Facultad 4

MÓDULO DE AGENDA VIRTUAL PARA UN ASISTENTE PERSONAL VIRTUAL

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

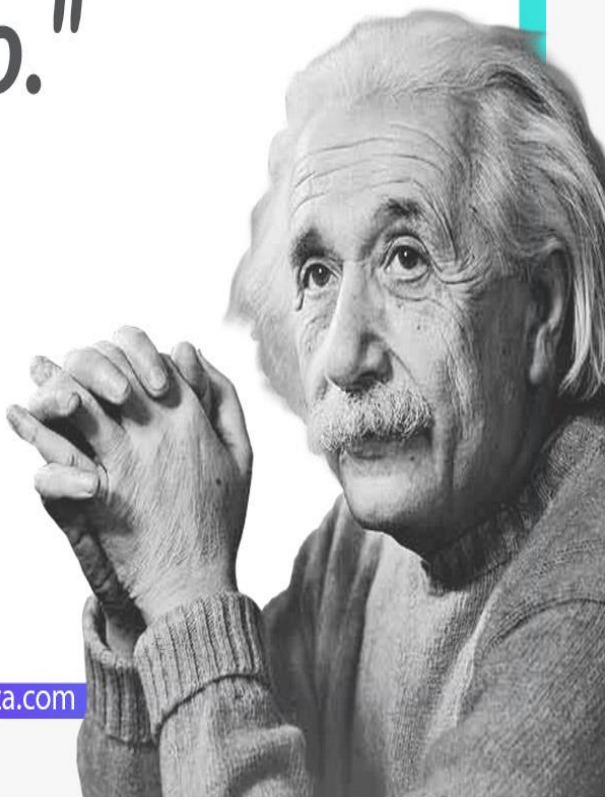
Autora: Geidy Alvarez Gómez
Tutores: MSc. Sailyn Salas Hechavarria
Ing. David Pino González

La Habana, diciembre de 2022

Año 64 de la Revolución

"El genio se hace con un 1% de talento y un 99% de trabajo."

–Albert Einstein



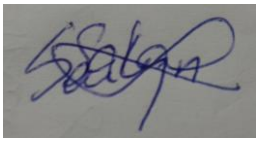
mayneza.com

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título "**Módulo de agenda para un asistente personal**", concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 8 días del mes de diciembre del año 2022.

_____
Geidy Alvarez Gómez
Firma del Autor

_____
Ing.David Pino Gonzáles
Firma del Tutor



_____
Msc.Sailyn Salas Hechevarria
Firma del Cotutor

DATOS DE CONTACTO

Sailyn Salas Hechavarria: Profesora Auxiliar de la Facultad 4 de la Universidad de las Ciencias Informáticas. Máster en Calidad de Software. Imparte docencia en la disciplina de Matemática Aplicada, investiga, participa en eventos científicos y tiene varias publicaciones tanto naciones como internaciones.

David Pino Gonzáles: Graduado de nivel: Universitario
Especialidad: Ingeniería en Ciencias Informáticas
Septiembre 2016 – noviembre 2019 fecha: Universidad de las Ciencias Informáticas. Facultad 4. Centro de Informática Industrial (CEDIN). Especialista B en Ciencias Informáticas.
Noviembre 2019 – febrero 2021: Universidad de las Ciencias Informáticas. Facultad 4. Docente José Antonio Echeverría. Centro de Tecnologías Interactivas. Especialista B en Ciencias Informáticas. Asesor de Seguridad Informática.
Febrero 2021 – hasta la fecha: Universidad de las Ciencias Informáticas. Facultad 4. Docente José Antonio Echeverría. Centro de Tecnologías Interactivas. Especialista A en Ciencias Informáticas. Jefe de proyecto del Asistente de voz personal.
Desde la fecha en que comenzó a trabajar en el CEDIN actual Centro de Tecnologías Interactivas se desempeña en el rol de desarrollador participando así en el desarrollo, despliegue y mantenimiento de diferentes sistemas informáticos.

AGRADECIMIENTOS

A mis tutores David Pino y Sailyn Salas por guiarme durante todo este proceso de la tesis.

A todos los profesores que me ayudaron a lo largo de esta carrera, porque se, que si no fuera por muchos de ellos no estaría donde estoy ahora. En especial a Yadira, Yasiry, Rafael, Luis Manuel y Julio.

A todos los buenos amigos que hice aquí en la UCI por formar parte de mi vida, ayudarme a distraerme cuando más lo necesitaba y sobre todo por aguantar todos mis cambios de humor y malas caras. En especial a Arlin, aunque se pusiera peor que mi mamá cuando me enfermaba, a Marco el pirata, a Manuel que cuando lo conocí no lo podía ni ver, le agradezco por hacerme reír con sus ocurrencias aunque a la misma vez me dieran ganas de matarlo, a Valentina, Mercadet, Yessica, David, Marlin, Amelia la diabla, Leinier, Felix y a todos los cangrejos (Daryan por aguantarme todos los días que me aburria y el pobre sufría las consecuencias, Christian con su frase célebre *A qué huele* y Frank con su amigo el de los principios). Y al Nene por ser mi mayor apoyo y por siempre haber estado ahí en todos estos años de universidad, porque sino fuera por ti hoy no estaría aquí, así que, se puede decir que este logro es de los dos.

A Yesenia por ser una buena amiga y quererme tanto a pesar de ser mi madrastra.

A mi padrastro Tomás por quererme como una hija más y estar ahí siempre que tenía algún problema.

A mi cuñado Yasser por adaptarse tan rápido a la idea de ser familia y quererme desde un principio, por servirme de taxi todo este tiempo y por no mandarme bien lejos cuando nos juntábamos las 4G y nos poníamos intensas.

A mi hermana, que más que hermana es mi mejor amiga, por aguantarme todos estos años que se que no ha sido fácil. Por todos esos consejos que, aunque no los pedía, ella sabía que los necesitaba. Por todo el amor y cariño que nos tenemos a pesar de las peleas de todos los días por cualquier tontería, incluyendo a mostachón. Y sobre todo por haberme dado a esa sobrina que amo como si fuera mi hija y a la cual también le agradezco por hacerme olvidar todos mis problemas con solo darme un abraso y decirme tía.

A mi *Papi* querido que, aunque hoy no se encuentra aquí sabe que lo amo con la vida y que soy quien soy gracias a todo el apoyo que me ha dado. Por ser mi amigo y darme esa confianza de poder contarle todo, aunque a veces se lo tomara muy en serio.

Y, por último, pero no menos especial a la mujer que le debo mi vida, mi madre querida. Sabes que este título es para ti, que, aunque parezca difícil de creer te has esforzado tanto o más que yo para conseguirlo. Gracias por todo el amor incondicional, las noches de desvelo ya sea por cuidarme cuando estaba enferma o por preocupación porque había salido, casi siempre por esta última. Por ser la mejor amiga que tendré nunca, a la que puedo contar todos mis problemas y errores porque sé que me aconsejaras lo mejor, aunque venga con un regaño detrás. En fin, no me alcanza la vida para agradecerte, así que hoy solo me queda darte las gracias y decirte que te amo con la vida.

DEDICATORIA

A mis padres Dora Gómez Estrada y Roberto Alvarez Carraceo por ser las personas más importantes en mi vida a las cuales amo y dedico todos mis logros.

A mi hermana Geisy Gómez Gómez por ser mi mejor amiga y por darme su cariño y apoyo incondicional.

A mi sobrina Alma Miranda Gómez por ser la luz de mi vida.

RESUMEN

El desarrollo científico y tecnológico alcanzado por la humanidad en la rama de la informática, ha permitido que las condiciones estén creadas para agilizar los procesos de gestión del tiempo. El presente trabajo aborda los principales aspectos que se desarrollaron con el fin de realizar un módulo de agenda para gestionar las tareas de una persona. En la investigación realizada se presenta un estudio del entorno de desarrollo Visual Studio Code, el servidor de aplicación Nginx, entre otras herramientas y tecnologías utilizadas en el proceso de la creación de la solución. Por otro lado, se especifican las pruebas a las que fue sometida la solución elaborada, quedando demostrado que la misma cumple con las funcionalidades requeridas por el cliente. El resultado de esta investigación deja evidencia de la metodología AUP versión UCI en su escenario número 4 donde se utilizan las historias de usuario para describir los requisitos funcionales; utilizada para dar cumplimiento al objetivo general propuesto.

Palabras claves: agenda, gestionar, tarea

ABSTRACT

The scientific and technological development achieved by humanity in the field of information technology has allowed the conditions to be created to streamline time management processes. This paper addresses the main aspects that were developed in order to make an agenda module to manage the tasks of a person. In the research carried out, a study of the Android Studio development environment, the Nginx application server, among other tools and technologies used in the process of creating the solution, is presented. On the other hand, the tests to which the elaborated solution was submitted are specified, demonstrating that it complies with the functionalities required by the client. The result of this research leaves evidence of the UCI version AUP methodology in its scenario number 4 where user stories are used to describe the functional requirements; used to fulfill the proposed general objective.

Keywords: *diary, manage, task*

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desde que el hombre comenzó a razonar se ha planteado metas a seguir para cumplir sus objetivos, por ende se ha visto en la necesidad de planificarse. El proceso de planificación es utilizado para realizar planes y proyectos de cualquier tema. Incluso en el desenlace de las grandes batallas de la historia, la victoria siempre ha estado del lado de los grandes planificadores, evidenciando la importancia de una excelente planificación para llevar a buen término cualquier tarea a realizar. La propia evolución humana y el surgimiento de formas de organización más especializadas, en las que intervienen un gran número de personas y que están presentes en todos los ámbitos de la vida como son: industrias, gobiernos, sistemas de enseñanza, sistemas de salud, entre otros; ha requerido que se realice una mejor planificación y control de las actividades que se ejecutan para lograr propósitos específicos. La evolución de las tecnologías, al mismo tiempo, ha permitido asistir el proceso de planificación mediante la creación de sistemas informáticos capaces de propagar y controlar la información relativa a la planificación de recursos y actividades.

Al administrar el tiempo se obtienen una serie de ventajas:

- Señala la necesidad de cambios futuros. Dirige la atención hacia los objetivos ayudando a determinar funcionalidades necesarias.
- Proporciona una base para el control.
- Contribuye a las actividades ordenadas. Todos los esfuerzos están apuntados hacia los resultados deseados y se logra una secuencia efectiva y sincronizada de tales esfuerzos.
- Ayuda a tener siempre presente, por parte de todos los componentes de la organización, los objetivos de esta y la adecuación de ellos al medio, cuando es necesario (Columnistas, 2019).

En la Universidad de las Ciencias Informáticas (UCI), el Centro de Tecnologías Interactivas (VERTEX) de la Facultad 4 tiene entre sus temáticas de investigación y desarrollo crear un asistente personal. En esta temática se desarrolla el proyecto del Asistente Personal Cubano con el objetivo de optimizar el tiempo de las personas. El asistente virtual actualmente no posee un agente de software que realice la planificación de tareas de forma automática por un usuario, por lo que el proceso de identificar, organizar y planificar las actividades no se cumple, además no existe un medio que permita vincular la planificación de tareas con los principales escenarios que el asistente debe de ejecutar.

Debido a esta situación problemática se plantea como **Problema de investigación:** ¿Cómo contribuir con el asistente personal cubano mediante la creación de una agenda virtual para la planificación de tareas?

Teniendo en cuenta lo anterior se define como **objeto de estudio** de la presente investigación: La planificación de tareas.

Enmarcado como **Campo de acción:** Planificación de tareas de un usuario en un asistente virtual.

Para la solución del problema planteado se propone como **Objetivo general:** Desarrollar un módulo que permita la gestión de las tareas de un usuario a través de un asistente virtual.

