Universidad de las Ciencias Informáticas
Facultad 3

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

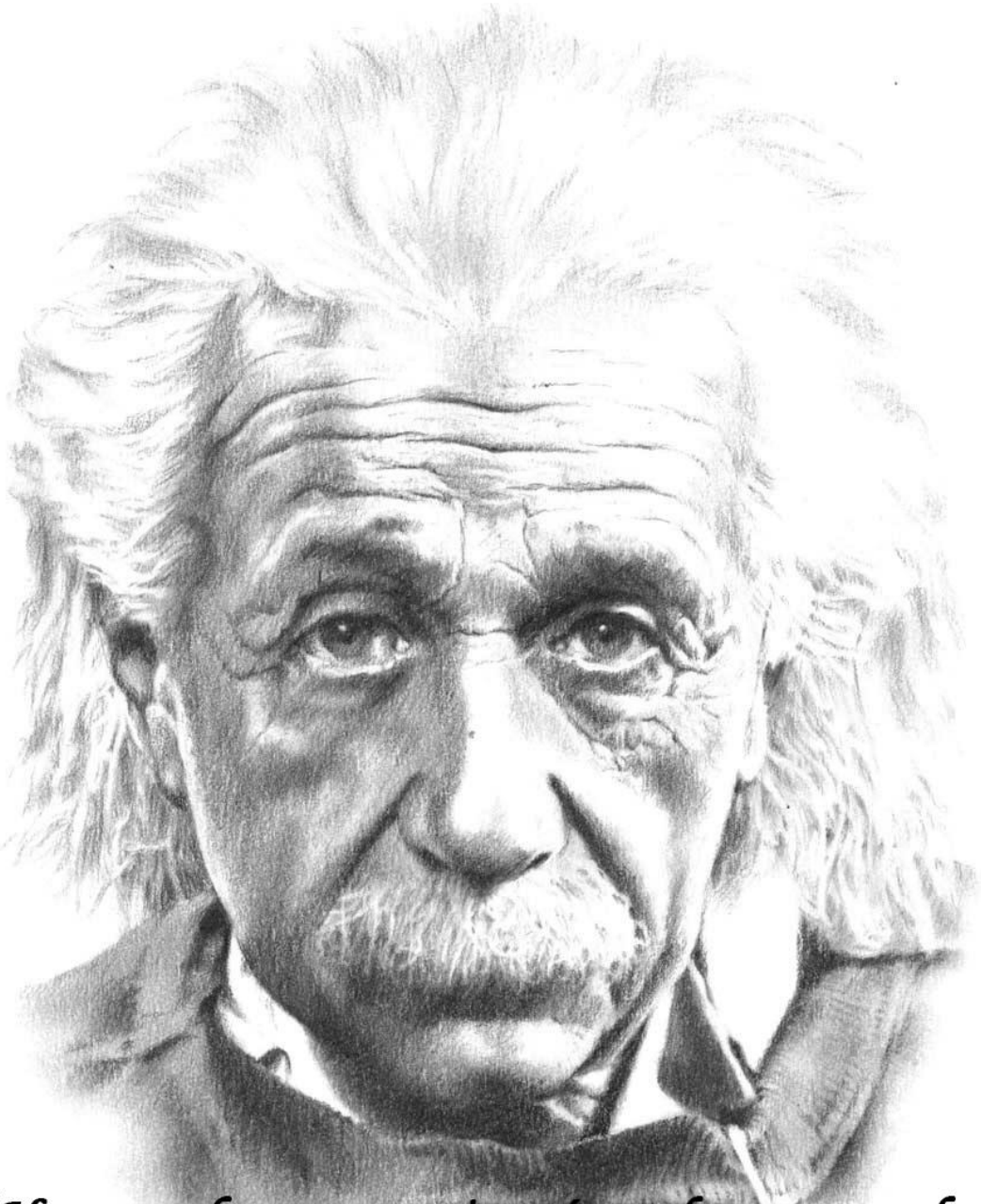
Título: Aplicación móvil en Android con la información estadística de la planificación en tiempo real para el Sistema de Planificación de Actividades 3

Autor: Yasniel Samper Hernández

Tutora: Ing. Liset Gómez Chávez

La Habana, junio de 2018

"Año 60 de la Revolución"



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".

Albert Einstein

Declaración de autoría

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Autor: Yasniel Samper Hernández

_____

Tutora: Ing. Liset Gómez Chávez

Dedicatoria

A mi madre por su cariño, apoyo incondicional y por ser la promotora de todos mis logros, en particular este que es especialmente para ti.

Agradecimientos

A mi mamá, la persona más importante en mi vida, por estar siempre a mi lado, en los buenos y malos momentos, por apoyarme y ayudarme, es evidente que no podría expresar todo lo que siento por ti y todo lo que tengo que agradecerte en tan cortas líneas.

A mis hermanos por secar lo mejor de mí en todo momento. Tanto como Raimundo como Owar y los de Ciego Omar, Yamilka y Llaylka, a todos los quiero mucho.

A todas esas personas que formaron parte de mi vida en la universidad.

A los chuchumecos que ellos saben bien quienes son así que para que mencionarlos, pero por si acaso: Abelito, Chandy, Arielito, y si alguien se queda por favor disculpen.

Especial agradecimiento a mi esposa y madre de mi hija por darme su apoyo incondicional y lo más lindo que tengo en la vida que es mi pequeñita Alisson. Por venir arrastrando conmigo desde hace unos años ya, Te amo mi reina.

A Robe, gracias mi hermano por toda tu ayuda a pesar del poco tiempo que tiene nuestra amistad.

Adrián mi hermano, gracias por las madrugadas sin dormir y sé que no hay forma posible de pagarte lo que has hecho por mí.

A mi tutora Liset por cargar conmigo todo este tiempo que sé que no ha sido fácil.

Agradecer en general a todos los profesores que de una forma u otra me ayudaron o no, en el largo paso por esta casa de altos estudios.

Resumen

El Sistema de Planificación de Actividades se basa en la instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la planificación de los objetivos y actividades. Actualmente facilita la gestión de las actividades a todos los niveles organizacionales, permite interrelacionar objetivos de trabajo y actividades en tiempo real, garantizando el desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Sin embargo, no posibilita el análisis de la información unificada para evaluar el comportamiento de diferentes indicadores. Los reportes que agrupan la totalidad de la misma para su posterior análisis se torna de manera engorrosa, ya que su forma de visualización no es la adecuada. Incidiendo así, de manera negativa en la toma de decisiones y en la prevención de comportamientos futuros de las entidades.

Para la construcción del sistema se realiza un estudio de la metodología a utilizar para guiar el proceso de desarrollo de software, así como de las principales herramientas y tecnologías utilizadas en la implementación de la solución. Además, se describe la arquitectura que da soporte al sistema desarrollado. La implementación de la solución propuesta fue validada a través de pruebas de caja blanca, caja negra, unitarias y aceptación. Además, se valida la presente investigación haciendo uso de la técnica de IADOV con el objetivo de medir el índice de satisfacción grupal con respecto a la aplicación SIPACgraphic. Con el desarrollo de la presente investigación se contribuye a mejorar la forma de visualizar la información a través de gráficas con el objetivo de favorecer los procesos de Ejecución y Control de la Planificación Estratégica y Operativa.

Palabras clave: información, gráficas, planificación estratégica y operativa, sistema de planificación de actividades.

Contenido

Índice de tablas

Índice de figuras

Introducción

En la actualidad la planificación estratégica y operativa constituye un nuevo modelo de planificación que tiene como principal función, planificar y organizar los procesos de: elaboración, aprobación-conciliación, puntualización, ejecución y control, así como la evaluación de los objetivos. La misma se encarga de establecer resultados finales hacia los cuales se dirigen las actividades organizacionales e individuales, en aras de cumplir los objetivos económicos y sociales que demanda el desarrollo integral de la sociedad cubana. La puesta en funcionamiento en las entidades de este nuevo modelo de planificación, posibilita observar organizadamente la totalidad de las actividades a cumplir por parte de los involucrados. En correspondencia con esta premisa, se garantiza la planificación eficaz y una petición más acertada de los recursos; lo que posibilita que se defina una prioridad en los objetivos trazados por la entidad.

La dirección del Estado Cubano en conjunto con el Centro de Informatización de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) desarrollaron el Sistema de Planificación de Actividades (SIPAC) basado en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la Planificación de los objetivos y actividades en los órganos, Organismos de la Administración Central de Estado, entidades nacionales y Administraciones locales del Poder Popular. En el apartado h del acápite primero, referente a los principios fundamentales que debe cumplir el proceso de planificación de los objetivos de trabajo y de las actividades, se define como uno de estos principios, el registro y control sobre la marcha del cumplimiento de los planes, mediante un sistema de registro, comprobación y monitoreo, que permita asegurar el cumplimiento de los planes (1). Este sistema está destinado a facilitar la gestión de las actividades a todos los niveles organizacionales, permite interrelacionar objetivos de trabajo y actividades en tiempo real. También crea el flujo de información hacia los diferentes niveles a partir de la definición de estados y transiciones, así como la recuperación de la información de acuerdo al modelo de planificación actual a través de reportes. Todas estas operaciones que SIPAC permite realizar generan información estratégica u operativa, la primera es la base de la toma de decisiones a largo plazo en las entidades y organismos, la segunda es la información latente de la planificación.

Independientemente de las funcionalidades que brinda el SIPAC, no da la posibilidad de analizar la información unificada para evaluar el comportamiento de los diferentes indicadores, esto se traduce en:

1. Obtener la cantidad de tareas por estado, asociadas a un usuario en un período determinado.
2. Obtener la cantidad de tareas por tipo de actividad, asociadas a un usuario en un período determinado.
3. Obtener la cantidad de tareas extraplanes asociadas a un usuario en un período determinado.
4. Mostrar el porciento de cumplimiento de los planes mensuales de un período determinado.

5. Conocer la cantidad de Factores que Influyen en el Plan (FIP) para cada usuario involucrado en el proceso de planificación de actividades.
6. Mostrar el estado de cumplimiento de los objetivos.
7. Mostrar la cantidad de Factores que Influyen en el Plan (FIP) para cada usuario involucrado en el proceso de planificación de actividades para un período de tiempo.

Este conjunto de factores influye en el tiempo necesario para el análisis y estudio de la información. La obtención de resultados medibles asociados a cada área de análisis en un período determinado, se torna engorrosa, a partir de la forma en que se obtienen los reportes. Todo esto puede incidir de manera negativa en la toma de decisiones y en la prevención de comportamientos futuros de las entidades.

A partir de lo antes descrito surge como **problema a resolver:** ¿Cómo obtener información de la planificación en tiempo real de las actividades, en contribución al seguimiento de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa?

La investigación se centra en el **objeto de estudio**: visualización de información mediante gráficas, enmarcada en el **campo de acción:** visualización de información mediante gráficas en aplicaciones Android.

Para dar solución al problema planteado se traza como **objetivo general:** desarrollar una aplicación informática para dispositivos Android, que permita la obtención de gráficas sobre el cumplimiento de la planificación de actividades en el sistema SIPAC.

Se definieron las siguientes tareas de la investigación**:**

1. Definición de los elementos teóricos y metodológicos asociados a la visualización de información mediante gráficas.
2. Estudio del estado actual de las soluciones informáticas que permiten la visualización de información mediante gráficas y que están en correspondencia con el campo de acción.
3. Análisis de los lenguajes y herramientas, para seleccionar los más adecuados para el desarrollo de la propuesta de solución.
4. Diseño de la solución propuesta para la posterior implementación de la misma.
5. Implementación de la propuesta de solución, de acuerdo a la metodología, lenguajes y herramientas seleccionados y en correspondencia con las necesidades de gestión.
6. Validación de la propuesta de solución, para comprobar el correcto funcionamiento de la misma, de acuerdo a la estrategia de prueba definida.

Métodos de investigación:

Los principales **métodos teóricos** utilizados son:

- **Analítico-Sintético:** se utilizó para el análisis de la bibliografía relacionada con el tema de investigación, que permite realizar un resumen de los elementos más importantes de los procesos y Sistemas de planificación estadística. Facilita la selección de las tecnologías y herramientas para el desarrollo de la propuesta de solución.
- **Histórico-Lógico:** fue empleado para realizar un estudio del estado del arte analizando sus principales funcionalidades, características y ventajas.
- **Inductivo-Deductivo:** a partir de este método se logró identificar las generalidades de los Sistemas de planificación estadística a partir del estudio de un conjunto de soluciones informáticas y deducir las características particulares que debe poseer la propuesta de solución.

