Facultad de Ciencias y Tecnologías Computacionales

Título: Componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

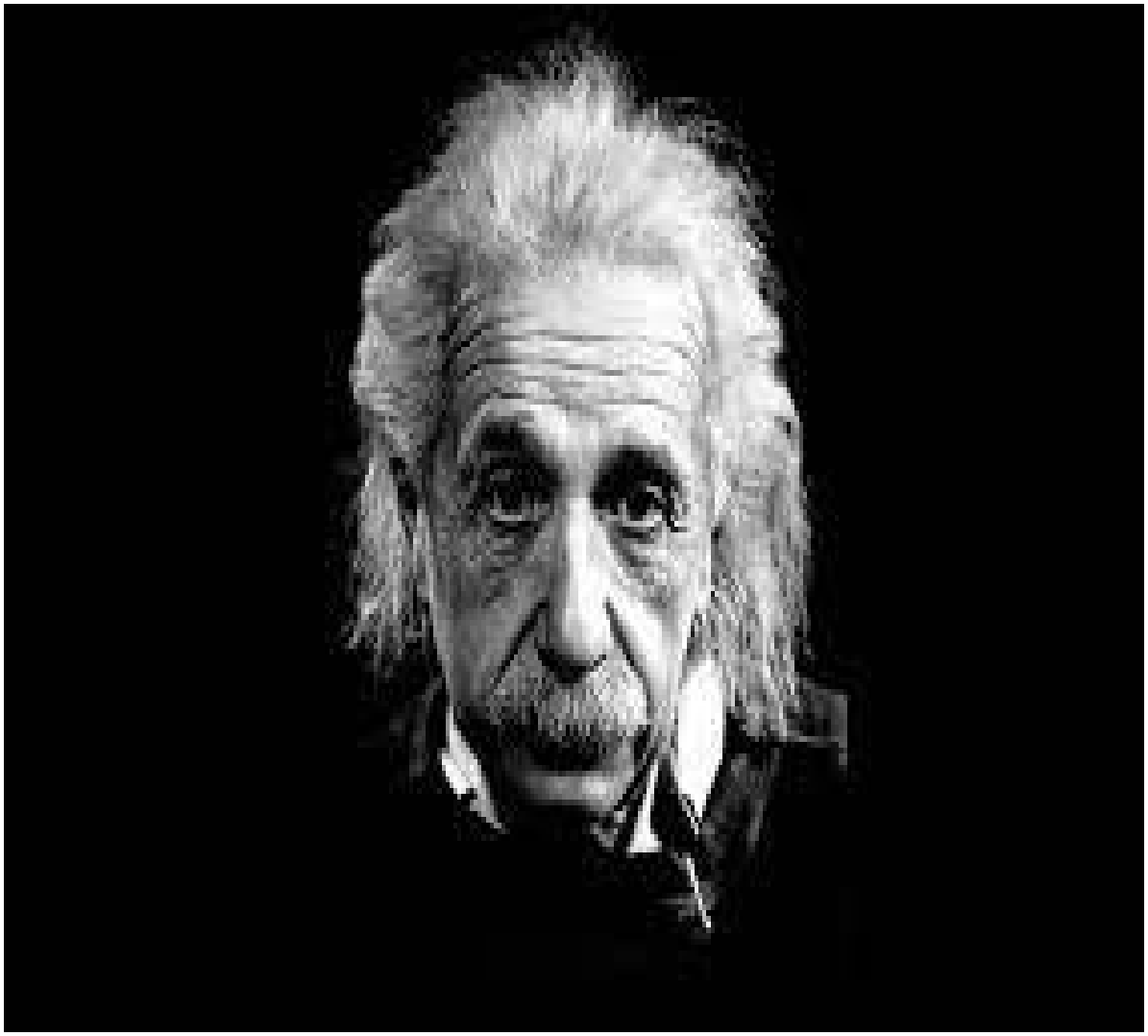
Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: José Joel Camejo Rodríguez

Tutor: Ing. Ángel Ernesto Hernández Bandera

La Habana, mayo de 2022

Año 64 de la Revolución



Nosotros, los mortales, logramos la inmortalidad en las cosas que creamos en común y que quedan después de nosotros.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas que tiene por título “Componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop” y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de ____ del año ____.

Ing. Ángel Ernesto Hernández Bandera

_____
Firma del Tutor

DATOS DE CONTACTO

Autor:

José Joel Camejo Rodríguez

Universidad de las Ciencias Informáticas (UCI)

e-mail: josejcr@estudiantes.uci.cu

Tutores

Ing. Ángel Ernesto Hernández Bandera

Universidad de las Ciencias Informáticas

e-mail: aehernandez@uci.cu

AGRADECIMIENTOS

De pequeño mi familia me enseño que se debe dar gracias por todas las cosas buenas (y también malas de la vida). Por eso, en esta tesis me encantaría agradecer.

Gracias, a mi madre y a mí padre, que me dieron todo lo que necesité, ustedes han sido siempre el motor que impulsa mis sueños y esperanzas, me apoyaron siempre en todo momento con su sostén incondicional, demostrándome que si se puede y que la peor batalla es aquella que no nos atrevemos a luchar y junto a ustedes no me cabe duda que todo se puede.

Me encantaría agradecerle a mi abuela Ela por todos su sacrificio , por todas las oraciones que me dedicó, por todas las veces que me consintió contar de que yo fuera feliz.

A mi hermano Hernán que siempre ha estado presente en todo momento brindándome su cariño, amistad y ayuda ilimitada para que yo pudiera seguir adelante.

Mi tío Dioelis que estoy seguro que se sentirá muy orgulloso al contar que tiene un sobrino que es Ingeniero, a ti muchas gracias por todos los consejos que me brindas día a día.

A mi novia que se ha convertido en una de las personas más importantes de mi vida, gracias por ser parte de mi día a día, por las peleas para que no me rindiera y así seguir adelante, gracias por todo ese amor que me manifiestas en todo momento demostrándome que eres la persona que quiero tener en a mi lado, gracias por llegar y no irte.

A mis amigos y compañeros de viaje Juampi, Karel, Adriel, Rainer, Frank, Eleandris, Adrian, Miguel, Luis David, Pablo, hoy culminan esta maravillosa aventura y no puedo dejar de recordar todos esos momentos que hemos vividos juntos. Hoy nos toca cerrar un capítulo maravilloso en esta historia de vida y no puedo dejar de agradecerles por su apoyo y constancia, al estar en las horas más difíciles, por compartir horas de estudio. Gracias por estar siempre allí.

A mi suegra y a Otto por toda su ayuda incondicional y amor en todo momento, a toda mi familia en general que se que siempre he podido contar con ustedes para todo lo que ha hecho falta.

A todos mis profesores que me han ayudado a lo largo de toda mi vida como estudiante.

Gracias a la vida por este triunfo ,gracias a todas las personas que de una forma u otra me apoyaron y me demostraron que no estaba solo, este titulo de Ingeniero en Ciencias Informáticas es para ustedes también.

¡¡¡A todos, muchas Gracias !!! .... Jochi

DEDICATORIA

Le dedico el resultado de este trabajo a toda mi familia. Principalmente, a mis padres que me apoyaron y contuvieron en los momentos malos y en los menos malos. Gracias por enseñarme a afrontar las dificultades sin perder nunca la cabeza ni morir en el intento. Me han enseñado a ser la persona que soy hoy, mis principios, mis valores, mi perseverancia y mi empeño. Todo esto con una enorme dosis de amor y sin pedir nada a cambio. Gracias, muchas gracias, los amo.

RESUMEN

XAUCE AKADEMOS es una herramienta multiplataforma que contribuye al perfeccionamiento de los procesos académicos de una institución. Su uso permite el desarrollo coherente de una estrategia organizacional en los diferentes niveles de la estructura educacional con la que cuenta el Ministerio de Educación de la República de Cuba. El siguiente trabajo se enmarca en desarrollar un componente, el cual permita la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop. Para guiar el desarrollo de la solución propuesta se utilizó la metodología AUP variación UCI, así como el Framework de desarrollo Django, y como sistema de bases de datos PostgreSQL, entre otras herramientas y tecnologías. Se realizaron pruebas unitarias, de sistemas y de integración, evidenciando que el componente se encontraba libre de errores y listo para ser utilizado. Como resultado se obtiene un componente el cual permite la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

PALABRAS CLAVE

Componente, Gestión Académica, Integración, Interoperabilidad, Plataforma

ABSTRACT

XAUCE AKADEMOS is a multiplatform tool that contributes to the improvement of the academic processes of an institution. Its use allows the coherent development of an organizational strategy at the different levels of the educational structure of the Ministry of Education of the Republic of Cuba. The following work is framed in developing a component, which allows the integration between the Academic Management System XAUCE AKADEMOS for MI-NED and XAUCE AKADEMOS Desktop. To guide the development of the proposed solution, the AUP UCI variation methodology was used, as well as the Django development Framework, and PostgreSQL database system, among other tools and technologies. Unit, system and integration tests were performed, showing that the component was free of errors and ready to be used. As a result, a component was obtained which allows the integration between the Academic Management System XAUCE AKADEMOS for MINED and XAUCE AKADEMOS Desk-top.

KEYWORDS

Component, Academic Management, Integration, Interoperability, Platform

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La gestión académica en los establecimientos educativos de todo el mundo juega un papel de vital importancia en la educación para mejorar los índices de eficiencia y eficacia de cada institución educativa, como aporte al mejoramiento de la calidad del aprendizaje, es un proceso sistemático que está orientado al fortalecimiento de las instituciones educativas y a sus proyectos, con el fin de enriquecer los procesos pedagógicos, directivos, comunitarios y administrativos; conservando la autonomía institucional, para así responder de una manera más acorde, a las necesidades educativas locales, regionales y mundiales (1).

Para mejorar el alcance a diferentes tipos de población, la educación se ha apoyado en tecnologías soportadas sobre Internet denominadas *e-learning* (2). El *e-learning* ha permitido a la educación estructurarse en modalidades presenciales, semipresenciales y virtuales. Todo en procura de realizar prácticas pedagógicas a menudo personalizadas para lograr la interiorización y apropiación efectiva de los conocimientos que conforman el proceso educativo (3). Las nuevas tecnologías han afectado todas las áreas del desarrollo social y se han convertido en elementos inherentes al desarrollo en todas las esferas de la vida. En el campo de la educación, su impacto cubre numerosas áreas de aplicación, desde la educación a distancia hasta la utilización de terminales portátiles y herramientas informáticas en las escuelas. Otro efecto muy importante es el impacto de internet, el cual puede compensar problemas como el bajo nivel de capacitación de los educadores o la falta de material educativo (4).

Investigar y conocer cómo ha transcurrido la formación de maestros y profesores, es un elemento que hace que la memoria histórica educacional revolucionaria esté presente en las transformaciones actuales y futuras. El pasado es raíz de lo presente. Ha de saberse lo que fue porque lo que fue está en lo que es (5).

Hoy día, Cuba no está excepto de los avances tecnológicos que existen en el mundo, es por ello que el sector de la educación es uno de los más favorecidos con estos adelantos, pues ha permitido una mejor forma de enseñar y aprender. La Universidad de las Ciencias Informáticas (UCI) es una de las instituciones encargadas de la producción de aplicaciones y servicios informáticos en nuestro país, precisamente el Centro de Tecnologías para la Formación (FORTES) adscrito a la Facultad de Tecnologías Educativas. Este centro tiene como misión: desarrollar tecnologías que permitan ofrecer servicios y productos con el fin de brindar soluciones de formación aplicando las Tecnologías de la Información y las Comunicaciones (TIC), a todo tipo de instituciones con diferentes modelos de formación y condiciones tecnológicas. En esta institución se desarrollan productos y servicios para la implemen-

tación de soluciones de formación. También garantiza la calidad de las soluciones y la formación de los recursos humanos a partir de investigaciones que combinan los elementos pedagógicos y tecnológicos más avanzados, integrando así los procesos de formación, producción e investigación.

El Sistema de Gestión Académica XAUCE AKADEMOS para el MINED es desarrollado por este Centro. El software permite la gestión del proceso académico a partir de varios módulos, a través de los cuales se controla el proceso de matriculación, las evaluaciones, planes de estudio, asistencias, solicitudes de cursos y prácticas profesionales, entre otras.

XAUCE AKADEMOS es una herramienta multiplataforma que contribuye al perfeccionamiento de los procesos académicos de una institución. Su uso permite el desarrollo coherente de una estrategia organizacional en los diferentes niveles de la estructura educacional con la que cuenta el Ministerio de Educación de la República de Cuba. Debido a la complejidad que suponía desarrollar un software de su tipo, fue necesario emplear las mejores prácticas de desarrollo disponibles, desde el punto de vista arquitectónico. Entre las opciones más acordes, para dar solución a las necesidades del cliente, teniendo en cuenta las características técnicas de la infraestructura del MINED, se encontraba la adopción de una arquitectura orientada a microservicios, la cual entre sus virtudes permite un desarrollo desacoplado de componentes con independencia del lenguaje o tecnología, y que pueden ser integrados a la solución con gran facilidad, luego de rebasar un proceso de pruebas satisfactoriamente.

En nuestro país el bloqueo económico, político y financiero impuesto por los estados Unidos continúa siendo el principal impedimento para poder obtener un superior flujo de información y un mejor acceso a internet y las (TIC). Esto trae consigo que no se pueda contar con la infraestructura necesaria para poder llevar a todos los rincones de la isla el internet. Es por esto que existe el impedimento de que este software pueda ser utilizado en todos los centros educativos de Cuba, es así que brota la necesidad de crear una solución informática para poder importar y exportar la información de las instituciones educativas que cuenten con equipos informáticos de bajas prestaciones y otras que no pueden contar con el servicio a Internet, las cuales pueda utilizar este software de manera offline.