Objetivos específicos:

1. Conceptualizar los asistentes personales virtuales y agendas virtuales.
2. Identificar y caracterizar las soluciones informáticas
3. Identificar, caracterizar y seleccionar metodología, herramientas y gestores de base de datos para el desarrollo de software.
4. Levantar los requisitos funcionales y no funcionales de la solución informática.
5. Modelar el patrón arquitectónico a utilizar.
6. Realizar las historias de usuario para cada requisito funcional.
7. Identificar los patrones de diseño a utilizar.
8. Realizar pruebas unitarias y pruebas de aceptación a la herramienta informática.

Los métodos de investigación utilizados se exponen a continuación:

Métodos Empíricos: Modelo de investigación que pretende obtener conocimiento a partir de la observación de la realidad. Por ende, está basado en la experiencia. En este modelo la observación de la realidad es el punto de partida para formular hipótesis, las cuales deben ser sometidas a prueba mediante la experimentación (Andrés Vicente, 2021).

- **Observación:** Mediante este método se estudiará lo relacionado al objeto de estudio de la investigación, las acciones, causas y consecuencias logrando conocer la esencia del problema planteado, analizando desde varios puntos de vista la propuesta de solución y otras soluciones existentes.
- **Análisis de documentos:** Se pone de manifiesto al revisar la documentación sobre los asistentes personales y las agendas virtuales.

Métodos Teóricos: Permiten revelar las relaciones esenciales del objeto de investigación no observables directamente, cumpliendo así una función gnoseológica importante al posibilitar la interpretación conceptual de los datos empíricos encontrados, la construcción y desarrollo de teorías, creando las condiciones (Del Sol Fabregat et al., 2017).

- **Analítico-Sintético:** Este método permitirá realizar el estudio teórico a través del análisis de documentos, libros, artículos y otras fuentes bibliográficas de diferentes autores.

- **Histórico-Lógico:** Mediante este método se analizará el desarrollo de los asistentes personales con módulo de agenda y sus elementos más importantes; con el propósito de crear la base teórica de la investigación.

El presente documento se encuentra estructurado en 3 capítulos:

Capítulo1: Se describe el estudio del estado del arte de la investigación, centrándose en los conceptos y características fundamentales de los asistentes personales y las agendas virtuales. Además, se describen la metodología y herramientas utilizadas para implementar la solución.

Capítulo 2: Diseño de la solución propuesta al problema de investigación. En este capítulo se realiza el levantamiento de requisitos funcionales y no funcionales. Se diseña las historias de usuario para cada requisito funcional, así como el modelo conceptual y diagrama de despliegue.

Capítulo 3: Validación de la solución propuesta. Se establece el modelo de implementación mediante los estándares de codificación. Se le aplican pruebas a la aplicación para verificar que sea correcto su funcionamiento.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

El presente capítulo está dirigido a abordar los elementos teóricos que sustentan el objetivo de la investigación. Se hace énfasis en los conceptos y elementos importantes relacionados con el tema. Se realiza un estudio de las herramientas existentes para generar una agenda; así como los sistemas homólogos de los mismos. También se presentan las diferentes tecnologías y lenguajes, además de la metodología que se emplea en el proyecto para dar apoyo al cumplimiento de los objetivos y tareas propuestas y de esta forma dar solución al problema planteado.

1.1 Asistente Virtual

Los asistentes virtuales han ganado popularidad en los últimos tiempos gracias a su capacidad para facilitar algunas de las tareas que se realizan habitualmente con los dispositivos móviles. Así, y con sólo pedirlo de viva voz, ayudan a buscar datos en Internet o a hacer consultas relacionadas con la actualidad o el tiempo. Estos, al estar dotados de Inteligencia Artificial, se puede interactuar con ellos empleando un lenguaje natural, mientras que van aprendiendo según sean utilizados; además tienen múltiples beneficios y aplicaciones educativas. Para una mejor comprensión del término Asistente Virtual se brindan los siguientes conceptos:

- Un asistente virtual consiste en un programa cuya interfaz de uso es el reconocimiento y procesamiento de la voz del usuario, de forma que el asistente “comprende” lo que el usuario le está solicitando y ejecuta la acción, siempre y cuando esta se encuentre dentro de sus posibilidades (González, 2016).
- Los asistentes virtuales son robots que responden a comandos de voz; cada vez más sofisticados e inteligentes. Estos sistemas aprenden a través de interacciones con el consumidor, y mejoran continuamente su repertorio; entre más interactúan con las personas más aprenden (Valois, 2019).
- Un asistente personal virtual o Virtual Personal Assistant (VPA) es un agente de software que ayuda a usuarios en la automatización y realización de tareas con una mínima interacción hombre-máquina. La interacción se realiza usando texto o voz y el asistente virtual procesa, interpreta y responde de la misma manera (Big Data Marketer, 2018).

Después del análisis de los conceptos planteados se pudo entender que un **Asistente Virtual** es un software cuya interfaz permite el procesamiento y reconocimiento de la información a partir de la voz

del usuario. Permitiendo así, desde lo solicitado, la ejecución de la acción; teniendo en cuenta la cantidad de información almacenada en su base de datos. Los mismos se desarrollan a partir de la interacción con el consumidor, mientras más intercambio, se evidencia un mayor aprendizaje.

Los asistentes virtuales son entidades capaces de percibir su entorno, los cuales pueden procesar lo que perciben y tener una reacción, es decir, actuar de manera racional. Pueden servir de guías y ayuda en la búsqueda de información de un tema o en la organización de tareas; para ello tienen integrado varios módulos, entre ellos el de la agenda.

1.1.1 Asistente personal cubano

En el Centro de Tecnologías Interactivas (VERTEX), tiene dentro de sus líneas de desarrollo la Automática Aplicada. En esta línea de investigación se desarrolla un prototipo de Asistente Personal Virtual, que se caracteriza como un agente tipo software que puede realizar tareas y ofrecer servicios a un individuo. Uno de los aspectos clave del Asistente Personal Virtual, es la habilidad para conectarse con dispositivos DAQ (tarjetas de adquisición de datos) mediante un mecanismo de comunicación basada en protocolos de Internet de las Cosas, este realiza tareas que pueden ser ejecutadas por dispositivos automatizados relacionado con la domótica, así como ejecutar ordenes en dispositivos de control para el encendido o apagado de luces y electrodomésticos, como son: televisores, aires acondicionados, entre otros, permitiendo un determinado confort en una habitación cerrada. Otra de las funcionalidades es la asistencia personal relacionado con indicadores de salud, por ejemplo: la cantidad diaria de calorías, el ritmo cardiaco, presión arterial, nivel de azúcar, el volumen de ejercicio y las recomendaciones para mejorar la calidad de vida. Además, ejecuta servicios basados en datos de entrada de usuario, reconocimiento de ubicaciones y la habilidad de acceder a información de una variedad de recursos en línea (como el clima, noticias nacionales e internacionales, entre otros).

1.2 Agenda Virtual

La agenda es un objeto donde se escriben los datos del usuario y las citas próximas. Tiene la función de planificar, organizar y documentar reuniones entre otros acontecimientos personales. El hecho de llevar una agenda personal ayuda a la persona a cumplir con sus metas y hacer más productivo en beneficio de él y de todas las personas que le rodean. A continuación, se define mejor el termino de agenda virtual:

- Es aquella donde se asientan, disponen y programan, de manera ordenada, una serie de tareas o actividades relacionadas con el desempeño laboral. Como tal, permite al usuario organizar su tiempo de acuerdo con los objetivos de su gestión para maximizar su rendimiento, eficacia y productividad. En este sentido, es una herramienta muy útil a la hora de organizar tareas y

eventos. Posee la ventaja de encontrar un plan de actividades actualizado y preciso de forma sencilla en todo momento (Andrés Vicente, 2021).

- Como agenda electrónica o digital se denomina al dispositivo electrónico de bolsillo que funciona como una agenda personal. Como tal, posee múltiples funciones orientadas a la gestión del tiempo, y tiene la capacidad de almacenar todo tipo de datos, así como de organizar tareas y actividades (Becas Santander, 2022).

A partir del análisis de los conceptos planteados anteriormente se puede decir que una **Agenda Virtual** es la que permite la programación de una serie de tareas y actividades que mejoran el desarrollo del desempeño laboral, ya que se dispone de manera organizada y permite al usuario el cumplimiento de sus objetivos con una mayor productividad y rendimiento; brindándole al mismo un recordatorio de la hora y fecha de eventos fundamentales de su agenda diaria.

1.3 Análisis de Sistemas Homólogos

Para un desarrollo eficiente del módulo es necesario conocer cómo se gestiona la información en las agendas de los asistentes personales. Con el objetivo de lograr un mayor entendimiento, se investigaron los siguientes VPA:

Alexa

Es un asistente de planificación inteligente que permite administrar los calendarios de sus usuarios. Los usuarios pueden programar reuniones en sus calendarios con el uso de la voz. El asistente de planificación inteligente de Alexa busca automáticamente el tiempo libre en el calendario del usuario para la reunión, lo que les ahorra a los usuarios el tiempo y el esfuerzo que requeriría hacer esto de forma manual.

Las características del asistente de planificación inteligente de Alexa se habilitan de forma automática. Para usar estas características, los usuarios deben vincular sus calendarios corporativos y, de modo opcional, su calendario personal a la cuenta de Alexa. Una vez que han vinculado sus calendarios, pueden decir “Alexa, programa una reunión con John”, y el asistente de planificación inteligente de Alexa ofrecerá sugerencias de horarios en los que ambos asistentes aparezcan como disponibles. También revisará los calendarios personales para asegurarse de que la reunión no se programe en un horario que se muestre como ocupado. Para reprogramar una reunión, los usuarios pueden decir “Alexa, mueve mi reunión de análisis financiero para mañana a las 4 p. m.” y Alexa cambiará la reunión de lugar y enviará una invitación actualizada a todos los participantes. Si existe otro evento programado para el horario propuesto, Alexa le advertirá sobre esto y sugerirá una alternativa (Amazon Web Services, 2018).

La única forma de acceder a la lista de tareas pendientes es dentro de la aplicación Amazon Alexa en Android o iOS o yendo a alexa.amazon.com en el navegador web. Desde allí, se puede agregar un elemento manualmente, marcar, editar y eliminar elementos. Además, se puede ver las tareas completadas (Martin, 2016).

Siri

Al igual que cualquier otro asistente virtual, el método de uso es parecido. El usuario puede pedirle que realice una tarea únicamente diciendo en voz alta “Hola, Siri” a su dispositivo.

Siri tiene acceso total a el calendario del usuario y a todos los eventos, así que si el usuario desea saber que eventos tiene en su calendario solo debe preguntarle a Siri, por ejemplo: “Siri ¿Qué hay en mi calendario hoy?”. Siri escanea sus calendarios, y reporta cualquier hallazgo, si el usuario usa preguntas más amplias que tienen un mayor lapso de tiempo, Siri incluso le proporcionará un pequeño calendario incrustado en la respuesta. Si no se encuentra nada que tenga que ver con la pregunta, entonces llegará una respuesta como “no tienes citas ese día” o “no he encontrado ninguna cita para hoy”. Por supuesto, también puede añadir y modificar eventos de calendario, reuniones y fechas (Tecnologismo, 2022).

Google Assistant

Google Assistant es un asistente virtual desarrollado por Google que está disponible en dispositivos móviles y terminales domóticos (Frutos, 2018).

El usuario puede agregar eventos y solicitar información de eventos por voz. También puede añadir elementos a su calendario de Google desde otros dispositivos habilitados y acceder a ellos con Google Assistant. Al crear un evento, en el apartado interno de Notificación el usuario puede añadir un aviso mediante el envío de un correo, o un aviso unos minutos antes, unas horas, unos días o incluso semanas antes (Cernadas, 2019).

El Asistente de Google permite llevar a cabo desde las tareas más fáciles a las más cotidianas. El ejemplo perfecto de ello es la posibilidad de establecer alarmas y recordatorios simplemente pidiéndoselo con el comando de voz oportuno (M. Martínez, 2022).

