FACULTAD 2

Sistema informático para la gestión de procesos en la Residencia Estudiantil de la Facultad 2.

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

Autores: Javier Alejandro Paret Estrada.
Héctor Eduardo Pérez Acosta.
Tutor: MSc. Darling Darias Pérez.

La Habana, diciembre de 2023.

Año 65 de la Revolución.

DECLARACIÓN DE AUTORÍA

Los autores del trabajo de diploma con título: "**Sistema informático para la gestión de procesos en la Residencia Estudiantil de la Facultad 2"**, conceden a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declaran únicos autores de este trabajo. Para que así conste firman la presente a los 7 días del mes de diciembre del año 2023.

Javier Alejandro Paret Estrada

_____
Firma del Autor

Héctor Eduardo Pérez Acosta

_____
Firma del Autor

MSc. Darling Darias Pérez

_____
Firma del Tutor

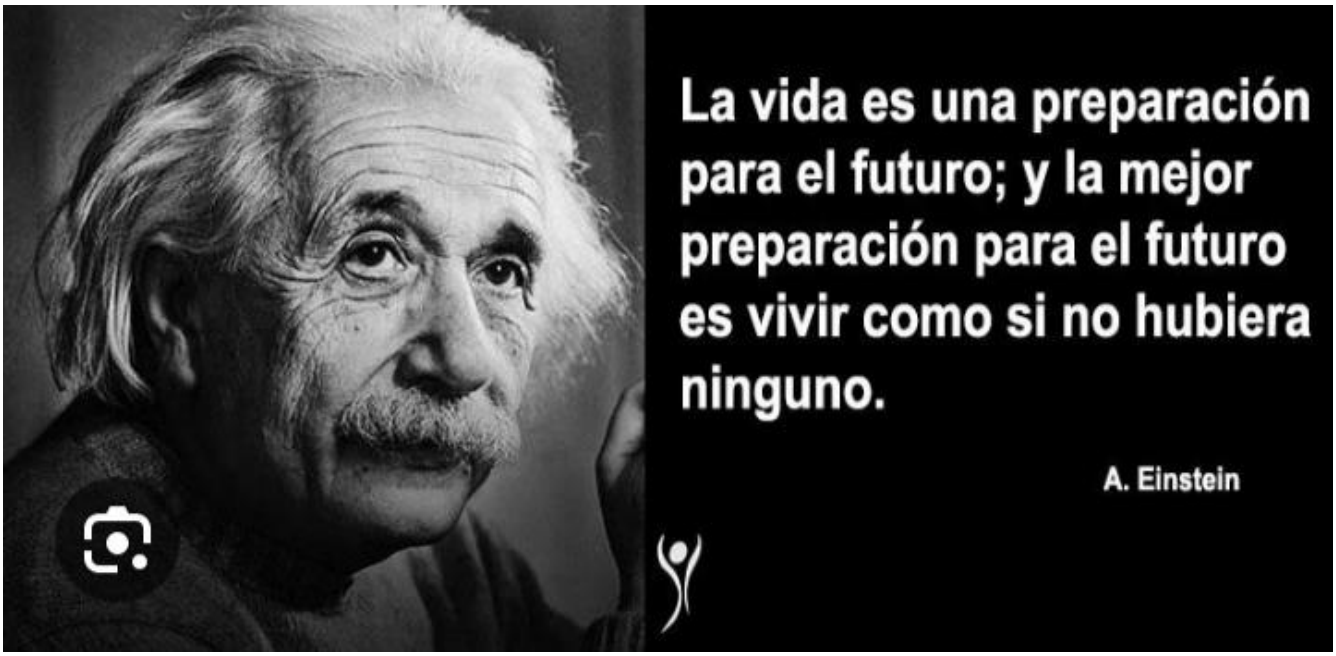
DATOS DE CONTACTO

Graduada de Ingeniera en Ciencias Informáticas. Profesora Principal de 5to año en la carrera de ICI de la Facultad 2. Posee 15 años de experiencia laboral impartiendo clases de las asignaturas de Ingeniería de Software, Gestión de Software, Base de datos, Proyecto de Investigación y Desarrollo e Introducción a las Ciencias Informáticas. De ellos, 6 años como directiva en un Centro y luego en la facultad. (subdirectora del Centro de Informática Médica: CECIM y Vicedecana de la Facultad 7) Además de tener experiencia en el rol de Analista y probadora de la calidad en dicho centro productivo. Realizó y defendió la maestría externa de Informática en Salud, en el Centro de Cibernética Aplicada a las Matemáticas. Ha participado en eventos científicos y grupos investigativos.

Correo electrónico: ddarias@uci.cu.

Teléfono: 53103636.

Frase


La vida es una preparación para el futuro; y la mejor preparación para el futuro es vivir como si no hubiera ninguno.
A. Einstein

AGRADECIMIENTOS

Héctor

Quisiera comenzar agradeciendo a mi madre, mi mayor apoyo, mi fortaleza, mi guía, la mujer que más amo y amare por el resto de mis días, la que ha sabido estar en todos los momentos durante mi trayectoria como estudiante, este logro lleva tu nombre mucho antes que el mío. A mi abuelo por ser mi fuente de conocimientos por aportar valores de alto valor a mi personalidad, por ser mi modelo a seguir como padre como hombre. Karlita, mi otra mitad, mi soporte, confidente y cómplice, gracias por confiar en mí, por celebrar mis logros y compartir mis penas. Toda mi familia, que siempre ha estado al tanto de mis esfuerzos y como recompensación también estarán en mis logros. Mi otra familia, la que creamos en estos 5 años de convivencia, mis amigos Eddy, Daidelín, Oscar y los que no estudiaron conmigo pero siempre me han acompañado desde que nos conocimos, David, Andy, Leo, en fin. No puede faltar nuestra tutora que a peasr de ser una mujer con muchas tareas, responsabilidades y muy poco tiempo ha apartado un tiempo para ayudarnos en la culminación de nuestra carrera, muchas gracias a todos.

Javier

Yo quisiera agradecer primero que todo a mi familia por su apoyo y presencia incondicional a lo largo de todos estos años, gracias a mis bisabuelos, a mi abuela Rebeca que siempre van a estar conmigo. Agradecer a Dios .Agradecer a estos personajes que conocí aquí en la escuela el Luiso, Guido, el Riki, mi chamaco David, Abraham, Yadi. Gracias por ese pikete del 146103 por todos los buenos ratos que pasamos. Doy las gracias a todas las personas con las que compartí en las fiestas, buenas locuras hice. Gracias al jurado el profe Antonio que me impartió media carrera , la profe Rosa ese pulover ella sabe gracias también, agradecer a la profe Yaneisi ,gracias a la tutora por su guía y recomendaciones,gracias al oponente.Gracias a la Universidad de Ciencias Informáticas por todo lo que me enseño me voy siendo mejor persona.

RESUMEN

La presente investigación se centra en el desarrollo de un sistema informático para la gestión de los procesos que ocurren en la residencia estudiantil de la facultad 2 de la Universidad de Ciencias Informáticas en Cuba. Se guió para la identificación, modelación y descripción de las funcionalidades, por la metodología XP. En cuanto a las tecnologías, se emplearon: HTML5, CSS3, JavaScript, respaldadas por el marco de diseño Bootstrap. El lenguaje utilizado fue Python mediante su framework Django, se empleó el Visual Studio Code como entorno de desarrollo, SQLite para las bases de datos. Se implementaron funcionalidades en la gestión de edificios, apartamentos, grupos, estudiantes, evaluaciones, medios básicos, cuartelería y guardia, agilizando los procesos que se involucran y relacionan en dicho entorno. Se llevaron a cabo pruebas de software a la propuesta de solución, las cuales proporcionan niveles de confirmación de las funcionalidades y la satisfacción por parte del cliente.

Palabras claves: cuartelería, evaluaciones, guardia, medios básicos

ABSTRACT

The present research study focuses on the creation of a management system for the processes that occur in the student residence of faculty 2 at the University of Informatics Sciences in Cuba. The XP methodology was used to identify, model, and describe business processes. HTML5, CSS3, JavaScript, backed by the Bootstrap design framework, were used for user-side technologies. Python was used as the server-side language through its Django framework, Visual Studio Code was used as the development environment, and SQLite was used for databases. Functionalities were incorporated into the management of buildings, apartments, students, groups, evaluations, basic media, barracks, and guard. Software tests were carried out on the proposed solution, which provide levels of confirmation of the functionalities and user satisfaction.

Keywords:

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los conocimientos que ha adquirido el hombre a través de los años y su aplicación, propician que las nuevas generaciones logren del mundo de la Tecnología de la Información y las Comunicaciones (TIC), el eslabón principal para cualquier avance tecnológico, revolucionando así la vida diaria y el mercado laboral, permitiendo que el trabajo sea sumamente fácil. (Hernández Alfonso, 2015).

Hoy en día, los sistemas de gestión son muy utilizados por la manipulación sencilla, acceso y disponibilidad, siendo usados en la mayoría de empresas. Los sistemas de gestión se encuentran en la internet para el ámbito académico, laboral o empresarial; facilitando la automatización de procesos. Estos sistemas se encuentran alojados en servidores web, con el objetivo de dar una respuesta rápida a todos los usuarios que realicen o soliciten información segura y accesible en todo momento. Estos sistemas permiten automatizar los diferentes procesos que se manejan dentro de una organización, siendo versátiles, manteniendo la comunicación de forma digital y al instante, generando una mejor manipulación de datos, rendimiento y agilizando su gestión (Hernández Alfonso, 2015).

La gestión por procesos, nace como un enfoque integral que concentra la atención sobre las actividades de la organización y orienta la satisfacción de las perspectivas de sus usuarios. Contempla una visión sistémica que maneja la sinergia entre los conceptos de sistema, proceso y gestión. Permitiendo determinar las fortalezas y debilidades dentro de una organización, y aplicar métodos de mejora continua de procesos, para obtener eficiencia y eficacia (Matute, 2020).

En Cuba, la informatización de sus instituciones ha alcanzado un gran auge, donde el principal objetivo radica, en lograr buena calidad y rapidez a la hora de procesar los datos que se manejan en el desarrollo de sectores tanto productivos como educacionales. Con el avance de la informática y las comunicaciones, muchas corporaciones se ven obligadas a la digitalización de la mayoría de sus procesos, desarrollándose de cada uno de ellos varios

sistemas de adquisición de datos. Las universidades, no están exentas de este desarrollo, al ser muy factible la gestión de la información cuando se cuenta con algún sistema automatizado (Matute, 2020).

La Universidad de las Ciencias Informáticas (UCI), tiene entre sus objetivos la informatización de todas sus áreas, con el propósito de mejorar el funcionamiento de los procesos que tienen lugar en el centro. La residencia estudiantil, identificada como una de las áreas de resultados claves en el desarrollo creciente de la universidad, se sustenta sobre la base de la planificación, organización y control de todas las actividades no curriculares realizadas por los estudiantes (Rodríguez Rodríguez & Fernández Guevara, 2016).

La residencia es una de las áreas más extensas dentro del campus universitario, la cual cuenta con un aproximado de 210000 m^2. Se encuentra dividida en áreas pertenecientes a cada facultad y dentro de cada área se divide en edificios de estudiantes y profesores. En conjunto, las áreas asignadas a estudiantes de la Facultad 2 están compuestas por 28 edificios para un total de 224 apartamentos donde conviven cerca de 672 estudiantes (Rodríguez Rodríguez & Fernández Guevara, 2016).

En ella se llevan a cabo los procesos referentes al control y gestión de medios básicos por apartamento y edificio; la cuartelería, guardia estudiantil y evaluaciones integrales de cada estudiante, así como las incidencias constructivas por cada edificio y apartamento, esto propicia que se generen grandes volúmenes de datos a procesar. Las instructoras, especialistas y directivos que constantemente visitan el área residencial hacen una recopilación de estos datos de forma manual utilizando hojas y lápices, lo que favorece la pérdida, deterioro y duplicidad de la información.

Los procesos no dejan de ser lentos y engorrosos, propiciando que determinadas informaciones en diferentes momentos no estén a tiempo, o estén incorrectas por la premura o los pasos a seguir de forma manual y normal.

Con un sistema de este tipo se daría un gran paso en la planificación, organización y control de todas las actividades no curriculares, es por ello que, a partir de lo anteriormente descrito, se plantea como **problema de investigación:** ¿Cómo facilitar la organización, centralización

y control de los procesos que ocurren en la residencia estudiantil de la Facultad 2?, se plantea como **objeto de estudio:** la gestión de los procesos para el control y conservación de la información en la residencia estudiantil.

Para solucionar el problema planteado, se establece el siguiente **objetivo general:** Desarrollar un sistema informático para la gestión de los procesos en la Residencia Estudiantil de la Facultad 2.

Con el fin de alcanzar el objetivo general establecido, se derivan los siguientes **objetivos específicos**:

1. Elaborar un marco teórico-metodológico sólido que sustente la investigación.
2. Realizar el modelado y la especificación de las funcionalidades para la propuesta de solución de manera detallada.
3. Llevar a cabo la implementación correspondiente a la planificación y ejecutar las funcionalidades en el Sistema Informático para gestión de procesos en la Residencia Estudiantil de la Facultad 2.
4. Validar el correcto funcionamiento de la propuesta de solución.

El **campo de acción** de esta investigación se centra en la gestión de procesos en la Residencia Estudiantil de la Facultad 2**.** Con el fin de guiar el desarrollo de la investigación, se plantean las siguientes preguntas científicas:

1 ¿Cuáles son las corrientes teóricas predominantes en relación con los sistemas para gestión y su aplicación en el proceso llevados a cabo en la Residencia Estudiantil de la Facultad 2?

2 ¿Cuál es el estado real de desarrollo alcanzado en la informatización de las áreas involucradas en el flujo y manejo de la información de los procesos en la Residencia Estudiantil de la Facultad 2?

3 ¿Cuál es el diseño óptimo de un sistema para gestión que se adapte a las particularidades de los procesos llevados a cabo en la Residencia Estudiantil de la Facultad 2?

4 ¿Cómo se puede implementar un sistema para la gestión de procesos que garantice su destreza en la ejecución?

Tareas de investigación científica.

1. Referentes históricos y teóricos predominantes en relación con los sistemas para gestión y su aplicación en el proceso llevados a cabo en la Residencia Estudiantil de la Facultad 2.
2. Caracterización del estado real de desarrollo alcanzado en la informatización de las áreas involucradas en el flujo y manejo de la información de los procesos en la Residencia Estudiantil de la Facultad 2.
3. Defínción de un diseño para el sistema de gestión que se adapte a las particularidades de los procesos de llevados a cabo en la Residencia Estudiantil de la Facultad 2.
4. Implementación de un sistema para la gestión de procesos llevados a cabo en la Residencia Estudiantil de la Facultad 2 que garantice destreza en su ejecución.
5. Validación del sistema propuesto que cuente con la calidad requerida por el cliente.

Métodos teóricos:

Analítico - sintético: Se va a utilizar en el análisis de los elementos bibliográficos y definiciones sobre la gestión de los procesos de la residencia, con el objetivo de arribar a conclusiones que respalden la necesidad de la investigación.

Histórico-lógico: Este método es utilizado para realizar el estudio de los antecedentes históricos y el comportamiento de desarrollo de los sistemas para la gestión y su aplicación en el proceso de planificación y ejecución de los mismos en residencias estudiantiles.

Modelación: Se va a emplear para la creación de modelos y diagramas, brindando así una reproducción ampliada de la realidad. Posibilitará descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Métodos empíricos:

Entrevista: Va a permitir contactar personas que están involucradas con los procesos en la Residencia Estudiantil y que puedan aportar información valiosa para la investigación. En este caso se va a realizar una entrevista a directivos como instructoras y psicopedagogas de la Dirección de Residencia de la Facultad 2.

Observación: Este método se va a utilizar para realizar un seguimiento cercano y analizar los resultados de la gestión de los procesos en el área de Residencia Estudiantil de la Facultad 2.

El presente trabajo de diploma se encuentra estructurado en tres capítulos los cuales están distribuidos de la siguiente forma:

CAPÍTULO 1: Fundamentos y referentes teórico-metodológicos: Incluyendo una revisión de análisis de soluciones existentes similares en sistemas para la gestión de procesos de planificación y ejecución de servicios de la Residencia Estudiantil de la Facultad 2. Además, se va a presentar la metodología de desarrollo de software a aplicar en el sistema, junto con las herramientas seleccionadas para su correcta implementación.