Tomando como punto de partida de esta investigación la situación descrita previamente, se identifica el siguiente **problema científico**: ¿Cómo integrar la información entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop?

Por lo que se deduce como **objeto de estudio** la integración entre sistemas informáticos y como **campo de acción** proceso de integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

Para dar solución al problema planteado se definió como **objetivo general** de esta investigación: Desarrollar un componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

Del **objetivo general** trazado, se derivan los siguientes **objetivos específicos**:

- Establecer los referentes teóricos y metodológicos relacionados a la integración entre sistemas informáticos en específico las plataformas virtuales y sistemas de gestión de información.
- Realizar el análisis y diseño del componente propuesto para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.
- Implementar un componente que permita la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.
- Caracterización de los principales conceptos asociados a la integración entre sistemas de gestión académica.
- Análisis de soluciones tecnológicas para la integración entre sistemas de gestión académica.
- Selección de requisitos e infraestructura para el desarrollo de la solución.
- Implementación de los requisitos de la solución para la integración entre sistemas de gestión académica.
- Aplicación de las pruebas necesarias que demuestren la efectividad de la solución.

Para dar cumplimiento a los **objetivos específicos** planteados, se proponen las siguientes **tareas de investigación**:

- Análisis de soluciones similares existentes.
- Investigación y selección de las técnicas, herramientas y metodologías a emplear en el desarrollo del componente.
- Implementación de un componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.
- Definición de los instrumentos y métodos para la validación de la propuesta de solución.
- Validación de la propuesta de solución.

Como **posible resultado** se desarrollará un componente que garantice la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

Para la realización de la investigación fueron empleados **métodos empíricos y teóricos**, con el objetivo de dar solución a la problemática planteada.

Los **métodos teóricos** utilizados para desarrollar la investigación son:

Teóricos:

Permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad (6).

Análisis histórico – Lógico: se utiliza con el fin de analizar bibliografías y sistemas homólogos para sintetizar los elementos importantes que se relacionan con esta investigación. Este método se utiliza también para tomar decisiones en cuanto a los resultados arrojados por las encuestas, las entrevistas realizadas y así asentar las bases teóricas del desarrollo de la tesis.

Analítico – Sintético: utilizado para el análisis y estudio de documentos que se basen en el marco teórico de la investigación con el objetivo de conocer teóricamente cómo han evolucionado los sistemas de gestión de información referente a los procesos de investigación, desarrollo e innovación tecnológica para así extraer los elementos más importantes de los mismos.

Modelación: se emplea con el objetivo de crear modelos y diagramas que son abstracciones del producto final, permitiendo tener un dominio inicial de la información que se va a modelar, representar de forma estática los requisitos y obtener una versión del sistema original para validar los requisitos con el cliente.

Métodos Empíricos:

Estos describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional (6).

Observación: usado para la recopilación de datos y necesidades existentes en el desarrollo del proyecto, permite comparar observaciones obtenidas a través de vías diferentes y escoger la mejor.

Entrevista: se utiliza para conocer las necesidades del cliente y recopilar información para el desarrollo de la propuesta de solución. Se realiza de forma semi-abierta mediante conversaciones de carácter profesional con el cliente.

Consulta de fuentes de información: se emplea para la elaboración del marco teórico de la investigación y permite realizar un estudio actual de las distintas bibliografías que existen sobre la integración entre sistemas de gestión académica.

El documento estará estructurado en tres capítulos organizados de la siguiente forma:

Capítulo 1. Fundamentación teórica: en este capítulo se hace referencia a los elementos teóricos en los cuales está basado la investigación, contiene un estudio del estado del arte de este tema. Se exponen los lenguajes de programación utilizados, así como las metodologías, herramientas y tecnologías en el desarrollo de la solución. Se encuentran los principales conceptos relacionados con el contenido. Además, se realiza un análisis de las soluciones similares a nivel nacional e internacional, y las arquitecturas de software para una selección de la más adecuada para la solución propuesta.

Capítulo 2. Análisis y diseño del componente: en esta sección se describe la propuesta de solución para integrar los sistemas antes mencionados, se detallan las principales características, junto a la modelación del flujo de información del sistema, se describen los patrones arquitectónicos y los de diseños utilizados; se realiza una explicación de la técnica utilizada y del procedimiento seguido para el desarrollo del sistema.

Capítulo 3. Validación de la propuesta de solución: en este apartado se describen las fases de implementación y pruebas. Se realiza la implementación de todas las funcionalidades identificadas, logrando un componente que logre satisfacer las principales necesidades del cliente. Se detallan además las pruebas realizadas al sistema, una vez que concluye la implementación, para asegurar que este cumple con las especificaciones requeridas, para de esa manera asegurar la calidad y eficiencia de la solución.

CAPÍTULO I: Fundamentación Teórica

En este capítulo se exponen los principales conceptos relacionados con la fundamentación teórica. También se analizan las soluciones similares a nivel mundial y nacional hasta la actualidad. Además, se incluyen las descripciones de las herramientas, tecnologías, metodologías y lenguajes de programación que se utilizarán en el desarrollo de la solución.

1.1 Conceptos asociados al dominio del problema

XAUCE AKADEMOS: es una herramienta multiplataforma que contribuye al perfeccionamiento de los procesos académicos de una institución. Su uso permite el desarrollo coherente de una estrategia organizacional en los diferentes niveles de la estructura educacional con la que cuenta el Ministerio de Educación de la República de Cuba.

Interoperabilidad:

Según la norma estadounidense ANSI/NISO Z39.19:2005 y la norma británica BS8723-4:2007, coinciden en definir la interoperabilidad como la capacidad que tienen dos o más sistemas o componentes de intercambiar información y usar esa información que se ha intercambiado (7).

Según la definición que expresa la Red de Transparencia y Acceso a la información (RTA) la interoperabilidad es la habilidad de los sistemas de tecnologías de la información y las comunicaciones (TIC) y de los procesos de negocio que ellas soportan, de intercambiar datos y posibilitar compartir información y conocimiento (8).

Como definen José Manuel Huidobro Moya y David Martínez Roldán, la Interoperabilidad es un concepto básico para un correcto funcionamiento en el marco de la administración electrónica. Se define también como la habilidad necesaria para que organizaciones y sistemas diversos puedan interactuar con objetivos consensuados y comunes, además de tener como finalidad común la persecución de beneficios comunes. La interoperabilidad se estructura en tres dimensiones: técnica, semántica y organizativa (9).

Teniendo en cuenta los conceptos planteados anteriormente sobre la interoperabilidad es la capacidad que tienen dos sistemas informáticos o más para juntos poder intercambiar información, conocimiento para así lograr el beneficio común entre estos sistemas informáticos.

Sistemas Informáticos: Un sistema de información es un conjunto de datos que interactúan entre sí con un fin común. En informática, los sistemas de información ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización. La importancia de un sistema de información radica en la eficiencia en la correlación de una gran cantidad de datos ingresados a través de procesos diseñados para cada área con el objetivo de producir información válida para la posterior toma de decisiones (10).

1.2 Análisis de soluciones similares en el mundo

El desarrollo de las TIC, ha impulsado el surgimiento de diferentes iniciativas encaminadas a lograr la interoperabilidad entre las herramientas de apoyo a los procesos de enseñanza-aprendizaje. En este ámbito, existen disímiles plataformas educativas que implementan las principales tecnologías, estándares de interoperabilidad entre sistemas y protocolos de comunicación utilizados a nivel mundial. Para así obtener conocimiento sobre la integración entre sistemas que existen en el mundo y hallar una solución al problema existente. Algunas de las soluciones relevantes son analizadas a continuación.

Integración entre Moodle y Mahara (Mahoodle).

¿Qué es Moodle?

Moodle es un sistema de Gestión de cursos (*Open Source Course Management System,* CMS), conocido también como un Sistema de Gestión del Aprendizaje (*Learning Management System,* LMS) o como un Entorno de Aprendizaje Virtual (*Virtual Learning Environment*, VLE). Es una aplicación web donde los docentes pueden utilizar para crear lugares de aprendizaje efectivo en línea. La palabra Moodle corresponde al acrónimo de Modular *Object-Oriented Dynamyc Learning Environment*. De una manera más coloquial, podemos decir que Moodle es un paquete de software para la creación de cursos y sitios Web basados en Internet, o sea, una aplicación para crear y gestionar plataformas educativas, es decir, espacios donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por unos docentes y organiza el acceso a esos recursos por los estudiantes, y además permite la comunicación entre todos los implicados (11).

¿Qué es Mahara?

Mahara es una aplicación web de código abierto para gestionar e-Portafolios y redes sociales. Nace en Nueva Zelanda a mediados del año 2006 como proyecto de colaboración en el que participaron las Universidades de Massey *University*, *Auckand University of Tecnology*, *The* Open *Polytechnic* of New *Zeland*, and Victoria *University* of *Wellington*. Dicha aplicación puede ayudar a crear un Entorno Personal de Aprendizaje conocido como PLE/PLN (Personal *Learning Environment*/*Network*), en contraste con la mayoría de Sistemas de Gestión de Aprendizaje más centrado en la institución (LMS). Por lo que Mahara es una aplicación más centrada en el aprendiz que en el curso (11).

"Mahoodle" es el nombre común que se ha dado a la integración entre Mahara (plataforma de portafolio en línea-*ePortfolio*) y *Moodle* (sistema de gestión de conocimiento en línea LMS), ambas plataformas de código libre las cuales estarán integradas de tal manera que le permitirá: Inicio de sesión único y transferencia de contenido entre plataformas. Los dos sistemas se han incorporado en el apoyo a los otros en forma de:

- Inicio de sesión único (SSO, Single Sing On)
- Transferencia de contenido: Permite exportar diversos tipos de objetos desde Moodle a Mahara (API de portafolio) y permite importar objetos desde Mahara a Moodle (API de repositorio). Sólo se puede vincular uno a uno Moodle Mahara ya sea en el nivel de lugar o en una institución individual. No se puede vincular Moodle en diversas instituciones en Mahara (11).

Delphos

Delphos es el sistema de gestión administrativa y académica de los centros educativos de Castilla-La Mancha. Integra en un único sistema las funcionalidades necesarias para llevar a cabo la gestión de los centros educativos, cubriendo todos los procesos administrativos y académicos de los Centros Educativos e implica a todos los gestores de la red educativa. El sistema permite agilizar el intercambio de datos entre Centros, y eliminar flujos de información redundantes con la Consejería, aprovechando las posibilidades de las tecnologías actuales. El sistema presta sus servicios a través de Internet, permitiendo el acceso desde cualquier punto y en cualquier momento (12).

1.2.1 Análisis de soluciones similares en Cuba

Nuestro país también se ha encargado de desarrollar proyectos los cuales contribuyen a la integración entre sistemas. Algunos ejemplos de estos se muestran a continuación.

Integración del Juez en Línea Caribeño y la plataforma educativa Zera 2.0.

Plataforma educativa Zera 2.0: Es una plataforma de gestión de aprendizaje, donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por unos docentes y organiza el acceso a esos recursos por los estudiantes, y además permite la comunicación entre todos los implicados (alumnado y profesorado), contribuyendo de esta manera a la evolución de los procesos de enseñanza y aprendizaje. Es decir, Zera2.0 se encarga, entre otros aspectos, de presentar los cursos a los usuarios y del seguimiento de la actividad de los estudiantes (13).

Juez en línea: Es la evolución de sistemas similares llamados evaluadores automáticos. Es una aplicación web para entornos generalmente académicos, que incluye varios problemas de distintas materias para ser resueltos mediante técnicas de programación, además, evalúa automáticamente las soluciones de sus usuarios en los disímiles lenguajes de programación disponibles (14).

Interacción entre la plataforma Croda y Rhoda.

Croda:

Es una herramienta de autor web creada en la UCI, que facilita la creación de Objetos de Aprendizaje interoperables, reutilizables, accesibles y duraderos, de manera flexible, aplicando para ello el estándar para la descripción de recursos *Learning Object Metadata* (LOM) y el estándar de contenido *Sharable* Content *Object* Reference *Model* (SCORM). Esta herramienta de autor web permite la creación de cursos para los distintos entornos educativos.

La herramienta Croda consume servicios web del repositorio Rhoda de tipo SOAP, aunque estas dos plataformas se encuentran ya actualmente en desuso, tienen una gran importancia en cuanto a ejemplo de consumir servicios web (15).

Rhoda:

Es un Repositorio de Objetos de Aprendizaje (ROA), desarrollado por especialistas de la UCI. Tiene como objetivo fundamental almacenar y gestionar los objetos de aprendizaje que se crean por parte de la comunidad universitaria. Es una aplicación web modular y multiplataforma, que provee un lugar común accesible a través de un navegador, donde los usuarios pueden almacenar, recuperar y consultar objetos de aprendizaje, destinados a fortalecer el proceso de enseñanza-aprendizaje (15).

Tabla 1. Analisis de soluciones encontradas.

