



**ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA**
Universidad de Córdoba

**EP
SC**

TRABAJO DE FIN DE GRADO

- Memoria Técnica -

Grado en Ingeniería Informática

Mención en Computación

**Simulador de algoritmos de planificación de
procesos para la asignatura de Sistemas
Operativos del Grado de Ingeniería Informática
de la Universidad de Córdoba**

Autor

Julen Pérez Hernández

Director

Juan Carlos Fernández Caballero



UNIVERSIDAD DE CÓRDOBA

D. Juan Carlos Fernández Caballero, Profesor Contratado Doctor del Departamento de Informática y Análisis Numérico en la Escuela Politécnica Superior de Córdoba de la Universidad de Córdoba e investigador del grupo AYRNA (Aprendizaje y Redes Neuronales Artificiales).

Informan:

Que el presente Trabajo Fin de Grado de Ingeniería Informática titulado “Simulador de algoritmos de planificación de procesos para la asignatura de Sistemas Operativos del Grado de Ingeniería Informática de la Universidad de Córdoba”, constituye la memoria presentada por Julen Pérez Hernández para aspirar al título de Graduado en Ingeniería Informática, y ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de Córdoba de la Universidad de Córdoba, reuniendo, a mi juicio, las condiciones necesarias exigidas en este tipo de trabajos.

Y para que conste, se expide y firma el presente informe en Córdoba, junio de 2023.

- Autor:

Fdo: Julen Pérez Hernández

- Director:

Fdo: Juan Carlos Fernández Caballero

Agradecimientos

Quiero expresar mi más sincero agradecimiento al profesor tutor de este proyecto, Juan Carlos Fernández Caballero, por su dedicación, paciencia y valiosas sugerencias, que me permitieron llevar a cabo este trabajo con éxito.

Asimismo, no puedo dejar de mencionar a mi familia, quienes siempre estuvieron a mi lado brindándome un gran apoyo emocional y económico, lo que me permitió enfocarme completamente en mis estudios universitarios.

También quiero agradecer a mis amigos, quienes me alentaron a seguir adelante cuando las cosas se ponían difíciles, y me dieron su apoyo incondicional en todo momento.

Finalmente, me gustaría agradecerme a mí mismo, por nunca darme por vencido y continuar trabajando duro hasta alcanzar el final de esta importante etapa.

Índice General

I. Introducción	1 -
1. Introducción	3 -
2. Definición del problema	5 -
2.1. Identificación del problema real.....	5 -
2.2. Identificación del problema técnico.....	6 -
3. Objetivos.....	15 -
3.1. Objetivos formales.....	15 -
3.2. Objetivos operacionales.....	16 -
4. Antecedentes.....	19 -
4.1. Sitios Web Online.....	19 -
4.2. Aplicación descargable para ordenador personal.....	23 -
4.3. Dispositivos Móviles	27 -
4.4. Conclusiones	31 -
5. Restricciones	33 -
5.1. Factores Dato	33 -
5.2. Factores Estratégicos	34 -
6. Recursos.....	37 -
6.1. Recursos Humanos	37 -
6.2. Recursos Materiales.....	37 -
II. Análisis del sistema y especificaciones de requisitos	39 -
7. Especificación de requisitos.....	41 -
7.1. Requisitos de Información	41 -
7.2. Requisitos Funcionales	43 -
7.3. Requisitos No Funcionales	44 -
8. Análisis funcional	47 -
8.1. Identificación de los usuarios objetivo de la app.....	48 -

8.2. Casos de uso de contexto	- 48 -
8.3. Casos de uso	- 50 -
III. Diseño del sistema.....	- 59 -
9. Arquitectura del sistema	- 61 -
10. Diseño de la interfaz.....	- 63 -
10.1. Pantalla de carga	- 65 -
10.2. Página principal.....	- 66 -
10.3. Página de resultados temporales.....	- 69 -
10.4. Página de diagrama de Gantt.....	- 70 -
10.5. Página de colas de procesos	- 71 -
IV. Pruebas	- 73 -
11. Pruebas	- 75 -
11.1. Estrategia de pruebas.....	- 75 -
11.2. Pruebas de caja blanca	- 77 -
11.3. Pruebas de caja negra	- 82 -
11.4. Pruebas de integración	- 86 -
11.5. Pruebas de sistema.....	- 87 -
11.6. Pruebas de aceptación.....	- 89 -
V. Conclusiones y futuras mejoras	- 93 -
12. Conclusiones	- 95 -
12.1. Objetivos operacionales alcanzados	- 95 -
12.2. Objetivos formales alcanzados	- 96 -
13. Futuras mejoras	- 99 -
Bibliografía	- 101 -

Índice de Figuras

Figura 1. Ejemplo de ejecución del simulador de Boonsuen.	20 -
Figura 2. Ejemplo de ejecución del simulador de la Universidad de Dortmund.	21 -
Figura 3. Ejemplo sencillo de ejecución del simulador.	23 -
Figura 4. Ejemplo de ejecución del algoritmo FIFO.	24 -
Figura 5. Ingreso de datos para simular los procesos.	25 -
Figura 6. Gráfico resultante de la simulación.	25 -
Figura 7. Ejemplo de simulación y selección del algoritmo FIFO.	26 -
Figura 8. Ejemplo de simulación del algoritmo RR.	28 -
Figura 9. Selección de datos iniciales.	29 -
Figura 10. Resultado en gráfico de barras.	30 -
Figura 11. CUC-00.	49 -
Figura 12. CU-01.	51 -
Figura 13. CU-02.	53 -
Figura 14. CU-03.	54 -
Figura 15. CU-04.	55 -
Figura 16. CU-05.	56 -
Figura 17. CU-06.	57 -
Figura 18. CU-07.	58 -
Figura 19. Pantalla de carga.	65 -