Los **métodos empíricos** utilizados son**:**

- **Entrevista:** Proporciona datos importantes acerca de las necesidades de los usuarios en el sistema, además sobre su satisfacción una vez terminado el producto.

El presente trabajo de diploma está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica: en este capítulo se abordan los elementos teóricos y metodológicos asociados a la visualización de información mediante gráficas. Se realiza un análisis del estado actual de las soluciones informáticas que se corresponden con el campo de acción. Se fundamenta la elección de la metodología de desarrollo a utilizar, teniendo en cuenta los elementos y fases que la componen. Además, se definen las herramientas y lenguajes de programación necesarios para la implementación de la propuesta de solución.

Capítulo 2: Análisis y diseño de la propuesta de solución: en este capítulo se realiza una descripción de la aplicación propuesta como solución y sus principales funcionalidades. Se argumenta el uso de los patrones de diseño y se detalla las particularidades de la misma, en correspondencia con la metodología seleccionada.

Capítulo 3: Implementación y validación de la solución: en este capítulo se describen los estándares de codificación empleados en la implementación de la propuesta de solución. Se muestran los elementos necesarios para la utilización de la aplicación, a través del diagrama de despliegue. Se describe la estrategia de prueba definida para validar la aplicación y se realiza un análisis de los resultados obtenidos luego de aplicadas las mismas.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se esboza el marco teórico, donde se abordan los principales conceptos y definiciones necesarios para el desarrollo de la investigación. Se describen los sistemas existentes que responden al objeto de estudio y campo de acción de la investigación, ya sea en el escenario nacional o internacional, realizando posteriormente una valoración de los mismos. Además, se explica la selección de la metodología de desarrollo de software y se fundamenta mediante un estudio los lenguajes, herramientas y tecnologías utilizadas para el desarrollo de la propuesta de solución.

1.2 Marco conceptual

Existen diferentes representaciones de datos, la representación escrita, que es usada cuando la información a representar contiene pocos valores; la representación tabular, cuando los datos se presentan a través de un conjunto de filas y de columnas que responden a un ordenamiento lógico; y la presentación gráfica que proporciona mayor rapidez en la compresión de los datos (2).

A continuación, se hace referencia a algunos tipos de gráficos existentes, teniendo en cuenta su uso para el desarrollo de la propuesta de solución.

Gráfico de barras: representa gráficamente un conjunto de datos o valores en forma de barras rectangulares de longitudes en correspondencia a dichos valores. Se utiliza para comparar las frecuencias o valores de las distintas categorías o grupos. Las barras pueden orientarse vertical u horizontalmente, aunque en sentido horizontal y ordenadas de manera ascendente, facilitan la interpretación de los datos. Además, deben ser más anchas que los espacios entre ellas, las cuales se recomienda no sobrepasen el 40% del ancho de la barra (2).

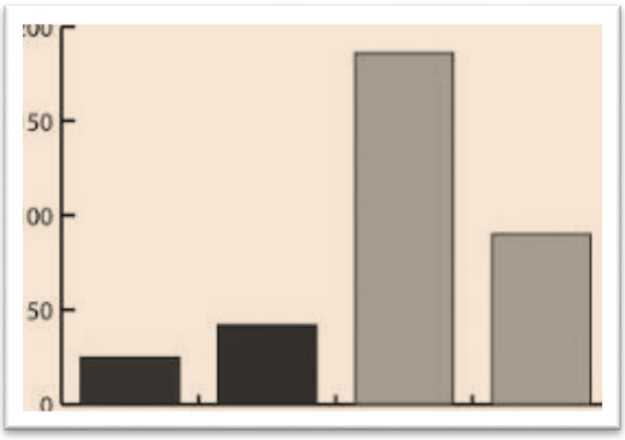


Figura 1: Gráfico de barras

Gráfico de líneas: presenta los datos como una serie de puntos conectados por una línea, su principal uso es para representar los datos de un gran número de grupos o visualización de tendencias en los datos a lo largo del tiempo. Sus parámetros pueden ser modificados para mejorar la interpretación, siempre y cuando se preserven los datos (2).

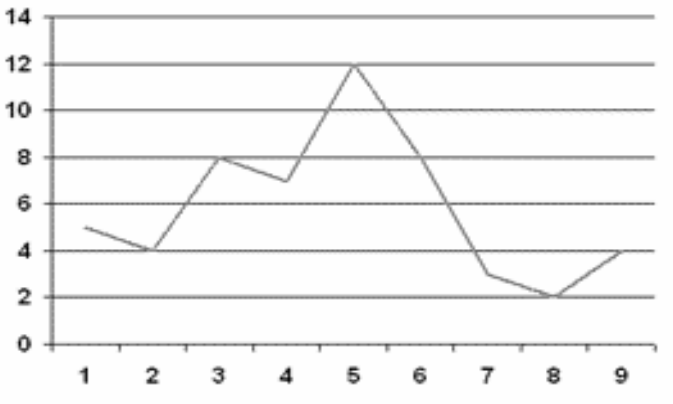


Figura 2: Gráfico de líneas

Gráfico de pastel: muestra los datos cómo áreas rellenas de colores o diseños, es usado para representar los datos de un número de grupos limitados. El uso de este tipo de gráficos no es recomendado por muchos estadísticos, debido a que puede resultar tediosa la comparación entre los diferentes segmentos del pastel. Una alternativa que puede usarse es la de etiquetar los segmentos con sus valores reales o nombres de las categorías que representan (2).

Figura 3: Gráfico de pastel

1.3 Sistemas para la visualización gráfica de información estudiados

Como parte del estudio del estado actual de los sistemas informáticos que permiten la visualización de información mediante gráficas se encontraron referentes a nivel nacional e internacional. Aunque el campo de acción de la investigación se centra solamente en aplicaciones desarrolladas para Android, el estudio se centra en los sistemas que permiten la visualización gráfica de información.

1.3.1 Sistemas Internacionales

1.3.1.1 Mint

Mint se destaca por sus amplias posibilidades, brillante interfaz y hermosas presentaciones gráficas. Esta aplicación permite controlar las finanzas personales sin que nada se te escape, brinda herramientas para ver los gastos semanales o mensuales, armando gráficos de barras o de pastel para mejorar la comprensión de la información. Deja ver las categorías más influyentes en el gasto de tus ingresos y permite llevar un relevamiento día a día sobre cuánto dinero tienes disponible. Esta versión es un cliente de la página web con el mismo nombre, es un maravilloso asistente para los desorganizados con el dinero. Disponible para iOS y Android (4).

1.3.1.2 SQCpack

SQCpack es un software de gráficos y gestión de datos de PQ Systems. El software almacena sus datos de calidad y genera gráficos de rendimiento del proceso. Los gráficos pueden usar datos internos de SQCpack o datos externos de una variedad de fuentes de datos para ayudar a optimizar los procesos y demostrar el rendimiento de calidad. Diseñado con un enfoque en proporcionar una experiencia de usuario simplificada, SQCpack agiliza el proceso de creación y personalización de gráficos para que pueda tomar decisiones comerciales basadas en datos de manera más fácil. StatBoard le permite visualizar rápidamente el estado de un grupo de gráficos. Cuenta con un asistente de creación de gráficos el cual da una guía por los pasos para crear un nuevo gráfico. Agiliza el proceso de creación y edición de gráficos a través de una experiencia de usuario y un flujo de trabajo innovadores. Los gráficos pueden usar datos de varias fuentes de datos externas para proporcionar información de mejora del proceso a partir de datos empresariales que se almacenan fuera de SQCpack. Los gráficos de datos externos obtienen los datos más recientes sin tener que importar los datos para que sus cuadros de SPC[1] estén siempre actualizados automáticamente (6).

[1] Statistical Process Control. es un conjunto de técnicas orientadas a detectar variaciones en un proceso de producción y tomar las acciones correctivas oportunas a fin de conseguir una mejora en la calidad del producto obtenido.

1.3.1.3 SE SPC

El SE SPC (o control estadístico de procesos – CEP) es una solución desarrollada para colectar y analizar datos fácilmente, permitiendo monitorear el desempeño, como alcanzar mejoras sustentables en la calidad y rentabilidad. La solución usa controles gráficos como la principal herramienta para el control estadístico de los procesos. Los gráficos son usados para determinar si un proceso de negocios o de fabricación está en estado de control estadístico o no. También son usados junto a las mediciones del producto para analizar la capacidad del proceso y para medir los esfuerzos de mejora continua de los procesos. Además, múltiples gráficos de CEP pueden ser simultáneamente exhibidos y actualizados en una única ventana en tiempo real (8).

1.3.2 Sistemas Nacionales

1.3.2.1 Olimpia

Es una aplicación desarrollada sobre el framework Symfony. Es multiplataforma. Soporta imágenes, gráficos y sub-reportes, así como varios orígenes de datos. Proporciona a los usuarios, entre otras opciones, agilizar la toma de decisiones, generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos, objetos de este producto. Está compuesto por varias aplicaciones entre las que se encuentran el Visor de reportes, Diseñador de modelos y el Diseñador de reporte. Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos. Además, el entorno de trabajo está estructurado de forma que es difícil guiarse en la creación y generación de reportes.

1.3.3 Resultados del estudio de los sistemas nacionales e internacionales:

En el análisis de estos sistemas se tuvieron en cuenta los siguientes indicadores:

- Que fueran sistemas que visualizaran información mediante gráficas.
- Que fueran aplicaciones para el sistema operativo Android.
- Que obtengan la información a través de servicios.

Como resultado del estudio se concluye que ningún de los sistemas estudiados constituyen una solución al problema de la investigación. En caso de la aplicación Mint, a pesar de que permite visualizar información mediante gráficas y estar disponible para Android, la información no se obtiene a partir del consumo de servicios web. Los sistemas SQCpack, SE SPC y Olimpia permiten la visualización gráfica

de información, pero no están disponibles para Android y no obtienen la información a través de servicios web.

Los elementos expuestos anteriormente evidencian la necesidad de llevar a cabo una solución que brinde respuesta al problema planteado. Sin embargo, el estudio realizado contribuyó a una mejor comprensión de los principales elementos a tener en cuenta.

1.4 Metodología de desarrollo de software

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (3). Respetando las políticas adoptadas por el proyecto SIPAC del centro CEIGE se decidió utilizar como metodología ágil la variación de *Agile Unified Process* (AUP por sus siglas en inglés) para la UCI, AUP-UCI. Esta metodología está orientada de manera genérica ya que para su utilización se debe adaptar a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

1.4.1 Descripción de las Fases

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la siguiente Tabla (3):

Tabla 1. Fases AUP UCI

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP-UCI propone 7 disciplinas las cuales son: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de aceptación y pruebas de liberación. Además, para la disciplina de requisitos la misma propone 4 escenarios, para la presente investigación, se escogió el escenario 3, ya que es el que mejor se adapta a las condiciones de desarrollo del sistema, es el empleado en el proyecto SIPAC del centro CEIGE. Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que ejecutan y controlan, proporcionando objetividad, solidez y su continuidad.

1.5 Lenguajes utilizados

Lenguaje de modelado

Además de lo esencial que resulta una metodología de software como hilo conductor de todo el ciclo de vida del sistema, también es importante tener en cuenta que un proceso de desarrollo de software es necesario contar con algún elemento que describa el aspecto y la conducta del producto, estos elementos son llamados lenguaje de modelado.

Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML por sus siglas en ingles), permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Permite a los creadores de sistema generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas (12).

Este leguaje dispone de reglas para combinar tales elementos y permite la modelación de sistemas con tecnologías orientadas a objetos. Los diagramas son entes importantes de UML, cuya finalidad es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. El modelo gráfico de UML tiene un vocabulario en el que se identifican: elementos, relaciones y diagramas (12).

Lenguaje de programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación (14).

Se utiliza Java, ya que es el lenguaje nativo de Android, por lo que su uso se ha extendido en dispositivos móviles. Puede ser modificado por cualquiera, circunstancia que lo convierte en lo que comúnmente se denomina “código abierto”. Su portabilidad, la simpleza de su estructura y las grandes posibilidades que brinda con su utilización, son características que lo han convertido en un lenguaje ampliamente utilizado por la comunidad de desarrolladores de software.

