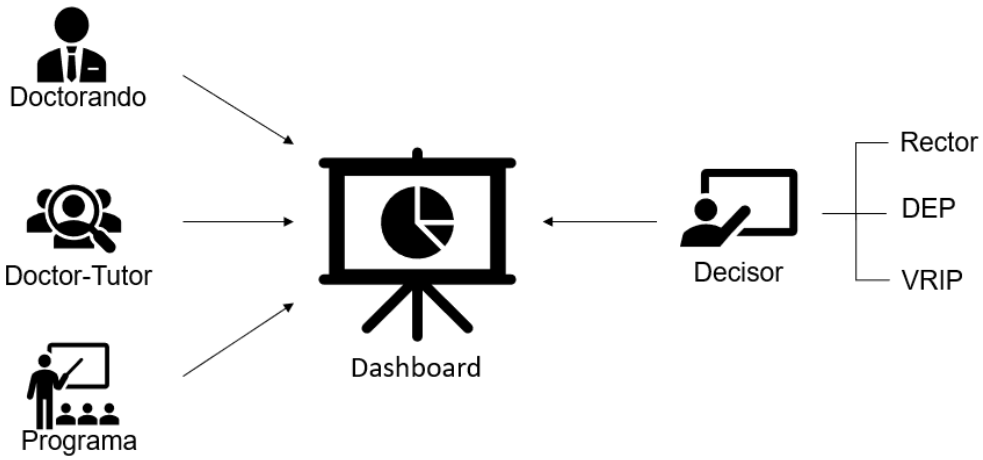


Figura 1.1. Relación entre los conceptos asociados al dominio del problema

1.2. Análisis de sistemas homólogos

Los paneles de control de información se han vuelto herramientas esenciales y estratégicas en la era digital. Permiten a las empresas supervisar en tiempo real métricas clave, tomar decisiones fundamentadas, optimizar la eficiencia operativa y alinear sus acciones con las metas organizacionales. Estos paneles transforman una inmensa cantidad de datos en información clara y útil, proporcionando la visibilidad y flexibilidad necesarias para mantener la competitividad en un entorno empresarial cada vez más dinámico.(RockContent, 2021).

Power BI:

Power BI es un conjunto de herramientas basado en la inteligencia empresarial o *Business Intelligence* y que funciona en la nube de Microsoft. Este servicio permite acceder a los datos de forma rápida, por medio de un sistema predictivo con información tangible (Becker y Gould, 2019).

El uso de paneles de control como Power BI facilita un análisis de datos simplificado, permitiendo evaluar el crecimiento, así como las fortalezas y debilidades de la organización, con el fin de tomar decisiones más acertadas. Estos paneles permiten detectar patrones sutiles en los datos a través de métodos rápidos, como la previsión y la agrupación, sin necesidad de conocimientos en programación, lo que facilita la predicción de resultados estratégicos. Además, Power BI permite la integración con Excel, posibilitando a las empresas exportar y conectar toda la información de forma sencilla para realizar análisis en tiempo real.

Esta herramienta es adaptable a diversas áreas de una empresa, tales como contabilidad, recursos humanos, logística y producción, lo que la convierte en una solución versátil. Al estar alojada en la nube, Power BI facilita el acceso y el intercambio de información en tiempo real con socios y clientes, promoviendo una comunicación más ágil. Las organizaciones pueden optar por la versión gratuita para conectar distintas fuentes de datos y generar informes, o elegir la versión PRO, que ofrece mayor almacenamiento y actualización de datos.

Power BI permite a las empresas gestionar grandes volúmenes de datos (big data), descubrir información valiosa, extraer conclusiones y respaldar decisiones informadas. Al trabajar en un entorno en la nube, la herramienta elimina los problemas de capacidad y permite la creación de paneles y visualizaciones interactivas, intuitivas y atractivas. Además, tiene la capacidad de conectarse con bases de datos en diversos formatos, como XML, JSON, SQL Server y Azure, así como con servicios en línea, incluyendo Google Analytics, Google Drive, Salesforce y redes sociales, consolidando todos estos datos en un único cuadro de mando para un análisis más exhaustivo (Becker y Gould, 2019).

Tableau:

La plataforma de Tableau es una de las opciones de inteligencia de negocios que se encuentran liderando el mercado. Utiliza diferentes tipos de datos de numerosos sistemas y los convierte en información útil de manera rápida y sencilla (Milligan, 2019).

Tableau ofrece una serie de ventajas que la posicionan como una herramienta ideal para la gestión y visualización de datos en diversas estructuras empresariales. Permite visualizar los datos de forma segura, ya sea en un navegador, en dispositivos móviles, en el escritorio o en una aplicación, adaptándose a las necesidades específicas de cada usuario. Además, brinda la flexibilidad de integrar la infraestructura de datos tanto en instalaciones físicas como en la nube, proporcionando opciones según los requerimientos de cada organización.

Esta plataforma es adecuada tanto para científicos de datos como para usuarios comerciales, sin importar su nivel de conocimientos técnicos, ya que facilita la exploración y administración de datos de manera intuitiva. Tableau se destaca por ser segura, flexible y eficaz, respaldada por muchos años en el mercado que avalan su fiabilidad. Asimismo, ofrece recursos de capacitación y cuenta con una comunidad activa que brinda soporte, además de ser compatible con diversas fuentes de datos, lo que amplía su versatilidad en múltiples entornos empresariales (ibíd.).

Sistema de Gestión de la Nueva Universidad (SIGENU):

El Sistema de Gestión de la Nueva Universidad (SIGENU) fue implementado en 2004 por el Ministerio de Educación Superior de Cuba, como respuesta a la necesidad de un sistema eficiente para el almacenamiento y procesamiento de la información académica en el ámbito universitario. Desde su creación, SIGENU ha evolucionado para convertirse en una herramienta integral que facilita la gestión de la actividad docente en todo el país.

En 2006, se introdujeron los primeros módulos en la Universidad Tecnológica de La Habana (CUJAE), y para 2007, se realizó la matrícula a través del sistema, lo que marcó un hito en la informatización de la gestión académica. Actualmente, SIGENU opera en todos los centros adscritos al Ministerio, ofreciendo módulos como Portal, Secretaría, Profesores y Administrador, que permiten una gestión centralizada y accesible.

La implementación de SIGENU ha permitido que estudiantes y profesores accedan a información relevante en tiempo real, como evaluaciones y registros académicos, contribuyendo a una toma de decisiones más eficiente a nivel institucional. Además, facilita procesos como la pre-matrícula en línea y el registro de evaluaciones, optimizando la administración académica.

Reconocido por su innovación, SIGENU ha sido galardonado en múltiples ocasiones, destacando su papel crucial en la transformación digital de la educación superior cubana. A medida que se expande su uso, se espera que continúe mejorando la calidad de la gestión educativa y contribuya significativamente al desarrollo del sistema de educación superior en el país (Antón Rodríguez, 2023).

Resultado del análisis de soluciones homólogas

Tras un exhaustivo análisis de sistemas orientados al mejoramiento de la gestión de información en ámbitos académicos y otros sectores, se concluyó que no existe un sistema que satisfaga completamente las necesidades del proceso de formación doctoral de la UCI. Aunque inicialmente se contempló el desarrollo de un sistema de gestión de información vinculado a un tablero de operaciones utilizando Power BI, el estudio de sistemas homólogos reveló que esta opción no era viable.

Por otro lado, el uso de Power BI y Tableau para el panel de control presenta inquietudes de seguridad, ya que requieren acceso total a la base de datos del proceso de formación doctoral de la UCI. Además, estas herramientas implican altos costos de licencias y dependen de una conexión a internet, lo que limita su accesibilidad en ciertos momentos. Ambas plataformas también presentan una curva de aprendizaje considerable, lo que exige un gasto significativo de tiempo y recursos, y sus limitaciones en la personalización dificultan su adaptación a las necesidades específicas del cliente. Es importante destacar que, desde Cuba, el acceso a estos recursos se ve comprometido por el bloqueo que enfrenta la isla, complicando aún más su viabilidad, así como las actualizaciones y el soporte técnico.

El Sistema de Gestión de la Nueva Universidad (SIGENU) presenta funcionalidades y características que se alinean con los procesos y requerimientos de las instituciones que lo respaldan. Este sistema se centra principalmente en la gestión de la información relacionada con la actividad académica de los cursos de pregrado, dejando de lado la formación doctoral.

Finalmente, tras evaluar los diversos sistemas, se decidió no implementar ninguno como solución para esta investigación; sin embargo, se identificaron valiosas ideas y características que pueden enriquecer el desarrollo del nuevo software. En la tabla 1.1 se muestra un resumen de la comparación entre los sistemas analizados.

Sistemas de gestión	Aplicación web o de escritorio	Autenticación de usuario	Pago	Bloqueada	Gestión de estadísticas	Elemento KPI
PowerBI	Ambas	Sí	Sí	Sí	Sí	Indicadores generales
Tableau	Web	Sí	Sí	Sí	Sí	Indicadores generales
SIGENU	Web	Sí	No	No	Sí	Indicadores del proceso académico de pregrado

Tabla 1.1. Comparativa de sistemas de gestión

1.3. Herramientas y tecnologías

En este epígrafe, se detallan las diferentes herramientas y tecnologías de cabecera que son utilizadas en la actualidad para el desarrollo de un entorno basado en la web. Se presenta la información necesaria a los desarrolladores sobre dichas herramientas y tecnologías, así como sus ventajas y desventajas.

1.3.1. Lenguaje del lado del servidor

El lenguaje del lado del servidor, conocido como *backend*, desempeña un papel crucial en el desarrollo web. Se refiere a la parte del sistema que maneja la lógica de negocio, el procesamiento de datos y la interacción con servidores y bases de datos. Aunque esta capa no es visible para los usuarios finales, es la responsable de ofrecer las características y funcionalidades esenciales de la aplicación en el servidor. Sin el *backend*, la experiencia del usuario sería limitada, ya que es donde se toman decisiones y se gestionan los datos que alimentan la interfaz que los usuarios ven. (GeeksforGeeks, 2023).

Las responsabilidades del *backend* son diversas y fundamentales para el funcionamiento de una aplicación web. En primer lugar, se encarga de gestionar la base de datos, lo que implica almacenar, recuperar, actualizar y eliminar datos de manera segura. Esta gestión es crucial para garantizar la integridad y la confidencialidad de la información.

El *backend* procesa la lógica de la aplicación que incluye la implementación de funcionalidades esenciales como la autenticación de usuarios y la generación de contenido dinámico. En esta capa se define cómo interactúan los diferentes componentes del sistema.

Un aspecto importante del lenguaje del lado del servidor es la exposición de APIs. Estas interfaces de programación permiten que el frontend y otras aplicaciones externas se comuniquen con el *backend* de manera eficiente. De esta forma, se facilita la integración y la interoperabilidad entre diferentes sistemas.

Esta parte del sistema es el encargado de manejar las solicitudes de los usuarios. Recibe las peticiones HTTP del frontend, las procesa y devuelve las respuestas adecuadas. Se encarga de integrar servicios externos, conectándose con otros sistemas, APIs y microservicios para enriquecer la funcionalidad de la aplicación. Esta capacidad de interacción con el entorno externo es esencial para ofrecer una experiencia completa y satisfactoria al usuario.

Algunos de los lenguajes de programación utilizados en el desarrollo backend son PHP, Java y Python. Estos lenguajes son populares por su robustez, flexibilidad y amplia comunidad de soporte, lo que facilita el desarrollo de aplicaciones web eficientes y escalables.

PHP:

Diseñado por Rasmus Lerdorf y lanzado en 1995, PHP o Hypertext Preprocessor es un lenguaje de secuencias de comandos del lado del servidor para el desarrollo web y también se utiliza como lenguaje de programación de propósito general. Los lenguajes de implementación incluyen Perl, C, C++, Java (Pérez; Quispe; Mullicundo y Lamas, 2021).

El lenguaje PHP se destaca por su facilidad de aprendizaje, ya que a lo largo de su desarrollo se han simplificado varias especificaciones, como la definición de variables primitivas y su uso en arreglos. Este lenguaje está orientado al desarrollo de aplicaciones web dinámicas, lo que le permite acceder fácilmente a datos almacenados en bases de datos. Además, PHP ofrece una excelente conectividad, siendo capaz de conectarse a casi todos los motores de bases de datos utilizados hoy en día, destacando su eficiencia con PostgreSQL y MySQL.

Otro aspecto importante de PHP es que el código fuente escrito en este lenguaje no es visible para los clientes ni para los navegadores web, ya que el servidor se encarga de ejecutar el código y enviar el resultado en forma de HTML. Esto proporciona una programación más confiable y segura. PHP también cuenta con una amplia documentación disponible en su sitio web oficial, donde se pueden encontrar descripciones y explicaciones de las funciones del sistema. Además, el lenguaje tiene un gran potencial de extensibilidad a través de módulos y extensiones, permitiendo a los desarrolladores ampliar sus funcionalidades según las necesidades de sus proyectos.

PHP presenta una serie de ventajas que lo convierten en una opción popular para el desarrollo web. Se trata de un lenguaje de programación libre y abierto, lo que permite a los desarrolladores utilizarlo sin coste alguno y adaptarlo según sus necesidades. Su curva de aprendizaje es muy baja, facilitando que los nuevos programadores comiencen rápidamente a escribir código. Además, se configura de manera rápida y sencilla en su entorno de desarrollo, lo que lo hace accesible para una amplia variedad de usuarios.

Posee un excelente integración con bases de datos, que permite un acceso eficiente a los datos. Con una comunidad extensa que respalda su uso, los desarrolladores pueden encontrar recursos y soporte fácilmente, lo que incrementa la demanda de PHP en el mercado laboral. Además, es un lenguaje multiplataforma, lo que significa que puede ejecutarse en diversos sistemas operativos y está especialmente diseñado para el desarrollo de aplicaciones web dinámicas, permitiendo ejecutar código del lado del servidor y generar HTML para los navegadores.

A pesar de poseer múltiples ventajas, PHP también presenta algunas desventajas que es importante considerar. Uno de sus principales inconvenientes es que el código fuente no puede ocultarse de manera eficiente, lo que puede resultar en problemas de seguridad si no se protege adecuadamente. La falta de concentración en el código puede dificultar su lectura y mantenimiento, lo que puede ser un desafío para los desarrolladores a medida que el proyecto crece. Además, para garantizar la seguridad del código, es necesario crear un servidor propio, lo que implica más trabajo y recursos, y puede desincentivar a algunos desarrolladores o pequeñas empresas.

Un aspecto negativo del lenguaje PHP es que las variables no son tipadas, lo que complica la asistencia que los entornos de desarrollo integrados (IDEs) pueden ofrecer en la tipificación del código. Esta falta de tipado puede conducir a errores en tiempo de ejecución, dificultando la depuración y aumentando la posibilidad de fallos en la aplicación. A pesar de los inconvenientes que puede presentar, es importante resaltar que muchos de ellos, pueden ser gestionadas mediante la adopción de buenas prácticas de programación y un enfoque proactivo en la seguridad, lo que permite a los desarrolladores aprovechar al máximo las ventajas que PHP ofrece.