Figura 20. Página principal.	- 66 -
Figura 21. Página principal con proceso y E/S seleccionado.	- 67 -
Figura 22. Página principal con procesos con E/S.....	- 68 -
Figura 23. Página de resultados.	- 69 -
Figura 24. Página de gráfica de Gantt.	- 70 -
Figura 25. Página de colas de procesos.....	- 72 -

Índice de Tablas

Tabla 1. Plantilla de Caso de Uso	- 47 -
Tabla 2. CUC-00. Caso de uso de contexto.	- 48 -
Tabla 3. CU-01. Selección del algoritmo e Introducción de datos.	- 50 -
Tabla 4. CU-02. Visualizar y Modificar los datos.	- 52 -
Tabla 5. CU-03. Iniciar simulación.	- 54 -
Tabla 6. CU-04. Visualizar los resultados.	- 55 -
Tabla 7. CU-05. Visualizar el diagrama de Gantt.	- 56 -
Tabla 8. CU-06. Visualiza las colas de estados de los procesos.	- 57 -
Tabla 9. CU-07. Cambio de tema de la aplicación.	- 58 -
Tabla 10. Pruebas de caja blanca para CU-01.	- 78 -
Tabla 11. Pruebas de caja blanca para CU-02.	- 79 -
Tabla 12. Pruebas de caja blanca para CU-03.	- 79 -
Tabla 13. Pruebas de caja blanca para CU-04.	- 80 -
Tabla 14. Pruebas de caja blanca para CU-05.	- 80 -
Tabla 15. Pruebas de caja blanca para CU-06.	- 81 -
Tabla 16. Pruebas de caja blanca para CU-07.	- 81 -
Tabla 17. Pruebas de caja negra para CU-01.	- 82 -
Tabla 18. Pruebas de caja negra para CU-02.	- 83 -
Tabla 19. Pruebas de caja negra para CU-03.	- 83 -
Tabla 20. Pruebas de caja negra para CU-04.	- 84 -

Tabla 21. Pruebas de caja negra para CU-05.	- 85 -
Tabla 22. Pruebas de caja negra para CU-06.	- 85 -
Tabla 23. Pruebas de caja negra para CU-07.	- 86 -

Bloque I. Introducción

Capítulo

1. Introducción

La planificación de procesos es una tarea fundamental del sistema operativo. Los entornos de simulación se utilizan para el estudio del comportamiento de los algoritmos de planificación.

La eficiencia de un algoritmo de planificación de procesos es uno de los apartados más importantes a considerar. El sistema operativo es el encargado de aplicar estos algoritmos en función de las políticas de planificación que se estén llevando a cabo en cada caso, gestionando la entrada y tiempo de procesamiento de cada uno de los procesos dentro de la CPU.

Las principales características de estos algoritmos de planificación son el rendimiento I/O, el tiempo de espera, la utilización de la CPU, el tiempo de respuesta, la prioridad y el tiempo total requerido por cada proceso.

Teniendo lo anterior en cuenta, se puede definir OSSAS (Operating Systems Scheduling Algorithms Simulator) como las siglas bajo las que se denomina a las múltiples herramientas para simular distintos escenarios, para que el sistema operativo determine el orden en que se adecúa el procesador a los procesos que lo vayan necesitando y a las políticas que se utilizarán en la eficiencia del tiempo esperado en el sistema [\[1\]](#).

Existen diversos algoritmos para dirigir la ordenación de procesos [\[2\]](#) como, por ejemplo, el algoritmo de planificación FCFS (First come First served), el algoritmo de planificación SJF (Short Job First), algoritmo de turno rotatorio (Round Robin), y algoritmo de prioridades, entre otros [\[3\]](#). Desde la perspectiva de la enseñanza, estos temas suelen abordarse llevando a cabo numerosos ejercicios en papel o pizarra. Esto deriva,

frecuentemente, en la necesidad de simplificar en exceso los ejercicios o, por el contrario, en tener que desarrollar esquemas de compleja visualización.

Tras ahondar en esta perspectiva, surge la idea de la creación de una herramienta de apoyo a la docencia en torno a la simulación de las políticas de planificación del sistema operativo, para la asignatura de Sistemas Operativos de 2º de Grado de Ingeniería Informática de la Escuela Politécnica Superior de Córdoba. De esta manera, tanto alumnos como profesores se beneficiarían de su uso al poder realizar un mayor número de ejercicios en este ámbito, permitiendo agregarles cierta complejidad sin sacrificar, a cambio, su comprensión o sencillez visual. Por otro lado, el desarrollo de una herramienta digital también permitiría al alumnado simular ejercicios en diferentes dispositivos y en cualquier lugar, ya que se realizará un desarrollo dirigido a los dispositivos cuyo sistema operativo se encuentre basado en Android.

Capítulo

2. Definición del problema

En esta sección, se expondrá de forma concisa el problema principal que impulsa esta investigación, explorando su importancia y alcance. Se analizarán los elementos fundamentales para comprender su naturaleza y aportar posibles soluciones.

2.1. Identificación del problema real

El ámbito educativo se ha beneficiado enormemente de los avances tecnológicos en las últimas décadas, y la integración de herramientas digitales en la enseñanza se ha convertido en un recurso esencial para mejorar la comprensión y el aprendizaje de conceptos complejos. En este contexto, se plantea la cuestión de cómo mejorar la enseñanza y el entendimiento de los algoritmos de planificación de procesos haciendo uso de estas nuevas tecnologías.

En las aulas universitarias, la comprensión de los algoritmos de planificación de procesos es fundamental para formar a futuros profesionales capaces de optimizar el rendimiento de sistemas informáticos. Sin embargo, esta área a menudo presenta desafíos conceptuales que pueden ser difíciles de abordar únicamente a través de métodos tradicionales de enseñanza. Aquí es donde surge la necesidad de una herramienta que proporcione una experiencia interactiva y visual para el aprendizaje de estos algoritmos.