Luego del análisis de estos sistemas similares se puede decir que estos sistemas tienen varias funcionalidades en común como, por ejemplo: crear, modificar, eliminar y visualizar tareas; además de estas características se tendrán en cuenta para el desarrollo del módulo de agenda para el asistente personal cubano las funcionalidades de establecer recordatorios y alarmas vinculados a cada tarea.

1.4 Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático (Becas Santander, 2022).

El uso de una metodología específica está asociado a características propias del proceso de desarrollo de software. Cada metodología tiene sus diferencias evidenciadas en la manera en que gestionan con precisión los artefactos, roles y actividades del proyecto (Letelier & Penadés, 2019).

Debido que VERTEX en el desarrollo del proyecto del Asistente Personal Cubano trabaja con la metodología AUP-UCI, se decidió escoger la misma, que cuenta con tres fases (Inicio, Ejecución y Cierre).

Para el Modelado del dominio se plantean tres formas de encapsular los requisitos: Casos de uso del sistema (CUS), Historias de usuario (HU) y por último la Descripción de requisitos de procesos (DRP), de los cuales surgen cuatro escenarios para modelar el sistema en los proyectos quedando de la siguiente forma:

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

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

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

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

Debido que en la investigación no se modela el negocio, solo se puede modelar el sistema, se escoge el escenario No.4 para realizar la descripción de los requisitos (Rodríguez Sánchez, 2015).

1.5 Herramientas de desarrollo

Visual Paradigm (versión 15.1)

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software (Hernández Pérez, 2017).

Visual Studio Code (versión 1.73.1)

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación (Flores, 2022).

Principales características:

- **Multiplataforma:** Está disponible para Windows, GNU/Linux y macOS.
- **IntelliSense:** Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código. Como su nombre lo indica, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etc. Con la ayuda de extensiones se puede personalizar y conseguir un IntelliSense más completo para cualquier lenguaje.
- **Depuración:** VS Code incluye la función de depuración que ayuda a detectar errores en el código. De esta manera, se evita tener que revisar línea por línea a puro ojo humano para encontrar errores. VS Code también es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.
- **Uso del control de versiones:** VS Code tiene compatibilidad con Git, por lo que se puede revisar diferencias, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC). Los demás SMC están disponible por medio de extensiones.
- **Extensiones:** Las extensiones permiten personalizar y agregar funcionalidad adicional de forma modular y aislada, y lo más importante, no afectan en el rendimiento del editor, ya que se ejecutan en procesos independientes.

SQLite (versión 3.40)

Es un motor de base de datos SQL transaccional de código abierto, ligero, autónomo, de configuración simple y sin servidor, que se caracteriza por almacenar información persistente de forma sencilla, SQLite gracias a sus características se diferencia de otros gestores de bases de datos, proporcionando grandes ventajas sobre ellos.

Así mismo, por ser de dominio público es gratuito tanto para fines privados como para comerciales, se puede descargar de forma libre desde su sitio oficial. Además, cuenta con varios enlaces a lenguajes de programación entre los que se destacan: Java, C, C ++, JavaScript, C #, Python, VB Script, entre otros. Las bases de datos generadas con SQLite son ligeras, esto con la finalidad de que la aplicación desarrollada pueda interactuar con los datos desde dispositivos con menores prestaciones. (Muradas, 2018).

Framework o marco de trabajo

Un *Framework* es una estructura previa que se puede aprovechar para desarrollar un proyecto (Muente, 2020). Flutter y React Native son los dos *frameworks* más populares del mercado. Son bien conocidos por ser fáciles de usar. Son únicos porque facilitan el desarrollo de aplicaciones y la codificación. Ambos

ofrecen las mismas características y beneficios, pero son muy diferentes en muchos aspectos. React Native tiene una comunidad más grande y proporciona una plataforma para que los desarrolladores creen aplicaciones. Por otro lado, Flutter se enfoca más en herramientas integradas y menos dependencia del software de terceros (Ruchir, 2021). Debido a esto se decide escoger Flutter.

Flutter (versión 3.3.3)

Flutter es un Software Development Kit (SDK) desarrollado por Google para crear aplicaciones móviles tanto para *Android* como para iOS (Apple) (TI Consultors S.L, 2019).

Estas son las tres principales ventajas que ofrece Flutter respecto a otras frameworks de desarrollo de aplicaciones multiplataforma:

1. Compila el código en el lenguaje nativo de las plataformas, tanto en Android como en iOS.
2. La creación interfaces gráficas es muy flexible, puedes combinar diferentes Widgets (elementos gráficos) para crear las vistas.
3. El desarrollo es muy rápido, permite ver el resultado de forma instantánea mientras se escribe el código.

FastAPI (versión 0.8.1)

FastAPI es un framework para construir APIs de forma sencilla y rápida con Python. Actualmente se considera como uno de los frameworks basados en Python más rápidos y, además, proporciona también una gran velocidad a la hora de abordar el desarrollo de una API al incorporar, entre otras cosas, validación de datos y de documentación de forma automática, lo cual lo convierte en un candidato ideal para realizar el backend de cualquier aplicación (Mesa, 2022).

Se considera pertinente utilizar dicho framework debido a que tienen una curva de aprendizaje muy poco empinada y permite en un plazo razonable de tiempo obtener los conocimientos necesarios para el desarrollo de dicha aplicación (TI Consultors S.L, 2019).

Servidor Web

Un servidor web es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación del lado del cliente. Mientras que comúnmente se utiliza la palabra servidor para hacer referencia a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones antes descritas. Una de las principales razones para utilizar este servidor web es que se trata de un software que es asíncrono, a diferencia de Apache que está basada en procesos. La ventaja principal de ser asíncrono, es su escalabilidad. En un sistema basado en procesos, cada conexión simultánea requiere de un hilo, lo que puede llevar a sobrecargar el servidor, mientras que en un servidor asíncrono se gestionan las

peticiones en muy pocos hilos, reduciendo las posibilidades de sobrecarga en el servidor (Betania, 2022). Para que la API soporte un mayor número de conexiones se decide usar Nginx.

Nginx (versión 1.22.1)

Nginx es un servidor web que también puede ser usado como proxy inverso, balanceador de carga y proxy para protocolos de correo.

Además de otras tareas, los servidores web son los encargados de la entrega de aplicaciones web, respondiendo a peticiones HTTPS realizadas por usuarios, normalmente desde un navegador web (WEBEMPRESA EUROPA S.L, 2022).

1.6 Lenguajes de programación

Un lenguaje de programación es una herramienta que permite desarrollar software o programas para computadora. Los lenguajes de programación son empleados para diseñar e implementar programas encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora. Lo anterior se logra mediante la creación e implementación de algoritmos de precisión que se utilizan como una forma de comunicación humana con la computadora. A grandes rasgos, un lenguaje de programación se conforma de una serie de símbolos y reglas de sintaxis y semántica que definen la estructura principal del lenguaje y le dan un significado a sus elementos y expresiones (Ceballos, 2021).

Existen varios lenguajes de programación, entre los que se encuentran Dart y Paython; los utilizados en el desarrollo del módulo de agenda virtual para un asistente personal.

Dart (versión 2.18.2)

Dart es un lenguaje de programación orientado a objetos, de propósito general y de código abierto con sintaxis de estilo C desarrollado por Google en 2011. El propósito de la programación de Dart es crear una interfaz de usuario para la web y las aplicaciones móviles. Está inspirado en otros lenguajes de programación como Java, JavaScript, C# y está fuertemente tipado (Jaiswal, 2021).

Principales características de Dart (Cordón, 2021):

- **Programación estructurada y flexible**: Google diseñó Dart para poder ser utilizado en proyectos de una sola persona hasta proyectos más desarrollados o complejos.
- **Lenguaje familiar y fácil de aprender:** Es un lenguaje realmente sencillo y fácil de aprender. En su sitio web se pueden encontrar varios tutoriales, y también permite colaboraciones de otros desarrolladores.
- **Permite la adaptación de nueva herramienta a cualquier navegador web:** El lenguaje de programación Dart se puede ejecutar de dos maneras; en una máquina virtual (MV), o en un

motor de JavaScript utilizando un compilador para traducir el código. Esto le permite adaptarse a cualquier navegador.

- **Lenguaje basado en clases e interfaces:** Gracias a sus basamentos en clases o en la programación orientada a objetos, se facilita la encapsulación y la reutilización del código.

Principales ventajas de Dart:

- Es de acceso gratuito para cualquier persona.
- Detrás de su programación se encuentra Google, lo que ofrece perspectivas a largo plazo para el desarrollo del lenguaje.
- Dart es fácil de aprender debido a que los desarrolladores han simplificado características complicadas de otros lenguajes.
- Funciona en todos los navegadores móviles y de escritorio actuales**.**

Python (versión 3.10.0)

Python es un lenguaje de programación de computadoras que a menudo se usa para crear sitios web y software, automatizar tareas y realizar análisis de datos. Es un lenguaje de propósito general, lo que significa que puede usarse para crear una variedad de programas diferentes y no está especializado para ningún problema específico. Esta versatilidad, junto con su facilidad de uso para principiantes, lo ha convertido en uno de los lenguajes de programación más utilizados en la actualidad (Coursera, 2022).

Principales características de Python (Amazon Web Services, Inc, 2022):

- **Un lenguaje fácil de utilizar:** utiliza palabras similares a las del inglés. A diferencia de otros lenguajes de programación, Python no utiliza llaves. En su lugar, utiliza sangría.
- **Un lenguaje interpretado:** ejecuta directamente el código, línea por línea. Si existen errores en el código del programa, su ejecución se detiene. Así, los programadores pueden encontrar errores en el código con rapidez.
- **Un lenguaje tipeado dinámicamente:** los programadores no tienen que anunciar tipos de variables cuando escriben código porque Python los determina en el tiempo de ejecución. Debido a esto, es posible escribir programas de Python con mayor rapidez.
- **Un lenguaje de alto nivel:** es más cercano a los idiomas humanos que otros lenguajes de programación. Por lo tanto, los programadores no deben preocuparse sobre sus funcionalidades subyacentes, como la arquitectura y la administración de la memoria.
- **Un lenguaje orientado a los objetos:** considera todo como un objeto, pero también admite otros tipos de programación, como la programación estructurada y la funcional.

Principales ventajas de Python:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

Conclusiones del capítulo

En este capítulo se profundizaron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- El estudio del estado del arte realizado, permitió llegar a un mayor entendimiento de los términos y funcionalidades comunes de los sistemas similares necesarios para el desarrollo de la aplicación móvil.
- Se seleccionó como metodología de desarrollo de software la AUP-UCI debido a que es la metodología adoptada por el proyecto del Asistente Personal Cubano del centro VERTEX.
- El estudio de las herramientas y tecnologías a utilizar permitió una mayor comprensión de las mismas para su correcto uso.

CAPÍTULO II: PLANIFICACIÓN Y DISEÑO DEL MÓDULO DE AGENDA VIRTUAL PARA UN ASISTENTE PERSONAL

En el presente capítulo se describen las principales características del sistema propuesto, donde se ejecutan las fases de inicio y ejecución de la metodología de desarrollo de software AUP en su versión UCI en concordancia con el escenario escogido, así como los diferentes artefactos generados en cada una de ellas. Se definen los requisitos funcionales y no funcionales, se presenta la propuesta de solución del sistema.

2.1 Propuesta solución

Se propone como posible solución el desarrollo de una aplicación móvil la cual visualizará un calendario y representado en él los días en los que hay tareas, las cuales se podrán añadir, editar y eliminar. Este itinerario de tarea se guardará en la base de datos alojada en una Interfaz de Programación de Aplicaciones (API), por lo que la aplicación solo funcionará online.

Ilustración 1 Propuesta de solución

2.2 Modelo conceptual

El modelo conceptual de un sistema de software constituye una abstracción externa que describe mediante diagramas y notaciones con distinto grado de formalidad el conocimiento que debe poseer una persona acerca de un sistema, conocimiento que se encuentra almacenado en la Memoria a Largo Plazo (Granollers, 2021). Es la representación de uso de un sistema a través de una interfaz. El diseño

de un modelo conceptual es la herramienta utilizada para comunicar la intención del diseño. Cuanto mejor se relacione el modelo conceptual con los modelos mentales (representación que una persona tiene en mente acerca del objeto con el cual está interactuando) existentes de los usuarios, más fácil será explicar lo que se pretende hacer con la aplicación (Hernández, 2018).

