Universidad de las Ciencias Informáticas

Facultad CITEC

Aplicación web para la gestión del alojamiento en la Residencia Estudiantil de la CUJAE

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autor:

Raikol Álvarez León

Tutores:

Dr. C. Modesto Ricardo Gómez Crespo

Ms. C. Dianelys Tejeda Villazón

La Habana, Noviembre de 2022

DECLARACIÓN DE AUTORÍA

Declaro que soy el autor de este trabajo y autorizo a la Facultad CITEC de la Universidad de las Ciencias Informáticas y a la Dirección de la Residencia Estudiantil de la Universidad Tecnológica de La Habana “José Antonio Echeverría” a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los 25 días del mes de noviembre del año 2022.

_____

Firma del autor

Raikol Álvarez León

_____

Firma del tutor

Dr. C. Modesto Ricardo Gómez Crespo

_____

Firma del tutor

Ms. C. Dianelys Tejeda Villazón

AGRADECIMIENTOS

Quiero agradecer a nuestra Revolución y a nuestro Comandante por permitirnos estudiar en esta Universidad, y darme la oportunidad de formarme como profesional. A mis tutores el Dr. C. Modesto Ricardo Gómez Crespo y la M.Sc Dianelys Tejeda Villazón por las acertadas orientaciones para el desarrollo de esta investigación y ayudarme a crecer por los caminos de la ciencia. Al personal de la Residencia que estuvo frente al proyecto y a los muchachos del Laboratorio.

Al tribunal y oponente por su seriedad científica y certeras orientaciones.

A los profesores de todos mis años de carrera por las enseñanzas que me brindaron durante toda mi formación.

A mi madre por ser mi luz, mi fortaleza y mi constancia en la vida, te amo.

A mis abuelos por estar siempre presentes, de principio a fin.

A mis amigos de la UCI por brindarme su más sincera amistad y apoyo todos estos años de experiencias compartidas.

A mis amigos de la CUJAE por acogerme en el último momento y darme las ganas, el cariño y la fuerza para terminar este proyecto.

A Indio por ser alguien que estuvo en los momentos más certeros y al resto de la tropa.

A todos los que siempre estuvieron a mi lado y de alguna forma hicieron esto posible,

Muchas gracias.

DEDICATORIA

La presente tesis se la dedico a mi madre por su amor incondicional, dedicación sin límites, por enseñarme a dar pasos firmes y confiar siempre en mí. A mis abuelos, por formar los valores que hoy poseo y estar junto a mí. A mi tío por interesarse por todos mis logros y dificultades. A mi familia por ser un pilar constante en el trayecto. A mi tío tutor por sus saberes y el tiempo que me dedicó con paciencia y cariño. A mis amigos de la UCI por ser mis hermanos, con los que siempre pude y podré contar. A mis amigos de la CUJAE por los buenos dotasos y momentos que compartimos. A todos los que siempre están a mi lado, gracias.

RESUMEN

En la Educación Superior Cubana se establece como garantía el acceso a los estudios universitarios para todos los estudiantes que por sus resultados docentes logran obtener una carrera de nivel superior, sin importar su lugar de residencia. Sin embargo, en la Universidad Tecnológica de La Habana “José Antonio Echeverría” se hace necesario avanzar en la informatización de la gestión del alojamiento de su residencia estudiantil, para responder a la dinámica del comportamiento de esta actividad y a las restricciones que imponen las condiciones materiales de la infraestructura de sus edificios; de modo que se asegure la veracidad y la actualización constante de la información que se maneja. Debido a las particularidades que presenta esta área de trabajo, no se cuenta con herramientas o aplicaciones informáticas que se ajusten y faciliten el proceso de asignación de capacidades. La presente investigación tiene como objetivo desarrollar una aplicación web para la asignación de capacidades de alojamiento para los estudiantes becados de la Residencia Estudiantil de la Universidad Tecnológica de la Habana “José Antonio Echeverría”, que cumpla con las exigencias del trabajo encargado a la dirección de esta área de la universidad. En la investigación se aborda el análisis de los componentes de la propuesta de solución, y el diseño e implementación de esta mediante una aplicación web. El proceso de desarrollo de la aplicación está guiado por el uso de la metodología Programación Extrema y la utilización de tecnologías de código abierto. El resultado de la investigación fue validado aplicando pruebas de software.

PALABRAS CLAVES: aplicación web, gestión de capacidades, residencia estudiantil

ABSTRACT

In Cuban Higher Education, access to university studies is guaranteed for all students who, according to their academic results, are able to obtain a university career, regardless of their place of residence. However, at Universidad Tecnológica de La Habana “José Antonio Echeverría”, it is necessary to advance in the digitalization process of capacities allocation assigment of its students' residence, in order to respond to the dynamics of the behavior of this activity and the restrictions imposed by the material conditions of the infrastructure of its buildings; ensuring the veracity and constant updating of the information that is handled. Due to the particularities of this work area, there are no tools or computer applications that can be adapted to facilitate the capacity allocation process. The objective of this research is to develop a web application for the allocation of capacities for scholarship students from Universidad Tecnológica de La Habana “José Antonio Echeverría” which meet the work requirements of this area of the university. The research addresses the analysis of the components of the solution proposal, and its design and implementation by means of a web application. The application development process is guided by the phases of the Extreme Programming methodology and the use of open source technologies. The result of the research was validated by applying testing software.

KEYWORDS: capacity management, student residence, web application

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La búsqueda del perfeccionamiento de la Educación Superior y las transformaciones que se producen en la actualidad, apuntan a la necesidad de elevar la calidad de los procesos y servicios universitarios, como garantía de desarrollo y de aseguramiento de la formación integral de los futuros profesionales (Horruitiner-Silva 2006).

Al respecto, en uno de sus postulados, Julio Antonio Mella plantea: “(…) no pueden ser las universidades meras fábricas de títulos (…)”, y en 1964 lo ratifica el Comandante en Jefe Fidel Castro con la siguiente afirmación: "(…) el concepto de Universidad tiene que ampliarse, y tiene que ser algo más que ese sitio donde se reúnen unos a enseñar y otros a aprender".

En los últimos años, la utilización de herramientas como soporte a la gestión ha ido evolucionando progresivamente, de modo que actualmente se puede encontrar una amplia representación de planes estratégicos para el funcionamiento de las universidades [CITATION Abe04 \l 1033][CITATION Ric11 \l 1033]. En [CITATION Lli11 \l 1033] se analiza y reflexiona sobre el papel de estas herramientas a lo largo de su evolución.

Se observa que en el funcionamiento de los procesos intervienen un conjunto de factores que demandan formas de gestión más eficientes y eficaces para la obtención de resultados potenciadores del desarrollo. Lo anteriormente planteado es ratificado por (Lazo Fernández, García González y García Rodríguez 2017) al referirse a la gestión de los procesos universitarios.

En Cuba se aplica la concepción de preparar a los individuos para la vida y en ello el sector de la educación tiene un papel determinante. Por tal razón, se convierte en un reto permanente garantizar la formación integral y contribuir así a la construcción de una sociedad próspera y sostenible.

Para lograr dicho cometido, en las instituciones de la Educación Superior Cubana se desarrolla una amplia actividad de planificación estratégica que alcanza a las residencias estudiantiles por ser uno de los procesos de la gestión universitaria. El sistema de residencias estudiantiles es, por tanto, uno de los elementos que ha caracterizado el desarrollo de las universidades en Cuba, como expresión de la oportunidad de acceso. Este espacio está llamado a convertirse en un importante escenario de actuación, esencialmente educativo, siendo una de las áreas donde los estudiantes becados pasan la mayor parte de su tiempo en la universidad (Lazo Fernández, García González y García Rodríguez 2017), constituyendo entonces un componente básico del enfoque integral por su estrecha vinculación con la docencia y su expresión en la vida. (Gómez Crespo y Tejeda Villazón, 2015). Las residencias son, por tanto, áreas de soporte o apoyo necesarias para el funcionamiento permanente

de la universidad. Contribuyen a cumplir con las demandas de profesionales de la sociedad y a garantizar los estudios a los estudiantes que viven lejos de la universidad.(Ricardo et al., 2018)

Uno de los aspectos principales que ocurren en la actualidad en estos espacios universitarios es la situación material de sus instalaciones debido al deterioro por el paso de los años, lo que reduce la disponibilidad de capacidades de alojamiento y complejiza el proceso de asignación de estas.
A diferencia de otros procesos y servicios de la universidad, en este caso, son insuficientes las herramientas y aplicaciones informáticas establecidas que facilitan la toma de decisiones para la asignación de capacidades, por lo que se hace necesario el desarrollo de tales aplicaciones, desde una concepción modular y escalable que permita una gestión apropiada de este proceso.

Específicamente, la Residencia Estudiantil de la Universidad Tecnológica de La Habana “José Antonio Echeverría”, CUJAE, presta servicio de alojamiento a estudiantes de diferentes provincias y nacionalidades, que de una forma u otra se les otorgó una carrera universitaria. Actualmente existen más de 1 500 estudiantes becados, entre hembras y varones. Para realizar la asignación de capacidades se tienen en cuenta una serie de variables que caracterizan el trabajo y determinan las posibilidades de desarrollarlo de manera rápida, eficiente y efectiva.

En los momentos actuales, el volumen de información a manejar por la dirección del área y la administración de los edificios es muy alto, y se trabaja generalmente de forma manual. Se tabulan los datos y luego se archivan en tarjetas y papeles. Esto viene dado por la insuficiente informatización de las actividades de la Residencia Estudiantil. Las causas que lo determinan son: el bajo nivel de soporte técnico y la ausencia de programas informáticos para perfeccionar la gestión de las actividades.
Los directivos de la Residencia Estudiantil han elaborado herramientas utilizando aplicaciones ofimáticas como el tabulador electrónico Excel, para manejar diferentes datos representados de modo matricial y gráfico, y el procesador de textos Word. De este modo crean ficheros donde almacenan la información necesaria. Cada vez que se requiere completar o verificar los datos de los estudiantes, deben recuperarlos de manera asincrónica, a través de la herramienta SIGENU.
Sin embargo, todo esto no permite la interoperabilidad entre los diferentes roles que participan. Para que todos los involucrados dispongan de la información necesaria, habría que copiar cada fichero en la computadora de cada persona que desempeña cada uno de los roles que intervienen en los procesos, y garantizar que se actualicen los ficheros cada vez que tiene lugar un cambio. Esto no resulta eficiente y puede provocar que los directivos tomen decisiones erróneas, por no contar con la información actualizada.

Por tanto, la gestión de la información de los estudiantes becados puede llegar a ser muy engorrosa y demorada. Es por ello que la situación actual que presenta el proceso de Residencia Estudiantil en la CUJAE imposibilita que exista un buen flujo y manejo de la información asociada con el alojamiento, lo cual trae como consecuencia que la búsqueda de los datos y la actualización de la información no sea la óptima, lo que conlleva a la ineficiencia en el control y la prestación de los servicios.

A partir de la problemática antes descrita se genera la necesidad de resolver el siguiente **problema de investigación**: ¿cómo gestionar la asignación de capacidades de alojamiento en la Residencia Estudiantil de la Universidad Tecnológica de La Habana “José Antonio Echeverría”?

Tomando en cuenta el problema antes expuesto se define como **objeto de estudio:**

Los sistemas informáticos para la gestión del alojamiento en residencias estudiantiles.

Y se identifica como **campo de acción:**

Las aplicaciones web para la gestión del alojamiento en la Universidad Tecnológica de La Habana “José Antonio Echeverría”.

Determinándose como **objetivo general:**

Desarrollar una aplicación web para la gestión del alojamiento en la Residencia Estudiantil de la Universidad Tecnológica de La Habana “José Antonio Echeverría”.

Se desglosan del objetivo general los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación en función de los términos relacionados con la gestión del alojamiento y los sistemas informáticos específicos de trabajo.
- Realizar el diseño de la aplicación web que permita la gestión del alojamiento en la Residencia Estudiantil de la Universidad Tecnológica de La Habana “José Antonio Echeverría”.
- Implementar en un lenguaje de programación las funcionalidades de la aplicación web.
- Realizar pruebas de validación a la aplicación web propuesta, empleando técnicas y métricas de validación de software.

Las **tareas de la investigación** desarrolladas para cumplir con los objetivos planteados, se mencionan a continuación:

- Análisis de los fundamentos de la gestión del alojamiento, las aplicaciones existentes, metodologías, técnicas y herramientas para desarrollar la solución informática propuesta.
- Identificación de los requisitos funcionales y no funcionales para el desarrollo de la aplicación web.
- Diseño de la base de datos.
- Generación y depuración del código fuente que implementa la aplicación web.
- Realización de pruebas de validación a la aplicación web obtenida, empleando métricas y técnicas de validación.
- Realización del informe final de la investigación.

Métodos teóricos:

Analítico – sintético: se utilizó para analizar y caracterizar el problema de investigación en elementos por separado que permitieron profundizar en el estudio de los sistemas de gestión del alojamiento. La síntesis permitió la unión de las partes previamente analizadas, para realizar un razonamiento que parte de lo general a lo particular, obteniendo los elementos más importantes y comunes de cada bibliografía analizada.

Inductivo – deductivo: propició un razonamiento que parte de lo particular a lo general, reflejando en el resultado final lo común que existe en cada bibliografía estudiada.

Análisis histórico – lógico: permitió la realización de un estudio sobre el desarrollo de los aspectos fundamentales del proceso de gestión de residencias y las herramientas que se han empleado dentro del ámbito nacional e internacional.

Modelación: se empleó para representar el modelo conceptual y el modelo de la base de datos, así como la representación de la información a través del uso de diagramas y esquemas.

Métodos empíricos:

- **Entrevista:** se les realizó a profesionales del área de la Residencia Estudiantil, así como a especialistas en gestión del alojamiento, con el objetivo de obtener información verídica de cómo se lleva a cabo dicha tarea en algunos centros del país.

- **Medición:** se aplicó mediante el empleo de instrumentos de medición de la calidad de los requisitos que permitieran estimar y validar la calidad del diseño informático realizado.

- **Observación:** se aplicó para obtener información sobre los diversos sistemas de gestión del alojamiento enfocados a la residencia estudiantil que existen a nivel nacional e internacional,

con la intención de realizar la propuesta de interfaz, reportes de salida y demás componentes de la aplicación.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

- En el capítulo 1, titulado: Análisis de los elementos que conforman el marco de la investigación, se aborda el marco teórico referencial de la gestión en las universidades y en las residencias estudiantiles, sintetizando los conceptos relacionados con la problemática y el campo de acción antes mencionados. Se describe y caracteriza la metodología utilizada para guiar todo el proceso de desarrollo del software, las herramientas actuales a utilizar para el desarrollo de la aplicación: el Entorno de Desarrollo Integrado (IDE), el sistema gestor de base de datos y la herramienta de Ingeniería de Software Asistida por Computadora (CASE).

- El capítulo 2, titulado: Análisis y diseño de la propuesta de solución, realiza un análisis del estado actual y la caracterización del proceso de gestión del alojamiento en la Residencia Estudiantil de la CUJAE. Además, se exponen los principales elementos que modelan y describen la aplicación web propuesta, las fases de análisis y diseño del proceso de desarrollo de estos sistemas vinculadas con la metodología de diseño de software seleccionada, y la descripción de la base de datos diseñada para la gestión de la información vinculada.

- El capítulo 3, titulado: Implementación y pruebas de software, resume los resultados obtenidos en las fases de implementación y pruebas de la aplicación web desarrollada, a partir de dar cumplimiento a las tareas planteadas. Se plantean los estándares de codificación que se emplearon, así como las técnicas de validación de los requisitos del sistema. El capítulo dedica sus últimas secciones a presentar las pruebas realizadas y sus resultados.

Finalmente, en el documento se presentan las conclusiones a las que se arriba en este trabajo y las recomendaciones que se proponen para darle continuidad al mismo.

CAPÍTULO 1: ANÁLISIS DE LOS ELEMENTOS QUE CONFORMAN EL MARCO DE LA INVESTIGACIÓN

El presente capítulo describe los conceptos relacionados con los sistemas de gestión y la gestión de alojamiento. Se analizan varios de los sistemas que se encuentran en aplicación en otras instituciones y que pudieran constituir una posible variante de solución, justificándose su no conveniencia. Se realiza un estudio de las metodologías de desarrollo más usadas, el lenguaje de modelado a emplear y las herramientas informáticas a utilizar. Se plantea la propuesta de solución y finalmente se fundamentan las valoraciones parciales a las que se arriba.

1.1 Conceptos relacionados con la gestión de alojamiento

A continuación, se relacionan los principales conceptos analizados y estudiados en el presente trabajo.

- **Gestión:** Se registra en el latín en las formas *gestío*, *gestiōnis*. Se lo asocia a gesto, identificado en el latín gestus, en este contexto vinculado a lo realizado y no como una simple expresión emocional, y con raíz en el verbo gerere, sobre la idea de hacer o emprender algo. También significa conjunto de operaciones que se realizan para administrar y dirigir un negocio, acción o trámite que se ejecuta para conseguir o resolver algo (Veschi 2022). El autor considera que éste es el significado que más se corresponde con el estudio y el objetivo planteado en este trabajo.
- **Gestión de Alojamiento:** En el marco de este trabajo será considerada la gestión de alojamiento como la acción o proceso de dar hospedaje a determinadas personas por determinadas razones. Existen diferentes tipos de instalaciones dedicadas a dar alojamiento. Cada una de estas instalaciones cuenta con sus criterios de alojamiento y realiza esta gestión de forma diferente. Dentro de las principales tareas que se llevan a cabo en este proceso se encuentran: llevar el control de todo el personal que se tiene alojado, de las capacidades con que cuenta la instalación, el tiempo de estancia de cada persona, entre otras que varían en dependencia del tipo de instalación.
- **Sistemas de gestión de alojamiento:** Los sistemas dedicados a la gestión de alojamiento son todas aquellas aplicaciones informáticas que informatizan el proceso de alojar a una persona o conjunto de personas en una institución dedicada a este fin. Es la forma de tener informatizado todo lo que se hace de forma manual y suele tratarse de un problema complejo en muchos de los casos. Estos sistemas brindan determinadas funciones que favorecen todo

el trabajo.

1.1.1 Aplicaciones web existentes relacionadas con el objeto de estudio

Como punto de partida, a partir del objeto y campo de estudio planteado, se especifican los conceptos de sistemas informáticos y aplicaciones web que, en opinión del autor, más se corresponden con el contexto de solución al problema abordado.

Se puede entender como sistema informático aquel sistema que está compuesto de hardware, de software, la información, y que mantiene una estrecha interacción con los usuarios. Puede considerarse también como aquella agrupación de personas, recursos, procesos y también equipos e instrumentos prediseñados, creados para colecciones, registros, procesos, almacenamientos, recuperación e identificación de la información (HellHacker 1999).

Dentro de los sistemas informáticos se encuentran las aplicaciones web. Una aplicación web es un programa que se almacena en un servidor remoto y que se accede a él a través de Internet, mediante una interfaz de navegador. Los datos o los archivos sobre los que se trabaja son procesados y almacenados dentro de la web. Estas aplicaciones por lo general no necesitan ser instaladas en la computadora. Pueden contener elementos que permiten la comunicación entre el usuario y la información con la que se trabaja, además del acceso a datos de modo interactivo a través de eventos (Molina Ríos 2017).

En la revisión realizada, se encontraron diversos sistemas de hospedaje que poseen características entre propias y semejantes. Los aspectos semejantes que presentan se manifiestan en los procesos que implementan, los cuales, al estar dirigidos hacia la gestión de la residencia, son similares en funciones como pueden ser el tratamiento organizado de datos sobre las personas, la información que brindan a los diferentes usuarios y la realización de reportes para monitorear la información.