Java:

Java es un lenguaje de programación altamente versátil y ampliamente utilizado para desarrollar aplicaciones web. A lo largo de más de dos décadas, ha sido una opción preferida entre los desarrolladores, y hoy en día, millones de aplicaciones Java están en funcionamiento en diversas industrias. Con su naturaleza multiplataforma, orientada a objetos y centrada en la red, Java no solo se presenta como un lenguaje de programación, sino también como una plataforma robusta por sí misma. Es reconocido por su rapidez, seguridad y confiabilidad, lo que lo convierte en la elección ideal para crear una amplia gama de soluciones, desde aplicaciones móviles y software empresarial hasta sistemas de macrodatos y tecnologías de servidor (Amazon, 2024).

Presenta numerosas ventajas, especialmente en el ámbito del desarrollo web. Una de sus principales fortalezas es la arquitectura escalable que proporciona, lo que permite a las aplicaciones manejar grandes volúmenes de tráfico y datos sin comprometer el rendimiento. La plataforma *Java Enterprise Edition* (Java EE) incluye características que facilitan la creación de aplicaciones web distribuidas, permitiendo que estas se implementen en múltiples servidores y procesadores. Esto no solo mejora la disponibilidad de las aplicaciones, sino que también optimiza su rendimiento en situaciones de alta demanda.

Java se destaca por sus robustas características de seguridad, cruciales para proteger las aplicaciones web contra ataques externos.Proporciona herramientas avanzadas para la autenticación y autorización, así como medidas de seguridad contra ataques como la inyección SQL y *Cross-Site Scripting* (XSS). La capacidad de manejar la gestión de sesiones refuerza la seguridad de las aplicaciones. Asimismo, Java es un lenguaje de programación multiplataforma, lo que significa que las aplicaciones pueden ejecutarse en cualquier sistema que tenga instalada la Máquina Virtual de Java (JVM), eliminando la necesidad de reescribir el código para diferentes plataformas y sistemas operativos. Esto es especialmente valioso para los desarrolladores que buscan crear aplicaciones que funcionen de manera consistente en diversos entornos.

Sin embargo, Java también presenta desventajas que pueden ser relevantes para quienes consideran su uso. En primer lugar, su complejidad puede resultar un desafío para los nuevos programadores, lo que implica una curva de aprendizaje más empinada en comparación con otros lenguajes de programación web. Esto puede llevar más tiempo para dominar el lenguaje y sus herramientas, lo que puede desmotivar a algunos desarrolladores. A pesar de que proporciona marcos de trabajo para mejorar la eficiencia en el desarrollo, el proceso puede requerir más tiempo en comparación con otros lenguajes debido a su complejidad inherente. Las aplicaciones web desarrolladas en Java pueden exigir más recursos de hardware, como memoria y potencia de procesamiento, lo que puede suponer un problema para las empresas con presupuestos limitados o que no cuentan con hardware de alta gama. Aunque Java es un lenguaje multiplataforma, pueden surgir problemas de compatibilidad entre diferentes versiones de Java y navegadores web, lo que podría complicar la implementación y ejecución de aplicaciones. Al ser una tecnología comercial, pueden existir costos asociados con la adquisición de herramientas y licencias necesarias para el desarrollo de aplicaciones web en Java (Solís Macias, 2023).

Python:

Python es un lenguaje de programación de alto nivel, interpretado y de código abierto, creado por Guido van Rossum en 1991. Su diseño se centra en la simplicidad y la legibilidad del código, lo que permite a los desarrolladores expresar conceptos en menos líneas en comparación con otros lenguajes como C++ o Java. Esto hace que Python sea más accesible tanto para principiantes como para expertos, facilitando su aprendizaje y uso.

Este lenguaje es increíblemente versátil y puede aplicarse en una amplia variedad de campos, desde el desarrollo web y la creación de software hasta la ciencia de datos, la inteligencia artificial y el aprendizaje automático. Gracias a su extensa biblioteca de módulos y herramientas, Python permite a los desarrolladores llevar a cabo tareas complejas de manera más eficiente y con menos esfuerzo. Además, la comunidad de Python es robusta y muy activa, lo que garantiza un constante flujo de recursos, módulos y bibliotecas externas que amplían sus capacidades y facilitan el desarrollo desde el primer momento (Python, 2021).

Python ofrece numerosas ventajas, lo que lo convierte en una opción popular entre desarrolladores de diversos niveles de experiencia. Uno de los principales aspectos en los que destaca Python es su sintaxis simple y fácil de entender. Este estilo de codificación no solo facilita la lectura y escritura del código, sino que también acelera el desarrollo y mejora la colaboración en equipos. Los desarrolladores pueden enfocarse más en resolver problemas y menos en la complejidad del lenguaje, lo que resulta en una mayor productividad. Además, Python cuenta con una comunidad activa que contribuye continuamente con bibliotecas, tutoriales y soporte en foros, brindando un respaldo invaluable para el aprendizaje y la resolución de problemas.

Posee una extensa biblioteca estándar, que cubre diversas áreas, desde la manipulación de archivos hasta el desarrollo web. Esta variedad permite a los desarrolladores aprovechar herramientas y funcionalidades existentes sin tener que reinventar la rueda, lo que acelera aún más el proceso de desarrollo. Además, al ser un lenguaje multiplataforma, el código escrito en Python puede ejecutarse en diferentes sistemas operativos sin necesidad de modificaciones importantes. Esta característica simplifica el desarrollo de aplicaciones que

pueden utilizarse en una amplia gama de entornos. La facilidad de escritura y la sintaxis concisa también hacen de Python una opción ideal para el desarrollo rápido de prototipos, permitiendo a los desarrolladores experimentar con ideas antes de comprometerse con implementaciones más complejas. Además, Python se integra fácilmente con otros lenguajes, como C y C++, lo que permite optimizar el rendimiento cuando es necesario.

Sin embargo, Python también tiene algunas desventajas que pueden ser relevantes dependiendo del contexto de uso. Como lenguaje interpretado, puede ser más lento en comparación con lenguajes compilados como C++ o Java, lo que puede ser un inconveniente en aplicaciones que requieren un rendimiento extremadamente rápido, como el desarrollo de juegos o tareas de cómputo intensivo. Aunque la gestión automática de memoria es una ventaja para muchos desarrolladores, puede ser una desventaja en aplicaciones que requieren un control preciso sobre los recursos de memoria, ya que Python no ofrece la misma flexibilidad que los lenguajes de bajo nivel. A pesar de que existen frameworks como Kivy o BeeWare, Python no es considerado el lenguaje principal para el desarrollo de aplicaciones móviles, donde otros lenguajes como Swift o Kotlin son más comunes y preferidos (Fontecha Zabaleta, 2017). Aunque Python tiene algunas limitaciones, sus ventajas en facilidad de uso y versatilidad lo convierten en una herramienta valiosa en el arsenal de cualquier desarrollador.

Selección del lenguaje del lado del servidor:

Luego de un exhaustivo análisis de los principales lenguajes de programación para el desarrollo del lado del servidor, se determina que Python, en su versión 3.11.5, es la opción más adecuada para el desarrollo del proyecto. Este lenguaje destaca por su sintaxis simple y fácil de entender, lo que facilita la lectura, el mantenimiento y la colaboración en el código. La extensa biblioteca estándar que posee, junto con su activa comunidad de desarrolladores, proporcionan un sólido ecosistema de herramientas y recursos que aceleran el desarrollo y simplifican la resolución de problemas. Asimismo, la capacidad que posee para integrarse con otros lenguajes y gestores de bases de datos como MySQL; así como su sencillez, versatilidad y amplio respaldo lo convierten en la opción más apropiada para este proyecto.

1.3.2. Lenguaje del lado del cliente

En el desarrollo web, el "lado del cliente"se refiere a todo lo que ocurre en la aplicación web desde la perspectiva del usuario final, es decir, en el dispositivo que utiliza para acceder a la aplicación. Esto abarca todo lo que el usuario puede ver y con lo que puede interactuar, incluyendo texto, imágenes y la interfaz de usuario en general. Además, cualquier acción que la aplicación realice en el navegador del usuario también se considera parte del lado del cliente.

Los lenguajes de marcado, como HTML y CSS, son fundamentales para el lado del cliente, ya que son interpretados por el navegador para renderizar el contenido y el diseño de la página. En la actualidad, muchos desarrolladores están adoptando un enfoque más centrado en el cliente al crear aplicaciones web, trasladando una parte considerable de la lógica de negocio al lado del cliente. Por ejemplo, en aplicaciones web modernas, es común que la lógica para manejar páginas web dinámicas se ejecute en el navegador del usua-

rio, lo que mejora la velocidad y la interactividad. Este tipo de procesos del lado del cliente se implementan principalmente utilizando JavaScript.

El lado del cliente a menudo se asocia con el término *frontend*, aunque hay una distinción importante entre ambos conceptos. El lado del cliente se refiere específicamente a dónde se ejecutan los procesos, mientras que el *frontend* se centra en los tipos de procesos y la presentación que se ejecutan en el lado del cliente. En este sentido, aunque los dos términos están interrelacionados, cada uno tiene su propio enfoque dentro del desarrollo web (Pérez; Quispe; Mullicundo y Lamas, 2021).

HTML:

HTML (Lenguaje de marcado de hipertexto o HyperText Markup Language por sus siglas en inglés) es un lenguaje descriptivo que especifica la estructura de las páginas web. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc (A. C. Luna, 2024)

CSS:

CSS (Cascading Style Sheets - en español Hojas de Estilo en Cascadas) es un lenguaje de estilo utilizado para describir la presentación de documentos escritos en lenguajes de marcado como HTML y XHTML. Aunque no se considera un lenguaje de programación, CSS es fundamental para el diseño web, ya que permite definir la apariencia de los elementos en pantalla. Su uso no se limita a HTML, sino que también se puede aplicar a otros lenguajes de marcado como SVG y XML. Con CSS, los desarrolladores pueden controlar aspectos como el color del texto, el estilo de las fuentes, los espacios entre párrafos, así como el tamaño y la disposición de las columnas. Además, CSS permite que una misma página web se visualice de manera óptima en diferentes dispositivos y tamaños de pantalla, asegurando una experiencia de usuario consistente y atractiva. CSS es la herramienta clave para dar estilo y estructura a los documentos HTML, definiendo cómo deben mostrarse en diferentes contextos. (Qahramon o'g'li et al., 2024).

JavaScript:

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo web, pensado para agregar potencial de interacción y dinamismo a las páginas web.

Es compatible con todos los navegadores modernos y se ejecuta del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario final. Además de su uso en el desarrollo web, JavaScript también se utiliza en aplicaciones de servidor (con tecnologías como Node.js) y en el desarrollo de aplicaciones móviles y de escritorio.

JavaScript proporciona una amplia gama de funcionalidades, como manipulación del DOM (Document Object Model) para interactuar con elementos de una página web, manejo de eventos, comunicación con servidores a través de AJAX, creación de animaciones, validación de formularios y mucho más. Es un lenguaje flexible y dinámico que permite a los desarrolladores crear experiencias interactivas y ricas en contenido (F. Luna, 2019).

Se decide elegir HTML, CSS y JavaScript para el desarrollo del *frontend* debido a sus numerosas ventajas. HTML proporciona la estructura básica de las páginas web, permitiendo organizar el contenido de manera efectiva, mientras que CSS añade estilo y diseño, garantizando que el contenido sea accesible en diferentes dispositivos. Por su parte, JavaScript añade interactividad y dinamismo, permitiendo manipular el DOM y crear experiencias de usuario ricas y atractivas. Juntos, estos lenguajes forman una combinación poderosa que facilita la creación de aplicaciones web completas y funcionales.

1.3.3. Herramienta CASE Visual Paradigm

CASE, que significa *Computer Aided Software Engineering* o Ingeniería de Software Asistida por Computadora, en idioma español, se refiere al uso de herramientas informáticas para organizar y gestionar el desarrollo de software. El propósito de las herramientas CASE es brindar apoyo a lo largo del ciclo de vida del proyecto, abarcando todos sus componentes y facilitando la planificación, diseño, implementación y mantenimiento del software (Vázquez; González; Farías; Aguayo y Méndez, 2018).
Visual Paradigm es una herramienta que se ha destacado como uno de los más completos y versátiles instrumentos de modelado y diseño de software en el mercado. Esta aplicación proporciona una amplia gama de funcionalidades que la convierten en un aliado estratégico para los equipos de desarrollo que enfrentan los desafíos del software moderno. Entre sus principales ventajas, destaca su capacidad para diseñar software utilizando diversos estándares de modelado, como UML, ERD, DFD y SoaML, lo que permite a desarrolladores y arquitectos crear representaciones visuales estandarizadas de la estructura, flujos y funcionalidades de los sistemas en desarrollo.
La documentación gráfica que ofrece Visual Paradigm facilita la comprensión y el análisis del software tanto para el equipo de trabajo como para los stakeholders. Su capacidad para generar código a partir de los modelos diseñados acelera significativamente el proceso de desarrollo, permitiendo a los programadores comenzar con una base sólida y bien definida en lugar de tener que construir todo desde cero. También ofrece la posibilidad de realizar ingeniería inversa, generando diagramas a partir del código fuente existente. Al ser un programa multiplataforma, Visual Paradigm es accesible para equipos de desarrollo que operan en diferentes entornos de trabajo, como Windows y Linux (Paradigm, 2024).

1.3.4. Sistema gestor de base de datos MySQL

Un sistema gestor de base de datos (SGBD) es un software diseñado para administrar y controlar bases de datos, permitiendo el almacenamiento y organización de grandes cantidades de información de manera estructurada. Su función principal incluye la creación, lectura, actualización y eliminación de registros, así como la realización de consultas complejas para extraer información relevante. Además, el SGBD se encarga del control de acceso y la seguridad, gestionando los permisos de los usuarios y protegiendo la base de datos contra accesos no autorizados, lo que asegura la integridad y confidencialidad de los datos almacenados (Peña O'Shea, 2017).

MySQL es un sistema gestor de bases de datos (SGBD) reconocido y ampliamente utilizado por su simplicidad y excelente rendimiento. A pesar de que no incluye algunas características avanzadas que ofrecen otros SGBD en el mercado, su facilidad de uso y el corto tiempo de implementación lo convierten en una opción atractiva para aplicaciones comerciales y de entretenimiento. Además, su distribución gratuita en Internet bajo la licencia GPL proporciona beneficios adicionales, como un alto grado de estabilidad y un desarrollo ágil, lo que permite a los desarrolladores aprovechar al máximo sus capacidades sin incurrir en costos elevados (Baca; Valdivia y Solís, 2022).

1.3.5. Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE) es un sistema de software diseñado para facilitar la creación de aplicaciones, combinando herramientas comunes para desarrolladores en una sola interfaz gráfica de usuario. Un IDE incluye un editor de texto que asiste en la escritura del código, ofreciendo funciones como el resaltado de sintaxis, el autocompletado específico del lenguaje y la comprobación de errores en tiempo real. Estas características ayudan a los desarrolladores a escribir código de manera más eficiente y precisa.
Además, un IDE automatiza tareas sencillas y repetitivas, como la creación de una compilación local del software, que incluye la compilación del código fuente en código binario, el empaquetado del código y la ejecución de pruebas automatizadas. También incorpora un depurador, una herramienta esencial para probar programas, que permite identificar y localizar errores en el código original de manera gráfica (Saabith; Vinothraj y Fareez, 2021). Gracias a estas funcionalidades, los IDEs no solo mejoran la productividad del desarrollador, sino que también optimizan el proceso de desarrollo de software en su totalidad.

Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero potente que funciona en escritorio y está disponible para Windows, macOS y Linux. Ofrece soporte integrado para JavaScript, TypeScript y Node.js, junto con un extenso ecosistema de extensiones que abarcan otros lenguajes y entornos de ejecución como C++, C#, Java, Python, PHP, Go y .NET. Esta herramienta de desarrollo proporciona una experiencia de usuario excepcional gracias a características avanzadas como la finalización de código inteligente, depuración integrada, control de versiones y diversas herramientas de productividad que optimizan el trabajo de los desarrolladores. Su interfaz personalizable y su soporte multiplataforma han contribuido a que Visual Studio Code se convierta en uno de los editores de código más populares y utilizados en la comunidad de programadores a nivel mundial (Code, 2024).
Se elige Visual Studio Code por su ligereza y potencia, así como por su amplia compatibilidad con diversos lenguajes de programación. Su interfaz personalizable, junto con la finalización de código inteligente y la depuración integrada, permiten una experiencia de desarrollo excelente. Además, el ecosistema de extensiones facilita adaptar la herramienta a las necesidades específicas del proyecto en desarrollo.

1.3.6. Servidor de aplicaciones web Apache

Un servidor web es una aplicación que responde a solicitudes de navegadores, entregando recursos a través del protocolo HTTP (Hypertext Transfer Protocol) o HTTPS (Hypertext Transfer Protocol Secure) para una transmisión segura (Palma Pérez, 2020). Como explica Cubas Fernández (2019), el servidor web es un programa que recibe y procesa solicitudes HTTP, ya sea localizando una página web o ejecutando un programa en el servidor, siempre generando una respuesta en HTML para el cliente. Este elemento es esencial en el desarrollo de aplicaciones del lado del servidor, pues en él se ejecutan y gestionan dichos procesos.

El servidor web Apache destaca por su estabilidad y fiabilidad, cualidades esenciales en la gestión de sitios web exitosos. No obstante, su rendimiento puede mejorarse mediante ajustes en la configuración, el equilibrio de carga y la agrupación en clúster. Al ser software de código abierto, Apache es gratuito y personalizable, adaptándose a necesidades específicas y ofreciendo la capacidad de gestionar múltiples solicitudes simultáneamente, ideal para sitios con alto tráfico.

Para optimizar su rendimiento, es recomendable ajustar el número de procesos de trabajo, lo cual permite gestionar el uso de memoria según la capacidad del servidor. La implementación de caché también reduce tiempos de respuesta al guardar en memoria contenido frecuentemente solicitado, mejorando la experiencia del usuario.

Además, el equilibrio de carga distribuye las solicitudes entre varios servidores, aliviando la carga en cada uno y maximizando su eficiencia. La agrupación en clúster, por otro lado, aumenta la resiliencia al conectar varios servidores; si uno falla, los demás asumen su carga sin interrupciones (Tiama y Euclides, 2023).

1.3.7. Marco de trabajo para el desarrollo de la solución informática

Un framework, o marco de trabajo, es un conjunto estructurado de reglas y convenciones que facilita el desarrollo de software de forma más rápida y eficiente. Su propósito es optimizar tiempo y esfuerzo al proporcionar herramientas y bibliotecas que sirven como base para los desarrolladores, permitiéndoles enfocarse en las funcionalidades específicas de sus aplicaciones. Además, los frameworks resuelven problemas comunes del desarrollo, eliminando la necesidad de abordar obstáculos técnicos recurrentes y agilizando el proceso de creación (Lucena, 2023).

Django:

Django es un framework web de alto nivel para Python que promueve un desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, permite simplificar las tareas complejas del desarrollo web, permitiendo a los programadores centrarse en su aplicación sin tener que reinventar soluciones comunes. Además, es gratuito y de código abierto (Vidal-Silva; Sánchez-Ortiz; Serrano y Rubio, 2021).

Una de sus principales ventajas es su potente ORM (Object-Relational Mapping), que permite interactuar con bases de datos mediante objetos de Python, evitando la necesidad de consultas SQL directas. Esto facilita la gestión de datos y asegura una mayor portabilidad entre sistemas de bases de datos. También cuenta con un sistema de administración para crear interfaces personalizadas y gestionar los datos de manera eficiente.

Django incluye herramientas de autenticación y protección contra ataques como la inyección SQL y el cross-site scripting (XSS), asegurando una gestión de permisos robusta y segura. Su sistema de enrutamiento y vistas facilita la definición de URLs y el manejo de solicitudes HTTP, permitiendo una navegación intuitiva y flexible para estructurar el código.

El sistema de plantillas de Django separa la lógica de la aplicación de su presentación visual, promoviendo una estructura limpia y fácil de mantener. Este enfoque modular facilita la colaboración entre desarrolladores y diseñadores y permite adaptar la aplicación fácilmente a futuros cambios.

Para proyectos globales, Django facilita la internacionalización, permitiendo su uso en múltiples idiomas y regiones. Su amplia comunidad activa aporta una abundancia de bibliotecas, paquetes y recursos, que no solo enriquecen el ecosistema de Django, sino que también ofrecen soluciones, mejores prácticas y mejoras continuas que agilizan el desarrollo.

Se eligió Django en su versión 4.2.6 como framework de desarrollo por su enfoque pragmático y su capacidad para acelerar la creación del sistema solicitado por el cliente. La facilidad de uso y la calidad del soporte comunitario lo convierten en una opción confiable para un proyecto que requiere eficiencia y adaptabilidad. Además, su naturaleza de código abierto permite personalizar y escalar según las necesidades específicas, lo que lo transforma en una herramienta ideal para mantener un entorno de trabajo flexible y dinámico.

React:

React es una biblioteca JavaScript declarativa, eficiente y flexible, diseñada específicamente para construir interfaces de usuario. Su principal fortaleza radica en la capacidad de componer UI complejas a partir de componentes pequeños y aislados, lo que promueve la reutilización del código y facilita el mantenimiento del mismo (Maratkar y Adkar, 2021). Cada componente en React puede gestionar su propio estado, lo que permite a los desarrolladores crear interfaces interactivas y dinámicas de manera efectiva.

Una de las características más destacadas de React es su uso del Virtual DOM (Document Object Model). En lugar de actualizar directamente el DOM real con cada cambio en el estado de la aplicación, React primero realiza modificaciones en una copia virtual del DOM. Luego, compara el Virtual DOM con el DOM actual y aplica solo las diferencias necesarias, lo que resulta en actualizaciones más rápidas y eficientes, mejorando así el rendimiento general de la aplicación.

Además, React utiliza una sintaxis llamada JSX, que permite a los desarrolladores escribir código HTML dentro de JavaScript. Esto hace que la creación de componentes y la representación de la interfaz sean más declarativas y legibles, facilitando el proceso de desarrollo y mejorando la comprensión del código.

El flujo de datos unidireccional en React también contribuye a su efectividad, ya que permite un seguimiento claro de cómo los datos afectan la interfaz de usuario. Este enfoque ayuda a los desarrolladores a entender mejor el flujo de información dentro de la aplicación y a depurar problemas más fácilmente (ibíd.).

React en su versión 18.3.1 se presenta como una elección destacada para el desarrollo de la interfaz de usuario del proyecto, ya que permite crear una experiencia interactiva y ágil. Su enfoque modular facilita tanto la escalabilidad como el mantenimiento, lo que resulta en un código más organizado. Además, la comunidad activa de React brinda acceso a una amplia variedad de recursos y herramientas, asegurando que

se puede avanzar de manera eficiente y abordar cualquier desafío que surja durante el desarrollo.

1.4. Metodología de desarrollo de software utilizada

La metodología para el desarrollo de software es un enfoque sistemático que permite gestionar y administrar un proyecto, aumentando las posibilidades de éxito. La metodología abarca los procesos que deben seguirse de manera ordenada para concebir, implementar y mantener un producto de software, desde la identificación de la necesidad hasta el cumplimiento de los objetivos establecidos (Morales-Carrillo; Cedeño-Valarezo; Bravo y Calderón, 2022).

En la actualidad, se pueden clasificar las metodologías de desarrollo de software en dos grupos principales. En primer lugar, las metodologías tradicionales, que enfatizan un proceso riguroso con un enfoque en la planificación predictiva, el seguimiento minucioso, la documentación exhaustiva y la negociación contractual. Por otro lado, surgieron las metodologías ágiles como respuesta a las limitaciones de las tradicionales. Estas últimas se centran en entregas frecuentes de versiones funcionales del software, priorizando la planificación adaptativa, la colaboración con el cliente y la capacidad de respuesta ante cambios inevitables en el desarrollo (ibíd.).

Dado el enfoque rígido y la carga documental de las metodologías tradicionales, se determinó que no se alineaban con las expectativas del proyecto ni con las necesidades de los usuarios. Por ello, se optó por la metodología ágil para el desarrollo e implementación del sistema, gracias a su flexibilidad y capacidad de adaptación a entornos y requisitos cambiantes.

SCRUM:

Scrum es un proceso ágil y ligero diseñado para gestionar y controlar el desarrollo de software de manera efectiva. Este enfoque se basa en ciclos cortos de construcción iterativa e incremental, donde cada iteración, que generalmente dura entre 2 y 4 semanas, culmina en un producto de software ejecutable que incorpora nuevas funcionalidades. Scrum se utiliza frecuentemente como un marco de referencia para integrar otras prácticas de ingeniería de software, como RUP o Extreme Programming.

La metodología Scrum se centra en priorizar tareas según su valor para el negocio, lo que maximiza la utilidad de lo que se desarrolla y asegura un retorno de inversión significativo. Está especialmente diseñado para adaptarse a cambios en los requisitos, lo que es fundamental en un entorno competitivo. A lo largo del proyecto, los requisitos y las prioridades se revisan y ajustan en intervalos regulares, permitiendo que el producto se ajuste en tiempo real a las necesidades del cliente y garantizando que el software entregue soluciones efectivas, aumentando así la satisfacción del usuario.

En Scrum, el equipo se concentra en una única meta: construir software de calidad. La gestión del proyecto se enfoca en definir las características necesarias del producto, estableciendo qué se debe construir, en qué orden y eliminando cualquier obstáculo que pueda afectar la productividad del equipo. Esta estructura busca maximizar la efectividad y eficiencia de los equipos de desarrollo.

Con un conjunto reducido de reglas, Scrum se basa en principios de inspección continua, adaptación, auto-

gestión e innovación. Este enfoque no solo motiva al cliente, que observa el progreso del producto iteración tras iteración, sino que también alinea el desarrollo con los objetivos comerciales de la empresa. Además, proporciona un ambiente propicio para el crecimiento profesional de los desarrolladores, lo que a su vez incrementa la motivación y el compromiso del equipo (Riano, 2021).

Extreme Programming (XP):

Extreme Programming (XP) es la metodología más prominente dentro de los procesos ágiles de desarrollo de software, formulada por Kent Beck. Se distingue de las metodologías tradicionales principalmente por su enfoque en la adaptabilidad en lugar de la previsibilidad.

Los defensores de XP sostienen que los cambios en los requisitos son un aspecto natural, inevitable e incluso deseable en el desarrollo de proyectos. Esta metodología propone que la capacidad de adaptarse a estos cambios en cualquier etapa del proyecto es una estrategia más efectiva y realista que intentar definir todos los requisitos desde el principio y luego lidiar con los ajustes necesarios. Así, XP fomenta un enfoque dinámico que permite alinearse continuamente con las necesidades del cliente, mejorando la calidad del producto final (Peralta Marini e Hilasaca Apaza, 2020).

Extreme Programming (XP) se basa en varias características fundamentales que fomentan un desarrollo ágil y eficiente. El método se caracteriza por un enfoque iterativo e incremental, donde se implementan pequeñas mejoras de manera continua. Se realizan pruebas unitarias de forma regular y automatizada, incluyendo pruebas de regresión, y se recomienda escribir el código de prueba antes de la codificación para asegurar la calidad desde el inicio. La programación por parejas es otra práctica clave, donde dos desarrolladores trabajan juntos en el mismo código, lo que mejora la calidad al permitir la revisión constante. Además, se promueve una interacción frecuente entre el equipo de programación y el cliente, con un representante del cliente colaborando directamente con los desarrolladores.

XP enfatiza la corrección de todos los errores antes de añadir nueva funcionalidad y realiza entregas frecuentes para mantener un flujo de trabajo continuo. La refactorización del código, que busca aumentar su legibilidad y mantenibilidad sin modificar su comportamiento, es igualmente importante. También se fomenta la propiedad compartida del código, permitiendo que cualquier miembro del equipo pueda corregir y extender cualquier parte del proyecto, lo que ayuda a detectar errores mediante pruebas de regresión. Por último, se prioriza la simplicidad en el código, ya que se considera que una solución sencilla es más fácil de mantener y modificar que una complicada.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores (ibíd.).

Extreme Programming (XP) ofrece ventajas significativas para entornos de desarrollo volátiles, como la capacidad de adaptarse rápidamente a cambios, lo que ayuda a reducir costos. Su planificación transparente permite a los clientes conocer las fechas de entrega de funcionalidades importantes, mientras que la definición de objetivos en cada iteración y la retroalimentación continua de los usuarios garantizan que se aborden

las prioridades adecuadas. Además, la presión se distribuye a lo largo del proyecto, evitando acumulaciones en una entrega final.
No obstante, XP presenta desventajas, siendo la principal, la dificultad para delimitar el alcance del proyecto con el cliente. Esta falta de claridad puede generar confusiones o expectativas no cumplidas, afectando la satisfacción del cliente. Por ello, es esencial mantener una comunicación abierta para gestionar adecuadamente las expectativas y asegurar que el proyecto se mantenga alineado con sus objetivos.

Selección de la metodología de desarrollo de software

Se opta por Extreme Programming (XP) como la metodología de desarrollo de software dentro de los modelos ágiles. Esta elección se fundamenta en la capacidad de XP para entregar el sistema en el menor tiempo posible, sin comprometer la calidad. Además, la experiencia previa en el uso de metodologías ágiles en la construcción de la plataforma a la que se integra la solución facilita su adopción, garantizando una transición fluida y eficiente.

Conclusiones Parciales

En este primer capítulo se ha establecido un marco teórico y práctico que sustenta la propuesta de solución al problema planteado. A continuación, se destacan las conclusiones más relevantes:

- La adecuada definición de conceptos clave facilitó la identificación de las necesidades del sistema y estableció una base sólida para su desarrollo.
- A partir de un análisis detallado, se seleccionaron como herramientas y tecnologías fundamentales para el desarrollo del sistema los lenguajes de programación Python v3.11.5 y JavaScript vES13, así como las tecnologías Django v4.2.6, React v18.3.1 y el gestor de base de datos MySQL v5.2.1.
- Se eligió XP (Extreme Programming) como metodología de desarrollo para permitir una interacción continua con el cliente, asegurando que la calidad del producto final estuviera alineada con sus expectativas y requerimientos.

CAPÍTULO 2

Diseño de la solución propuesta al problema científico

En este capítulo, se presenta el diseño de la solución propuesta para abordar el problema científico identificado. Se comenzará por establecer los requisitos esenciales que la aplicación debe cumplir, tanto desde una perspectiva funcional como no funcional. Además, se detallarán los principales componentes y herramientas de ingeniería de software que se emplearán, basados en la metodología seleccionada. Este enfoque sistemático permitirá garantizar que la solución no solo sea efectiva, sino también sostenible y adaptable a futuros desarrollos.