CAPÍTULO 2: Características y estructura del sistema: Se propone una solución basada en el análisis del proceso de planificación y ejecución de servicios de la Residencia Estudiantil de la Facultad 2. Se describe el sistema para la gestión de procesos en la residencia estudiantil y se crea un modelo representativo. Además, se plantea la estructura del sistema.

CAPÍTULO 3: Implementación y validación del sistema: Se describe el diseño de las pruebas que se le aplican al sistema para comprobar su correcto funcionamiento. Se adopta un enfoque de pruebas de acuerdo con determinadas condiciones.

CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS

Introducción

En el presente capítulo se reflejarán conceptos y definiciones que estarán relacionados con el alojamiento en residencias estudiantiles, el cual permitirá la mejor compresión de los procesos referente a este dominio. Se realizará un estudio de sistemas con características y funciones semejantes al sistema en cuestión y abordará la metodología seleccionada para realizar el proyecto de la forma más óptima. Se describen además las tecnologías de desarrollo utilizadas para el análisis, diseño e implementación del sistema sobre las cuales se apoya la propuesta.

I.1 Conceptos asociados al objeto de estudio

Las **residencias estudiantiles** son un espacio físico en donde la población estudiantil, proveniente de zonas alejadas o de difícil acceso, de escasos recursos económicos y con categoría de beca socioeconómica 9, 10 y 11, se aloja durante el año lectivo para llevar a cabo su plan académico (Regueyra Edelman & María Gabriela, 2010).

El **inventario** se puede definir como la administración de existencias de todo producto o artículo que es utilizado para la comercialización dentro de una organización. Es decir, todo lo relativo al control y manejo de las existencias de determinados bienes, en la cual se aplican métodos y estrategias que pueden hacer rentable y productivo la tenencia de estos bienes y a la vez sirve para evaluar los procedimientos de entradas y salidas de dicho producto (Rodríguez Viera, 2015).

La **Guardia Estudiantil** es el proceso llevado a cabo en el Ministerio de Educación Superior (MES) mediante el cual los estudiantes y trabajadores de los centros de estudio de manera planificada velan por el cuidado y protección de los recursos humanos y materiales en sus respectivas instituciones. Las indicaciones mencionadas con anterioridad delegan al decanato de cada facultad de la universidad la máxima responsabilidad en su planificación (Hector Alexey Garcés Rey, 2022)**.**

I.2 Análisis de soluciones existentes similares.

El estudio de los sistemas homólogos existentes a nivel internacional y nacional es primordial, logrando una mejor comprensión de las características del sistema a desarrollar. También se identifican las similitudes al propuesto en la presente investigación. A continuación, se expone el análisis desarrollado.

Soluciones similares existentes a nivel internacional.

1. **Aplicación web progresiva dedicada a la gestión de incidencias para la Residencia Universitaria de Tafira, España 2022.**

Dentro de las funcionalidades implementadas, la aplicación permite a sus usuarios registrar cualquier tipo de incidencia que tenga lugar dentro de las instalaciones de la residencia o de los apartamentos universitarios, además de realizar un seguimiento hasta su posterior resolución. La aplicación además permite notificar a los residentes de cualquier evento o anuncio importante que considere oportuno el personal de la residencia. Asimismo, otra de las funcionalidades consideradas en la aplicación actual es la de poder registrar las ausencias que los residentes tengan previstas durante el transcurso del curso (Martín Socas, 2022).

2. **Sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor Universidad Cooperativa de Colombia 2019.**

En este trabajo se presenta la sistematización del inventario para el almacén de pinturas y ferretería Ferrecolor, el cual carecía de una herramienta informática que le permitiera ejercer mayor control sobre los productos que comercializa. El administrador utilizaba hojas de cálculo y anotaciones en apuntes para manejar su inventario, lo cual le impedía tener información clara, rápida y precisa, sobre los artículos que ofrece para la venta al público. Gracias al análisis de los requerimientos, para la gestión de inventario y las buenas prácticas de SCRUM, fue posible dar solución creando una aplicación web con el estándar de JAVA EE, el framework JSF (Java Server Faces) y el motor de base de datos MySQL. Esta solución le permitirá al almacén y su administrador, consultar y controlarla base de datos de

su inventario, y con ello enfocar los recursos y lograr un mejor crecimiento empresarial (Romero, 2019).

Soluciones similares existentes a nivel nacional.

3. **El Sistema de gestión de información para la Residencia Universitaria en Holguín.**

Analiza los procesos de matrícula y el inventario de los medios básicos con que cuentan. La matrícula se realiza mediante la recogida manual de información. Se organizan, se tabulan los datos y luego se archivan en tarjetas y papeles. En el inventario se hace un levantamiento de medios existentes en cada local de la Residencia. Toda esta información es almacenada en modelos de inventario y luego es enviada al Departamento Económico. Se usó el Sistema de Administración de Contenidos (CMS) con la tecnología Procesador de Hipertexto (PHP) y el servidor de Base de Datos MySQL. Se utilizó como servidor Web Apache (Jenny Ruiz de la Peña, 2010)

4. **Sistema de gestión de información de la Residencia Estudiantil de la Universidad de Ciego de Ávila.**

La residencia estudiantil de la Universidad de Ciego de Ávila Máximo Gómez Báez (UNICA), es un espacio para brindar no solo servicio de alojamiento a estudiantes, sino un espacio activo de formación. Es un área universitaria donde de forma natural se puede integrar las dimensiones formación curricular, extensión universitaria y actividad sociopolítica. A pesar de la importancia en la formación del estudiante que la universidad delega en esta área se pudo constatar, a partir de un estudio realizado, que existen deficiencias en algunos de sus procesos, reflejadas en:

1. Análisis inadecuados de los indicadores establecidos para la evaluación de los estudiantes becados.
2. La información manejada en esta área no llega en ocasiones en el tiempo oportuno a la dirección del centro.
3. Deficiente planificación de las actividades de la residencia.
4. Inadecuada disponibilidad de información entre las dos sedes de residencia.

A partir del 2010, varias residencias estudiantiles, a nivel mundial y nacional, han hecho uso de diferentes herramientas informáticas para apoyar de alguna manera la labor diaria

que se realiza en esta área, sobre todo en función del trabajo educativo que se efectúa con los estudiantes (Lesvy Alemán Mateo, 2021).

5. **Sistema de Gestión de Información de la Facultad 8. Modulo para la Gestión de la Residencia Estudiantil Versión 2.**

Implementación de nuevas funcionalidades que le agregan al Módulo para la Gestión de la Residencia Estudiantil de la Facultad 8 de la Universidad de las Ciencias Informáticas (UCI). Proporcionarle a la dirección de la facultad la posibilidad de tener registrados en algún servidor de base de datos todo lo referido a los procesos de cuartelería, visitas a apartamentos, trabajos socialmente útiles y guardias estudiantiles, lo cual hace más rápida y eficiente la búsqueda de una información determinada sobre alguno de ellos (Dávila Pérez et al., 2008).

6. **Sistema para la planificación de la Guardia Obrera Estudiantil de la Facultad 1**

El sistema es capaz de almacenar disímiles datos que son de interés y análisis en el momento de otorgar un turno de guardia a una persona, dígase: última guardia realizada, ubicación residencial de la persona, pruebas a realizar en el mes, además de los datos comunes de registro: nombre, apellidos, año académico (en el caso de los estudiantes).

El sistema genera un horario de forma asistida, haciendo uso de las restricciones de cada persona en el momento de otorgar un turno, una vez realizada la planificación se puede editar, así como intercambiar personas entre turnos, dicha planificaciones se exporta a formato Excel para su posterior envío al personal. El administrador puede realizar consultas a un historial de guardia en caso de que surja la necesidad, así como introducir los incumplidores y enviar esta información al personal pertinente (Lidia del Carmen Leyva Feijoó & Alejandro López Rodríguez, 2015).

Debido a que sus características no son las óptimas y más ventajosas estos sistemas no son factibles para obtener mejores resultados en la gestión de los procesos la Residencia Universitaria.

La novedad científica de esta investigación es la elaboración de un sistema que favorece la gestión de procesos de la Residencia Universitaria de la Universidad de las Ciencias Informáticas (UCI) en cuanto a los procesos de matrícula de becados, supervisión del cumplimiento de la cuartelería, la guardia estudiantil y del inventario de medios básicos de los apartamentos.

Tabla 1Comparación de sistemas similares

Sistemas Homólogos	Gestuionar residencia	Gestionar Medios básicos	Gestionar Cuartelería	Gestionar la Guardia	Asignar Evaluación	Tecnología	Metodología
Aplicación web progresiva dedicada a la gestión de incidencias para la Residencia Universitaria de Tafira.	No	No.	No.	No.	Sí.	No.	No.
Sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor.	No	Sí	No	No	No	No	No

El Sistema de gestión de información para la Residencia Universitaria en Holguín.	No	Sí	No	No	Sí	No	No
Sistema de gestión de información de la Residencia Estudiantil de la Universidad de Ciego de Ávila.	Sí	No	No	No	Sí	Sí	No
Sistema de Gestión de Información de la Facultad 8. Modulo para la Gestión de la Residencia Estudiantil Versión 2	Sí	No	Sí	Sí	No	No	No
Sistema para la planificación de la Guardia Obrera Estudiantil de la Facultad 1	No	No	No	Sí	No	No	No

Tras el estudio de las soluciones similares existentes fuera y dentro del país se llega a la conclusión que ninguna de las estudiadas resuelve la problemática planteada anteriormente, puesto que cada una de las residencias pertenecientes a las universidades presentan características diferentes. Además, los Sistemas son diseñados con herramientas propietarias o no contienen módulos que permitan gestionar la información de los procesos que se tratan en la presente investigación.

I.3 Metodología.

Una **metodología de software** es un conjunto de enfoques, técnicas, procedimientos y herramientas estandarizados que se utilizan en el ciclo de vida del desarrollo de software para planificar, diseñar, implementar, probar y mantener sistemas de software de manera sistemática y efectiva.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Optimiza el proceso y el producto. Presenta métodos que guían en la planificación y en el desarrollo del software. Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. No tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo (Pressman, 2014).

Programación Extrema (XP)

La programación extrema es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos y con menos documentación. La estructura del proceso está determinada por 5 valores fundamentales, 5 reglas y 12 prácticas de XP.

Al igual que otras metodologías ágiles, la Programación Extrema es un método de desarrollo de software dividido en sprints de trabajo. Los marcos ágiles siguen un proceso iterativo, en el que se completa y revisa el marco al final de cada sprint, refinándolo para adaptarlo a las funcionalidades cambiantes y alcanzar la eficiencia máxima. Al igual que otros métodos ágiles, el diseño de la Programación Extrema permite a los desarrolladores responder a las solicitudes de los clientes, adaptarse y realizar cambios en tiempo real. Sin embargo, la

programación extrema es mucho más disciplinada; realiza revisiones de código frecuentes y pruebas unitarias para realizar cambios rápidamente. Además, es muy creativa y colaborativa, ya que promueve el trabajo en equipo durante todas las etapas de desarrollo (Letelier, 2006).

Es el más destacado de los procesos ágiles de desarrollo de software. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos (Maida, EG, Pacienzia, 2016).

Las historias de usuario son la técnica utilizada en XP para especificar las funcionalidades del sistema. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento pueden eliminarse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (BECK, 1999).

XP como metodología al fin, presenta determinados artefactos para guiar el desarrollo del proyecto. Los artefactos son las Historias de Usuario (HU), las Tareas de Ingeniería (TI) y las tarjetas Clase-Responsabilidad Colaboradores (CRC) (BECK, 1999).

XP contiene varios roles: Programador, cliente, Encargado de pruebas (Tester), Encargado de seguimiento (Tracker), Entrenador (Coach), Consultor y Gestor (Big boss (BECK, 1999).
Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo (Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP), 2006).

I.3.1 Características de XP

Prefabricación

Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio (Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP), 2006).

Programación en pares

Una particularidad de esta metodología es que propone la programación en pares, 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. Mientras uno programa el otro consulta el algoritmo. (Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP), 2006).

Pruebas Unitarias

Se basa en las pruebas realizadas a los principales procesos, de tal manera que, adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como adelantarse a obtener los posibles errores (Maida, EG, Pacienzia, 2016).

I.3.2 Fases

Los valores de la programación extrema son los aspectos más filosóficos. Las reglas o fases, por otro lado, son los usos prácticos de cómo se realiza el trabajo. Necesitarás contar con ambos aspectos para gestionar un equipo XP efectivo. A continuación la descripción las cinco fases:

1. Planificación

Durante la etapa de planificación de la programación extrema, se deberá determinar si el proyecto es viable y si se adapta al método XP. Para hacer esto, se deberá evaluar:

Las historias de usuarios para confirmar que coinciden con el valor de simplicidad y garantizar que el cliente esté disponible para participar del proceso. Si la historia del usuario es más compleja o fue creada por un cliente anónimo, es probable que no sea adecuada

para el método XP. El valor comercial y el nivel de prioridad del proyecto para asegurarte de que esté alineado con la idea de “realizar el trabajo más importante primero”. La programación extrema se adapta mejor al desarrollo en sus etapas iniciales y no será un método tan efectivo para iteraciones posteriores. Una vez confirmado que el proyecto es viable para implementar la programación extrema, se recomienda crear un cronograma de lanzamiento. Sin embargo, es importante tener en cuenta que se debe lanzar tan pronto sea posible y con frecuencia para poder obtener comentarios. Para hacer esto:

1. Se divide el proyecto en iteraciones y crea un plan para cada una.
2. Se establecen plazos realistas y un ritmo de trabajo sostenible.
3. Se comparten actualizaciones a medida que ocurren para permitir que el equipo sea lo más honesto y transparente posible.
4. Se comparten actualizaciones en tiempo real para ayudar al equipo a identificar, adaptar y realizar cambios más rápidamente.
5. Se usa una herramienta de gestión de proyectos para crear un tablero Kanban o un cronograma para poder dar seguimiento al progreso en tiempo real.

2. Gestión

Uno de los elementos clave de la programación extrema es el espacio de trabajo. Los puristas de XP recomiendan usar un espacio de trabajo abierto donde todos los miembros del equipo puedan trabajar juntos. Debido a que la programación extrema es altamente colaborativa, es una ventaja contar con un espacio donde puedan reunirse todos los colaboradores físicamente. Sin embargo, en estas épocas, esta no siempre es una solución práctica o factible. Si se trabaja en un equipo remoto, se recomienda una plataforma que fomente el trabajo asincrónico para la colaboración remota. De esta manera, todos los miembros podrán trabajar juntos en el proyecto, incluso si no se encuentran en el mismo espacio físico.

Al igual que con otros métodos ágiles, se puede organizar reuniones diarias de actualización para verificar el estado del trabajo y fomentar las comunicaciones abiertas y constantes. Se

recomienda implementar un ciclo semanal y un ciclo trimestral. Durante el ciclo trimestral, el equipo revisa las historias que guían el trabajo a realizar. También analizarán los procesos de XP, para identificar brechas u oportunidades para realizar cambios. Luego, se trabaja en ciclos semanales, que comienzan con una reunión con el cliente. El cliente elige la historia de usuario para que los programadores trabajen esa semana.

El gerente o líder de equipo debe centrar su atención en mantener el progreso constante del trabajo, supervisar el ritmo, gestionar la carga de trabajo del equipo para que puedan trabajar en los errores o problemas a medida que surjan, o cambiar el proceso de XP para que se ajuste a tu proyecto e iteración actuales. El objetivo de la programación extrema es ser flexible y ágil, por lo que el trabajo estará muy centrado en el trabajo actual del equipo y en poder responder rápidamente a cualquier cambio.

3. Diseño

Los equipos de metodología XP a menudo usan tarjetas de clase, responsabilidad y colaboración (CRC, por sus siglas en inglés) para mostrar cómo interactúa cada elemento en el diseño global. Al completar cada campo de la tarjeta, podrás obtener una imagen visual de cómo se relacionan e interactúan todas las funciones. Las tarjetas CRC incluyen:

1. Clase (conjunto de elementos similares).
2. Responsabilidades (según la clase).
3. Colaboradores (clase que interactúa con el colaborador en cuestión).

Las tarjetas CRC son útiles para simular el proceso y detectar problemas potenciales. Independientemente del tipo de diseño, se recomienda usar un sistema que permita reducir los cuellos de botella potenciales. Para esto, asegúrarse de prevenir los riesgos de manera proactiva. Tan pronto como se descubra una amenaza potencial, asignar uno o dos miembros del equipo para encontrar una solución en caso de que suceda.