El objetivo central de este Trabajo de Fin de Grado es desarrollar una aplicación móvil para dispositivos Android que simule de manera interactiva diversos algoritmos de planificación de procesos. Esta aplicación se concibe como una herramienta complementaria a la educación en la universidad, diseñada para brindar a los estudiantes una comprensión práctica y tangible de cómo funcionan estos algoritmos en la gestión de la CPU.

La necesidad de esta aplicación se fundamenta en la idea de que la simulación visual y práctica puede simplificar conceptos complejos y mejorar la retención del conocimiento. Al proporcionar a los estudiantes la capacidad de interactuar con diferentes escenarios de planificación de procesos, se espera que la aplicación facilite la asimilación de conceptos teóricos y, en última instancia, mejore la calidad de la educación en este campo.

A lo largo de este apartado, se analizará en mayor profundidad la relevancia de esta problemática, se identificarán los beneficios potenciales de la solución propuesta y se delinearán las metas y objetivos específicos que guiarán el desarrollo de la aplicación.

2.2. Identificación del problema técnico

En este segmento, se abordará el desafío técnico central que enfrenta la creación de una aplicación simuladora de algoritmos de planificación de procesos para dispositivos Android, empleando como guía la PDS (Especificación de Diseño de Producto). Se explorarán las complejidades inherentes a la implementación y visualización precisa de estos algoritmos en un entorno móvil, considerando aspectos como eficiencia, interactividad y experiencia del usuario. Este análisis técnico sentará las bases para diseñar soluciones efectivas que permitan superar los obstáculos técnicos y lograr los objetivos de la aplicación educativa propuesta.

2.2.1. Funcionamiento

Respecto al funcionamiento de la aplicación se dispondrá de las siguientes funcionalidades:

1. **Selección de Algoritmos:** La aplicación ofrecerá la posibilidad de elegir entre distintos algoritmos de planificación de procesos, permitiendo al usuario especificar la metodología que desea estudiar.
2. **Ingreso de Datos Precisos:** En cada instancia, se proporcionarán campos requeridos para una entrada exhaustiva de datos pertinentes al funcionamiento del algoritmo seleccionado. Esto asegurará una simulación precisa y detallada.
3. **Flexibilidad en la Configuración de Procesos:** Se habilitará la modificación de procesos añadidos, posibilitando la ejecución de múltiples simulaciones con diferentes datos o algoritmos, fomentando un análisis comparativo.

4. **Visualización Avanzada de Resultados:** La aplicación ofrecerá diversas modalidades de presentación de resultados para facilitar el entendimiento del usuario. Esto incluirá tablas de datos y diagramas temporales, contribuyendo a una representación clara de la información.
5. **Múltiples Pantallas Focalizadas:** Se dispondrá de distintas pantallas que permitirán al usuario concentrarse en una representación visual específica de los resultados obtenidos, optimizando la comprensión de cada aspecto analizado.
6. **Diagrama de Tiempos de la Cola de Procesos en CPU:** Una de las pantallas permitirá la observación detallada del diagrama temporal que ilustra la dinámica de la cola de procesos en la Unidad Central de Procesamiento (CPU).

2.2.2. Entorno

La aplicación se desplegará en un entorno operativo a partir de la versión 7.0 de Android o posteriores. Los usuarios podrán acceder y adquirir la aplicación a través de la plataforma oficial de distribución de aplicaciones, la Play Store de Google. Este entorno asegura la compatibilidad con dispositivos que cumplen con los requisitos de sistema establecidos por Android 7.0 y proporciona una amplia accesibilidad a través de una fuente confiable y reconocida para los usuarios.

2.2.3. Vida esperada

El ciclo de vida de la aplicación no está restringido por un límite temporal predeterminado, siempre y cuando se cumplan los requisitos mínimos de compatibilidad con el sistema operativo Android 7.0 o versiones superiores.

Al establecer la compatibilidad con versiones actuales y futuras del sistema operativo Android, junto con la posibilidad de mantenimiento y actualizaciones, la aplicación puede mantener su relevancia y utilidad sin una restricción temporal rígida. La duración del ciclo de vida dependerá en gran medida de su utilidad continuada para los usuarios y su capacidad de adaptación a los cambios tecnológicos y educativos.

2.2.4. Ciclo de mantenimiento

En el alcance de este proyecto, se concibe el producto final como una versión definitiva y completa. Aunque no se contempla un ciclo de mantenimiento continuo, se implementará un diseño de aplicación altamente modular.

Este enfoque modular permitirá abordar de manera eficiente la corrección de posibles fallos en el futuro y la adición de nuevas funcionalidades. La estructura modular garantizará que cualquier cambio o mejora se pueda llevar a cabo con el menor esfuerzo y perturbación posible en la funcionalidad principal del producto.

2.2.5. Competencia

Dentro del panorama actual, se constata la existencia de diversas alternativas a la presente aplicación, abarcando incluso distintas plataformas. Sin embargo, el enfoque central de este proyecto no radica en competir con estas alternativas, sino en unir las características destacadas identificadas en dichas soluciones. Estas cualidades positivas serán expuestas en detalle en el próximo apartado de Antecedentes.

La visión primordial consiste en concebir una aplicación que capture la esencia distintiva de la Universidad de Córdoba, con un doble propósito: servir como una herramienta complementaria al proceso docente y como un punto de partida modular, permitiendo a futuros alumnos expandir y enriquecer sus funcionalidades como parte de sus propios Trabajos de Fin de Grado.