2.1. Descripción de la propuesta de solución

Al ingresar a la dirección web de la aplicación, se presenta una pantalla de inicio de sesión con un formulario que requiere el nombre de usuario y la contraseña, ambos deben ser correctos para acceder al sistema. Una vez autenticado, el usuario es dirigido a la pantalla principal, que muestra gráficos estadísticos y una barra de navegación a la izquierda. Esta barra ofrece diversas opciones según el rol del usuario, permitiendo añadir, actualizar, modificar y consultar información en tablas, facilitando así la gestión de datos necesarios en la aplicación. Ver Figura 2.1.

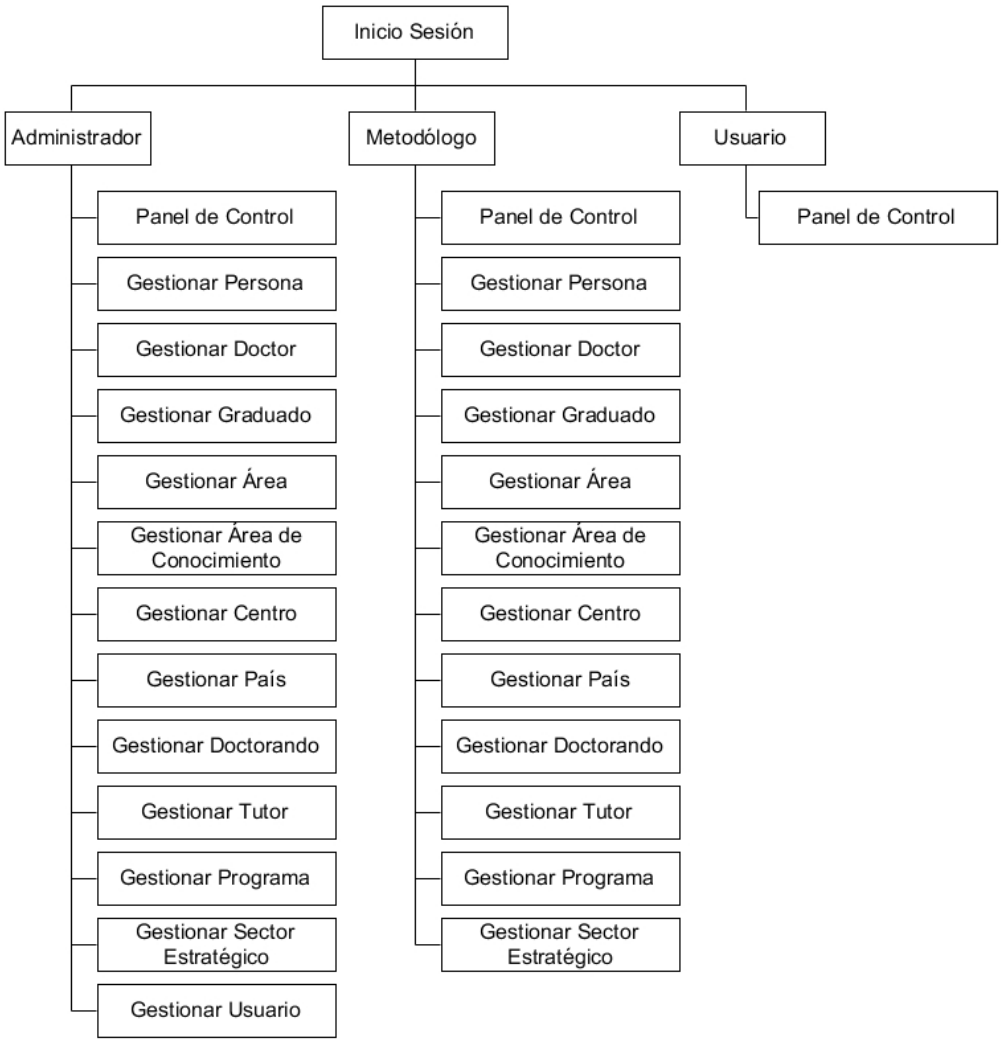


Figura 2.1. Mapa de Navegación

2.2. Historias de Usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden eliminarse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Menzinsky; López; Palacio; Sobrino; Álvarez y Rivas, 2018).

A continuación se muestran las historias de usuario definidas:

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Gestionar Persona
Usuario: Administrador, Gerente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir una persona a partir de los campos nombre, apellidos, carnet de identidad, sexo, plantilla, país, centro y sector estratégico al que pertenece. Además debe permitir modificar, eliminar, buscar y ver el listado de todas las personas.	
Observaciones: Depende de que haya países, áreas, centros y sectores estratégicos registrados en el sistema.	

Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Gestionar Área
Usuario: Administrador, Gerente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un área a partir del campo nombre. Además debe permitir modificar, eliminar, buscar y ver el listado de todas las áreas.	
Observaciones: Ninguna	

Tabla 2.3. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Gestionar Área de Conocimiento
Usuario: Administrador, Gerente	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un área de conocimiento a partir del campo nombre. Además debe permitir modificar, eliminar, buscar y ver el listado de todas las áreas de conocimiento.	
Observaciones: Ninguna	

El resto de las historias de usuario se muestran en el apéndice *B* de los anexos.

2.3. Requisitos no funcionales

Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema software (tal como estándares de calidad, o requisitos del diseño) (García-Peñalvo; García-Holgado y Vázquez-Ingelmo, 2024).
A continuación se muestran los requisitos no funcionales definidos para el sistema:
Usabilidad:
RNF1 Interfaz: El sistema debe tener una interfaz accesible y usable por personas con discapacidades visuales.
RNF2 Simplicidad: El sistema debe ser simple, sin animaciones innecesarias, de forma tal que se evite el desorden y el exceso de información.
Software:
RNF3 Sistema operativo: El sistema requiere sistema operativo para funcionar Linux ,Windows, MAC, IOS u Android.
RNF4 Navegador: El sistema requiere navegador Mozilla Firefox superior a la versión 130.0.
Seguridad:
RNF5 Accesibilidad: El sistema deberá garantizar que los usuarios solo tengan acceso a realizar cambios en la información después de autenticarse en el sistema.

2.4. Tarjetas CRC

Las tarjetas de Clases, Responsabilidades y Colaboraciones (CRC) son una herramienta de diseño utilizada en el desarrollo de software. Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella (Lara y Figueroa, 2020).
A continuación se muestran las tarjetas CRC definidas:

Tabla 2.4. Tarjeta CRC # 1

Tarjeta CRC	
Clase: persona	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.4. Continuación de la página anterior

• Registrar las personas • Modificar las personas • Eliminar las personas • Buscar las personas • Mostrar el listado de las personas	area pais centro sectorest PersonaViewSet Persona_Serializer

Tabla 2.5. Tarjeta CRC # 2

Tarjeta CRC	
Clase: area	
Responsabilidad	**Colaboración**
• Registrar las áreas • Modificar las áreas • Eliminar las áreas • Buscar las áreas • Mostrar el listado de las áreas	AreaViewSet Area_Serializer

Tabla 2.6. Tarjeta CRC # 3

Tarjeta CRC	
Clase: areadeconocimiento	
Responsabilidad	**Colaboración**
• Registrar las áreas de conocimiento • Modificar las áreas de conocimiento • Eliminar las áreas de conocimiento • Buscar las áreas de conocimiento • Mostrar el listado de las áreas de conocimiento	AreadeconocimientoViewSet Areadeconocimiento_Serializer

El resto de las tarjetas crc se muestran en el apéndice *C* de los anexos.

2.5. Plan de lanzamiento

Una reunión de planificación de entregas o lanzamientos se utiliza para establecer un Plan de Lanzamiento que proporciona una visión general del proyecto. Este plan es esencial para desarrollar los planes de iteración de cada ciclo de trabajo individual (Orozco; Pardo y Vásquez, 2020).

Durante la reunión de planificación, el equipo de desarrollo se enfoca en estimar cada historia de usuario en términos de semanas de programación ideales. Una semana ideal representa el tiempo que el equipo anticipa que se requeriría para implementar esa historia específica. A su vez, el cliente participa activamente en el proceso al identificar cuáles historias tienen mayor importancia o prioridad para ser completadas, asegurando que el desarrollo se alinee con sus expectativas y necesidades (Orozco; Pardo y Vásquez, 2020).
A continuación se muestra el plan de lanzamiento elaborado a partir de las prioridades establecidas en las historias de usuario:

Tabla 2.7. Plan de lanzamiento

Lanzamiento	**Descripción de la iteración**	**Orden de las HU a implementar**	**Duración Total**
Iteración 1	En esta iteración se realizarán las HU que tienen prioridad alta.	HU1, HU2, HU4, HU5, HU12, HU14, HU15	2 semanas
Iteración 2	En esta iteración se realizarán las HU que tienen prioridad media.	HU3, HU6, HU7, HU9, HU13	2 semanas
Iteración 3	En esta iteración se realizarán las HU que tienen prioridad baja.	HU8, HU10, HU11	1 semana

2.6. Patrón de arquitectura

La arquitectura de software de un programa es fundamental, ya que define la estructura y organización del sistema, incluyendo sus componentes, relaciones y los principios que guían su diseño y evolución. También abarca los elementos y propiedades visibles externamente, que son las suposiciones que otros componentes pueden hacer sobre un elemento (Engelenburg; Janssen y Klievink, 2019).
Para este proyecto, se ha optado por el framework Django, lo que lleva a elegir la arquitectura Modelo, Plantilla y Vista (MTV) como la mejor práctica, una variante del Modelo, Vista y Controlador (MVC). En esta arquitectura, el Modelo representa la capa de acceso a los datos, donde las clases corresponden a las tablas de la base de datos, permitiendo controlar el comportamiento de la información. La Vista se encarga de la lógica del negocio, conectando el acceso al modelo con la presentación, gestionando las solicitudes y devolviendo los datos apropiados, todo ello utilizando código Python en lugar de SQL. Por último, la Plantilla se ocupa de la capa de presentación, definiendo cómo se mostrará la información al usuario final mediante estilos HTML y tecnologías como CSS y JavaScript (Nilve Balseca, 2021).
El mapeo de rutas se gestiona a través de URLs que dirigen las vistas correspondientes (URLConf). Su objetivo es identificar la URL solicitada por el cliente y responder con la vista adecuada, enviando cualquier variable necesaria para completar la solicitud (ibíd.). En la Figura 2.1 se presenta la arquitectura MTV

implementada en Django.

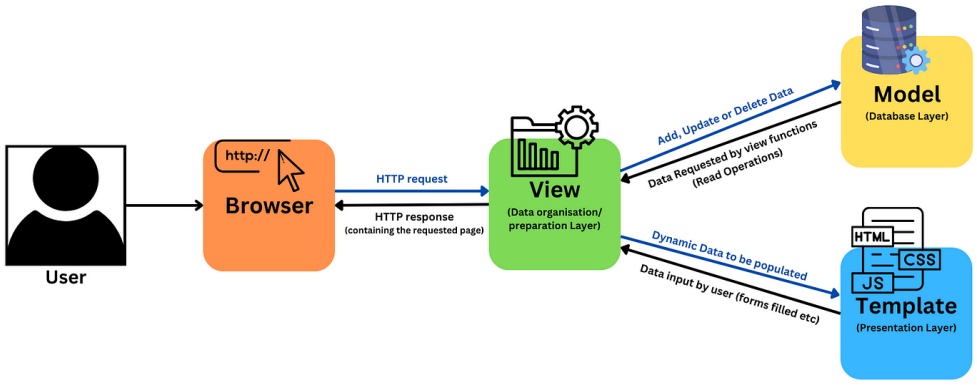


Figura 2.2. Patrón de Arquitectura Model-Template-View (MTV)

2.7. Patrones de Diseño

Los patrones de diseño son esenciales en la ingeniería de software, ya que ofrecen soluciones probadas y documentadas para problemas recurrentes en el desarrollo. Según Ortega, 2021, un patrón de diseño es una solución efectiva a un problema de diseño que ha demostrado su eficacia en situaciones similares. Además, debe ser reutilizable, lo que implica que puede aplicarse para resolver problemas análogos en diferentes contextos. A continuación, se presentan los patrones empleados en el diseño y desarrollo de la propuesta de solución:

Patrones GRASP:

- Experto: Asigna la responsabilidad a la clase que tiene la información necesaria para cumplirla, promoviendo la cohesión. Se utilizó en los archivos **models.py** en los métodos que operan sobre los atributos de cada clase.
- Creador: Sugiere que una clase debe ser responsable de crear instancias de otra clase si tiene una relación de composición o agregación con ella.
- Controlador: Define una clase que actúa como intermediario entre la interfaz de usuario y el modelo, gestionando la lógica de la aplicación. Se utilizó en los archivos **view.py**.
- Bajo Acoplamiento: Promueve la asignación de responsabilidades que minimizan las dependencias entre clases, facilitando el mantenimiento y la reutilización.
- Alta Cohesión: Sugiere que las clases deben tener responsabilidades estrechamente relacionadas, mejorando la claridad y la facilidad de uso. Se utilizó en la clase Centro en el archivo **models.py**.
- Fabricación Pura: Propone crear una clase que no representa un concepto del dominio, pero que ayuda a cumplir con las responsabilidades de diseño, evitando que las clases del dominio se vuelvan demasiado complejas. Se utilizó en la app **sesion**.

- Indirección: Introduce un intermediario para reducir el acoplamiento entre dos clases. Utiliza intermediarios como middleware para gestionar la lógica de autenticación o autorización en lugar de tener esa lógica dispersa en las vistas.

Patrones GOF:

- Facade: Proporciona una interfaz simplificada a un conjunto de interfaces en un subsistema. Se utilizó en la app **estadistica** en el archivo **views.py** donde se crea una vista que actúa como una fachada para acceder a varios servicios relacionados con estadísticas sobre los diferentes actores como son: doctores, doctorandos, tutores, entre otros. Este enfoque ayuda a mantener el código limpio y organizado, a la vez que proporciona una interfaz coherente y fácil de usar para los consumidores de la API.
- Chain of Responsibility: Permite pasar solicitudes a lo largo de una cadena de manejadores.