Fuente. Elaboración propia

Sistema	Internacionales	Tipo de Software	Estado	Código Abierto	Permite el cifrado de datos
Moodle	Internacional	Privada	Activa	No	No
Mahara	Internacional	Privada	Activa	No	No
Delphos	Internacional	Privada	Altiva	No	No
Zera 2.0	Nacional	Libre	Absoleta	Si	No
Croda	Nacional	Libre	Absoleta	Si	No
Rhoda	Nacional	Libre	Absoleta	Si	No

Estas soluciones internacionales proveen gran cantidad de facilidades Sin embargo, las mismas son soluciones con características avanzadas de pago y su adquisición y puesta en marcha requiere el desembolso de altas sumas de dinero, no acorde con las políticas de independencia tecnológica adoptadas en el país y en la UCI.

De manera general, no resulta favorable la utilización de algunas de las soluciones antes mencionadas; debido a que estas contienen funcionalidades que no son de interés para la propuesta de solución atentando contra la propia concepción.

Luego de realizar el estudio de soluciones homólogas y determinar la necesidad de confeccionar una nueva solución, se hace necesario hacer un análisis de la metodología que regirá el proceso de desarrollo, así como las tecnologías y herramientas a emplear. La adopción y puesta en marcha de lo antes expuesto estará vinculado a las experiencias adquiridas en el estudio de las soluciones homólogas.

1.3 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayuda a documentar el desarrollo de productos (16). Estas a su vez se dividen en ágiles y tradicionales.
Las metodologías tradicionales se focalizan en documentación, planificación y procesos. Plantillas, técnicas de administración, revisiones, etc. Ejemplos de estas metodologías son: Proceso Racional Unificado (RUP) y Microsoft *Solution Frame* (MSF).

Las metodologías ágiles ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significar reducir su coste. Ejemplos de estas metodologías son Programación Extrema (XP) y Proceso Unificado Ágil (AUP) (17).

Principios de agilidad

Se define 12 principios de agilidad para aquellos que la quieran alcanzar (17):

- La prioridad más alta es satisfacer al cliente a través de la entrega pronta y continua de software valioso.
- Son bienvenidos los requerimientos cambiantes, aun en una etapa avanzada del desarrollo.
- Los procesos ágiles dominan el cambio para provecho de la ventaja competitiva del cliente.
- Entregar con frecuencia software que funcione, de dos semanas a un par de meses, de preferencia lo más pronto que se pueda.
- Las personas de negocios y los desarrolladores deben trabajar juntos, a diario y durante todo el proyecto.
- Hay que desarrollar los proyectos con individuos motivados. Debe darse a éstos el ambiente y el apoyo que necesiten, y confiar en que harán el trabajo.
- El método más eficiente y eficaz para transmitir información a los integrantes de un equipo de desarrollo, y entre éstos, es la conversación cara a cara.

- La medida principal de avance es el software que funciona.
- Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben poder mantener un ritmo constante en forma indefinida.
- La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
- Es esencial la simplicidad: el arte de maximizar la cantidad de trabajo no realizado.
- Las mejores arquitecturas, requerimientos y diseños surgen de los equipos con organización propia.
- El equipo reflexiona a intervalos regulares sobre cómo ser más eficaz, para después afinar y ajustar su comportamiento en consecuencia.

Tabla 2. Diferencias entre metodologías ágiles y tradicionales.

Fuente: Elaboración propia.

Metodologías ágiles	**Metodologías tradicionales**
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Grupos pequeños (menos de diez integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas
Pocos roles.	Muchos roles.
Menos énfasis en la arquitectura de software	La arquitectura de software es esencial y se expresa mediante modelos
Pocos artefactos.	Muchos artefactos.

1.3.1 Metodología a utilizar y su fundamentación.

En el desarrollo del componente se empleará la metodología definida por el centro FORTES para el desarrollo de proyectos. La misma consiste en una versión desarrollada por la UCI basada en la metodología de desarrollo de software AUP en su tercer escenario.

La UCI decidió hacer una variación de la metodología Proceso Unificado Ágil o Agile *Unified Process* (AUP) llamada AUP-UCI. Con esta adaptación de AUP se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define el estándar internacional de calidad CMMI-DEV v1.3. También permite que se adapte el ciclo de vida definido para la actividad productiva de la institución (18).

La Metodología de Desarrollo AUP-UCI tiene como eje principal la aplicación de técnicas ágiles incluyendo:

- Desarrollo dirigido por pruebas.
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de Base de datos para mejorar la productividad.

Esta metodología facilita el trabajo en proyectos de pequeña envergadura y proporciona un ambiente de desarrollo de software iterativo e incremental. En AUP-UCI sólo se utilizan los artefactos que son imprescindibles y realmente necesarios para la realización del producto. Aplica técnicas ágiles incluyendo: desarrollo dirigido por pruebas, modelado ágil, gestión de 12 cambios ágil y refactorización de base de datos para mejorar la productividad y también que permite realizar cambios durante la duración del proyecto (18).

AUP propone 7 disciplinas, pero para la variación del ciclo de vida de los proyectos de la Uci se tienen 7 disciplinas también, lo que a un nivel más atómico que el definido en AUP.

Definidas de la siguiente forma:

1. Modelado de negocio.
2. Requisitos.
3. Análisis y diseño.
4. Implementación.

5. Pruebas internas.
6. Pruebas de liberación.
7. Pruebas de aceptación.
8. Despliegue.

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

- Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.
- Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.
- Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.
- Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU (19).

De esta metodología se decide usar el escenario 4 por ser un proyecto en el que se pretende tener el negocio muy bien definido y se tiene la intención de que el cliente esté siempre acompañando al equipo de desarrollo, además de no ser un proyecto muy extenso. En este escenario se brindan resultados completos en las iteraciones para llegar a un producto final de manera creciente, desarrollando al menos un requisito en cada iteración.

1.4 Herramientas y tecnologías asociadas al desarrollo del componente

A continuación, se proporciona una relación de las principales herramientas y tecnologías que se emplearán durante el desarrollo del componente, así como la fundamentación de la elección realizada.

1.4.1 Framework de desarrollo.

Django (V 4.0)

Es un entorno de programación en Python diseñado para el desarrollo de aplicaciones web. Es un *framework* sencillo de instalar, posee una documentación muy completa y extensa, y que viene con

un Sistema Gestor de Bases de Datos ya incorporado. Fomenta un desarrollo rápido, un diseño limpio y pragmático. Construido por desarrolladores experimentados, se encarga de gran parte de las molestias del desarrollo web, para que puedas centrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto (20).

Características generales del *framework* Django:

- La utilización de un sistema de modelos de bases de datos, que hace que el uso de los Gestores de Bases de Datos sea independiente de la aplicación, ya que el uso se realiza mediante código en la aplicación (20).

- La posibilidad de dividir la aplicación, en aplicaciones con funcionalidades pequeñas que luego se junten en una, permitiendo darle al código un aspecto más conciso, ya que cada aplicación se encargará de un requisito de la aplicación (20).

- Permite el uso de andamiaje o *scaffolding*, que consiste en usar plantillas de código predefinidas, para determinadas situaciones, para generar el código final mediante el que el programador podrá especificar como se usa la base de datos, es decir, permite leer, crear, modificar y eliminar entradas de la base de datos (20).

Angular (V 12.1.4)

Angular es un *framework opensource* desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (*Single Page Application*). Separa completamente el *frontend* y el *backend* en la aplicación, evita escribir código repetitivo y mantiene todo más ordenado gracias a su patrón MVC (Modelo Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones (21).
Entre otras ventajas, este *framework* es modular y escalable adaptándose a nuestras necesidades y al estar basado en el estándar de componentes web, y con un conjunto de interfaz de programación de aplicaciones (API) permite crear nuevas etiquetas HTML personalizadas que pueden reutilizarse (21).
El lenguaje principal de programación de Angular es *Typescript*, y así toda la sintaxis y el modo de hacer las cosas en el código es el mismo, lo que añade coherencia y consistencia a la información, permitiendo, por ejemplo, la incorporación de nuevos programadores, en caso de ser necesarios, ya que pueden continuar su trabajo sin excesiva dificultad (21).

1.4.2 Sistemas de bases de datos.

PostgreSQL (v 11.1)

Es un SGBD objeto-relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Se ejecuta en la mayoría de los sistemas operativos, incluidos Linux, UNIX y Windows. Admite conjuntos de caracteres internacionales, codificaciones de caracteres *multibyte*, Unicode, y es compatible con el formateo y la distinción entre mayúsculas y minúsculas. Es altamente escalable tanto en la gran cantidad de datos que puede administrar como en la cantidad de usuarios simultáneos que puede manejar. Hay sistemas PostgreSQL activos en entornos de producción que administran más de 4 terabytes de datos (22).

Es conocido como el SGBD de código abierto más avanzado del mundo. Presenta estabilidad y confiabilidad, es extensible, multiplataforma, existen excelentes herramientas gráficas de diseño y administración de bases de datos y tiene buena escalabilidad ya que es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta (22).

1.4.3 Almacenamiento de datos en memoria

Redis (V 6.0.6)

Es un "almacén de base de datos en memoria" de código abierto muy popular, muy potente y muy utilizado como "almacén de estructura de base de datos en memoria". Se ejecuta como un proceso en segundo plano y se puede acceder a él localmente o a través de una red, lo que lo convierte en una opción muy popular para los "cachés de datos" (23).

Redis cuenta con replicación integrada, scripts Lua, desalojo LRU, transacciones y diferentes niveles de persistencia en disco, y proporciona alta disponibilidad a través de Redis *Sentinel* y partición automática con Redis *Cluster* (23).

Redis está escrito en ANSI C y funciona en la mayoría de los sistemas POSIX como Linux, *BSD y Mac OS X, sin dependencias externas. Linux y OS X son los dos sistemas operativos en los que más se ha desarrollado y probado Redis, por lo que recomendamos utilizar Linux para su despliegue. Redis puede funcionar en sistemas derivados de Solaris como *SmartOS*, pero el soporte es el mejor esfuerzo. No hay soporte oficial para las construcciones de Windows (24).

1.4.4 Sistema operativo

Ubuntu Server 20.04 LTS (soporte extendido)

Es una distribución de GNU/Linux, es un software libre que nació a partir de la distribución de Debian, y es conocida por el gestor integrado de paquetes que facilita la instalación de las aplicaciones eficientemente, esta versión es orientada tanto a usuarios de escritorios como a servidores, ya que brida una base estable y minimalista para cumplir con los requisitos de la infraestructura tecnológica y permite la ejecución de varias aplicaciones de forma optimizada para las diferentes necesidades de la organización (25).

Como dato curioso, Ubuntu nació desde una ideología étnica sudafricana para distinguir las alianzas y relaciones con los demás, entonces la traducción de la palabra Ubuntu enfocada a la distribución vendría siendo el objetivo de conectar a la humanidad por medio de enlaces, por medio de facilidad y libertad de uso; esta distribución es patrocinada por Canonical y el mantenimiento destinado por una comunidad de desarrolladores (26).

1.4.5 Salvas automáticas

Para realizar las salvas automáticas de la información del proyecto estas se realizarán a través de tareas programadas desde el servidor de aplicaciones.

Diariamente se realiza un *backup* de la base de datos en producción y se almacena en el servidor de medias distribuido (*MinIO*) configurado e integrado a la arquitectura del proyecto.

1.4.6 Herramienta de monitoreo

Grafana (V 5.0.4)

Para la realización del *dashboard* se decide utilizar *Grafana*, un software libre que permite construir un entorno de monitorización gráfico con diferentes tipos de paneles como pueda ser una tabla, un gráfico de líneas o de barras, etc. comunicándose directamente con las diferentes tablas de la base de datos para poder adquirir y representar dicha información. Permite el análisis y visualización de datos de serie temporales. A partir de una serie de datos recolectados obtendremos un panorama gráfico de la situación de una empresa u organización. Se utiliza frecuentemente para visualizar de una forma elegante series de datos en el análisis de infraestructuras y aplicaciones (27).

1.4.7 Sistema de modelado.

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. Es un lenguaje artificial que sirve para expresar un modelo con símbolos gráficos, en forma de diagramas. Al estar estandarizado permite a los desarrolladores documentar el software en cuanto a funcionalidades, procesos de negocios y conceptos asociados (28).

El Lenguaje de Modelado Unificado (UML: *Unified Modeling Language*) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s. Fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (26). Incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Es el lenguaje utilizado en el proyecto al cual va dirigido este trabajo de tesis (29).

Para el modelado con UML se utilizará Visual *Paradigm* for UML 16.2 por ser una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación.

1.5 Conclusiones del capítulo.

Los conceptos relacionados al dominio del problema enmarcado en este capítulo, enriquecieron el conocimiento sobre la integración entre sistemas de gestión informáticos, las plataformas y la interoperabilidad. El estudio sobre la integración entre plataformas virtuales, favoreció a elaborar una propuesta apropiada para poder dar solución al problema expuesto, no obstante estos sistemas no contribuyeron a la solución del componente, puesto que no se adaptan a las necesidades del proyecto, los cuales no tienen código abierto para así poder consultar información sobre sus funcionalidades, pues utilizan diferentes protocolos para la integración entre sistemas y trabajan con diferentes herramientas privativas.

CAPÍTULO II: Análisis y Diseño del componente.

El desarrollo que este trabajo persigue es un proceso de análisis y diseño el cual facilita los cimientos bajo los cuales se va a desarrollar el componente. Es por esto que en este capítulo se puntualizan los conocimientos de ingeniería de software, análisis y diseño que se involucran para el desarrollo del componente e implementación definidas por la metodología seleccionada. Se describen los requisitos funcionales (RF) y no funcionales (RnF) que responden al desarrollo de la solución, así como los patrones de diseño de servicios utilizados en la definición del componente propuesto.