4. Codificación

Uno de los aspectos clave de la programación extrema es el contacto permanente que se mantiene con el cliente durante todo el proceso de codificación. Esta interacción permite

probar e incorporar comentarios dentro de cada iteración, en lugar de esperar hasta el final de cada sprint. Sin embargo, las reglas de codificación son bastante estrictas en el método XP, algunas incluyen:

1. Todo el código debe cumplir con el estándar de programación.
2. Se deben realizar pruebas unitarias para definir los requisitos y desarrollar todos los aspectos del proyecto.
3. La programación se realiza en parejas: dos desarrolladores trabajan juntos y simultáneamente en la misma computadora. El tiempo de programación sigue siendo el mismo; sin embargo, se duplica el enfoque para lograr resultados de la más alta calidad posible.
4. Se deben usar integraciones continuas para agregar código nuevo y probarlo de inmediato.
5. Solo un par de desarrolladores puede actualizar el código en un momento dado para reducir los errores.
6. Propiedad colectiva del código: cualquier miembro del equipo puede cambiar el código en cualquier momento.

5. Prueba

Se deben realizar pruebas durante todo el proceso de programación extrema. Todo el código debe someterse a pruebas unitarias antes de su lanzamiento. En caso de detectar errores durante estas, crear pruebas adicionales para corregirlos. Más adelante, incorporar la misma historia de usuario en la que se ha estado trabajando en una prueba de aceptación. Durante esta prueba, el cliente examinará los resultados para comprobar que has implementado correctamente la historia de usuario al producto (Letelier, 2006).

I.3.4 Desventajas

Es recomendable emplearla solo en proyectos a corto plazo por lo que, en caso de fallar, las comisiones son muy altas. Puede no siempre ser más fácil que el desarrollo tradicional (Letelier, 2006).

El proyecto define esta metodología ya que los equipos de desarrollo son pequeños, el cliente forma parte del proyecto y la documentación generada es de pequeña envergadura (Letelier, 2006).

I.4 Tecnologías y herramientas.

Herramienta CASE Visual Paradigm 15.1

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El sistema de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (Asana, 2022).

Lenguajes de programación

Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación para la web dinámicos, que permitieran interactuar con los usuarios y utilizaran sistemas de bases de datos. Actualmente existen diferentes lenguajes de programación para desarrollar en la Web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Cada uno de ellos posee diferentes características que lo hacen útiles para desarrollar determinadas aplicaciones. Estos se clasifican de acuerdo a la arquitectura Cliente/Servidor en:

- Lenguajes del lado del servidor: son aquellos reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él entre los que se encuentran PHP, ASP, JSP, Perl, Python y Ruby.

- Lenguajes del lado del cliente: son aquellos que pueden ser directamente comprendidos por el navegador como el VBScript, el JavaScript y Ajax. Estos dos últimos son incluidos en el código HTML (Asana, 2022).

Ambos tienen ventajas y desventajas puesto que un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio mientras que un lenguaje del lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo; por otra parte, los scripts son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente.

HTML5

Lenguaje de Etiquetado de Hipertexto (Hypertext Markup Language): Es un lenguaje comúnmente utilizado para la publicación de hipertexto en la web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la web (W3C, 2015). La última versión de este lenguaje es la conocida HTML5, la misma está pensada con una mayor integración con los lenguajes CSS3 y JavaScript. HTML5 ha revolucionado la web y no solo se ve como el presente, sino como el futuro, por las numerosas novedades que trae con respecto a la versión anterior. La selección de HTML5 como el lenguaje para realizar el maquetado de la propuesta de solución está sustentada en el hecho de ser este un estándar establecido por la W3C (Asana, 2022).

CSS3

Hojas de Estilo en Cascada (Cascading Style Sheets): Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, cómo se va a imprimir, o cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos (W3C, 2015).

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS

permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento (Eguiluz, 2015).

La selección de CSS 3 como lenguaje para escribir el estilo visual de la propuesta de solución se hizo teniendo en cuenta que el mismo es un estándar determinado por la W3C y posee un amplio uso y difusión por parte de maquetadores, diseñadores y desarrolladores web.

JavaScript v5

Es un lenguaje de programación interpretado, se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico, es más conocido como el lenguaje de script para páginas web. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguiluz, 2015).

Bootstrap v4.3

Es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML5 y CSS3 con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript, además permite la adaptación de la interfaz dependiendo del tamaño del dispositivo en el que se visualice sin que el usuario tenga que hacer nada, esto se denomina Responsive Web Design. El Framework trae varios elementos con estilos predefinidos fáciles de configurar e integración jQuery para ofrecer ventanas y tooltips dinámicos (Eguiluz, 2015).

Python v3.11

Es un lenguaje de programación creado en el año 1990 por Guido van Rossum y es el sucesor del lenguaje de programación ABC. Python3 es comparado habitualmente con Perl. Los usuarios lo consideran como un lenguaje más limpio para programar. Permite la creación de todo tipo de programas incluyendo los sitios web. Su código no necesita ser compilado. Es un lenguaje de programación multiparadigma, lo cual fuerza a que los programadores adopten un estilo de programación particular (Asana, 2022).

Django v4.1.3

Es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código, fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD (W3C, 2015).

IDE de desarrollo

Para desarrollar, normalmente solo es necesario un editor de texto, un intérprete o compilador y una terminal de líneas de comando, un Entorno de Desarrollo Integrado (IDE); Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment, es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software. Esto simplifica el trabajo y ahorra tiempo de desarrollo (W3C, 2015).

Visual Studio Code v1.82

Visual Studio Code es un editor de código fuente open-source desarrollado por Microsoft para Windows, Linux, macOS, Web y OSX. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Es gratuito y también tiene diferentes extensiones que permite trabajar con varios lenguajes (Marín Rafael, 2019).

Sistemas Gestores de Base de Datos (SGBD)

Un Sistema Gestor de Base de Datos (SGBD) o DataBase Managenent System (DBMS) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de información del modo más eficiente posible (Marín Rafael, 2019).

El sistema gestor de bases de datos SQLite v2.3

SQLite es un SGBD relacional, famoso por su pequeño tamaño. A diferencia de otros sistemas cliente-servidor el motor de SQLite no es un proceso independiente, lo que hace que la latencia sea menor y el acceso más eficiente. Debido a su facilidad de uso, su

pequeño tamaño y su versatilidad SQLite es utilizado en una gran variedad de aplicaciones. Su uso ha sido muy popular en las aplicaciones para Smartphone con sistema operativo Android o iOS. Teniendo en cuenta la bibliografía consultada se puede afirmar que este gestor es mucho más rápido y ligero que MySQL y PostgreSQL. Se ejecuta en muchas plataformas, es de dominio público y por tanto sin costo. Por lo cual, las empresas de desarrollo de software en nuestro país tienen un aumento considerable de la producción de aplicaciones para móviles con sistema operativo Android que utilizan este SGBD, como por ejemplo EnZona, Participación Popular, ONAT, TransferMóvil (Marín Rafael, 2019).

Conclusiones del capítulo

- En este capítulo se expusieron las condiciones y problemas que inciden en el objeto de estudio; a través de los conceptos y definiciones planteadas.
- Se evidenció la necesidad de implementar un sistema que permita llevar a cabo el control y gestión de la información del Área de Extensión Universitaria en la Facultad 2.
- Para desarrollar el sistema se hace uso del lenguaje de programación Python y con soporte de base de datos en SQLite.
- La metodología utilizada es XP teniendo en cuenta las características del proyecto y de la propuesta de solución, así como del entorno

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA.

Introducción

En este capítulo se realiza una descripción general de la propuesta de solución, definiendo los requisitos funcionales y no funcionales del sistema. Las funcionalidades se describen mediante las historias de usuario (HU). Se realiza el plan de iteraciones donde se muestran las HU a realizar en cada iteración según su prioridad en los procesos.

II.1Propuesta de solución.

En la primera fase de la metodología XP, los clientes plantean a grandes rasgos las funcionalidades que son de interés para la elaboración del producto, transformándose en historias de usuario. Partiendo de la información obtenida, el equipo de desarrollo evaluará de forma general el tiempo de codificación, se familiarizará con las herramientas, tecnologías y prácticas que serán utilizadas en el proyecto, además, se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo para ello.

II.1.2 Relacionado con el sistema

Uno de los principales factores que se deben tener en cuenta cuando se comienza el desarrollo de un sistema informático es la delimitación de la audiencia a la cual va dirigido el mismo, lo cual, en algunas ocasiones, puede referirse a un usuario relacionado con el sistema o alguien completamente ajeno a él. Se debe especificar que esta audiencia a su vez puede ser dividida en grupos atendiendo a sus necesidades. Se define como usuario relacionado con el sistema, a todo aquel que de una manera u otra interactúa con este, obteniendo un resultado de uno o varios procesos que se ejecutan en el mismo.

Tabla 2 Usuarios presentes en el sistema

Usuario	Descripción
Vicedecana de Extensión y Residencia de la Facultad 2	Tiene completo acceso al sistema y posee el rol de Administrador
Especialista General	Tiene acceso a visualizar la información de la residencia pero no a modificarla o eliminarla. También tiene completo acceso para gestionar cuartelería, guardia y medios básicos
Psicopedagoga	Tiene acceso a visualizar la información de la residencia pero no a modificarla o eliminarla. También tiene completo acceso para gestionar cuartelería, guardia y medios básicos
Instructor de la Residencia de la Facultad 2	Tiene acceso a visualizar la información de la residencia pero no a modificarla o eliminarla. También tiene completo acceso para gestionar cuartelería, guardia y medios básicos

II.2 Lista de funcionalidades del Sistema

Una funcionalidad es la condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. También se define como la condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación u otra documentación formalmente impuesta (Lazo & Botero, 2016).

Lista de las funcionalidades del sistema con su descripción.

1. Autenticar usuario.
2. Gestionar usuario.
   2.1. Añadir usuario.
   2.2. Modificar usuario.
   2.3. Eliminar usuario.
   2.4. Listar usuario.
3. Buscar usuario.
4. Gestionar edificio
   4.1. Añadir edificio.
   4.2. Modificar edificio.
   4.3. Eliminar edificio.
   4.4. Listar Edificio.

5. Buscar edificio.
6. Gestionar apartamento.
    6.1. Añadir apartamento.
    6.2. Modificar apartamento.
    6.3. Eliminar apartamento.
    6.4. Listar apartamento.
7. Buscar apartamento.
8. Gestionar grupo docente.
    8.1. Añadir grupo docente.
    8.2. Modificar grupo docente.
    8.3. Eliminar grupo docente.
9. Buscar grupo docente.
10. Gestionar estudiante.
    10.1. Añadir estudiante.
    10.2. Modificar estudiante.
    10.3. Eliminar estudiante.
    10.4. Listar estudiante.
11. Buscar estudiante.
12. Listar estudiante en apartamento.
13. Gestionar medio básico.
    13.1. Añadir medio básico.
    13.2. Modificar medio básico.
    13.3. Eliminar medio básico.
    13.4. Listar medios básicos
14. Buscar medio básico.
15. Gestionar medio básico en un apartamento.
    15.1. Añadir medio básico a un apartamento.
    15.2. Listar medios básicos de un apartamento.

15.3. Modificar medio básico en un apartamento.

16. Gestionar cuartelería.
    16.1. Planificar cuartelería.
    16.2. Modificar cuartelería.
    16.3. Eliminar cuartelería.
17. Asignar evaluación a la cuartelería.
18. Buscar cuartelería.
19. Gestionar guardia.
    19.1. Planificar guardia.
    19.2. Modificar guardia.
    19.3. Eliminar guardia.
20. Asignar evaluación a la guardia.
21. Buscar guardia.

II.3 Historias de usuario

Una historia de usuario es una explicación informal de una función de sistema, escrita desde la perspectiva del usuario final. Estas historias deben escribirse utilizando un lenguaje no técnico para brindar contexto al equipo de desarrollo.

¿Qué es una historia de usuario?

El propósito de escribir historias de usuario es representar con precisión de qué manera una función de sistema de gestión se traduce en valor para el usuario. En otras palabras, ¿Qué impacto produce esta función en el usuario final?

El usuario final, también conocido como cliente, no necesariamente es un consumidor externo. El usuario final también puede ser un cliente interno o un miembro del equipo que se beneficiará de este trabajo. Quién es el usuario final va a determinar en última instancia el propósito de la función de sistema que se está creando (Asana, 2022).

Las historias de usuario son un componente central en el enfoque ágil. Se puede redactar de muchas maneras, hasta con notas autoadhesivas o fichas, pero la forma más eficaz de crear y realizar un seguimiento de las historias de usuario es con un software de gestión de proyectos. El sistema de gestión de proyectos eficaz permite ajustar, editar y dar seguimiento a las historias de usuario en tiempo real para que el equipo sepa exactamente cómo pueden

brindar un mejor servicio a sus usuarios finales. Teniendo en cuenta que el desarrollo de software ágil tiene como objetivo poner a las personas en primer lugar, y las historias de usuario te permiten hacerlo al posicionar tu atención en el usuario final (Asana, 2022).

Las HU son representadas mediante tablas divididas por las siguientes secciones:

- **Número**: Esta sección representa el número incremental en el tiempo de la historia de usuario que se describe.
- **Nombre de Historia de Usuario**: Identifica la HU que se describe entre los desarrolladores y el cliente.
- **Usuario**: Rol del usuario que realiza la funcionalidad.
- **Prioridad en negocio**: Se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.
- **Riesgo en Desarrollo**: Se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU.
- **Iteración asignada**: Número de la iteración donde va a desarrollarse la HU.
- **Puntos Estimados**: Es el tiempo estimado en semanas ideales (40 horas) que se demorará el desarrollo de la HU.
- **Descripción:** Breve descripción de la HU.
- **Observaciones**: Señalamiento o advertencia del sistema.(Rivero, 2015).

A continuación, en las tablas se muestran ejemplos de las HU correspondientes a las funcionalidades **“Autenticar usuario** “y **“Gestionar Apartamento”**. El resto de las HU se encuentran en los anexos del presente documento.

Tabla 3 Historia de Usuario Gestionar Cuartelería

Historia de Usuario	
Número:17	Nombre: Gestionar Cuartelería
Usuario: Todos	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: Cuenta con un las funcionalidades de generar cuartelería (planifica la cuartelería del mes por edificios), modificarla, eliminarla, así como asignarle una evaluación al estudiante encargado de la cuartelería en su edificio correspondiente.	
Observaciones: Para un mejor flujo de trabajo deben incorporarse estas funcionalidades tomando como referencia un edificio y dependiendo de la cantidad de escaleras que posea se debe planificar acotando los estudiantes por los apartamentos de escalera y por escalera.	

Tabla 4 Historia de Usuario Gestionar Guardia

Historia de Usuario	
Número: 20	Nombre: Gestionar Guardia
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos de la guardia, como generar la guardia del mes (se agrupan los estudiantes por grupo docente) de forma automatizada, así como modificar y eliminar una guardia, y además asignar una evaluación relacionada con el desempeño de los estudiantes durante su turno de guardia.	
Observaciones: La guardia en la facultad 2 se realiza a partir de los días 21 de cada mes hasta el último día de dicho mes. El grupo docente se divide en 2 semigrupos que a su vez realizan la guardia en 2 turnos (1er turno: de 10-2am y el 2do turno: de 2-6 am).	

II.4 Planificación

Una vez descritas e identificadas las historias de usuario y estimado el esfuerzo necesario para el desarrollo de cada una de ellas, se procede a realizar la planificación de las etapas de implementación del sistema. Con este fin se define un plan que contiene las iteraciones a realizar y la relación de las historias de usuario que serán implementadas y en qué orden para cada iteración, teniéndose en cuenta la duración de las mismas. En la primera iteración se crea la arquitectura de todo el sistema; esto es logrado seleccionando las historias que hacen cumplir la estructura para todo el sistema. El cliente selecciona las historias para cada iteración. Al final de la última iteración, el sistema estará listo para ser entregado y llevarlo a producción.