2.8. Modelo de Datos

Un modelo de datos es una representación abstracta que describe la estructura, relaciones y restricciones de los datos en un sistema de gestión de bases de datos. Su propósito es ofrecer un marco conceptual que facilite la organización y gestión eficiente de la información, optimizando así el almacenamiento y acceso a los datos. Este modelo incluye elementos como entidades, atributos y relaciones, y puede ser visualizado mediante diagramas, como los diagramas entidad-relación. En el contexto de este proyecto, los modelos de datos son esenciales para el diseño de la base de datos, ya que aseguran la coherencia e integridad de la información, al mismo tiempo que sirven de referencia tanto para desarrolladores como para usuarios en la comprensión del sistema de datos (Rodriguez, 2024).
A continuación se muestra el modelo de datos del sistema:

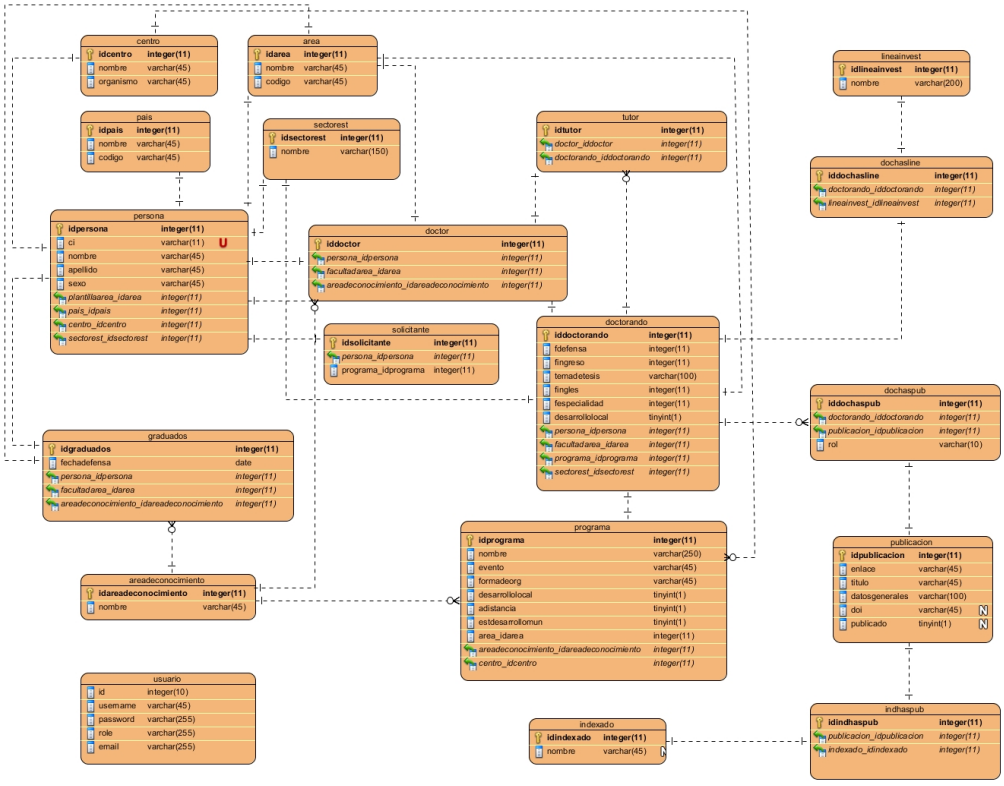


Figura 2.3. Modelo de datos

Conclusiones del capítulo

En este segundo capítulo, se ha trazado el camino para desarrollar la solución a nuestro problema científico. A continuación, se destacan las conclusiones más relevantes:

- Se definieron 15 historias de usuario que guiaron el desarrollo de la aplicación, asegurando que cumpliera con las expectativas de los usuarios y los estándares de calidad necesarios.
- Se estableció la arquitectura Modelo, Plantilla y Vista (MTV), promoviendo la cohesión y el bajo acoplamiento entre los componentes del sistema, lo que facilitó su mantenimiento.
- Se adoptaron patrones de diseño GRASP y GoF, que optimizaron la organización del código y fomentaron la reutilización, garantizando un sistema más sólido y flexible ante futuros cambios.
- Se delineó un modelo de datos que reflejó las entidades y relaciones necesarias para el funcionamiento de la aplicación, asegurando la integridad y consistencia de los datos a lo largo del desarrollo.

CAPÍTULO 3

Validación de la solución propuesta

En este capítulo, se analizan los aspectos relacionados con la implementación de la aplicación y las pruebas realizadas para evaluar su funcionamiento y calidad.

3.1. Tareas de Ingeniería

En la fase de Planeación, se establecieron los elementos funcionales necesarios para el desarrollo del sistema, lo que permite avanzar a la fase de implementación. En esta etapa, se utilizarán las Tareas de Ingeniería (TI) como herramienta principal para guiar el proceso.
Las tareas ingenieriles son actividades específicas y sistemáticas que los ingenieros realizan durante el desarrollo y mantenimiento de sistemas y productos. Estas tareas abarcan desde el análisis de requisitos y el diseño hasta la implementación, pruebas y documentación. Su objetivo es garantizar que los proyectos se ejecuten de manera eficiente, cumpliendo con los estándares de calidad y satisfaciendo las necesidades del cliente. Al seguir un enfoque estructurado, las tareas ingenieriles contribuyen a la creación de soluciones efectivas y sostenibles (Tamayo Espinosa y Silega Martínez, 2021)
A continuación se muestran las Tareas Ingenieriles definidas:

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** N/A
Nombre de la tarea: Diseño de la base de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Diseñar la base de datos con los campos y entidades necesarias.	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** N/A
Nombre de la tarea: Implementación de la base de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Crear la base de datos con los campos y entidades necesarias.	

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 1
Nombre de la tarea: Gestionar Persona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de las personas.	

El resto de las tareas ingenieriles se muestran en el apéndice *D* de los anexos.

3.2. Pruebas

Las pruebas de software son un proceso sistemático que abarca múltiples actividades, desde la planificación y análisis hasta la ejecución y el cierre. Su objetivo principal es garantizar que el software funcione correctamente, cumpliendo con los requisitos establecidos y evitando comportamientos indeseables. Este enfoque estructurado permite detectar y prevenir defectos, asegurando que el producto final sea predecible y confiable para los usuarios (Marin Diaz; Trujillo Casañola y Buedo Hidalgo, 2020).
Dentro de la metodología XP, las pruebas del sistema se dividen en dos categorías principales: las pruebas unitarias, que son diseñadas por los programadores para validar el funcionamiento del código, y las pruebas de aceptación, elaboradas por el cliente final para confirmar que se ha cumplido con la funcionalidad requerida al finalizar cada iteración.

3.2.1. Pruebas unitarias

A menudo, se ignora la importancia de las pruebas unitarias en el desarrollo de microservicios. Estas pruebas son cruciales, ya que permiten verificar que las diferentes funciones y clases del código se comporten de acuerdo con las expectativas. Aunque su implementación puede parecer una tarea técnica y especializada, un conjunto bien definido de pruebas unitarias actúa como un mecanismo de protección esencial. Este mecanismo ayuda a identificar y mitigar problemas inesperados que pueden surgir cuando se realizan mo-

dificaciones en el código. Además, estas pruebas proporcionan a los desarrolladores información específica sobre qué partes del código han sido afectadas, lo que facilita la resolución de cualquier problema que pueda comprometer la funcionalidad y permite la creación de aplicaciones más robustas y confiables (Mamani, 2023).

Las pruebas unitarias se llevaron a cabo utilizando APITestCase, una clase proporcionada por Django REST Framework que extiende la clase TestCase de Django. En total, se realizaron 11 pruebas unitarias enfocadas en diversas clases, específicamente evaluando las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de los recursos. Este enfoque permite verificar que cada funcionalidad de la API se comporta como se espera, garantizando la integridad y la robustez del sistema. Durante estas pruebas, se comprobaron aspectos como la creación de recursos, la recuperación de datos, la actualización de registros existentes y la eliminación de recursos, asegurando así un funcionamiento adecuado de la aplicación en diferentes escenarios.

A continuación, se muestran parte del código y los resultados de la aplicación de estas pruebas a la clase **area**:

tests.py M

Apps > area > tests.py > ...

```python
from django.urls import reverse
from rest_framework import status
from rest_framework.test import APITestCase
from .models import *

class AreaTests(APITestCase):

    Tabnine | Edit | Test | Explain | Document | Ask
    def setUp(self):
        self.area = Area.objects.create(nombre="Test Area")

    Tabnine | Edit | Test | Explain | Document | Ask
    def test_create_area(self):
        url = reverse('area-list')
        data = {'nombre': 'Nueva Area', 'codigo': 'New A'}
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(Area.objects.count(), 2)
        self.assertEqual(Area.objects.get(idarea=2).nombre, 'Nueva Area')

    Tabnine | Edit | Test | Explain | Document | Ask
    def test_get_area(self):
        url = reverse('area-detail', args=[self.area.idarea])
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data['nombre'], self.area.nombre)

    Tabnine | Edit | Test | Explain | Document | Ask
    def test_update_area(self):
        url = reverse('area-detail', args=[self.area.idarea])
        data = {'nombre': 'Area Actualizada', 'codigo': 'New B'}
        response = self.client.put(url, data, format='json')
        self.area.refresh_from_db()
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(self.area.nombre, 'Area Actualizada')
        self.assertEqual(self.area.codigo, 'New B')

    Tabnine | Edit | Test | Explain | Document | Ask
    def test_delete_area(self):
        url = reverse('area-detail', args=[self.area.idarea])
        response = self.client.delete(url)
        self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
        self.assertEqual(Area.objects.count(), 0)
```

Figura 3.1. AreaTests

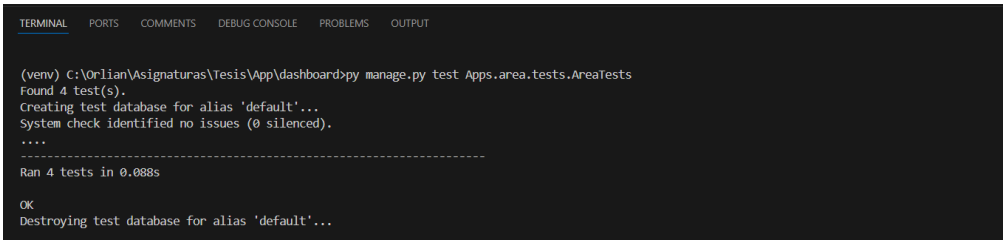


Figura 3.2. Resultado AreaTests

El resto de los resultados de las pruebas unitarias se muestran en el apéndice *E* de los anexos.

3.2.2. Pruebas de Aceptación

Las pruebas de aceptación, también conocidas como pruebas del cliente, se enfocan en las características y funcionalidades del sistema que son directamente visibles y evaluables por el cliente. Este tipo de pruebas se fundamenta en las historias de usuario, donde los propios clientes son responsables de redactar los casos de prueba correspondientes a cada historia que necesita ser validada. Su relevancia radica en garantizar que el producto final no solo cumpla con los requisitos técnicos, sino que también satisfaga las necesidades y expectativas del cliente, asegurando así una mayor satisfacción y confianza en el software entregado (Figueredo, 2021). A continuación, se describen los casos de pruebas de aceptación definidos:

Tabla 3.4. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Listar Persona	
Descripción: Prueba la funcionalidad de listar las personas.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Persona.	
Resultados esperados: El sistema carga una página con el listado de personas.	

Tabla 3.5. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU1_P2	**Historia de usuario:** 1
Nombre: Registrar Persona	
Descripción: Prueba la funcionalidad de registrar una persona.	

Continúa en la próxima página

Tabla 3.5. Continuación de la página anterior

Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de personas.
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema registra una nueva persona si los datos fueron correctos, en caso contrario muestra un mensaje de error.

Tabla 3.6. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU1_P3	**Historia de usuario:** 1
Nombre: Modificar Persona	
Descripción: Prueba la funcionalidad de modificar una persona.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de personas.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica una persona si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla 3.7. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU1_P4	**Historia de usuario:** 1
Nombre: Eliminar Persona	
Descripción: Prueba la funcionalidad de eliminar una persona.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de personas.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina una persona del sistema.	

El resto de las pruebas de aceptación se muestran en el apéndice *F* de los anexos.

3.3. Resultado de las pruebas

Para asegurarnos de que todos los requisitos funcionales se implementaron de manera adecuada, se llevaron a cabo las pruebas definidas por la metodología XP. En total, se ejecutaron 51 pruebas de aceptación y 13 pruebas unitarias. A continuación, se presenta un gráfico de elaboración propia que ilustra la cantidad de pruebas realizadas en cada iteración.

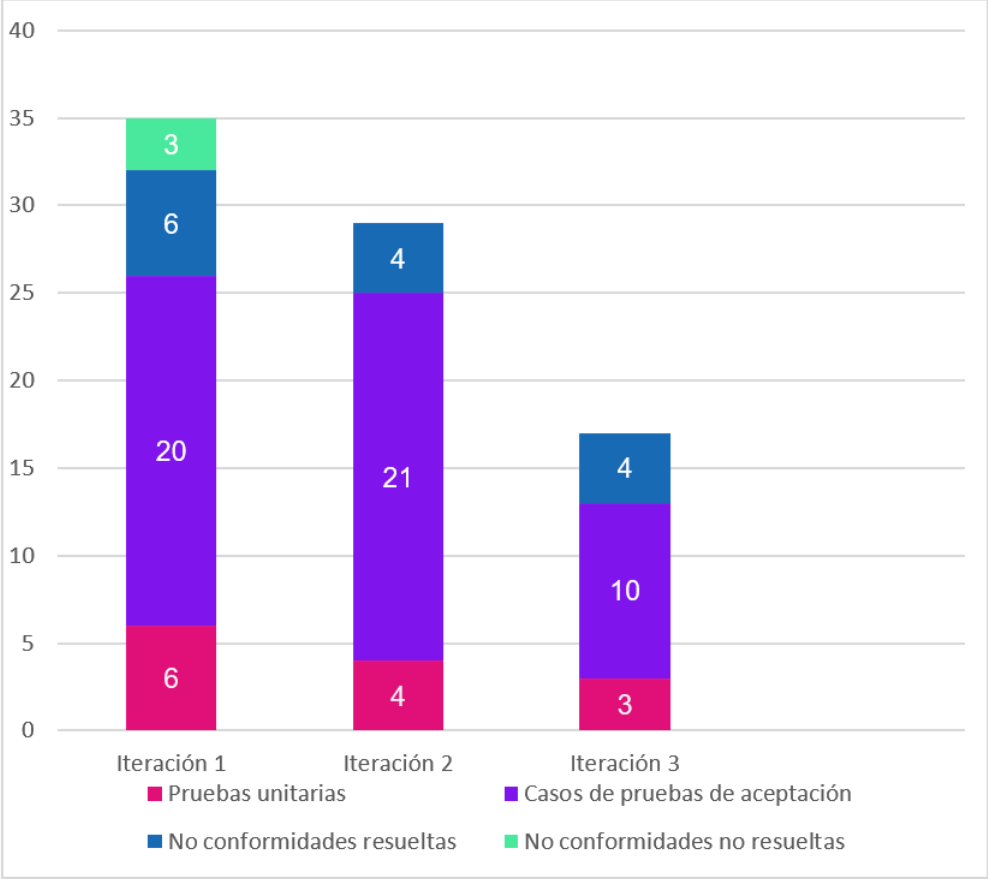


Figura 3.3. Resultado Pruebas

Conclusiones del capítulo

En este tercer capítulo, se detallan las fases de implementación y pruebas de software de la solución propuesta. A continuación, se destacan las conclusiones más relevantes:

- Las tareas de ingeniería facilitaron al equipo del proyecto la identificación, control y seguimiento de los requisitos y modificaciones en cualquier fase del desarrollo, asegurando una gestión eficiente del mismo.

- Las pruebas unitarias realizadas en el software permitieron verificar la calidad y funcionalidad del sistema propuesto, garantizando que cada componente operara como se esperaba.
- La realización de las pruebas de aceptación confirmó tanto el correcto funcionamiento del sistema como su alineación con los requisitos establecidos, asegurando que el producto final satisficiera las necesidades del usuario.