Ejemplo de esto es *WebRezPro*, un sistema de administración de propiedades de software como servicio (SAAS), creado por *World Web Technologies Inc*., una empresa de desarrollo de software de Internet que se especializa en las industrias del turismo y la hotelería. Fundada en 1994, *World Web Technologies* es una empresa privada con sede en Calgary, Alberta (WebRezPro 2022).

Existen también otros sistemas foráneos de similar alcance en sus tareas, pero con servicios definidos para diferentes ámbitos. Tal es el caso del sistema informático orientado a la gestión de la residencia estudiantil *GreenStudent***,** el cual gestiona con eficacia los Colegios Mayores Universitarios (CMU's) y residencias de estudiantes privadas o públicas, laicas o religiosas, permanentes o de temporada (GreenStudent 2022).

A nivel nacional, en varias universidades también se han desarrollado soluciones. Tal es el caso del Sistema Automatizado Informativo para la Comunidad de la Residencia Universitaria de la

Universidad de Pinar del Río. El software fue desarrollado con uso del gestor de base de datos Microsoft Access y de su lenguaje residente Visual Basic para Aplicaciones. Debido a la forma en que la solución está implementada, se puede afirmar que no es un sistema distribuido.

En la Universidad de las Ciencias Informáticas (UCI) se encuentra el Módulo Sistema de Gestión Universitaria XAUCE SGU. Este es un módulo de fácil configuración que forma parte del Sistema de Gestión de Residencia Akademos y está integrado al Sistema de Gestión Universitaria. Dentro de sus funcionalidades están:

- Filtrar: Permite mostrar un listado de estudiantes a partir de los parámetros especificados en la aplicación, permitiendo la búsqueda rápida y efectiva de información.
- Gestionar: Se realiza la gestión de cada uno de los estudiantes registrados en sistema, permitiendo la actualización de su situación en la beca estudiantil.
- Asignación: Se asigna a cada uno de los estudiantes su ubicación en la beca estudiantil, a partir de las capacidades disponibles en el campus universitario.

Este sistema permite gestionar de forma detallada el proceso de hospedaje que se lleva a cabo en la Residencia Estudiantil, supervisándose cada área de residencia por su respectiva facultad. En opinión del autor, estas funcionalidades son requeridas y deben garantizarse en todo sistema de gestión de alojamiento, y por tanto están incluidas en la propuesta de solución que se presenta.

Por las consultas realizadas, se puede decir que los procesos de las residencias guardan una estrecha relación entre sí y a su vez están vinculados con otros procesos que se desarrollan en la universidad. Particularmente, dado que un propósito clave en las universidades cubanas es lograr la formación integral de sus egresados, no basta con que el estudiante cumpla con sus deberes en la residencia, sino que debe tener adecuados resultados académicos y viceversa.

Como consecuencia, tanto los directivos de las facultades, como los directivos de la residencia, deben disponer de información actualizada de los estudiantes becados, de su ubicación en la beca, su situación académica, su estado de salud y del cumplimiento de sus deberes en la residencia, entre otros aspectos.

Además, dado que son varios los roles que participan en estos procesos y que se necesita tomar en cuenta disímiles datos, algunos de los cuales se manejan en sistemas ya existentes en las universidades, como es el caso de SIGENU (herramienta aprobada por el MES para gestionar los datos personales y la situación académica de los estudiantes), es necesario disponer de herramientas informáticas que apoyen la gestión de estos procesos y sean capaces de interactuar con los sistemas existentes.

Adicionalmente, la propia residencia contempla el seguimiento a los servicios de limpieza y custodia de los edificios, la asignación de deberes a los estudiantes, y el proceso de evaluación integral del

becado. Por todo lo anterior, se considera que las aplicaciones estudiadas, si bien cumplen con los objetivos para los que han sido implementadas, no poseen las funcionalidades específicas que demanda la CUJAE, por lo que el autor considera que no es conveniente su utilización para gestionar integralmente los componentes básicos del servicio en la universidad, y su crecimiento.

1.1.2 Valoración sobre los sistemas de alojamiento estudiados

Los sistemas de gestión mencionados anteriormente presentan una serie de características similares, como son: los métodos de gestión del alojamiento, la facilidad que presentan para llevar a cabo tareas de búsqueda de información sobre los recursos que administran, la facilidad de generar reportes de estado, obteniendo un mayor control en cuanto a la gestión de los mismos. Estas características fueron tomadas en cuenta como base para la creación de la aplicación de gestión del alojamiento que se implementó en el presente trabajo de tesis.

No obstante estas facilidades, se valora que los sistemas estudiados en ambos casos, tanto los sistemas foráneos, como los nacionales, en lo específico no cumplen con las condiciones de trabajo propias planteadas por la CUJAE, debido a que la organización o forma de operar de estos sistemas es diferente, los directivos que participan en los procesos de otorgamiento del alojamiento no tienen las mismas responsabilidades y en algunos casos difieren a los de la CUJAE.

Además, la reutilización de un sistema ya existente en el mercado implica el estudio del mismo para comprobar su factibilidad en las tareas de la institución o el área al que sea destinado. Cuando los procesos que se realizan no están estandarizados, se incurre en el esfuerzo de adaptar un producto con características predeterminadas a las propias de la universidad, en este caso las de la CUJAE. Se invierte tiempo en la preparación de desarrolladores para la implementación de las nuevas funcionalidades, para el despliegue y testeo del sistema.

Adicionalmente, la propia residencia de la CUJAE controla otros aspectos, como las actividades de limpieza y custodia de los edificios. Por lo tal motivo, estas aplicaciones no pueden ser utilizadas para dar solución al problema que se aborda en este trabajo, pues, entre otros aspectos, no permiten la asignación de deberes a los estudiantes, ni completan un proceso de evaluación integral del becado, atendiendo a las particularidades de la Residencia Estudiantil de la CUJAE.

A continuación, se muestra en la Tabla 1.1 los resultados de la comparación entre los sistemas antes expuestos:

Tabla 1. 1 Comparación entre los sistemas estudiados. Fuente: Elaboración propia

Sistemas	Software Libre	Código abierto	Funcionalidades del trabajo educativo	Evaluación	Gestión de incidencias
WebRezPro	No	No	No	No	No
GreenStudent	No	No	No	No	No

XAUCE XGU	No	Sí	No	No	No

Teniendo en cuenta los aspectos explicados anteriormente, se concluye que resulta necesario desarrollar una aplicación web como solución y respuesta al objetivo planteado, que permita el acceso de forma distribuida. Debe además facilitarle a los diferentes roles que intervienen en los procesos de la Residencia Estudiantil (tanto becados, como directivos a nivel de la universidad, facultad y de residencia), obtener información veraz y actualizada de los becados, interoperando con los sistemas existentes en la universidad, que constituyen fuentes de datos aprobadas.

1.2 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas, procedimientos y soporte a la documentación para el desarrollo de softwares, las cuales indican cómo debe realizarse una actividad determinada paso a paso, para informatizar un proceso dado a partir de la metodología en sí. Señala qué personas participan en el desarrollo de estas actividades y los roles que desempeñan en la realización de sus tareas.
Hacen uso de diagramas, modelos o representaciones gráficas, con el empleo de procedimientos detallados, además de describir la información que se debe producir como resultado de una actividad y la información necesaria para implementarla.

En la actualidad, existen diversas metodologías, pero no todas se integran a las necesidades requeridas para un buen desarrollo de un software. En el desarrollo de la presente investigación se realizó un estudio exploratorio para la selección de la metodología a emplear como guía y descripción de las especificaciones de la ingeniería del software para la implementación de la aplicación, asegurando se adecuara a las especificidades de este proyecto, minimizando los riesgos que pudieran surgir, para así obtener los resultados esperados con el menor costo y esfuerzo posible.

En estos últimos años se han desarrollado dos corrientes en lo referente a las metodologías de desarrollo, las llamadas Metodologías Tradicionales o Métodos Pesados y las Metodologías Ágiles o los Métodos Ligeros. Mientras los Métodos Pesados intentan conseguir el objetivo común, por medio de orden y documentación, los Métodos Ligeros tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso (Molpeceres 2002).

Se determinó que, de modo general, cuando el proyecto en el que se está trabajando es de una magnitud considerable, se hace uso comúnmente de las metodologías tradicionales. Como regularidad, se caracterizan por implementarse en grandes grupos de trabajo, lo cual conlleva a la existencia de una mayor cantidad de roles, además de la realización por parte del equipo de

desarrollo de una documentación exhaustiva de todo el desarrollo del software y del cumplimiento de la planificación en las primeras fases del proyecto.

Por otro lado, las metodologías ágiles se distinguen por estar orientadas al cambio en el transcurso del proyecto. Los grupos de trabajo que implementan este tipo de metodología son reducidos, lo que implica menor cantidad de roles, y poseen bajos niveles de formalización de la documentación requerida.

Entre las metodologías de desarrollo de software más utilizadas, que fueron objeto de estudio, se encuentran las mencionadas a continuación:

- **Programación Extrema (*Extreme Programing, XP*):**

  Se clasifica como Metodología Ágil y es una de las más exitosas y popular en la actualidad. Generalmente es utilizada en proyectos de corto plazo, con equipos de desarrollo pequeños y cuyo plazo de entrega es en tiempo record. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto (Asmo 2018).

  Tiene como características principales las pruebas unitarias, la re-fabricación y la programación en pares. La primera se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantan algo hacia el futuro, haciendo uso de pruebas de las fallas que pudieran ocurrir. Se puede concebir como una forma de adelantarse a obtener los posibles errores. La segunda implementa la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio. La programación en pares es una particularidad de esta metodología la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento (Peño 2015).

  Lo fundamental de esta metodología recae en la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales («Software Engineering | Extreme Programming (XP) - GeeksforGeeks» 2022). Los manejos de los cambios se convierten en parte sustantiva del proceso. Se suele comenzar en menor escala y se van añadiendo las funcionalidades que sean necesarias según se avanza en el

proyecto, nunca antes. El costo del cambio no depende de la etapa en que se encuentre el desarrollo del software.

Los roles más comunes son el de programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador, consultor y gestor. El primero escribe las pruebas unitarias y produce el código del sistema. El cliente se encarga de las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

El rol de encargado de seguimiento proporciona retroalimentación al equipo, verificando el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado para mejorar futuras estimaciones. El entrenador es responsable del proceso global, el cual debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente (Al-Saqqa, Sawalha y Abdel-Nabi 2020). Los consultores son miembros externos del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas. El gestor es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas y su labor esencial es de coordinación (Letelier Torres y Hilario Canós 2003).

Esta metodología consta de 4 fases:

- **Planificación:** consiste en realizar una recopilación de todos los requerimientos del proyecto, se crean las Historias de Usuario (HU), el plan de iteraciones y el plan de entregas.

- **Diseño:** se basa en conseguir diseños simples y sencillos, con el objetivo de procurar hacerlo todo lo menos complicado posible para el cliente o usuario, se crean las tarjetas CRC, las cuales definen una clase expresando sus funcionalidades y las otras clases con las que colabora.

- **Desarrollo:** consiste en establecer una buena comunicación entre el equipo y el cliente, para que los desarrolladores puedan codificar todo lo necesario para el proyecto que se

requiere. Además, se definen las tareas de ingeniería para que los desarrolladores tengan una guía para implementar todas las HU.

- **Pruebas:** consiste en comprobar el funcionamiento de la codificación que se haya implementado, garantizando la evaluación de las distintas tareas en las que han sido divididas las HU.

Los pasos del ciclo de desarrollo de XP a grandes rasgos son:

- El cliente define el valor de negocio a implementar.
- El programador estima el esfuerzo necesario para su implementación.
- El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
- El programador construye ese valor de negocio.
- Regresar al paso 1.

- **Marco de Soluciones Microsoft *(Microsoft Solution Framework, MSF*):**

  Es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Consiste en una metodología flexible e interrelacionada, con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo, la gestión de proyectos tecnológicos y se centra en los modelos de proceso y de equipo, dejando en un segundo plano las elecciones tecnológicas. MSF reconoce que la tecnología no es la única pieza clave para implantar soluciones exitosas (Azmat y Kummer 2020). Para lograr el éxito, es necesario incorporar estrategias de manejo de riesgos y administración de recursos, así como una definición clara de roles y responsabilidades del equipo (Lizbeth Carrillo 2018).

  La metodología MSF presenta características como la adaptabilidad, la escalabilidad, la flexibilidad y la tecnología agnóstica. La primera es parecida a un compás que puede ser usado en cualquier parte de un mapa, de modo que su uso logre adaptarse y limitarse al lugar específico en cuestión. La segunda puede organizar equipos tanto pequeños, entre 3 ó 4 personas, como también proyectos que requieren 50 personas o más. La flexibilidad es la capacidad de poder ser utilizada en el ambiente de desarrollo de cualquier cliente, y la última puede ser usada para desarrollar soluciones basadas en cualquier tecnología (Peño 2015).

  Los equipos organizados bajo MSF son pequeños y multidisciplinarios. En ellos los miembros

comparten responsabilidades y balancean las destrezas para mantenerse enfocados en el proyecto que están desarrollando. Comparten una visión común del proyecto y se enfocan en implementar la solución, con altos estándares de calidad y deseos de aprender (Zykov, Alexander y Kazantsev 2018).

- **Scrum**

  Es una metodología ágil y flexible que se utiliza para gestionar el desarrollo de software, realizando entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor. Por ello, está especialmente indicada para proyectos en entornos complejos, donde se necesita obtener resultados pronto, los requisitos son cambiantes o poco definidos, y la innovación, la competitividad y la productividad son fundamentales (Xavier Albaladejo 2022). Además, cuenta con fases tales como: conceptualización, negocio, requerimientos, diseño e implementación y prueba.

- **SXP**

  Es un híbrido cubano de las metodologías ágiles XP y Scrum, desarrollado en el 2008 por UNICORNIOS: un grupo de proyectos de investigación y desarrollo que formó parte del polo productivo de Software Libre (SWL), de la Universidad de las Ciencias Informáticas (UCI) [CITATION Ley09 \l 3082]. SXP está especialmente indicada para proyectos de pequeños equipos de trabajo y requisitos imprecisos, donde existe un alto riesgo técnico. Consta de 4 fases principales: Planificación-Definición, Desarrollo, Entrega y Mantenimiento, cada una desglosada en flujos y actividades que generan artefactos (Romero 2008).

- **AUP**

  Es una versión simplificada del Proceso Unificado de Rational (RUP). Esta describe, de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando métodos ágiles y conceptos que aún se mantienen válidos en RUP. Esta metodología se preocupa especialmente de la gestión de riesgos, planteando que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la fase de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos claves del software, los cuales determinan los riesgos técnicos. Consta de 4 fases: Concepción, Elaboración, Construcción y Transición (AUP 2015).

- **Proceso Unificado de Desarrollo (RUP).**

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo («Rational Unified Process (RUP) - apppm» 2022).

Los flujos de trabajo en RUP se basan en el modelado del negocio, los requerimientos, el análisis y diseño, la implementación, las pruebas, la administración del proyecto, administración de configuración y cambios, y el ambiente (Tia 2019). El primero describe los procesos de negocios, identificando quiénes participan y las actividades que requieren automatización. Los requerimientos definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

El análisis y diseño describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas, e indica con precisión lo que se debe programar. La implementación define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, y la estructura de capas de la aplicación.

La etapa de prueba busca los defectos a lo largo del ciclo de vida.

Instalación: Produce el lanzamiento del producto y realiza actividades para entregar el software a los usuarios finales.

La administración del proyecto involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes. Por otra parte, la administración de configuración y cambios describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos y control de versiones. El ambiente contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización (MANCHEGO PEÑA 2019).

RUP define 4 fases fundamentales: inicio, elaboración, construcción y transición (Siregar, Irmayani y Dar 2021). El primero describe el negocio y delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema. En otras palabras, el objetivo en esta etapa es determinar la visión del proyecto, su puesta en marcha.

El segundo define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales)

identificados de acuerdo al alcance definido.

La fase de elaboración obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios lanzamientos del producto que han pasado las pruebas. Se ponen estos lanzamientos a consideración de un subconjunto de usuarios.

La fase de transición indica que ya está listo para su instalación en las condiciones reales. Puede implicar en la mayoría de los casos la reparación de errores.

Las características fundamentales de RUP son:

- Guiado y manejado por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Los grupos de roles en esta metodología son el analista, desarrolladores, administradores, soporte y roles adicionales.

1.2.1 Metodología de desarrollo de software utilizada

Para decidir qué metodología emplear, se deben tener en cuenta las necesidades del negocio y del equipo. La propuesta de solución a desarrollar debe responder a características que apuntan a que el proceso de desarrollo se ajuste a las particularidades siguientes:

- El período de tiempo para el desarrollo de la aplicación es corto o bien ajustado.
- A la aplicación se le incorporan cambios o funcionalidades en cualquier etapa de su desarrollo, tanto como entiendan el desarrollador o el cliente, por lo que el cliente es miembro clave del equipo.
- El equipo de desarrollo del sistema cuenta con menos de diez integrantes.
- Según el proyecto avance, pueden surgir recomendaciones o ideas las cuales deban ser incorporadas, lo que obliga a tener un margen de adaptabilidad en cuanto al desarrollo del producto.

Teniendo en cuenta estas características, se requiere entonces de un proceso de desarrollo ágil.

A partir de las definiciones de las metodologías ágiles mencionadas en el epígrafe anterior y tomando en cuenta los elementos expuestos sobre las metodologías ágiles analizadas, el autor de la presente investigación decide emplear Programación Extrema (XP) como metodología de desarrollo de software, pues se adecua a los requerimientos exigidos, dado que se trata de un proyecto de corta duración, su equipo de desarrollo es pequeño y el cliente forma parte del mismo, lo que contribuye a fomentar las relaciones interpersonales y el buen clima de trabajo.

Se ha tenido en cuenta que XP define la constante comunicación y la retroalimentación, teniendo

como uno de sus fines fundamentales la construcción de un producto que vaya en línea con los requerimientos del cliente y las necesidades del proyecto. La aplicación será desarrollada por un programador, no estrictamente coincidiendo con la característica de programación en parejas que propone la misma, pero basado en el trabajo entre el cliente y el programador.

1.3 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*), es un lenguaje de propósito general, que pueden usar todos los modeladores. Es uno de los más conocidos y utilizado en la actualidad. UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (Jacobson, Booch y Rumbaugh 2000).

UML incluye aspectos conceptuales como son los procesos de negocios y funciones del sistema, además de aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Perez-Castillo 2022). Puede utilizarse para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware y organizaciones del mundo real. Dentro de sus rasgos distintivos, ofrece diferentes tipos de diagramas. Algunos de estos diagramas son:

- Diagrama de clases: Tipos de diagrama UML que muestran las clases de objetos en un sistema y sus relaciones.
- Diagrama de estado: Tipo de diagrama UML que muestra los estados de un sistema y los eventos que disparan una transición de un estado a otro.
- Diagrama de casos de uso: diagrama que documenta los casos de usos, los cuales son una técnica para el descubrimiento de requisitos del sistema.
- Diagrama de secuencia: Diagrama que muestra la secuencia de las interacciones requeridas para completar cierta operación. En UML, los diagramas de secuencia pueden asociarse con casos de uso.