A continuación, se describen los conceptos que conforman este modelo:

Tareas: Información principal a gestionar por el usuario.

Usuario: Encargado de gestionar las tareas.

Alarmas: Notifica al usuario del evento.

Recordatorios: Notifica al usuario de la proximidad de un evento.

Fecha y Hora: Almacena la hora y fecha del evento.

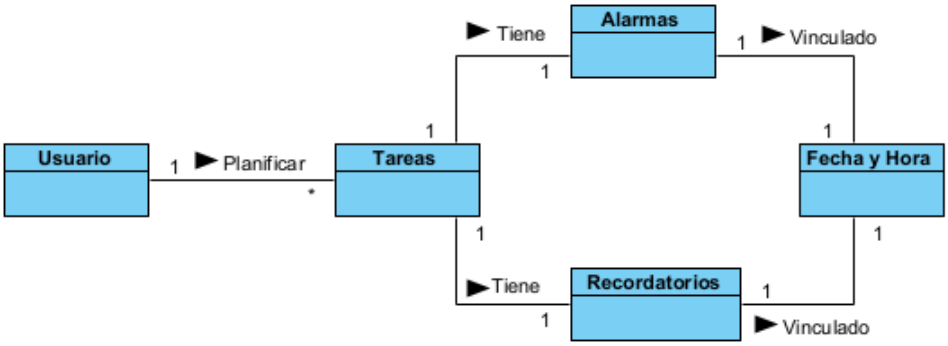


Ilustración 2 Modelo Conceptual

2.3 Especificación de requisitos

Requisitos funcionales

Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requisitos funcionales también explican lo que no debe hacer el sistema (Sommerville, 2011).

Tabla 1 Requisitos Funcionales

No.	Nombre	Descripción
RF 1	Insertar Tarea	La aplicación debe mostrar al usuario una interfaz con los campos a llenar: Título, Nota, Fecha,

		Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota
RF 2	Modificar Tarea	La aplicación debe mostrar al usuario una interfaz con los campos a modificar: Título, Nota, Fecha, Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota
RF 3	Eliminar Tarea	La aplicación debe permitir al usuario eliminar la tarea y a su vez el sistema la elimina de la base de datos de la API.
RF 4	Visualizar Tareas	La aplicación muestra un listado con las tareas registradas para el día seleccionado
RF 5	Configurar Alarma	El sistema se comunica con la aplicación de reloj del teléfono y configura la alarma con la hora correspondiente.
RF 6	Establecer Recordatorio	El sistema añade a la cola los recordatorios creados en el menú de creación-modificación de tareas.
RF 7	Actualizar listado de tareas	El sistema accede a la API que gestiona las tareas y recupera el listado de tareas para ser visualizado
RF 8	Conectar la aplicación con una API	El sistema accede a la API para actualizar las tareas.
RF 9	Autenticar usuario	El sistema permite acreditar la identidad de un usuario mediante las credenciales ``Nombre´´ y ``Contraseña´´
RF 10	Registrar usuario	La aplicación muestra una interfaz con los campos a llenar: Nombre, Correo y Contraseña y el botón de ``Registrar´´
RF 11	Editar usuario	La aplicación muestra un cuestionario con los datos del usuario a modificar

RF 12	Cerrar sesión	El sistema debe permitir al usuario cerrar la sesión
RF 13	Verificar registro	El sistema enviará a través de correo un enlace de verificación
RF 14	Recuperar contraseña	El sistema debe permitir recuperar la contraseña enviando un código de recuperación al correo

Requisitos no funcionales

Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requisitos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema (Sommerville, 2011).

Tras el análisis realizado se definieron los siguientes requisitos:

Requerimientos de software

- Teléfonos inteligentes con Android 8.0 o superior.

Requerimientos de hardware

- El sistema donde se instalará la aplicación requiere 256 MB de RAM como mínimo
- El sistema requiere un procesador de 1 GHz de velocidad de procesamiento o superior.

Interfaz

- La aplicación tendrá los colores blanco y azul; con las etiquetas rojas, verdes y azul.

2.4 Historias de Usuario

Las historias de usuario son descripciones muy cortas y esquemáticas, que resumen la necesidad concreta de un usuario al utilizar un producto o servicio, así como la solución que la satisface. Como muchas otras herramientas ágiles, las historias de usuario surgieron como una respuesta orientada al sector de desarrollo de software. Su función principal es identificar problemas percibidos, proponer soluciones y estimar el esfuerzo que requieren implementar las ideas propuestas (Solving Ad Hoc, 2017).

Tabla 2 HU Insertar Tarea

Historia de usuario	
Número: 1	Usuario: Programador
Nombre: Insertar Tarea	
Prioridad: Alta	Riesgo en desarrollo: Medio

Tiempo estimado: 8 días	

Descripción:

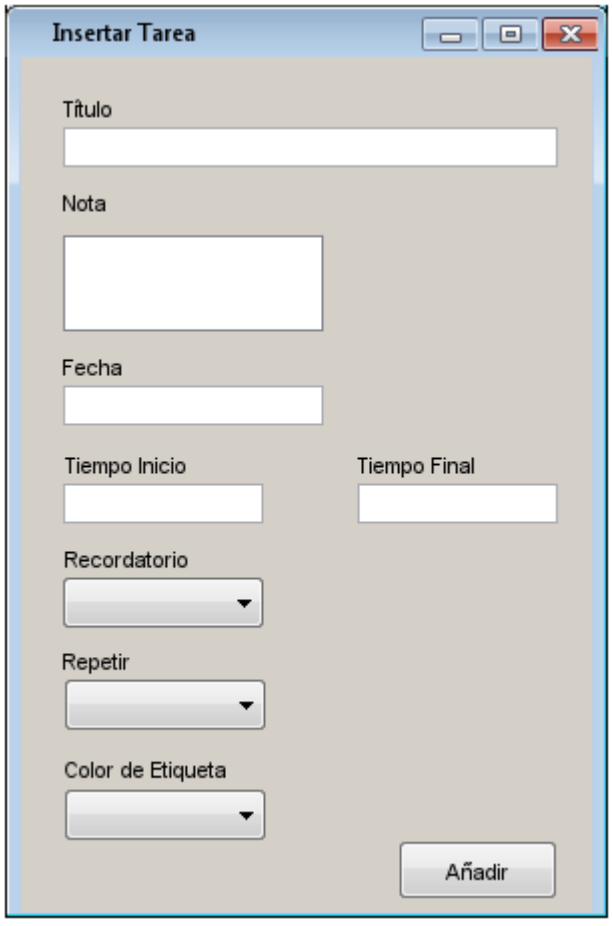
1-Objetivo: Permitir crear una tarea

2-Condiciones para lograr el objetivo:

-Seleccionar previamente la fecha de la tarea

3-Flujo de la acción a realizar: Cuando el usuario selecciona el botón para crear una tarea, el sistema muestra una nueva ventana con los campos a llenar (Título, Nota, Fecha, Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota) y los botones de guardar y cancelar. El usuario llena los campos y presiona guardar, el sistema guarda de manera consistente la tarea nueva y envía la solicitud de actualizar las tareas en la API.

Observaciones:



2.5 Diseño arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2005).

2.5.1 Diseño arquitectónico para la aplicación móvil

Para el desarrollo de la aplicación móvil se escoge como base el patrón Modelo-Vista-Controlador (MVC). Este patrón surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo.

- **Modelo**: El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones.
- **Vista**: La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario.
- **Controlador**: El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo.

En Android el patrón arquitectónico MVC posee una particularidad en cuanto a la vista y el controlador, definiéndose estos dos como un par Controlador-Vista. En este caso particular el controlador notifica al modelo y el modelo notifica a todos los dependientes (pares Controlador-Vista) asociados a él. Para este par la vista contiene una instancia de su controladora y la controladora además posee una instancia de la vista (Álvarez, 2020).

El diagrama describe la aplicación de la siguiente manera:

Vista: son las actividades donde se muestra la información de las tareas y la gestión del usuario.

Controlador: es la clase apihelper que es la encargada de controlar el flujo entre el modelo y la vista.

Modelo: se centra en la base de datos.

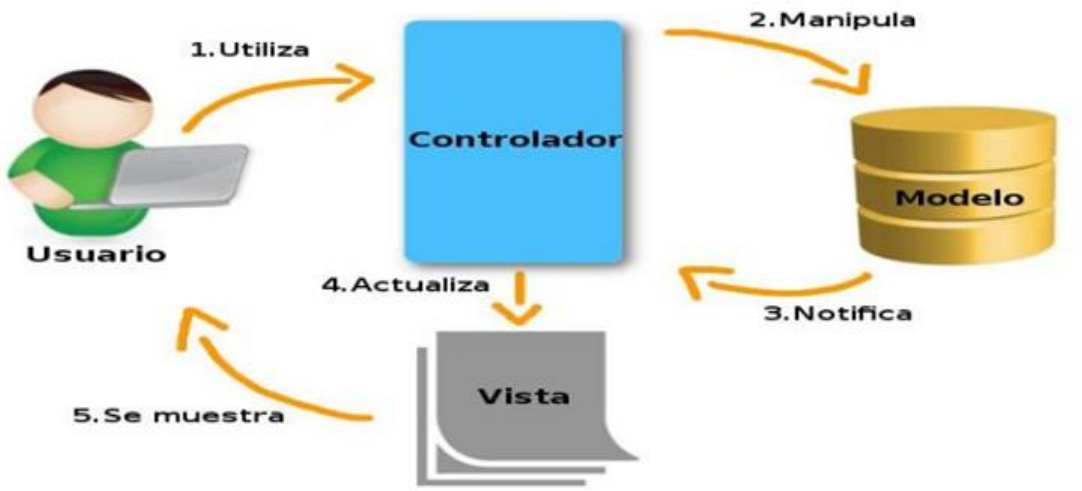


Ilustración 3 Patrón arquitectónico MVC

2.5.2 Diseño arquitectónico para la API

Para el desarrollo de la API se escoge como base la arquitectura N-Capas. Las capas son una forma de separar responsabilidades y administrar dependencias. Cada capa tiene una responsabilidad específica. Una capa superior puede utilizar los servicios de una capa inferior, pero no al revés (EdPrice-MSFT, 2021).

El diagrama describe la API de la siguiente manera:

- **Capa de la lógica del negocio:** Componentes que modelan la lógica de negocio. Además, utilizan la capa de datos para manipular información. Integración sencilla y eficaz con sistemas externos.
- **Capa de acceso a datos:** Almacenamiento, actualización y consulta de los datos del sistema. Puede estar en el mismo servidor de la lógica de negocio o distribuida.

Ilustración 4 Arquitectura N-Capas

2.5.3 Diseño arquitectónico del sistema

Para el desarrollo del sistema (unión de la aplicación móvil con la API) se basa en la arquitectura Cliente-Servidor. Está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente. En dicha arquitectura existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado. En esta arquitectura, el Cliente y el Servidor son desarrollados como dos aplicaciones diferentes, de tal forma que cada una puede ser desarrollada de forma independiente, dando como resultado dos aplicaciones separadas, las cuales pueden ser construidas en tecnologías diferentes, pero siempre respetando el mismo protocolo de comunicación para establecer comunicación (Blancarte Iturralde, 2020).

Ilustración 5 Arquitectura Cliente-Servidor

2.6 Desarrollo de la solución propuesta

Para el desarrollo de la aplicación móvil se hace uso del *framework* Flutter con el fin de obtener una aplicación Android la cual consuma los servicios creados en la API y se encargue de manejar la interacción del usuario con la API a través de una interfaz amigable. La misma al insertar una nueva tarea creará una alarma de reloj en el dispositivo con los datos asociados a la tarea creada. Además, le da la capacidad al usuario de administrar su foto de perfil; así como, la posibilidad de tener un resumen de sus tareas creadas categorizándolas en completas, incompletas y total de tareas.