Java SE JDK 1.8

Es el conjunto de recursos Java, o plataforma, que conforman la denominada Java Standard Edition, fundamental para el desarrollo de aplicaciones Java y requerido por el entorno de desarrollo. Este

conjunto de recursos no es propio de la plataforma Android, pero debe ser instalada en el ordenador para poder desarrollar aplicaciones(16).

1.6 Herramientas y tecnologías utilizadas

1.6.1 Herramienta CASE[2]

Con el objetivo de apoyar y automatizar la metodología de software y el lenguaje de modelado se emplean las herramientas CASE. Estas fueron diseñadas para aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero. Reemplazan al lápiz y al papel por la computadora para transformar la actividad de creación de aplicaciones informáticas en un proceso automatizado.

Esta tecnología contribuye a elevar la calidad en los sistemas. Nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (3).

Como herramienta CASE se utiliza Visual Paradigm for UML 8.0. Una de sus ventajas sobre las demás herramientas CASE es su condición multiplataforma, por lo que tiene la capacidad de ejecutarse sobre diferentes Sistemas Operativos. Permite modelado de sistemas mediante una interfaz agradable y sencilla que permite además la interacción con el usuario. Con esta herramienta se puede modelar casi todo el ciclo de desarrollo de un software.

1.6.2 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (Integrated Development Enterprise, IDE por sus siglas en inglés) es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Normalmente consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La gran mayoría cuenta con un autocompletado inteligente de código.

Android Studio

Es el IDE proporcionado por Google para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece funciones que aumentan la productividad durante la compilación de apps para Android, como las siguientes (21):

[2] CASE: Computer Aided Software Engineering

- Sistema de compilación flexible basado en Gradle[3].
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run, para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub, para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, uso, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.

Se decide usar esta herramienta para la implementación de la propuesta de solución ya que provee una compilación rápida de la app en tiempo real gracias al emulador. La misma permite realizar dicha emulación directamente desde el móvil permitiendo visualizar los cambios realizados.

Android SDK

Android SDK siglas en inglés de Software Development Kit (kit de desarrollo de software) es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, en este caso para Android. Este paquete o kit de desarrollo incluye las API siglas en inglés de *Application Programing Interface* (Interfaz de Programación de Aplicaciones) y herramientas necesarias para desarrollar las aplicaciones, utilizando Java como lenguaje de programación (23).

Gradle 3.3

Es un sistema de compilación basado en JVM (Máquina Virtual de Java, del inglés Java Virtual Machine) a su vez es un plugin[4], lo que facilita su actualización y su exportación de un proyecto a otro. Gradle permite reutilizar código fácilmente, hace sencilla la tarea de configurar y personalizar la compilación, permite la distribución sencilla de código al resto del mundo y gestiona las dependencias de forma potente y cómoda ya que está basado en Maven[5]. Esta herramienta emplea al javac (Compilador de Java o Java Compiler) para programar mediante "Scripting[6]" el funcionamiento de la integración modular de la aplicación y permite compilar la aplicación mientras es desarrollada(25).

[3] Gradle es un sistema de compilación que reúne en un uno solo las mejores prestaciones de otros sistemas de compilación. Está basado en JVM (Java Virtual Machine), lo que significa que puedes escribir tu propio script en java, y que Android Studio lo entenderá y lo usará.
[4] Plugin: aplicación que añade una funcionalidad o una nueva característica al software.
[5] Maven: repositorio de Android.
[6] Scripting: secuencias de comandos.

1.6.3 JSON

(JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript[7]. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (27).

JSON está constituido por dos estructuras:

Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas.

1.7 Conclusiones parciales

La descripción de los principales conceptos posibilitó adquirir una mayor comprensión de los temas relacionados con el objeto de estudio de la presente investigación. El análisis de las aplicaciones descritas permitió abordar en el estado actual de los sistemas que permiten visualizar información a través de gráficas, así como las principales características y funcionalidades de los mismos. A partir de este análisis se evidenció la necesidad de implementar una aplicación para dispositivos móviles que permita visualizar la información sobre la planificación en tiempo real del SIPAC, mediante gráficas. Por otra parte, el análisis de diferentes herramientas y tecnologías permitió alcanzar los conocimientos necesarios para seleccionar las adecuadas para el desarrollo de la solución. Se selecciona como metodología a utilizar AUP-UCI, como entorno de desarrollo integrado Android Studio y como lenguaje de programación Java para la implementación de la propuesta de solución.

[7] JavaScript (JS) es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Capítulo 2: Análisis y diseño de la propuesta de solución

2.1 Introducción

En el presente capítulo se especifica la propuesta de solución a la problemática planteada. Se describe el proceso haciendo uso de la metodología, arquitectura, lenguaje de programación y herramientas seleccionadas. Se especifican los requisitos funcionales y no funcionales, así como mediante sus prototipos de interfaz de usuarios. Además, se realiza el diseño de la arquitectura del sistema.

2.2 Propuesta de solución

Se propone el desarrollo de una aplicación móvil en Android para visualizar información mediante gráficas, dicha información será recopilada a través del consumo de servicios web, con la utilización de una API-REST desde la base de datos del SIPAC 3. Esta información estará en contribución al seguimiento de los procesos de Ejecución y Control de la Planeación Estratégica y Operativa, además de servir como apoyo al proceso de toma de decisiones. En dicha aplicación el usuario podrá visualizar la información correspondiente a través de gráficos desde el móvil independientemente de la opción seleccionada, ya sea: conocer el estado de cumplimiento de los planes y actividades en un instante determinado de tiempo, cantidad de involucrados a un ARC, entre otras funcionalidades.

Figura 4. Propuesta de Solución.

2.3 Modelado del negocio

El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes (3):

1. Casos de Uso del Negocio (CUN).
2. Descripción de Proceso de Negocio (DPN).
3. Modelo Conceptual (MC).

A partir del escenario seleccionado de la metodología se utiliza para el modelado del negocio:

2.3.1 Descripción de Procesos de Negocio

En la tabla 2 se muestra la descripción del proceso Visualizar información donde se describe el mismo para un mayor entendimiento por parte del desarrollador de la presente investigación.

Tabla 2. Descripción del proceso Visualizar información.

Objetivo	Visualizar información para la planificación en tiempo real.
Evento(s) que lo genera(n)	Necesidad de visualizar la información existente.
Pre condiciones	Debe existir información que se desea visualizar.
Responsable	Directivo de la entidad.
Clientes internos	Directivo de la entidad.
Clientes externos	N/A
Entradas	Información.
Flujo de eventos	
Flujo básico	
1. Autenticar.	El usuario debe estar autenticado en la aplicación.
2. Seleccionar opción.	El usuario selecciona la opción que desea visualizar.
Post-condiciones	
1. Se genera la gráfica correspondiente a la opción seleccionada por el usuario.	
Salidas	
1. Se muestra una gráfica con la información correspondiente a la opción seleccionada por el usuario.	

Diagrama de proceso de negocio

El diagrama de proceso de negocio es un método que modela los pasos de un proceso planificado de principio a fin. Un aspecto clave de la gestión de procesos de negocio es que representa visualmente una secuencia detallada de los flujos de información y las actividades necesarias para realizar un proceso. A continuación, se muestra el diagrama del proceso Visualizar información (29).

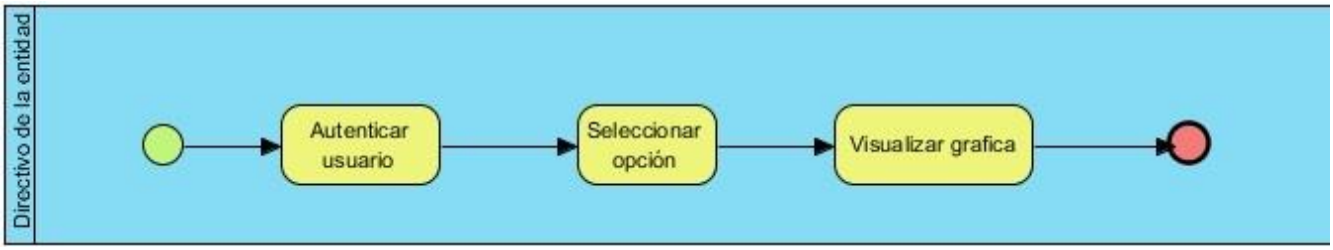


Figura 5.Diagrama del proceso Visualizar información

2.3.2 Modelo Conceptual

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos (31). En este modelo se definen cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema, en este caso describe los conceptos relacionados con el negocio del sistema SIPACgraphic.

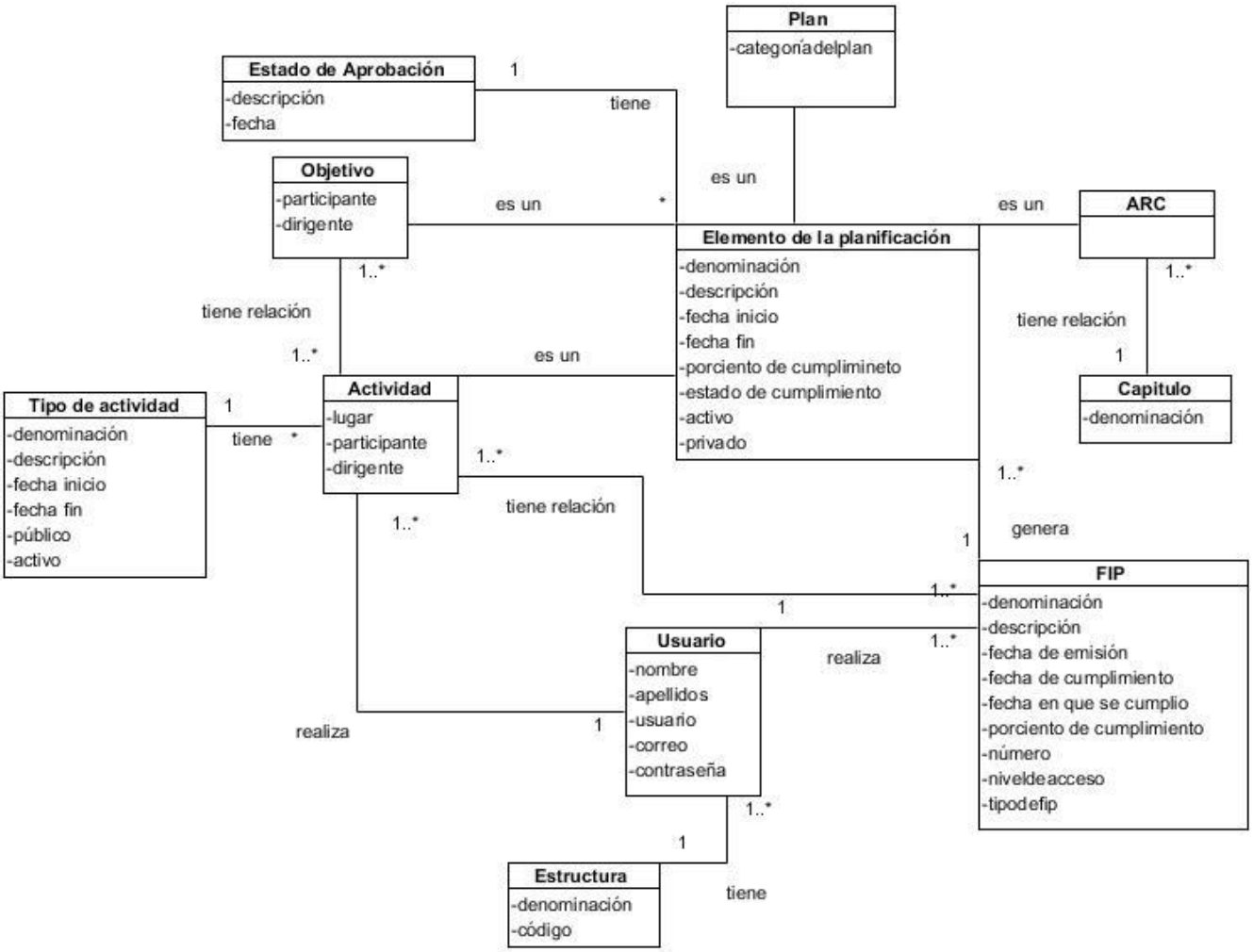


Figura 6: Modelo Conceptual

2.4 Requisitos de software

2.4.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (5).