II.5 Estimación de esfuerzo de las historias de usuario

Una vez asignadas las prioridades de las historias de usuario, se prosigue a estimar el esfuerzo necesario para la elaboración de cada una de ellas por parte de los desarrolladores. Esta estimación se basa principalmente en la velocidad del equipo de desarrollo y en la semejanza con historias de usuario desarrolladas con anterioridad. Las HU de la presente investigación tienen un valor entre 1 y 3 puntos, los puntos estimados según la metodología XP son expresados en semanas ideales (), ha de tenerse en cuenta que muy pocas veces el cronograma se lleva a cabo exactamente como se planifica. El esfuerzo necesario para construir las historias de usuario está basado en la técnica de estimación para el desarrollo ágil, quedando conformada de la siguiente manera.

Tabla 5 Estimación de Esfuerzo de las HU

Historia de Usuario	**Puntos estimados (Semanas)**
Autenticar usuario.	1.4
Gestionar usuario	2
Buscar usuario	0,4
Gestionar edificio	2
Buscar edificio.	0,4
Gestionar apartamento	2
Buscar apartamento.	0.4
Gestionar grupo docente	2
Buscar grupo docente.	0.4
Gestionar estudiante	2.2
Buscar estudiante	0.4
Listar estudiante en apartamento	1
Gestionar medio básico.	2
Buscar medio básico.	0,4
Gestionar medio básico en un apartamento	2
Gestionar cuartelería	2.6

Asignar evaluación a la cuartelería.	1
Buscar cuartelería.	0.4
Gestionar guardia	2.6
Asignar evaluación a la guardia.	1
Buscar guardia.	0,4
Total	**27**

II.6 Plan de Iteraciones.

De los planteamientos anteriores se determina que para el desarrollo de la aplicación web en cuestión se necesita un total de dos iteraciones, teniendo en cuenta la prioridad asignada a cada HU en correspondencia con las solicitudes del cliente.

Tabla 6 Plan de Iteraciones

Iteración	Descripción de la iteración	HU	Duración
1	En la primera iteración se seleccionaron las HU de alta prioridad de acuerdo a la solicitud del cliente de manera que el tiempo de desarrollo de las iteraciones esté balanceado y se conforme un producto que aún sin terminar pueda mostrar las funcionalidades básicas del sistema.	1. Autenticar usuario. 2. Gestionar usuario. 3. Buscar usuario. 4. Gestionar edificio 5. Buscar edificio 6. Gestionar apartamento 7. Buscar apartamento. 8. Gestionar grupo docente. 9. Buscar grupo docente. 10. Gestionar estudiante. 11. Buscar estudiante. 12. Listar estudiante en apartamento. 13. Gestionar medio básico. 14. Buscar medio básico. 15. Gestionar medio básico en un apartamento.	18 semanas
2	En la segunda iteración, ya implementadas las funcionalidades básicas, se realiza el desarrollo de las restantes HU. Además, se corregirán errores o disconformidades del cliente con las historias de usuario implementadas en la iteración anterior.	16. Gestionar cuartelería. 17. Asignar evaluación a la cuartelería. 18. Buscar cuartelería. 19. Gestionar guardia. 20. Asignar evaluación a la guardia. 21. Buscar guardia.	9 semanas

II.7 Plan de entrega

En el momento de planificar la entrega de un software se debe llevar un balance de este, pues si se realiza muy pronto no se tendrán suficientes funcionalidades que ameriten dicha liberación, por otro lado, esperar mucho tiempo, conlleva a que el software desarrollado quede atrás respecto a la competencia.

En la metodología XP se debe llevar a cabo una liberación cada vez que es terminada una iteración, las liberaciones son lanzadas con un producto funcionando en óptimas condiciones, pero esto no ocurre siempre y un ejemplo de un escenario donde no ocurre, es al momento de dividir historias de usuario lo cual conlleva a esperar varias iteraciones, de forma tal que al ser liberado, éste contenga todas las funcionalidades necesarias de la historia de usuario original.

La planificación de la entrega es un esfuerzo unido entre el cliente y el desarrollador, el primero decide qué historias de usuario tienen la mayor prioridad y el segundo estima el tiempo que le llevará implementar las mismas.

Entre los elementos a tener en cuenta en el cronograma de liberación, es fundamental saber que no siempre un plan fluye respecto a lo acordado, por esa razón el plan de entrega está en constante cambio. Un ejemplo de ello se tiene, cada vez que el usuario cambia la prioridad de una historia, cuando el desarrollador divide o une historias de usuario, aparecen imprevistos en las tecnologías a utilizar, entre otras acciones. Es por esto que los cambios deben ser aceptados. El cronograma de entrega de la presente investigación queda conformado según se muestra en la siguiente tabla.

Tabla 7 Plan de entrega

Iteración	**Entrega**
Primera Iteración	3ta semana septiembre 2023
Segunda Iteración	4ta semana noviembre 2023

II.8 Características del sistema.

El sistema propuesto consiste en un sistema informático, el cual tiene como objetivo fundamental la gestión de los procesos extensionistas de la Facultad 2. Dentro de las características del sistema propuesto, se encuentran, como lenguaje de programación Python3 del lado del servidor y Bootstrap5 como framework de programación del lado del cliente. Se utiliza la versión v1.82 del Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) Visual Studio Code y como Sistema Gestor de Bases de Datos (SGBD) el SQLite en su versión 2.3. Todo el desarrollo de la propuesta de solución es guiado a través de un proceso de desarrollo de software con enfoque ágil, en el que se aplican buenas prácticas de los procesos ágiles pues se cuenta con un equipo de desarrollo pequeño, el cliente forma parte de la organización y la rápida entrega de resultados tangibles es quien guía el desarrollo. El sistema contará con varios módulos, a través de los cuales, los usuarios podrán acceder y si su rol lo permite, modificar las siguientes:

1. **Autenticar usuario**: Se encarga de la seguridad del sistema, recogiendo los datos de usuario y contraseña, si la persona que introduce los datos no es un usuario “conocido” o válido, no podrá tener acceso al sistema. En esta funcionalidad se establecen los permisos para cada rol de la aplicación.
2. **Gestionar usuario:** Maneja la información relacionada con los usuarios que accede o forma parte de la aplicación.
3. **Gestión de estudiantes en la residencia**: Permite el control de los estudiantes, teniendo constancia de su ubicación dentro de la Residencia Estudiantil, teniendo en cuenta edificio, apartamento y escalera correspondiente.
4. **Cuartelería:** Posibilita la programación y control de los estudiantes que realizan esta tarea, así como la asignación de una evaluación diaria dependiendo de su desempeño.
5. **Guardia estudianti**l: Permite la gestión de las diferentes secciones de la Residencia Estudiantil, así como insertar, modificar y eliminar los eventos y grupos del mismo. Permite asignar a cada estudiante a una respectiva área y la evaluación correspondiente. Genera un reporte con la información de los estudiantes mostrando la participación de cada uno de ellos y los resultados obtenidos durante la guardia.

6. **Medios Básicos**: Facilita la asignación y monitoreo de los medios básicos activos en la residencia relacionados a los edificios y apartamentos.

Reglas del Negocio

El Grupo de Reglas de Negocio (BRG por sus siglas en inglés) del Grupo de Gestión de Objetos (OMG por sus siglas en inglés), por ejemplo, considera que una Regla de Negocio es una “declaración que define o restringe algún aspecto del negocio” (BRG, 2018). Una definición que brinda una buena explicación de regla de negocio, desde la perspectiva del negocio, afirma que una regla negocio es “una regla que está bajo la jurisdicción del negocio” (BRG, 2018).

Los siguientes son ejemplos de reglas de negocio descritas desde la perspectiva del negocio:

R1: Cuando un artículo alcance su punto mínimo, se debe emitir una solicitud de pedido de ese producto.

R2: Un cliente es moroso cuando, a la fecha debe más de dos facturas vencidas.

R3: Una factura está vencida cuando la fecha actual es mayor, en dos días o más, que la fecha de pago.

R4: A un cliente moroso no se le debe conceder nuevos créditos.

La regla 1 describe cuando ejecutar una acción, las **reglas 2 y 3** define aspectos concretos del negocio y **la regla 4** restringe una acción. Nótese además que las reglas 1, 2 y 3 conforman un conjunto del cual se puede deducir cierto conocimiento con base en las evidencias que se presenten. Sin embargo, para incorporar estos tipos de reglas en el proyecto es necesario, realizar un proceso de transformación reescritura a la luz del negocio y de la tecnología hasta convertirlas en reglas que sean procesables por un motor de reglas (Bocayoya, 2022).

II.9 Arquitectura del sistema.

Aunque está fuertemente inspirado en el Modelo Vista Controlador (MVC) no se observa estrictamente este paradigma. Por ejemplo, lo que se llamaría <controlador> en un

<verdadero> framework MVC se llama en Django <vista>, y lo que se llamaría <vista>se llama <plantilla>.

En un sitio web tradicional basado en datos, una aplicación web espera peticiones HTTP del explorador web (o de otro cliente). Cuando se recibe una petición, la aplicación elabora lo que se necesita basándose en la URL y posiblemente en la información incluida en los datos POST o GET. Dependiendo de qué se necesite, se puede leer o escribir información desde una base de datos o realizar otras tareas requeridas para satisfacer la petición. La aplicación devolverá a continuación una respuesta al explorador web, con frecuencia creando dinámicamente una página HTML para que el explorador la presente insertando los datos recuperados en marcadores de posición dentro de una plantilla HTML.

Las aplicaciones web de Django normalmente agrupan el código que gestiona cada uno de estos pasos en ficheros separados. Django interpreta el modelo MVC en un modelo MVT (Modelo Vista Template). La figura siguiente muestra un esquema de este funcionamiento.

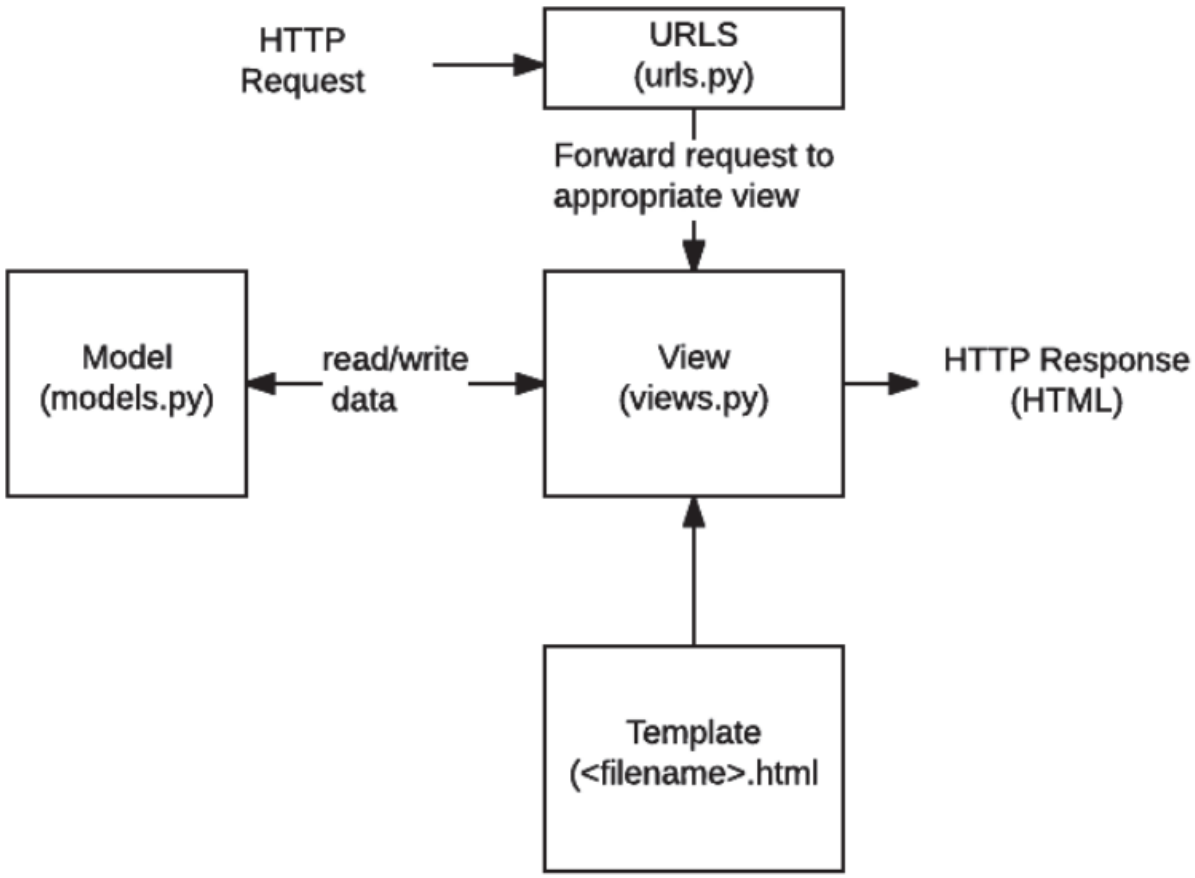


Figura 1: Figura sobre la arquitectura Modelo Vista Template.

II.10 Patrones de diseño.

Los patrones de diseño son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño.

Los patrones de diseño (design patterns) son elementos reutilizables creados para resolver problemas comunes. Es decir que con su aplicación y utilización podremos corregir

diferentes problemas que presenta nuestro código de una manera segura, estable y testeada por cientos de programadores de todo el mundo (Craf-Code, 2021).

Para ello se aíslan los aspectos comunes y su solución y se añaden cuantos comentarios y ejemplos sean oportunos.

Los patrones GRASP, más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Entre ellos, los patrones Experto, Creador, Bajo acoplamiento y Alta cohesión pueden ser tratados en un curso de lógica de programación, porque guardan directa relación con la creación y asignación de responsabilidades a los objetos. Sin embargo, el patrón Controlador, que también conforma los patrones GRASP, se puede estudiar en un curso superior de ingeniería de software o en un laboratorio de programación, dado que sirve como intermediario entre la interfaz de usuario y las clases que encapsulan los datos (Tabares, 2015).

1-Experto.

Problema*:* ¿Cuál es el principio más básico para añadir responsabilidades en una clase?

Solución: Asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para llevar la tarea a cabo**.**

Consecuencias: Encapsulamiento de la información y, por ende, el bajo acoplamiento. El comportamiento distribuido entre las clases, es decir, clases más cohesiva.

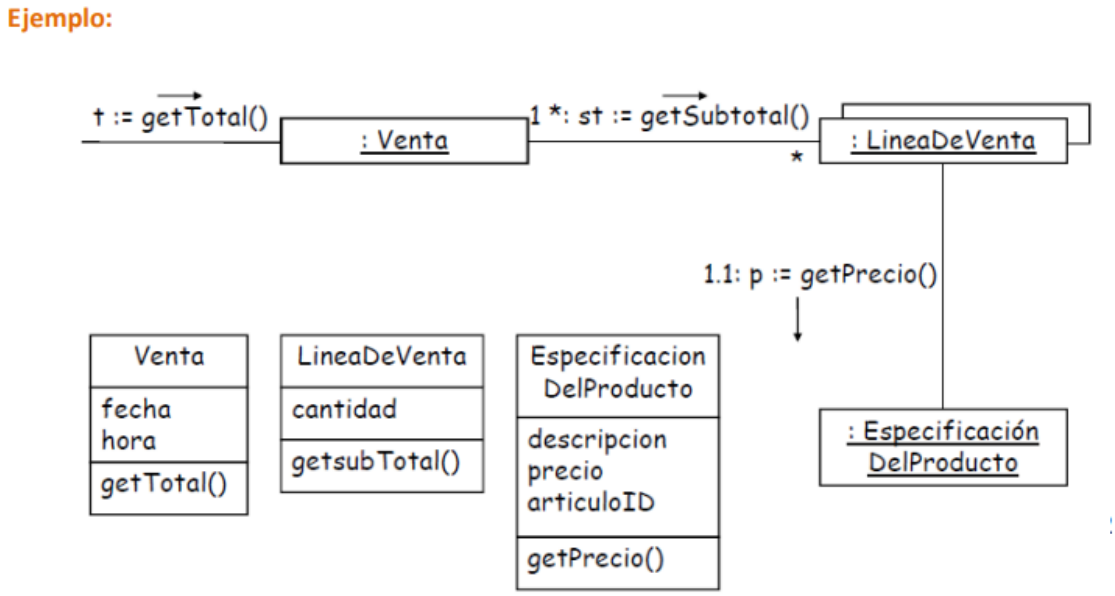


Figura 2 Ejemplo de Patrón Experto

Contraindicaciones: En algunas ocasiones no es una guía deseable sobre todo por problemas de cohesión, acoplamiento y duplicación de código. (Si venta debe ser almacenada en una base de datos, el Experto sugiere a Venta como responsable, pues contiene mucha de la información a almacenar**.**

Patrones relacionados: Bajo Acoplamiento y Alta Cohesión**.**

2-Creador.