Para el desarrollo de la API en la capa de la lógica del negocio se centra la gestión de las tareas asociadas a los usuarios; dando la capacidad al sistema de adicionar (genera una notificación en el móvil y en el correo del usuario), modificar, eliminar y listar dichas tareas. La autenticación del usuario se gestiona a través de JSon Web Token (JWT) (Heffelfinger, 2022) y Oauth2 (Siriwardena, 2020) con el fin de desarrollar un sistema estable y seguro dadas las características de dichas tecnologías.

En esencia al registrar un usuario, este introduce los campos de nombre del usuario, contraseña y correo; una vez enviada la información al sistema dicho usuario verifica su registro a través de un correo que es enviado automáticamente, y una vez confirmado el registro es que puede hacer uso del sistema. Cuando el usuario se autentica se genera un token con la información del usuario para así poder acceder a las funcionalidades de gestionar sus tareas, las cuales se encuentran protegidas debido a la autenticación del usuario y la verificación de su registro.

En caso que el usuario haya olvidado su contraseña, el sistema cuenta con un mecanismo para dar la oportunidad al usuario de actualizar su contraseña, solicitando su nombre, y enviando un código único a su correo para luego actualizar su contraseña usando el código enviado.

En la capa del modelo haciendo uso del tortoise (Khaild, 2022) se crean los modelos, definiendo la estructura para los usuarios y las tareas asociadas para los mismos.

2.7 Patrones de diseño

Los patrones de diseño son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error (Martínez Canelo, 2020).

De los patrones de diseño estudiados, se seleccionaron los que se considerarán necesarios en el desarrollo de la aplicación. A continuación, se muestra dicha selección en conjunto con una breve descripción del patrón.

Patrones GRASP

Son guías o principios que sirven para asignar responsabilidades a las clases (Kord, 2017).

- **Controlador:** Proporciona guías acerca de las opciones generalmente aceptadas y adecuadas para manejar eventos. Es conveniente utilizar la misma clase controlador para todos los eventos del sistema de un requisito, de manera que es posible manejar la información acerca del estado del caso de uso en el controlador. Se utilizó en la clase TaskController
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Se utilizó en la clase api_helper que es la encargada de crear las de las tareas en la API.
- **Experto:** Propone que las responsabilidades se deben ser asignadas a los objetos que poseen la información para poder ejecutarlas (Vallejo Guerra, 2022). Se utilizó en la clase Task que es la encargada de la estructura de las tareas.
- **Bajo acoplamiento:** Soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio (Cárdenas Campaña, 2022).
- **Alta cohesión:** Tiene un número relativamente pequeño de métodos, con funcionalidades altamente relacionadas, y no realiza mucho trabajo, colaborando con otros objetos para compartir el esfuerzo si la tarea es extensa (Gutiérrez Cabrera, 2018).

Patrones GOF

Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla (Pérez, 2018).

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

2.8 Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama UML que muestra la arquitectura de ejecución de un sistema, incluyendo nodos como entornos de ejecución de hardware o software, y el middleware que los conecta. Se utilizan normalmente para visualizar el hardware y el software físico de un sistema. Usándolo puedes entender cómo el sistema se desplegará físicamente en el hardware (Cinergix Pty. Ltd, 2022).

Para desplegar el sistema se requiere de un dispositivo móvil con la aplicación instalada y de una computadora o servidor con una API que brinde el servicio de almacenamiento y recuperación de los datos necesarios para su correcto funcionamiento.

Dispositivo móvil: dispositivo donde se encuentra instalada la aplicación de la agenda.

Servicio: servidor que contiene la API con la base de datos.

Ilustración 6 Diagrama de Despliegue

Conclusiones del capítulo

En este capítulo se realizó el análisis y diseño de la aplicación, arribando a las siguientes conclusiones:

- El levantamiento de requisitos permitió determinar las funcionalidades que debe cumplir el sistema.
- La realización del modelo conceptual permitió una mejor comprensión de los conceptos asociados al negocio.
- Los artefactos generados durante el análisis y diseño de la solución contribuyeron a un mejor entendimiento del sistema para dar paso a la implementación de la solución propuesta.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DEL MÓDULO DE AGENDA VIRTUAL PARA UN ASISTENTE PERSONAL

En el siguiente capítulo se desarrollará la implementación de la propuesta de solución, teniéndose como precedente, los artefactos generados y la descripción de cada uno de los requisitos. Además, se llevará a cabo el proceso de prueba a la solución desarrollada con el objetivo de alcanzar la calidad de la misma.

3.1 Modelo de implementación

El modelo de implementación garantiza la implementación de la propuesta de solución, basándose en su arquitectura, sus componentes, de acuerdo a su mecanismo de estructuración, y los lenguajes de programación viables.

3.1.1Estándares de codificación

Los estándares de codificación incorporan principios de ingeniería sólidos para la programación en sus respectivos lenguajes y forman la base de cualquier enfoque preventivo. El costo de un buen software es menor que el costo de un software malo (Hamilton, 2020). A continuación, se especifican los estándares de codificación utilizados en la construcción de la solución.

- El tamaño máximo de las líneas de código debe ser de cien a ciento veinte caracteres aproximadamente, de manera tal que se garantice la completa visibilidad de las líneas de código sin necesidad de realizar desplazamiento horizontal.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Get.isDarkMode?Colors.grey[600]:Colors.white,
      leading: IconButton(
        onPressed: (() => Get.back()),
        icon: Icon(Icons.arrow_back_ios),
        color: Colors.black,


        ),
      title: Text(this.Label.toString().split("|")[0]),
    ),

  );
}
```

Ilustración 7 Estándar de codificación límite de caracteres

- Los nombres de las clases y las funciones adoptarán la notación CamelCase y no se utilizará el guion bajo como delimitador entre palabras.

```
class NotifyPage extends StatelessWidget {
```

Ilustración 8 Estándar de Codificación CamelCase

- Las sentencias pertenecientes a un bloque de código estarán ubicadas a un nivel más a la derecha con respecto a la sentencia que las contiene. El carácter inicio de bloque "{" debe situarse al final de la línea que inicia el bloque y "}" debe situarse en una nueva línea tras la última línea del bloque y alineada con respecto al primer carácter de dicho bloque. Todas las sentencias de un bloque deben encerrarse entre llaves "{...}", aunque el bloque contenga solo una sentencia. Esta práctica permite añadir código sin cometer errores accidentalmente al olvidar añadir las llaves.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Get.isDarkMode?Colors.grey[600]:Colors.white,
      leading: IconButton(
        onPressed: (() => Get.back()),
        icon: Icon(Icons.arrow_back_ios),
        color: Colors.black,


        ),
      title: Text(this.Label.toString().split("|")[0]),
    ),

  );
}
```

Ilustración 9 Estándar de Codificación Sentencias

- Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.

```
int? id;
String? title;
String? note;
int? isCompleted;
String? date;
String? startTime;
String? endTime;
int? color;
int? remind;
String? repeat;
```

Ilustración 10 Estándar de Codificación Una variable por línea

3.2 Prueba de software

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software (SDLC). Las pruebas son la forma en que se puede asegurar la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que realice las pruebas manualmente o a través de la automatización, cuanto antes se lleven a cabo las pruebas, más probable es identificar errores, asegurando que la aplicación de software haya sido revisada y auditada a fondo antes de que esté frente a los usuarios. Existen diversos métodos para realizar las pruebas de software, entre las más importantes se encuentran la prueba de Caja Blanca y prueba de Caja Negra (E. S. Martínez, 2014).

3.2.1 Prueba de caja blanca

Las pruebas de caja blanca describen pruebas o métodos en los que se conocen los detalles y el funcionamiento interno del software que se está probando.

Dado que conoce las funciones, los métodos, las clases, cómo funcionan y cómo se unen, generalmente está mejor equipado para examinar la integridad lógica del código. Las pruebas unitarias suelen ser caja blanca (Devops Latam, 2021).

Las pruebas unitarias se centran en probar piezas/unidades individuales de una aplicación de software al principio del SDLC. Cualquier función, procedimiento, método o módulo puede ser una unidad que se someta a pruebas unitarias para determinar su corrección y comportamiento esperado. Las pruebas unitarias son las primeras pruebas que los desarrolladores realizan durante la fase de desarrollo (Dotcom-Monitor, Inc, 2020).

Un test unitario prueba una sola función, método o clase. El paquete test proporciona el marco principal para escribir pruebas unitarias, y el paquete flutter_test proporciona utilidades adicionales para probar Widgets (Graciano, 2019).

Pasos a seguir para la realización de la prueba:

1. Agregar la dependencia test o flutter_test
2. Crear un archivo de prueba
3. Crear una clase para probarla
4. Escribir un test para nuestra clase
5. Combinar múltiples tests en un grupo
6. Ejecutar las pruebas

```dart
import 'package:flutter_test/flutter_test.dart';
```

Ilustración 11 Importación de la clase flutter_test

```dart
/// Open the PopupMenu
Future<void> openPopupMenu(WidgetTester tester) async {
  final popup = find.byKey(const Key('appMenuButton'));
  expect(popup, findsOneWidget);
  await tester.tap(popup);
  await tester.pump(const Duration(seconds: 1));
}

Future<void> openInterfaceMenu(WidgetTester tester) async {
  /// Open popup menu.
  await openPopupMenu(tester);

  /// Switch the Interface.
  final interface = find.byKey(const Key('interfaceMenuItem'));
  expect(interface, findsOneWidget);
  await tester.tap(interface);
  await tester.pump(const Duration(seconds: 1));
}

Future<void> openApplicationMenu(WidgetTester tester) async {
  /// Open popup menu.
  await openPopupMenu(tester);

  /// Switch the application.
  final application = find.byKey(
```

Ilustración 12 Clase de pruebas unitarias

Resultados de las pruebas unitarias

Para probar el correcto funcionamiento de la aplicación móvil se realizaron tres iteraciones de pruebas. En la tabla que se muestra a continuación se presentan los resultados obtenidos en cada iteración, así como la corrección de cada uno de los errores.

Tabla 3 No Conformidades de las Pruebas Unitarias

No conformidades (NC)	Primera iteración	Segunda iteración	Tercera iteración
Detectadas	4	2	0
Resueltas	2	3	1
Pendientes	2	1	0

En la tabla 7 se evidencia el comportamiento de las NC encontradas durante el proceso de prueba, donde se observa que en la primera iteración se encontraron cuatro NC de validaciones incorrectas, en la segunda iteración se encontraron dos NC, de ellas una al cargar la vista de una tarea no carga la hora seleccionada y la otra la funcionalidad de adicionar tarea da error al pulsar el botón de adicionar.