Utilizando la entrevista como método, para SIPACgraphic se identificaron un total de 10 requisitos funcionales (RF):

RF 1: Autenticar usuario.

RF 2: Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado.

RF 3: Obtener la cantidad de tareas por tipo de actividad asociadas a un usuario en un período determinado.

RF 4: Obtener la cantidad de tareas extraplanes asociadas a un usuario en un período determinado.

RF 5: Obtener la cantidad de tareas cumplidas entre estructura para un período determinado.

RF 6: Mostrar el porciento de cumplimiento de los planes mensuales de un período determinado.

RF 7: Mostrar el estado de cumplimiento de los objetivos.

RF 8: Mostrar la cantidad de actividades relacionadas a un Objetivo.

RF 9: Mostrar la cantidad de Factores que Influyen en el Plan (FIP) para cada usuario involucrado en el proceso de planificación de actividades para un período de tiempo.

RF 10: Mostrar el porciento de las Áreas de Resultados Clave (ARC) asociadas a capítulo.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares(5). A continuación, son especificados los requisitos no funcionales con que debe cumplir el sistema que se propone, agrupados en las siguientes categorías:

Requisitos de usabilidad: Las interfaces gráficas implementadas deben concebirse con un ambiente sencillo y de navegación fácil para el usuario, debido a que la aplicación podrá ser usada por usuarios con conocimientos básicos de dispositivos móviles.

Requisitos de seguridad: El acceso a la aplicación debe estar regido por la autenticación del usuario y estar en correspondencia con permisos y roles definidos para el acceso al SIPAC.

Requisitos de sistema (hardware):

El dispositivo móvil debe soportar tecnología Wi-Fi 802.11 b/g/n.

Se requiere un microprocesador con una velocidad de 650 MHz o superior.

Se requiere un mínimo de 256 Mb de memoria RAM.

Requisitos de sistema (software): Se debe disponer en el dispositivo móvil una versión 4.0 o superior del sistema operativo Android.

2.5 Descripción de Requisitos por Proceso

Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de Usuario (HU) y Descripción de Requisitos por Proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio (3).

Siguiendo la metodología seleccionada para la investigación y el escenario escogido se hace necesario utilizar la Descripción de Requisitos por Proceso para encapsular los requisitos.

2.5.1 Especificación de Requisito < Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado >

Tabla 3: Especificación de RF2 Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado

Precondiciones	El usuario se ha autenticado en el sistema y cuenta con los permisos necesarios para ejecutar esta acción. El usuario selecciona la opción Tareas por estado.
Flujo de eventos	
Flujo básico Tareas por estado	
•	Se muestra la interfaz principal de la aplicación. El usuario accede al menú principal de la aplicación de dos formas: 1. Presionando el botón que aparece en la esquina superior izquierda. 2. Deslizando el dedo de izquierda a derecha desde el borde de la ventana.
1.	Selecciona en el menú la opción Actividades y dentro de éste selecciona la opción Tareas por estado.
2.	Se muestra la interfaz Tareas por estado para que el usuario inserte los datos: Fecha inicio y Fecha fin, usuario y selecciona el tipo de gráfica en el que desee visualizar la información.
3.	Se validan los datos introducidos (Ver validación 1).
4.	Si los datos introducidos son correctos se muestra la información en la gráfica seleccionada.
5.	Concluye el requisito.
Pos-condiciones	
•	Se muestra la información en la gráfica seleccionada por el usuario.
Flujos alternativos	
Flujo alternativo 4.a Campos vacíos	
1.	No se introduce la fecha de inicio o la fecha de fin y selecciona uno de los tipos de gráficas.
2.	El sistema muestra un mensaje de error “Debe seleccionar la fecha de inicio y fin”.
3.	Volver al paso 3 del flujo básico.
Pos-condiciones	
1.	N/A
Flujo alternativo 4.b Datos incorrectos-	
1.	Se introduce la fecha de inicio superior a la fecha de fin.
2.	La fecha de inicio no puede ser superior a la fecha final.
Pos-condiciones	
1.	No se muestra la información.
Validaciones	

1.	El sistema valida que la fecha de inicio no sea superior a la fecha de fin.	
	Fecha inicio	**Visibles en la interfaz**: Fecha inicio
	Fecha fin	**Visibles en la interfaz**: Fecha fin
	Usuario	**Visibles en la interfaz**: Usuario
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo elemental de interfaz gráfica de usuario < Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado >

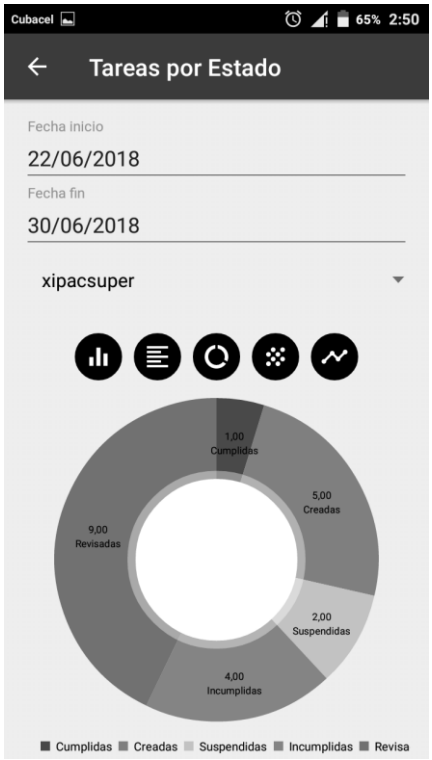


Figura 7: Prototipo del RF2 Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado

2.5.2 Especificación de Requisito < Obtener la cantidad de tareas por tipo de actividad asociadas a un usuario en un período determinado >

Tabla 4: Especificación de RF3 Obtener la cantidad de tareas por tipo de actividad asociadas a un usuario en un período determinado

Precondiciones	El usuario se ha autenticado en el sistema y cuenta con los permisos necesarios para ejecutar esta acción. El usuario selecciona la opción Tareas por actividad.
Flujo de eventos	
Flujo básico Tareas por tipo de actividad	
•	Se muestra la interfaz principal de la aplicación. El usuario accede al menú principal de la aplicación de dos formas: 1. Presionando el botón que aparece en la esquina superior izquierda. 2. Deslizando el dedo de izquierda a derecha desde el borde de la ventana.
2.	Selecciona en el menú la opción Actividades y dentro de éste selecciona la opción Tareas por actividad.
3.	Se muestra la interfaz Tareas por actividad para que el usuario inserte los datos: Fecha inicio y Fecha fin, usuario y selecciona el tipo de gráfica en el que desee visualizar la información.
4.	Se validan los datos introducidos (Ver validación 1).
5.	Si los datos introducidos son correctos se muestra la información en la gráfica seleccionada.
6.	Concluye el requisito.
Pos-condiciones	
•	Se muestra la información en la gráfica seleccionada por el usuario.
Flujos alternativos	
Flujo alternativo 4.a Campos vacíos	
2.	No se introduce la fecha de inicio o la fecha de fin y selecciona uno de los tipos de gráficas.
2.	El sistema muestra un mensaje de error “Debe seleccionar la fecha de inicio y fin”.
3.	Volver al paso 3 del flujo básico.
Pos-condiciones	
1.	N/A
Flujo alternativo 4.b Datos incorrectos-	
1.	Se introduce la fecha de inicio superior a la fecha de fin.
2.	La fecha de inicio no puede ser superior a la fecha final.
Pos-condiciones	
1.	No se muestra la información.
Validaciones	

7.	El sistema valida que la fecha de inicio no sea superior a la fecha de fin.	
	Fecha inicio	**Visibles en la interfaz**: Fecha inicio
	Fecha fin	**Visibles en la interfaz**: Fecha fin
	Usuario	**Visibles en la interfaz**: Usuario
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Prototipo elemental de interfaz gráfica de usuario < Obtener la cantidad de tareas por tipo de actividad asociadas a un usuario en un período determinado >

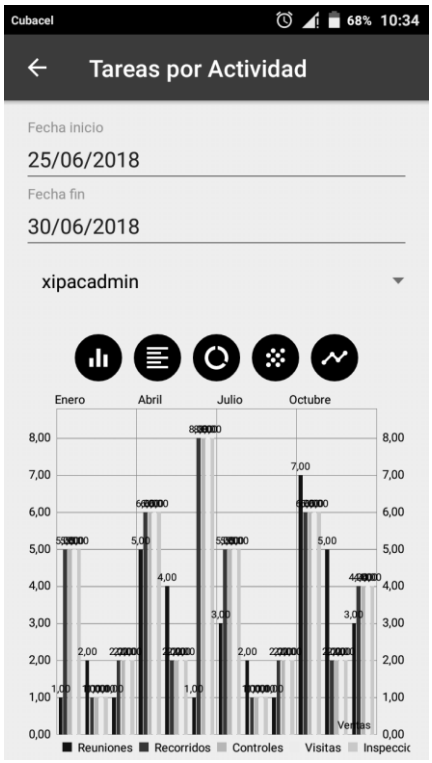


Figura 8: Prototipo del RF3 Obtener la cantidad de tareas por tipo de actividad asociadas a un usuario en un período determinado

2.6. Arquitectura de software

La arquitectura de software de un sistema o programa de computación es la estructura o estructuras del sistema, que comprenden elementos de software, las propiedades externamente visibles de estos elementos, y las relaciones entre ellos (7).

Arquitectura Cliente/Servidor

Según Microsoft (9), el estilo arquitectónico Cliente/Servidor describe sistemas distribuidos que implican un sistema independiente de clientes y servidores, y una red que los conecta. La forma más simple de sistema Cliente/Servidor consiste en una aplicación servidor a la que se accede directamente por varios clientes, a la cual se refiere como un estilo arquitectónico de 2 niveles.

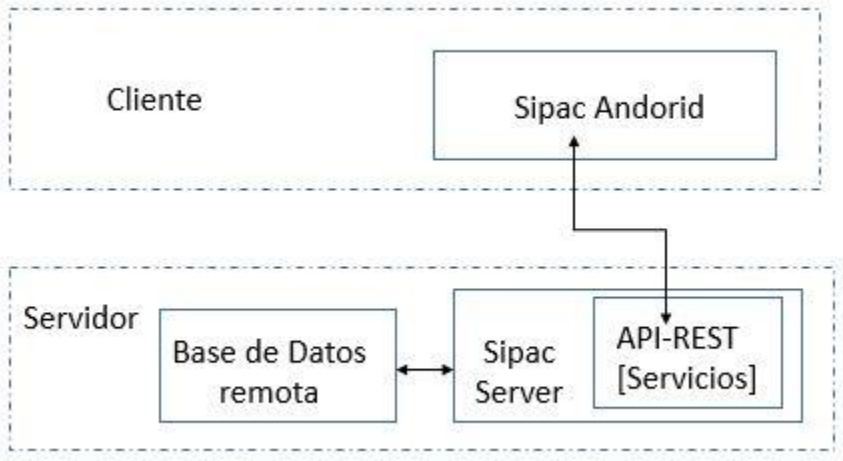


Figura 9. Arquitectura Cliente/Servidor desglosada en sus dos niveles.

2.7 Patrones de diseño utilizados

Cuando existen situaciones o problemas comunes y ampliamente observados, surgen soluciones que pueden aplicarse, basándose en una resolución anterior. Durante el diseño Orientado a Objetos es frecuente encontrarse repetidamente con estos tipos de problemas, es por ello que para analizar, compartir y documentar el conocimiento sobre estos problemas se han desarrollado los patrones de diseño.

Existen clasificaciones para estos patrones entre los que se encuentran los Patrones Generales de Software para Asignar Responsabilidades (GRASP por sus siglas en inglés) y los Patrones del Grupo de los Cuatro (GOF). A continuación, se presentan los patrones de diseño que son utilizados para el desarrollo de la solución.

Patrones GRASP

(Patrones de Software de Asignación de Responsabilidad General, del inglés General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. A continuación se explican los principales patrones utilizados (11). Patrones GRASP utilizados (11):

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). En una clase, que contiene toda la información necesaria para realizar la labor es donde se debe implementar el método. Un ejemplo de este patrón se evidencia en la clase Tareas.java la cual posee la información asociada a cada tipo de Tareas, por tanto, es la encargada de brindar información de cada Tarea mediante las funcionalidades getAllTareas(), getTareasPorActividad(), getTareasPorEstado(), getTareasCumplidas(), entre otras.