2.1 Modelo del Negocio

Se puede identificar al modelo de negocio como un instrumento o estrategia que describe cómo una empresa u organización ofrece valor a sus clientes. Desde una visión más simplificada, un modelo de negocio proporciona información sobre el mercado objetivo de una organización, la necesidad de ese mercado y el papel que desempeñarán los productos o servicios de la empresa para satisfacer esas necesidades.

El modelo de negocio describe como desarrollar una nueva versión de la nueva organización basada en esta nueva versión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de clases de uso del negocio. Además, permite tener claridad en el modo por el cual se ofrece una solución a una necesidad del mercado. Estos modelos tienen que ser claros y definir las estrategias que deben emplearse en la búsqueda de los objetivos (30).

Debido al avanzado estado del sistema XAUCE AKADEMOS su negocio se encuentra modelado, es por esto que no se desarrollará en esta investigación.

2.2 Descripción de la propuesta de solución.

El sistema propuesto tiene como objetivo la integración de la información entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop. Para el desarrollo del componente se deben utilizar las herramientas capaces de permitir concluir la implementación de todas las funcionalidades requeridas. Es necesario que la propuesta cuente con un ambiente de fácil uso en cuanto a la manipulación de las principales acciones que intervienen en la misma. El componente debe poseer las funcionalidades que le permita importar y exportar la información mediante un archivo JSON (*JavaScript Object Notation*) entre el sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS *Desktop*, para una mejor seguridad del

fichero JSON, ya sea a la hora de exportar como de importar se le aplicará el estándar de interoperabilidad XML *Encryption* obteniendo así mayor seguridad a la hora de la importación y exportación de la información.

2.3 Determinación de los requisitos del sistema.

Alcanza todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requerimientos de los clientes. Estos se dividen en dos grupos: requisitos funcionales (RF) y requisitos no funcionales (RNF). Para a captura de requisitos la técnica utilizada fue la entrevsita que se encuenra en el anexo 3.

2.3.1 Requisitos Funcionales del componente.

Los requisitos funcionales permiten identificar las condiciones que debe cumplir un sistema para satisfacer un contrato, estándar, especificación u otra documentación formalmente impuesta. Estos describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones (31).

Para el correcto funcionamiento de la solución propuesta se espera que el sistema permita:

- RF1. Importar información desde el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop: Permite que el usuario pueda importar información entre los sistemas antes expuestos para así un mejor flujo de la información.

- RF2. Exportar información desde el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop: Permite al usuario poder exportar un documento con la información deseada.

2.3.2 Requisitos no Funcionales del componente.

Requerimientos que permiten definir las cualidades o propiedades que el software debe tener, con el objetivo de crear un producto final que sea fácil de utilizar, rápido y confiable. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como, por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del

sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, entre otros (32).

A continuación, se listan los requisitos no funcionales identificados:

Tabla 3. Descripción de los Requisitos no Funcionales (RnF).

Fuente: Elaboración propia.

No.	Clasificación	Descripción
RNF#1	Usabilidad	El componente debe contar con una adecuada organización de las acciones permitiendo la fácil interacción con el usuario.
RNF#2	Confidencialidad	Existencia de distintos roles que establezcan que la información solo sea vista por aquellos usuarios que posean los privilegios suficientes; restringir la ejecución de acciones a usuarios sin credenciales que intenten acceder a las mismas y la verificación de que el usuario esté autenticado antes de acceder a cualquier operación.
RNF#3	Interfaces externas	La interfaz debe ser simple, con una navegación por menú donde se pueda identificar de manera rápida las acciones a llevar. De este modo se logra una navegación efectiva en el sitio. Los mensajes deben ser claros, sin ambigüedades y con buena ortografía.

2.4 Historias de Usuario.

Una historia de usuario es la descripción de una necesidad en un lenguaje comprensible por cualquier persona sin conocimientos técnicos o informáticos. Éstas se usan en el contexto de la ingeniería de requisitos ágil como una herramienta de comunicación que combina los mejores atributos de ambos medios: escrito y verbal, y se centra en las necesidades o el problema que resuelve (Alexander Menzinsky et al., 2020). En total se definieron 2 HU, a continuación, se describe la perteneciente al

RF 2 debido a su prioridad para el cliente. El resto de las historias de usuarios podrán ser consultadas en el anexo del documento.

Tabla 4. Historia de Usuario RF #2 Exportar fichero.

Fuente**:** Elaboración propia.

Número: 2	**Requisito**: Exportar fichero
Programador: José Joel Camejo Rodríguez	**Iteración Asignada**: 1
Prioridad: Alta	**Tiempo Estimado**: 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real**: 6 horas
Descripción: Funcionalidad que permitirá exportar la información seleccionada de la base de datos. **Botón**: • Exportar: botón que permite exportar un fichero en el formato deseado.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

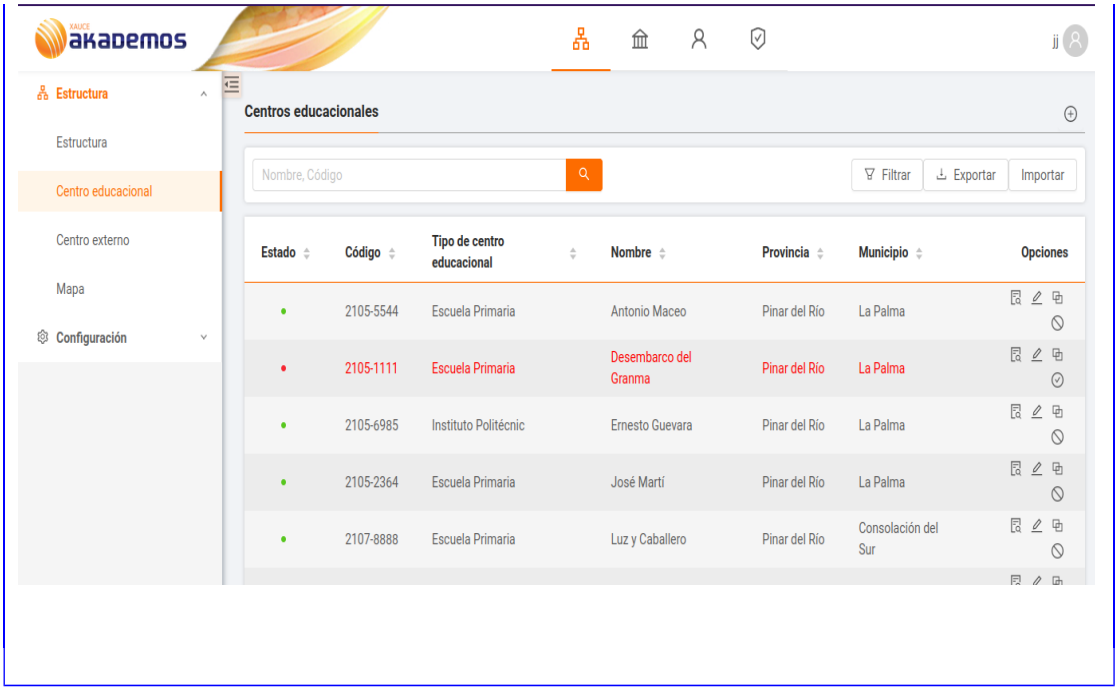


Tabla 5. Descripción de los requisitos funcionales Exportar Datos.

Fuente: Elaboración propia.

2.4.1 Descripción de los requisitos no funcionales del componente.

RnF 1. Requisito de Usabilidad.

1.1. El componente debe contar con una adecuada organización de las acciones permitiendo la fácil interacción con el usuario.

Tabla 6. Descripción del requisito no funcional Usabilidad 1.1.

Fuente: Elaboración propia.

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Comportamiento temporal
Objetivo	Proporcionar una adecuada organización de las acciones.
Origen	Humano
Artefacto	Sistema

Entorno	El sistema está funcionando correctamente

1.2. El sistema debe poseer un diseño “Responsive”.

Tabla 7. Descripción del requisito no funcional Usabilidad 1.2.

Fuente: Elaboración propia.

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Estética de la interfaz de usuario
Objetivo	Garantizar la adecuada visualización y organización de las acciones en múltiples dispositivos de acceso
Origen	Sistema
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Realizar una acción en el sistema	
El usuario accede a una vista del sistema.	Se muestra la interfaz correspondiente.
Medida de respuesta	
Se muestra la interfaz correspondiente adaptada al tamaño de la pantalla del dispositivo.	

1.3. El sistema debe poseer iconos que sugieran las acciones a realizar.

Tabla 8. Descripción del requisito no funcional Usabilidad 1.3.

Fuente: Elaboración propia.

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Capacidad de aprendizaje
Objetivo	Permitir al usuario aprender a reconocer las acciones en el sistema

Origen	Sistema
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo **1.a Acceder a una vista**	**Respuesta: Flujo de eventos (Escenarios)**
El usuario accede a una vista en el sistema.	Se muestra la vista correspondiente.
Medida de respuesta	
Los iconos presentes en la vista sugieren las acciones a realizar.	

RnF 2. Requisito de Seguridad.

1.1. El Sistema debe permitir que se establezca que la información solo sea vista por aquellos usuarios que posean los privilegios suficientes.

Tabla 9. Descripción del requisito no funcional Seguridad 1.1.

Fuente: Elaboración propia.

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	Confidencialidad
Objetivo	Proteger los datos y la información contra aquellos usuarios que no poseen privilegios suficientes.
Origen	Interno / Externo / No autorizado
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo **1.a Acceso no autorizado**	**Respuesta: Flujo de eventos (Escenarios)**
Intentos de acceso a una vista sin privilegios suficientes.	1. Se muestra el mensaje de información. 2. Se deniega el acceso a la acción.
Medida de respuesta	
No se muestra la vista	

1.2. El Sistema debe restringir la ejecución de acciones a usuarios sin credenciales que intenten acceder a las mismas.

Tabla 10. Descripción del requisito no funcional Seguridad 1.2.

Fuente: Elaboración propia.

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	Confidencialidad
Objetivo	Proteger los datos y la información contra accesos sin credenciales
Origen	Interno / Externo / No autorizado
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Acceso no autorizado	
Intentos de acceso a una vista sin cre denciales.	3. Se muestra el mensaje de información. 4. Se deniega el acceso a la acción.
Medida de respuesta	
No se muestra la vista	

1.3. El Sistema debe permitir la verificación de que el usuario esté autenticado antes de acceder a cualquier operación.

Tabla 11. Descripción del requisito no funcional Seguridad 1.3.

Fuente: Elaboración propia.

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	Confidencialidad
Objetivo	Proteger los datos y la información contra accesos sin autenticación
Origen	Interno / Externo / No autorizado
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Acceso no autorizado	
Intentos de acceso a una vista sin aut enticación.	1. Se muestra el mensaje de información. 2. Se deniega el acceso a la acción.
Medida de respuesta	
No se muestra la vista.	

RnF 3. Requisito de Apariencia o Interfaz Externa. El Sistema debe de tener interfaces externas simpl es.

Tabla 12. Descripción del requisito no funcional Apariencia o Interfaz externa.

Fuente: Elaboración propia.

Atributo de Calidad	Apariencia
Sub-atributos/Sub-características	Interfaz externa
Objetivo	Mostrar interfaces simples.
Origen	• Humano • Sistema
Artefacto	Sistema
Entorno	El sistema está funcionando correctamente
Estímulo **1.a Acciones del usuario**	**Respuesta: Flujo de eventos (Escenarios)**
El usuario accede a una vista del sistema.	1. Una interfaz simple. 2. La navegación por menú debe permitir identificar de manera rápida las acciones a llevar a cabo.
Medida de respuesta	
Los menús presentes en la vista sugieren las acciones a realizar.	

2.5 Descripción de la arquitectura.

Patrón arquitectónico MVT

Django está diseñado para promover un acoplamiento flexible y un aislamiento estricto entre los componentes de la aplicación. Esta filosofía facilita el cambio de ubicaciones específicas dentro de una aplicación sin afectar otros elementos.

Los sistemas que separan la lógica de acceso a la base de datos, la lógica empresarial y la lógica de presentación a menudo implican un concepto denominado patrón de arquitectura de software Modelo-Vista-Controlador (MVC). En este modelo, “modelo” se refiere al acceso a la carpeta de datos, “vista” se refiere a la parte del sistema que elige que mostrar y cómo, y “controlador” se refiere a la parte del sistema. Una vista que accede al modelo según sea necesario y se renderiza en función de la entrada del usuario (33).

Django también aplica este modelo. Los controladores son manejados por el mismo marco y la parte más importante ocurre en modelos, Modelos y Vista. Se sabe que Django usa el patrón arquitectónico *Model*, *View Template* (MVT) (34).

M: significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

T: significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: ¿cómo algunas cosas son mostradas sobre una página web u otro tipo de documento?

V: significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es como un puente entre los modelos y las plantillas.

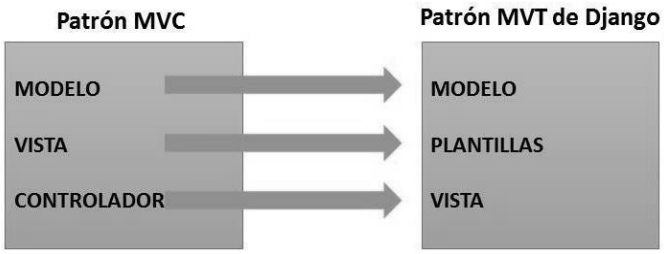


Ilustración 8*:* Relación estructural entre los MVC y MVT.