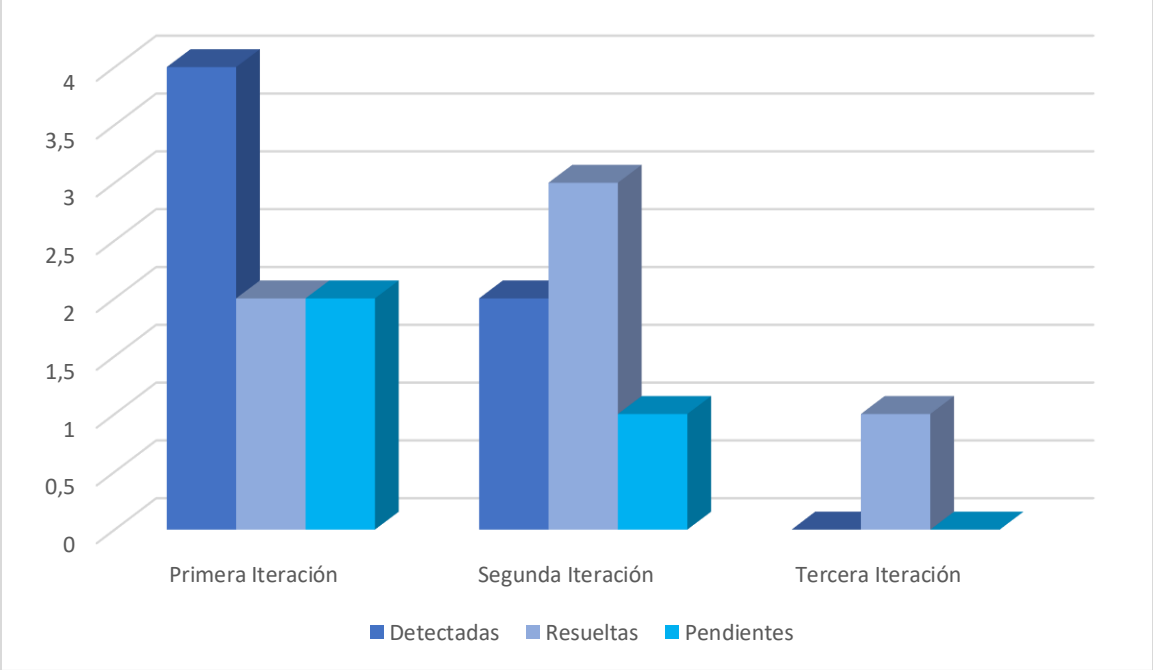


Ilustración 13 Comportamiento de las NC de las pruebas unitarias en cada iteración

3.2.2 Prueba de aceptación

Las pruebas de aceptación son importantes dado que significan la satisfacción del cliente con el producto desarrollado, el final de una iteración y el comienzo de la siguiente. Se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra (Pressman, 2005).

En la realización de las pruebas de aceptación se definieron los siguientes elementos:

- **CU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

La siguiente tabla muestra las pruebas de aceptación del CU1 perteneciente a la primera iteración de sistema. La cual fue escogida por ser una de los procesos más relevantes en el desarrollo del sistema.

Tabla 4 Caso de Prueba perteneciente al RF Insertar Tarea

CU	HU_1
Nombre	Insertar Tarea
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir insertar una nueva tarea en la fecha seleccionada en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.

Condiciones de ejecución	El usuario selecciona la fecha en que desee ingresar la tarea, luego se muestra una vista con el cuestionario correspondiente a una tarea y los botones crear tarea y atrás
Entradas/Pasos de ejecución	Seleccionar fecha y el botón aceptar y aparecen los campos Título, Nota, Fecha, Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota
Resultado esperado	El usuario creo la tarea y se mostró correctamente en el listado de tareas
Evaluación de la prueba	Prueba satisfactoria

Resultados de las pruebas de aceptación

En la siguiente tabla se muestra el comportamiento de las no conformidades encontradas durante el proceso de prueba:

Tabla 5 No Conformidades de las pruebas de aceptación

NC	Primera Iteración	Segunda Iteración	Tercera Iteración
Detectadas	5	1	0
Resueltas	2	2	2
Pendientes	3	2	0

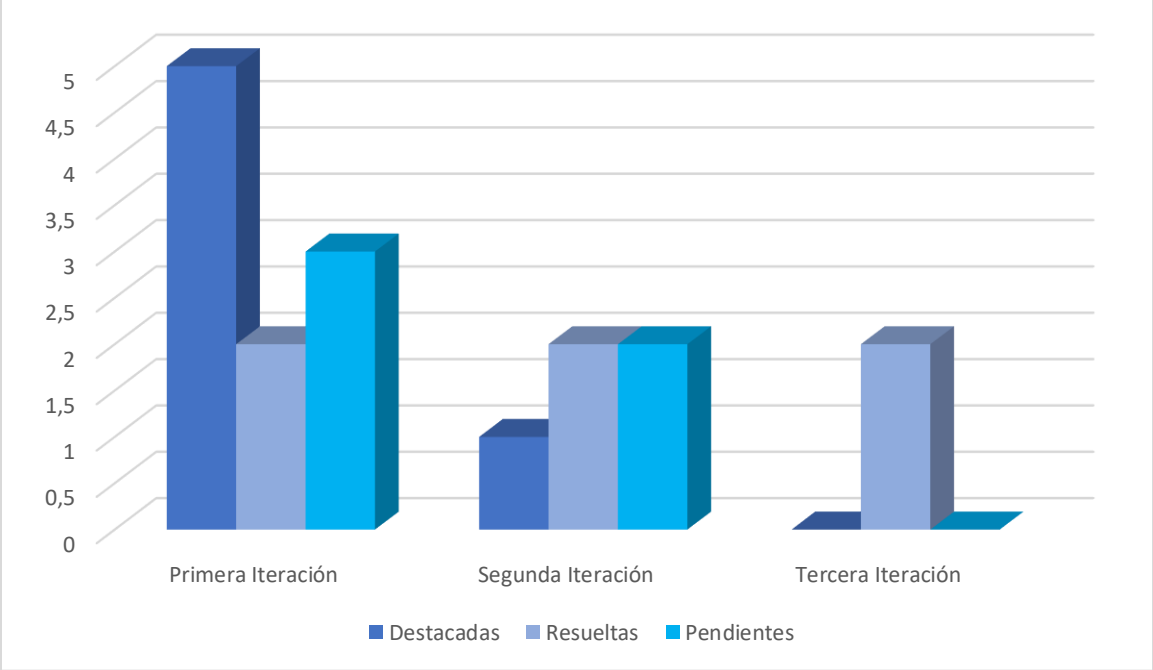


Ilustración 14 Comportamiento de las NC de las pruebas de aceptación en cada iteración

Conclusiones del capítulo

Al concluir este capítulo se termina la realización de la aplicación, obteniendo los siguientes resultados:

- El estándar de codificación seleccionado para el lenguaje de programación Dart permitió organizar el código de la solución.
- La elaboración de los diseños de casos de pruebas permitió identificar los errores del sistema.
- Las pruebas de software aplicadas permitieron validar el correcto funcionamiento de la propuesta de solución.

CONCLUSIONES FINALES

Con la culminación del presente trabajo se ha dado cumplimiento a los objetivos trazados en la investigación, obteniéndose como resultado principal, un módulo de agenda para la gestión de las tareas de una persona. A continuación, se exponen las conclusiones generales a las que se arribó luego del desarrollo de la solución:

- Se diseñó una solución capaz de gestionar las tareas de una persona; además, permite que los demás módulos del asistente consuman de ella. El producto fue desarrollado haciendo uso de estándares y buenas prácticas de programación de acuerdo a lo descrito en el acápite 3.1.1 de la investigación, garantizando limpieza y organización en el código del sistema.
- Se realizaron las pruebas unitarias y de aceptación, permitiendo detectar los errores presentes, corregirlos y entregarle al cliente una aplicación con mayor calidad.

RECOMENDACIONES

Una vez terminado el módulo de agenda para el asistente personal y analizado los resultados obtenidos en la investigación, se mantiene la idea de seguir perfeccionando la solución final, por lo que se recomienda:

- Integrar el módulo de agenda para la gestión de tareas al asistente personal cubano una vez este sea creado.
- Añadir una funcionalidad que le permita al usuario modificar el estado de las tareas.

Referencias Bibliográficas

Álvarez, M. A. (2020, julio 28). *Qué es MVC*. https://desarrolloweb.com/articulos/que-es-mvc.html

Amazon Web Services. (2018, mayo 22). *Alexa ahora le permite programar reuniones privadas y re-programar reuniones en su calendario*. https://aws.amazon.com/es/about-aws/whats-new/2018/05/alexa-now-lets-you-schedule-1-1-meetings-and-move-meetings-in-yo/

Amazon Web Services, Inc. (2022). *¿Qué es Python?* | *Guía de Python para principiantes de la nube* | *AWS*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/python/

Andrés Vicente, E. (2021). *Significado de Agenda*. Significados. https://www.significados.com/agenda/

Becas Santander. (2022). *Becas Santander*. https://app.becas-santander.com/

Betania. (2022, octubre 28). *¿Qué es un servidor web? Todo lo que necesitas saber*. https://www.hostinger.es/tutoriales/que-es-un-servidor-web

Big Data Marketer. (2018, febrero 21). *Virtual Personal Assistants (VPA), la era de los Asistentes Personales Virtuales—Big Data Social*. https://www.bigdata-social.com/virtual-personal-assistants-vpa-la-era-de-los-asistentes-personales-virtuales/

Blancarte Iturralde, O. J. (2020). *Introducción a la arquitectura de software* (Primera Edición). https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Cárdenas Campaña, J. A. (2022). *Bajo acoplamiento · Atributos de calidad y patrones de diseño*. https://fjimenezg.gitbooks.io/atributos-de-calidad-y-patrones-de-diseno/content/bajo-acoplamiento.html

Ceballos, F. J. (2021). *Lenguajes de Programación*. https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html#

Cernadas, I. (2019, diciembre 4). Google Calendar: ¿cómo usar y aprovechar todas sus ventajas? *Marketeros de Hoy*. https://marketerosdehoy.com/marketing-digital/google-calendar/

Cinergix Pty. Ltd. (2022, octubre 18). *Tutorial de Diagrama de Despliegue* | *¿Qué es un Diagrama de Despliegue*. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/

Columnistas, P. de. (2019, noviembre 13). *Las ventajas de administrar el tiempo eficientemente*. https://www.themarkethink.com/mercadotecnia/las-ventajas-de-administrar-el-tiempo-eficientemente/

Cordón, M. J. M. (2021, junio 30). ¿Qué es el lenguaje de programación Dart? *Blog de Hiberus Tecnología*. https://www.hiberus.com/crecemos-contigo/que-es-el-lenguaje-de-programacion-dart/

Coursera. (2022, noviembre 14). *What Is Python Used For? A Beginner's Guide*. Coursera. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Del Sol Fabregat, L. A., Tejeda Castañeda, E., & Mirabal Díaz, J. M. (2017). Los métodos teóricos: Una necesidad de conocimiento en la investigación científico-pedagógica. *EDUMECENTRO*, *9*(4), 250-253.

Devops Latam. (2021, abril 15). *15 métodos de prueba que todos los desarrolladores deben conocer—DevOps Latam*. https://devopslatam.com/15-metodos-de-prueba-que-todos-los-desarrolladores-deben-conocer/

Dotcom-Monitor, Inc. (2020, octubre 16). *Tipos de pruebas de software: Diferencias y ejemplos—LoadView*. https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/

EdPrice-MSFT. (2021). *Estilo de arquitectura de n niveles—Azure Architecture Center*. https://learn.microsoft.com/es-es/azure/architecture/guide/architecture-styles/n-tier

Flores, F. (2022, julio 22). *Qué es Visual Studio Code y qué ventajas ofrece*. OpenWebinars.net. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

Frutos, A. M. de. (2018, junio 2). *¿Qué es Google Assistant?* Computer Hoy. https://computerhoy.com/noticias/tecnologia/que-es-google-assistant-257409

González, A. (2016, noviembre). *Defínción de Asistente Virtual*. Definición ABC. https://www.definicionabc.com/tecnologia/asistente-virtual.php

Graciano, E. (2019, marzo 5). *Introducción a las pruebas unitarias*. https://flutter-es.io/docs/cookbook/testing/unit/introduction

Granollers, T. (2021). *Modelo Mental y Modelo Conceptual | Curso de Interacción Persona-Ordenador*. https://mpiua.invid.udl.cat/fases-mpiua/diseno/modelo-mental-y-modelo-conceptual/

Gutiérrez Cabrera, E. (2018). *Soluciones informáticas*.

Hamilton, D. (2020, abril 24). *Estándares de codificación de software y pautas de programación*. Parasoft. https://es.parasoft.com/blog/an-ounce-of-prevention-software-safety-security-through-coding-standards/

Heffelfinger, D. R. (2022). *Payara Micro Revealed*. Apress, Berkeley, CA.