1.4 Herramientas y tecnologías

En la actualidad existen diversas aplicaciones informáticas que constituyen herramientas y tecnologías cuyo propósito es aumentar la productividad y celeridad en el desarrollo y mantenimiento del software, reduciendo el coste de los mismos en términos de tiempo y dinero. A continuación se relacionan las empleadas en este trabajo y las características que favorecieron su elección.

- **PyCharm** como herramienta principal para el trabajo con el lenguaje de programación Python del lado del servidor. Proporciona finalización de código inteligente, inspecciones de código, resaltado de errores sobre la marcha y soluciones rápidas, junto con refactorizaciones de

código automatizadas y capacidades de navegación enriquecidas. (JetBrains: Essential tools for software developers and teams 2022)

- **Visual Studio Code** para el trabajo visual (front-end) de la aplicación. Este programa es ligero y versátil gracias al amplio arsenal de extensiones con las que puede interactuar, además de ser capaz de trabajar con las tecnologías más actuales y las necesarias para el trabajo de sistemas o aplicaciones orientados a la web («Visual Studio Code - Code Editing. Redefined» 2022).
- Para el modelado de la aplicación web se decide utilizar **Visual Paradigm** para UML en su versión 8.0, por ser una herramienta que soporta el ciclo de vida completo en el desarrollo de software: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite dibujar todo tipo de diagrama de clases, generación de código a partir de diagramas y generar documentación. Provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java.NET y PHP (Baquero Hernández y Mendoza Peña 2016).

- **HTML5** para la gestión de los contenidos web. HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente, pero incluso, aún cuando algunas Interface de Programación de Aplicaciones (API) y la especificación de CSS3 no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y JavaScript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada, bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y JavaScript hace el resto que, como se conoce, es extremadamente significativo. Más allá de esta integración, la estructura sigue siendo parte esencial de cualquier desarrollo. La misma provee los elementos necesarios para ubicar contenido estático o dinámico, y es también una plataforma básica para aplicaciones.

  Con la variedad de dispositivos para acceder a Internet y la diversidad de interfaces disponibles para interactuar con la web, un aspecto básico como la estructura se vuelve parte vital del documento. Ahora la estructura debe proveer forma, organización y flexibilidad. Para trabajar y crear sitios webs y aplicaciones con HTML5, se necesita conocer primero cómo esa estructura es construida. Crear cimientos fuertes ayudará más adelante a aplicar el resto de los componentes para aprovechar completamente estas nuevas tecnologías.

- **CSS** es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas. La separación de los contenidos y su presentación posee numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos") (Gauchat 2012).

  Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en múltiples dispositivos diferentes. Si el lenguaje HTML/XHTML se utiliza para marcar los contenidos, es decir, para designar lo que es un párrafo, lo que es un titular o lo que es una lista de elementos, el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, entro otros.

  De modo que CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de éste, características tales como las capas o layouts, los colores y las fuentes. Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css y reducir la complejidad y la repetición de código en la estructura del documento.

- **Javascript** es un lenguaje interpretado usado para múltiples propósitos, considerado inicialmente como un complemento. Una de las innovaciones que ayudó a cambiar el modo en que se reconoce actualmente Javascript fue el desarrollo de nuevos motores de interpretación creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio (Gauchat 2012).

  Esta capacidad mejorada permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web. Para aprovechar esta plataforma de trabajo ofrecida por los nuevos navegadores, Javascript fue expandido en relación con la portabilidad e integración. A la vez, interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales.

Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores. La idea es hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web (Stoff 2021).

Actualmente todos los navegadores modernos de gran relevancia soportan completamente ECMAScript 5.1 e incluso las versiones más actuales. Los navegadores más antiguos soportan por lo menos ECMAScript 3. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Modelo de Objeto de Documento (DOM).

- Se decide utilizar **Python** como tecnología en el lado del servidor, por sus características como lenguaje de programación potente y fácil de aprender. Posee estructuras de datos de alto nivel y un enfoque sencillo pero eficaz de la programación orientada a objetos. Su sintaxis elegante y su tipificación dinámica, junto con su naturaleza interpretada, lo convierten en un lenguaje apropiado para la creación de scripts y el desarrollo rápido de aplicaciones en muchos ámbitos y en la mayoría de las plataformas (Python 2022).

  El intérprete de Python se puede ampliar fácilmente con nuevas funciones y tipos de datos implementados en C o C++ (u otros lenguajes invocables desde C). Python también es adecuado como lenguaje de extensión para aplicaciones personalizables (Python 2022).

- **SQLite** fue seleccionado como gestor de base de datos. Es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional, autónomo, sin servidor y sin configuración. El código de SQLite es de dominio público y, por tanto, es de uso gratuito para cualquier propósito comercial o privado (SQLite 2022). A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor separado. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con varias tablas, índices, disparadores y vistas está contenida en un solo archivo de disco.

  SQLite funciona muy bien como motor de base de datos para la mayoría de los sitios web de tráfico bajo a medio. La cantidad de tráfico web que SQLite puede manejar depende de la cantidad de uso que el sitio web haga de su base de datos. SQLite es capaz de soportar grandes cantidades de peticiones, siendo un estimado, la cifra de 100 000 visitas. La infraestructura que presenta la Universidad Tecnológica de La Habana “José Antonio Echeverría”, CUJAE, no supera la cantidad estimada de 5000 visitas diarias. La cifra de 100

000 visitas/día es una estimación conservadora, no un límite superior estricto. Se ha demostrado que SQLite funciona, con 10 veces esa cantidad de tráfico (Hipp 2022).

- **Django** es el framework a utilizar para la implementación de la aplicación web propuesta. Surgió de una necesidad muy práctica: World Online, una operación web de un periódico, para la construcción de aplicaciones web intensivas en los plazos del periodismo. En la fuerte dinámica de una sala de redacción, World Online facilitó en sólo cuestión de horas llevar una complicada aplicación web desde el concepto hasta el lanzamiento público.

  En 2003, los desarrolladores de World Online abandonaron PHP y empezaron a utilizar Python para desarrollar sus sitios web. A medida que construían sitios intensivos y muy interactivos, como Lawrence.com, empezaron a extraer un marco de desarrollo web genérico que les permitía construir aplicaciones web cada vez más rápidamente. Este marco se ajustaba constantemente, añadiendo mejoras a lo largo de dos años. En 2005, World Online decidió abrir el software resultante, Django («Django: The web framework for perfectionists with deadlines» 2022).

- **Bootstrap 5:** es un marco de trabajo libre, de código abierto, para la creación de sitios web y aplicaciones. Una de las principales ventajas que ofrece es que los sitios y aplicaciones que lo implementan son adaptables a cualquier tipo de dispositivo; permitiendo a los usuarios hacer uso de él con una gran flexibilidad y de una manera mucho más sencilla e intuitiva (Barrera M 2018).

1.5 Patrones de diseño a utilizar

Los patrones de diseño se aplican a un elemento específico como un agregado de componentes para resolver un problema determinado. Representan una descripción de las clases y objetos, comunicándose entre sí para resolver un problema de diseño general en un contexto particular. El uso de patrones posibilita estandarizar el modo en que se realiza el diseño y proporciona reusabilidad, extensibilidad y mantenimiento del código (Pressman 2003).

A partir de los análisis realizados, se determinó la utilización de patrones de diseño que se incluyen dentro de los dos grandes grupos siguientes: los GRASP (*General Responsibility Assignment Software Patterns*, por sus siglas en inglés), que son patrones generales de software para asignación de responsabilidades; y los GOF (*Gang of Four*, por sus siglas en inglés), encargados de la inicialización, agrupación y comunicación de los objetos (Design Patterns 2009).

A continuación, se describen los patrones utilizados en esta investigación:

Patrones de diseño GOF

- **Comando (*Command*):** este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma.

- **Plantilla (*Template*):** se utilizaron vistas genéricas basadas en clases donde el acceso a los algoritmos se realiza desde estas plantillas, sin necesidad de cambiar la estructura de los mismos.

Patrones de diseño GRASP

- **Alta cohesión:** resuelve el problema de asignar una responsabilidad de manera que la cohesión permanezca alta. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. El patrón se evidencia en cada una de las clases de la herramienta, de tal forma que se elimina la sobrecarga de responsabilidades.

- **Bajo acoplamiento:** el acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencias con otras clases. Este patrón se tuvo presente debido a la importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización.

- **Experto:** este patrón es utilizado para resolver el problema de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (Larman, 2003)**.** Es utilizado para identificar la información que tiene cada clase y así otras puedan manipularlas.

- **Controlador:** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocio debe estar separada de la capa de presentación, lo que aumenta la reutilización de código y permite a la vez tener un mayor control.

Principio de diseño asociado:

- ***Don't Repeat Yourself* (DRY):** es una filosofía de definición de procesos que promueve la reducción de la duplicación, especialmente en la informática. Es implementado en todo el código, ya que es uno de los principios asociados del framework Django.

1.6 Conclusiones parciales del capítulo

El estudio realizado, apoyado en los métodos de investigación científica, permitió la elaboración del marco teórico conceptual que soporta la investigación.

Se definieron las características generales que se tuvieron en cuenta para el desarrollo de la propuesta de solución, la cual se implementó utilizando la metodología XP para guiar el proceso de desarrollo y el lenguaje UML para el modelado de la solución mediante Visual Paradigm en su versión 8.0. Para la implementación de la aplicación web se determinó la utilización de la herramienta PyCharm, junto a las tecnologías del lado del cliente y del lado del servidor previamente mencionadas.

El análisis de los requerimientos de los sistemas informáticos existentes y de las funcionalidades identificadas en el ámbito del objeto de estudio, permitió determinar la conveniencia de desarrollar una aplicación web, con carácter modular y capacidades para garantizar la interoperabilidad con otras aplicaciones y sistemas de apoyo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

El presente capitulo está dedicado a la elaboración de una propuesta de solución teniendo como base las fases de planificación y diseño definidas en la metodología XP. Se realiza un diagrama de clases para la comprensión del modelo conceptual y de la relación entre las clases definidas, se exponen las historias de usuario, los usuarios del sistema, los patrones de diseño, la arquitectura implementada y el uso de las tarjetas clase-responsabilidad-colaboración de la propuesta de solución.

1.1. Descripción del negocio

La Residencia Estudiantil es un área y proceso de apoyo en la CUJAE que sirve de aseguramiento del proceso de formación y escenario de actuación del proceso de extensión universitaria. Es un área que se subordina a la Vicerrectoría de Extensión Universitaria (VREU). La dirección está compuesta por el Director, dos Subdirectores (uno para atender la parte administrativa y el otro para atender la informatización), una Especialista en Atención Integral al Becario (AIB), un Técnico A en Gestión de los Recursos Humanos y la Secretaria.

Además, se hace necesario contar con un personal de administración en cada uno de los edificios, conformado por un Especialista en AIB, un Técnico A en AIB y un Técnico B en AIB.

De modo general, en la Figura 2.1 se muestra la estructura organizativa de la Residencia Estudiantil de la CUJAE.

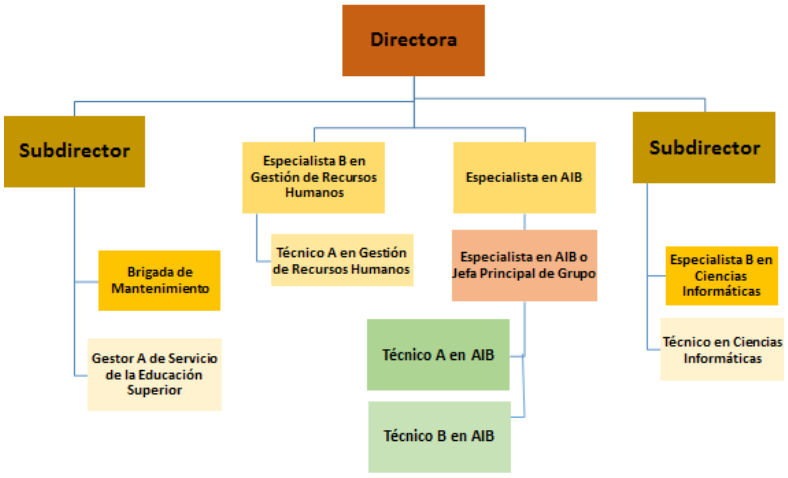


Figura 2. 1 Organigrama de la Residencia Estudiantil de la CUJAE

Las actividades principales que funcionan como subprocesos del área son:

- **Alojamiento:** su objetivo es garantizar condiciones aceptables para ofrecer hospedaje a los estudiantes becados.
- **Trabajo educativo:** es el trabajo de formación continua e integral en valores y político-ideológico con los estudiantes becados.
- **Dirección:** es la encargada de supervisar, controlar y dar solución a los principales problemas que existen en el área. Es una de las actividades que más dinamismo y proyección exige. Se rige y vela por el cumplimiento de lo establecido en los procedimientos y la adecuada actualización y uso de la documentación y los recursos. Se concreta a partir de cumplir y hacer cumplir la secuencia de actos o conjunto de reglas, políticas y actividades establecidas en el área y la Universidad.
- **Mantenimiento:** consiste en organizar y realizar la reparación y mantenimiento de las instalaciones, inmuebles, equipos y otros recursos para prolongar su uso y vida útil.
- **Gestión de Aprovisionamiento:** su objetivo es realizar de manera eficiente y eficaz las actividades de adquisición, almacenamiento y distribución de los recursos necesarios para el funcionamiento de la Residencia Estudiantil y sus servicios.
- **Gestión de Recursos Humanos:** consiste en fortalecer la labor educativa y el desarrollo integral de los trabajadores, fomentando el buen desempeño, el compromiso, la responsabilidad, la excelencia y la mejora continua.

En el mapa de procesos que muestra la Figura 2.2, se identifican las principales actividades que, según su clasificación, juegan un papel determinante en el logro del estado deseado de la Residencia Estudiantil.

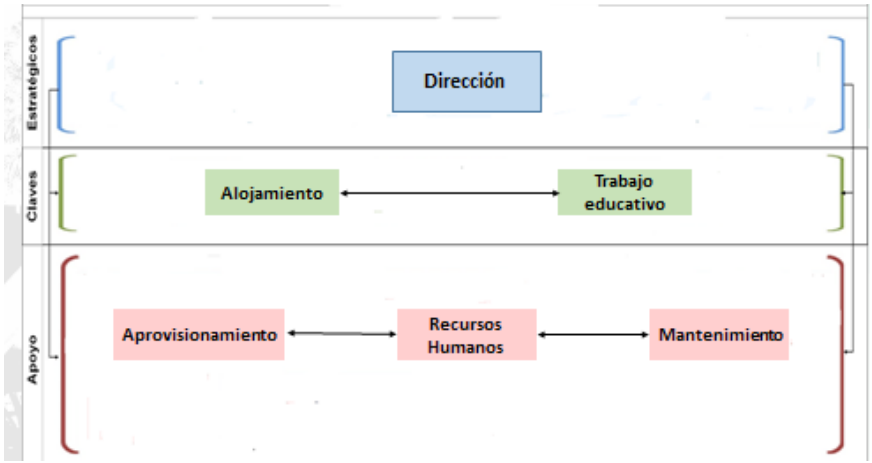


Figura 2. 2 Mapa de procesos de la Residencia Estudiantil de la CUJAE

El diseño según estructura de la Residencia Estudiantil es de 10 edificios con un total de 410 cuartos. La capacidad máxima por cuarto es de 8 en los edificios 100, 200 y 300. En los restantes edificios, es decir el 400 y 500, la capacidad es de 12 estudiantes por cuarto.

Para realizar la asignación de estudiantes por cuartos se hace necesario consultar al personal administrativo sobre las condiciones reales que presentan cada uno de ellos. El análisis se apoya en la información almacenada en una base de datos elaborada en Microsoft Excel. Esta aplicación almacena la información de la situación actual de la residencia en cuanto a su capacidad cubierta y ayuda a la planificación de la actividad de asignación de capacidades de alojamiento para los estudiantes de nuevo ingreso o por excepcionalidad.

La Figura 2.3 ejemplifica el tipo de condición que se puede presentar en la Residencia Estudiantil con relación a las capacidades disponibles por cuartos de los edificios, de acuerdo con lo anteriormente expuesto. Esto ilustra la necesaria interrelación entre los diferentes actores que forman parte del proceso. De modo que para determinar esas capacidades la dirección de la Residencia Estudiantil y las Facultades juegan un papel fundamental, para mantener siempre actualizada la situación del estudiante en la Universidad y las capacidades de alojamiento existentes y disponibles.

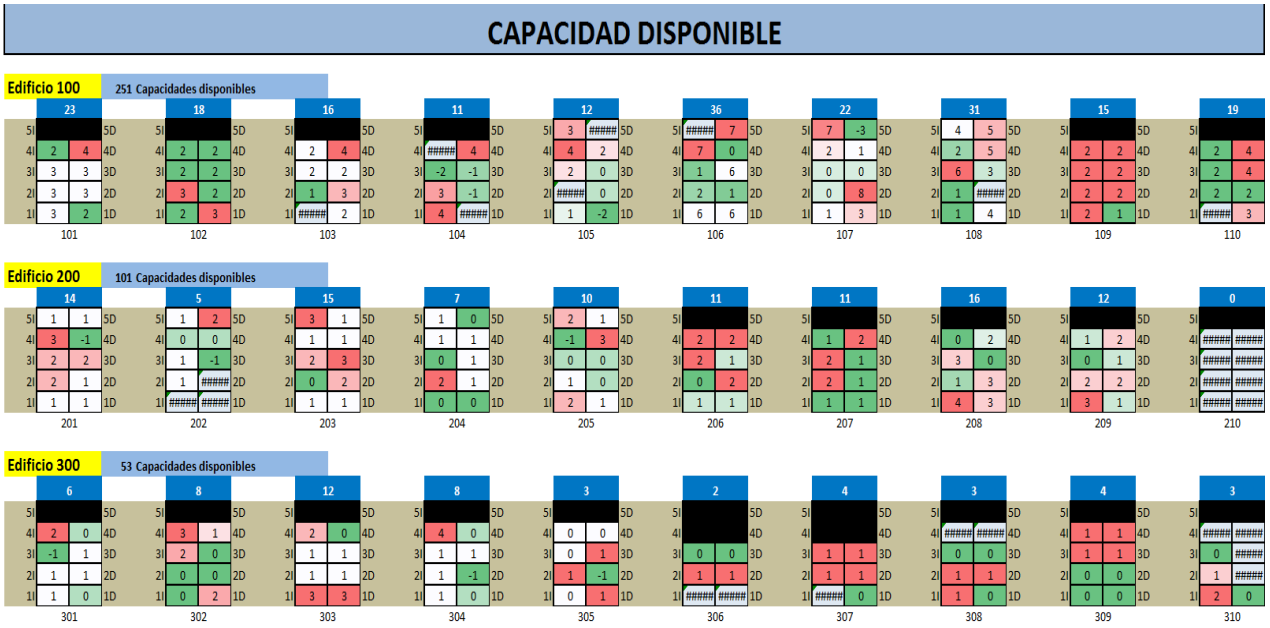


Figura 2. 3 Control de capacidades por edificios

En el presente, se reduce a 346 los cuartos disponibles en la Residencia Estudiantil. La cantidad de becados es en total de 1532 estudiantes, de ellos 588 hembras y 944 varones. El porciento de ocupación por edificios se muestra en la Figura 2.4. En la asignación de los estudiantes a los edificios se procura una distribución a fin con las carreras y el año que estudian.