Fuente: Elaboració propia

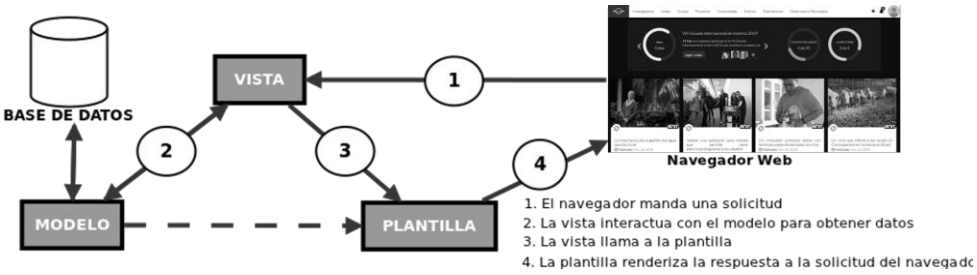


Ilustración 9: Ciclo del MVT.

Fuente: Elaboración propia

Para el desarrollo de la propuesta de solución se utiliza el patrón arquitectónico MVT debido a que este separa la lógica de negocio de la interfaz de usuario en tres capas diferentes, cada una con sus funcionalidades bien definidas, reduciendo el esfuerzo en la implementación del componente y así garantizando una mejor organización del trabajo.

2.6 Patrones de diseño empleados.

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.” En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares; ayudan al éxito del proyecto, pues permiten la reutilización de código, garantizan la robustez y extensibilidad del software (36).

Son útiles pues en el desarrollo del software:

- Ahorran tiempo.
- Te ayudan a estar seguro de la validez de tu código.
- Establecen un lenguaje común.

2.6.1 Patrones GRASP utilizados.

Se decide emplear para estructurar el diseño del sistema Patrones Generales de Software para Asignar Responsabilidades, más conocidos como patrones GRASP (General *Responsibility Asignment Software Patterns*, por sus siglas en inglés), los cuales serán muy útiles para lograr un software con alto grado de usabilidad (37).

Los patrones GRASP codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

Dentro de los patrones GRASP utilizados para el diseño del componente y que incluye por defecto en su arquitectura el *framework* de desarrollo seleccionado se destacan los siguientes:

Experto: consiste en asignar una responsabilidad al experto en información, se asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Se aplica el patrón de experto ya que existen métodos y/o clases encargadas específicamente en realizar una acción en este caso el método es el encargado de realizar la conversión a xml.

```
@login_required
def convertir_xml(request):
    if request.method=='POST':
            m=request.POST['tablax']
            mod=str(m)
            data=readfromjson("C:/Users/"+getuser()+"/Downloads/"+mod+".json")
            response1=HttpResponse(json2xml.Json2xml(data).to_xml())
            response1['Content-Disposition']='atachment; filename='+mod+".xml"''
            remove("C:/Users/"+getuser()+"/Downloads/"+mod+".json")
            return response1
```

Ilustración 1. Patrón Experto.

Fuente: Elaboración propia.

Controlador: Se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. Patrón Controlador se aplica en este caso porque existe una clase controladora ubicada en el archivo view.py la cual se encarga de la gestión de la lógica relacionada con lo que está en la vista en cuestión en este caso el acceso a todos los modelos de la base de datos de la aplicación.

```
class SubIndex(TemplateView):
    template_name='Exportar/modelos.html'

    def get_context_data(self,**kwargs):
        context={}
        app=apps.get_app_config(self.kwargs['app'])
        models=[]
        for model in app.get_models():
            models.append({'real_name':model.__name__,
                           'name':model.__name__.lower() })
        context['models']=models
        context['app']=self.kwargs['app'].upper()
        context['app_real']=self.kwargs['app']
        return context
```

Ilustración 2. Patrón Controlador.

Fuente: Elaboración propia.

2.6.2 Patrones GOF.

Los patrones de diseño GOF se clasifican en tres grupos: creacionales, estructurales y de comportamiento y comprenden un total de 23 patrones. A continuación, se muestran algunos de los que se evidencian en la solución (38).

Template Method **(Método plantilla):** Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.

En este caso se aplica mediante la creación de templates genéricos como este archivo llamado index.html el cual tiene un componente llamado block body.

Ilustración 3. Patrón Template Method (Método plantilla).

Fuente: Elaboración propia.

En este archivo llamado principal.html del proyecto se está heredando de la plantilla anterior utilizando el componente {% extends 'Base/index.html'%}.

```
{% extends 'Base/index.html' %}

{% load crispy_forms_tags %}
{% block body %}
<h4>Total de Modelos en la Base de Datos</h4>
<h4>{{total_models}}</h4>
{% for app in applications %}
<li>
    <a href="{% url 'exp' app.real_name %}" class="btn btn-success">{{app.name}}</
</li>
{% endfor %}
<li>
<a href="{% url 'importar' %}" class="btn btn-success">Importar</a>
</li>
{% endblock %}
```

Ilustración 4. Método {% extends 'Base/index.html'%}.

***Command* (Orden):** Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

En este caso se encapsula en un objeto todo lo que contiene un modelo de la base de datos para exportarlo sin necesidad de conocer lo que tiene dentro el mismo.

```
@login_required
def exportacion(request):
    if request.method=='POST':
        m=request.POST['tabla']
        a=request.POST['app']
        mo=str(m)
        modelo = apps.get_model(a,mo)
        tramest_resource=resources.modelresource_factory(model=modelo)()
        dataset=tramest_resource.export()
        response=HttpResponse(dataset.json,content_type='text/json')
        response['Content-Disposition']='atachment; filename='+mo+".json"''
        key=Fernet.generate_key()
        with open("C:/Users/"+getuser()+"/Downloads/"+mo+".key",'wb') as filekey:
            filekey.write(key)
        return response
```

Ilustración 5. Patrón Command (Orden).

Fuente: Elaboración propia.

Instancia única (*Singleton*): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto.

El patrón *singleton* es un patrón de diseño de creación, que asegura que una clase tenga solo un objeto de un tipo específico y proporciona un punto de acceso global. La intención es:

- Asegúrese de que solo se cree un objeto de la clase.
- Proporcione un punto de acceso para el objeto para que el programa pueda acceder al objeto globalmente.
- Controle el acceso concurrente a los recursos compartidos.

En este caso se accede de forma única a la clase “readfromjson” a través de objeto data.

```
    data=readfromjson("C:/Users/"+getuser()+"/Downloads/"+mod+".json")
    response1=HttpResponse(json2xml.Json2xml(data).to_xml())
    response1['Content-Disposition']='atachment; filename='+mod+".xml"''
    remove("C:/Users/"+getuser()+"/Downloads/"+mod+".json")
    return response1
```

Ilustración 6. Patrón Instancia única (*Singleton*).

Fuente: Elaboración propia.

2.7. Diagramas de clases del diseño.

Un diagrama de clases de diseño muestra la especificación para las clases de software de una aplicación. Incluye clases, asociaciones y atributos, interfaces con sus operaciones y constantes, métodos, navegabilidad y dependencias. El diagrama de clases de diseño muestra definiciones de entidades software más que conceptos del mundo real. El diagrama de clases recoge las clases de objetos y sus asociaciones (39).

La figura muestra el diagrama de clases del diseño de la función exportar de la solución propuesta. Muestra propiedades, métodos y relaciones entre clases. En el anexo 1 se puede apreciar el diagrama de clases del diseño de la función importar.

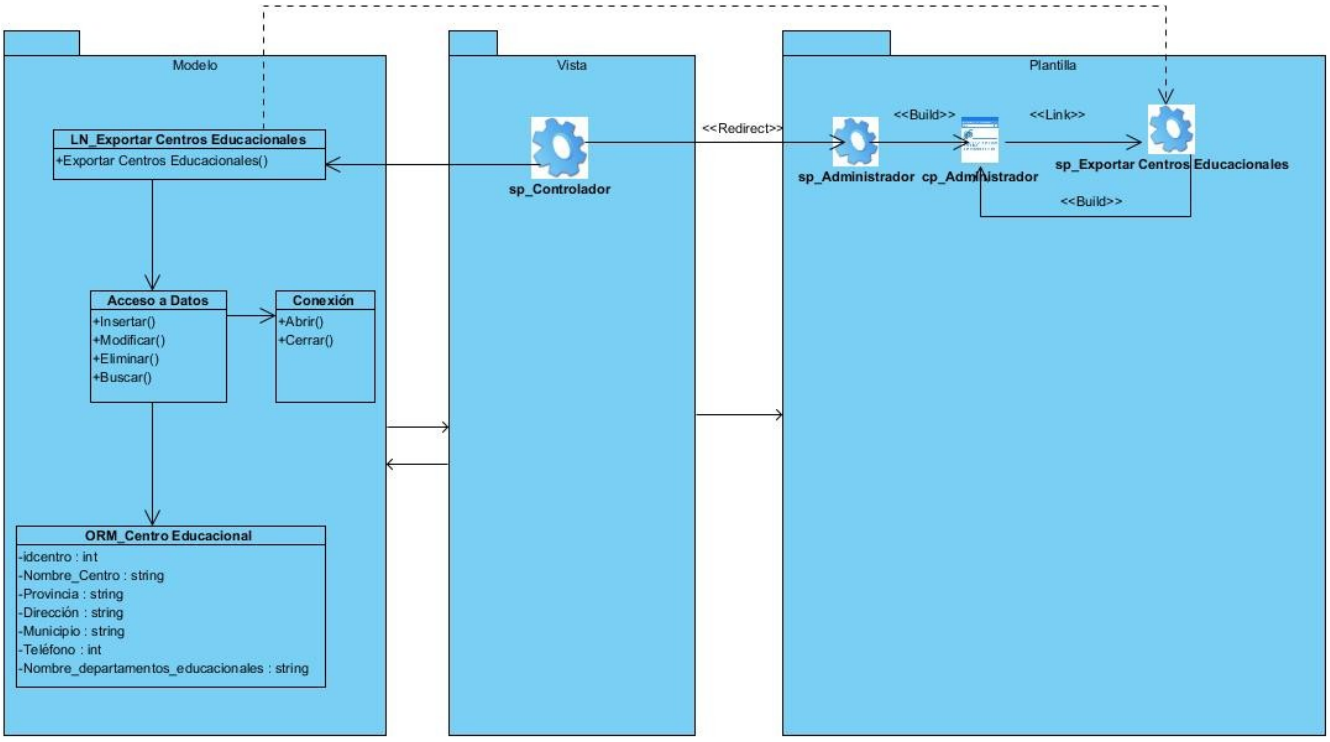


Ilustración 7. Diagrama de clase del diseño de la función “Exportar datos”.

Fuente: Elaboración propia.

2.8 Diagrama de Componente.

Una vez terminado el diseño y análisis de la propuesta de solución se prosigue a diseñar la vista de implementación de la propuesta de solución a través del diseño a nivel de componentes. En esta etapa se ha establecido la estructura general de los datos y del programa del software. El objetivo es traducir el modelo del diseño a software operativo. Pero el nivel de abstracción del modelo de diseño existente es relativamente alto y el del programa operativo es bajo (17).

Los componentes son partes modulares de un sistema independientes entre sí, que pueden reemplazarse con componentes equivalentes. Son autocontenidos y encapsulan estructuras de cualquier grado de complejidad. Los elementos encapsulados solo se comunican con los otros a través de interfaces. Los componentes no solo pueden proporcionar sus propias interfaces, sino que también pueden utilizar las interfaces de otros componentes, por ejemplo, para acceder a sus funciones y servicios. A su vez, las interfaces de un diagrama de componentes documentan las relaciones y dependencias en una arquitectura de software (59).

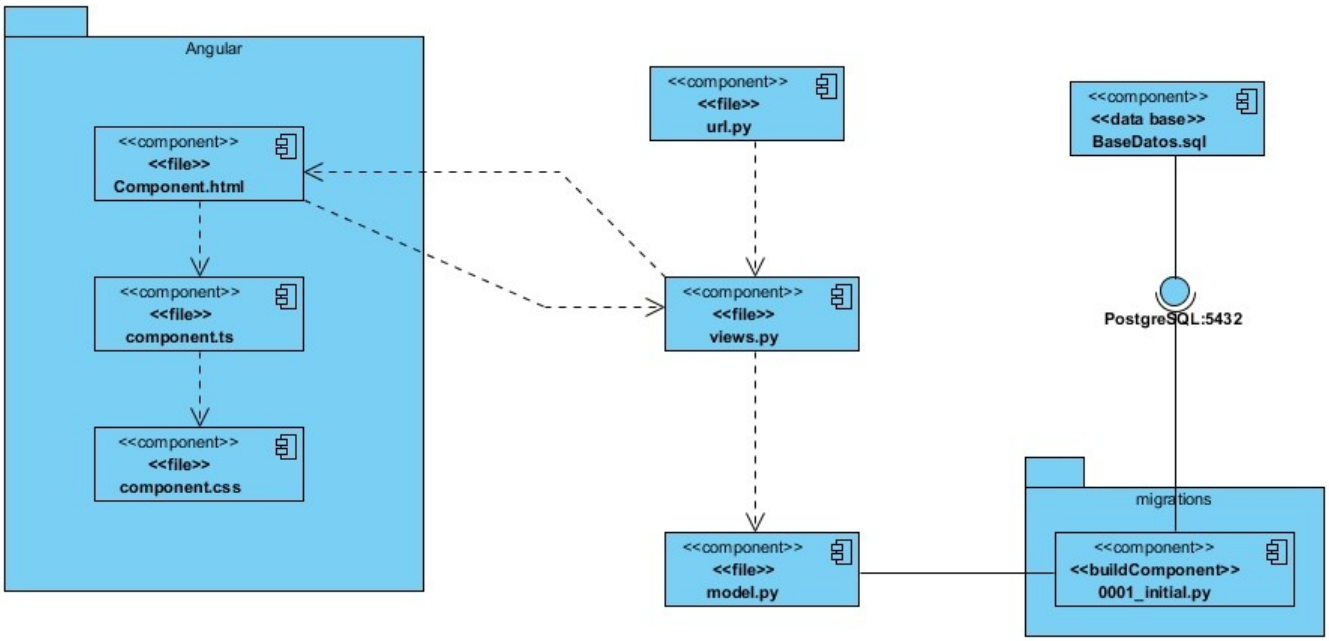


Ilustración 8. *Diagrama de componente*

Fuente*: Elaboración propia.*

2.9 Conclusiones del capítulo

La metodología AUP UCI en su tercer escenario permitió realizar el análisis y diseño del componente, generando así los elementos fundamentales para el desarrollo del Componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop. La definición de los requisitos tanto funcionales como no funcionales permitió el diseño de la propuesta de solución. Por otra parte, los patrones de diseño, garantizaron la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

CAPÍTULO III: VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

En esta sección se describen los elementos que vienen determinados por el proceso de desarrollo utilizado y que dan respuesta a la implementación de la solución. Partiendo del diseño descrito en el apartado anterior, la aplicación comienza a dar paso a la fase de pruebas; segundo, los casos de prueba están diseñados para los requisitos funcionales del componente y los resultados de las pruebas. También se incluyen los resultados de las pruebas e inspecciones realizadas a los componentes para verificar que el sistema cumple con los requisitos especificados.