2.2.6. Aspecto externo

Con relación al aspecto de la aplicación, se buscan destacar los siguientes aspectos:

1. **Diseño Amigable e Intuitivo:** Se persigue lograr un diseño visual altamente amigable para el usuario, con el objetivo de que la aplicación sea intuitiva en su uso, sin requerir la necesidad de consultar manuales. El diseño se centrará en una experiencia de usuario fluida y natural.
2. **Paleta de Colores Minimalista y Representativa:** Los colores empleados han sido cuidadosamente seleccionados para adoptar un enfoque minimalista, al mismo tiempo que se mantienen en sintonía con la identidad visual característica de la universidad [\[4\]](#). Esta elección cromática contribuirá a una coherencia visual y a una identificación inmediata con la institución.
3. **Visualización mediante Paginación Continua:** La disposición de las pantallas de la aplicación se implementará en un formato de paginación continua. Esta elección busca brindar al usuario la sensación de interactuar con una herramienta compacta

y organizada, facilitando la navegación y el acceso a las distintas funcionalidades de manera fluida.

En conjunto, estos elementos de diseño contribuirán a la creación de una aplicación visualmente atractiva, fácil de usar y en sintonía con la estética de la universidad, promoviendo una experiencia de usuario positiva y efectiva.

2.2.7. Estandarización

En pro de garantizar la coherencia, eficiencia y calidad en el desarrollo de la aplicación, se han establecido rigurosos estándares y prácticas de estandarización. Estos elementos fundamentales contribuirán a mantener la integridad y la robustez del proyecto, así como a facilitar futuras expansiones y mantenimiento. Entre las principales áreas de estandarización se incluyen:

1. **Nomenclatura Uniforme:** Se ha adoptado una nomenclatura coherente y descriptiva para las variables, funciones y clases en el código fuente. Esto permitirá una comprensión más rápida y precisa de la estructura del proyecto, optimizando la posible futura colaboración de nuevos desarrolladores.
2. **Documentación Detallada:** Se ha implementado una documentación exhaustiva que abarca la estructura general del código, las funcionalidades clave y los algoritmos empleados. Esta documentación servirá como recurso esencial para futuros desarrolladores y como guía para la expansión y optimización del proyecto.
3. **Comentarios y Anotaciones:** El código se encuentra enriquecido con comentarios claros y anotaciones pertinentes que explican procesos, decisiones de diseño y secciones críticas. Estas adiciones permitirán un mayor nivel de legibilidad y comprensión para quienes trabajen en el código en etapas posteriores.
4. **Diseño Responsivo:** La interfaz de usuario se ha desarrollado siguiendo principios de diseño responsivo, garantizando una experiencia visual y funcional coherente en una variedad de tamaños de pantalla.
5. **Pruebas y Validación:** Se han establecido procedimientos de pruebas unitarias y de integración exhaustivas para verificar la funcionalidad y calidad del código (ver [Bloque 4. Pruebas](#)). Estas pruebas se aplicarán tanto a las características existentes

como a las nuevas implementaciones, asegurando un alto estándar de rendimiento y eficacia.

6. **Control de Versiones:** La gestión de versiones se lleva a cabo utilizando un sistema de control de versiones (Git), lo que permite un seguimiento preciso de los cambios realizados en el código con el tiempo. Esto facilita la colaboración y proporciona la capacidad de revertir cambios si fuera necesario.
7. **Compatibilidad:** La aplicación se ha diseñado siguiendo las mejores prácticas de desarrollo y optimización de Android, intentando siempre asegurar la compatibilidad con diferentes versiones de Android (versión 7.0 y posteriores) y dispositivos, lo que garantiza un rendimiento óptimo en una variedad de contextos.
8. **Actualizaciones Futuras:** La arquitectura de la aplicación se ha concebido de manera modular, lo que permitirá incorporar nuevas funcionalidades, mejoras y correcciones de errores sin comprometer la estabilidad general de la aplicación (ver [Capítulo 13. Futuras Mejoras](#)).
9. **Usabilidad:** El diseño de la interfaz de usuario sigue los principios de usabilidad y diseño centrado en el usuario, garantizando interacciones intuitivas y fluidas que mejoren la experiencia general del usuario.

Estos elementos de estandarización se han implementado con el objetivo de asegurar la calidad, la flexibilidad y la continuidad en el desarrollo y evolución de la aplicación, aportando una base sólida para su presente y futuro.

2.2.8. Calidad y fiabilidad

La calidad y fiabilidad de la aplicación son aspectos esenciales para garantizar su funcionalidad, estabilidad y aceptación por parte de los usuarios. En este contexto, se han adoptado medidas exhaustivas que abarcan desde el proceso de desarrollo hasta las pruebas y validaciones rigurosas. Estos enfoques se han diseñado con el propósito de asegurar una experiencia de usuario superior y una aplicación confiable en todo momento.

1. **Pruebas Exhaustivas:** La aplicación ha sido sometida a pruebas extensas para verificar su comportamiento bajo diferentes escenarios y condiciones.

2. **Control de Calidad Continuo:** Se ha implementado un proceso de control de calidad continuo que incluye revisión de código por parte de pares y supervisión regular del progreso del desarrollo. Esto garantiza que el código cumpla con los estándares de calidad establecidos y que los problemas se detecten y aborden tempranamente.
3. **Optimización de Rendimiento:** Se han realizado esfuerzos para optimizar el rendimiento de la aplicación, y buscar siempre tiempos de respuesta rápidos y correcta eficiencia en la ejecución de procesos. Esto se traduce en una experiencia fluida y ágil para el usuario.
4. **Gestión de Errores y Excepciones:** Se ha implementado un sistema sólido de manejo de errores y excepciones para anticipar y controlar situaciones inesperadas. Esto contribuye a la estabilidad de la aplicación y minimiza posibles fallas críticas.
5. **Compatibilidad de Plataforma:** La aplicación ha sido diseñada para cumplir con los requisitos de Android 7.0 y versiones posteriores, garantizando su funcionamiento en un amplio rango de dispositivos.