Problema: ¿Quién debe ser responsable en la creación de una nueva instancia de una clase?

Solución: Una clase B tiene la responsabilidad para crear una instancia de la clase A si:

- B agrega objetos de A.
- B contiene objetos de A.
- B almacena objetos de A.
- B usa objetos de A.
- B tiene los datos necesarios para inicializar a A cuando este es creado.

Consecuencias: Soporta el bajo acoplamiento, es decir, este no se incrementa.

Ejemplo: En la aplicación PDV, ¿quién debería crear una LineaDeVenta? Venta de hecho contiene (agrega) muchos objetos de LIneaDeVenta, debe tener, por lo tanto, la responsabilidad de la creación de instancias de esta clase y por lo tanto contener un método crearLineaDeVenta**.**

Contraindicaciones: A menudo la creación de instancias es una tarea compleja, en estos casos es aconsejable delegar la creación a una clase auxiliar denominada Factoría.

Patrones relacionados: Bajo Acoplamiento y Factoría.

3-Alta Cohesión.

Problema: ¿Cómo lograr que la complejidad sea lo más manejable posible? Alta cohesión: lo relacionadas que están las responsabilidades de una clase, o una clase con responsabilidades altamente relacionadas y que no lleva a cabo gran cantidad de trabajo**.**

Solución: Asignar responsabilidades procurando que la cohesión sea lo más alta posible**.**

Consecuencias:

- Se incrementa la claridad y facilita la comprensión, se simplifica el mantenimiento.
- Implica casi siempre bajo acoplamiento, incrementando la reutilización.

Patrones relacionados: Bajo Acoplamiento.

Ejemplo: Es una contradicción, es decir, el ejemplo va en contra del patrón de Alta Cohesión.

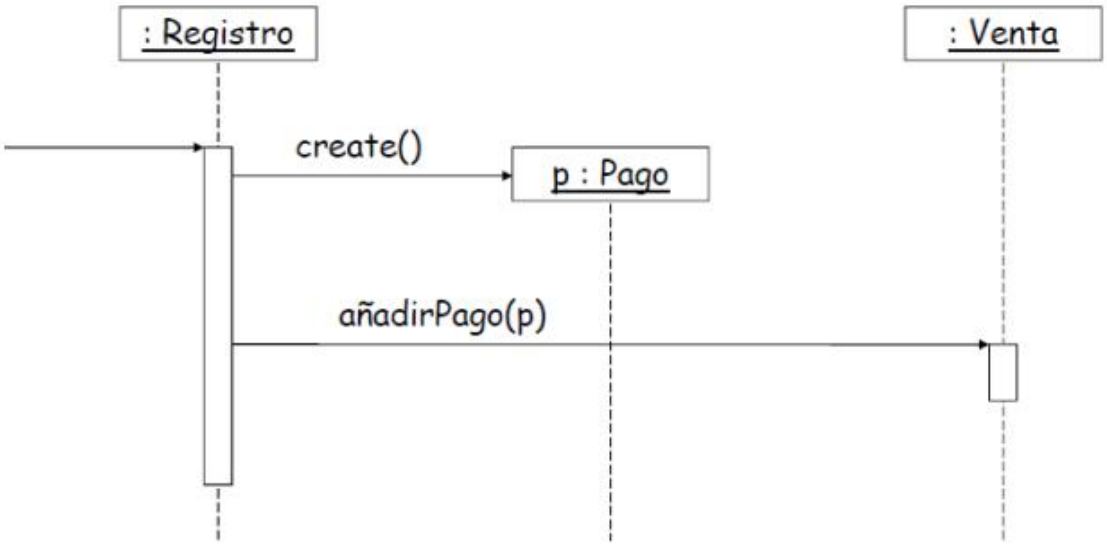


Figura 3 Ejemplo de Patrón de Alta Cohesión.

4-Bajo Acoplamiento.

Problema: ¿Cómo soportar baja dependencia e incrementar la reutilización? Acoplamiento indica que tan fuertemente está conectada una clase con otra, tiene conocimiento de, o influye sobre otra clase, una clase con bajo acoplamiento no depende de otras clases.

Solución: Asignar responsabilidades de tal manera que el acoplamiento sea el menor posible**.**

Consecuencias:

- Mejor compresión de las clases aisladas.
- Facilitan la reutilización de código.
- No afectan los cambios en otros componentes.

Patrones relacionados: Variaciones Protegidas.

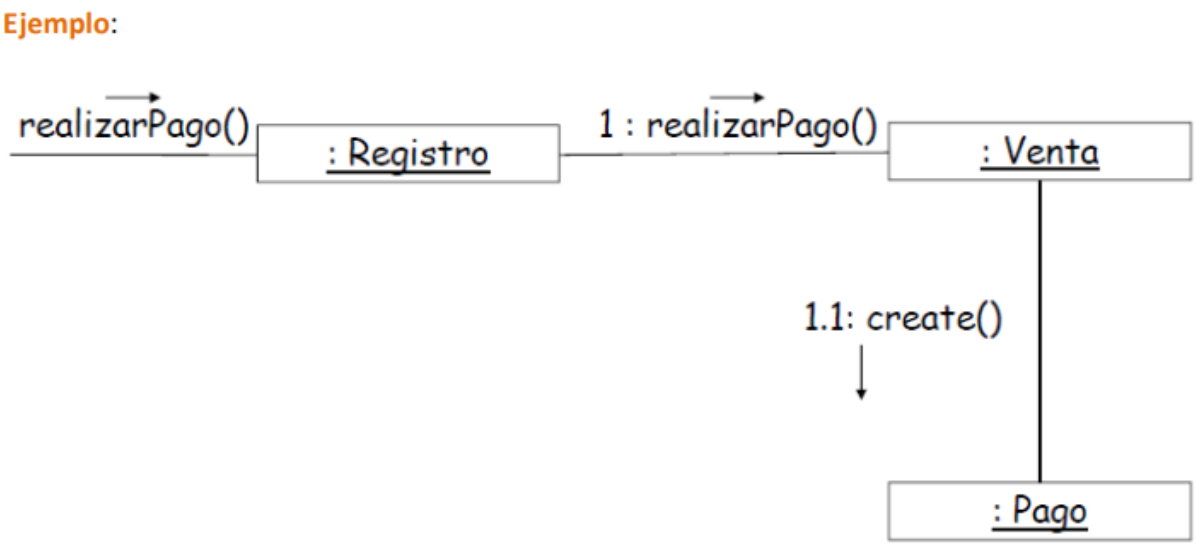


Figura 4 Ejemplo de Bajo Acoplamiento

5- Controlador.

Problema: ¿Quién debe manejar eventos del sistema? Un controlador es un objeto responsable del manejo de los eventos del sistema, que no pertenece a la interfaz del usuario, el controlador recibe la solicitud del servicio desde la capa GUI y coordina su realización delegando a otros objetos.

Solución: Responsabilidades para el manejo de mensajes de eventos del sistema a una clase que: – Representa al conjunto del sistema o negocio (Controlador Fachada). – Representa algo del mundo real que está activo (Controlador de Roles). – Representa un administrador artificioso para todos los eventos del sistema (Controlador Caso de Uso). **Ejemplo**: En el PDV hay varias operaciones del sistema como finalizarVenta(), introducirArticulo() o introducirArticuloDevuelto(). Siguiendo el patrón Controlador podríamos asignar la responsabilidad de manejar estos eventos a una clase que represente al sistema como Registro (Controlador Fachada). O si hay muchos eventos de sistema, también podríamos asignársela a controladores de los distintos casos de uso como:

ProcesarVentaControlador o GestionarDevolucionesControlador. Si el Registro asume demasiadas responsabilidades pierde la cohesión (el compromiso).

Consecuencias:

- Potencial para reutilizar interfaces conectables.
- Razonamiento sobre el funcionamiento de los casos de uso, es decir, valida la secuencia de operaciones, capturando información de estado.

Patrones relacionados: Command, Fachada, Capas, Fabricación Pura.

6-Polimorfismo.

Problema: ¿Cómo manejar alternativas basadas en el tipo? ¿Cómo crear componentes conectables?

Solución: Cuando el comportamiento relacionado varía según el tipo (clase) asigne la responsabilidad para el comportamiento utilizando operaciones polimórficas a los tipos para los que varía el comportamiento.

– **Corolario:** Evítense las comprobaciones acerca del tipo de un objeto y procure no utilizar la lógica condicional.

Consecuencias:

- Se añaden fácilmente extensiones necesarias para nuevas variaciones.
- Las nuevas implementaciones se introducen sin afectar a los clientes.

Patrones relacionados: Variaciones Protegidas, varios de los patrones de diseño GoF como Adaptador, Command, Composite, Proxy, Estado y Estrategia**.**

Ejemplo:

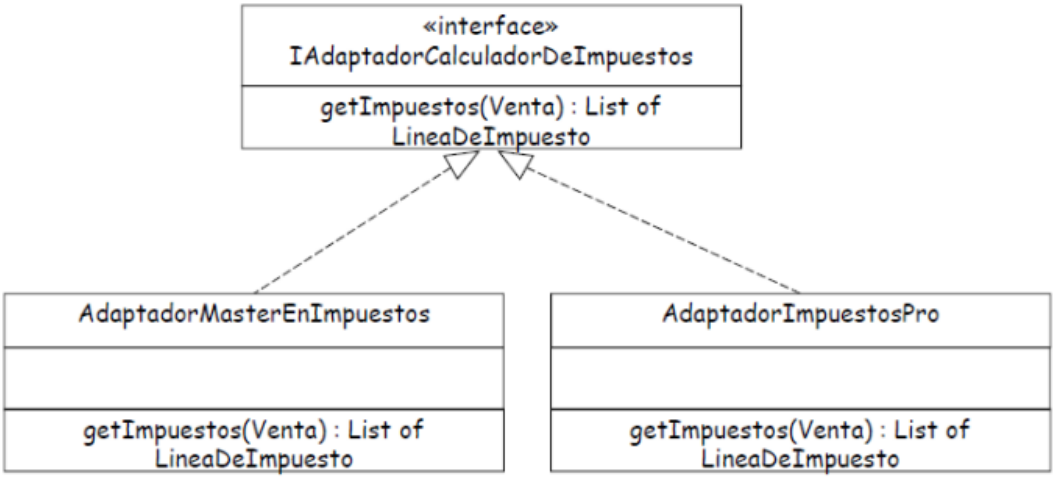


Figura 5 Ejemplo de patrón de Polimorfismo

7-Fabricación Pura.

Problema: ¿Qué objetos deberían tener la responsabilidad cuando no se quiere violar los objetivos de alta cohesión y bajo acoplamiento**?**

Solución: Asigne un conjunto de responsabilidades altamente cohesivas a una clase artificial que no representa un concepto del dominio**.**

Consecuencias: • Se soporta alta cohesión puesto que las responsabilidades se factorizan en una clase de grano fino.

• El potencial para reutilizar aumenta**.**

Ejemplo: – Se necesita soporte para almacenar la Venta en una base de datos se podría suponer según el Experto que la propia clase debería tener la responsabilidad.

– La tarea implica un número amplio de operaciones de bases de datos no relacionadas con las ventas (Baja cohesión).

– Además, tiene que acoplarse con el interfaz de la base de datos.

– Y por último la tarea de almacenar objetos puede ser necesaria para más objetos.

– Podríamos crear una clase AlmacenamientoPersistente**.**

Contraindicaciones: Llevar el principio al extremo**.**

Patrones relacionados: Bajo Acoplamiento, Alta Cohesión, los GoF (Adaptador, Command, Estrategia, etc) y prácticamente el resto de patrones de diseño.

8-Indirección.

Problema: ¿Dónde asignar una responsabilidad para evitar el acoplamiento directo entre dos objetos?

Solución: Asigne la responsabilidad a un objeto intermedio que medie entre otros componentes para que no estén acoplados directamente, el intermediario crea una indirección**.**

Consecuencias:

- Disminuye el acoplamiento entre componentes.

Ejemplos:

– Muchos de los patrones existentes son especializaciones de Indirección.

– Está relacionado con Variaciones Protegidas, Bajo Acoplamiento, muchos GoF como Adaptador, Puente, Fachada, Observador y Mediador.

9-Variaciones Protegidas.

Problema: ¿Cómo diseñar componentes de manera que las variaciones en ellos no tengan repercusión en otros elementos?

Solución: Asigne responsabilidades para crear una interfaz estable alrededor de los puntos de variaciones previstas o inestabilidad**.**

Consecuencias:

- Se añaden fácilmente las extensiones para nuevas variaciones.
- Se pueden introducir nuevas implementaciones sin afectar a los clientes.
- Se reduce el acoplamiento.

Ejemplo: Mediante la Indirección de la interfaz y el polimorfismo se consigue una protección frente a las variaciones en las interfaces externas, integrando el sistema con otros sistemas externos.

Patrones relacionados*:* La mayoría de los patrones son mecanismos de Variaciones Protegidas**.** Se relaciona con Polimorfismo, Indirección y la mayor parte de los GoF.

Patrones GoF

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la pandilla de los cuatro (gang of four). Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además se introducirá más notación UML.

Se clasifican según su propósito en patrones:

Factoría (creación).

Problema: ¿Quién es el responsable de la creación de objetos cuando existen consideraciones especiales, como una lógica de creación compleja o el deseo de separar las responsabilidades de la creación para mejorar la cohesión?

Solución: Cree un objeto Fabricación Pura denominado Factoría (Factoría Concreta) que resuelva la creación cuando existan consideraciones especiales.

Ejemplo: ¿Quién crea Adaptador y cuál debería crearse? Si se encarga a algún objeto de dominio excedería la lógica de la aplicación o disminuiría su cohesión. Resulta mejor asignar a una FactoriaDeServicios dicha responsabilidad, así se crea el Adaptador dinámicamente. De esta manera se podría cambiar de Adaptador sin cambiar código**.**

Consecuencias:

- Separar la responsabilidad de la creación compleja en objetos de apoyo.
- Ocultar la lógica de creación potencialmente compleja.
- Permitir introducir estrategias para mejorar el rendimiento de la gestión de memoria.

Factoría Abstracta (creación) (Creación de objetos).

Problema: ¿Cómo crear familias de clases relacionadas que implementan una interfaz común? Solución: Proporcione una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas. Ejemplo: Aplicación para crear interfaces de usuario que admita múltiples estándares como Motif o Presentation Manager, la Factoría Abstracta define una interfaz para cada una de las utilidades básicas, cada subclase concreta define la utilidad apropiada para su estándar particular.

Aplicabilidad: Un sistema es independiente de cómo sus productos son creados, compuestos y representados. Un sistema debe ser configurado como una familia de productos de entre varias. Una familia de objetos producto relacionados está diseñada para ser utilizada conjuntamente y se necesita hacer cumplir esta restricción. Se quiere proporcionar una librería de productos y se quiere revelar sólo la interfaz no la implementación**.**

Consecuencias:

- Aísla los clientes de las clases concretas de implementación.
- Favorece la consistencia entre productos.
- Pero es difícil soportar nuevas clases de productos.

13-Singleton (creación) (Creación de objetos).

Problema: ¿Cómo podemos asegurar que una clase tiene exactamente una única instancia y que ésta sea fácilmente accesible?

Solución: Defina un método estático de la clase que devuelva el singleton.

Consecuencias:

- Permite el manejo de objetos únicos y que sean accesibles a otros objetos.
- Acceso controlado a la única instancia**.**

Patrones relacionados: Se utiliza a menudo para Factoría y Fachad.

14-Estrategia (comportamiento) (Modo en que las clases y objetos interactúan y se reparten las responsabilidades).

Problema: ¿Cómo Diseñar diversos algoritmos o políticas que estén relacionadas? ¿Cómo diseñar que éstos puedan cambiar?

Solución: Defina cada algoritmo, política o estrategia en una clase independiente con una interfaz común.

Participantes:

- ***Estrategia*.** Interfaz común a todos los algoritmos permitidos.
- ***Estrategia concreta***. Implementa los algoritmos correspondientes, usando la interfaz Estrategia.
- ***Contexto*.** Mantiene una referencia al objeto Estrategia, se configura con un objeto Estrategia Concreta.