3.1 Estrategias de Pruebas de Software

Según Pressman una estrategia de pruebas envuelve las actividades de revisión y ejecución de las pruebas de software, cuyo objetivo es descubrir errores en el contenido, función, usabilidad, navegavilidad, rendimiento, capacidad y seguridad de la aplicación web.

El desarrollo del software implica en sí, errores que pueden empezar a ocurrir desde el primer momento del proceso en el que los requerimientos pueden estar especificados de forma errónea, así como en los posteriores pasos del diseño e implementación. Es por esto, que se hace necesario contar con una actividad que garantice la calidad. Las pruebas es la actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados y una evaluación es hecha de algún aspecto del sistema o componente. (40). Las pruebas son un conjunto de actividades para asegurar que el software implemente correctamente una función específica y que se corresponda con los requisitos del cliente (41).

Luego del desarrollo de la propuesta de solución, se hace necesario verificar y validar el sistema implementado a través de una estrategia de pruebas que permita comprobar el cumplimiento de las especificaciones del diseño y de la codificación e identificar los posibles errores cometidos.

Para comprobar que la solución implementada es válida y que cumple con los requisitos acordados con el cliente, se propone la siguiente estrategia de pruebas.

Tabla 13. Estrategia de Pruebas.
Fuente: Elaboración propia.

Niveles de Prueba	Tipo de Pruebas	Método
Pruebas Unitarias	Funcional	Caja Blanca
Pruebas de Integración	Funcional	Caja Negra
Pruebas de Sistema	Funcional (Interfaz)	Caja Negra (Partición de equivalencia)
Pruebas de Aceptación	Prueba de tipo Alfa	Caja Negra

3.2 Tipos de Pruebas

Las pruebas de software son una de las etapas más importantes y más usadas, porque con ellas se puede evitar y reducir los riesgos en el software. Los principales objetivos de las pruebas de software son asegurar que el sistema que se está desarrollando se ajuste a los requisitos del cliente y revelar errores (46). Existen diferentes tipos de pruebas para verificar el correcto funcionamiento del sistema entre ellas se encuentras:

Pruebas Funcionales

Las pruebas funcionales, se centran en los requisitos funcionales del software. Estas le posibilitan al desarrollador obtener un conjunto de datos de entrada del sistema que evalúan todos los requisitos funcionales. Por ello las pruebas funcionales, son pruebas específicas, concretas y exhaustivas para probar y validar que el software hace lo que debe y sobre todo, lo que se ha especificado (47, 48). Este tipo de prueba se aplicó en los niveles de pruebas unitarias, de sistema y de integración.

Pruebas Alfas

La prueba alfa se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado (49). Este tipo de prueba fue aplicada en el nivel de pruebas de aceptación.

3.3 Pruebas Unitarias

Las pruebas unitarias son las pruebas que se realizan a los elementos más pequeños del componente. Son aplicables a los componentes representados en el modelo de implementación para verificar que los flujos de control y de datos estén cubiertos y que funcionen como se espera. Constituyen la primera fase de las pruebas que se aplican a un sistema y su objetivo es verificar que estos componentes estén correctamente codificados (43, 50). Estas pruebas se le aplica al código fuente mediante el método de caja blanca.

Método de Caja Blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Estas comprueban los caminos lógicos del software, proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Requieren del conocimiento de la estructura interna del sistema y son derivadas a partir de las especificaciones internas de diseño o el código (43). La prueba de caja blanca evalúa el código y la estructura interna de un programa. Estas pruebas implican observar la estructura del código para garantizar que las operaciones internas se realizan de acuerdo con la especificación (51). Para realizar el método de caja blanca el lenguaje de programación Python viene con un módulo de pruebas unitarias que proporciona un conjunto de herramientas para crear y ejecutar pruebas unitarias. El marco de pruebas unitarias para pruebas unitarias se inspiró originalmente en *JUnit* y proporciona similitudes con marcos de pruebas unitarias más grandes en otros idiomas. Admite la automatización de pruebas, la configuración cruzada, el código de exclusión de pruebas, la agrupación de pruebas y la independencia de pruebas de la infraestructura de referencia.

```python
def test_structure_Centro_educacional(self):
    content_type = ContentType.objects.get_for_model(Structure)
    permission = Permission.objects.get(
        codename='change_structure',
        content_type=content_type,
    )
    self.user.user_permissions.add(permission)
    self.user.refresh_from_db()

    assign_perm('structure.change_structure', self.user, self.structure)
    self.user.refresh_from_db()

    self.assertTrue(self.user.has_perm('change_structure', self.structure))

    response = self.client.patch(self.url, {
        "address": "update",
    })
    self.assertEquals(status.HTTP_200_OK, response.status_code)
```

Ilustración 9. *Pruebas unitarias de la clase Test_Structure_centro_educacionales.*

Fuente: *Elaboración propia.*

A continuación, las respuestas arrojadas por la llamada al método descrito:

```
jj@jj-HP-255-G7-Notebook-PC: ~/ak-mined-arch-test/server
 Archivo  Editar  Ver  Buscar  Terminal  Ayuda
System check identified some issues:

WARNINGS:
auditlog.LogEntry.additional_data: (django_jsonfield_backport.W001) You are using Django 3.1 or newer, which already has a built-in JSONField.
        HINT: Use django.db.models.JSONField instead.

System check identified 1 issue (0 silenced).
E
======================================================================
ERROR: test_list (unittest.loader._FailedTest)
----------------------------------------------------------------------
ImportError: Failed to import test module: test_list
Traceback (most recent call last):
  File "/usr/lib/python3.8/unittest/loader.py", line 154, in loadTestsFromName
    module = __import__(module_name)
ModuleNotFoundError: No module named 'test.test_list'


----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (errors=1)
(venv) jj@jj-HP-255-G7-Notebook-PC:~/ak-mined-arch-test/server$ python manage.py test test
System check identified some issues:

WARNINGS:
auditlog.LogEntry.additional_data: (django_jsonfield_backport.W001) You are using Django 3.1 or newer, which already has a built-in JSONField.
        HINT: Use django.db.models.JSONField instead.

System check identified 1 issue (0 silenced).
.....................s.....................
----------------------------------------------------------------------
Ran 43 tests in 9.172s

OK (skipped=1)
(venv) jj@jj-HP-255-G7-Notebook-PC:~/ak-mined-arch-test/server$
```

Ilustración 10. Resultado de las pruebas unitarias realizadas a la clase “Test_Structure_centro_educacionales”.

Fuente: *Elaboración propia.*

3.4 Prueba de Integración.

Las pruebas de integración son las pruebas que se realizan para asegurarse de que los componentes en el modelo de implementación funcionen correctamente cuando se combinan para ejecutar una funcionalidad. Estas pruebas descubren errores en las especificaciones de las interfaces de los paquetes. Verifican que las interfaces entre las entidades externas y las aplicaciones funcionan correctamente y que las especificaciones de diseño sean alcanzadas. Es el proceso de combinar y probar múltiples componentes juntos (43). Es una técnica para construir la estructura del sistema mientras que, a su vez, se llevan a cabo pruebas para encontrar errores asociados con la interacción. El objetivo es coger los componentes probados anteriormente y construir una estructura de programa que esté de acuerdo con lo que se modela en el diseño (52, 53).

Este tipo de pruebas tiene dos enfoques: En el enfoque Big Bang se combinan todos los componentes y el programa es probado en todo su conjunto. El resultado puede ser caótico con un gran conjunto de fallos y la consiguiente dificultad para identificar el componente (o componentes) que los provocó. Por el contrario, se puede aplicar la integración incremental en la que el programa se construye y prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir, es más probable que se puedan probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática (54, 55).

Luego de haber aplicado las pruebas unitarias y de probar todas las funcionalidades del sistema por separado y verificar que funcionan correctamente, se procedió a integrar todos los componentes implementados por separado, y una vez integrados, se realizaron pruebas integrales a todo el sistema funcionando como un todo y para ello se utilizó el método de caja negra con partición de equivalencia. Estas pruebas validaron la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop. Las realizaciones de las pruebas unitarias permitieron afirmar que las funcionalidades establecidas a integrar y demás elementos del sistema operaron satisfactoriamente, ajustándose a los requisitos especificados.

3.5 Pruebas de Sistema

Las pruebas de sistema son las pruebas que se realizan cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados. El objetivo de esta prueba es asegurar la apropiada navegación dentro del componente, ingreso de datos, procesamiento y recuperación (43). Se realizan después de la construcción del sistema. Las mismas prueban a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción. En este nivel se prueba el sistema integrado para comprobar el cumplimiento de requisitos especificados (56). Estas pruebas fueron aplicadas mediante el método de caja negra.

Método de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Permiten encontrar funciones incorrectas o ausentes, errores de interfaz, rendimiento, inicialización y terminación, así como errores en estructuras de datos o en accesos a la base de datos externas. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software y son realizadas sin tener conocimiento interno del sistema (43).

Dentro del método de caja negra se hace uso de la técnica Partición Equivalente por ser una de las más efectivas para examinar los valores válidos e inválidos de las entradas existentes en el sistema. La Partición Equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales se derivan casos de prueba (43).
Para la aplicación de esta prueba se seleccionó la **técnica de Partición de Equivalencia** la cual nos permite examinar los valores válidos e inválidos de las entradas existentes en el software.

Tabla 14. **Caso de prueba para la funcionalidad Exportar Fichero.**
Fuente**: Elaboración propia.**

Escenario	Descripción escenario	Centro Educaconal	Formato del fichero	Respuesta del sistema

	nario			
EC 1.1 Exportar Fichero	Se selecciona el centro estudiantil que se desea exportar.	NA	NA	El sistema muestra un mensaje indicando que debe seleccionar los campos
		vacío	vacío	
		V	NA	El sistema muestra un mensaje indicando que debe seleccionar el campos
		Antonio Maceo	vacío	
		V	V	El sistema exporta un fichero en el formato deseado con la información sobre el centro Educacioanl seleccionado.
		Antonio Maceo	JSON	

Como parte de todo el proceso de pruebas, se realizaron tres iteraciones de pruebas hasta no detectarse nuevas no conformidades. En la figura 10 se muestra el resultado de las no conformidades detectadas en cada una de las iteraciones de casos de pruebas.

Durante la aplicación de las pruebas de sistemas se detectaron una serie de no conformidades, las cuales en su mayoría fueron asociadas a errores en las funcionalidades y otras asociadas a errores ortográficos, como, por ejemplo: informaciones de los servicios sin traducir al inglés. En una primera iteración se identidficaron nueve no conformidades, que fueron solucionadas, en una segunda iteración se detectaron tres no conformidades, las cuales fueron corregidas y posteriormente en una tercera iteración no se identificaron no conformidades.

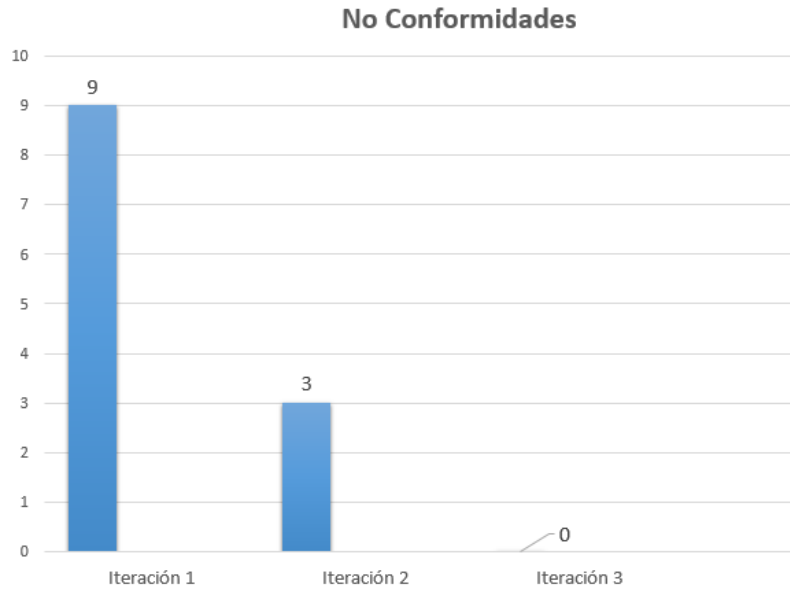


Ilustración 11. No conformidades identificadas por iteraciones de prueba.