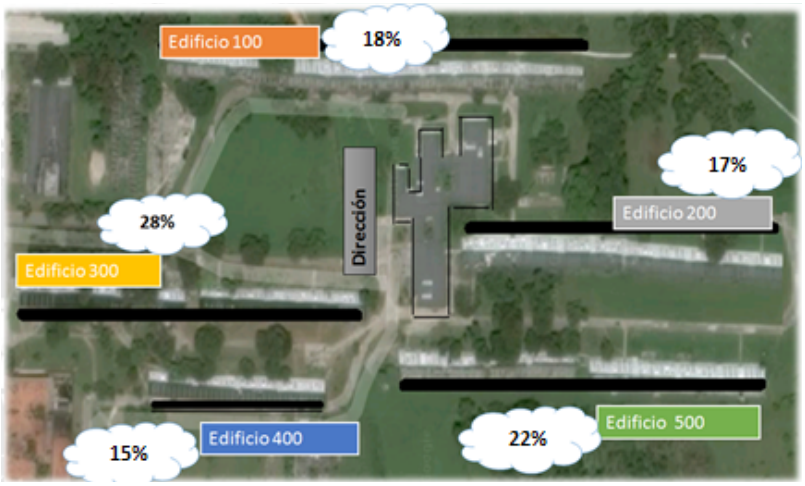


Figura 2. 4 Distribución de la ocupación porcentual por edificios

Desde la Dirección de la Residencia Estudiantil se coordinan todas las actividades relacionadas con la gestión del alojamiento y la asignación de capacidades. Este nivel es el encargado de supervisar,

controlar y dar solución a las principales problemáticas de carácter organizativo y de asignación. Para ello se tienen en cuenta las variables que intervienen en la asignación de capacidades:

- Estrategia docente determinada por la Universidad de acuerdo a los contextos.
- Capacidad del cuarto (de 8 o 12 estudiantes).
- El número de taquillas disponibles por cuarto.
- Condiciones materiales de los cuartos.
- Matrícula Carrera/Año.
- Cantidad de Facultades.
- Ubicación Hembras/Varones.

El proceso de ubicación de los estudiantes guarda una estrecha relación con la instancia inicial que lo recibe, el flujo de información y el diseño de las contrapartidas en el trabajo que realizan las áreas de la Universidad que intervienen. Para todo estudiante que realice su matrícula en una facultad, se registran sus datos en el SIGENU (herramienta que permite la automatización de los procesos docentes y estadísticos de la Universidad) y se especifica si es estudiante becado o externo. Esta información determina su acceso a la Residencia Estudiantil, en el caso de ser becado, para finalizar el proceso de matrícula.

La participación de la facultad a través del manejo de la información durante el proceso de matrícula por el SIGENU, posibilita que el estudiante de provincia llegue a la Residencia con una primera orientación y con los datos primarios para su posterior asignación a un cuarto de un edificio.

La Figura 2.5 caracteriza la información resultante del recorrido del estudiante en cuanto a la realización del proceso de matrícula. Por una parte, la facultad, por medio de la secretaría docente, recoge la información a través del SIGENU, y por la otra, la Residencia Estudiantil recibe esa información primaria y solicita al estudiante otro grupo importante de datos, almacenando todo ello en sus propias herramientas y registros. Esto evidencia la necesidad de unificar la información y crear sistemas informáticos que digitalicen la información física.

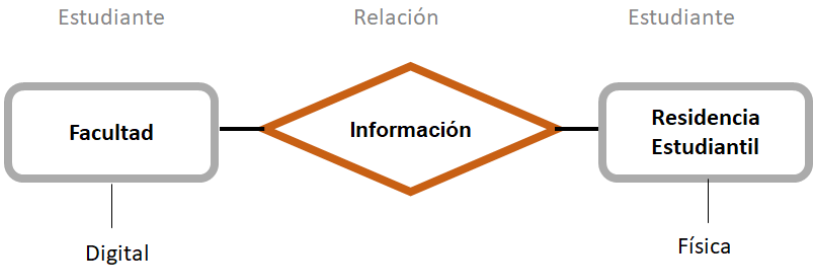
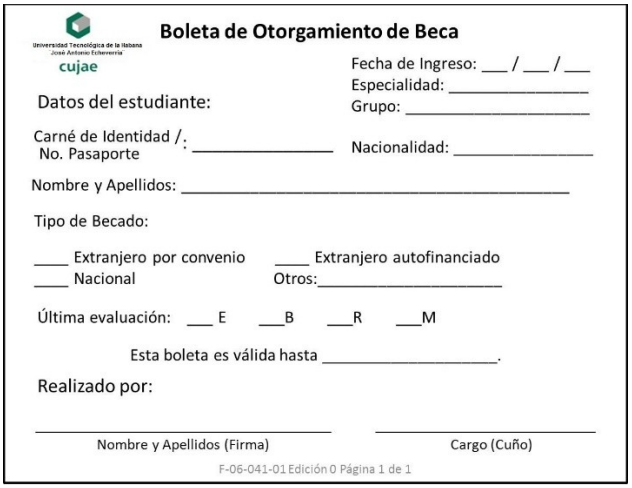


Figura 2. 5 Caracterización del flujo de información en el proceso de matrícula del estudiante

El documento que notifica que luego del recorrido del estudiante por la facultad, se ha realizado la recepción de la información y ya está habilitado para completar el proceso de matrícula y asignarle una capacidad de alojamiento en la Residencia Estudiantil, es la boleta de otorgamiento de beca. Esta boleta es firmada por los encargados del proceso de matrícula en la facultad. A continuación, la Figura 2.6 muestra este documento de control y recepción.



Boleta de Otorgamiento de Beca

cujae

Fecha de Ingreso: ___ / ___ / ___
Especialidad: _____
Grupo: _____

Datos del estudiante:

Carné de Identidad /. No. Pasaporte: _____ Nacionalidad: _____

Nombre y Apellidos: _____

Tipo de Becado:

____ Extranjero por convenio ____ Extranjero autofinanciado
____ Nacional Otros:_____

Última evaluación: ___ E ___B ___R ___M

Esta boleta es válida hasta _____.

Realizado por:

_____ _____
Nombre y Apellidos (Firma) Cargo (Cuño)

F-06-041-01 Edición 0 Página 1 de 1

Figura 2. 6 Boleta de control para el proceso de matrícula

A través de esta boleta de otorgamiento de beca se hace posible unificar la información necesaria para que la facultad y la Residencia Estudiantil puedan conocer los datos de los estudiantes, así como su ubicación. Esto permite y garantiza la mejor organización del trabajo en la Universidad.

También, como parte del alcance que debe tener la Residencia Estudiantil como proceso de apoyo de la Universidad en cuanto a su estructura organizativa funcional y la gestión de sus recursos para el funcionamiento del área, se recogieron los criterios abordados por la MS.c Dianelys Tejeda Villazón

en su tesis de maestría “Aplicación del Procedimiento de Gestión Integrada de Riesgos para la mejora de la Capacidad de Prevención Estratégica de la Residencia Estudiantil de la CUJAE”. La autora afirma que para mejorar el funcionamiento del proceso era necesario identificar los principales riesgos y causas que pueden ocurrir en cada uno de los subprocesos del proceso Residencia Estudiantil (Tejeda Villazón 2019).

En la Figura 2.7 se muestra el total de riesgos por cada subproceso del proceso Residencia Estudiantil, así como las causas que los provocan, determinándose un total de 13 riesgos y 47 causas asociadas a los mismos.

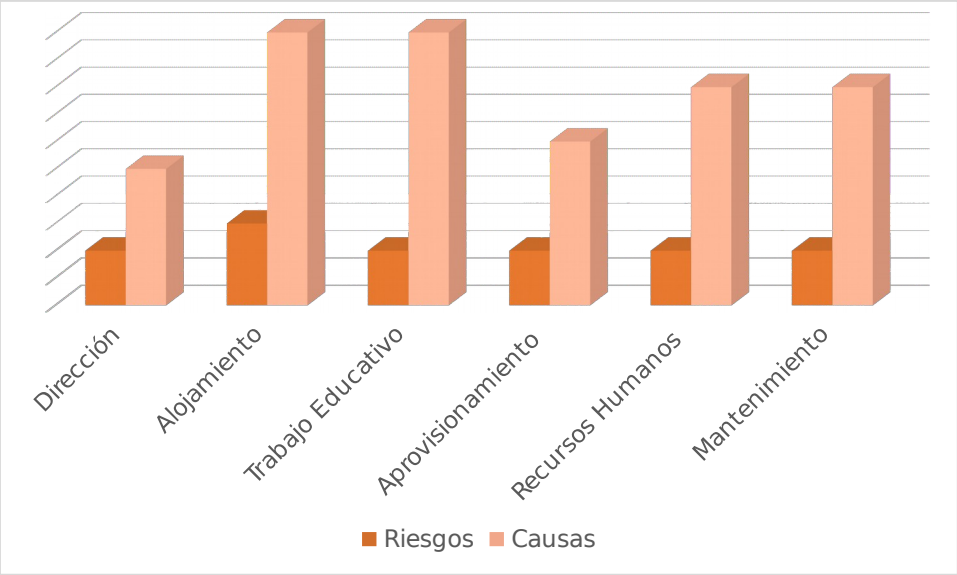


Figura 2. 7 Riesgos por actividades y causas asociadas

Se observa en la Figura 2.7 que el subproceso 2 Alojamiento es el más afectado con respecto a la cantidad de riesgos que posee, mientras que los subprocesos 2 y 3 Alojamiento y Trabajo Educativo son los más implicados con relación a la cantidad de causas que poseen. Después le siguen los subprocesos de Recursos Humanos y Mantenimiento, con dos riesgos y 8 causas cada uno. Finalmente, los subprocesos de Dirección y Aprovisionamiento tienen 2 riesgos cada uno, identificándose 5 y 6 causas, respectivamente.

A continuación, se ejemplifica en la Tabla 2.1 los riesgos del subproceso de alojamiento y las causas asociadas, así como la probabilidad e impacto estimado, la cuantía del riesgo y la evaluación.

Tabla 2. 1 Análisis de los riesgos en el subproceso alojamiento. Fuente: (Tejeda Villazón 2019)

Subprocesos	Descripción del riesgo	Causa del riesgo	Probabilidad del riesgo			Impacto estimado del riesgo (I)	Valor del riesgo VAR = PS*I	Evaluación del riesgo
			ni	nd	PS= $\sqrt{¿*n}$			
Sp 2 Alojamiento	R0102 Ausencia de procesos de automatización de las actividades en la Residencia Estudiantil.	C010201 Insuficiente soporte técnico C010202 Ausencia de programas informáticos para perfeccionar la gestión de las actividades.	0,8	0,8	0,8	8	6,4	**Alto**
	R0202 Disminución de las capacidades de alojamiento en la Residencia Estudiantil	C020201 Deterioro de las condiciones constructivas en la Residencia Estudiantil. C020203 Insuficientes recursos materiales para las reparaciones y mantenimiento constructivo de las instalaciones. C030102 Formación de personal errónea o insuficiente	0,8	0,8	0,8	9	7,2	**Alto**

1.2. Fases del proceso de desarrollo

De acuerdo con lo indicado por la metodología XP, a continuación se relacionan las fases del proceso de desarrollo del software y las tareas principales dentro de ellas.

1.2.1. Planificación

En esta fase se llevan a cabo tareas como la realización de las historias de usuario (HU), las cuales deben ser elaboradas por o con el cliente, y es en esta misma fase donde los programadores llevan a cabo la estimación de esfuerzo de cada una de estas historias de usuario. Se realizan debates sobre qué características debe presentar la primera iteración y se establecen los requisitos funcionales y no funcionales. También se define un cronograma de trabajo de conjunto con el cliente. Al final de esta fase se devuelve un plan de entregas.

1.2.1.1. Requisitos Funcionales y No Funcionales

Uno de las actividades de suma importancia para el aseguramiento de la calidad del software es la captura de los requisitos del sistema, lo que se enmarca dentro de la temática de la ingeniería de requisitos. Con tal propósito se establecieron los requisitos funcionales y no funcionales a tener en cuenta en el diseño e implementación de la aplicación. Por una parte, los requisitos funcionales definen comportamientos o funciones específicas, mientras los requisitos no funcionales especifican criterios que pueden usarse para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos.

En este trabajo se consultó, para tener criterios sobre el tratamiento de la calidad de la aplicación web en lo referido a la ingeniería de requisitos, la (Norma ISO/IEC 25010: 2016 2016), la cual es una adopción idéntica por el método de traducción de la Norma Internacional ISO IEC 25010:2011 *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*.

Para el desarrollo de este trabajo se utilizó dentro de las divisiones de la serie SQuaRE, la ISO/IEC 2501n - División de Modelo de la Calidad. Las Normas Internacionales que conforman esta división presentan un modelo de la calidad detallado para los sistemas informáticos y productos de software, la calidad en el uso y datos. También se proporciona orientación práctica sobre el uso de los modelos de la calidad.

- **Requisitos funcionales:**

  Los requisitos funcionales (RF) describen lo que la aplicación debe hacer, así como las condiciones y capacidades que debe cumplir el software, o producto en general, para que las peticiones del cliente queden satisfechas. (Sommerville 2015)

  A través de las técnicas de captura de requisitos (entrevistas con el usuario y tormentas de ideas), se obtuvieron los requisitos funcionales que se muestran en la tabla 2.2.

Tabla 2. 2 Descripción de los requisitos funcionales identificados. Fuente: Elaboración propia

Requisito	Descripción
RF1. Autenticar usuario	La aplicación debe permitir autenticar los usuarios con las credenciales correspondientes
RF2. Adicionar rol	Permite adicionar un rol en la aplicación
RF3. Visualizar rol	Permite visualizar los roles en la aplicación
RF4. Modificar rol	Permite actualizar los datos de un rol en la

	aplicación
RF5. Eliminar rol	Permite eliminar los datos de un rol en la aplicación
RF6. Adicionar usuario	Permite adicionar un usuario en la aplicación
RF7. Visualizar usuario	Permite visualizar los usuarios en la aplicación
RF8. Modificar usuario	Permite actualizar los datos de un usuario en la aplicación
RF9. Eliminar usuario	Permite eliminar los datos de un usuario en la aplicación
RF10. Adicionar edificio	Permite adicionar un edificio en la aplicación
RF11. Visualizar edificio	Permite visualizar los edificios en la aplicación
RF12. Modificar edificio	Permite actualizar los datos de un edificio en la aplicación
RF13. Eliminar edificio	Permite eliminar los datos de un edificio en la aplicación
RF14. Adicionar bloque	Permite adicionar un bloque en la aplicación
RF15. Visualizar bloque	Permite visualizar los bloques en la aplicación
RF16. Modificar bloque	Permite actualizar los datos de un bloque en la aplicación
RF17. Eliminar bloque	Permite eliminar los datos de un bloque en la aplicación
RF18. Adicionar cuarto	Permite adicionar un cuarto en la aplicación
RF19. Visualizar cuarto	Permite visualizar los cuartos en la aplicación
RF20. Modificar cuarto	Permite actualizar los datos de un cuarto en la aplicación
RF21. Eliminar cuarto	Permite eliminar los datos de un cuarto en la aplicación
RF22. Alojar estudiante	Permite el alojamiento de forma manual de los estudiantes
RF23. Definir facultades por edificio	Los edificios deben tener facultades asociadas para identificar de forma más eficiente a los estudiantes
RF24. Adicionar estudiante	Permite adicionar un estudiante en la aplicación
RF25. Visualizar estudiante	Permite visualizar los estudiantes en la aplicación
RF26. Modificar estudiante	Permite actualizar los datos de un estudiante en la aplicación
RF27. Eliminar estudiante	Permite eliminar los datos de un estudiante en la aplicación
RF28. Elaborar reporte	Permite elaborar reportes sobre el estado de los estudiantes
RF29. Exportar reporte	Permite exportar los reportes generados en formato .pdf
RF30. Buscar estudiante	Permite la búsqueda de estudiantes a partir de las credenciales correspondientes

- **Requisitos No Funcionales:**

Los requisitos no funcionales (RNF) definen las características que debe presentar la aplicación para ofrecer una respuesta rápida y una interfaz legible, simple de usar, atractiva e interactiva. A partir del trabajo con la (Norma ISO/IEC 25010: 2016 2016), para el desarrollo de la presente investigación se definieron los RNF siguientes:

1. Usabilidad
   - Reconocibilidad: Grado en que la aplicación web permite al usuario reconocer si la misma es apropiada para sus necesidades.
   - Protección ante errores de usuarios: Grado en que la aplicación web protege a los usuarios de cometer errores.
   - Accesibilidad: Grado en que la aplicación web pueda ser utilizada por personas con un amplio rango de características y capacidades para alcanzar un objetivo definido, en un contexto de uso especificado.
2. Fiabilidad
   - Madurez: Grado en que la aplicación web cumple con la fiabilidad requerida en condiciones de operación normales.
   - Disponibilidad: Grado en que la aplicación web está operativa y accesible cuando sea necesario para su uso.
   - Tolerancia ante fallos: Grado en que la aplicación web opera según lo previsto, independientemente de la presencia de fallos en el hardware o software.
   - Recuperabilidad: La aplicación web debe permitir recuperar los datos directamente afectados y restablecer el estado deseado, cuando ocurre una interrupción o una falla.
3. Seguridad
   - Confidencialidad: La aplicación web debe permitir que los datos sean accesibles solo por las personas autorizadas.
   - Integridad: La aplicación web debe impedir el acceso no autorizado, o la modificación de programas o datos.
   - No rechazo: Grado en el que las acciones o eventos pueden probarse que han tenido lugar para que posteriormente no sean negadas.
   - Responsabilidad: Grado en el que las acciones de una entidad pueden atribuirse únicamente a esta.
   - Autenticidad: Grado en el que la identidad de un sujeto o recurso puede probar ser quien dice ser.
4. Mantenibilidad
   - Modularidad: La aplicación debe estar integrada por componentes individuales de tal manera que un cambio en uno de estos tiene un impacto mínimo en los otros componentes.

- Modificabilidad: La aplicación debe permitir ser modificada de forma eficaz y eficiente sin introducir defectos o degradar su calidad.
- Testabilidad: Grado de eficacia y eficiencia con la que los criterios de prueba se pueden establecer para un sistema, producto o componente y determinar si se han cumplido los criterios.

5. Portabilidad
   - Adaptabilidad: La aplicación web debe ser adaptada de forma eficaz y eficiente para diferentes hardware o software en evolución, u otros entornos operativos o de uso.
   - Instalabilidad: La aplicación web debe ser eficaz y eficiente permitiendo ser instalada y/o desinstalada con éxito en un entorno específico.

1.2.1.2. Historias de usuario

Las HU es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que la aplicación debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas (Letelier Torres y Hilario Canós 2003).

Para almacenar la información correspondiente a cada nueva instancia de proyectos, cuyas especificaciones funcionales son descritas a través de HU, se ha definido la plantilla que se muestra en la Figura 2.8. Se han adicionado algunos conceptos, tales como: Variables, Dominio de las variables, Resultado esperado y Escenarios (Verona Marcos 2018).

Identificador	Nombre HU	Prioridad	Puntos estimados
Usuario:			
Descripción:			
Programador responsable:			
Escenarios			
Nombre	**Condición**	**Resultado esperado**	
Variables			
Nombre	**Tipo o Dominio**	**Descripción**	
Iteración asignada			
Observaciones			

Figura 2. 8 Plantilla propuesta para describir la HU.