Figura 10: Evidencia del patrón Experto

Creador: Define quien es el encargado (o quien debería tener la responsabilidad) de crear un determinado objeto. Asigna la responsabilidad a una clase de crear objetos, práctica muy frecuente en los sistemas orientados a objetos. Entre sus beneficios se encuentra el brindar soporte a un bajo acoplamiento lo que trae como consecuencia una menor dependencia respecto al mantenimiento y la reutilización en mayor medida. A continuación se muestra un ejemplo de la aplicación de este patrón en la clase Gráfica, perteneciente al paquete cu.uci.sipac. La clase Grafica.java es la única responsable de la creación de objetos de la clase Tareas.

Figura 11: Evidencia del patrón Creador

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. El patrón propone el diseño de clases más independientes, lo que reduce el impacto del cambio y facilita la reutilización en otros sistemas. Por supuesto es casi imposible que no existan dependencias entre algunas clases de un proyecto.

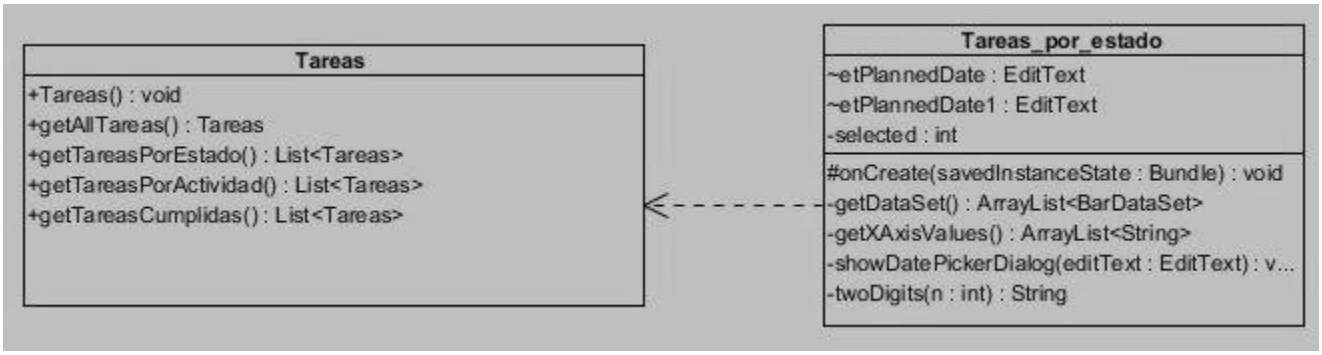


Figura 12: Evidencia del patrón Bajo Acoplamiento

Alta Cohesión: Asigna las responsabilidades en función de mantener la complejidad dentro de los límites posibles. La cohesión es una medida que refleja cuán enfocadas y relacionadas se encuentran las responsabilidades en una clase. Las clases que presentan una alta cohesión se caracterizan por poseer responsabilidades estrechamente relacionadas que no hagan un enorme trabajo.

- Este patrón se evidencia en todas las clases teniendo en cuenta que todas almacenan información coherente y que están relacionadas con cada uno de los objetos que representa cada clase.

Patrones GOF

Los patrones del Grupo de los Cuatro del inglés The Gang of Four proponen soluciones a problemas concretos, ya que no son teorías genéricas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). Se dividen en tres categorías: patrones de creación, patrones estructurales y patrones de comportamiento. A continuación, se describe el patrón utilizado en la solución (13):

Adaptador (del inglés Adapter): convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían tener interfaces incompatibles (13).
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.

El patrón de diseño Maestro/Esclavo es usado cuando se tienen dos o más procesos que necesitan ejecutarse simultánea y continuamente, pero a diferentes velocidades. National Instruments lo describe como un patrón de diseño que consiste en múltiples ciclos paralelos. Cada ciclo puede ejecutar tareas a velocidades distintas. De estos, uno actúa como el maestro y los otros como esclavos. El ciclo maestro controla todos los ciclos esclavos y se comunica con ellos utilizando arquitecturas de mensajería (15).

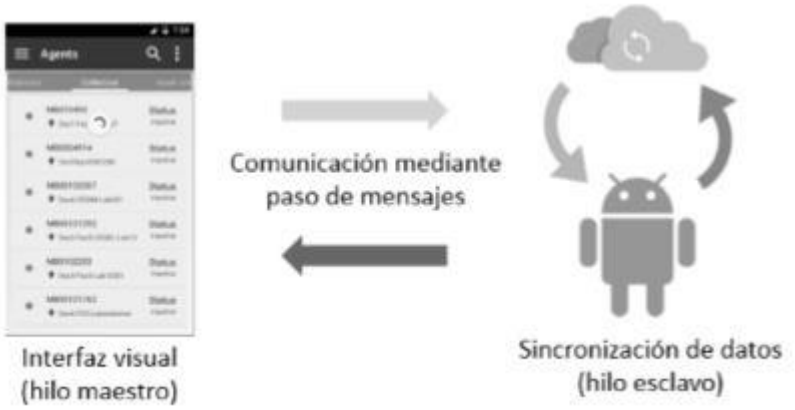


Figura 13: Patrón Maestro/Esclavo

2.8 Conclusiones parciales

En este capítulo se definieron los elementos referentes al análisis y diseño de la propuesta de solución. Se describió la aplicación móvil en Android con la visualización de información de la planificación en tiempo real para el SIPAC 3. El modelado del negocio se realizó haciendo uso de la descripción de proceso del negocio y el modelo conceptual, mientras que el modelado del sistema se desarrolló mediante la descripción de requisitos por proceso, de acuerdo al escenario seleccionado de la metodología AUP-UCI. Se definió como arquitectura del sistema cliente-servidor y como patrones de diseño utilizados, creador, experto, alta cohesión y bajo acoplamiento dentro de los GRASP y el adaptador, singleton por parte de los GoF.

Capítulo 3: Implementación y prueba de la solución

3.1 Introducción

En el presente capítulo se describen los estándares de codificación empleados en la implementación de la propuesta de solución. Se muestran los elementos necesarios para la utilización de la aplicación, a través del diagrama de despliegue. Se describe la estrategia de prueba definida para validar la aplicación y se realiza un análisis de los resultados obtenidos luego de aplicadas las mismas.

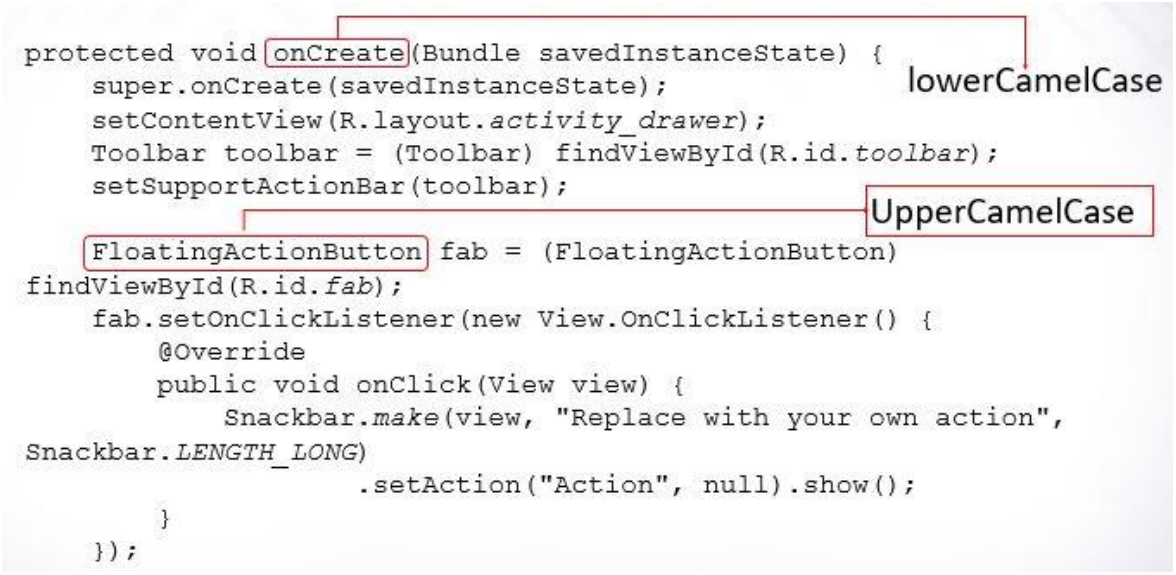
3.2 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea posible darle soporte con mayor facilidad (17).

Para la realización de SIPACgraphic se utilizó:

Java CamelCase

La notación CamelCase es la práctica de escribir palabras compuestas, o frases, tales que cada palabra o abreviatura comienza con una letra mayúscula y omite guiones. CamelCase puede comenzar con una letra mayúscula, estilo que es también conocido como UpperCamelCase y que es usado sobre todo para nombrar clases, o con una letra minúscula para nombrar variables y métodos. Es un estilo recomendado por varias comunidades y empresas desarrolladoras de Java, tales como Oracle Corporation; además, este estilo de codificación está integrado directamente en el sistema de auto completamiento de código de Android Studio, lo cual acelera el proceso de escritura del código manteniendo su coherencia y legibilidad (37).




```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_drawer);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
        }
    });
```

Figura 14: Estándar de codificación CamelCase

Convención de Nombres

La convención de nombres es un conjunto de normas y reglas para la escritura de nombres, código fuente, identificadores y comentarios dentro de la programación, que facilitan y hacen más comprensible su lectura (18).

Clases

- Las clases representan “cosas” y no “acciones”, por tal motivo evitar verbos como nombre de clase.
- El nombre de la clase debe estar en singular, salvo que la clase represente multiplicidad de cosas.
- Las Nombres de las clases deberían ser Sustantivos: ejemplo *carro, hombre, tienda, país, empleado, proveedor.*
- Cada clase debe tener un bloque de documentación según la norma del lenguaje.
- La inicial en mayúscula ya sea simple o compuesto el nombre de la clase.

Figura 15: Asignación de nombres a las clases

Métodos

Los nombres de los métodos deberían ser un verbo, dado que describe una acción; ejemplo *remover(), enviar(), cargar().*
Los Métodos dentro de las clases siempre debe declarar su visibilidad tales como *privadas, protegidas, públicas.*
La primera letra de la primera palabra en minúsculas, el resto de las palabras empiezan por mayúsculas.

```
private void prepareListData () {...}

public void onBackPressed () {...}
```

Variables

- Evitar variables que sean de un solo carácter, Los nombres comunes para las variables temporales son i, j, k, m, y n para los números enteros; c, d, y e para los caracteres.
- Nombres de variables sólo pueden contener caracteres alfanuméricos.
- Nombres de variables deben ser camelCase.
- Deben comenzar por minúscula y no contener caracteres extraños como tildes o guiones bajos.

```
public int contador = 1;

public int posicion = 0;
```

3.3 Diagrama de despliegue

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución en las instancias de los nodos de proceso (20). Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo.
El destino de despliegue está generalmente representado por un nodo que es, o bien de los dispositivos de hardware o bien de algún entorno de ejecución de software. Los nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria.

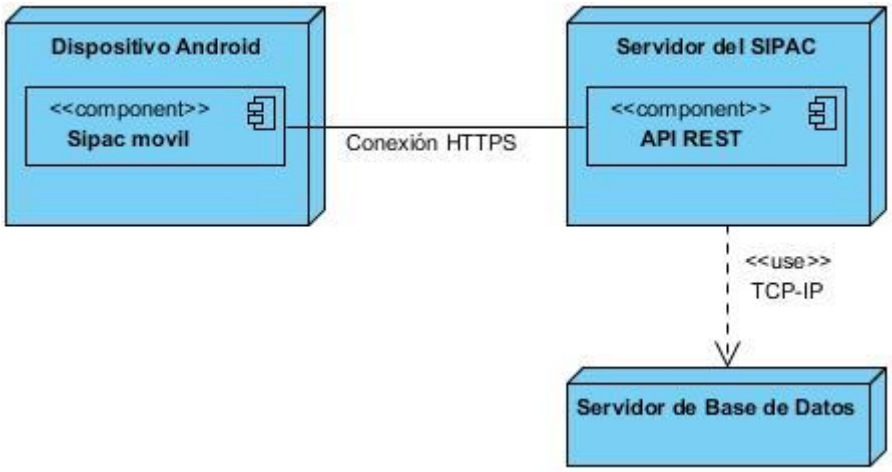


Figura 16: Diagrama de despliegue para la propuesta de solución.