Conclusiones

La investigación realizada abordó de manera integral el problema planteado, y se obtuvo como resultado una solución efectiva alineada con las necesidades del cliente. A continuación se presentan las conclusiones más significativas:

- La clara definición de conceptos clave facilitó la identificación de las necesidades específicas del sistema, y permitió establecer una base sólida para entender los procesos de gestión de información de los doctorandos en la UCI.
- La selección de lenguajes de programación como Python y JavaScript, junto con tecnologías como Django y React, fue fundamental para el desarrollo de una aplicación robusta; el uso de MySQL como gestor de base de datos garantizó un manejo eficiente de la información.
- Se logró un módulo de gestión de la información que organiza y permite el acceso a datos relevantes del proceso de formación doctoral, facilitando así una gestión eficaz.
- Se implementó un panel de control que visualiza los indicadores clave, permitiendo una monitorización efectiva de la formación doctoral y mejorando la toma de decisiones.
- Las pruebas de software realizadas validaron que la calidad y las funcionalidades del sistema se ajustaron a las expectativas del cliente.

Recomendaciones

Debido al continuo proceso de mejora y evolución que caracteriza a cualquier sistema de software, se sugieren las siguientes acciones:

- Incorporar nuevos indicadores de desempeño de la formación doctoral.
- Incorporar nuevos gráficos para la representación de datos y así mejorar la comunicanicación
- Añadir nuevas funciones para generar reportes.

Referencias bibliográficas

AMAZON. *¿Qué es Java?* [online] [visitado 2024-06-15]. Disponible desde: https://aws.amazon.com/es/what-is/java/ (vid. pág. 11).

ANTÓN RODRÍGUEZ, Susana, 2023. *SIGENU: La revolución digital en la gestión universitaria*. Juventud Técnica. Url: https://www.juventudtecnica.cu/articulos/sigenu/ (vid. pág. 8).

BACA, Herwin Alayn Huillcen; VALDIVIA, Flor de Luz Palomino y SOLÍS, Iván Soria, 2022. *Introducción a las Bases de Datos con MySQL*. Herwin Alayn Huillcen Baca (vid. pág. 16).

BECKER, Louis T y GOULD, Elyssa M, 2019. Microsoft power BI: extending excel to manipulate, analyze, and visualize diverse data. *Serials Review*. Vol. 45, n.º 3, págs. 184-188 (vid. págs. 6, 7).

CODE, Visual Studio. *Documentation for Visual Studio Code* [online] [visitado 2024-06-17]. Disponible desde: https://code.visualstudio.com/docs (vid. pág. 16).

CUBAS FERNÁNDEZ, Luis Francisco, 2019. ANÁLISIS COMPARATIVO DEL RENDIMIENTO Y EL ESFUERZO MEDIANTE PRUEBAS DE CARGA EN SERVIDORES WEB (vid. pág. 17).

EDUCACIÓN DE POSGRADO, Dirección de, 2023. *Estrategia de Formación Doctoral de la Universidad de las Ciencias Informáticas para el período 2023-2030* (vid. págs. 1, 5).

ENGELENBURG, Sélinde van; JANSSEN, Marijn y KLIEVINK, Bram, 2019. Design of a software architecture supporting business-to-government information sharing to improve public safety and security: Combining business rules, Events and blockchain technology. *Journal of Intelligent information systems*. Vol. 52, págs. 595-618 (vid. pág. 27).

ESTADO, Consejo de, 2019. *Resolución No. 139/2019 (GOC-2019-775-O65)*. Gaceta Oficial de la República de Cuba. Url: http://www.gacetaoficial.gob.cu/ (vid. pág. 4).

FERNÁNDEZ Fastuca, Lorena, 2018. *Pedagogía de la formación doctoral*. Teseo (vid. pág. 5).

FEW, Stephen, 2024. Pervasive Hurdles to Effective Dashboard Design (vid. pág. 5).

FIGUEREDO, Lisandra, 2021. Proceso de pruebas de software para un modelo de calidad en Cuba. *I+ D Tecnológico*. Vol. 17, n.º 1, págs. 23-35 (vid. pág. 35).

FONTECHA ZABALETA, Joan Alexander, 2017. Python en la seguridad informática (vid. pág. 13).

GARCÍA-PEÑALVO, Francisco José; GARCÍA-HOLGADO, Alicia y VÁZQUEZ-INGELMO, Andrea, 2024. Ingeniería de requisitos (vid. pág. 25).

GEEKSFORGEEKS, 2023. *Backend Development Complete Guide* [online] [visitado 2024-06-14]. Disponible desde: `https://www.geeksforgeeks.org/backend-development/`. Section: Node.js (vid. pág. 9).

LARA, Cecilia y FIGUEROA, Liliana Maria, 2020. Metodología ágil para el desarrollo de aplicaciones móviles educativas. En: *Metodología ágil para el desarrollo de aplicaciones móviles educativas. XV Congreso Nacional de Tecnología en Educación y Educación en Tecnología (TE&ET 2020)(Neuquén, 6 y 7 de julio de 2020)* (vid. pág. 25).

LUCENA, Paola, 2023. *¿Qué es el framework?* [online] [visitado 2024-06-17]. Disponible desde: `https://www.cesuma.mx/blog/que-es-el-framework.html` (vid. pág. 17).

LUNA, Ainoa Celaya, 2024. *Creación de páginas web: HTML 5*. ICB, SL (Interconsulting Bureau SL) (vid. pág. 14).

LUNA, Fernando, 2019. *JavaScript-Aprende a programar en el lenguaje de la web*. RedUsers (vid. pág. 14).

MAMANI, Cesar Adolfo Laura, 2023. Pruebas de Software para Microservicios. *Innovación y Software*. Vol. 4, n.º 1, págs. 151-160 (vid. pág. 33).

MARATKAR, Pratik Sharad y ADKAR, Pratibha, 2021. React JS–An Emerging Frontend Javascript Library. *Iconic Research And Engineering Journals*. Vol. 4, n.º 12, págs. 99-102 (vid. pág. 18).

MARIN DIAZ, Aymara; TRUJILLO CASAÑOLA, Yaimí y BUEDO HIDALGO, Denys, 2020. Estrategia de pruebas para organizaciones desarrolladoras de software. *Revista Cubana de Ciencias Informáticas*. Vol. 14, n.º 3, págs. 83-104 (vid. pág. 32).

MENZINSKY, Alexander; LÓPEZ, Gertrudis; PALACIO, Juan; SOBRINO, Miguel Ángel; ÁLVAREZ, Rubén y RIVAS, Verónica, 2018. Historias de usuario. *Ingeniería de requisitos ágil* (vid. pág. 23).

MILLIGAN, Joshua N, 2019. *Learning Tableau 2019: Tools for Business Intelligence, data prep, and visual analytics*. Packt Publishing Ltd (vid. pág. 7).

MORALES-CARRILLO, Jessica; CEDEÑO-VALAREZO, Luis; BRAVO, Jesús Stefano Cajape y CALDERÓN, Jonathan Geovanny Ormaza, 2022. Metodologías de desarrollo de software y su ámbito de aplicación: Una revisión sistemática. *Revista Ibérica de Sistemas e Tecnologias de Informação*. N.º E47, págs. 29-45 (vid. pág. 19).

NILVE BALSECA, Ronny Andrés, 2021. Desarrollo de un sistema para la gestión del campeonato de fútbol de la Liga Cantonal de Penipe aplicando el framework Django y el patrón de diseño MTV. (Vid. pág. 27).

OROZCO, Carlos; PARDO, César y VÁSQUEZ, Sebastián, 2020. SCMOnto: Una ontología para soportar la gestión de la configuración de software. *Revista Ibérica de Sistemas e Tecnologias de Informacion*. N.º E38, págs. 75-90 (vid. págs. 26, 27).

ORTEGA, Gilberto Andrés Vargas, 2021. Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP. *Ciencia e Ingeniería: Revista de investigación interdisciplinar en biodiversidad y desarrollo sostenible, ciencia, tecnología e innovación y procesos productivos industriales*. Vol. 8, n.º 2, págs. 4 (vid. pág. 28).

PALMA PÉREZ, Nurisel, 2020. Solución informática para la selección del servidor web durante la migración a código abierto. *Revista Cubana de Ciencias Informáticas*. Vol. 14, n.º 2, págs. 49-69 (vid. pág. 17).

PARADIGM, Visual. *Visual Paradigm - UML, Agile, PMBOK, TOGAF, BPMN y más.* [online] [visitado 2024-06-17]. Disponible desde: `https://www.visual-paradigm.com/features/` (vid. pág. 15).

PEÑA O'SHEA, Sergio de la, 2017. *SGBD e instalación*. Ediciones Paraninfo, SA (vid. pág. 15).

PERALTA MARINI, Jhonny Yerson e HILASACA APAZA, Ever Saul, 2020. Desarrollo de un sistema de control de inventarios para pymes comercializadoras aplicando la metodología personalizada de XP (vid. pág. 20).

PÉREZ, Susana Graciela; QUISPE, José Rolando; MULLICUNDO, Felipe Fernando y LAMAS, Daniel Alberto, 2021. Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd. En: *Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd*. *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*, págs. 963-968 (vid. págs. 10, 14).

PYTHON, Why, 2021. Python. *Python releases for windows*. Vol. 24 (vid. pág. 12).

QAHRAMON O'G'LI, Mamayusufov Mirkomil et al., 2024. CSS (CASCADING STYLE SHEETS). *International journal of artificial intelligence*. Vol. 4, n.º 03, págs. 562-565 (vid. pág. 14).

RIANO Nossa, Norma Daniela, 2021. Estudio comparativo de metodologías tradicionales y ágiles aplicadas en la gestión de proyectos (vid. pág. 20).

ROCKCONTENT, 2021. *Dashboard: ¿qué es y cómo utilizar uno en tu empresa?* [online] [visitado 2024-06-17]. Disponible desde: `https://rockcontent.com/es/blog/dashboard/` (vid. pág. 6).

RODRIGUEZ, Miguel Angel, 2024. Aprendizaje Basado en Problemas para el Modelado de Datos. *Revista de Innovación y Buenas Prácticas Docentes*. Vol. 13, n.º 1, págs. 77-90 (vid. pág. 29).

SAABITH, Sayeth; VINOTHRAJ, T y FAREEZ, M, 2021. A review on Python libraries and Ides for Data Science. *Int. J. Res. Eng. Sci*. Vol. 9, n.º 11, págs. 36-53 (vid. pág. 16).

SABORIDO Loidi, José Ramón, 2018. Universidad, investigación, innovación y formación doctoral para el desarrollo en Cuba. *Revista Cubana de Educación Superior*. Vol. 37, n.º 1, págs. 4-18 (vid. pág. 4).

SOLÍS MACIAS, Nathalie Dayana, 2023. *Análisis comparativo de lenguajes de programación PHP y Java web orientados a los servicios de internet.* B.S. thesis. Babahoyo: UTB-FAFI. 2023 (vid. pág. 12).

TAMAYO ESPINOSA, Lisandra y SILEGA MARTÍNEZ, Nemury, 2021. Gestión de la mantenibilidad desde etapas tempranas en el desarrollo de software. *Revista Cubana de Ciencias Informáticas*. Vol. 15, n.º 1, págs. 52-69 (vid. pág. 31).

TIAMA, Yunapanta y EUCLIDES, Gilson, 2023. *Estudio comparativo del rendimiento de los servidores web IIS y APACHE.* B.S. thesis. Babahoyo: UTB-FAFI. 2023 (vid. pág. 17).

VÁZQUEZ, Juan Ignacio Cerca; GONZÁLEZ, Luis Alberto López; FARÍAS, José Jesús Sánchez; AGUAYO, Oscar Grimaldo y MÉNDEZ, Ramón Eduardo Mendoza, 2018. Uso de herramientas CASE para la gestión de proyectos de software. *Pistas Educativas*. Vol. 35, n.º 110 (vid. pág. 15).

VIDAL-SILVA, Cristian L; SÁNCHEZ-ORTIZ, Aurora; SERRANO, Jorge y RUBIO, José M, 2021. Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formación universitaria*. Vol. 14, n.º 5, págs. 85-94 (vid. pág. 17).

Generado con LaTeX: 27 de noviembre de 2024: 6:00pm

Apéndices

APÉNDICE A

Entrevista

Preguntas realizadas a la Dr. C. Yamilis Fernández Pérez:

- ¿Cuál es la meta principal del programa de capacitación doctoral?
- ¿Cuáles son las clases de datos e información que se manejan en la actualidad para los doctorandos?
- ¿Cómo se conservan y estructuran actualmente los datos de los doctorandos?
- ¿Cuáles son los retos o inconvenientes que enfrenta actualmente al administrar la información de los doctorandos?
- ¿Qué funcionalidades o cualidades específicas le gustaría tener en el sistema de gestión de información de doctorandos?

APÉNDICE B

Historias de Usuario

Tabla B.1. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Gestionar Centro
Usuario: Administrador, Gerente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un centro a partir de los campos nombre y organismo. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los centros.	
Observaciones: Ninguna	

Tabla B.2. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Gestionar Doctor
Usuario: Administrador, Gerente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un doctor a partir de los campos persona y facultad. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los doctores.	
Observaciones: Depende de que haya personas y áreas registradas en el sistema.	

Tabla B.3. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Gestionar Doctorando

Continúa en la próxima página

Tabla B.3. Continuación de la página anterior

Usuario: Administrador, Gerente	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un doctorando a partir de los campos fecha de defensa, fecha de ingreso, tema de tesis, fecha de inglés, fecha de especialidad, desarrollo local, persona, facultad, programa y sector estratégico al que pertenece. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los doctorandos.	
Observaciones: Depende de que haya personas, áreas, programas y sectores estratégicos registradas en el sistema.	

Tabla B.4. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Gestionar País
Usuario: Administrador, Gerente	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un país a partir de los campos nombre y código. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los países.	
Observaciones: Ninguna	

Tabla B.5. Historia de usuario # 8

Historia de usuario	
Número: 8	**Nombre:** Gestionar Programa
Usuario: Administrador, Gerente	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un programa a partir de los campos nombre, sector estratégico, desarrollo local, a distancia, estado de desarrollo municipal, área y área de conocimiento al que pertenece. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los programas.	
Observaciones: Depende de que haya áreas y áreas de conocimiento registradas en el sistema.	

Tabla B.6. Historia de usuario # 9

Historia de usuario	
Número: 9	**Nombre:** Gestionar Sector Estratégico

Continúa en la próxima página

Tabla B.6. Continuación de la página anterior

Usuario: Administrador, Gerente	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un sector estratégico a partir del campo nombre. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los sectores estratégicos.	
Observaciones: Ninguna	

Tabla B.7. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Gestionar Tutor
Usuario: Administrador, Gerente	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un tutor a partir de los campos doctor y doctorando. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los tutores.	
Observaciones: Depende de que haya doctores y doctorandos registradas en el sistema.	

Tabla B.8. Historia de usuario # 11

Historia de usuario	
Número: 11	**Nombre:** Gestionar Graduado
Usuario: Administrador, Gerente	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario añadir un graduado a partir de los campos fecha de defensa, persona, facultad y área de conocimiento. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los graduados.	
Observaciones: Depende de que haya personas, áreas y áreas de conocimiento registradas en el sistema.	

Tabla B.9. Historia de usuario # 12

Historia de usuario	
Número: 12	**Nombre:** Iniciar Sesión
Usuario: Administrador, Gerente, Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A

Continúa en la próxima página

Tabla B.9. Continuación de la página anterior

Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario acceder al sistema al introducir su usuario y contraseña.	
Observaciones: Depende de que haya usuarios registrados en el sistema.	