Fuente: Elaboración propia.

3.6 Pruebas de Aceptación

Las pruebas de aceptación se hacen con el objetivo de evaluar el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Con su empleo se persigue evaluar la disposición del sistema para su despliegue y posterior uso. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software (46).

Mediante la aplicación del tipo de prueba Alfa, la cual se realizó con varios usuarios donde todos interactuaron con el sistema probando cada una de las funcionalidades, se mostró que el sistema no presentó ningún tipo de error.

3.7 Evaluación de la satisfacción del cliente con el sistema desarrollado.

Para evaluar la satisfacción de los clientes con respecto a la utilidad y aplicabilidad del sistema en entornos reales, el Iadov. Esta técnica permite estudiar el grado de satisfacción del personal implicado en un proceso analizado.

La técnica de Iadov consta de cinco preguntas clave: tres cerradas y dos abiertas, que se reformulan en la investigación para evaluar el grado de satisfacción de los clientes con el sistema de recomendación de servicios propuesto. Una vez establecidas las preguntas, se forma la "tabla lógica

de Iadov" y el número resultante de la interrelación de las tres preguntas indica la posición de los sujetos en la escala de satisfacción. La escala de satisfacción viene dada por los criterios (57, 58):

1. Satisfacción clara.
2. Más satisfechos que insatisfechos.
3. No definido.
4. Más insatisfechos que satisfechos.
5. Insatisfacción clara.
6. Contradictorio

A continuación, se muestra la tabla lógica de Iadov generada a partir de la encuesta a los clientes (véanse los detalles de la encuesta en el Anexo 3).

Tabla 15. Cuadro lógico de Iadov para la investigación.
Fuente: Elaboración propia.

¿Considera que el formato escogido para exportar e importar la información a sus exigencias?	¿Consideraría que este formato de exportar e importar la información es de utilidad?								
	No (N)			No lo sé (NLS)			Sí (S)		
	¿Está satisfecho con el resultado al que se llega producto a la integración del componente?								
	S	NLS	N	S	NLS	N	S	NLS	N
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
No me importa	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

En la tabla 15 se especifican la escala, los índices y los niveles de satisfacción para una mejor comprensión de la técnica.

Tabla 16. Tabla lógica de Iadov.
Fuente: Elaboración propia.

Escala de satisfacción	**Índice individual**	**Nivel de satisfacción**

Clara satisfacción	1	+1,0
Más satisfecho que insatisfecho	2	+0,5
No definida	3	0,0
Más insatisfecho que satisfecho	4	-0,5
Clara insatisfacción	5	-1,0
Contradictoria	6	0,0

A partir de la triangulación de las respuestas de los encuestados en las preguntas cerradas que conforman el cuadro lógico de Iadov, se determina el índice de satisfacción individual. Luego, el índice de satisfacción grupal (ISG) se calcula mediante la siguiente ecuación a partir de los niveles de satisfacción:

$$ISG = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

Donde:

A es el número de encuestados con índice individual 1.

B es el número de encuestados con índice individual 2.

C es el número de encuestados con índice individual 3 o 6.

D es el número de encuestados con índice individual 4.

E es el número de encuestados con índice individual 5.

N es el número total de encuestados.

Al aplicarse las encuestas a 9 personas se obtuvieron los siguientes resultados:

Tabla 17. Resultado de la aplicación de la técnica de Iadov.
Fuente: Elaboración propia.

Nivel de satisfacción	Cantidad	%
Máxima satisfacción	9	90
Más satisfecho que insatisfe-cho	1	10
No definida o contradictoria	0	0,00

Más insatisfecho que satisfecho	0	0,00
Máxima insatisfacción	0	0,00

Al procesar las respuestas a las encuestas en la tabla lógica de Iadov, se obtiene un índice de satisfacción del grupo de 0,90, lo que se traduce en una clara satisfacción con el uso del sistema para recomendar servicios al cliente de acuerdo con sus necesidades. En el criterio de los expertos sobre la utilidad del sistema, hubo un 100,00% de acuerdo con la calificación "Muy adecuado". En cuanto a la aplicabilidad del sistema, hubo un 90,00% de acuerdo con la calificación "Muy adecuado" y el otro 10,00% lo calificó como "Bastante adecuado".

La aplicación de la técnica de Iadov aportó datos significativos sobre el grado de satisfacción de los clientes. Los resultados obtenidos y los criterios emitidos validan la solidez de la propuesta, reflejando una opinión muy positiva respecto a la satisfacción del cliente con la solución propuesta, así como la consideración de que la reutilización de experiencias es útil y aplicable en entornos reales.

3.8 Estándares de Codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

CSS

- Una declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves.
- Para hacer un código CSS legible, ponga una declaración en cada línea.
- Coloque la llave que cierra en una línea nueva.
- Para nombres compuestos de una propiedad utilice el guion (-).

En la figura 9 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```css
caption {
  padding-top: 0.5rem;
  padding-bottom: 0.5rem;
  color: #6c757d;
  text-align: left;
}
```

Ilustración 12. Ejemplo de código en CSS.

Fuente: Elaboración propia.

HTML

- No coloque espacios entre la relación atributos-valor.
- Utilice lowercase para el nombre de los atributos de las etiquetas.

En la figura 10 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```html
<form class="" action="{% url 'Exportacion' %}" method="post">
    {% csrf_token %}
    <h3>Selecciona tabla de la BD</h3>
    <select name="tabla" id="tabla">
        <option name="TramiteTbdTramiteEstudiante" value="TramiteTbdTramiteEstudiante" >TramiteTbdTramiteEst
        <option name="a" value="a">Azul</option>
        <option name="v" value="v">Verde</option>
    </select>

```

Ilustración 13. Ejemplo de código en HTML.

Fuente: Elaboración propia.

Django

- Un espacio a cada lado del operador binario [=, -, +=, ==,>, in,is not, and]
- La declaración de importación debe escribirse en líneas separadas.
- El lenguaje natural usa comillas dobles " . . . "
- ID de máquina Utilice comillas simples ' . . . '