Hernández, L. (2018, julio 19). *Diseña un modelo conceptual que combine con diferentes modelos mentales | by Leonardo Hernández | Medium*. https://medium.com/@leonardo.hernandez/dise%C3%B1a-un-modelo-conceptual-que-combine-con-diferentes-modelos-mentales-7401cce8b82e

Hernández Pérez, L. E. (2017, febrero 13). *Visual Paradigm para UML es una herramienta para desarrollo*. prezi.com. https://prezi.com/3v2w1mpsptxx/visual-paradigm-para-uml-es-una-herramienta-para-desarrollo/

Jaiswal, S. (2021). *Flutter: What is Dart Programming - Javatpoint*. Www.Javatpoint.Com. https://www.javatpoint.com/flutter-dart-programming

Khaild, T. (2022, agosto 14). Python Tortoise ORM Integration with FastAPI. *Nerd For Tech*. https://medium.com/nerd-for-tech/python-tortoise-orm-integration-with-fastapi-c3751d248ce1

Kord, M. (2017, diciembre 3). *Patrones GRASP. Patrones GoF. Diferencia entre GRASP y GoF - TuxNots*. https://sites.google.com/site/tuxnots/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof

Letelier, P., & Penadés, M. C. (2019, abril 15). *Metodologías 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/v5n2a1.htm

Martin, T. (2016, octubre 13). *How to make a to-do list with Alexa*. CNET. https://www.cnet.com/home/smart-home/how-to-make-a-to-do-list-with-alexa/

Martínez Canelo, M. (2020, junio 24). Qué son los Patrones de Diseño de software / Design Patterns. *Profile Software Services*. https://profile.es/blog/patrones-de-diseno-de-software/

Martínez, E. S. (2014, diciembre 31). Procedimiento para realizar pruebas de Caja Blanca. *Informática Jurídica*. https://www.informatica-juridica.com/trabajos/procedimiento-realizar-pruebas-caja-blanca/

Martínez, M. (2022, septiembre 20). *Qué es Google Assistant y qué puede hacer en el móvil*. MovilZona. https://www.movilzona.es/tutoriales/software/google-assistant/

Mesa, L. (2022). *FastAPI: Cómo construir una API con Python en media hora. - Sngular*. https://www.sngular.com/es/fastapi-como-construir-una-api-con-python-en-una-media-hora/

Muradas, Y. (2018, marzo 23). *SQLite para Android: La herramienta definitiva*. OpenWebinars.net. https://openwebinars.net/blog/sqlite-para-android-la-herramienta-definitiva/

Pérez, M. (2018). *Patrones de Diseño*.

Pressman, R. S. (2005). *Ingeniería del Software Un enfoque práctico* (Séptima Edición). www.FreeLibros.me

Rodríguez Sánchez, T. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI*. Universidad de las Ciencias Informáticas.

Ruchir, C. (2021, enero 2). *¿Por qué debería elegir Flutter sobre el desarrollo de aplicaciones nativas en 2022?* https://www.cisin.com/coffee-break/es/technology/choose-flutter-over-native-app-development-in-2022.html

Siriwardena, P. (2020). *Advanced API Security* (Segunda Edición). Apress Berkeley, CA.

Solving Ad Hoc. (2017, diciembre 18). Qué son las historias de usuario y su función en Agilidad. *Solving Ad Hoc - Resolviendo a medida tus necesidades de cambio*. https://solvingadhoc.com/las-historias-usuario-funcion-agilidad/

Sommerville, I. (2011). *INGENIERÍA DE SOFTWARE* (Novena Edición).

Tecnologismo. (2022). ¦› *Haga que Siri muestre su calendario y sus citas en iPhone, iPad o Mac*. https://tecnologismo.com/haga-que-siri-muestre-su-calendario-y-sus-citas-en-iphone-ipad-o-mac/

TI Consultors S.L. (2019, julio 10). ¿Qué es Flutter? - Desarrollo de Aplicaciones móviles | Aures Tic. *Aurestic*. https://aurestic.es/que-es-flutter/

Vallejo Guerra, R. A. (2022). *Experto en información · Atributos de calidad y patrones de diseño*. https://fjimenezg.gitbooks.io/atributos-de-calidad-y-patrones-de-diseno/content/experto-en-informacion.html

Valois, M. A. (2019, septiembre 20). *Alexa y Siri: ¿Sabes qué es un asistente virtual inteligente?* https://www.hostgator.mx/blog/alexa-siri-asistente-virtual-inteligente/

WEBEMPRESA EUROPA S.L. (2022). *¿Qué es Nginx y cómo funciona? - Webempresa*. https://www.webempresa.com/hosting/nginx-que-es.html

ANEXOS

Tabla 6 HU Modificar Tarea

Historia de usuario	
Número: 2	**Usuario:** Programador
Nombre: Modificar Tarea	
Prioridad: Alta	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 5 días	
Descripción: **1-Objetivo:** Permitir modificar una tarea **2-Condiciones para lograr el objetivo:** **-**Debe existir al menos una tarea **3-Flujo de la acción a realizar:** Cuando el usuario selecciona el botón para modificar una tarea, el sistema muestra una nueva ventana con los campos a modificar (Título, Nota, Fecha, Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota) llenos con la información actual y los botones de guardar y cancelar. El usuario actualiza los campos y presiona guardar, el sistema guarda de manera consistente la tarea nueva y envía la solicitud de actualizar las tareas en la API.	
Observaciones:	

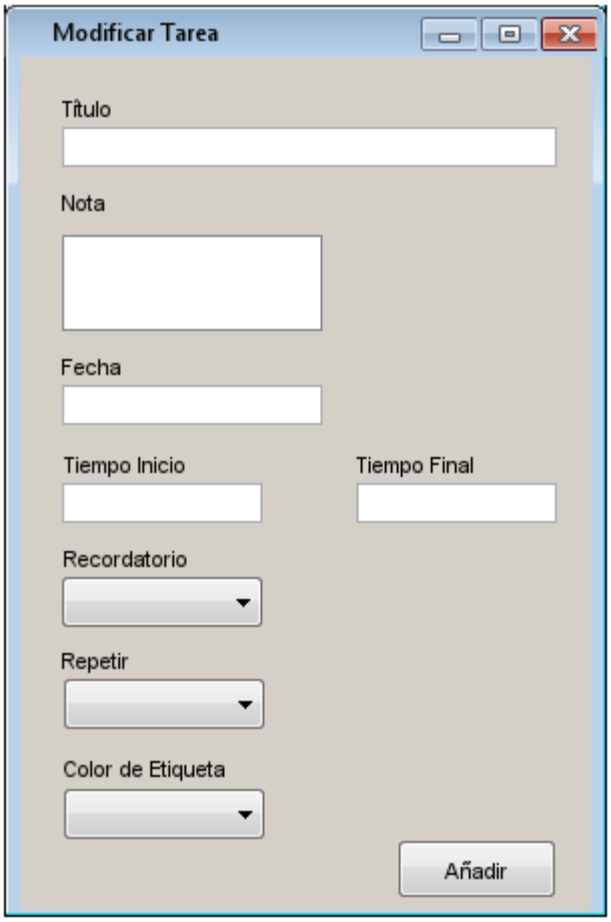


Tabla 7 HU Eliminar Tarea

Historia de usuario	
Número: 3	**Usuario:** Programador
Nombre: Eliminar Tarea	
Prioridad: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 2 días	
Descripción: **1-Objetivo:** Permitir eliminar una tarea **2-Condiciones para lograr el objetivo:** **-**Debe existir al menos una tarea. **3-Flujo de la acción a realizar:** El usuario selecciona la tarea que desea eliminar y el sistema muestra los botones de eliminar y modificar. El usuario presiona eliminar, el sistema elimina y manda la solicitud de actualizar la API.	

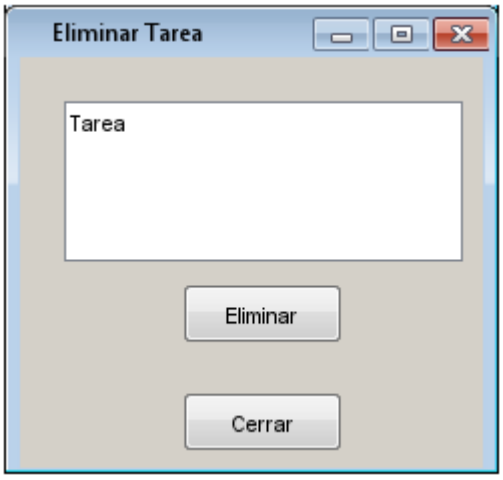
Observaciones:


Tabla 8 HU Visualizar Tareas

Historia de usuario	
Número: 4	**Usuario:** Programador
Nombre: Visualizar Tareas	
Prioridad: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 1 día	
Descripción: **1-Objetivo:** Permitir visualizar las tareas **2-Condiciones para lograr el objetivo:** **-**Debe existir al menos una tarea. **3-Flujo de la acción a realizar:** El usuario presiona el día del que desee ver las tareas y el sistema muestra las tareas pertenecientes al día seleccionado.	

Tabla 9 HU Configurar Alarma

Historia de usuario	
Número: 5	**Usuario:** Programador
Nombre: Configurar Alarma	
Prioridad: Media	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 2 días	
Descripción: **1-Objetivo:** Permitir configurar la alarma	

2-Condiciones para lograr el objetivo: **-**Se debe crear o modificar una tarea **3-Flujo de la acción a realizar:** El sistema comprueba que coincidan las fechas de alguna tarea con la suya propia y ejecuta la alarma.

Tabla 10 HU Establecer Recordatorio

Historia de usuario	
Número: 6	**Usuario:** Programador
Nombre: Establecer Recordatorio	
Prioridad: Media	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 1 día	
Descripción: **1-Objetivo:** Permitir crear una notificación **2-Condiciones para lograr el objetivo:** **-**Se debe crear o modificar una tarea **3-Flujo de la acción a realizar:** El usuario selecciona la tarea que desee modificar o la fecha en que desee crear una tarea y establece el tiempo de antelación del recordatorio, el sistema muestra una notificación a la hora correspondiente.	

Tabla 11 Actualizar listado de tareas

Historia de usuario	
Número: 7	**Usuario:** Programador
Nombre: Actualizar listado de tareas	
Prioridad: Media	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 1 día	
Descripción: **1-Objetivo:** Permitir actualizar el listado de tareas que se muestran en pantalla **2-Condiciones para lograr el objetivo:** -El dispositivo debe estar conectado a la API **3-Flujo de la acción a realizar:** El usuario presiona el botón tareas y el sistema muestra la lista de tareas correspondientes al día seleccionado	

Tabla 12 HU Conectar la aplicación a la API

Historia de usuario

Número: 8	**Usuario:** Programador
Nombre: Conectar la aplicación a la API	
Prioridad: Media	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 3 día	
Descripción: **1-Objetivo:** Conectar la aplicación a la API **2-Condiciones para lograr el objetivo:** **-**La API debe estar pública en la red. **3-Flujo de la acción a realizar:** El sistema se conecta a la API y envía la solicitud de actualización de las tareas.	