Descripción de los nodos del diagrama de despliegue

Dispositivo Android: Dispositivo Android, que debe tener instalada la versión 4.0 o superior del sistema operativo Android. En este nodo es donde estará la aplicación, su función es la ejecución de la aplicación y la interacción con el sistema según las necesidades del usuario.

Servidor del SIPAC: Corresponde al servidor de aplicación, que contiene el API REST con los servicios que se utilizarán para acceder a los datos del sistema.

Servidor de base de datos: Representa el servidor de base de datos del sistema SIPAC.

3.4 Pruebas

Luego de la implementación de la solución, se realizan un conjunto de actividades para verificar la calidad del producto y su cumplimiento con los requisitos definidos. El objetivo de esta fase es detectar y solucionar los errores que presenta el componente desarrollado, y perfeccionar la solución implementada.

Las pruebas son "…un componente importante de calidad del software..., proceso de ejecutar un programa para detectar errores" (22).

3.4.1 Métodos de Prueba

Los métodos de pruebas definen estrategias para descubrir fallos en el sistema. Como métodos de prueba, Pressman en su 7ma edición, propone los siguientes:

Pruebas de caja blanca

Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: garanticen que se ejercite por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales, y que se ejerciten las estructuras internas de datos para asegurar su validez (24).

Estas pruebas se realizan al código fuente para asegurar que la operación interna se ajuste a las especificaciones.

Pruebas de caja negra

Las pruebas de caja negra también conocidas como pruebas funcionales o pruebas de entrada y salida, son las que se ejecutan sobre la interfaz del software. Mediante el uso de estas se examinan todas las funcionalidades. Estas tienen poca relación con el comportamiento interno del software (24). Estas pruebas permiten demostrar que las funciones del sistema sean operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.

3.4.2 Estrategia de prueba

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar estos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes (24).

Roger Pressman en la 6ta edición del libro “Ingeniería de software. Un enfoque práctico” propone cuatro niveles de prueba:

Pruebas unitarias: son pruebas de caja blanca que se realizan con el objetivo de detectar errores de implementación en el componente desarrollado. Además, se identifican errores de entrada o salida de datos.

Pruebas de integración: verifican que cada componente desarrollado no presente errores cuando se integre con los demás.

Pruebas de validación: son pruebas de caja negra que se enfocan en la satisfacción de las necesidades del cliente, verificado las acciones que el usuario realiza en el sistema y la correcta entrada y salida de datos.

Pruebas del sistema: son pruebas que confirman el correcto funcionamiento de las funciones desarrollas.

Para la validación de la propuesta de solución de la presente investigación, se definió una estrategia de prueba que incluye pruebas de unidad y pruebas de validación, en correspondencia a dos de los cuatro niveles antes mencionados. Además, está estrategia se diseñó teniendo en cuenta las disciplinas de Pruebas internas y Pruebas de Aceptación de la metodología AUP-UCI, seleccionada para guiar el desarrollo de la aplicación. Para las pruebas internas se empleó el método de Caja blanca a través de la técnica del camino básico, mientras que para las pruebas de aceptación se utilizó el método de Caja negra con las técnicas de partición de

equivalencia y gráfico de prueba. Además, se empleó la técnica de IADOV para evaluar la satisfacción de los usuarios con la aplicación implementada.

3.4.3 Pruebas unitarias

Las pruebas unitarias se aplican a un componente del software. Podemos considerar como componente (elemento indivisible) a una función, una clase, una librería. Estas pruebas las ejecuta el desarrollador, cada vez que va probando fragmentos de código o scripts para ver si todo funciona como se desea. Estas pruebas son muy técnicas. Por ejemplo, probar una consulta, probar que un fragmento de código envíe a imprimir un documento, probar que una función devuelva un flag (26).

Las pruebas unitarias fueron ejecutadas sistemáticamente, cada vez que se terminaba de implementar cada una de las iteraciones. Dichas pruebas fueron desarrolladas utilizando la herramienta Android JUnit. Esta herramienta permite probar componentes específicos mediante clases de casos de prueba. Estas clases proporcionan métodos auxiliares para la creación de objetos de imitación y métodos que ayudan a controlar el ciclo de vida de una aplicación. A continuación, se muestran los resultados de las pruebas realizadas a las funcionalidades en cada una de las iteraciones:

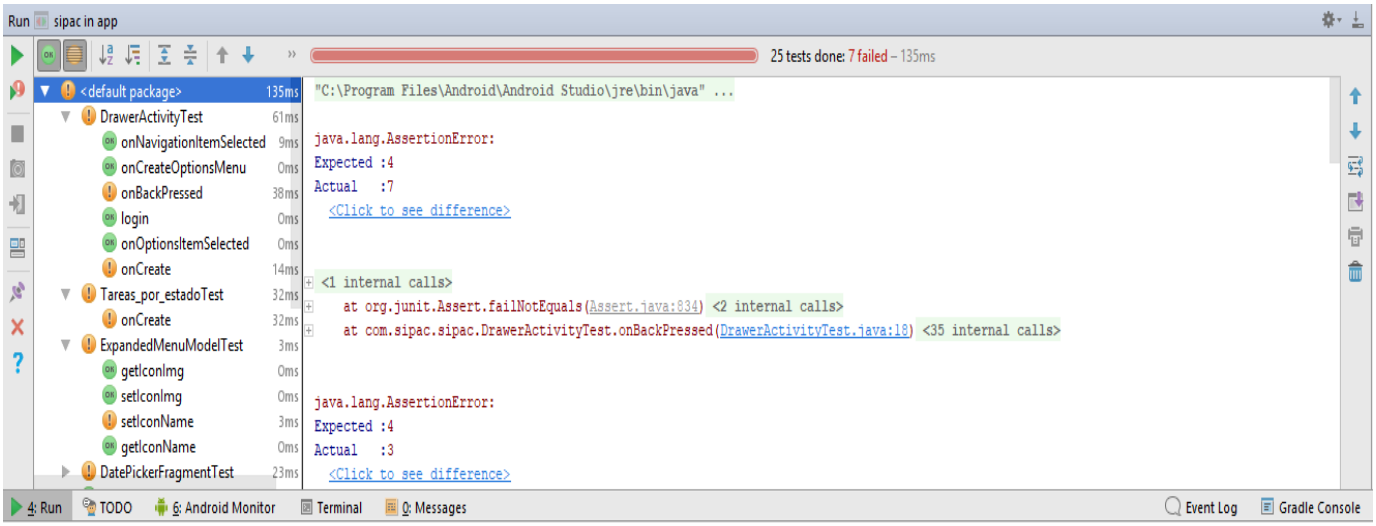


Figura 17:Resultados de las pruebas unitarias en la iteración 1.

Figura 18:Resultados de las pruebas unitarias en la iteración 2.

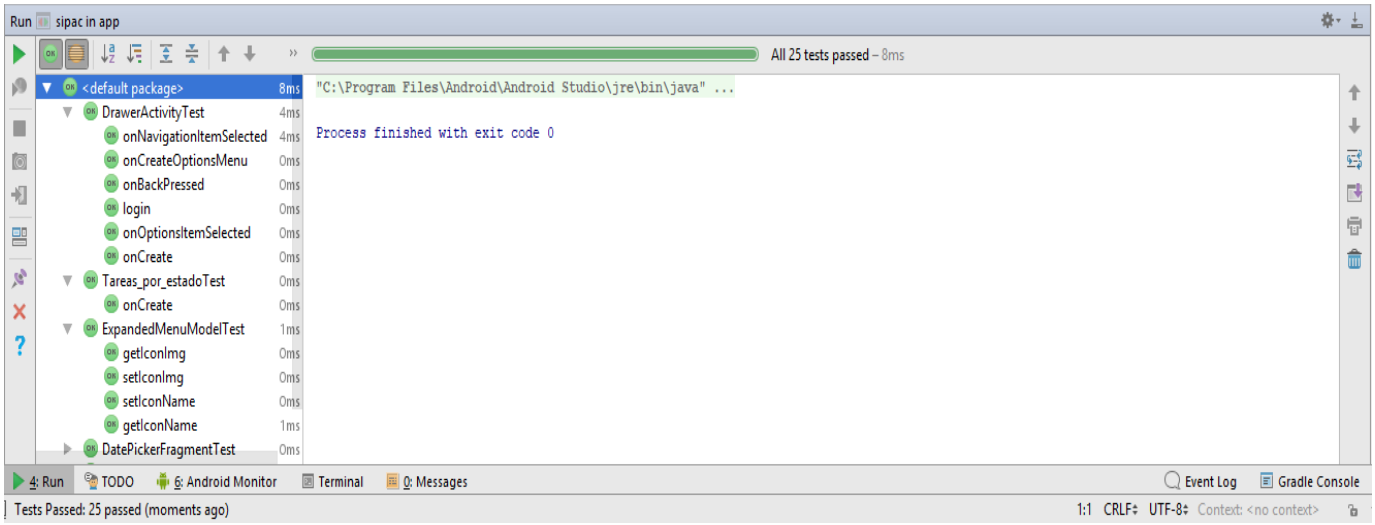


Figura 19:Resultados de las pruebas unitarias en la iteración 3.

Para realizar la prueba de caja blanca se utiliza la técnica **camino básico** para obtener una medida de la complejidad de un algoritmo y un conjunto básico[8] de caminos de ejecución de este, utilizados luego para obtener los casos de prueba. La técnica de Camino Básico se realiza calculando la complejidad ciclomática del algoritmo o fragmento de código a analizar. En la siguiente figura se muestra la funcionalidad onClick encargada de validar los campos fecha, donde:

- La fecha inicio y la fecha fin no pueden ser campos vacios.
- La fecha inicio no puede ser superior a la fecha final.

[8] Conjunto básico: es el conjunto de caminos independientes.

```
public void onClick(View view) {  (1)
     (2) if (etPlannedDate.getText().toString().trim().isEmpty() || etPlannedDate1.getText().toString().trim().isEmpty()) {
     (3)     Toast.makeText(Tareas_Cumplidas.this, "Debe seleccionar la fecha de inicio y fin", Toast.LENGTH_SHORT).show();
     (4) } else {
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            Date strDate = null;
     (5)    try {
                strDate = sdf.parse(etPlannedDate.getText().toString());
     (6)    } catch (ParseException e) {
                e.printStackTrace();
            }
     (7)    Date strDate1 = null;
     (8)    try {
                strDate1 = sdf.parse(etPlannedDate1.getText().toString());
     (9)    } catch (ParseException e) {
                e.printStackTrace();
            }
     (10)   if (strDate1.getTime() < strDate.getTime()) {
     (11)       Toast.makeText(Tareas_Cumplidas.this, "La fecha de inicio no puede ser superior a la final", Toast.LENGTH_SHORT).show();
     (12)   } else {

                chart1.setVisibility(View.GONE);
                pieChart.setVisibility(View.GONE);
                scatterChart.setVisibility(View.GONE);
                lineChart.setVisibility(View.GONE);
                chart.setVisibility(View.VISIBLE);
                chart.setDrawGridBackground(false);
                BarData data = new BarData(getXAxisValues(), getDataSet1());
                chart.setData(data);
                chart.setDescription("Ventas");
                chart.animateXY(1000, 1000);
                chart.invalidate();
            }
        }
    }
});
```

Figura 20: Método onClick

Para el cálculo de la complejidad ciclomática se representa el grafo del flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la siguien figura.