Se generaron para el desarrollo de este trabajo a partir de los requisitos funcionales un total de 8 historias de usuario:

1. HU#1 Autenticar usuario: comprende el RF1.
2. HU#2 Gestionar rol: comprende los requisitos funcionales RF2, RF3, RF4 y RF5.
3. HU#3 Gestionar usuario: comprende los requisitos funcionales RF6, RF7, RF8 y RF9.
4. HU#4 Gestionar edificio: comprende los requisitos funcionales RF10, RF11, RF12 y RF13.
5. HU#5 Gestionar bloque: comprende los requisitos funcionales RF14, RF15, RF16 y RF17.
6. HU#6 Gestionar cuarto: comprende los requisitos funcionales RF18, RF19, RF20 y RF21.

7. HU#7 Gestionar estudiante: comprende los requisitos funcionales RF24, RF25, RF26 y RF27.
8. HU#8 Gestionar asignación: comprende los requisitos funcionales RF22, RF23 y RF30.
9. HU#9 Gestionar reporte: comprende los requisitos funcionales RF28 y RF29.

En las tablas 2.3, 2.4 y 2.5, se muestran tres de las HU desarrolladas en este trabajo:

Tabla 2. 3 Historia de usuario "Gestionar reporte"

Identificador	**Nombre HU**	**Prioridad**	**Puntos estimados**
HU 9	Gestionar Reporte	Alta	
Usuario: Administrador/a del edificio de la Residencia Estudiantil			
Descripción: la aplicación debe permitir que los usuarios con los permisos correspondientes elaboren, visualicen, modifiquen y eliminen reportes y los exporten, a partir de los datos válidos de los estudiantes almacenados en el sistema.			
Programador responsable: Raikol Álvarez León			
Escenarios			
Nombre	**Condición**	**Resultado esperado**	
Elaboración exitosa	Los datos del estudiante deben ser comprobados contra el SIGENU El usuario debe tener los permisos correspondientes	Se generan los reportes de los estudiantes a partir de los datos existentes en la aplicación.	
Exportación exitosa	El reporte que se desea exportar debe ser elaborado exitosamente	Se exportan con éxito los reportes generados al tipo de formato deseado.	
Variables			
Nombre	**Tipo o Dominio**	**Descripción**	

Estudiantes	Conjunto de Estudiantes	Listado de estudiantes de la Residencia Estudiantil con los valores correspondientes de cada estudiante
Estudiante	Objeto con atributos: nombre_estudiante de tipo cadena entre 50 y 100, genero de tipo cadena opciones {masculino, femenino}, carrera de tipo cadena entre 50 y 100	Cada estudiante registrado en la aplicación web
Reportes	Conjunto de Reportes	Listado de reportes que han sido elaborados
Reporte	Objeto con atributos: nombre_estudiante de tipo cadena entre 50 y 100, genero de tipo cadena opciones {masculino, femenino}, carrera de tipo cadena entre 50 y 100	Cada reporte que ya está realizado en la aplicación web
Formatos	Conjunto de formatos disponibles	Formatos disponibles para exportar los reportes existentes en la aplicación web
Formato	Objeto con tipos de formatos disponibles(xls, pdf)	Formato especifico al que puede ser exportado los reportes existentes en la aplicación web
Iteración asignada	8	
Observaciones		

Tabla 2. 4 Historia de usuario "Gestionar cuarto"

Identificador	**Nombre HU**	**Prioridad**	**Puntos estimados**
HU 6	Gestionar Cuarto	Alta	
Usuario: Administrador/a de la Residencia Estudiantil de la CUJAE			
Descripción: la aplicación debe permitir que los usuarios con los permisos correspondientes adicionen, visualicen, actualicen y eliminen cuartos a partir de la infraestructura real existente en la Residencia Estudiantil.			
Programador responsable: Raikol Álvarez León			

Escenarios		
Nombre	**Condición**	**Resultado esperado**
Creación exitosa	En una primera instancia se comprueba mediante un chequeo sistemático el estado actual de los cuartos. Después, se comunican por las administradoras de los edificios, dígase especialistas en AIB, los avances que se tengan en el proceso de reparación y mantenimiento de los cuartos por los terceros. Luego se entran en la aplicación los datos verificados y necesarios para la creación de un cuarto. El usuario debe tener los permisos correspondientes	Se generan los cuartos en la aplicación web a partir de los datos insertados.
Variables		
Nombre	**Tipo o Dominio**	**Descripción**
Cuartos	Conjunto de Cuartos	Listado de cuartos de la Residencia Estudiantil con sus valores correspondientes.
Cuarto	Objeto con atributos: block_fk de tipo foranea , identificacion de tipo cadena entre 10 y 20, capacidad_total_planificada de tipo numerica entre 4 y 12, capacidad_real de tipo numerico entre 4 y 12.	Cada cuarto registrado en la aplicación web con sus valores correspondientes.
Iteración asignada	5	
Observaciones		

Tabla 2. 5 Historia de usuario "Gestionar edificio"

Identificador	**Nombre HU**	**Prioridad**	**Puntos estimados**
HU 4	Gestionar Edificio	Alta	
Usuario: Administrador/a de la Residencia Estudiantil de la CUJAE			

<table>
<tr><td colspan="6">Descripción: la aplicación debe permitir que los usuarios con los permisos correspondientes adicionen, visualicen, actualicen y eliminen edificios a partir de la infraestructura real existente en la Residencia Estudiantil.</td></tr>
<tr><td>Programador responsable:
Raikol Álvarez León</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="6">Escenarios</td></tr>
<tr><td>Nombre</td><td colspan="2">Condición</td><td colspan="3">Resultado esperado</td></tr>
<tr><td rowspan="3">Creación exitosa</td><td colspan="2"></td><td colspan="3" rowspan="3">Se generan, actualizan, listan y eliminan los edificios en la aplicación web a partir de los datos insertados.</td></tr>
<tr><td colspan="2">En una primera instancia se comprueba mediante un chequeo sistemático el estado actual de los edificios. Después, se comunican por las administradoras de los edificios, dígase especialistas en AIB, los avances que se tengan en el proceso de reparación y mantenimiento de los edificios por los terceros. Luego se entran los datos verificados y necesarios en la aplicación para la creación de un edificio.</td></tr>
<tr><td colspan="2">El usuario debe tener los permisos correspondientes</td></tr>
<tr><td colspan="6">Variables</td></tr>
<tr><td>Nombre</td><td colspan="2">Tipo o Dominio</td><td colspan="3">Descripción</td></tr>
<tr><td>Edificios</td><td colspan="2">Conjunto de Edificios</td><td colspan="3">Listado de edificios de la Residencia Estudiantil con sus valores correspondientes.</td></tr>
</table>

Edificio	Objeto con atributos: identificacion de tipo cadena entre 10 y 20, capacidad_total_planificada de tipo numerica entre 4 y 12, capacidad_real de tipo numerico entre 4 y 12.	Cada edificio registrado en la aplicación web con sus valores correspondientes.
Iteración asignada	3	
Observaciones		

1.2.1.3. Estimación de esfuerzo por historias de usuario

Previamente a la implementación del sitio web propuesto, se elaboró una tabla de estimación de esfuerzo por cada una de las HU identificadas con anterioridad. A continuación, se muestran en la Tabla 2.6 los resultados:

Tabla 2. 6 Estimación de esfuerzo por Historia de Usuario. Fuente: Elaboración propia

Historias de usuario	Puntos de estimación (horas)
Autenticar usuario	70
Adicionar rol	112
Visualizar rol	80
Modificar rol	130
Eliminar rol	90
Adicionar usuario	2
Visualizar usuario	2
Modificar usuario	2
Eliminar usuario	2
Adicionar edificio	115
Visualizar edificio	91
Modificar edificio	140
Eliminar edificio	70
Adicionar bloque	121
Visualizar bloque	84
Modificar bloque	66
Eliminar bloque	72
Adicionar cuarto	132
Visualizar cuarto	144
Modificar cuarto	81
Eliminar cuarto	55
Alojamiento manual	250
Definir facultades por edificio	160
Adicionar estudiante	170
Visualizar estudiante	193
Modificar estudiante	211

Eliminar estudiante	84
Crear reporte por estudiante	31
Exportar reporte	23
Buscar estudiante	147

1.2.1.4. Plan de iteraciones

Se describen a continuación cada una de las iteraciones propuestas por el equipo, donde la duración total de iteraciones en días se obtiene con base al esfuerzo estimado por el desarrollador:

Iteración #1: en esta iteración se implementó la HU-1, la cual hace referencia a “Autenticar Usuario” y se realizó su posterior programación.

Iteración #2: en esta iteración se implementó la HU-2 y HU-3, la cual se refiere a “Gestionar Rol” y “Gestionar Usuario” respectivamente. Se corrigieron los posibles errores encontrados en la iteración anterior, se discutió con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #3: en esta iteración se implementó la HU-4, la cual se refiere a “Gestionar Edificio”. Se corrigieron los posibles errores encontrados en las iteraciones anteriores, se obtuvo retroalimentación por parte del cliente y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #4: en esta iteración se implementó la HU-5, la cual se refiere a “Gestionar Bloque”. Se corrigieron los posibles errores encontrados en la iteración anterior, se realizaron reuniones con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #5: en esta iteración se implementó la HU-6, la cual se refiere a “Gestionar Cuarto”. Se corrigieron los posibles errores encontrados en la iteración anterior, se realizaron reuniones con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #6: en esta iteración se implementó la HU-7, la cual se refiere a “Gestionar Estudiante”. Se corrigieron los posibles errores encontrados en la iteración anterior, se realizaron reuniones con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #7: en esta iteración se implementó la HU-8, la cual se refiere a “Gestionar Alojamiento”. Se corrigieron los posibles errores encontrados en la iteración anterior, se realizaron reuniones con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

Iteración #8: en esta iteración se implementó la HU-9, la cual se refiere a “Gestionar Reporte”. Se corrigieron los posibles errores encontrados en la iteración anterior, se realizaron reuniones con el cliente obteniendo retroalimentación y se llevaron a cabo las nuevas funcionalidades correspondientes a esta iteración.

A continuación, se muestra en la Tabla 2.7 el tiempo estimado para cada iteración y el orden de su implementación:

Tabla 2. 7 Plan de duración de las iteraciones

Historias de usuario	**Puntos de estimación (horas)**
Autenticar usuario	70
Adicionar rol	112
Visualizar rol	80
Modificar rol	130
Eliminar rol	90
Adicionar usuario	2
Visualizar usuario	2
Modificar usuario	2
Eliminar usuario	2
Adicionar edificio	115
Visualizar edificio	91
Modificar edificio	140
Eliminar edificio	70
Adicionar bloque	121
Visualizar bloque	84
Modificar bloque	66
Eliminar bloque	72
Adicionar cuarto	132
Visualizar cuarto	144
Modificar cuarto	81
Eliminar cuarto	55
Alojamiento manual	250
Definir facultades por edificio	160
Adicionar estudiante	170
Visualizar estudiante	193
Modificar estudiante	211

Eliminar estudiante	84
Crear reporte por estudiante	31
Exportar reporte	23
Buscar estudiante	147

1.2.1.5. Plan de entregas

Una vez identificadas por el cliente las HU y confeccionado por los desarrolladores el plan de iteración, se procede a la confección del plan de entrega, con la intención de que los clientes obtengan una estimación real del tiempo que conllevará la implementación de las funcionalidades que describe cada HU. Seguidamente, se muestra en la Tabla 2.8 el plan de entrega elaborado por el equipo de desarrollo para la fase de implementación:

Tabla 2. 8 Plan de entregas

Iteración	Fecha de entrega	Historia de Usuario
1	10/07/2022	HU-1
2	29/07/2022	HU-2, HU-3
3	16/08/2022	HU-4
4	01/09/2022	HU-5
5	20/09/2022	HU-6
6	18/11/2022	HU-7
7	21/11/2022	HU-8
8	7/12/2022	HU-9

1.2.2. Diseño

La metodología de desarrollo de software XP propone que las tareas definidas por el equipo de desarrollo deben realizarse de forma iterativa e incremental, obteniendo con cada iteración un producto de calidad, funcional y de aporte al producto final, el cual es mostrado al cliente para la gestión asociada con la retroalimentación entre los desarrolladores y el cliente.

XP no es estricto en cuanto a la tarea de diseñar se refiere. En cambio, se puede hacer uso de diferentes herramientas, desde diagramas de clases, diagrama de paquetes, todo esto con el uso del lenguaje de modelado UML, y también el uso de tarjetas Clase Responsabilidad Colaboración.

Los modelos conceptuales no son más que una representación de un sistema hecha desde la composición de conceptos que se utilizan para ayudar a las personas a conocer, comprender o simular un tema que representa el modelo. Incluye las entidades más relevantes y las relaciones

entre ellas. Además, se puede entender como modelo conceptual el adoptar una perspectiva conceptual en la interpretación de los diagramas, si se dibuja un diagrama que represente los conceptos del dominio que se está estudiando.

Estos conceptos se relacionan de manera natural con las clases que los implementan, pero con frecuencia no hay una correlación directa. De hecho, los modelos conceptuales se deben dibujar sin importar (o casi) el software con que se implementarán, por lo cual se pueden considerar como independientes del lenguaje (Fowler 1999).

A continuación, en la Figura 2.9 se muestra el diagrama de clases de la propuesta de solución.

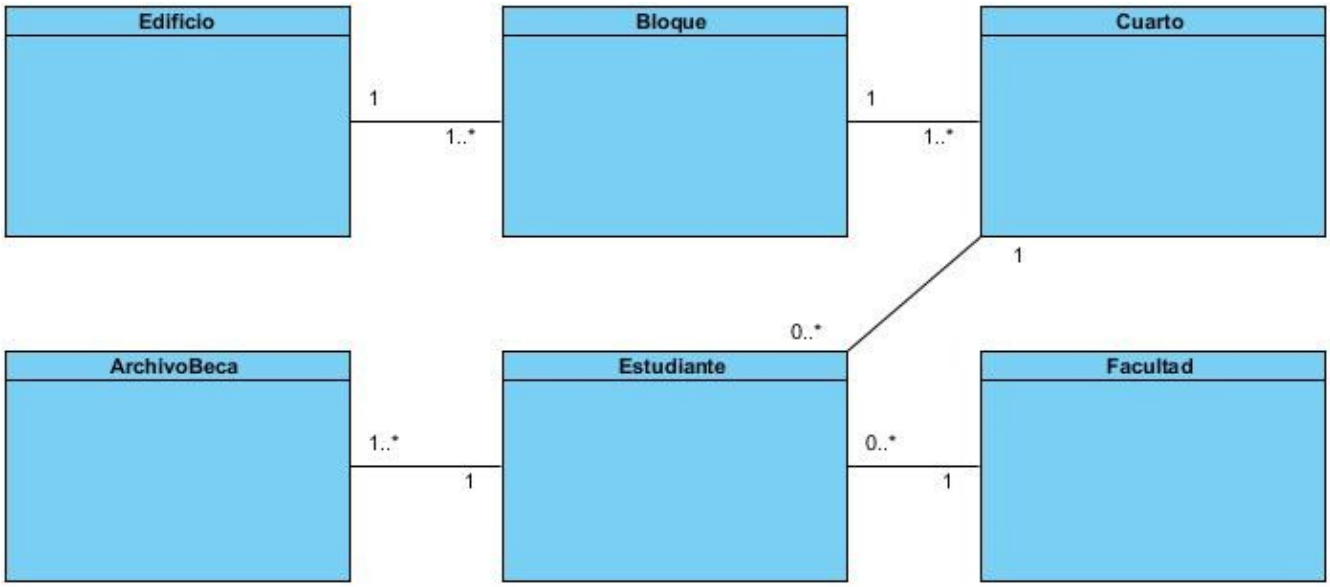


Figura 2. 9 Modelo conceptual. Fuente: Elaboración propia

1.2.2.1. Estilo arquitectónico Llamada - Retorno

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Masuda et al. 2022).

Los estilos arquitectónicos son el proceso de definir una colección de componentes de hardware y software y sus interfaces para establecer el marco para el desarrollo de un sistema informático. El software creado para sistemas basados en computadora puede exhibir uno de muchos estilos arquitectónicos. Estos estilos se basan en:

- Un conjunto de componentes que realizarán una función requerida por el sistema.
- El conjunto de conectores que ayudarán en la coordinación, comunicación y cooperación entre los componentes.
- Las condiciones de cómo se pueden integrar los componentes para formar el sistema.

- Modelos semánticos que ayudan al diseñador a comprender las propiedades generales del sistema.

Para el cumplimiento de estos aspectos se hizo uso de la arquitectura Llamada - Retorno. Se tuvo en cuenta que esta arquitectura es usada para crear programas que sean fáciles de escalar y modificar. Los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. De modo general, refleja la estructura del lenguaje de programación y trabaja con un razonamiento jerárquico. Las tareas que se ejecutan se distribuyen en múltiples procesadores e hilos de control simple, soportados por el lenguaje de programación.

1.2.2.2. Patrón arquitectónico

Un patrón arquitectónico brinda la descripción de un problema particular y recurrente, que aparece en contextos de diseño específicos, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que lo constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí. El patrón arquitectónico es quien define la estructura básica de la aplicación. (Camacho, E. et al. 2004)

En el desarrollo de la aplicación web se tomó como base el Modelo Vista Controlador (MVC). Pero al utilizar Django como framework de desarrollo web, este implementa el Modelo Vista Plantilla (MVT), el cual se encarga de separar la lógica del negocio de la interfaz del usuario (Siciliano 2022). A continuación, en la Figura 2.10 se muestra una descripción grafica del funcionamiento de este modelo.

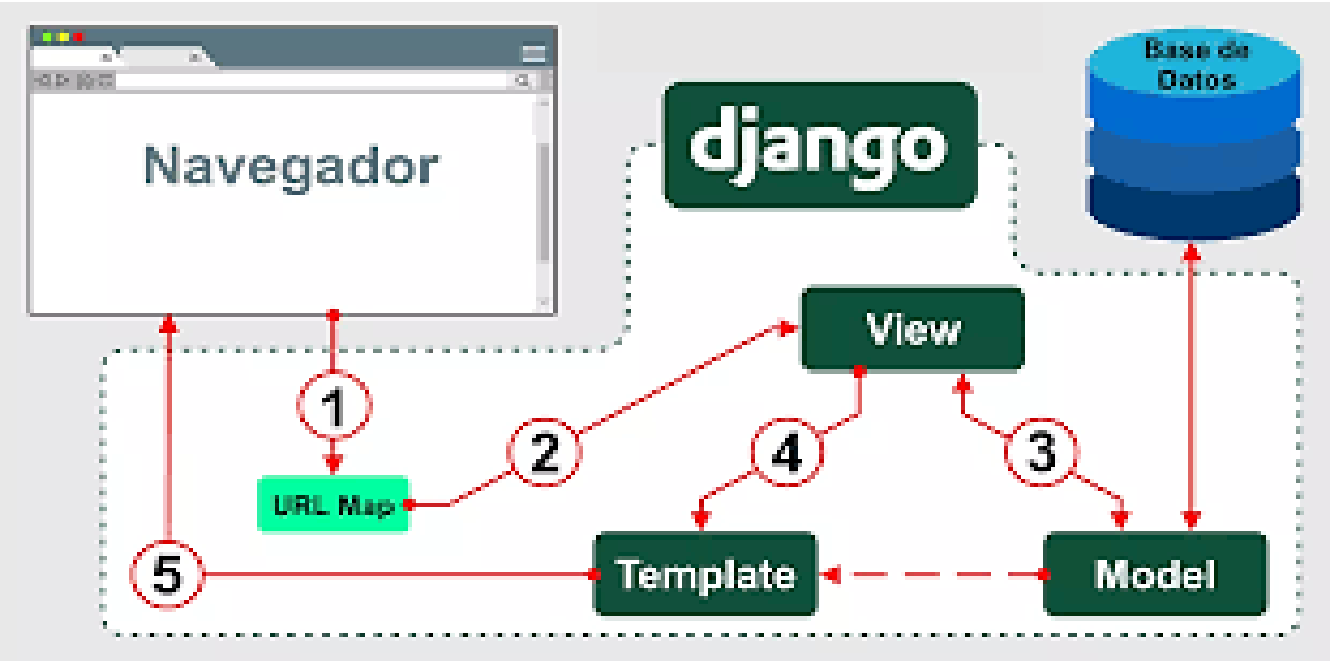


Figura 2. 10 Modelado de la solución propuesta

El MVT divide las aplicaciones en tres niveles de abstracción, los cuales se describen a continuación:

- **Modelo:** representa la información relacionada con el dominio de la aplicación, es decir, su lógica del negocio. Abstrae la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de base de datos utilizado. Para el componente una de las tareas más comunes es la persistencia y lectura de la base de datos (Chen 2018). En la solución implementada se utiliza la estructura propuesta por Django, la cual utiliza el mapeo de objeto relacional (ORM). Se entiende al modelo como clases de Python que heredan de *django.db.models.Model* y se cuenta con una poderosa API de acceso a base de datos.