Consecuencias:

- Configurar una clase con uno de varios comportamientos posibles.
- Se necesitan diferentes variantes de un algoritmo para un mismo comportamiento.
- Una clase define muchos comportamientos que aparecen como sentencias CASE en sus métodos, elimina sentencias CASE.
- El cliente puede elegir entre diferentes estrategias o implementaciones, pero debe conocer los detalles que las diferencian.

Patrones relacionados: Se crea mediante una Factoría que se accede como Singleton.

15-Composite (estructural) (Composición de clases y objetos).

Problema: ¿Cómo Tratar un grupo o una estructura compuesta del mismo modo que un objeto atómico?

Solución: Defina las clases para los objetos compuestos y atómicos de manera que implementen el mismo interfaz.

Ejemplo: Se puede crear una estrategia que implemente la misma interfaz y que al mismo tiempo englobe varias estrategias, se van añadiendo en distintos puntos.

Participantes:

- Componente. Declara la interfaz para los objetos de la composición e implementa también los métodos que determinan el comportamiento de las diferentes clases tanto compuestas como hijas y permite tener acceso a componentes en la estructura recursiva

. • Hoja. Define el comportamiento de los objetos que no tienen hijos en la composición.

- Compuesto. Maneja los hijos de los diferentes componentes compuestos y determina el comportamiento de aquellos componentes capaces de tener hijos.
- Cliente. Manipula los objetos de la composición a través del componente.

Consecuencias:

- Define jerarquías parte/todo.
- Los clientes ignoran la diferencia entre objetos compuestos y objetos individuales que los forman, pueden tratar objetos primitivos y compuestos de modo uniforme.

- Jerarquía con clases que modelan objetos primitivos y objetos compuestos, se permite composición recursiva.
- Es fácil añadir nuevos tipos de componentes.

16-Fachada (estructural) (Composición de clases y objetos).

Problema: Se requiere una interfaz común para un conjunto de implementaciones o interfaces dispares.

Solución: Defina un único punto de conexión de un subsistema, este objeto fachado presenta una única interfaz unificada y es responsable de colaborar con los clientes. (Controlador de fachada).

Consecuencias:

- Oculta a los clientes los componentes del subsistema.
- Promueve acoplamiento débil entre el subsistema y los clientes.
- No impide que las aplicaciones usen las clases del subsistema.

17-Observador (comportamiento) (Modo en que las clases y objetos interactúan y se reparten las responsabilidades).

Problema: Diferentes tipos de objetos suscriptores están interesados en el cambio de estado o eventos de un emisor y quieren reaccionar cada uno a su manera, además el emisor quiere mantener bajo acoplamiento con los suscriptores. **Solución:** Defina una interfaz “suscriptor” u “oyente” (listener) para que objetos registrados como suscriptores de un emisor la implementen y se les pueda notificar cuando ocurren cambios de estado o eventos del emisor.

Pasos a seguir:

– Se define una interfaz con la operación para actualizar los objetos ante cambios en el emisor.

– Se define la ventana(s) que implementa la interfaz y el método anterior.

– Cuando se inicializa la ventana se le pasa la instancia de la clase de la que está mostrando sus datos.

– La ventana se registra o suscribe a la instancia de la clase por medio de la operación de suscripción definida en la clase.

Consecuencias:

- Proporciona un modo de acoplar débilmente los objetos en términos de comunicación.
- Los emisores conocen a los suscriptores sólo a través de una interfaz y desconocen incluso el número de objetos con que se comunica.
- Los suscriptores pueden registrarse o darse de baja dinámicamente (Ingeniería del Software II, Tema 6,2013).

Despliegue del sistema.

II.11 Tarjetas CRC

Las características más sobresalientes de las tarjetas CRC son su simpleza y adaptabilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema, las cuales permiten también que el equipo completo contribuya en la tarea del diseño. Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema.

La principal funcionalidad que tienen estas, es ayudar a dejar el pensamiento procedimental para incorporarse al enfoque orientado a objetos. Cada tarjeta representa una clase con su nombre en la parte superior, en la sección inferior izquierda están descritas las responsabilidades y a la derecha las clases que le sirven de soporte(Echeverry Tobón & Delgado Carmona, 2007).

El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente, tal y como se muestra a continuación, el resto de las Tarjetas CRC se encuentran en los anexos del presente documento.

Tabla 8 Tarjeta CRC #16

Tarjeta CRC	
Clase: Gestionar Cuarteleria	
Responsabilidad: Listar_Cuarteleria Insertar_Cuarteleria Modificar_ Cuarteleria Eliminar_ Cuarteleria	**Colaboración:** • Estudiante • Apartamento • Edificio

Tabla 9 Tarjeta CRC #20

Tarjeta CRC	
Clase: Gestionar Guardia	
Responsabilidad: Listar_ Guardia Insertar_ Guardia Modificar_ Guardia Eliminar_ Guardia	**Colaboración:** • Estudiante • Grupo_Docente

II.12 Modelo de datos

Una vez que se han diseñado las tarjetas CRC y definidas las clases de entidades, se procede a diseñar el modelo físico de la base de datos que el sistema utilizará para almacenar la información. El modelo de datos permite describir las principales clases que componen la capa del modelo del sistema, los tipos de datos de los atributos que contendrá la base de datos y las relaciones entre las entidades. Facilita la descripción de los elementos de la realidad que participan en un problema específico y cómo se relacionan entre sí. A continuación, se presenta el diagrama entidad-relación que representa el modelo físico de datos diseñado para el sistema.

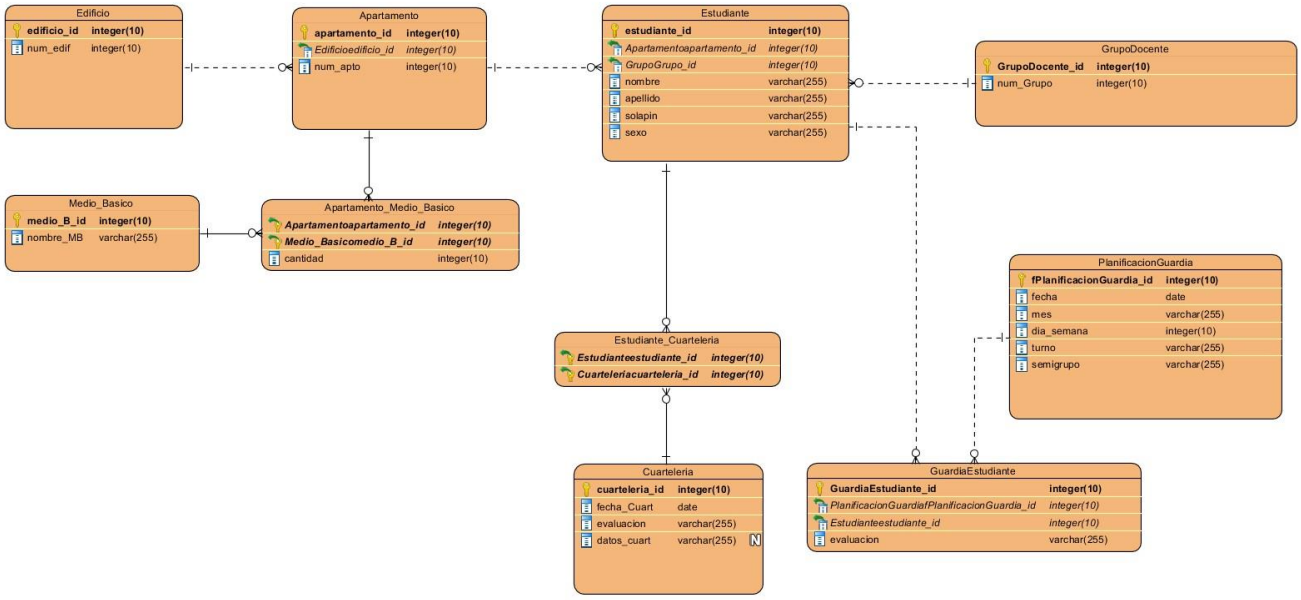


Figura 6 Modelo de la base de datos del sistema.

II.13 Conclusiones del capítulo

- Durante el desarrollo del capítulo fueron analizadas las funcionalidades a incluir en el software a través de las Historias de Usuario, siguiendo el ciclo de desarrollo que propone la metodología XP.
- Se realizó la descripción de las mismas precisando por el cliente la prioridad de cada una, definiendo así el orden en el que serán implementadas.
- Quedó elaborado el modelo necesario para llevar a cabo la implementación del sistema, mediante la descripción de los estándares de codificación y la arquitectura utilizada.
- Se definió por medio del diagrama de despliegue la distribución física, mediante la cual funcionará el Sistema.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

En el presente capítulo se lleva a cabo la descripción de los diferentes mecanismos utilizados para llevar a cabo el desarrollo y validación del sistema propuesto. En la etapa de implementación se traza la estrategia a seguir en cuanto al código, se describen las tareas de ingeniería que es el artefacto que genera la metodología de desarrollo de software ágil XP para detallar las HU descritas en el capítulo anterior. Se describen igualmente las pruebas realizadas y se indican las respuestas de la aplicación en el empleo de las diferentes funcionalidades, así como los posibles mensajes de error. Se utilizan clases diseñadas específicamente para llevar a cabo las pruebas unitarias y funcionales.

III.1 Implementación

La presencia del cliente en las diferentes fases de XP como parte del equipo de trabajo es indispensable. A la hora de implementar una HU es necesaria su presencia pues son ellos los que crean las mismas junto al analista del equipo, ambos negocian los tiempos en los que serán implementadas, especifican detalladamente lo que esta hará y verifican que la historia implementada cumple la funcionalidad especificada cuando se realicen las pruebas. La codificación debe ajustarse a los estándares; y los programadores los deben seguir al pie de la letra. Ello mantiene el código sólido facilitando su comprensión.

III.2 Estándares de codificación

Los estándares de codificación mejoran la seguridad y la protección. El objetivo de los estándares de codificación de software es inculcar prácticas de programación comprobadas que conduzcan a un código seguro, confiable, comprobable y mantenible.

Un estándar de codificación es una serie de reglas que determinan cómo debe escribirse el código. El objetivo es lograr un código fácil de leer por todos (para la computadora mientras funcione todo lo demás da igual) (Asana, 2022).

- Las variables, funciones, paquetes y módulos llevan el nombre de 'lower_with_under' (Minúsculas y subrayado), como model_test.

```
def get_queryset(self):                    (function) get: Any
    edificio_id = self.kwargs.get('edificio_id')
    edificios = Edificio.objects.filter(id=edificio_id)
    apartamentos = Apartamento.objects.filter(edificio__in=edificios)
    return apartamentos
```

Figura 7 Estándar de codificación 1

- Las clases y excepciones llevan el nombre de CapWords (CamelCase: primera letra en mayúscula) como ModelTest.

```
#.............................Usuarios.................

class UserCreateView(LoginRequiredMixin, CreateView):
    model = User
    form_class = UserForm
    template_name = 'G_Users/Add_User.html'
    success_url = reverse_lazy('usuarios')
    permission_required = 'App.view_user'
```

Figura 8 Estándar de codificación 2

III.3 Tareas de ingeniería

Las Tareas de la ingeniería son utilizadas como apoyo a las historias de usuarios. Estas detallan con más profundidad la implementación de las HU, por eso son la base para la

implementación del módulo. Las tareas de ingeniería están descritas mediante tablas que contienen los siguientes campos:

Número de la tarea: Los números han de ser consecutivos.

Número de Historia de Usuario: Número de historia de usuario a la que pertenece la tarea.

Nombre de la tarea: Nombre que identifica la tarea.

Tipo de tarea: Las tareas pueden ser de Desarrollo, Mejora, Corrección, ext.

Puntos estimados: Tiempo estimados en días que se le asignará a su desarrollo.

Programadores responsables: Nombre y apellidos de los programadores.

Descripción: Breve descripción de la tarea.

Tabla 10 Tarea de Ingeniería Gestionar cuartelería

Tarea de Ingeniería	
Número de la tarea: 16	Número de Historia de Usuario: 16
Nombre de la tarea: Gestionar cuartelería	
Tipo de tarea: Desarrollo	Puntos estimados: 2.6
Fecha de inicio: 15/10/2023	Fecha de fin: 29/11/2023
Programador responsable: Javier A. Paret Estrada.	
Descripción: Se muestra una lista de las cuartelerías por edificio con sus respectivos datos y se puede escoger entre modificar los datos o eliminar la cuartelería, dentro de los datos de la cuartelería está la opción de asignar una evaluación al estudiante que la realiza.	

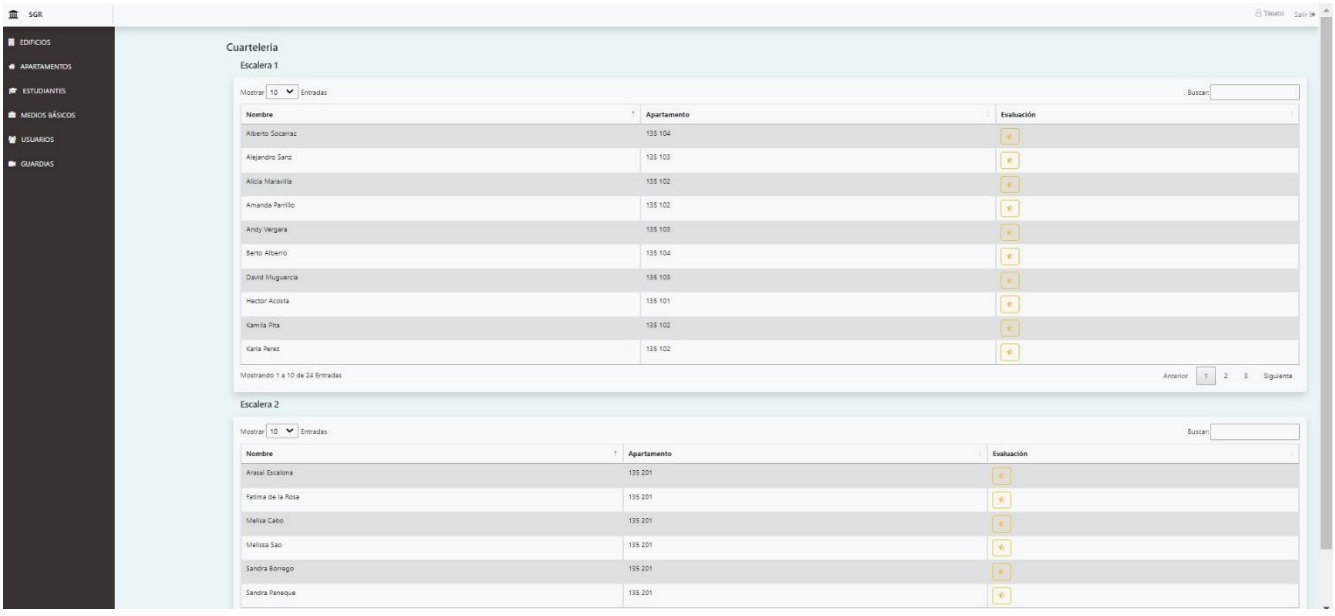


Figura 9 Gestionar Cuartelería

Tabla 11 Tarea de ingeniería Gestionar Guardia

Tarea de Ingeniería	
Número de la tarea: 19	Número de Historia de Usuario: 19
Nombre de la tarea: Gestionar Guardia	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 4/11/2023	Fecha de fin: 20/11/2023
Programador responsable: Héctor E. Pérez Acosta	
Descripción: Se puede generar una planificación de las guardias que se realizarán en el mes y se muestra una lista de las guardias por grupo docente con sus respectivos datos y se puede escoger entre modificar los datos o eliminar la guardia, dentro de los datos de la guardia está la opción de asignar una evaluación al estudiante que la realiza.	