2.2.9. Programa de tareas

Se llevará a cabo un proceso de desarrollo riguroso y estructurado que se dividirá en varias fases clave. Estas fases serán de vital importancia para la consecución de los objetivos planteados, y permitirán abordar de manera sistemática y eficiente cada uno de los aspectos que conforman el proyecto en su totalidad.

- **Estudio y análisis del problema:** En la fase de estudio y análisis del problema, se llevará a cabo una evaluación exhaustiva de los requisitos necesarios para la creación de una aplicación Android (ver [Bloque 2. Análisis del sistema y especificaciones de requisitos](#)).

En primer lugar, se realizará una revisión detallada de las funcionalidades requeridas para la correcta ejecución de los algoritmos, teniendo en cuenta que la aplicación va destinada al apoyo a la docencia de la asignatura de sistemas operativos, y por tanto de los tipos de ejercicios sobre planificación que en ella se proponen. A su vez, se llevará a cabo una investigación sobre las características más relevantes de los algoritmos de planificación de procesos. En esta fase también se tendrán en cuenta los aspectos técnicos necesarios para el correcto funcionamiento

de la aplicación, como el uso de herramientas y lenguajes de programación específicos para Android o la selección de una arquitectura adecuada, entre otros.

- **Diseño:** En la fase de diseño (ver [Bloque 3. Diseño del sistema](#)) se llevará a cabo la conceptualización y planificación de la aplicación a desarrollar. Se definirán los componentes y elementos necesarios para la creación de una aplicación intuitiva y atractiva que permita a los usuarios comprender y utilizar la aplicación de manera eficiente.

Asimismo, se establecerán las especificaciones técnicas y funcionales para la programación de la aplicación, la elección de la arquitectura y la definición de las características y funcionalidades de la aplicación.

La fase de diseño permitirá establecer las bases para la programación de una aplicación robusta y eficiente que cumpla con los requerimientos definidos en la fase de estudio y análisis del problema previos.

- **Codificación:** En la fase de codificación se llevará a cabo la implementación de la aplicación de simulación, siguiendo las especificaciones definidas en la fase de diseño. Los lenguajes de programación más comunes para el desarrollo de aplicaciones Android son Java y Kotlin, aunque será este último el seleccionado para este desarrollo debido a que es el lenguaje actual de desarrollo para Google y Android y tiene una gran cantidad de documentación y ejemplos de estudio. Además, se utilizarán herramientas de desarrollo integrado (IDE) para agilizar y facilitar el proceso de codificación. Android Studio es el IDE oficial de Android y ofrece una amplia variedad de herramientas para el desarrollo de aplicaciones. Este punto se ampliará en secciones posteriores.
- **Pruebas:** En la fase de pruebas (ver [Bloque 4. Pruebas](#)), se llevará a cabo un proceso de evaluación exhaustivo de la aplicación con el objetivo de detectar y corregir posibles errores o fallos en su funcionamiento. Se probará la aplicación en diferentes dispositivos móviles, tanto en móviles como en tabletas, para asegurar que la aplicación es compatible y funciona correctamente en diferentes tamaños de pantalla y resoluciones. Para llevar a cabo estas pruebas, se utilizarán herramientas de prueba y depuración, como el depurador de Android Studio o herramientas de automatización de pruebas, para facilitar el proceso y garantizar la precisión y calidad de las pruebas.

- **Documentación:** En esta fase, se elaborarán tres manuales esenciales para la comprensión y mantenimiento de este proyecto.

En primer lugar, se desarrollará un manual técnico que describirá los detalles técnicos del proyecto, incluyendo la arquitectura, diseño y funcionamiento de la aplicación. Este manual deberá ser utilizado por desarrolladores futuros para mantener y mejorar el proyecto.

En segundo lugar, se creará un manual de usuario que explicará cómo utilizar la aplicación, detallando las diferentes funcionalidades, pantallas y opciones. Este manual será útil para que los usuarios puedan utilizar la aplicación y comprender su funcionamiento.

Por último, se redactará un manual de código que describirá el código fuente del proyecto. Este manual será utilizado por otros desarrolladores que deseen trabajar con el proyecto en el futuro.

2.2.10. Pruebas

La realización de pruebas exhaustivas juega un papel fundamental en la creación de una aplicación robusta y confiable. En este contexto, se adopta un enfoque integral para garantizar que cada aspecto de la aplicación funcione correctamente y que los problemas se identifiquen y aborden de manera efectiva. Se destacan los siguientes puntos clave en el proceso de pruebas:

1. **Pruebas Independientes por Módulo:** Para garantizar la integridad de cada módulo individual, se realizan pruebas independientes para evaluar su funcionamiento. Esto se hace con el fin de detectar cualquier problema antes de que pueda propagarse a otras áreas de la aplicación.
2. **Pruebas de Integración:** Una vez que se han verificado los módulos de forma individual, se procede a las pruebas de integración. Estas pruebas validan el funcionamiento conjunto de los módulos, asegurando que la interacción entre ellos sea fluida y coherente.
3. **Herramientas de Android Studio:** Android Studio ofrece un conjunto de herramientas poderosas para facilitar el proceso de pruebas. La depuración en

tiempo real y la simulación de ejecución en diversos dispositivos virtuales permiten identificar y solucionar problemas de manera eficiente.

4. **Pruebas en Dispositivos Físicos:** Además de las pruebas en dispositivos virtuales, se llevan a cabo pruebas en dispositivos físicos. Esto proporciona una visión más precisa del rendimiento y la experiencia del usuario en condiciones del mundo real.
5. **Verificación de Compatibilidad:** Se verifica la compatibilidad de la aplicación en diferentes versiones de Android y dispositivos con diversas configuraciones. Esto garantiza que la aplicación funcione de manera óptima en una variedad de escenarios.
6. **Registro de Errores:** Se registra y documenta cualquier error encontrado durante las pruebas, lo que facilita su seguimiento y posterior corrección.
7. **Iteraciones y Mejoras:** Las pruebas son un proceso iterativo. Cualquier problema identificado se corrige, se realizan nuevas pruebas y se repite el ciclo hasta que la aplicación alcance un nivel de calidad óptimo.