- **Vista:** es la encargada de describir los datos que son presentados al usuario. Describe cuáles datos se pueden ver, y no precisamente cómo se muestran. MVT define la vista como una función de llamada de Python para un localizador de recursos uniforme (url), porque esa función de llamada describe cuáles datos son presentados. En Django, una vista describe cuales datos son presentados, pero una vista normalmente delega a una plantilla, la cual describe cómo son presentados los datos (Rubio 2017).

- **Plantilla:** es la encargada de presentar la forma en que se percibe visualmente la información, no necesariamente lo que se muestra, lo cual es delegado a la vista. Más específicamente, es recomendable separar el contenido de la presentación, donde la presentación en este caso son las plantillas. En Django una “vista” (*view*) describe cuales datos son presentados, y la plantilla describe cómo es presentada esa información.

No se incluye al controlador ya que este viene inmerso en el propio framework. De igual forma, aunque el controlador no se defina en las siglas; se encuentra presente y se encarga de realizar el papel de intermediario entre el modelo y las vistas. Además, maneja las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que sean mostrados al usuario. En el caso de Django, este nivel de abstracción probablemente sea el mismo marco de trabajo. La maquinaria que envía una solicitud a la vista apropiada, de acuerdo a la configuración de Django URL.

En la Figura 2.11 se muestra el diagrama de paquetes elaborado.

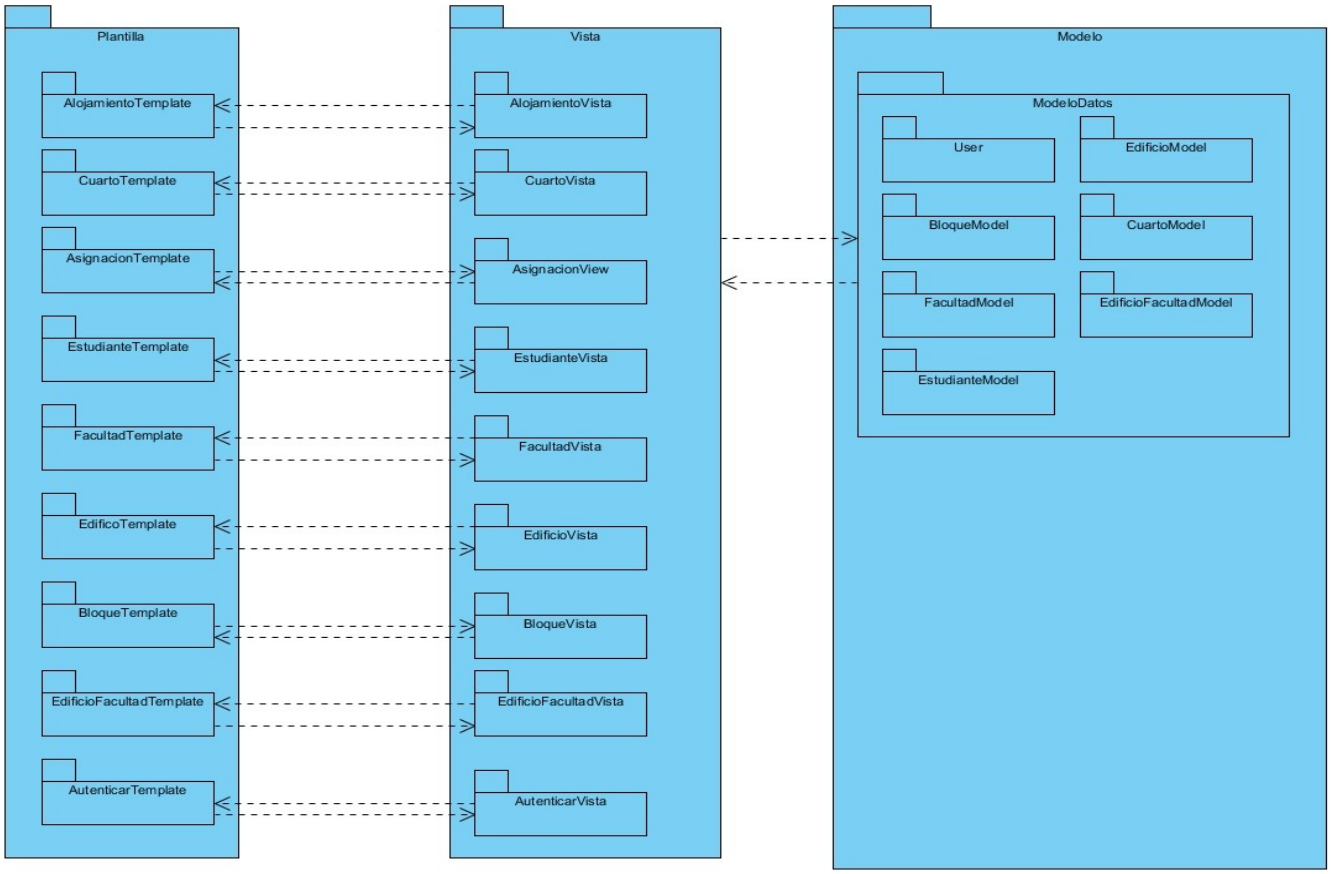


Figura 2. 11 Diagrama de paquetes. Fuente: Elaboración propia

1.2.2.3. Patrones de diseño

Para diseñar la herramienta se emplearon un conjunto de patrones de diseño que constituyen soluciones simples a problemas específicos y comunes del diseño orientado a objetos. A continuación, se describen los patrones *Gang of Four* (GOF), *General Responsibility Assignment Software Patterns* (GRASP), los patrones de diseño creacionales y los principios asociados:

Patrones de diseño GOF

- **Comando (*Command*):** se utilizó en la clase *HttpRequest*, para encapsular la petición como un objeto, con lo que además se facilitó la parametrización de los métodos.
- **Plantilla (*Template*):** se utilizó el lenguaje de plantillas propuesto por Django para la construcción de *templates* capaces de trabajar con la información de las vistas, evidenciándose en los módulos *views.py* y los subdirectorios bajo el nombre *templates*.

Patrones de diseño GRASP

- **Alta cohesión:** El patrón se evidencia en cada una de las clases de la aplicación, de tal forma que se elimina la sobrecarga de responsabilidades.
- **Bajo acoplamiento:** El patrón se aplicó a cada una de las clases diseñadas para la aplicación.
- **Experto:** En la aplicación web desarrollada se puede observar en los módulos *models.py*, que poseen funciones concretas de acuerdo con la información que gestionan.
- **Controlador:** En el framework de desarrollo web Django, el controlador es considerado el propio framework, ya que este gestiona las pautas y acciones que se realizan en el entorno. Es la maquinaria que envía peticiones a la vista apropiada, de acuerdo a la configuración de las rutas de Django.

Principios de diseño asociados

- ***Don't Repeat Yourself* (DRY):** se aplicó este principio en todo el desarrollo del código fuente, evidenciándose en los módulos *models.py* y *tests.py*.

1.3. Tarjetas CRC

Las tarjetas CRC son una herramienta que brinda soporte al equipo a la hora de identificar las clases que participan en el diseño de la aplicación. Permiten obtener las responsabilidades que debe cumplir cada clase y establece cómo colabora una clase con otras para cumplir con sus responsabilidades[CITATION Die02 \l 3082]. A continuación, se muestra en la Figura 2.12 la Tarjeta CRC Adicionar estudiante.

Adicionar Estudiante	
Responsabilidades	**Colaboradores**
Permite que el estudiante elabore su solicitud de beca estudiantil.	Student, Faculty, Room
Verifica que los datos introducidos sean correctos	Checks
Se realiza el envio de los datos	
Se notifica el estado del envio	

Figura 2. 12 Tarjeta CRC Adicionar Estudiante. Fuente: Elaboración propia

1.4. Descripción de la base de datos

La Figura 2.13 representa el modelo físico de la base de datos, en especial las tablas pertenecientes al módulo de alojamiento.

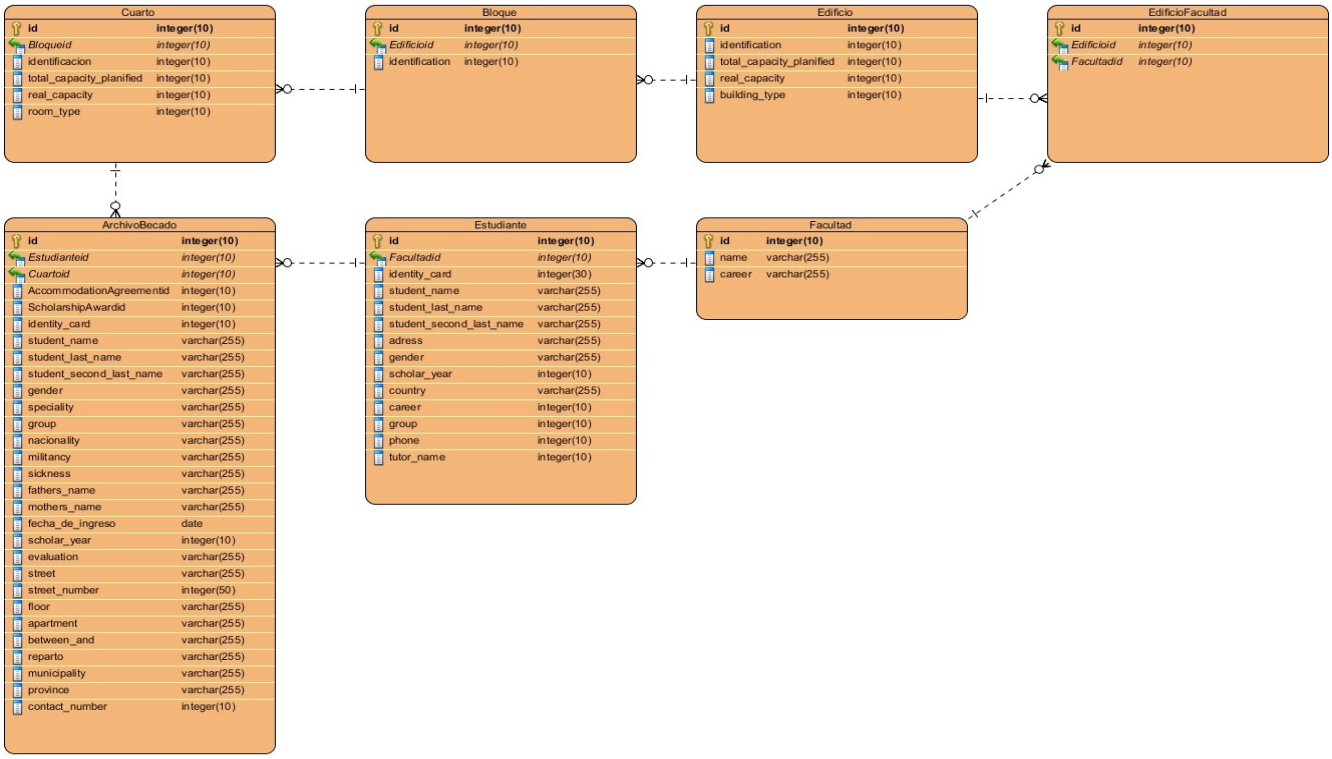


Figura 2. 13 Diagrama físico de la base de datos. Fuente: Elaboración propia

A continuación, se describe brevemente cada una de las tablas utilizadas para registrar la información asociada al módulo de alojamiento:

- **archivo_beca**: Registra la información del expediente del becado. Se registra el cuarto que se le asigna en beca.

- **estudiante**: Registra los datos de los estudiantes que inician el proceso para ser becados. Entre estos datos se encuentran: nombre, apellidos, facultad, sexo, dirección, así como otros campos necesarios para llevar a cabo los procesos de la Residencia Estudiantil. El estudiante pertenece a una facultad y se aloja en un cuarto de un edificio de la Residencia.

- **facultad**: Registra las facultades que existen en la Universidad.

- **edificio**: Registra la identificación del edificio, así como el campo que indica si el edificio es para que se alojen extranjeros o becados nacionales. Los edificios tienen bloques y pisos, donde están ubicados los cuartos.

- **edificio_facultad**: Registra los edificios donde se ubican los estudiantes becados de cada facultad.

- **bloque**: Registra el bloque con el edificio al que pertenece y su identificación, así como un campo para indicar si el bloque es para que se alojen hembras o varones.

- **cuarto**: Registra los cuartos con el piso al que pertenecen y la identificación de cada uno de estos. Los cuartos pueden ser de diferentes tipos. Además, almacenan la capacidad total de diseño y la capacidad real que está ocupada.

1.5. Conclusiones parciales

- Se precisaron los principales criterios que caracterizan el proceso Residencia Estudiantil de la CUJAE y las particularidades a tener en cuenta en la implementación de la aplicación web.

- El uso de la metodología XP ayudó a organizar la secuencia de actividades para la realización de las tareas propuestas, a partir de emplear los artefactos correspondientes, permitiendo la estructuración y descripción adecuadas para la aplicación desarrollada.

- El estilo arquitectónico Llamada – Retorno, el patrón arquitectónico Modelo Vista Plantilla y los patrones y principios de diseño utilizados, posibilitaron la estructura escalable de la aplicación y su naturaleza modular para la implementación de nuevas funcionalidades, así como la capacidad para gestionar las modificaciones surgidas en el desarrollo de la aplicación web.

- Se realizó el modelo físico de la base de datos que soporta la implementación de los requisitos de la aplicación y la persistencia de las entidades.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE SOFTWARE

En el presente capítulo se describen las fases de implementación y prueba de la propuesta de solución, teniendo en cuenta la metodología de desarrollo de software empleada. Se describen las pautas de codificación utilizadas. Finalmente se documenta el proceso de pruebas al que es sometida la aplicación web con el objetivo de comprobar el correcto funcionamiento de los requerimientos especificados.

2.1. Implementación

La implementación de un sistema informático es la parte del ciclo de vida del desarrollo de software donde se debe proceder a la generación del código correspondiente, someterlo a pruebas, crear la documentación pertinente y capacitar a los usuarios. En esta fase las HU se descomponen en tareas de programación o ingeniería, que a su vez son convertidas en código. Se tiene en cuenta el desarrollo de las iteraciones según el plan de iteraciones definido, la entrega de la aplicación en iteraciones pequeñas, y se busca llegar a un diseño simple y poco redundante del código que responda a las especificaciones y funcionalidades planteadas (Caldas 2003).

2.1.1. Tareas de ingeniería

Las HU se descomponen en tareas de programación o ingeniería (TI), que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser comprendidas por el cliente, pueden ser elaboradas en lenguaje técnico y son para el uso estricto de los programadores. En la Tabla 3.1 se describe la TI 25, que tiene como objetivo implementar la HU Adicionar estudiante:

Tabla 3. 1 Tarea de ingeniería 5

Tarea de ingeniería	
Número de tarea: 25	**Historia de Usuario (No.1):** Adicionar estudiante
Nombre de tarea: Implementar HU_ Adicionar estudiante	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2 días
Fecha de inicio: 10-10-2022	**Fecha de fin:** 12-10-2022
Programador responsable: Raikol Álvarez León	
Descripción: Permite adicionar un estudiante en la aplicación.	

2.2. Estándares de codificación

El código es la forma principal de documentación y comunicación entre los desarrolladores, por lo que debe estar escrito lo más claro posible, de manera que pueda ser identificado fácilmente por cualquier programador de otro equipo de desarrollo. Para la escritura del código fuente de la solución propuesta, el equipo de desarrollo utiliza los estándares de codificación definidos para el lenguaje de programación Python (PEP 8 – Style Guide for Python Code | peps.python.org 2022). A continuación, se muestran algunos de los utilizados en la implementación de la propuesta de solución:

- Los nombres de clase normalmente deben usar la convención *capWords*.
- Los nombres de las funciones se escribirán en minúsculas, y si son nombres compuestos, seguidos de guión bajo. Ejemplo: nombre1_nombre2.
- Utilizar 4 espacios por nivel de sangría.
- Todas las líneas limitadas a un máximo de 79 caracteres.
- El código en la distribución principal de Python siempre debe usar UTF-8 y no debe tener una declaración de codificación.
- Las importaciones deben realizarse en líneas separadas.
- Los comentarios deben ser oraciones completas y la primera palabra debe estar en mayúsculas.
- Se utiliza siempre para el primer argumento para instanciar *métodos.self*.

2.3. Técnicas de validación de los requisitos

Para desarrollar adecuadamente la aplicación web propuesta se hizo uso de las siguientes técnicas de validación de requisitos:

- **Revisiones formales de los requisitos:** se trabajó junto al cliente para la obtención de retroalimentación a partir de una serie de encuentros definidos en los que se analizaron de forma metódica los requisitos funcionales de la aplicación. Se enfatizó en el módulo de alojamiento como aspecto fundamental, y en el perfeccionamiento de los requisitos relacionados con el mismo. Se modificaron algunos aspectos y se añadieron funcionalidades para la obtención de un modelo funcional.

- **Generación de casos de prueba (CP):** como parte del proceso de validación de los requisitos funcionales, a partir de las HU elaboradas se realizaron un conjunto de CP.

- **Construcción de prototipos:** se confeccionaron prototipos no funcionales, dando la posibilidad al cliente de poder comprobar de forma visual cómo quedaría la aplicación web. Con el uso de esta técnica, cada uno de los prototipos definidos fueron aprobados por el cliente.

2.4. Pruebas de software

Las pruebas son un instrumento adecuado para determinar el estado de la calidad de un producto. Establecen la ejecución de un sistema o componente bajo condiciones o requerimientos especificados. Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de pruebas, el cual permite asegurar el éxito deseado al realizarse de manera continua, proporcionando la obtención de un producto de mayor calidad, pues los errores son detectados en un corto plazo de tiempo y se corrigen de una manera más sencilla. Existen diferentes tipos de pruebas, entre ellas se destacan las pruebas unitarias y las de aceptación.

2.4.1. Desarrollo dirigido por pruebas

El desarrollo dirigido por pruebas, es una técnica de diseño e implementación de software incluida dentro de la metodología XP. Consiste en guiar el desarrollo de una aplicación por medio de Test Unitarios; que son algoritmos que emulan lo que la aplicación debería hacer. Esta técnica permite saber qué, cómo, cuáles y cuántos algoritmos se necesitarán desarrollar para que la aplicación funcione como se espera y no tenga fallos imprevistos (Bahit 2012).
Una vez obtenidos resultados satisfactorios en las pruebas, se pasa a producción y se inicia el proceso de refactorización que consiste en limpiar y organizar el código, adaptarlo a los patrones y aumentar su legibilidad, sin modificar su comportamiento externo.