Tabla 13 HU Autenticar usuario

Historia de usuario	
Número: 9	**Usuario:** Programador
Nombre: Autenticar usuario	
Prioridad: Alta	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 3 día	
Descripción: **1-Objetivo:** Identificar el usuario que está usando la aplicación **2-Condiciones para lograr el objetivo:** **-**El usuario debe de estar registrado **3-Flujo de la acción a realizar:** El usuario abre la aplicación e introduce las credenciales y de ser correctas el sistema carga la aplicación con la información perteneciente al usuario.	
Observaciones:	

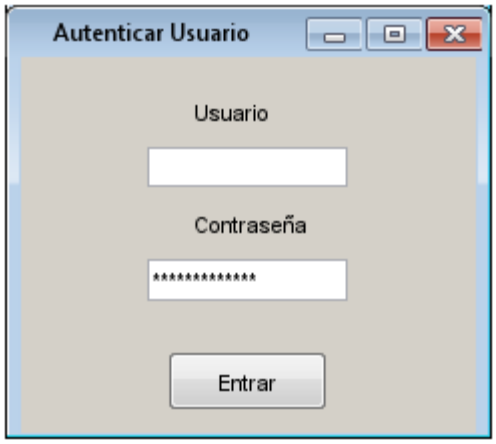


Tabla 14 HU Registrar usuario

Historia de usuario	
Número: 10	**Usuario:** Programador
Nombre: Registrar usuario	
Prioridad: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 3 día	
Descripción: **1-Objetivo:** Crear un usuario **2-Flujo de la acción a realizar:** El usuario llena los campos del cuestionario y presiona Registrar y el sistema envía un correo de confirmación	
Observaciones: 	

Tabla 15 HU Editar usuario

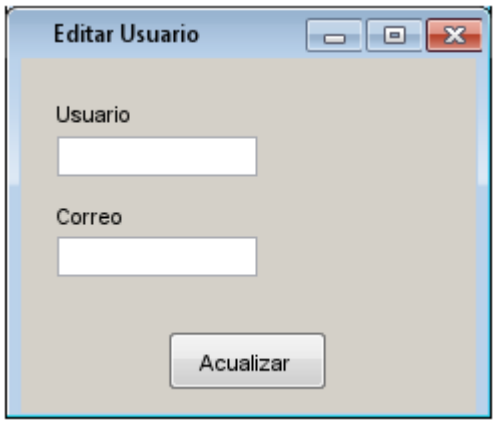
Historia de usuario	
Número: 11	**Usuario:** Programador
Nombre: Editar usuario	
Prioridad: Baja	**Riesgo en desarrollo:** Medio
Tiempo estimado: 2 días	
Descripción: **1-Objetivo:** Modificar el perfil del usuario **2-Condiciones para lograr el objetivo:** **-**Debe existir el usuario **3-Flujo de la acción a realizar:** El usuario modifica los campos que desee y el sistema guarda los cambios	
Observaciones: 	

Tabla 16 HU Cerrar sesión

Historia de usuario	
Número: 12	**Usuario:** Programador
Nombre: Cerrar sesión	
Prioridad: Baja	**Riesgo en desarrollo:** Medio
Tiempo estimado: 2 días	
Descripción: **1-Objetivo:** Garantizar que solo el usuario pueda usar la aplicación **2-Flujo de la acción a realizar:** El usuario presiona salir y el sistema cierra la sesión.	
Observaciones:	

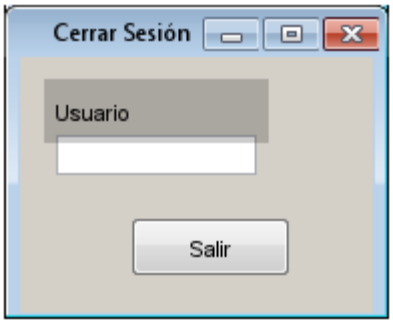


Tabla 17 HU Verificar registro

Historia de usuario	
Número: 13	**Usuario:** Programador
Nombre: Verificar registro	
Prioridad: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 4 días	
Descripción: **1-Objetivo:** Acreditar que la persona que se está registrando es el propietario del correo insertado. **2-Flujo de la acción a realizar:** El usuario accede con el link de verificación y el sistema muestra ``Ya puede usar la aplicación´´	

Tabla 18 HU Recuperar contraseña

Historia de usuario	
Número: 14	**Usuario:** Programador
Nombre: Recuperar contraseña	
Prioridad: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 4 días	
Descripción: **1-Objetivo:** El usuario vuelve a tener acceso a su cuenta **2-Flujo de la acción a realizar:** El usuario presiona contraseña olvidada, el sistema envía un código al correo, el usuario lo introduce junto con la nueva contraseña y el sistema actualiza la información en la base de datos.	
Observaciones:	

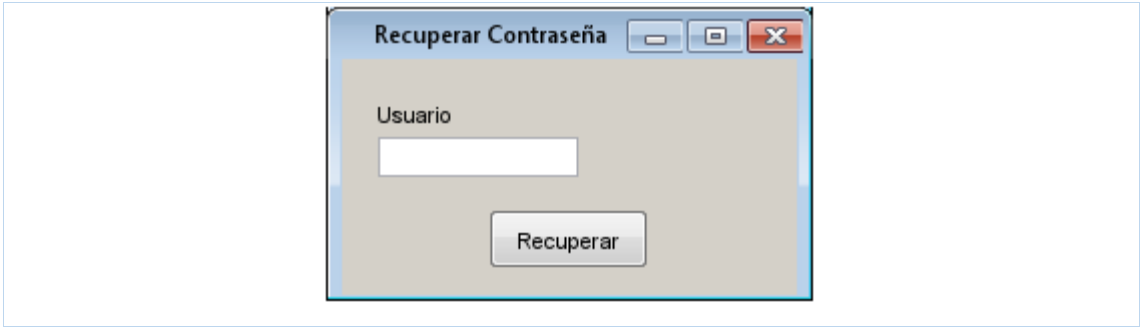


Tabla 19 Caso de Prueba perteneciente al RF Modificar Tarea

CU	HU_2
Nombre	Modificar Tarea
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir modificar la tarea seleccionada en caso de haber llenado correctamente todos los campos, en caso contrario deberá mostrar una notificación de campos requeridos.
Condiciones de ejecución	El usuario selecciona la tarea que desee modificar, luego se muestra una vista con el cuestionario correspondiente a una tarea y los botones crear tarea y atrás
Entradas/Pasos de ejecución	Seleccionar la tarea y el botón modificar y aparecen los campos Título, Nota, Fecha, Hora de Inicio, Hora de Fin, Recordatorio, Repetición y de qué color (azul, rojo y verde) desea que sea la nota
Resultado esperado	El usuario modifico la tarea y se mostró correctamente en el listado de tareas
Evaluación de la prueba	Prueba satisfactoria

Tabla 20 Caso de Prueba perteneciente al RF Eliminar Tarea

CU	HU_3

Nombre	Eliminar Tarea
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir eliminar la tarea seleccionada
Condiciones de ejecución	El usuario selecciona la tarea y presiona eliminar
Entradas/Pasos de ejecución	Seleccionar la tarea y presionar eliminar
Resultado esperado	El usuario elimino la tarea y se quita del listado de tareas
Evaluación de la prueba	Prueba satisfactoria

Tabla 21 Caso de Prueba perteneciente al RF Visualizar Tareas

CU	HU_4
Nombre	Visualizar Tareas
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir visualizar las tareas del día seleccionado
Condiciones de ejecución	El usuario selecciona la fecha de la que desee visualizar las tareas
Entradas/Pasos de ejecución	El usuario selecciona la fecha de la que desee visualizar las tareas
Resultado esperado	El usuario visualiza las tareas
Evaluación de la prueba	Prueba satisfactoria

Tabla 22 Caso de Prueba perteneciente al RF Configurar Alarma

CU	HU_5
Nombre	Configurar Alarma
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran

Descripción de la prueba	La aplicación debe permitir la creación de alarmas del sistema asociadas a las tareas creadas
Condiciones de ejecución	El usuario debe de estar creando o modificando una tarea
Entradas/Pasos de ejecución	El usuario selecciona la fecha de la que desee crear la tarea o la tarea que desee modificar
Resultado esperado	Al completar la acción de crear o modificar debe haberse configurado una nueva alarma en el sistema
Evaluación de la prueba	Prueba satisfactoria

Tabla 23 Caso de Prueba perteneciente al RF Establecer Recordatorio

CU	HU_6
Nombre	Establecer Recordatorio
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir la creación de notificaciones programadas para ser mostradas en la hora que determine el usuario
Condiciones de ejecución	El usuario selecciona la fecha de la que desee crear la tarea o la tarea que desee modificar
Entradas/Pasos de ejecución	El usuario selecciona la fecha de la que desee crear la tarea o la tarea que desee modificar
Resultado esperado	El sistema muestra la notificación a la hora correspondiente
Evaluación de la prueba	Prueba satisfactoria

Tabla 24 Caso de Prueba perteneciente al RF Actualizar listado de tareas

CU	HU_7
Nombre	Actualizar listado de tareas
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir acceder a la API y obtener el listado de tareas
Condiciones de ejecución	El usuario selecciona el botón Tareas estando conectado a la API
Entradas/Pasos de ejecución	El usuario selecciona el botón Tareas
Resultado esperado	El sistema muestra la lista de tareas correspondiente al día seleccionado
Evaluación de la prueba	Prueba satisfactoria

Tabla 25 Caso de Prueba perteneciente al RF Conectar la aplicación a la API

CU	HU_8
Nombre	Conectar la aplicación a la API
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	Al abrir la aplicación la aplicación debe establecer conexión con la API
Condiciones de ejecución	El dispositivo debe tener acceso a la red
Entradas/Pasos de ejecución	El usuario abre la aplicación
Resultado esperado	Al iniciar el sistema se muestra la lista de tareas correspondiente al día actual
Evaluación de la prueba	Prueba satisfactoria

Tabla 26 Caso de Prueba perteneciente al RF Autenticar usuario

CU	HU_9
Nombre	Autenticar usuario
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran

Descripción de la prueba	La aplicación debe verificar las credenciales y dar acceso a la aplicación solo en caso de ser correctas
Condiciones de ejecución	La aplicación debe tener acceso a internet
Entradas/Pasos de ejecución	El usuario abre la aplicación e introduce las credenciales.
Resultado esperado	El sistema carga la aplicación con la información perteneciente al usuario.
Evaluación de la prueba	Prueba satisfactoria

Tabla 27 *Caso de Prueba perteneciente al RF Registrar usuario*

CU	HU_10
Nombre	Registrar usuario
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	Luego de llenar los campos se presiona ``Registrar´´ y el sistema debe enviar un correo de verificación.
Condiciones de ejecución	La aplicación debe tener acceso a internet
Entradas/Pasos de ejecución	El usuario llena los campos y presiona registrar.
Resultado esperado	El sistema envía el correo
Evaluación de la prueba	Prueba satisfactoria

Tabla 28 Caso de Prueba perteneciente al RF Editar usuario

CU	HU_11
Nombre	Editar usuario
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	La aplicación debe permitir modificar los datos del usuario
Condiciones de ejecución	La aplicación debe tener acceso a internet

Entradas/Pasos de ejecución	El usuario introduce los nuevos datos y presiona actualizar perfil
Resultado esperado	El sistema guarda los cambios en la base de datos
Evaluación de la prueba	Prueba satisfactoria

Tabla 29 Caso de Prueba perteneciente al RF Cerrar sesión

CU	HU_12
Nombre	Cerrar sesión
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	El sistema debe permitir cerrar sesión
Condiciones de ejecución	La aplicación debe tener acceso a internet
Entradas/Pasos de ejecución	El usuario presiona ``Salir´´
Resultado esperado	El sistema cierra la sesión
Evaluación de la prueba	Prueba satisfactoria

Tabla 30 Caso de Prueba perteneciente al RF Verificar registro

CU	HU_13
Nombre	Verificar registro
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	El sistema debe permitir verificar el registro de un usuario
Condiciones de ejecución	La aplicación debe tener acceso a internet
Entradas/Pasos de ejecución	El usuario abre el link de verificación
Resultado esperado	El sistema muestra el mensaje de ``Ya puede usar la aplicación´´
Evaluación de la prueba	Prueba satisfactoria

Tabla 31 Caso de Prueba perteneciente al RF Recuperar contraseña

CU	HU_14

Nombre	Recuperar contraseña
Nombre de la persona que realiza la prueba	Julio Alberto Leiva Duran
Descripción de la prueba	El sistema permite cambiar la contraseña en caso de olvido demostrando acceso al correo electrónico
Condiciones de ejecución	La aplicación debe tener acceso a internet
Entradas/Pasos de ejecución	El usuario presiona ``Olvido de Contraseña´´, el sistema envía un código al correo y el usuario introduce la nueva contraseña junto al código y presiona confirmar
Resultado esperado	El sistema actualiza la contraseña correctamente
Evaluación de la prueba	Prueba satisfactoria