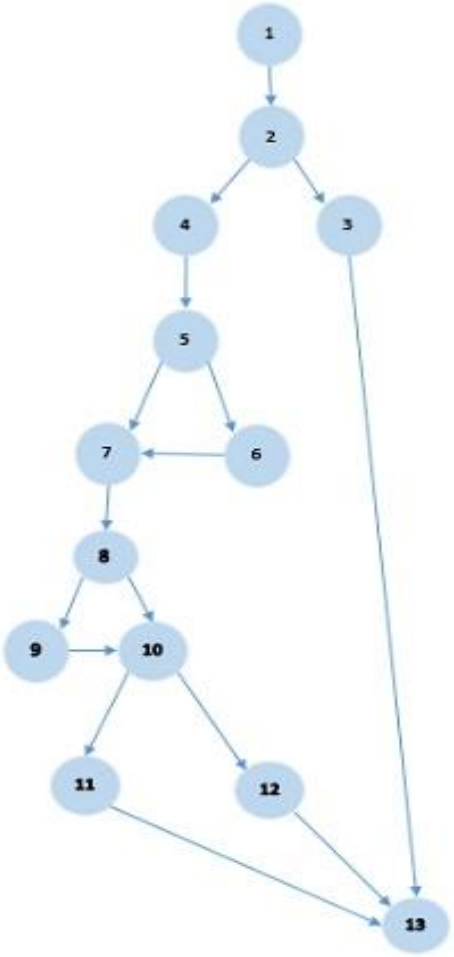


Figura 21: Grafo del método onClick

Una vez construido el grafo de flujo asociado al algoritmo anterior, se calcula la complejidad, el cálculo es necesario efectuarlo mediante tres fórmulas utilizando el mismo grafo en cada caso.

- **V (G) = (A - N) + 2**

Donde **A** es la cantidad total de aristas y **N** la cantidad total de nodos.

Resultado: V (G) = (16-13) + 2

V (G) = 5

- **V (G) = P + 1**

Donde **P** es la cantidad de nodos predicados[9]

Resultado: V (G) = 4 + 1

V (G) = 5

- **V (G) = R**

Donde **R** es la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado: V (G) = 5

[9] Nodos predicados: nodos de los cuales parten dos o más aristas.

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática es de 5, lo que significa que existen 5 posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de prueba para el procedimiento tratado. Seguidamente se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico #1: 1, 2, 3, 13

Camino básico #2: 1,2,4,5,6,7,8, 10,12,13

Camino básico #3: 1,2,4,5,6,7,8,10,11,13

Camino básico #4: 1,2,4,5,7,8,10,12,13

Camino básico #5: 1,2,4,5,7,8,9,10,11,13

Luego de obtenidos los caminos básicos, se realizan los casos de prueba para cada uno de ellos. En la siguiente tabla se muestra el caso de prueba generado para uno de los caminos de ejecución, específicamente para el camino 1, 2, 3, 13.

Camino básico # 1: 1, 2, 3, 13	
Descripción	A partir de la entrada de los datos incorrectos se muestra un mensaje de error.

Condición de ejecución	if (etPlannedDate.getText().toString().trim().isEmpty() || etPlannedDate1.getText().toString().trim().isEmpty()) {
	Toast.makeText(Tareas_Cumplidas.this, "Debe seleccionar la fecha de inicio y fin",
	Toast.LENGTH_SHORT).show();
	}

Entrada	Fecha inicio: ¨ ¨, Fecha fin: (22/06/2018)
Resultado esperado	Se muestra el mensaje: Debe seleccionar la fecha de inicio y fin.

Figura 22: Caso de prueba del método Caja Blanca

Una vez ejecutado el método de Camino básico a las funcionalidades más complejas en el desarrollo del sistema, se pudo comprobar que el flujo de trabajo de las funcionalidades es correcto, pues se probó que cada sentencia es ejecutada al menos una vez con los parámetros de entrada y los resultados esperados, que se ejerciten todas las decisiones lógicas en las vertientes verdaderas y falsas, así como las estructuras internas de datos para asegurar su validez.

3.4.4 Pruebas de validación

Las pruebas de validación se realizaron a través de casos de prueba, los cuales son …un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema (28). Las técnicas utilizadas para comprobar el funcionamiento del sistema fueron **partición equivalente** y **gráfico de prueba**.

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba (30). Esta técnica se utilizó para representar los diferentes estados posibles en cada condición de entrada, calificando cada estado en válidos o inválidos.

Otra de las técnicas empleadas en el desarrollo de las pruebas de caja negra es el gráfico de prueba, basada en "una colección de nodos que representan objetos, enlaces que representan la relación entre objetos, pesos de nodo que describen las propiedades de un nodo (como un valor de datos o un comportamiento de estado específico) y pesos de enlace que describen algunas características de un enlace" (24). Su utilización permite validar funcionalidades que no requieren entrada de datos.

A continuación, se muestra los casos de prueba realizados para verificar el cumplimiento de los requisitos funcionales, utilizando las técnicas antes mencionadas.

Tabla 5: Caso de prueba de partición equivalente del RF1 Autenticar usuario

Escenario	Descripción	Usuario	Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Introducir correctamente un usuario y una contraseña	El usuario introduce correctamente el nombre de usuario y la contraseña.	V	V	Se accede exitosamente a la aplicación.	- Introducir un usuario correctamente. - Introducir una contraseña correctamente. - Seleccionar la opción Iniciar.
		xipacsuper	explorer123		
EC 1.2 Introducir incorrectamente un usuario o una contraseña	El usuario introduce incorrectamente el nombre de usuario o la contraseña.	I	V	Muestra el mensaje ¨Credenciales incorrectas¨.	- Introducir un usuario incorrectamente. - Introducir una contraseña correctamente. - Seleccionar la opción Iniciar.
		xipacsuper123	explorer123		
		V	I		
		xipacsuper	explorer12345		

EC 1.3 No introducir un usuario o la contraseña.	El usuario no introduce el nombre de usuario o una contraseña.	V	I	Muestra el mensaje ¨No pueden haber campos vacíos¨.	- Introducir un usuario correctamente. - No introducir una contraseña. - Seleccionar la opción Iniciar.
		xipacsuper	N/A		
		I	V		
		N/A	explorer123		

Tabla 6: Caso de prueba de partición equivalente del RF2 Obtener la cantidad de tareas por estado asociadas a un usuario en un período determinado

Escenario	Descripción	Fecha Inicio	Fecha Fin	Usuario	Respuesta del sistema	Flujo central
EC 2.1 Seleccionar la opción Actividades, Seleccionar la opción Tareas por estado, haber seleccionado la fecha inicio menor que la fecha fin y un usuario.	El usuario selecciona la fecha inicio, la fecha fin y un usuario, donde la fecha inicio es menor que la fecha fin. El usuario selecciona el tipo de gráfico.	V	V	V	Se muestra un gráfico con la cantidad de tareas por estado asociadas a un usuario en un período determinado.	- Seleccionar la opción Actividades. - Seleccionar la opción Tareas por estado. - Seleccionar una fecha inicio, una fecha fin y un usuario. - Seleccionar el tipo de gráfico.
		13/06/2018	22/06/2018	xipacsuper		
EC 2.2 Seleccionar la opción Actividades, Seleccionar la opción Tareas por estado,	El usuario selecciona la fecha inicio, la fecha fin, donde la fecha inicio es menor que la fecha fin, y	V	V	I	Muestra el mensaje ¨Debe seleccionar un usuario¨.	- Seleccionar la opción Actividades. - Seleccionar la opción Tareas por estado.

haber seleccionado la fecha inicio menor que la fecha fin y no haber seleccionado un usuario.	no selecciona un usuario. El usuario selecciona el tipo de gráfico.	13/06/2018	22/06/2018	N/A		- Seleccionar una fecha inicio, una fecha fin y no selecciona un usuario. - Seleccionar el tipo de gráfico.
EC 2.3 Seleccionar la opción Actividades, Seleccionar la opción Tareas por estado, haber seleccionado la fecha inicio mayor que la fecha fin y un usuario.	El usuario selecciona la fecha inicio, la fecha fin y un usuario, donde la fecha inicio es mayor que la fecha fin. El usuario selecciona el tipo de gráfico.	I	I	V	Muestra el mensaje ¨La fecha de inicio no puede ser superior a la final¨.	- Seleccionar la opción Actividades. - Seleccionar la opción Tareas por estado. - Seleccionar una fecha inicio mayor que la fecha fin y selecciona un usuario. - Seleccionar el tipo de gráfico.
		22/06/2018	13/06/2018	xipacsuper		
EC 2.4 Seleccionar la opción Actividades, Seleccionar la opción Tareas por	El usuario no selecciona la fecha inicio o la fecha fin, y selecciona un usuario.	I	V	V	Muestra el mensaje ¨Debe seleccionar la fecha de inicio y fin¨.	- Seleccionar la opción Actividades. - Seleccionar la opción

estado, no haber seleccionado la fecha inicio o la fecha fin , y seleccionar un usuario.	El usuario selecciona el tipo de gráfico.	N/A	13/06/2018	xipacsuper		Tareas por estado. - No seleccionar una fecha inicio o una fecha fin, y selecciona un usuario. - Seleccionar el tipo de gráfico.

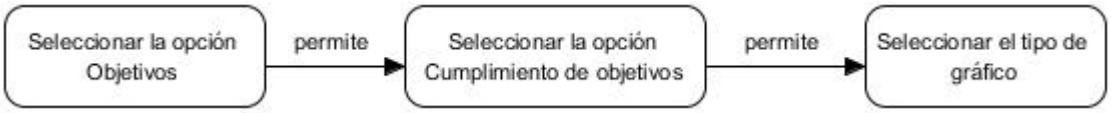


Figura 23: Gráfico correspondiente al caso de prueba del RF7 Mostrar el estado de cumplimiento de los objetivos

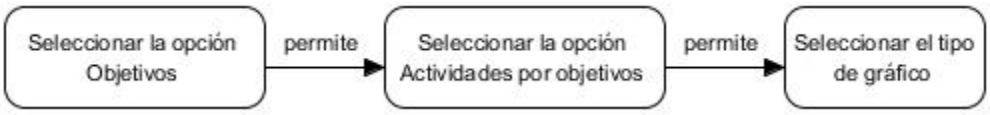


Figura 24: Gráfico correspondiente al caso de prueba del RF8 Mostrar la cantidad de actividades relacionadas a un Objetivo

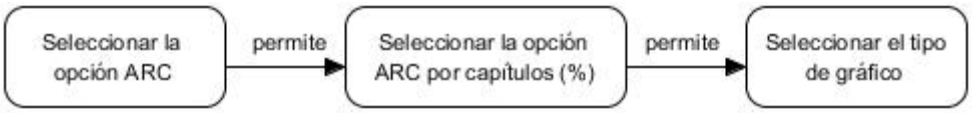


Figura 25: Gráfico correspondiente al caso de prueba del RF10 Mostrar el porciento de las Áreas de Resultados Clave (ARC) asociadas a capítulo

3.4.5 Pruebas de aceptación

Son las únicas pruebas que son realizadas por los usuarios expertos, todas las anteriores las lleva a cabo el equipo de desarrollo. Consiste en comprobar si el producto está listo para ser implantado para el uso operativo

en el entorno del usuario. Podemos distinguir entre dos tipos de pruebas; en ambas existe retroalimentación por parte del usuario experto (26):

- **Pruebas alfa:** las realiza el usuario en presencia de personal de desarrollo del proyecto haciendo uso de una máquina preparada para las pruebas.
- **Pruebas beta:** las realiza el usuario después de que el equipo de desarrollo les entregue una versión casi definitiva del producto.

A continuación, se muestra el gráfico de las NC detectadas por el cliente:

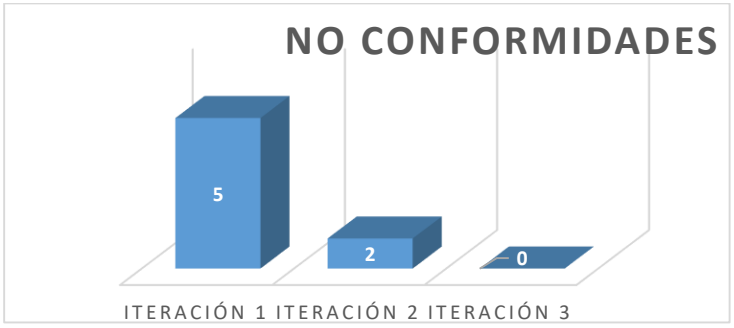


Figura 26:Iteraciones de pruebas de aceptación.