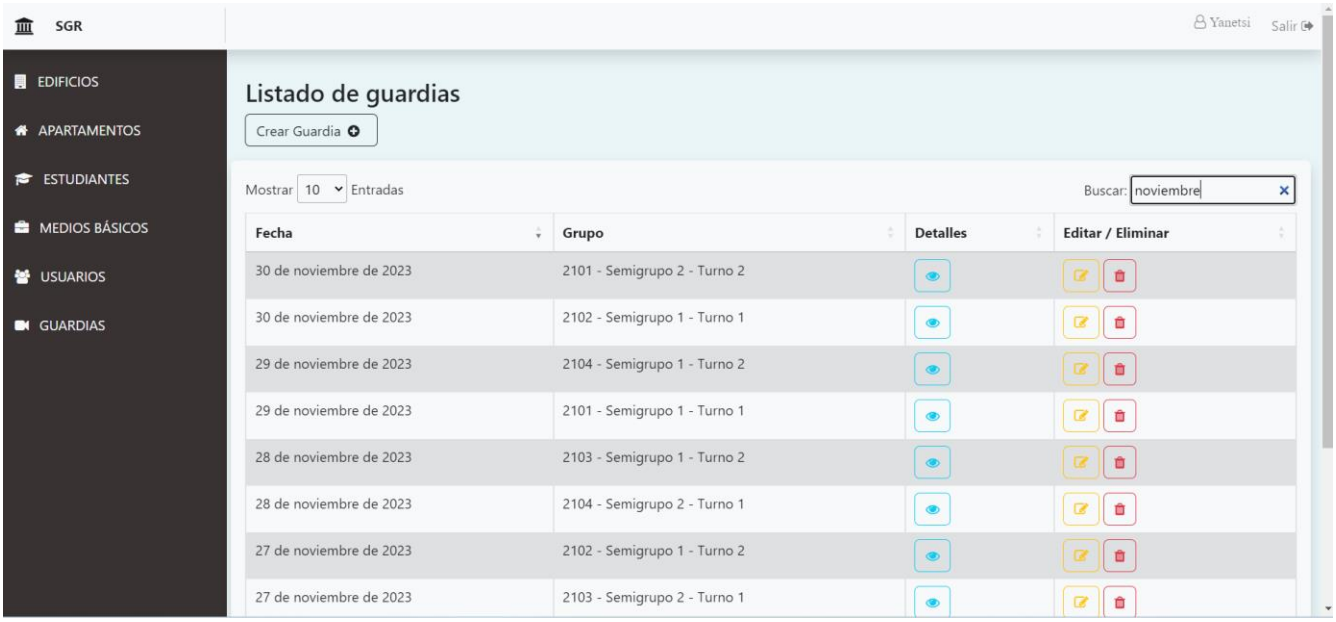


Figura 10 Gestionar Guardia

III.4 Pruebas del software

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se hayan implementado. Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y presenta una revisión final de las especificaciones del diseño y de la codificación. En la metodología XP hay dos tipos de pruebas; las unitarias o

desarrollo dirigido por pruebas (TDD test driven development), desarrolladas por los programadores verificando su código de forma automática, y las pruebas de aceptación, las cuáles son evaluadas luego de culminar una iteración, se verifica así, que se cumplió la funcionalidad requerida por el cliente. Con estas normas se obtiene un código simple y funcional de manera bastante rápida y eficiente. Por esto es importante pasar las pruebas al 100% (Pressman, 2010).

III.5 Estrategia de pruebas

En la metodología XP hay dos tipos de pruebas; las unitarias o desarrollo dirigido por pruebas (TDD test driven development), desarrolladas por los programadores verificando su código de forma automática, y las pruebas de aceptación, las cuáles son evaluadas luego de culminar una iteración, se verifica así, que se cumplió la funcionalidad requerida por el cliente. Con estas normas se obtiene un código simple y funcional de manera bastante rápida y eficiente. Por esto es importante pasar las pruebas al 100% (Pressman, 2010).

III.6 Pruebas Unitarias

Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeñas pruebas que validan el comportamiento de un objeto y la lógica. Las pruebas unitarias suele realizarse durante la fase de desarrollo de aplicaciones de software o móviles. Normalmente las llevan a cabo los desarrolladores. Hay una especie de mito respecto a las pruebas unitarias. Algunos desarrolladores están convencidos de que son una pérdida de tiempo y las evitan buscando ahorrar tiempo. Nada más alejado de la realidad. Con ellas se detectan antes errores que, sin las pruebas unitarias, no se podrían detectar hasta fases más avanzadas como las pruebas de sistema, de integración e incluso en la beta. Realizar pruebas unitarias con regularidad supone, al final, un ahorro de tiempo y dinero (Yeegply, 2022).

Prueba de Caja Blanca

La prueba de caja blanca, denominada a veces “prueba de caja de cristal “es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero

del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; todas las decisiones lógicas en sus vertientes verdadera y falsa; todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2010).

Para la investigación se utiliza la técnica del camino básico, la cual consiste en garantizar que se pueda probar al menos una vez cada una de las sentencias del programa, partiendo de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto básico de caminos de ejecución de este. Los cuales son utilizados para obtener los casos de prueba.

En la figura se muestra el proceso de pruebas realizado a la función form_valid() que se encuentra dentro de la clase ApartamentoFormView(CreateView). Se comienza por analizar el código y enumerar las instrucciones: Para obtener los casos de prueba a partir de la técnica seleccionada, se debe construir seguidamente el grafo de flujo correspondiente al código de la función.

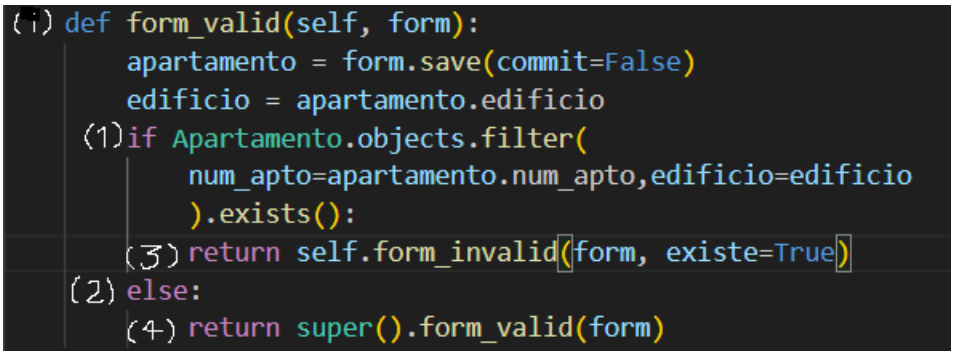


Figura 11 Función Form_valid() y sus nodos

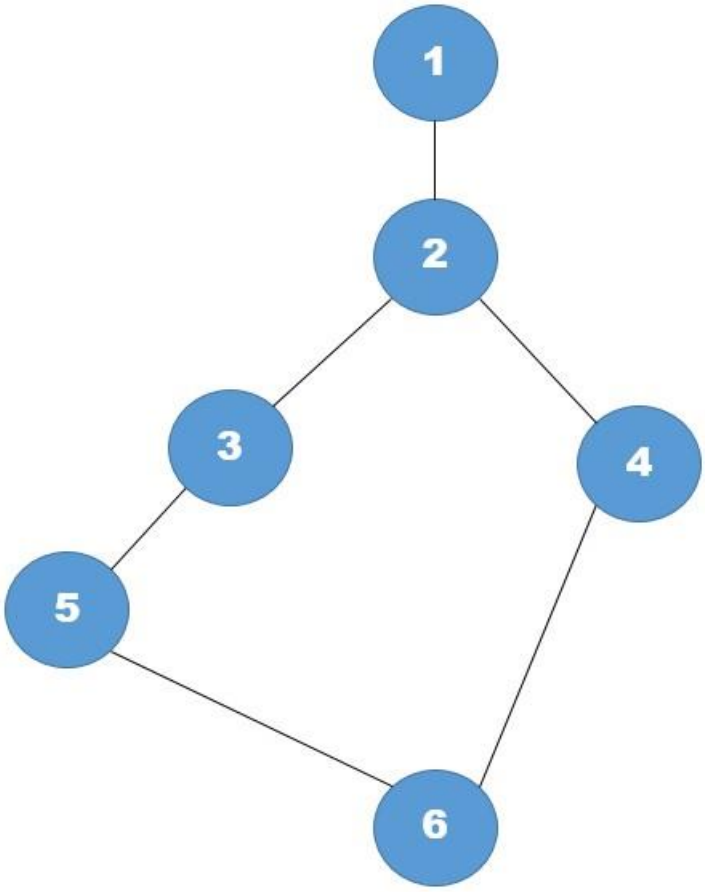


Figura 12 Grafo de la función

Luego se determina la complejidad ciclomática V(G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición.

La complejidad ciclomática puede ser calculada de 3 formas:

1. V(G) = A − N + 2, siendo A el número de arcos o aristas del grafo y N el número de nodos.

2. V(G) = R, siendo R el número de regiones cerradas del grafo.

3. V(G) = P + 1, siendo P el número de nodos de predicados.

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

1. V(G) = A − N + 2 = 6 – 6 + 2 = 2

2. V(G) = R = 2

3. V(G) = P + 1 = 1 + 1 = 2

Por lo que el conjunto de caminos básico es:

Tabla 12 Caminos Básicos del Grafo

No.	**Camino**
1	1, 2, 4, 6
2	1, 2, 3, 5, 6

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación, se muestra el resultado de las pruebas aplicadas a los caminos básicos 1 y 2.

Tabla 13 Caso de Prueba #1

Caso de prueba: Ruta independiente #1	
Código: HU6_P1	Historia de usuario: 6
Nombre: Ruta independiente #1	
Descripción: Llenar el formulario con un apartamento que ya existe dentro del edificio seleccionado.	
Condiciones de ejecución: Llenar el formulario con un valor de apartamento que ya existe.	
Entrada: apartamento = form.save(commit = False) Edificio = apartamento.edificio	
Resultados esperados: Salga un cartel de aviso que diga que ya existe un apartamento con ese número en ese edificio.	

Tabla 14 Caso de Prueba #2

Caso de prueba: Ruta independiente #2	
Código: HU6_P1	Historia de usuario: 6
Nombre: Ruta independiente #2	
Descripción: Llenar el formulario con un apartamento que no existe dentro del edificio seleccionado.	
Condiciones de ejecución: Llenar el formulario con un valor de apartamento que no existe.	
Entrada: apartamento = form.save(commit = False) Edificio = apartamento.edificio	
Resultados esperados: Se inserta el apartamento correctamente.	

Esta técnica se aplicó a cada uno de los métodos implementados, debido a que constituyen las funcionalidades del paquete. En el caso de prueba 1 al ejecutar el método se provee un número de apartamento que ya está en el sistema, lo que provoca el lanzamiento de una excepción, al momento de ejecución, siendo el resultado el esperado.

En el caso de prueba 2, se provee al método de un número de apartamento que está disponible, de tal forma que se obtiene la respuesta esperada, mostrando la información requerida. Una vez ejecutados todos los casos de pruebas obtenidos en esta técnica, se concluye que los mismos fueron probados satisfactoriamente.

III.7 Pruebas de Aceptación

Las pruebas de aceptación en XP, se pueden asociar con las pruebas de caja negra que se aplican en otras metodologías de desarrollo, solo que se crean a partir de las historias de usuario y no por un listado de requerimientos. Durante las iteraciones, las HU se traducen a pruebas de aceptación. En ellas se especifican desde la perspectiva del cliente, los escenarios para probar que la HU ha sido implementada correctamente. La misma puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo que persiguen estas pruebas, es garantizar que las funcionalidades solicitadas por el cliente han sido realizadas. Una HU no se considera completa hasta que no ha transitado por sus pruebas de aceptación. Luego de ver los paradigmas anteriores empleados para la realización de las pruebas y reunirse con el cliente para su análisis, el mismo decidió que se lleve a cabo el proceso mediante las pruebas de aceptación (González, 2014).

Tabla 15 Caso de prueba de Aceptación

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario: 1
Nombre: Gestionar Guardia	
Descripción: Prueba para verificar la funcionalidad de Gestionar Guardia	
Condiciones de ejecución: El usuario debe tener los permisos requeridos.	
Pasos de ejecución: 1. El usuario genera una guardia para un mes específico. 2. El usuario lista las guardias. 3. El usuario selecciona una guardia para modificarla, cambiando cualquier campo disponible dentro del formulario. 4. El usuario selecciona una guardia para eliminarla y acciona en Confirmar.	
Resultados esperados: 1. Resultado Satisfactorio: El usuario realiza todas las operaciones descritas sin detectar errores o inconveniencias. 2. Resultado Fallido: Se introduce algún error relacionado con las funcionalidades descritas o al usuario no lo satisface algo en el Gestionar Guardia.	
Evaluación de la prueba: Satisfactoria.	

Figura 13 Estudiantes por guardia

Resumen de los resultados de las pruebas de aceptación

Las pruebas de aceptación se realizan generalmente al final de cada ciclo de desarrollo para garantizar que sistema entregado cumpla con las expectativas del cliente y se ajuste a las especificaciones establecidas. Las iteraciones sucesivas se enfocan en identificar y corregir problemas detectados en las pruebas de aceptación, incluyendo errores de ortografía, validación y funcionalidad del sistema.

1ra Iteración

En la primera iteración, se identificaron un total de 8 no conformidades (NC): 2 relacionados con ortografía, 4 de validación y 2 de funcionalidad. Se llevaron a cabo 15 pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

2da Iteración

En la última iteración, no se registraron errores no conformidades (NC). Se llevaron a cabo 6 pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

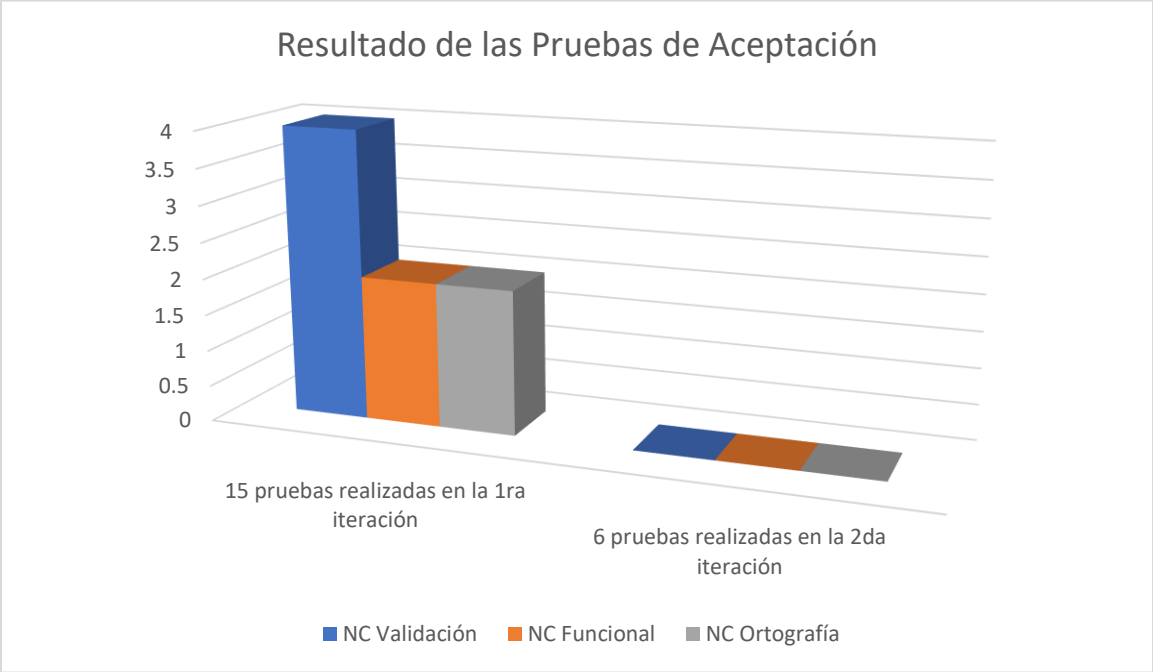


Figura 14 Resultado de las Pruebas de Aceptación

A medida que se avanzaba en el desarrollo del sistema, se encontraron menos problemas en las pruebas de aceptación. Al finalizar todas las pruebas de aceptación, se confirmó que el producto es maduro y estable, lo que indica que se ha logrado una alineación exitosa con los requisitos y expectativas del cliente.

III.8 Validación del Sistema

Al terminar las pruebas, se analizan los resultados obtenidos. El esquema diseñado puso a prueba todas las funcionalidades y lanzaron resultados exitosos sobre los test de implementación. La seguridad de la plataforma funciona correctamente en cada aspecto del sistema y gracias a las pruebas ahora queda corroborada la validez del proyecto. Los métodos utilizados permitieron revisar que funcionan correctamente los métodos implementados y ofrecieron la información necesaria para corregir los errores encontrados.