2.4.2. Niveles de prueba

Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes Niveles de Pruebas. (Jacobson 2000)

- Prueba de Unidad: Un componente es la unidad más pequeña especificada de un sistema. Las pruebas se llevan a cabo tras la construcción o realización de cada componente para verificar que la implementación se esté llevando conforme a los estándares acordados. El

objetivo es comprobar que el sistema, entendido como una unidad funcional, está correctamente codificado.

- Prueba 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.

- Prueba de Aceptación: Es la prueba final antes del despliegue del software. El propósito es confirmar que el software está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Para realizar la validación de la aplicación web, se desarrollaron las pruebas unitarias, las pruebas de sistema y las pruebas de aceptación.

2.4.3. Pruebas unitarias

Las pruebas unitarias de software, conocidas también como unit testing o test unitarios, pueden definirse como un mecanismo de comprobación del funcionamiento de las unidades de menor tamaño de un programa o aplicación en específico. Funcionan mediante el mecanismo de aislar una parte determinada del código fuente, con el objetivo de asegurar su funcionamiento; es decir, son pruebas pequeñas que validan el comportamiento de un fragmento del código.

Forman parte de la estrategia de metodología ágil del trabajo de desarrollo, donde se busca ofrecer piezas pequeñas de software en funcionamiento en un corto periodo de tiempo, con el objetivo de aumentar la satisfacción del cliente. Son de vital importancia para la detección de errores, ya que sin estas los errores no podrían identificarse, hasta fases más avanzadas del desarrollo. Esto implica que las pruebas unitarias de software evitan la escalada de errores en el código, al identificarlos de manera temprana.

Otra de las características de las pruebas unitarias de software es que usualmente se llevan a cabo como primera evaluación en la fase del desarrollo de las aplicaciones, para testear que todo marche en orden en la aplicación. Las ejecuciones de las pruebas unitarias de software traen ciertas ventajas, como, por ejemplo, la capacidad de demostrar que la lógica del código fuente de un programa o aplicación se encuentre en buen estado y funcionando de manera normal.

Testear una aplicación web es una tarea compleja, por estar conformada por varias capas de lógica, desde el manejo de solicitudes a nivel HTTP, hasta la validación y el procesamiento de formularios, y

la representación de plantillas. Sin embargo, con el marco de ejecución de pruebas de Django y los diferentes utilitarios, se pueden simular solicitudes, insertar datos de prueba, inspeccionar las salidas de la aplicación y, en general, verificar que el código esté haciendo lo que se espera.

Para la realización de pruebas se estudiaron varios módulos y frameworks de prueba correspondientes al lenguaje de programación Python, destacando los siguientes:

- Unittest: El marco de pruebas unitarias unittest se inspiró originalmente en JUnit y tiene un estilo similar al de los principales marcos de pruebas unitarias en otros lenguajes. Admite la automatización de las pruebas, la inicialización compartida, la compartición del código de configuración y cierre de las pruebas, la agregación de las pruebas en colecciones y la independencia de las pruebas del marco de informes y de la infraestructura que los reporta.

- PyTests: Pytest es un marco de pruebas de Python que se originó en el proyecto PyPy. Se puede utilizar para escribir varios tipos de pruebas de software, incluyendo pruebas unitarias, pruebas de integración, pruebas de extremo a extremo y pruebas funcionales. Sus características incluyen pruebas parametrizadas, accesorios y reescritura de afirmaciones.

Tomando en cuenta los elementos mencionados, se decide escoger para la realización de pruebas el framework unittest, Este viene integrado en el lenguaje de programación Python y presenta una interfaz estable. Las pruebas escritas requieren la importación de módulos y la instanciación de clases, lo que facilita la interpretación de las pruebas y los detalles de las mismas.

Además, plantea conceptos importantes como son: configuración de prueba (*fixture*), conjunto de pruebas, ejecutor de pruebas y caso de prueba, siendo este último donde se centra la atención en este trabajo.

Un caso de prueba es la unidad mínima de prueba. Verifica la respuesta específica a un juego particular de entradas unittest, proporcionando una clase base, TestCase, que se puede utilizar para crear nuevos casos de prueba.

Para la realización de estos casos de prueba, Python emplea las aserciones, las cuales son declaraciones que se pueden usar para establecer controles de lógica durante el proceso de desarrollo. Las aserciones permiten probar la corrección del código, al verificar si algunas condiciones específicas siguen siendo ciertas, lo que puede ser útil para su depuración. La condición de aserción siempre debe ser verdadera a menos que se tenga un error en el programa. Si la condición resulta ser falsa, la afirmación genera una excepción y finaliza la ejecución del programa.

A continuación, las figuras desde la 3.1 hasta la 3.5, muestran las diferentes clases, pruebas realizadas a la aplicación y resultados de estas.

```
Raikol
class TestUrls(SimpleTestCase):
    Raikol
    def test_block_list_url_is_resolved(self):
        url = reverse('alojamiento:block_list')
        print(resolve(url))
        self.assertEquals(resolve(url).func, block_list)

    Raikol
    def test_block_add_url_is_resolved(self):
        url = reverse('alojamiento:block_add')
        print(resolve(url))
        self.assertEquals(resolve(url).func, block_add)

    Raikol
    def test_student_list_url_is_resolved(self):
        url = reverse('alojamiento:student_list')
        print(resolve(url))
        self.assertEquals(resolve(url).func, student_list)

    Raikol
    def test_student_add_url_is_resolved(self):
        url = reverse('alojamiento:student_add')
        print(resolve(url))
        self.assertEquals(resolve(url).func, student_add)
```

Figura 3. 1 Pruebas unitarias de la clase TestUrls. Fuente: Elaboración propia

La clase *TestUrls* es la encargada de gestionar las rutas hacia las vistas y comprobar su comportamiento. La clase hereda de *SimpleTestCase*, la cual es una subclase de *unittest.TestCase*. Entre las funcionalidades a destacar de esta clase se encuentra el testeo robusto a fragmentos del código HTML para comprobar la equidad del código o el contenido.

Para la llamada y comprobación de los métodos *test_block_list_is_resolved* y *test_block_add_url_is_resolved*, contenidos en la clase TestUrls, se accedió desde el modulo *test.py* a las url diseñadas en el módulo *urls.py* y se hizo uso de la funcionalidad assert de Python para la comprobación de la equidad de las mismas. A continuación, en la Figura 3.2 se muestran las respuestas arrojadas por la llamada a los métodos descritos.

```
Found 8 test(s).
System check identified no issues (0 silenced).
ResolverMatch(func=alojamiento.views.block_add, args=(), kwargs={}, url_name='block_add', app_names=['alojamiento'], namespaces=['alojamiento'], rou
te='alojamiento/block/add/')
.ResolverMatch(func=alojamiento.views.block_list, args=(), kwargs={}, url_name='block_list', app_names=['alojamiento'], namespaces=['alojamiento'],
route='alojamiento/block/')
.ResolverMatch(func=alojamiento.views.building_add, args=(), kwargs={}, url_name='building_add', app_names=['alojamiento'], namespaces=['alojamiento
'], route='alojamiento/building/add/')
.ResolverMatch(func=alojamiento.views.building_list, args=(), kwargs={}, url_name='building_list', app_names=['alojamiento'], namespaces=['alojamien
to'], route='alojamiento/building/')
.ResolverMatch(func=alojamiento.views.room_add, args=(), kwargs={}, url_name='room_add', app_names=['alojamiento'], namespaces=['alojamiento'], rout
e='alojamiento/room/add/')
.ResolverMatch(func=alojamiento.views.room_list, args=(), kwargs={}, url_name='room_list', app_names=['alojamiento'], namespaces=['alojamiento'], ro
ute='alojamiento/room/')
.ResolverMatch(func=alojamiento.views.student_add, args=(), kwargs={}, url_name='student_add', app_names=['alojamiento'], namespaces=['alojamiento']
, route='alojamiento/student/add/')
.ResolverMatch(func=alojamiento.views.student_list, args=(), kwargs={}, url_name='student_list', app_names=['alojamiento'], namespaces=['alojamiento
'], route='alojamiento/student/')
.
----------------------------------------------------------------------
Ran 8 tests in 0.025s

OK
```

Figura 3. 2 Resultado de las pruebas unitarias realizadas a la clase TestUrls. Fuente: Elaboración propia

Para las pruebas realizadas a las vistas implementadas en la aplicación, las cuales se encuentran alojadas en el módulo views.py, se hizo uso del navegador Firefox versión 106.0.3 (64-bit), para la comprobación de la interfaz de usuario y el funcionamiento correcto de las posibles entradas de datos.

Para el levantamiento del navegador Firefox se utilizó la herramienta Selenium WebDriver, la cual acepta una serie de comandos y los envía a un navegador para la ejecución de ciertas tareas. Esto se implementa a través de un controlador específico para cada navegador, que envía los comandos y trae los resultados de regreso. En Selenium WebDriver no se requiere de un servidor especial para ejecutar las pruebas, en vez de ello WebDriver inicia una instancia del navegador y lo controla.

La Figura 3.3 muestra la clase *TestLoginClass*, la cual define tres funciones: *setUp*, *test_login_page* y *tearDown*. La primera es la encargada de levantar el navegador web especificado. La función *test_login_page* se encarga de acceder a una ruta especificada, en este caso se indica que acceda a la vista de autenticación, que introduzca un usuario y contraseña válidos, y que se autentique. Por último, la aplicación debe finalizar el proceso ejecutado.

```python
class TestLoginClass(unittest.TestCase):
    path = webdriver.Firefox(firefox_binary=r'C:\Program Files\Mozilla Firefox\firefox.exe')

    new *
    def setUp(self):
        self.driver = TestLoginClass.path

    new *
    def test_login_page(self):
        driver = TestLoginClass.path
        driver.get("http://127.0.0.1:8080/")
        self.assertIn("Autenticar", driver.title)
        elem = driver.find_element(By.NAME, "username")
        elem.clear()
        elem = driver.find_element(By.NAME, "password")
        elem.clear()
        username_input = driver.find_element(By.NAME, "username")
        username_input.send_keys('username')
        password_input = driver.find_element(By.NAME, "password")
        password_input.send_keys('password')
        username_input.send_keys(Keys.RETURN)
        password_input.send_keys(Keys.RETURN)
        driver.find_element(By.XPATH, '//input[@value="Autenticarse"]').click()
        self.assertNotIn("No results found.", driver.page_source)

    new *
    def tearDown(self):
        self.driver.close()
```

Figura 3. 3 Pruebas unitarias de la clase TestLoginClass. Fuente: Elaboración propia

La Figura 3.4 muestra la respuesta visual de la clase *TestLoginClass* obtenida en el navegador Firefox.

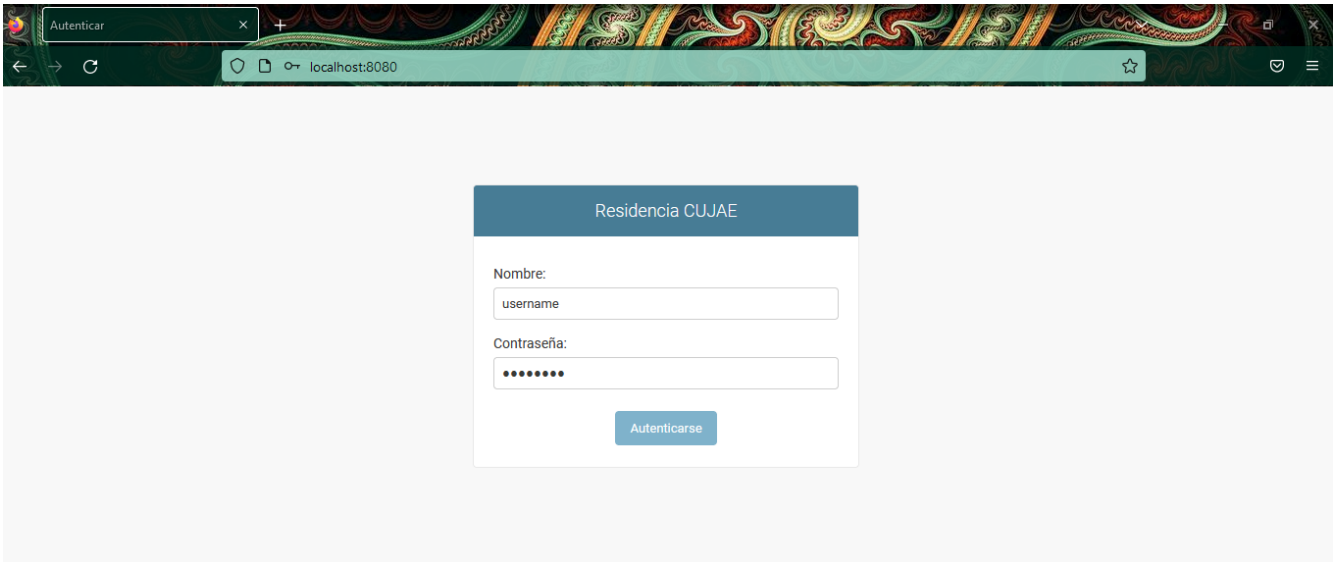


Figura 3. 4 Respuesta por navegador de la clase TestLoginClass del módulo test.py. Fuente: Elaboración propia

A continuación, se muestra en la Figura 3.5 la respuesta de la clase *TestLoginClass* obtenida por consola.

```
Found 1 test(s).
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 3.909s

OK
```

Figura 3. 5 Respuesta por consola de la clase TestLoginClass del módulo test.py. Fuente: Elaboración propia

2.4.4. Pruebas de sistema

Las pruebas de sistema es una serie de diferentes pruebas cuyo propósito principal es ejercitar por completo el sistema basado en computadora. Aunque cada prueba tenga un propósito diferente, todo el sistema funciona para verificar que los elementos se hayan integrado de manera adecuada y que se realizan las funciones asignadas. (Pressman 2010)

Entre las técnicas de pruebas que se realizan a la solución propuesta está la que evalúa la funcionalidad de esta. Según los parámetros de evaluación que abarca la técnica, se pueden distinguir distintos tipos de pruebas:

- Pruebas funcionales
- Pruebas de seguridad
- Pruebas de rendimiento

Estos tipos de pruebas se enfocan en validar la correcta implementación de las necesidades del cliente. La funcionalidad puede ser vinculada a los datos de entrada y de salida. Los datos de entrada serán ejecutados y mostrarán un resultado y dicho resultado será comparado con el resultado esperado (comportamiento) **(Campos Chiu 2015)**.

Pruebas funcionales

Las pruebas funcionales o de función están enfocadas en los requisitos funcionales y las reglas del negocio. Estas pruebas utilizan el método de Caja Negra.

El método de caja negra se centra en los requisitos funcionales del software. Es decir, permite al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. El método de caja negra no es una opción frente a caja

blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de describir una clase diferente de errores de los que se identifican con los métodos de caja blanca [CITATION MarcadorDePosición2 \l 21514].

Para la realización de las pruebas funcionales en Python, se utilizó las librerías disponibles en el propio *framework* Django, las cuales simulan el proceso que realizaría un usuario. En la Figura 3.6 se muestran los resultados de estas pruebas.

```
Found 8 test(s).
System check identified no issues (0 silenced).
ResolverMatch(func=alojamiento.views.block_add, args=(), kwargs={}, url_name='block_add', app_names=['alojamiento'], namespaces=['alojamiento'], rou
te='alojamiento/block/add/')
.ResolverMatch(func=alojamiento.views.block_list, args=(), kwargs={}, url_name='block_list', app_names=['alojamiento'], namespaces=['alojamiento'],
route='alojamiento/block/')
.ResolverMatch(func=alojamiento.views.building_add, args=(), kwargs={}, url_name='building_add', app_names=['alojamiento'], namespaces=['alojamiento
'], route='alojamiento/building/add/')
.ResolverMatch(func=alojamiento.views.building_list, args=(), kwargs={}, url_name='building_list', app_names=['alojamiento'], namespaces=['alojamien
to'], route='alojamiento/building/')
.ResolverMatch(func=alojamiento.views.room_add, args=(), kwargs={}, url_name='room_add', app_names=['alojamiento'], namespaces=['alojamiento'], rout
e='alojamiento/room/add/')
.ResolverMatch(func=alojamiento.views.room_list, args=(), kwargs={}, url_name='room_list', app_names=['alojamiento'], namespaces=['alojamiento'], ro
ute='alojamiento/room/')
.ResolverMatch(func=alojamiento.views.student_add, args=(), kwargs={}, url_name='student_add', app_names=['alojamiento'], namespaces=['alojamiento']
, route='alojamiento/student/add/')
.ResolverMatch(func=alojamiento.views.student_list, args=(), kwargs={}, url_name='student_list', app_names=['alojamiento'], namespaces=['alojamiento
'], route='alojamiento/student/')
.
----------------------------------------------------------------------
Ran 8 tests in 0.025s

OK
```

Figura 3. 6 Respuesta por consola de las pruebas funcionales del módulo test.py. Fuente: Elaboración propia

Pruebas de seguridad

La prueba de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema, en realidad lo protegerán de cualquier penetración impropia. Durante la prueba, quien realiza la prueba juega el papel del individuo que desea penetrar al sistema. Quien realice la prueba puede intentar adquirir contraseñas por medios administrativos externos; puede atacar el sistema con software a la medida diseñado para romper cualquier defensa que se haya construido; puede abrumar al sistema, y por tanto negar el servicio a los demás; puede causar a propósito errores del sistema con la esperanza de penetrar durante la recuperación; puede navegar a través de datos inseguros para encontrar la llave de la entrada al sistema (Pressman 2010).

En la implementación de la aplicación web se utilizó la protección *Cross Site Request Forgery* (CSRF) la cual evita que un usuario ejecute acciones usando las credenciales de otro usuario sin su consentimiento. La protección CSRF funciona comprobando una clave en cada solicitud POST. Esto asegura que un usuario malintencionado no pueda reproducir un formulario POST en su sitio web y que otro usuario que haya iniciado sesión, sin darse cuenta envíe ese formulario.

Pruebas de rendimiento

Las pruebas de rendimiento se diseñan para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. Esto ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema (Pressman 2010).

Como pruebas de rendimiento se consideró conveniente establecer la comprobación de las pruebas del módulo alojamiento en tiempo real. Se pueden observar los resultados en la Figura 3.7, que se muestra a continuación.

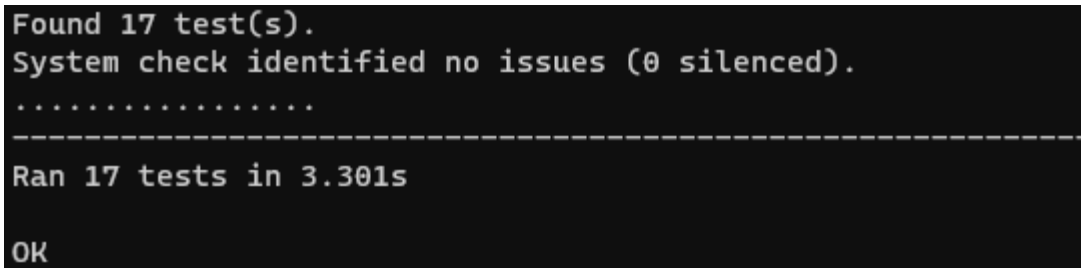


Figura 3. 7 Respuesta por consola del módulo test.py. Fuente: Elaboración propia

2.4.5. Pruebas de aceptación

Las pruebas de aceptación constituyen una técnica para garantizar que los requisitos hayan sido cumplidos y que el sistema es realmente lo que el cliente necesita, además de asegurar su correcto funcionamiento. Estas son creadas a partir de las HU y desde la perspectiva del cliente. El objetivo final es lograr que los requerimientos sean cumplidos y que el sistema sea aceptable. Tal es así, que una vez que todas las HU hayan pasado sus pruebas de aceptación, se considera entonces terminado el sistema. Seguidamente, la Tabla 3.2 muestra el caso de prueba de aceptación de la HU "Adicionar estudiante".