En la figura se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```
24  def exportacion(request):
25      if request.method=='POST':
26          m=request.POST['tabla']
27          a=request.POST['app']
28          mo=str(m)
29          modelo = apps.get_model(a,mo)
30          tramest_resource=resources.modelresource_factory(model=modelo)()
31          dataset=tramest_resource.export()
32          response=HttpResponse(dataset.json,content_type='text/json')
33          response['Content-Disposition']='atachment; filename='+mo+".json"''
34          key=Fernet.generate_key()
35          with open("C:/Users/"+getuser()+"/Downloads/"+mo+".key",'wb') as filekey:
36                  filekey.write(key)
37          return response
38
```

Ilustración 14. Ejemplo de código en Django.

Fuente: Elaboración propia.

3.9 Conclusiones del capítulo

A partir de las fases de implementación y prueba se obtuvo como resultado el componente con las funcionalidades requeridas por el cliente. El uso de estándares de codificación ayudó a mejorar el entendimiento del código a la hora de integrar el componente y propició la utilización de buenas prácticas de programación. Para la validación del software se realizaron las pruebas mediante el método de caja negra, que permitió detectar un grupo de no conformidades en cada iteración, las cuales fueron resueltas en su totalidad, trayendo consigo la realización de mejoras sustanciales en la implementación de funcionalidades e interfaces para así obtener el componente para la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop.

CONCLUSIONES FINALES

El desarrollo de la presente investigación permitió la integración entre el Sistema de Gestión Académica XAUCE AKADEMOS para el MINED y XAUCE AKADEMOS Desktop. De esta manera se puede concluir que:

- El análisis del marco teórico de la investigación a partir de los métodos teóricos y empíricos facilitó conocer el estado de soluciones similares y la integración entre Sistemas, además de permitir una adecuada selección de las tecnologías y herramientas para el desarrollo del componente.

- La definición de los requisitos del sistema en conjunto con el diseño de clases posibilitó la definición de las funcionalidades del componente para cumplir con los objetivos planteados.

- Mediante las pruebas a las que fueron sometidas las funcionalidades del componente una vez desarrollado, permitió validar su correcto funcionamiento y de esta forma verificar que se consumaran los objetivos propuestos para dar solución a la problemática planteada.

- Las opiniones de los desarrolladores y especialistas consultados, permiten afirmar que el componente desarrollado como parte de esta investigación, mejora las prestaciones de la plataforma actual y fortalece el intercambio de información entre esta y AKADEMOS Desktop.

RECOMENDACIONES

Al finalizar el presente Trabajo de Diploma se recomiendan un conjunto de aspectos considerados importantes, para ser tomados en cuenta para su mejora. A continuación, se enuncian:

- Permitir que se pueda exportar e importar la información en distintos formatos.
- Seguir trabajando en el diseño de las interfaces de las funcionalidades implementadas para lograr un mayor equilibrio y presentación de la información.

.

REFERENCIAS BIBLIOGRÁFICAS

1. HERNÁNDEZ, Ingrid Blanco; IBARGÜEN, Víctor Quesada. La Gestión Académica, criterio clave de la calidad de la gestión de las instituciones de educación superior. Recuperado de: http://www. ucv. ve/fileadmin/user_upload/vrac/documentos/Curricular_Documentos/Evento/Ponencias_1/Blanco_y_Quesada. pdf, 2008.

2. VÁZQUEZ, Larisa Enríquez, et al. LCMS y objetos de aprendizaje. 2004.

3. FACUANDO, A. educación a distancia/virtual en Colombia. Recuperado de http://portales. puj. edu. co/didactica/PDF/Tecnologia/EducacionvirtualenColombia. pdf, 2005.

4. KATZ, Raul Luciano. El Papel de las TIC en el Desarrollo. S.l.: Raul Katz. ISBN 978-84-08-09116-5. 2009.

5. Martí, J. (1976). Las fiestas de graduados en Estados Unidos. Escritos sobre educación. La Habana: Editorial de Ciencias Sociales, p. 196

6. CUSTODIO, Angela Ruiz.,. Métodos y técnicas de investigación científica [en línea]. 2022. S.l.: s.n. Disponible en: https://www.gestiopolis.com/metodos-y-tecnicas-de-investigacion-cientifica/

7. MARTÍNEZ TAMAYO, Ana M., et al. Interoperabilidad de sistemas de organización del conocimiento: el estado del arte. Información, cultura y sociedad, ISSN 1851-1740.2011. Disponible en: http://www.scielo.org.ar/scielo.php?script=sci_abstract&pid=S1851-17402011000100002&lng=es&nrm=iso&tlng=es, no 24, p. 15-37.

8. RTA. Red de Transparencia y Acceso a la información. Directrices – Interoperabilidad CPLT [En línea] 2022-06-09. Disponible en: http://mgd.redrta.org/mgd/site/artic/20150123/pags/20150123112234.html

9. MOYA, José Manuel Huidobro; MARTÍNEZ, David Roldán. Seguridad en redes y sistemas informáticos. Thomson Paraninfo, 2005.

10. PLUAS CERCADO, DANIELA FABIOLA. IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO PARA LA GESTIÓN Y ORGANIZACIÓN DE HISTORIAL CLÍNICO DE

PACIENTES EN LA CLÍNICA NUESTRA SEÑORA DE LAS MERCEDES DEL CANTÓN PEDRO CARBO. 2022. Tesis de Licenciatura. Jijpijapa. UNESUM.

11. GARCÍA PLANAS, María Isabel; TABERNA TORRES, Judit. El e-portafolio del estudiante en Mahara-Moodle y Google Sites. En ACTAS CONGRESO INNOVAGOGÍA2014. 2014. p. 1-10.

12. TALAVERA, María del Mar López; RODRÍGUEZ, Elena Real. Un ejemplo de las TIC aplicadas a la educación: La Escuela 2.0 en Castilla-La Mancha. Index. comunicación: Revista científica en el ámbito de la Comunicación Aplicada, 2014, vol. 4, no 1, p. 59-81.

13. DELGADO OLIVERA, Lisdania de la Caridad; DÍAZ ALONSO, Lexys Manuel. Impacto del Módulo de Comunicación Síncrona para la Plataforma Educativa ZERA 2.0. 2021.

14. Vázquez, Msc.Tomás Orlando Junco. UN JUEZ EN LÍNEA AJUSTADO A LAS NECESIDADES DE LA DOCENCIA. La Habana: s.n., 2012.

15. PÉREZ PINO, María Teresa, et al. Programa de entrenamiento en TIC como medio del proceso de enseñanza aprendizaje. Revista Cubana de Ciencias Informáticas, 2015, vol. 9, no 3, p. 138-152.

16. BARZANALLANA, R., 2017. Apuntes. Ingeniería del software. Sistemas Informáticos. Nivel de madurez software. Informática Aplicada. [en línea]. 2017. S.l.: s.n. Disponible en: http://www.u-m.es/docencia/barzana/IAGP/Iagp2.html.

17. PRESSMAN, R., 2021. Ingeniería de Software 9ª Edición. 2021. ISBN 978-1-4562-8772-6.

18. Sánchez, Tamara Rodríguez. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana, Cuba: s.n., 2015. ISSN.

19. Sanchez, Tamara Rodriguez. Metodologia de desarrollo para la Actividad productiva en la UCI. [En línea] 2015. [Citado el: 13 de 08 de 2022.]

20. GUERRERO SÁNCHEZ, Álvaro. Desarrollo de una aplicación web para un gimnasio en Django. 2022. Tesis Doctoral. Universidad Politécnica de Valencia.

21. ALEXANDER, Ramírez Arrenquín J.; BORJA, MC Alejandro Pérez Pasten; ORTIZ, María Beatriz Delgadillo. Desarrollo de punto de venta local (offline) utilizando el potencial de Angular en aplicaciones web.

22. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2022. About PostgreSQL [online] [visitado 2022-06-4]. Disponible desde: https://www.postgresql.org/about/ (vid. págs. 1, 6).

23. EDDELBUETTEL, Dirk. A Brief Introduction to Redis. arXiv preprint arXiv:2203.06559, 2022.

24. YANG, Junqing; CHEN, Lijun; BAI, Jiaqing. Redis automatic performance tuning based on eBPF. En 2022 14th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). IEEE, 2022. p. 671-676.

25. Canonical Ltd. (2020c). Ubuntu 20.04 LTS arrives. Ubuntu. https://ubuntu.com/blog/ubuntu-20-04-lts-arrives.

26. De la Hoz, E., & De la hoz, E. (2019). Linux - Ubuntu Server. In Ubuntu.Com (1era Edición). Corporación Universitaria de la Costa Barranquilla. https://www.ubuntu.com/download/server

27. MEDINA-PÉREZ, Juan A. SISTEMA DE MONITORIZACIÓN DE BAJO COSTE BASADO EN ARDUINO WIFI. 2022.

28. BOOCH, Grady, et al. El lenguaje unificado de modelado. Addison-Wesley, 1999.

29. PARADIGM, Visual. Visual paradigm for uml. Visual Paradigm for UML-UML tool for software application development, 2013, p. 72.

30. FAJARDO-VANEGAS, Pamela Del Rocío; AGUILAR-PAZMIÑO, Carlos Andres. Innovación del modelo de negocio y su impacto en las organizaciones. Revista Científica FIPCAEC (Fomento de la investigación y publicación en Ciencias Administrativas, Económicas y Contables). ISSN: 2588-090X. Polo de Capacitación, Investigación y Publicación (POCAIP), 2022, vol. 7, no 1, p. 35-47.

31. PMOinformatica.com La oficina de proyectos de informática. [En línea] [Citado el: 21 de 09 de 2022.] http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html.

32. Stellman, Andrew; Greene, Jennifer; Applied Software Project Management. ISBN 978-0-596-00948-9. p. 113

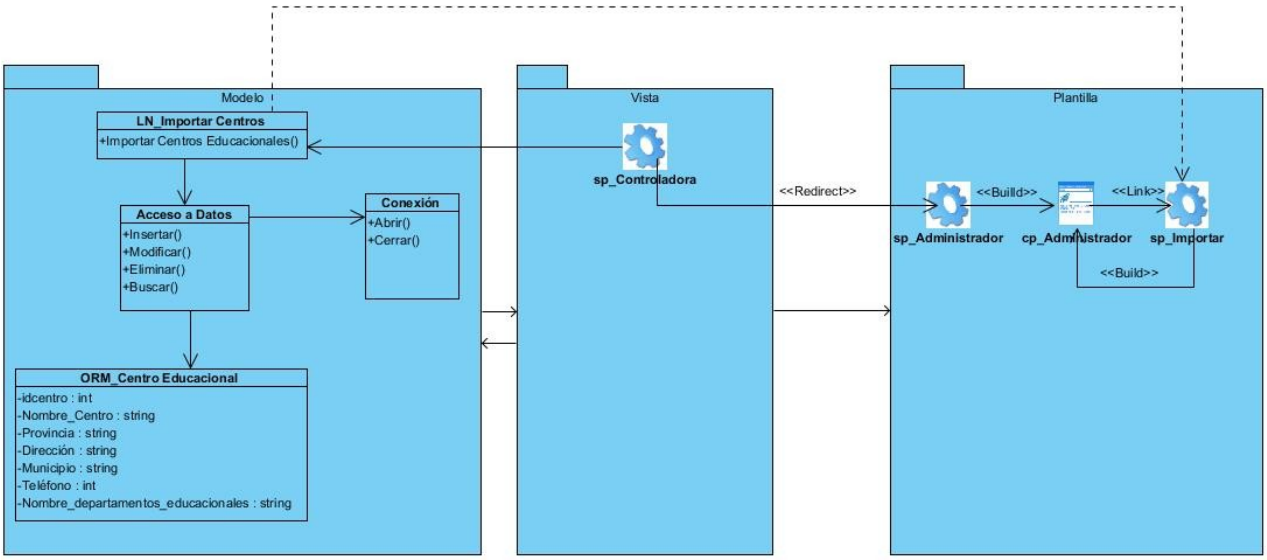
33. QUILAPE PAILLAL, Camila Soledad. Reestructuración y desarrollo del sistema de gestión de becas y trabajos para Latinity. 2022.
34. JIMBO CALVA, Jhunnior Javier, et al. Aplicación web para el seguimiento del ciclo de vida de personas con trastorno del espectro autista. 2019. Tesis de Licenciatura.
35. SOLÓRZANO ÁVILA, Juan Alfredo. Desarrollo de una aplicación web multiplataforma usando el framework Django, para publicitar eventos sociales, aplicado en el municipio del cantón Morona. 2018. Tesis de Licenciatura. Escuela Superior Politécnica de Chimborazo.
36. TEDESCHI, N., 2014. ¿Qué es un Patrón de Diseño? [en línea]. 2014. S.l.: s.n. Disponible en: https://msdn.microsoft.com/es-es/library/bb972240.aspx.
37. ORTEGA, Gilberto Andrés Vargas. Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP. Ciencia e Ingeniería, 2021, vol. 8, no 2, p. e5716304-e5716304.
38. CASTAÑO, Luz Eliana González, et al. Coffee Challenge: Un juego para la enseñanza de patrones de diseño de software. Revista Politécnica, 2021, vol. 17, no 33, p. 36-46.
39. UNAD. (2020, febrero 19). *stadium.unad.edu.co*. Retrieved from stadium.unad.edu.co: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html
40. GONZÁLEZ PINZÓN, Miguel Fernando; GONZÁLEZ SANABRIA, Juan Sebastián. Aplicación del estándar ISO/IEC 9126-3 en el modelo de datos conceptual entidad-relación. Revista Facultad de Ingeniería, 2013, vol. 22, no 35, p. 113-125.
41. CARBALLO MUÑOZ, Lenna; BARRIENTOS NÚÑEZ, Ivette. Las causas del cambio en los requerimientos de software. *Revista Cubana de Ciencias Informáticas*, 2020, vol. 14, no 2, p. 131-144.
42. SERNA, Edgar; MARTÍNEZ, Raquel; TAMAYO, Paula. Una revisión a la realidad de la automatización de las pruebas del software. *Computación y Sistemas*, 2019, vol. 23, no 1, p. 169-183.
43. MARIN DIAZ, Aymara; TRUJILLO CASAÑOLA, Yaimí; BUEDO HIDALGO, Denys. Estrategia de pruebas para organizaciones desarrolladoras de software. Revista Cubana de Ciencias Informáticas, 2020, vol. 14, no 3, p. 83-104.
44. SOMMERVILLE, Ian. Ingeniería del software. Pearson educación, 2005.

45. VENCE Pérez, Jose María. Plan de pruebas de integración. Madrid: s.n., 2010.
46. Castro Alvites, A.G., *Optimización de pruebas de software estructurales usando algoritmos genéticos: revisión sistemática.* 2019.
47. Marén, Y.R., Z.E.M. Tabares, and Y.T.J.S.C.d.l.U.d.l.C.I. Casañola, *Propuesta de automatización de pruebas funcionales durante el ciclo de vida del software en Desoft.* 2019. **12**(9): p. 112-127.
48. Montes Felix, F.D., *Ejecución de pruebas funcionales para certificar el sistema de agendamiento y realización de teleconsultas médicas "SPARTM".* 2022.
49. Pressman, R.S., *Software Engineering: A Practitioner's Approach*. 2010: McGraw-Hill Education.
50. Bedoya Alzate, E., *Implementación de pruebas unitarias.* 2021.
51. Syaikhuddin, M.M., et al., *Conventional software testing using white box method.* 2018: p. 65-72.
52. Sharif, A., D. Marijan, and M. Liaaen. *DeepOrder: Deep learning for test case prioritization in continuous integration testing*. in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2021. IEEE.
53. Medhat, N., et al., *A framework for continuous regression and integration testing in IoT systems based on deep learning and search-based techniques.* 2020. **8**: p. 215716-215726.
54. Blanco, R., et al., *Early integration testing for entity reconciliation in the context of heterogeneous data sources.* 2018. **67**(2): p. 538-556.
55. Brito, M.A., S.R. Souza, and P.S.J.S.Q.J. Souza, *Integration testing for robotic systems.* 2020: p. 1-33.
56. Paredes Flores, D.M., *Proceso de pruebas del sistema del fondo de empleados usando ISTQB–marzo 2018.* 2018.
57. Calle, W.A.C., et al., *Validation of the proof reversal on the inexistence of untimely dismissal by using neutrosophic IADOV technique.* 2019. **33**(1): p. 33-36.
58. Santamaría, D.A., J.W.S. Andachi, and O.F.S. Montoya, *Method for Evaluating the Principle of Interculturality in the Custodial Sentence using the Iadov Technique*. Vol. 37. 2020: Infinite Study.

59. IONOS. 2020. Digital Guide. [En línea] Septiembre 23, 2020. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-componentes/.

ANEXOS

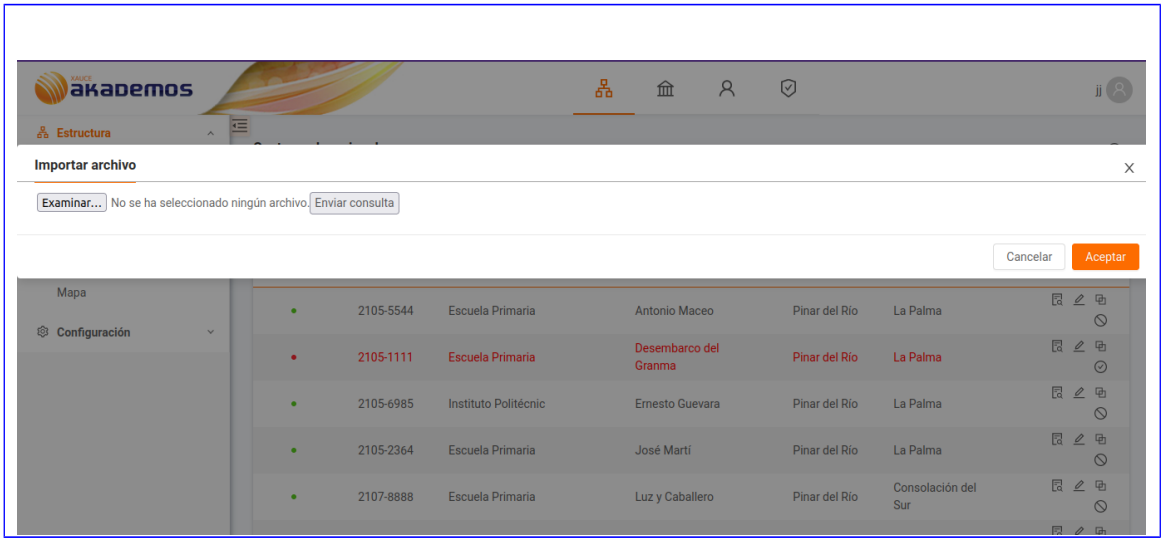
Anexo 1: Diagrama de clase del Diseño Importar Datos.



Fuente: *Elaboración propia.*

Anexo 2: Historia de usuario requisito Funcional Importar Datos.

Número: 2	**Requisito**: Importar fichero
Programador: José Joel Camejo Rodríguez	**Iteración Asignada**: 1
Prioridad: Alta	**Tiempo Estimado**: 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real**: 6 horas
Descripción: Funcionalidad que permitirá importar la informació seleccionada para la base de datos. **Botón**: • Importar: botón que permite importar un fichero en el formato deseado para la base de datos.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	



Fuente: Elaboración propia.

Anexo 3: Encuesta para evaluar la satisfacción del cliente con el sistema

La medición de la satisfacción del cliente con el sistema propuesto, así como su utilidad y aplicabilidad en entornos reales, se basará en su juicio. aplicabilidad en entornos reales, se basará en su juicio. Según su criterio, responda a las siguientes preguntas:

1. ¿Considera que el formato escogido para exportar e importar la información a sus exigencias?
   ___ Sí ___ No ___ No sé

2. ¿Consideraría que este formato de exportar e importar la información es de utilidad?

   ___ Sí ___ No ___ No sé

3. ¿Está satisfecho con el resultado al que se llega producto a la integración del componente?
   ___ Sí ___ No ___ No sé

4. Especifique el grado en el que valora que el componente propuesto es aplicable para la integración entre sistemas, siendo Muy Adecuado la puntuación más alta e Inadecuado la más baja.
   a) Muy adecuado (MA)
   b) Bastante adecuado (BA)
   c) Adecuado (A)
   d) Poco adecuado (PA)
   e) Inadecuado (I)

5. ¿Utilizaría el componente propuesto para la integración entre sistemas, basándose en las experiencias de otros con características similares?

___ Sí ___ No ___ No sé

a. ¿Recomendaría este componente a otros clientes?

___ Sí ___ No ___ No sé