III.9 Conclusiones del capítulo

- La realización de las tareas de ingeniería proporcionó solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema.
- Se realiza el desarrollo de las iteraciones a partir de la distribución de tareas por HU, y se les practicó las pruebas de aceptación a las funcionalidades de mayor importancia.
- El uso de un estándar de codificación permitió una mejor legibilidad de la aplicación y que está fuera de fácil entendimiento para posteriores desarrolladores de este sistema en versiones futuras.
- La culminación del desarrollo de las pruebas unitarias y de aceptación con resultados satisfactorios garantizó una buena calidad del sistema.
- El enfoque de caja negra permite desafiar al sistema para obtener los resultados esperados en todo momento.

CONCLUSIONES FINALES

Una vez finalizada la investigación, los objetivos inicialmente concebidos se satisfacen, por lo que es posible arribar a las siguientes conclusiones:

- ✓ El análisis de las herramientas informáticas a utilizar en la implementación permitió definir el ambiente de desarrollo para la realización de la propuesta de solución.
- ✓ La selección de la metodología XP permitió la guía del proceso y posibilitó la creación de los artefactos necesarios para la realización del sistema con la mejor calidad posible.
- ✓ La realización de una estrategia de pruebas permitió la detección de no conformidades y posteriormente la corrección de las mismas partiendo del cumplimiento de los requisitos definidos por el cliente y entregándosele una solución con la calidad requerida.

RECOMENDACIONES

En correspondencia con las conclusiones abordadas en el proyecto se recomienda:

- Analizar las funcionalidades existentes y realizarle mejoras en caso de ser necesarias o aplicables a otras tecnologías como las móviles.
- Implementar nuevas funcionalidades que faciliten la gestión de los proyectos que se realizan en la residencia como crear un apartado de incidencias, crear funcionalidades que mejoren la comunicación con los estudiantes, como enviarle correos informativos, o un chat para plantear insatisfacciones.

REFERENCIAS BIBLIOGRÁFICAS

Dávila Pérez, Y., Martínez Padrón, Y., & Cabrera González, Y. (2008). *Sistema de Gestion de Informacion de la Facultad 8. Modulo para la Gestion de la Residencia Estudiantil Version 2.* [bachelorThesis]. https://repositorio.uci.cu/jspui/handle/ident/TD_1134_08

Echeverry Tobón, L. M., & Delgado Carmona, L. E. (2007). *Caso práctico de la metodología ágil XP al desarrollo de software*. https://hdl.handle.net/11059/794

Hernández Alfonso, A., Sánchez González, A. M., & Cañizares Rivera, I. (2015). *Solución Informática para la gestión de solicitudes en el Sistema de Gestión de Residencia.* https://repositorio.uci.cu/jspui/handle/ident/8634

Jenny Ruiz de la Peña. (2010). *Universidad de Holguín.pdf*.

Lazo, A. T., & Botero, J. G. G. (2016). Especificación de requisitos de software: Una mirada desde la revisión teórica de antecedentes. *Entre Ciencia e Ingeniería*, *10*(19), 108-115.

Lesvy Alemán Mateo. (2021). *Sistema informático para la gestión de estudiantes becados de la Universidad de Ciego de Ávila*.

Letelier, P. (2006, abril 15). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)* [Artículo]. www.cyta.com.ar/ta0502/v5n2a1.htm; Técnica Administrativa issn:1666-1680. http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

Lidia del Carmen Leyva Feijoó & Alejandro López Rodríguez. (2015). *Gestión de los procesos extensionistas de la Facultad 1.pdf*.

Martín Socas, S. (2022). *Aplicación web progresiva dedicada a gestión de incidencias para la Residencia Universitaria de Tafira*. https://accedacris.ulpgc.es/jspui/handle/10553/113267

Matute, S., Avila Pesantez, D., & Avila, L. (2020). Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso. *Revista Peruana de Computación y Sistemas*, *3*, 1-10. https://doi.org/10.15381/rpcs.v3i2.19256

Regueyra Edelman & María Gabriela. (2010). *Gestionar Residencia*. http://localhost:8000/edif/

Rodríguez Rodríguez, E., & Fernández Guevara, Z. (2016). *Sistema para la planificación y control de la residencia estudiantil en la Facultad 5."* [bachelorThesis]. https://repositorio.uci.cu/jspui/handle/ident/9307

Rodríguez Viera, L., Rojas Escobar, E., & Rosabal Espinosa, G. Y. (2015). *"Sistema automatizado para la gestión de inventario de Activos Fijos Tangibles en la empresa UEB CAI Bartolomé Masó Márquez"* [bachelorThesis]. https://repositorio.uci.cu/jspui/handle/ident/8914

Romero, E. M. (2019). *Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería*.

Tabares, R. B. (s. f.). *Patrones Grasp y Anti-Patrones: Un Enfoque Orientado a Objetos desde Lógica de Programación*.

ANEXOS

Historias de Usuario

Historia de Usuario	
Número:1	Nombre: Autenticar Usuario
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 0.2	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: Permitir acceder al sistema una vez registrado correctamente para ingresar con los permisos asignados al usuario en cuestión	
Observaciones: Debe requerirse solo en nombre de usuario y su contraseña para una mayor seguridad.	

Historia de Usuario	
Número: 11	Nombre: Gestionar edificio
Usuario: Vicedecana de Extensión y Residencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 0.8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: En este proceso el responsable podrá gestionar el edificio señalado de la residencia. La gestión incluye acciones como: añadir, modificar, eliminar y visualizar la información del edificio.	
Observaciones:	

Historia de Usuario	
Número: 7	Nombre: Gestionar Apartamento
Usuario: Vicedecana de Extensión y Residencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos del apartamento. La gestión incluye acciones como: añadir, modificar, eliminar y visualizar la información de los apartamentos.	
Observaciones:	

Historia de Usuario	
Número: 10	Nombre: Gestionar Estudiante
Usuario: Vicedecana de Extensión y Residencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos del estudiante. La gestión incluye acciones como: añadir, modificar, eliminar, listar y visualizar la información de los estudiantes.	
Observaciones:	

Historia de Usuario	
Número: 8	Nombre: Gestionar Grupo Docente
Usuario: Vicedecana de Extensión y Residencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos del grupo docente. La gestión incluye acciones como: añadir, modificar, eliminar, y visualizar la información del grupo docente.	
Observaciones:	

Historia de Usuario	
Número: 13	Nombre: Gestionar Medio Básico
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,6	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: En este proceso el responsable podrá gestionar los datos del medio básico. La gestión incluye acciones como: añadir, modificar, eliminar, listar y visualizar la información del medio básico.	
Observaciones:	

Historia de Usuario	
Número: 7	Nombre: Buscar Apartamento
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0.8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar un apartamento entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 5	Nombre: Buscar Edificio
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar un edificio entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 3	Nombre: Buscar Usuario
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar un usuario entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 2	Nombre: Gestionar Usuario
Usuario: Vicedecana de Extensión y Residencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 3	Iteración asignada: 1
Programador responsable: Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos del usuario. La gestión incluye acciones como: añadir, modificar, listar y visualizar la información del usuario.	

Observaciones:

Historia de Usuario	
Número: 9	Nombre: Buscar Grupo Docente
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar un grupo docente entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 11	Nombre: Buscar Estudiante
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El sistema permitirá buscar un estudiante entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 14	Nombre: Buscar Medio Básico
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar un medio básico entre los ya existentes, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 18	Nombre: Buscar Cuartelería
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1

Programador responsable: Héctor Eduardo Pérez Acosta
Descripción: El sistema permitirá buscar una cuartelería entre las ya registradas, en caso de no encontrarse algún resultado se le notificará al usuario activo.
Observaciones:

Historia de Usuario	
Número: 21	Nombre: Buscar Guardia
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 0,8	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: El sistema permitirá buscar una guardia entre las ya registradas, en caso de no encontrarse algún resultado se le notificará al usuario activo.	
Observaciones:	

Historia de Usuario	
Número: 13	Nombre: Gestionar Medio Básico en un Apartamento
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,6	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: En este proceso el responsable podrá gestionar los datos del medio básico por apartamento. La gestión incluye acciones como: añadir, modificar, eliminar, listar y visualizar la información del medio básico.	
Observaciones:	

Historia de Usuario	
Número: 17	Nombre: Asignar Evaluación a la Cuartelería
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,6	Iteración asignada: 1
Programador responsable: Javier A. Paret Estrada	
Descripción: En este proceso el responsable podrá asignar una evaluación a la cuartelería realizada por el estudiante.	
Observaciones:	

Historia de Usuario

Número: 20	Nombre: Asignar Evaluación a la Guardia
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,6	Iteración asignada: 1
Programador responsable: Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá gestionar los datos del medio básico por apartamento. La gestión incluye acciones como: añadir, modificar, eliminar, listar y visualizar la información del medio básico.	
Observaciones:	

Historia de Usuario	
Número: 12	Nombre: Listar Estudiante en Apartamento
Usuario: Todos	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1,6	Iteración asignada: 1
Programador responsable: Héctor Eduardo Pérez Acosta	
Descripción: En este proceso el responsable podrá tener presente una de los estudiantes por apartamento.	
Observaciones:	

Tareas de Ingeniería

Tarea de Ingeniería	
Número de la tarea: 3	Número de Historia de Usuario: 3
Nombre de la tarea: Buscar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 4/10/2023	Fecha de fin: 6/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘Usuario. El sistema muestra un listado de todos los usuarios existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 4	Número de Historia de Usuario: 5
Nombre de la tarea: Buscar edificio	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 7/10/2023	Fecha de fin: 8/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘Edificio’. El sistema muestra un listado de todos los edificios existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 5	Número de Historia de Usuario: 7
Nombre de la tarea: Buscar apartamento	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 8/10/2023	Fecha de fin: 10/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘apartamento’. El sistema muestra un listado de todos los apartamentos existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 6	Número de Historia de Usuario: 9
Nombre de la tarea: Buscar grupo docente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 11 /10/2023	Fecha de fin: 13/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘grupo docente. El sistema muestra un listado de todos grupos docentes los existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 7	Número de Historia de Usuario: 11
Nombre de la tarea: Buscar estudiante	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 14/10/2023	Fecha de fin: 15/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘estudiante’. El sistema muestra un listado de todos los estudiantes existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 8	Número de Historia de Usuario: 14
Nombre de la tarea: Buscar medio básico	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 16/10/2023	Fecha de fin: 17/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción ‘Medio Básico’. El sistema muestra un listado de todos los medios básicos existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería

Número de la tarea: 9	Número de Historia de Usuario: 18
Nombre de la tarea: Buscar cuartelería	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 18/10/2023	Fecha de fin: 20/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta.	
Descripción: La instructora o administrador selecciona en el lateral la opción 'Edificio'. El sistema muestra un listado de todas las cuartelerías existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 10	Número de Historia de Usuario: 21
Nombre de la tarea: Buscar guardia	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 21/10/2023	Fecha de fin: 23/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: La instructora o administrador selecciona en el lateral la opción 'Guardia. El sistema muestra un listado de todas las guardias existentes en la base de datos, donde encontrará un cuadro de búsqueda para filtrar el listado por cualquier campo.	

Tarea de Ingeniería	
Número de la tarea: 11	Número de Historia de Usuario: 2
Nombre de la tarea: Gestionar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 24/10/2023	Fecha de fin: 25/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El administrador selecciona del lateral la opción 'Usuario' escoger la opción "*Eliminar*", donde el sistema muestra un mensaje de confirmación, en caso de confirmar la eliminación se selecciona el botón '*Confirmar*', en caso contrario el botón '*Cancelar*'.	

Tarea de Ingeniería	
Número de la tarea: 12	Número de Historia de Usuario: 4
Nombre de la tarea: Gestionar edificio	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 26/10/2023	Fecha de fin: 27/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta.	
Descripción: El administrador selecciona del lateral la opción 'Edificio', donde se puede *'Añadir'*. Se introducen los datos y seguido a esto se selecciona el botón '*Confirmar*', para registrarlo en la base de datos.	

Tarea de Ingeniería	
Número de la tarea: 13	Número de Historia de Usuario: 8
Nombre de la tarea: Gestionar grupo docente	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4

Fecha de inicio: 28/10/2023	Fecha de fin: 29/10/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El administrador selecciona del lateral la opción correspondiente, donde se puede '*Añadir'*. Se introducen los datos y seguido a esto se selecciona el botón '*Confirmar*', para registrarlo en la base de datos.	

Tarea de Ingeniería	
Número de la tarea: 14	Número de Historia de Usuario: 10
Nombre de la tarea: Gestionar estudiante	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 1/11/2023	Fecha de fin: 2/11/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El administrador selecciona del lateral la opción 'Estudiante*.'*, selecciona un estudiante del listado y escoge la opción "*Editar*", luego el sistema muestra los datos del estudiante, brindando la posibilidad de actualizarlos. Terminado los cambios, se selecciona el botón '*Confirmar"* para actualizar la información en la base de datos.	

Tarea de Ingeniería	
Número de la tarea: 15	Número de Historia de Usuario: 13
Nombre de la tarea: Gestionar medio básico	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 2/11/2023	Fecha de fin: 4/11/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El administrador selecciona del lateral la opción 'Medio Básico*'*, selecciona un medio básico del listado y escoge la opción "*Editar*", luego el sistema muestra los datos del e medio básico., brindando la posibilidad de actualizarlos. Terminado los cambios, se selecciona el botón '*Confirmar"* para actualizar la información en la base de datos.	

Tarea de Ingeniería	
Número de la tarea: 16	Número de Historia de Usuario: 16
Nombre de la tarea: Gestionar cuartelería	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 5/11/2023	Fecha de fin: 6/11/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta	
Descripción: El administrador selecciona del lateral la opción 'Edificio' escoge la opción "*Eliminar*", donde el sistema muestra un mensaje de confirmación, en caso de confirmar la eliminación se selecciona el botón '*Confirmar*', en caso contrario el botón '*Cancelar*'.	

Tarea de Ingeniería	
Número de la tarea: 17	Número de Historia de Usuario: 19
Nombre de la tarea: Gestionar guardia	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 5/11/2023	Fecha de fin: 6/11/2023
Programador responsable: Javier A. Paret Estrada y Héctor Eduardo Pérez Acosta.	

Descripción: El administrador selecciona del lateral la opción ‘Guardia’ escoge la opción “*Eliminar*”, donde el sistema muestra un mensaje de confirmación, en caso de confirmar la eliminación se selecciona el botón ‘*Confirmar*’, en caso contrario el botón ‘*Cancelar*’.

Tarjetas CRC

Tarjeta CRC	
Clase: Autenticar Usuario	
Responsabilidad: Se introduce el usuario y la contraseña para acceder al sistema.	**Colaboración:** • Todo

Tarjeta CRC	
Clase: Usuario	
Responsabilidad: Listar_ Usuario Insertar_ Usuario Modificar_ Usuario Eliminar_ Usuario	**Colaboración:** • Grupos

Tarjeta CRC	
Clase: Apartamento	
Responsabilidad: Listar_ Apartamento Insertar_ Apartamento Modificar_ Apartamento Eliminar_ Apartamento	**Colaboración:** • Edificio • Medio básico • Estudiante • Cuartelería

Tarjeta CRC	
Clase: Guardia	
Responsabilidad: Listar_ Guardia Insertar_ Guardia Modificar_ Guardia Eliminar_ Guardia	**Colaboración:** • Guardia • Estudiante • Grupo Docente

Tarjeta CRC	
Clase: Cuartelería	
Responsabilidad:	**Colaboración:**

Listar_ Cuartelería Insertar_ Cuartelería Modificar_ Cuartelería Eliminar_ Cuartelería	• Cuartelería • Estudiante • Edificio • Apartamento

Tarjeta CRC	
Clase: Medio básico	
Responsabilidad: Listar_ Medio básico Insertar_ Medio básico Modificar_ Medio básico Eliminar_ Medio básico	**Colaboración:** • Medio básico • Apartamento • Edificio

Tarjeta CRC	
Clase: Grupo Docente	
Responsabilidad: Listar_ Grupo Docente Insertar_ Grupo Docente Modificar_ Grupo Docente Eliminar_ Grupo Docente	**Colaboración:** • Estudiante

Tarjeta CRC	
Clase: Estudiante	
Responsabilidad: Listar_ Estudiante Insertar_ Estudiante Modificar_ Estudiante Eliminar_ Estudiante	**Colaboración:** • Grupo Docente • Apartamento • Guardia