En conclusión, las pruebas y la validación se abordan con un enfoque meticuloso y sistemático para garantizar la funcionalidad, la estabilidad y la calidad general de la aplicación. Se aprovechan las herramientas disponibles en Android Studio y se implementan métodos variados para asegurar que la aplicación satisfaga las expectativas del usuario y ofrezca una experiencia confiable y fluida.

Capítulo

3. Objetivos

En esta sección, se delinearán los objetivos del proyecto, distinguiendo entre los aspectos formales y operacionales. Los objetivos formales definen las metas generales del proyecto, mientras que los operacionales detallan las acciones específicas para lograr esas metas.

3.1. Objetivos formales

En esta sección, se establecen los objetivos generales del proyecto, trazando la dirección y el propósito de la investigación. Estos objetivos formales se han definido con el propósito de establecer una base sólida de conocimientos y habilidades que permitan alcanzar las metas generales del proyecto con un enfoque técnico y metodológico riguroso:

1. **Adquisición de Conocimientos de Desarrollo Integral:** Obtener una comprensión profunda de las complejidades intrínsecas a la creación de aplicaciones, explorando con detalle cada etapa del proceso de desarrollo.
2. **Manejo de Android Studio:** Adquirir experiencia en el manejo de Android Studio [\[6\]](#), la plataforma de desarrollo principal para la creación de aplicaciones en el ecosistema Android.
3. **Iniciación en el Lenguaje de Programación Kotlin:** Adquirir un nivel adecuado en el lenguaje de programación Kotlin [\[7\]](#), utilizando su potencial para el desarrollo efectivo de la aplicación.
4. **Aprendizaje en Jetpack Compose:** Familiarizarse en el uso de Jetpack Compose [\[8\]](#), la biblioteca de diseño de interfaces de última generación, para desarrollar experiencias de usuario intuitivas y modernas.

5. **Análisis y Resolución de Problemas:** Perfeccionar la capacidad de analizar y resolver problemas complejos que puedan surgir durante el desarrollo, garantizando la estabilidad y eficacia del producto.
6. **Toma de Decisiones Estratégicas para la Experiencia de Usuario:** Tomar decisiones estratégicas basadas en la comprensión profunda de las necesidades del usuario y los principios de diseño para mejorar la experiencia global del usuario.
7. **Optimización de Rendimiento:** Implementar estrategias avanzadas para optimizar los recursos y lograr un rendimiento adecuado en diversos escenarios de uso.
8. **Generación de Documentación Completa:** Elaborar una documentación completa y detallada que abarque todos los aspectos del proyecto, desde la concepción hasta la implementación, asegurando la transparencia y la replicabilidad del proceso.

3.2. Objetivos operacionales

Los objetivos operacionales detallan las acciones y pasos concretos que se deben emprender para alcanzar los objetivos formales. Estas metas específicas orientan el proceso de desarrollo y permiten una planificación precisa de las actividades necesarias para cumplir con los objetivos generales del proyecto.

3.2.1. Objetivo principal

El objetivo primordial del proyecto radica en la concepción y desarrollo integral de una aplicación dirigida a dispositivos móviles basados en el sistema operativo Android. Esta aplicación se concibe con el propósito de desempeñar un papel fundamental como herramienta de respaldo en la enseñanza de los conceptos relacionados con la planificación de procesos, específicamente en el contexto de la asignatura de Sistemas Operativos.

Uno de los pilares distintivos de la aplicación es su capacidad para simular los eventos de E/S (entrada/salida) de los procesos. La implementación de uno o dos algoritmos de planificación se sitúa como foco central del desarrollo. Esta selección se deriva de la aspiración de construir una base sólida y funcional, que pueda evolucionar de manera eficiente en el tiempo. La aplicación base se concibe como robusta y adaptable, permitiendo la incorporación futura de un repertorio más amplio de características y tipos de algoritmos, de acuerdo con las necesidades y objetivos de la educación en el ámbito de la planificación de procesos.

3.2.2. Objetivos específicos

En esta sección, se detallan los objetivos específicos que guiarán el desarrollo de la aplicación, abarcando desde la usabilidad y simulación de eventos hasta la presentación efectiva de resultados.

OBJ-1. Diseño Compacto y Accesible: Crear una interfaz de usuario que presente información de manera concisa y accesible. Se asegurará que los elementos clave estén visibles y organizados de manera intuitiva para que los usuarios puedan acceder a ellos sin necesidad de entrar en submenús adicionales.

OBJ-2. Navegación Clara entre Módulos: Diseñar una navegación fluida y lógica que permita a los usuarios moverse sin problemas entre los diferentes módulos de la aplicación.

En la perspectiva de la navegación, se ha seleccionado un diseño de formato paginado, donde cada módulo o sección se encuentra en una página individual, permitiendo una accesibilidad fluida mediante un desplazamiento lateral.

OBJ-3. Selección Sencilla de Algoritmos: Implementar un sistema de selección de algoritmos claro y fácil de usar. Permitir que los usuarios elijan de manera sencilla entre diferentes algoritmos de planificación, brindando una comprensión rápida de las opciones disponibles.

OBJ-4. Especificación Clara de Datos Necesarios: Desarrollar interfaces claras y precisas para que los usuarios ingresen los datos necesarios para cada algoritmo. Proporcionar formularios estructurados que guíen al usuario a introducir los valores requeridos de manera eficiente y unívoca.

OBJ-5. Gestión de Eventos de E/S: Diseñar un sistema que permita a los usuarios especificar eventos de entrada/salida para cada proceso agregado. Se dispondrá de campos específicos para poder introducir los momentos en los que el proceso en cuestión realiza las acciones de entrada y salida de la cola de listos en CPU.