Tabla B.10. Historia de usuario # 13

Historia de usuario	
Número: 13	**Nombre:** Cerrar Sesión
Usuario: Administrador, Gerente, Usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario cerrar la sesión y salir del sistema al seleccionar la opción de cerrar sesión.	
Observaciones: Depende de que haya usuarios autenticado en el sistema.	

Tabla B.11. Historia de usuario # 14

Historia de usuario	
Número: 14	**Nombre:** Mostrar Estadísticas
Usuario: Administrador, Gerente, Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al usuario visualizar las estadísticas obtenidas de los datos en el sistema.	
Observaciones:	

Tabla B.12. Historia de usuario # 15

Historia de usuario	
Número: 15	**Nombre:** Gestionar Usuario
Usuario: Administrador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: El sistema debe permitir al administrador añadir un usuario a partir de los campos nombre de usuario, contraseña, correo y rol. Además debe permitir modificar, eliminar, buscar y ver el listado de todos los usuarios.	
Observaciones: Ninguna	

APÉNDICE C

Tarjetas CRC

Tabla C.1. Tarjeta CRC # 4

Tarjeta CRC	
Clase: centro	
Responsabilidad	**Colaboración**
• Registrar los centros • Modificar los centros • Eliminar los centros • Buscar los centros • Mostrar el listado de los centros	CentroViewSet Centro_Serializer

Tabla C.2. Tarjeta CRC # 5

Tarjeta CRC	
Clase: doctor	
Responsabilidad	**Colaboración**
• Registrar los doctores • Modificar los doctores • Eliminar los doctores • Buscar los doctores • Mostrar el listado de los doctores	persona area DoctorViewSet Doctor_Serializer

Tabla C.3. Tarjeta CRC # 6

Tarjeta CRC
Clase: doctorando

Continúa en la próxima página

Tabla C.3. Continuación de la página anterior

Responsabilidad	**Colaboración**
• Registrar los doctorandos • Modificar los doctorandos • Eliminar los doctorandos • Buscar los doctorandos • Mostrar el listado de los doctorandos	area persona programa sectorest DoctorandoViewSet Doctorando_Serializer

Tabla C.4. Tarjeta CRC # 7

Tarjeta CRC	
Clase: pais	
Responsabilidad	**Colaboración**
• Registrar los países • Modificar los países • Eliminar los países • Buscar los países • Mostrar el listado de los países	PaisViewSet Pais_Serializer

Tabla C.5. Tarjeta CRC # 8

Tarjeta CRC	
Clase: programa	
Responsabilidad	**Colaboración**
• Registrar los programas • Modificar los programas • Eliminar los programas • Buscar los programas • Mostrar el listado de los programas	area areadeconocimiento ProgramaViewSet Programa_Serializer

Tabla C.6. Tarjeta CRC # 9

Tarjeta CRC	
Clase: sectorest	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla C.6. Continuación de la página anterior

• Registrar los sectores estratégicos • Modificar los sectores estratégicos • Eliminar los sectores estratégicos • Buscar los sectores estratégicos • Mostrar el listado de los sectores estratégicos	SectorestViewSet Sectorest_Serializer

Tabla C.7. Tarjeta CRC # 10

Tarjeta CRC	
Clase: tutor	
Responsabilidad	**Colaboración**
• Registrar los tutores • Modificar los tutores • Eliminar los tutores • Buscar los tutores • Mostrar el listado de los tutores	TutorViewSet Tutor_Serializer

Tabla C.8. Tarjeta CRC # 11

Tarjeta CRC	
Clase: graduado	
Responsabilidad	**Colaboración**
• Registrar los graduados • Modificar los graduados • Eliminar los graduados • Buscar los graduados • Mostrar el listado de los graduados	persona area areadeconocimiento GraduadoViewSet Graduado_Serializer

Tabla C.9. Tarjeta CRC # 12

Tarjeta CRC	
Clase: seguridad	
Responsabilidad	**Colaboración**
• Autenticar los usuarios en el sistema • Cerrar la sesión de los usuarios	usuario

Tabla C.10. Tarjeta CRC # 13

Tarjeta CRC	
Clase: estadistica	
Responsabilidad	**Colaboración**
• Calcular las estadísticas • Mostrar las estadísticas	doctor doctorando tutor graduado persona programa areadeconocimiento

APÉNDICE D

Tareas de Ingeniaría

Tabla D.1. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 2
Nombre de la tarea: Gestionar Área	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de las áreas.	

Tabla D.2. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 3
Nombre de la tarea: Gestionar Área de Conocimiento	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de las áreas de conocimiento.	

Tabla D.3. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 4
Nombre de la tarea: Gestionar Centro	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	

Continúa en la próxima página

Tabla D.3. Continuación de la página anterior

Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los centros.

Tabla D.4. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 5
Nombre de la tarea: Gestionar Doctor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los doctores.	

Tabla D.5. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 6
Nombre de la tarea: Gestionar Doctorando	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los doctorandos.	

Tabla D.6. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 7
Nombre de la tarea: Gestionar País	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los países.	

Tabla D.7. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 8
Nombre de la tarea: Gestionar Programa	

Continúa en la próxima página

Tabla D.7. Continuación de la página anterior

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los programas.	

Tabla D.8. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 9
Nombre de la tarea: Gestionar Sector Estratégico	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los sectores estratégicos.	

Tabla D.9. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 10
Nombre de la tarea: Gestionar Tutor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los tutores.	

Tabla D.10. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 11
Nombre de la tarea: Gestionar Graduado	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de la información de los graduados.	

Tabla D.11. Tarea de ingeniería # 14

Tarea

Continúa en la próxima página

Tabla D.11. Continuación de la página anterior

Número de tarea: 14	**Número de Historia de usuario:** 12,13
Nombre de la tarea: Seguridad del sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para la gestión de sesión de usuarios.	

Tabla D.12. Tarea de ingeniería # 15

Tarea	
Número de tarea: 15	**Número de Historia de usuario:** 14
Nombre de la tarea: Mostrar Estadística	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Programador responsable: Orlián Mesa Cáceres	
Descripción: Desarrollar los métodos y las interfaces necesarias para mostrar las estadísticas.	

APÉNDICE E

Pruebas Unitarias

Figura E.1. Resultado AreadeconocimientoTests

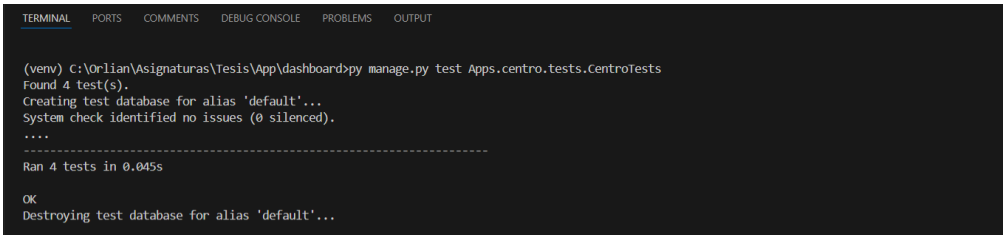


Figura E.2. Resultado CentroTests

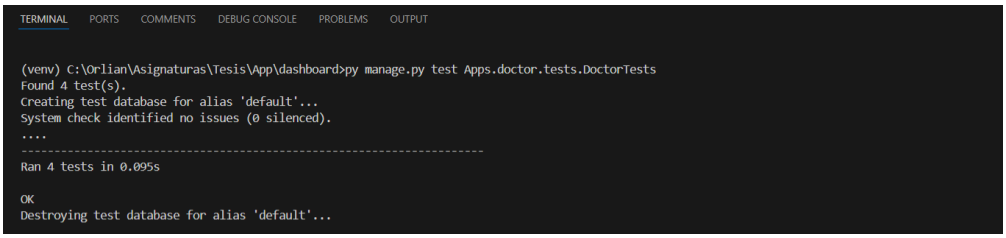


Figura E.3. Resultado DoctorTests

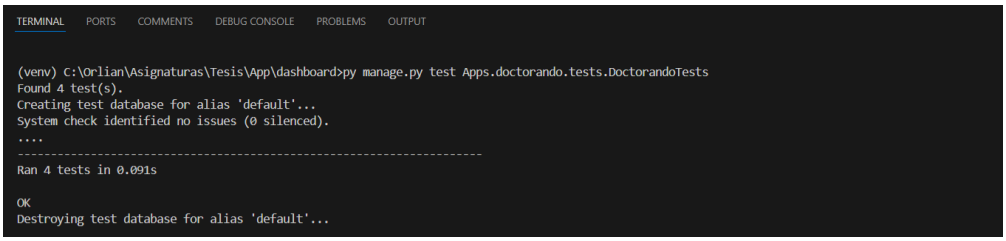


Figura E.4. Resultado DoctorandoTests

Figura E.5. Resultado GraduadoTests

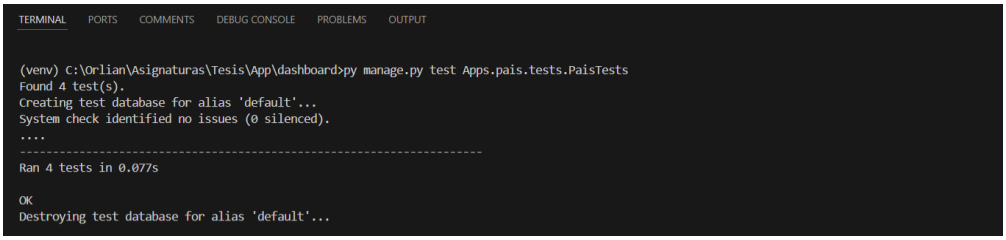


Figura E.6. Resultado PaisTests

Figura E.7. Resultado PersonaTests

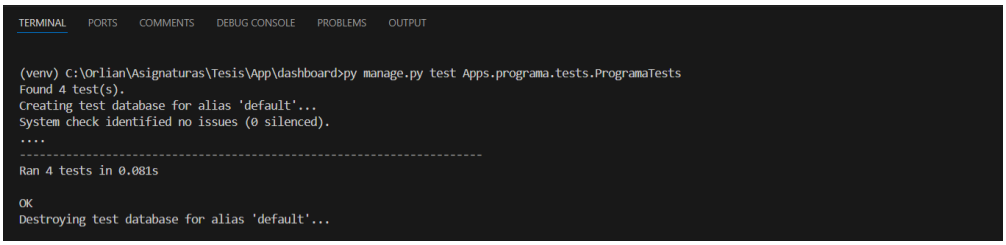


Figura E.8. Resultado ProgramaTests

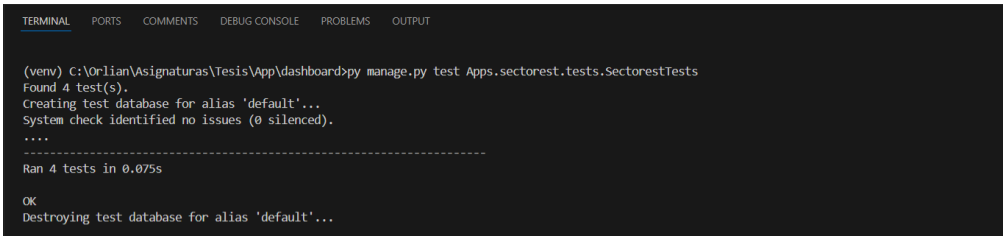


Figura E.9. Resultado SectorestTests

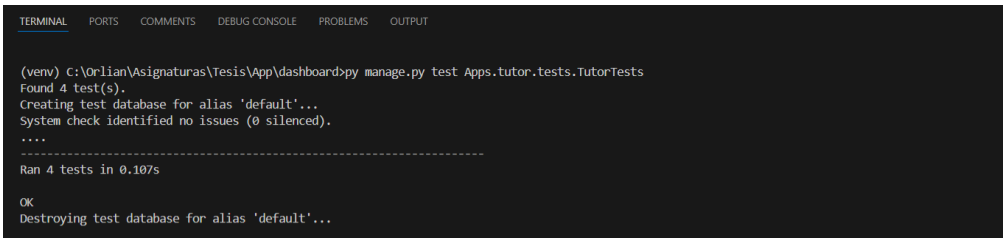


Figura E.10. Resultado TutorTests

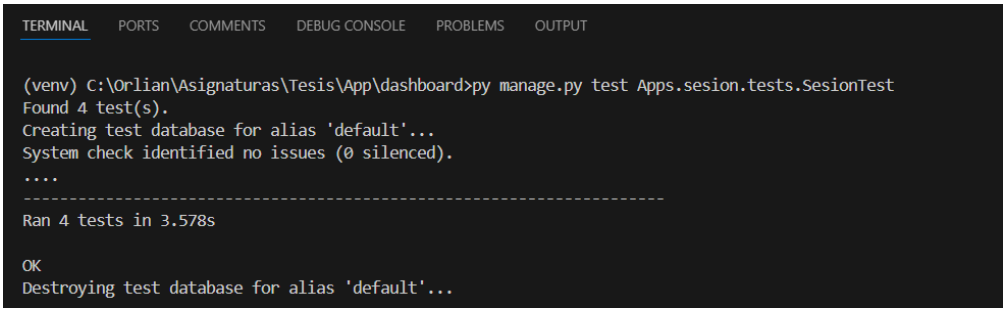


Figura E.11. Resultado SesionTests

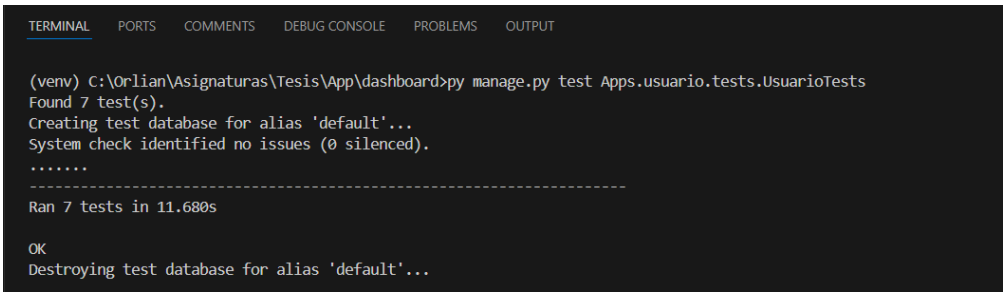


Figura E.12. Resultado UsuarioTests

APÉNDICE F

Pruebas de Aceptación

Tabla F.1. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Listar Área	
Descripción: Prueba la funcionalidad de listar las áreas.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Área.	
Resultados esperados: El sistema carga una página con el listado de áreas.	

Tabla F.2. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Registrar Área	
Descripción: Prueba la funcionalidad de registrar un área.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un área si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.3. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU2_P3	**Historia de usuario:** 2
Nombre: Modificar Área	
Descripción: Prueba la funcionalidad de modificar un área.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un área si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.4. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU2_P4	**Historia de usuario:** 2
Nombre: Eliminar Área	
Descripción: Prueba la funcionalidad de eliminar un área.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un área del sistema.	

Tabla F.5. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Listar Área de Conocimiento	
Descripción: Prueba la funcionalidad de listar las áreas de conocimiento.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Área de Conocimiento.	
Resultados esperados: El sistema carga una página con el listado de áreas de conocimiento.	