Tabla 3. 2 Caso de prueba de Aceptación de la HU "Adicionar estudiante"

	Caso de prueba de aceptación
Código de caso de prueba: 25	1. **Nombre historia de usuario:** Adicionar estudiante
Nombre de la persona que realiza el caso de prueba: Raikol Álvarez León	
Descripción de la prueba: revisar a través de la aplicación el correcto funcionamiento del RF *Adicionar*	

Estudiante.		
Condiciones de ejecución: todos los campos deben ser rellenados con datos válidos.		
Entrada/Pasos de ejecución		
Acción:	**Entrada:**	**Resultados esperados:**
Se selecciona la opción: “Adicionar”	Datos requeridos	La aplicación debe mostrar un listado de los estudiantes adicionados.
Evaluación de prueba: Satisfactoria		

2.5. Análisis de los resultados obtenidos de las pruebas de software

Se realizaron varios encuentros con los funcionarios de la Residencia Estudiantil de la CUJAE, entre ellos la directora de residencia Ms. C. Dianelys Tejeda Villazón, con el objetivo de revisar el sistema. Los resultados de estos encuentros fueron satisfactorios y cumplieron con los requisitos expuestos en este informe.

Las pruebas realizadas a la aplicación contribuyeron a la comprobación del correcto funcionamiento de las funcionalidades implementadas, identificando errores en el código en las iteraciones de las fases tempranas de su implementación, permitiendo su rectificación y posibilitando al ahorro de tiempo. Se muestra en la Figura 3.8 una representación gráfica de la cantidad de Inconformidades Significativas (IS), las cuales son errores en las funcionalidades implementadas, y la cantidad de Inconformidades No Significativas(INS), errores en la interfaz de la aplicación, el idioma o la navegabilidad, halladas en las iteraciones tempranas de las pruebas unitarias realizadas a la aplicación.

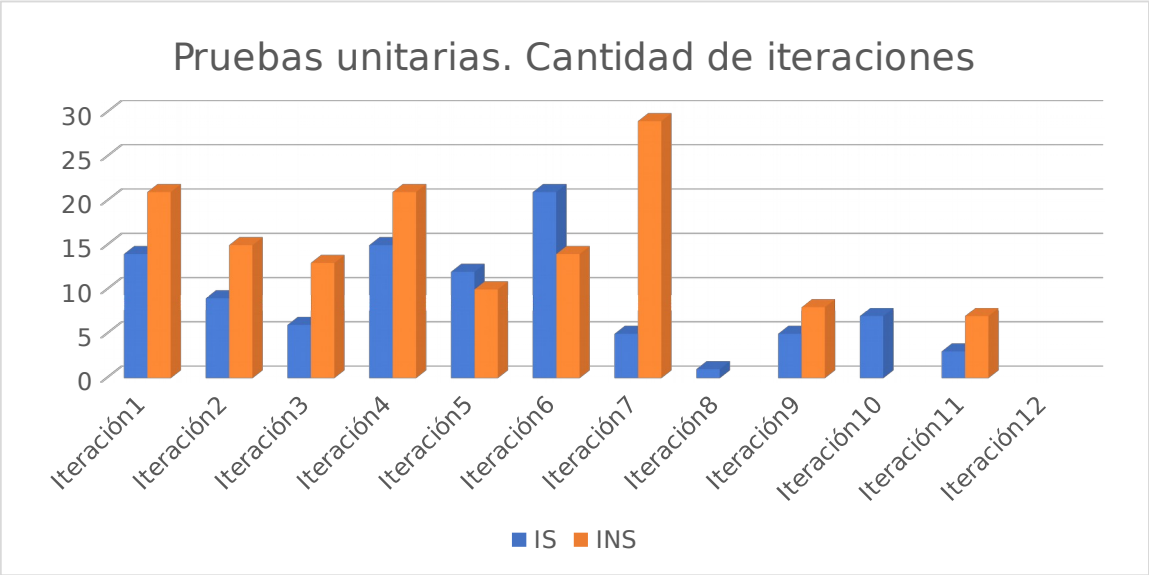


Figura 3.8 Iteraciones iniciales de las pruebas unitarias. Fuente: Elaboración propia

Como se puede apreciar, en las primeras 7 iteraciones se obtuvo un número considerable de IS, las cuales tuvieron que ser depuradas y corregidas. Para ello fue importante la retroalimentación con el cliente, que permitió la verificación del correcto funcionamiento de la aplicación. Los resultados en las iteraciones tuvieron un comportamiento no lineal, pues fluctuaron de una iteración a otra. Las INS presentaron un comportamiento similar, mostrando un mayor número de errores, debido a errores encontrados en el diseño de la interfaz, así como por el uso de diferentes idiomas en la aplicación. Las pruebas realizadas a la aplicación, además de la prevención de errores y el ahorro de tiempo, posibilitaron la obtención de un producto más completo.

Entre los beneficios de la implementación de las pruebas se encontraron:

- El acceso instantáneo y seguro a la información de los estudiantes becados.
- El soporte que brinda el sistema de autenticación de usuarios y la gestión de los permisos de los mismos.
- La gestión de la infraestructura de edificios, bloques y cuartos para el control del estado actual de los mismos.
- La posibilidad de importación de archivos con extensión .xls para el trabajo con bases de datos externas.

- La celeridad en la realización de tareas habituales en la gestión del alojamiento de la Residencia Estudiantil.

2.6. Conclusiones parciales

- La aplicación de las técnicas de validación de requisitos: construcción de prototipos, revisiones formales de los requisitos y generación de casos de prueba, permitieron obtener requerimientos entendibles y acordes a las necesidades del cliente, avalados por el acta de aceptación de requisitos obtenida al final del proceso.

- Estas técnicas facilitaron además el avance en la implementación de la aplicación web y la retroalimentación con el cliente, obteniendo peticiones más exactas sobre las funcionalidades requeridas.

- La realización de pruebas unitarias y de aceptación propiciaron el uso de buenas prácticas profesionales, además de la obtención de errores de forma automática, contribuyendo a la celeridad del desarrollo de la solución propuesta.

CONCLUSIONES

1. En la revisión bibliográfica realizada se corroboró la factibilidad de desarrollar una aplicación web, con carácter modular y capacidades para garantizar la interoperabilidad, como solución para la informatización del proceso de gestión del alojamiento en la Residencia Estudiantil de la CUJAE.

2. El empleo de técnicas de captura de requisitos, patrones de diseño y arquitectónicos, así como el uso de estándares de codificación en la implementación de la solución propuesta, permitió obtener una aplicación web flexible al mantenimiento y a la introducción de cambios.

3. Se alcanzó a partir de la implementación de pruebas a la aplicación web una solución robusta, que da soporte al proceso Residencia Estudiantil de la CUJAE, específicamente en la gestión del alojamiento.

Se pudo constatar que la aplicación web desarrollada introdujo mejoras significativas en el trabajo del área al garantizar mayor rapidez, veracidad y actualización constante de la información que se maneja. Todo lo anterior permite afirmar que fueron cumplidos los objetivos planteados al inicio del presente trabajo de tesis.

RECOMENDACIONES

- Continuar el trabajo realizado con el objetivo de añadir nuevas funcionalidades para la gestión de la totalidad de los subprocesos del proceso Residencia Estudiantil de la CUJAE.
- Extender el alcance de la aplicación web para incorporar los aspectos del trabajo educativo en la Residencia Estudiantil con el propósito de obtener la caracterización del becado, integrando funcionalidades de notificación de tareas y deberes de los estudiantes becados.
- Incorporar un modelo matemático, que tomando en cuenta todas las variables que intervienen en el proceso de asignación de capacidades de alojamiento, brinde soporte a los usuarios para la toma de decisiones, tanto en el momento de la matrícula, como ante la ocurrencia de eventualidades que se puedan presentar en el trabajo diario.
- Completar la documentación para que sirva como base para la continuidad del desarrollo de las nuevas funcionalidades que se implementen.

BIBLIOGRAFÍA

About SQLite. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://www.sqlite.org/about.html.

About Us, WebRezPro. *About Us, WebRezPro* [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://webrezpro.com/about-us/.

AL-SAQQA, S., SAWALHA, S. y ABDEL-NABI, H., 2020. Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, pp. 246. DOI 10.3991/ijim.v14i11.13269.

ASMO, 2018. Agile Methodology: An Overview. *Zenkit* [en línea]. [Consulta: 3 noviembre 2022]. Disponible en: https://zenkit.com/en/blog/agile-methodology-an-overview/.

AUP, 2015. AUP Ingeniería de Software. .

AZMAT, M. y KUMMER, S., 2020. Potential applications of unmanned ground and aerial vehicles to mitigate challenges of transport and logistics-related critical success factors in the humanitarian supply chain. *Asian Journal of Sustainability and Social Responsibility*, vol. 5, no. 1, pp. 3. ISSN 2365-6417. DOI 10.1186/s41180-020-0033-7.

BAHIT, E., 2012. *Scrum y eXtreme Programming para programadores*. S.l.: Autoedición.

BAQUERO HERNÁNDEZ, L.R. y MENDOZA PEÑA, D., 2016. Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso. ,

CALDAS, M.P., 2003. Management information systems: managing the digital firm. *Revista de Administração Contemporânea*, vol. 7, no. 1, pp. 223-223. ISSN 1982-7849, 1415-6555. DOI 10.1590/S1415-65552003000100014.

CAMACHO, E., JEFFRIES, R., ANDERSON, A., HENDRICKSON, C., CARDESO, F., y NUÑEZ, G., 2004. *Arquitecturas de Software-Guia de Estudio*. S.l.: Apr-2004.ming Installed". Addison-Wesley.

CAMPOS CHIU, C., 2015. Las pruebas en el desarrollo de software. En: Accepted: 2015-06-05T00:19:43Z [en línea], [Consulta: 4 octubre 2022]. Disponible en: http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/7627.

CHEN, S., 2018. Understanding of the management information system based on MVC pattern. *ADVANCES IN MATERIALS, MACHINERY, ELECTRONICS II: Proceedings of the 2nd International Conference on Advances in Materials, Machinery, Electronics (AMME 2018)* [en línea]. Xi'an City, China: s.n., pp. 040014. [Consulta: 3 noviembre 2022]. DOI 10.1063/1.5033678. Disponible en: http://aip.scitation.org/doi/abs/10.1063/1.5033678.

Design Patterns.pdf, [sin fecha]. S.l.: s.n.

Django: The web framework for perfectionists with deadlines. *Django: The web framework for perfectionists with deadlines*, 2022.

DR. MODESTO RICARDO GÓMEZ CRESPO, Ms.J.B.G. y VILLAZÓN, L.D.T., 2015. Caracterización sociocultural del empleo culto y sano del tiempo libre en la. ,

ECURED CONTRIBUTORS, 2019. Sistema informático - EcuRed. *Sistema informático* [en línea]. [Consulta: 19 noviembre 2022]. Disponible en: https://www.ecured.cu/Sistema_inform%C3%A1tico.

GAUCHAT, J.D., 2012. *El gran libro de HTML5, CSS3 y Javascript*. Barcelona: s.n.

GreenStudent » Software para residencias de estudiantes. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://www.greensoft.es/software-para-residencias-de-estudiantes/.

HIPP, D.R., 2022. SQLite Home Page. *SQLite Home Page*.

HORRUITINER-SILVA, P., 2006. El proceso de formación en la universidad cubana. En: Series Editors: _:n153Series Editors: _:n273Series Editors: _:n48Series Editors: _:n45, *El proceso de formación en la universidad cubana.* La Habana: s.n.,

How to deliver successful IT projects using MSF team model and MSF process model. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://www.pmi.org/learning/library/deliver-project-microsoft-solutions-framework-7413.

JACOBSON, I., 2000. *Uml El proceso unificado de desarrollo de software*. S.l.: s.n. ISBN 978-84-7829-036-9.

JACOBSON, I., BOOCH, G. y RUMBAUGH, J., 2000. *El proceso unificado de desarrollo de software*. S.l.: s.n.

JetBrains: Essential tools for software developers and teams. *JetBrains: Essential tools for software developers and teams*, 2022.

LAZO FERNÁNDEZ, Y., GARCÍA GONZÁLEZ, M. y GARCÍA RODRÍGUEZ, A., 2017. La labor educativa en la educación superior cubana desde la extensión universitaria: tendencias en las residencias estudiantiles. *Revista Cubana de Educación Superior*, vol. 36, no. 2, pp. 169-181. ISSN 0257-4314.

LETELIER TORRES, P. y HILARIO CANÓS, J., 2003. *Metodologías Ágiles en el Desarrollo de Software*. S.l.: s.n.

MANCHEGO PEÑA, F.A., 2019. UNIVERSIDAD JOSÉ CARLOS MARIÁTEGUI. , pp. 75.

MASUDA, S., HAGAR, J., NISHI, Y. y SUZUKI, K., 2022. Software Test Architecture Definition by Analogy with Software Architecture. *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. S.l.: IEEE, pp. 244-247. ISBN 1-66549-628-2.

MOLPECERES, A., 2002. Procesos de desarrollo: RUP, XP y FDD. *Recuperado de https://uvirtual. unet. edu. ve/pluginfile. php/270296/mod_resource/content/1/cualxpfddrup. PDF*,

PASCUAL, J.R., 2019. Capas y niveles: Diseño y confusión. *Disrupción Tecnológica* [en línea]. [Consulta: 21 septiembre 2022]. Disponible en: https://www.disrupciontecnologica.com/capas-y-niveles-diseno-y-confusion/.

PEÑO, J.M.S., 2015. Pruebas de Software. Fundamentos y Técnicas. , pp. 130.

PEP 8 – Style Guide for Python Code | peps.python.org. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://peps.python.org/pep-0008/.

PEREZ-CASTILLO, R., 2022. UML includes conceptual aspects such as business... - Google Académico. [en línea]. [Consulta: 3 noviembre 2022]. Disponible en: https://scholar.google.es/scholar?
start=0&q=UML+includes+conceptual+aspects+such+as+business+processes+and+system+functions,
+as+well+as+concrete+aspects+such+as+programming+language+expressions,
+database+schemas,
+and+reusable+software+components.&hl=es&as_sdt=0,5&as_ylo=2018.

PRESSMAN, R.S., 2003. *Ingeniería del Software. Un enfoque práctico. Sexta Edición*. S.l.: Mc Graw Hill. ISBN 970-10-5473-3.

PRESSMAN, R.S., 2010. *Ingeniería del software un enfoque práctico*. 7a.ed. S.l.: McGraw-Hill. ISBN 978-607-15-0314-5.

Rational Unified Process (RUP) - apppm. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: http://apppm.man.dtu.dk/index.php/Rational_Unified_Process_(RUP).

RICARDO, M., CRESPO, M.R., CARLOS, R., GARCÍA, BLANCO-GONZÁLEZ, J., DIANELYS, L., TEJEDA VILLAZÓN, D., ASISTENTE, P., ESTUDIANTIL, R., y CUJAE, 2018. *THE SOCIO-CULTURAL PROJECT IN THE STUDENTS INTEGRAL FORMATION AT CUJAE*. S.l.: s.n.

ROMERO, G.M.P., 2008. *MA-GMPR-UR2 Metodología ágil para proyectos de software libre. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas*. S.l.: s.n.

RUBIO, D., 2017. Django Model Forms and Class Views. En: D. RUBIO (ed.), *Beginning Django: Web Application Development and Deployment with Python* [en línea]. Berkeley, CA: Apress, pp. 403-439. ISBN 978-1-4842-2787-9. Disponible en: https://doi.org/10.1007/978-1-4842-2787-9_9.

SICILIANO, F., 2022. Re-engineering a digital ecosystem back-end: modeling and prototyping of a REST API architecture. ,

SIREGAR, S.A., IRMAYANI, D. y DAR, M.H., 2021. IMPLEMENTATION OF THE RUP METHOD ON THE LABUHAN BATU UNIVERSITY STUDENT ACTIVITY UNIT INFORMATION SYSTEM. , vol. 9, no. 2, pp. 7.

Software Engineering | Extreme Programming (XP) - GeeksforGeeks. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://www.geeksforgeeks.org/software-engineering-extreme-programming-xp/.

SOMMERVILLE, I., 2015. *SOFTWARE ENGINEERING*. S.l.: s.n. ISBN ISBN 10: 0-13-703515-2.

STOFF, M., 2021. Concept Map Mining as Browser Extension. , pp. 65.

TIA, T.K., 2019. SIMULATION MODEL FOR RATIONAL UNIFIED PROCESS (RUP) SOFTWARE DEVELOPMENT LIFE CYCLE. *SISTEMASI*, vol. 8, no. 1, pp. 176. ISSN 2540-9719, 2302-8149. DOI 10.32520/stmsi.v8i1.420.

VESCHI, B., 2022. Etimología de Gestión — Origen de la Palabra. *Etimología de Gestión — Origen de la Palabra*.

Visual Studio Code - Code Editing. Redefined. [en línea], 2022. [Consulta: 3 noviembre 2022]. Disponible en: https://code.visualstudio.com/.

Welcome to Python.org. *Python.org* [en línea], 2022a. [Consulta: 3 noviembre 2022]. Disponible en: https://www.python.org/.

Welcome to Python.org. *Python.org* [en línea], 2022b. [Consulta: 3 noviembre 2022]. Disponible en: https://www.python.org/.

XAVIER ALBALADEJO, 2022. Qué es SCRUM/ proyectoÁgiles. *ProyectosAgiles. org*.

ZYKOV, S., ALEXANDER, G. y KAZANTSEV, N., 2018. *Software Engineering for Enterprise System Agility: Emerging Research and Opportunities: Emerging Research and Opportunities*. S.l.: IGI Global. ISBN 978-1-5225-5590-2.

ANEXOS

Anexo1: Carta de aceptación:

cujae

Dirección de Alojamiento

Universidad Tecnológica de la Habana
"José Antonio Echeverría" Cujae
Calle 114, No.11901, e/ Ciclovía y Rotonda
Marianao, La Habana, Cuba. CP19390
http://cujae.edu.cu
Email: dir.alojamiento@tesla.cujae.edu.cu

En cumplimiento del desarrollo del trabajo de diploma: Aplicación web para la gestión del alojamiento en la Residencia Estudiantil de la CUJAE, se hace constancia de la aceptación por la parte del cliente de la solución que se relaciona a continuación:

- Ha sido implementada y probada la primera versión de aplicación web para gestionar la asignación de capacidades de alojamiento en la Residencia Estudiantil de la CUJAE.

Entrega:

Nombre y Apellidos: Raikol Álvarez León
Cargo: Tesista
Firma: _____

Recibe:

Nombre y Apellidos: Dr. Ing Modesto R. Gómez Crespo
Cargo: Rector, CUJAE
Firma: _____

Nombre y Apellidos: MS. c. Dianelys Tejeda Villazón
Cargo: Directora de Residencia Estudiantil, CUJAE
Firma: _____

Dirección cujae Alojamiento