OBJ-6. Visualización Clara de Datos Ingresados: Presentar de manera clara y organizada los datos que el usuario introduce en la aplicación mediante una tabla. Utilizar

elementos visuales y disposiciones limpias para asegurar que los datos sean fáciles de comprender y verificar.

OBJ-7. Eliminación Flexible de Procesos: Implementar la capacidad de eliminar procesos individualmente o en su totalidad. Se proporcionarán controles intuitivos que permitan a los usuarios administrar los procesos agregados según sus necesidades.

OBJ-8. Visualización Numérica de Resultados: Desarrollar un módulo que muestre los resultados numéricos obtenidos de la simulación. Presentar los datos de manera clara y organizada para que los usuarios puedan analizar los resultados de manera efectiva.

OBJ-9. Visualización Gráfica de Tiempos: Crear un módulo que permita a los usuarios ver los tiempos asociados a cada proceso mediante representaciones gráficas (en un diagrama de Gantt [\[9\]](#)). Se asegurará que los gráficos sean claros y comprensibles, incluyendo una leyenda que explique la información visualizada.

OBJ-10. Visualización de Estado de Colas: Implementar un módulo que muestre el estado de las diferentes colas de procesos en cada etapa de la simulación. Proporcionar una vista clara y actualizada para que los usuarios puedan seguir el progreso de los procesos.

Capítulo

4. Antecedentes

En el contexto del desarrollo de aplicaciones Android, los antecedentes se refieren a las soluciones previas, herramientas y enfoques utilizados en proyectos similares que han abordado problemáticas análogas. Estos antecedentes son fundamentales para comprender el panorama existente y aprovechar las mejores prácticas y lecciones aprendidas en la creación de nuevas aplicaciones.

En el caso específico de esta propuesta, se han explorado diversas opciones y se ha investigado ampliamente en el ámbito de la planificación de procesos, considerando alternativas tanto en el entorno de dispositivos móviles como en otras plataformas (páginas web o computadoras). Esto ha implicado un análisis minucioso de soluciones disponibles en sistemas operativos, software educativo y aplicaciones de simulación en general. Esta exploración se ha extendido también a plataformas distintas a los dispositivos móviles, en la búsqueda de enfoques exitosos y transferibles que puedan enriquecer la propuesta.

Este proceso de investigación y exploración de alternativas ha brindado una base sólida para identificar las fortalezas y debilidades de soluciones existentes, destacando así las oportunidades para mejorar y diferenciar la aplicación que se está desarrollando. La amalgama de conocimiento obtenida de diversos antecedentes ha permitido perfilar una propuesta que capitaliza las mejores prácticas y enriquece la propuesta con características distintivas y eficaces.

4.1. Sitios Web Online

Se han examinado plataformas que ofrecen simuladores de planificación de procesos accesibles directamente desde el navegador, eliminando la necesidad de instalar software adicional en los dispositivos de los usuarios. A continuación, se presentan algunos ejemplos de sitios web que hospedan tales simuladores, permitiendo una visión más amplia de las opciones disponibles en el mercado y proporcionando una base sólida para la propuesta en desarrollo.

4.1.1. Boonsuen Process Scheduling Solver

Este simulador pertenece al autor “Boonsuen” en GitHub [4], donde se podrá revisar el código utilizado por el simulador para poder comprobar la programación y el funcionamiento de cada uno de los métodos utilizados.

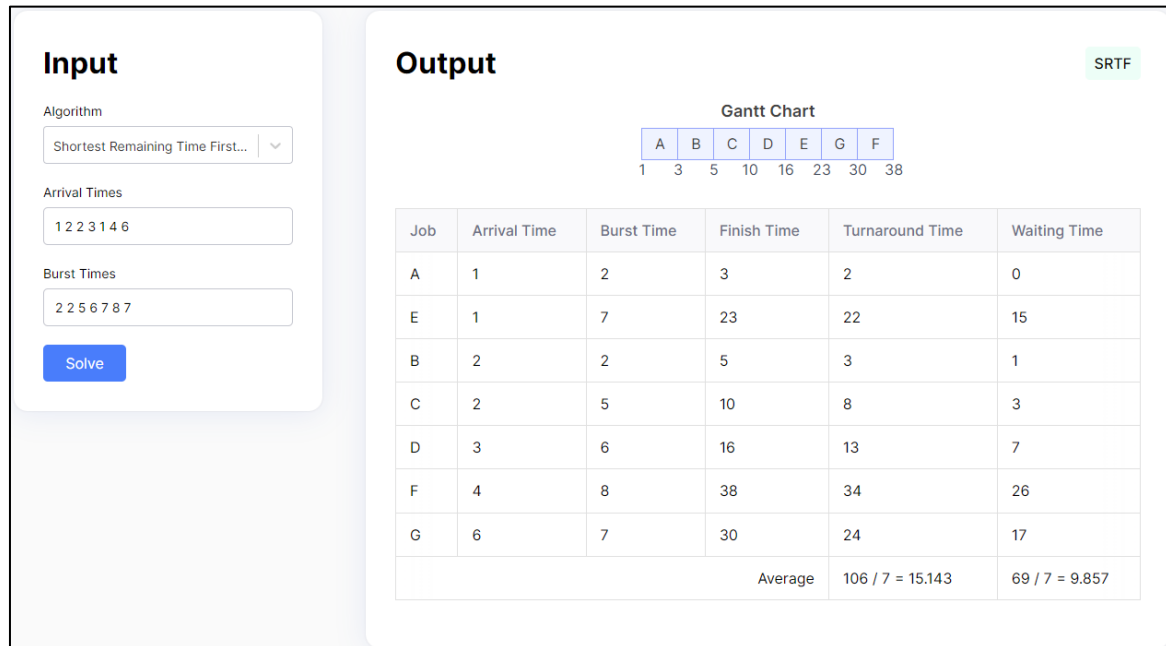


Figura 1. Ejemplo de ejecución del simulador de Boonsuen.