Tabla F.6. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** 3
Nombre: Registrar Área de Conocimiento	
Descripción: Prueba la funcionalidad de registrar un área de conocimiento.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas de conocimiento.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un área de conocimiento si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.7. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Modificar Área de Conocimiento	
Descripción: Prueba la funcionalidad de modificar un área de conocimiento.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas de conocimiento.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un área de conocimiento si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.8. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU3_P4	**Historia de usuario:** 3
Nombre: Eliminar Área de Conocimiento	
Descripción: Prueba la funcionalidad de eliminar un área de conocimiento.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de áreas de conocimiento.	

Continúa en la próxima página

Tabla F.8. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema elimina un área de conocimiento del sistema.

Tabla F.9. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Listar Centro	
Descripción: Prueba la funcionalidad de listar los centros.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Centro.	
Resultados esperados: El sistema carga una página con el listado de centros.	

Tabla F.10. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU4_P2	**Historia de usuario:** 4
Nombre: Registrar Centro	
Descripción: Prueba la funcionalidad de registrar un centro.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de centros.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un centro si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.11. Prueba de aceptación # 15

Caso de prueba de aceptación	
Código: HU4_P3	**Historia de usuario:** 4
Nombre: Modificar Centro	

Continúa en la próxima página

Tabla F.11. Continuación de la página anterior

Descripción: Prueba la funcionalidad de modificar un centro.
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de centros.
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema modifica un centro si los datos fueron correctos, en caso contrario muestra un mensaje de error.

Tabla F.12. Prueba de aceptación # 16

Caso de prueba de aceptación	
Código: HU4_P4	**Historia de usuario:** 4
Nombre: Eliminar Centro	
Descripción: Prueba la funcionalidad de eliminar un centro.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de centros.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un centro.	

Tabla F.13. Prueba de aceptación # 17

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Listar Doctor	
Descripción: Prueba la funcionalidad de listar los doctores.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Doctor.	
Resultados esperados: Resultados esperados: El sistema carga una página con el listado de doctores.	

Tabla F.14. Prueba de aceptación # 18

Caso de prueba de aceptación	
Código: HU5_P2	**Historia de usuario:** 5
Nombre: Registrar Doctor	
Descripción: Prueba la funcionalidad de registrar un doctor.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctores.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un doctor si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.15. Prueba de aceptación # 19

Caso de prueba de aceptación	
Código: HU5_P3	**Historia de usuario:** 5
Nombre: Modificar Doctor	
Descripción: Prueba la funcionalidad de modificar un doctor.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctores.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un doctor si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.16. Prueba de aceptación # 20

Caso de prueba de aceptación	
Código: HU5_P4	**Historia de usuario:** 5
Nombre: Eliminar Doctor	
Descripción: Prueba la funcionalidad de eliminar un doctor.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctores.	

Continúa en la próxima página

Tabla F.16. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema elimina un doctor.

Tabla F.17. Prueba de aceptación # 21

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Listar Doctorando	
Descripción: Prueba la funcionalidad de listar los doctorandos.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Doctorando.	
Resultados esperados: El sistema carga una página con el listado de doctorandos.	

Tabla F.18. Prueba de aceptación # 22

Caso de prueba de aceptación	
Código: HU6_P2	**Historia de usuario:** 6
Nombre: Registrar Doctorando	
Descripción: Prueba la funcionalidad de registrar un doctorando.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctorandos.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un doctorando si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.19. Prueba de aceptación # 23

Caso de prueba de aceptación	
Código: HU6_P3	**Historia de usuario:** 6
Nombre: Modificar Doctorando	

Continúa en la próxima página

Tabla F.19. Continuación de la página anterior

Descripción: Prueba la funcionalidad de modificar un doctorando.
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctorandos.
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema modifica un doctorando si los datos fueron correctos, en caso contrario muestra un mensaje de error.

Tabla F.20. Prueba de aceptación # 24

Caso de prueba de aceptación	
Código: HU6_P4	**Historia de usuario:** 6
Nombre: Eliminar Doctorando	
Descripción: Prueba la funcionalidad de eliminar un doctorando.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de doctorandos.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un doctorando.	

Tabla F.21. Prueba de aceptación # 25

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario:** 7
Nombre: Listar País	
Descripción: Prueba la funcionalidad de listar los países.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción País.	
Resultados esperados: El sistema carga una página con el listado de países.	

Tabla F.22. Prueba de aceptación # 26

Caso de prueba de aceptación	
Código: HU7_P2	**Historia de usuario:** 7
Nombre: Registrar País	
Descripción: Prueba la funcionalidad de registrar un país.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de países.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un país si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.23. Prueba de aceptación # 27

Caso de prueba de aceptación	
Código: HU7_P3	**Historia de usuario:** 7
Nombre: Modificar País	
Descripción: Prueba la funcionalidad de modificar un país.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de países.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un país si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.24. Prueba de aceptación # 28

Caso de prueba de aceptación	
Código: HU7_P4	**Historia de usuario:** 7
Nombre: Eliminar País	
Descripción: Prueba la funcionalidad de eliminar un país.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de países.	

Continúa en la próxima página

Tabla F.24. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema elimina un país.

Tabla F.25. Prueba de aceptación # 29

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** 8
Nombre: Listar Programa	
Descripción: Prueba la funcionalidad de listar los programas.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Programa.	
Resultados esperados: El sistema carga una página con el listado de programas.	

Tabla F.26. Prueba de aceptación # 30

Caso de prueba de aceptación	
Código: HU8_P2	**Historia de usuario:** 8
Nombre: Registrar Programa	
Descripción: Prueba la funcionalidad de registrar un programa.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de programas.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un programa si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.27. Prueba de aceptación # 31

Caso de prueba de aceptación	
Código: HU8_P3	**Historia de usuario:** 8
Nombre: Modificar Programa	

Continúa en la próxima página

Tabla F.27. Continuación de la página anterior

Descripción: Prueba la funcionalidad de modificar un programa.
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de programas.
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema modifica un programa si los datos fueron correctos, en caso contrario muestra un mensaje de error.

Tabla F.28. Prueba de aceptación # 32

Caso de prueba de aceptación	
Código: HU8_P4	**Historia de usuario:** 8
Nombre: Eliminar Programa	
Descripción: Prueba la funcionalidad de eliminar un programa.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de programas.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un programa.	

Tabla F.29. Prueba de aceptación # 33

Caso de prueba de aceptación	
Código: HU9_P1	**Historia de usuario:** 9
Nombre: Listar Sector Estratégico	
Descripción: Prueba la funcionalidad de listar los sectores estratégicos.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Sector Estratégico.	
Resultados esperados: El sistema carga una página con el listado de sectores estratégicos.	

Tabla F.30. Prueba de aceptación # 34

Caso de prueba de aceptación	
Código: HU9_P2	**Historia de usuario:** 9
Nombre: Registrar Sector Estratégico	
Descripción: Prueba la funcionalidad de registrar un sector estratégico.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de sectores estratégicos.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un sector estratégico si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.31. Prueba de aceptación # 35

Caso de prueba de aceptación	
Código: HU9_P3	**Historia de usuario:** 9
Nombre: Modificar Sector Estratégico	
Descripción: Prueba la funcionalidad de modificar un sector estratégico.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de sectores estratégicos.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un sector estratégico si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.32. Prueba de aceptación # 36

Caso de prueba de aceptación	
Código: HU9_P4	**Historia de usuario:** 9
Nombre: Eliminar Sector Estratégico	
Descripción: Prueba la funcionalidad de eliminar un sector estratégico.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de sectores estratégicos.	

Continúa en la próxima página

Tabla F.32. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema elimina un sector estratégico.

Tabla F.33. Prueba de aceptación # 37

Caso de prueba de aceptación	
Código: HU10_P1	**Historia de usuario:** 10
Nombre: Listar Tutor	
Descripción: Prueba la funcionalidad de listar los tutores.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Tutor.	
Resultados esperados: El sistema carga una página con el listado de tutores.	

Tabla F.34. Prueba de aceptación # 38

Caso de prueba de aceptación	
Código: HU10_P2	**Historia de usuario:** 10
Nombre: Registrar Tutor	
Descripción: Prueba la funcionalidad de registrar un tutor.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de tutores.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un tutor si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.35. Prueba de aceptación # 39

Caso de prueba de aceptación	
Código: HU10_P3	**Historia de usuario:** 10
Nombre: Modificar Tutor	

Continúa en la próxima página

Tabla F.35. Continuación de la página anterior

Descripción: Prueba la funcionalidad de modificar un tutor.
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de tutores.
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema modifica un tutor si los datos fueron correctos, en caso contrario muestra un mensaje de error.

Tabla F.36. Prueba de aceptación # 40

Caso de prueba de aceptación	
Código: HU10_P4	**Historia de usuario:** 10
Nombre: Eliminar Tutor	
Descripción: Prueba la funcionalidad de eliminar un tutor.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de tutores.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un tutor.	

Tabla F.37. Prueba de aceptación # 41

Caso de prueba de aceptación	
Código: HU11_P1	**Historia de usuario:** 11
Nombre: Listar Graduado	
Descripción: Prueba la funcionalidad de listar los graduados.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Graduado.	
Resultados esperados: El sistema carga una página con el listado de graduados.	

Tabla F.38. Prueba de aceptación # 42

Caso de prueba de aceptación	
Código: HU11_P2	**Historia de usuario:** 11
Nombre: Registrar Graduado	
Descripción: Prueba la funcionalidad de registrar un graduado.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de graduados.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un graduado si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.39. Prueba de aceptación # 43

Caso de prueba de aceptación	
Código: HU11_P3	**Historia de usuario:** 11
Nombre: Modificar Graduado	
Descripción: Prueba la funcionalidad de modificar un graduado.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de graduados.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un graduado si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.40. Prueba de aceptación # 44

Caso de prueba de aceptación	
Código: HU11_P4	**Historia de usuario:** 11
Nombre: Eliminar Graduado	
Descripción: Prueba la funcionalidad de eliminar un graduado.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de graduados.	

Continúa en la próxima página

Tabla F.40. Continuación de la página anterior

Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.
Resultados esperados: El sistema elimina un graduado.

Tabla F.41. Prueba de aceptación # 45

Caso de prueba de aceptación	
Código: HU12_P1	**Historia de usuario:** 12
Nombre: Autenticar Usuario	
Descripción: Prueba la funcionalidad de acceder al sistema a través de la autenticación de un usuario.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema.	
Pasos de ejecución: El usuario accede a la URL del sistema a través de un navegador web, el sistema muestra el formulario de inicio de sesión, el usuario introduce el nombre de usuario y contraseña y selecciona la opción Aceptar.	
Resultados esperados: El sistema muestra la página principal del sistema si los datos son correctos, en caso contrario muestra un mensaje de error.	

Tabla F.42. Prueba de aceptación # 46

Caso de prueba de aceptación	
Código: HU13_P1	**Historia de usuario:** 13
Nombre: Cerrar Sesión	
Descripción: Prueba la funcionalidad de cerrar la sesión.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Pasos de ejecución: El usuario selecciona la opción Cerrar Sesión en la barra de menú superior.	
Resultados esperados: El sistema muestra la página de login y cierra la sesión.	

Tabla F.43. Prueba de aceptación # 47

Caso de prueba de aceptación	
Código: HU14_P1	**Historia de usuario:** 14

Continúa en la próxima página

Tabla F.43. Continuación de la página anterior

Nombre: Mostrar estadísticas
Descripción: Prueba la funcionalidad de mostrar estadísticas.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.
Pasos de ejecución: El usuario accede a la pantalla principal.
Resultados esperados: El sistema muestra las gráficas de estadísticas.

Tabla F.44. Prueba de aceptación # 48

Caso de prueba de aceptación	
Código: HU15_P1	**Historia de usuario:** 15
Nombre: Listar Usuario	
Descripción: Prueba la funcionalidad de listar los usuarios.	
Condiciones de ejecución: El usuario debe estar autenticado.	
Pasos de ejecución: El usuario selecciona en el menú lateral la opción Usuario.	
Resultados esperados: El sistema carga una página con el listado de usuarios.	

Tabla F.45. Prueba de aceptación # 49

Caso de prueba de aceptación	
Código: HU15_P2	**Historia de usuario:** 15
Nombre: Registrar Usuario	
Descripción: Prueba la funcionalidad de registrar un usuario.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de usuarios.	
Pasos de ejecución: El usuario selecciona el botón de Añadir, el sistema carga una página con un formulario, el usuario introduce los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema registra un usuario si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.46. Prueba de aceptación # 50

Caso de prueba de aceptación	
Código: HU15_P3	**Historia de usuario:** 15
Nombre: Modificar Usuario	
Descripción: Prueba la funcionalidad de modificar un usuario.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de usuarios.	
Pasos de ejecución: El usuario selecciona el botón de Modificar, el sistema carga una página con un formulario con los datos cargados, el usuario modifica los datos necesarios y verifica que sean correctos, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema modifica un usuario si los datos fueron correctos, en caso contrario muestra un mensaje de error.	

Tabla F.47. Prueba de aceptación # 51

Caso de prueba de aceptación	
Código: HU15_P4	**Historia de usuario:** 15
Nombre: Eliminar Usuario	
Descripción: Prueba la funcionalidad de eliminar un usuario.	
Condiciones de ejecución: El usuario debe estar autenticado, debe tener cargado el listado de usuarios.	
Pasos de ejecución: El usuario selecciona el botón de Eliminar, el sistema muestra un cuadro de confirmación, el usuario presiona el botón Aceptar.	
Resultados esperados: El sistema elimina un usuario.	

APÉNDICE G

Indicadores

Indicadores y Metas:

No.	Líneas Asociadas	Indicador	Siglas
1	Proyección Institucional 2030	Porciento de doctores tutores	PDrT
2	Doctores y Tutores	Cantidad de doctorandos por doctor tutor	NDxDrT
3	Doctores y Tutores	Cantidad promedio de doctores dirigidos y defendidos por doctor tutor	PrmDExDr
4	Doctores y Tutores	Número de actividades orientadas a la superación de doctores y tutoría	NAc2Dr
5	Programas de Doctorado	Número de programas de doctorado	NPD
6	Programas de Doctorado	Porciento de programas acreditables acreditados	PPA
7	Programas de Doctorado	Número de actividades formativas por líneas de investigación especificando modalidad (formación especialidad)	NAcLI
8	Programas de Doctorado	Porciento de las líneas de investigación tienen actividades a distancia en la plataforma	PAcaD
9	Programas de Doctorado	Número de tesis asociadas al desarrollo local	NTLocal
10	Programas de Doctorado	Número de tesis asociadas a sectores estratégicos	NTSE
11	Ingreso y Doctorandos	Porciento de doctorandos menor de 35 años	P-35
12	Ingreso y Doctorandos	Número de doctorandos	NDrd
13	Ingreso y Doctorandos	Número de defensas efectuadas por renglón (interna, externa, sector estratégico, desarrollo local)	NDefreg(int, ext, se, dl)
14	Alianzas interinstitucionales e internacionalización	Número de estudiantes extranjeros	NDExt
15	Alianzas interinstitucionales e internacionalización	Número de profesores extranjeros	NDrExt
16	Alianzas interinstitucionales e internacionalización	Número de actividades de formación doctoral con participación de profesores extranjeros	NActDrExt

Figura G.1. Indicadores