3.4.6 Validación de la investigación

Para evaluar la satisfacción con la aplicación SIPACgraphic se escogieron 10 directivos de la institución. A la muestra seleccionada se le aplicó un cuestionario (Ver Anexo No1. Encuesta de satisfacción) que permitiera determinar el nivel de satisfacción individual y grupal a través de la técnica de IADOV. Se aplicaron 5 preguntas de las cuales 3 fueron cerradas (P1, P2 y P3) y dos abiertas (P4 y P5):

P1: ¿Sería más fácil para usted la toma de decisiones, sin utilizar la visualización de la información mediante gráficas, que le brinda la aplicación SIPACgraphic?

P2: ¿Considera que esta herramienta le es útil para agilizar su trabajo?

P3: ¿Le gusta la forma en que se diseñó la aplicación SIPACgraphic?

P4: ¿Qué importancia le concede a la aplicación SIPACgraphic?

P5: ¿Qué aspecto considera que debe mejorarse?

Las 3 preguntas cerradas se relacionaron a partir del cuadro lógico de IADOV.

Tabla 7: Cuadro lógico de IADOV

	1.¿Sería más fácil para usted la toma de decisiones, sin utilizar la visualización de la información mediante gráficas, que le brinda la aplicación SIPACgraphic?								
	No			No sé			Sí		
3.¿Le gusta la forma en que se diseñó la aplicación SIPACgraphic?	**2. ¿Considera que esta herramienta le es útil para agilizar su trabajo?**								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	2	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	6	4	3	4	4
No me gusta	6	6	6	6	6	4	6	4	5
No sé qué decir	2	3	6	3	6	3	6	3	4

La escala de satisfacción utilizada es la siguiente (32):

1. Clara satisfacción
2. Más satisfecho que insatisfecho
3. No definida
4. Más insatisfecho que satisfecho
5. Clara insatisfacción
6. Contradictorio

Para obtener el índice de satisfacción grupal (ISG) se parte de los niveles de satisfacción de cada encuestado, que están expresados a través de una escala numérica que oscila entre +1 y -1. En la siguiente tabla se muestra la cantidad de encuestados según la escala de satisfacción grupal.

Tabla 8: Cantidad de encuestados según la escala de satisfacción grupal

Escala	Significado	Cantidad de encuestados	%
+1	Clara satisfacción	7	70
0.5	Más satisfecho que insatisfecho	2	20
0	No definido y contradictorio	1	10
-0.5	Más insatisfecho que satisfecho	0	0
-1	Clara insatisfacción	0	0

El ISG se calcula utilizando la siguiente ecuación:

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

Ecuación 1. Cálculo del índice de satisfacción grupal (ISG)

Donde A, B, C, D y E representan la cantidad de encuestados ubicados en las posiciones 1, 2, 3 o 6, 4 y 5 de la escala de satisfacción y N es la cantidad total de encuestados.

Teniendo en cuenta los índices de satisfacción de cada encuestado según el cuadro lógico de IADOV, se determinó el ISG:

$$ISG = \frac{7(1) + 2(0{,}5) + 1(0) + 0(-0{,}5) + 0(-1)}{10} = 0{,}8$$

Los valores comprendidos entre -1 y -0,5 indican insatisfacción, los comprendidos entre -0,49 y 0,49 evidencian contradicción y los que están entre 0,5 y 1 indican que existe satisfacción por parte de los encuestados. Tomando en consideración que el ISG es de 0,8, se interpreta que los encuestados están satisfechos con la aplicación SIPACgraphic.

Respecto a la P4, los encuestados de manera general respondieron que la aplicación les permite reducir el tiempo de análisis del cumplimiento de los planes de trabajo y las actividades, en contribución a la toma de decisiones. Mientras que, como parte de la P5, sugirieron que además de mostrar mediante gráficas el cumplimiento de las actividades en un período de tiempo determinado, permitiera generar una comparación teniendo en cuenta iguales períodos de tiempo en años diferentes, posibilitando una evaluación más profunda.

3.5 Conclusiones parciales

En el presente capítulo se describieron elementos referentes a la implementación y validación de la solución propuesta. Se evidenciaron los estándares de codificación empleados en el desarrollo de SIPACgraphic y el diagrama de despliegue, con los recursos necesarios para la utilización de la misma. Se definió la estrategia de prueba seguida para la validación de la aplicación, documentando los resultados obtenidos a partir del empleo de las técnicas: partición de equivalencia, gráfico de prueba y camino básico. Además, mediante el empleo de

la técnica de IADOV, se evaluó la satisfacción de los usuarios de SIPACgraphic, con sus prestaciones y funcionalidades.

Conclusiones generales

Al finalizar la presente investigación y una vez implementada la propuesta de solución, se concluye que:

1. La definición de los elementos teóricos y metodológicos asociados a la visualización de información mediante gráficas, permitió profundizar en los aspectos esenciales para el desarrollo de la presente investigación.
2. El estudio de las aplicaciones existentes a nivel nacional e internacional que permiten la visualización de la información mediante gráficas, demostró que ninguna constituye una alternativa al problema planteado, por lo que se hizo necesario la implementación de la propuesta de solución de la presente investigación.
3. El análisis de los lenguajes, herramientas y tecnologías, permitió la selección de los más idóneos, que unidos a la metodología de desarrollo de software AUP-UCI, posibilitaron la implementación de la aplicación propuesta.
4. Se diseñó una solución para la visualización de información mediante gráficas para el SIPAC 3.
5. La implementación de SIPACgraphic se realizó teniendo en cuenta el uso de buenas prácticas del diseño y desarrollo de aplicaciones informáticas.
6. La estrategia de prueba definida para validar SIPACgraphic, permitió la identificación de no conformidades existentes en la aplicación y su posterior corrección.

Recomendaciones

Luego de haber cumplido los objetivos de la presente investigación y una vez implementada la propuesta de solución, se recomienda que:

- La aplicación permita mostrar mediante gráficas el cumplimiento de las actividades en un período de tiempo determinado, permitiendo generar una comparación teniendo en cuenta iguales períodos de tiempo en años diferentes, posibilitando una evaluación más profunda.

Bibliografía

1. Instrucción No. 1 del presidente de los Consejos de Estado y de Ministros.

2. Cómo hacer comprensibles los datos [online]. 2011. Available from: https://www.unece.org/fileadmin/DAM/stats/documents/writing/MDM3_SPANISH_version.pdf

3. AGUSTÍN SALABERRY. Aplicaciones móviles para llevar la estadística de tu vida. [online]. 17 February 2012. Available from: https://hipertextual.com/2012/02/aplicaciones-moviles-para-llevar-la-estadistica-de-tu-vida

4. Software CEP | SQCpack® – Análisis CEP mucho más fácil | Descripción general. [online]. [Accessed 13 June 2018]. Available from: http://www.pqsystems.com/languages/spanish/SQCpack/SQCpack.php

5. Software para Control Estadístico de Procesos - SE SPC. SoftExpert [online]. [Accessed 13 June 2018]. Available from: https://www.softexpert.com/es/produto/control-estadistico-procesos/El SE SPC hace el control estadístico continuo del proceso, proporcionando acceso fácil e inmediato a la información en forma de tablas y gráficos.

6. TAMARA, Rodríguez. Metodología de desarrollo para la Actividad productiva de la UCI. 2015.

7. CRAIG, Larman. UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid, 2000.

8. SEBESTA, Robert W. Concepts of programming languages. Pearson Education India, 1993. ISBN 81-317-2165-5.

9. BLANCO, Hermes and TÉCNICA-CBUES, Oficina. Manual de Instalación DSpace en OpenSuse 11. x. . 2011.

10. GÉNOVA, Gonzalo, FUENTES, J. and VALIENTE, M. Evaluación comparativa de herramientas CASE para UML desde el punto de vista notacional. Novática. 2006. Vol. 181, p. 59–64.

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

12. Conoce Android Studio | Android Studio. [online]. [Accessed 12 March 2018]. Available from: https://developer.android.com/studio/intro/index.html

13. Definicion de SDK. [online]. [Accessed 12 March 2018]. Available from: http://www.alegsa.com.ar/Dic/sdk.php

14. DEVELOPERS, Android. What is android. Android Developers, http://developer. android. com/guide/basics/what-is-android. html, accessed May, 2011.

15. JSON. [online]. [Accessed 12 March 2018]. Available from: http://www.json.org/json-es.html

16. Qué es la notación de modelado de procesos de negocio. Lucidchart [online]. 23 February 2017. [Accessed 24 June 2018]. Available from: https://www.lucidchart.com/pages/es/qu%C3%A9-es-la-notaci%C3%B3n-de-modelado-de-procesos-de-negocio

17. AMO, Fernando Alonso, NORMAND, Loïc Martínez and PÉREZ, Francisco Javier Segovia. Introducción a la ingeniería del software. Delta Publicaciones, 2005. ISBN 84-96477-00-2.

18. IAN, Sommerville. Ingeniería de Software. 2005.

19. BASS, Len, CLEMENTS, Paul and KAZMAN, Rick. Software architecture in practice. Addison-Wesley Professional, 2003. ISBN 0-321-15495-9.

20. GRANADA, Einer Zapata, RODRÍGUEZ, Luis Eduardo Sepúlveda, MONTOYA, Carlos Eduardo Gómez and URIBE, Christian Andrés Candela. ARQUITECTURAS DE SOFTWARE PARA ENTORNOS MÓVILES. Journal of Research of the University of Quindio. 2014. Vol. 25, no. 1.

21. Ingeniería del Software II. 2011.

22. CARLOS PLATERO DUEÑAS. Apuntes de Informática Industrial. 2014.

23. Patrones de diseño de aplicaciones: Maestro/Esclavo - National Instruments. [online]. [Accessed 8 June 2018]. Available from: http://www.ni.com/white-paper/3022/es/

24. Estándares de Codificaciónv1[1].3.doc. Google Docs [online]. [Accessed 21 May 2018]. Available from: https://docs.google.com/document/d/1rbxDFM0zsbFDNRZeM2FoXfRDbYSiSt6tCdbYPA0qdzs/edit?hl=en_US&usp=embed_facebookEstándares de Codificación 1 Introducción 1 2 Consideraciones General para Nomenclatura 3 2.1 Objetos para Ejecución 3 2.2 Objetos Multimedia 3 2.3 Objetos Fuentes 3 3 Nomenclatura Java 4 3.1 Convenciones para Variables 4 3.2 Convenciones para Constantes 5 3.3 Convenciones para Clases 5 3.4 Conve...

25. Estilo de programación y convención de nombres II. Codigolinea [online]. 25 May 2008. [Accessed 8 June 2018]. Available from: http://codigolinea.com/2008/05/25/estilo-de-programacion-y-convencion-de-nombres-ii/Estilo de programación y convención de nombres II

26. BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar, MARTÍNEZ, José Sáez and MOLINA, Jesús J. García. El lenguaje unificado de modelado. Addison Wesley Madrid, 1999.

27. IRENA JOVANOVI´C. Software testing methods and techniques. 2008.

28. ROGER S. PRESSMAN. Software Engineering. A practitioner's Approach. 7. Madrid, 2010. ISBN 978-0-07-557597-7.

29. Pruebas de Software CARRERAS PROFESIONALES CIBERTEC. 2013.

30. JOSÉ LUIS ARISTEGUI. Test cases in software test. 2010.

31. ROBERTO RUIZ TENORIO. Las pruebas de software y su importancia en las organizaciones. 2010.

32. La técnica de Iadov. Una aplicación para el estudio de la satisfacción de los alumnos por las clases de educación física. [no date].

Anexos

Anexo No1. Encuesta de satisfacción

Encuesta de Satisfacción

Esta encuesta tiene como objetivo conocer el grado de satisfacción grupal de los usuarios con la aplicación SIPACgraphic.

Pregunta # 1: ¿Sería más fácil para usted la toma de decisiones, sin utilizar la visualización de la información mediante gráficas, que le brinda la aplicación SIPACgraphic?

Sí _____ No _____ No sé _____

Pregunta # 2: ¿Considera que esta herramienta le es útil para agilizar su trabajo?

Sí _____ No _____ No sé _____

Pregunta # 3: ¿Le gusta la forma en que se diseñó la aplicación SIPACgraphic?

Sí _____ No _____ No sé _____

Pregunta # 4: ¿Qué importancia le concede a la aplicación SIPACgraphic?

_____

_____

_____

_____

Pregunta # 5: ¿Qué aspecto considera que debe mejorarse?

_____

_____

_____

_____

Figura 27: Encuesta de satisfacción