De forma sencilla permite añadir los tiempos pertenecientes a cada uno de los procesos que se desean simular, así como seleccionar el método de planificación deseado (*First Come First Service*, *Shortest Job First*, *Shortest Remaining Time First*, *Round-Robin* o *Priority*, y algunas variantes en versión *preemptive* o “preventiva”), como se puede observar en la [figura 1](#).

A continuación, se muestra una lista de las principales ventajas e inconvenientes o carencias identificados tras un análisis de la aplicación:

Ventajas:

- Buena portabilidad al poder utilizarse y visualizar la información correctamente desde cualquier dispositivo.
- Se destaca su sencillez y facilidad de uso.
- La página contiene enlaces de ayuda y soporte que redirige al usuario a manuales sencillos que explican el uso de la herramienta eficazmente.
- Posee una correcta variedad de algoritmos implementados.

Inconvenientes:

- La representación visual es algo limitada.
- No se observa que se pueda indicar la realización de operaciones de E/S.
- No se permite la modificación de datos ya introducidos.

4.1.2. AnimOS CPU-Scheduling

La Universidad Tecnológica de Dortmund ha desarrollado un simulador bastante completo en cuanto a visualización de resultados [11], pudiendo no sólo simular la organización de los procesos, sino que también permite comprobar paso a paso cómo afecta el método seleccionado a dicha organización (ver [figura 2](#)).

En cuanto a los algoritmos implementados, éstos únicamente se corresponden con FIFO, SJF y RR, por lo que se puede encontrar cierta limitación en este sentido.

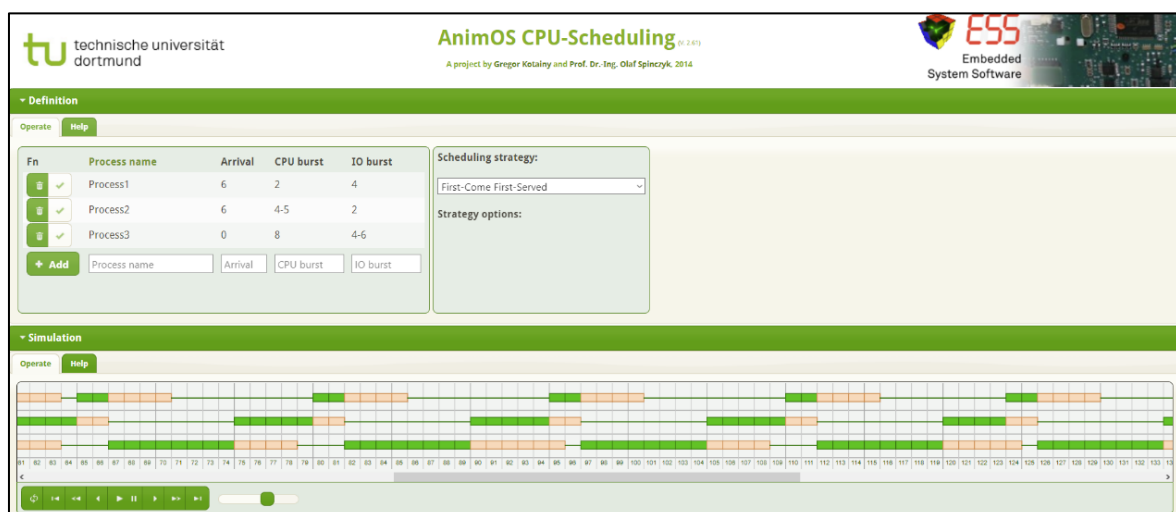


Figura 2. Ejemplo de ejecución del simulador de la Universidad de Dortmund.

La siguiente lista presenta las principales ventajas e inconvenientes de la aplicación identificadas:

Ventajas:

- La simplicidad y claridad en su diseño hacen de este simulador una herramienta muy usable.
- La visualización de resultados es completa y sencilla.

Inconvenientes:

- La legibilidad de los resultados se ve comprometida en dispositivos móviles con pantallas de menor tamaño.
- No presenta la implementación de una cantidad limitada de algoritmos.
- Carece de cualquier tipo de documentación disponible para el usuario.
- No se observa que se pueda indicar la realización de operaciones de E/S.

4.1.3. *CPU Scheduling Simulator by Hirusha Cooray*

Este sencillo simulador [\[6\]](#), hace uso de los cuatro principales algoritmos de planificación: FIFO (ver [figura 3](#)), SJF, SRTF y RR.

Tras su análisis, se identificaron las principales ventajas e inconvenientes de la aplicación que se presentarán a continuación:

Ventajas:

- Uno de los puntos fuertes de este simulador es su usabilidad y accesibilidad.
- Implementa los algoritmos más frecuentemente estudiados.
- La visualización de resultados es correcta independientemente del tamaño de pantalla del dispositivo.

Inconvenientes:

- La obtención de resultados es precaria y bastante simple.
- No presenta documentación alguna sobre su funcionamiento.
- No se observa que se pueda indicar la realización de operaciones de E/S.
- No se permite la modificación de datos ya introducidos.

Process ID	Arrival Time	Burst Time
1	2	4
2	1	2
3	1	3

Process ID	Arrival Time	Burst Time	Completed Time	Waiting Time	Turnaround Time
2	1	2	3	0	2
1	2	4	7	1	5
3	1	3	10	6	9

Select Scheduling Method

Average Turnaround Time

Average Waiting Time

Throughput

Figura 3. Ejemplo sencillo de ejecución del simulador.

4.2. Aplicación descargable para ordenador personal

A continuación, se considera todo software creado para ser descargado e instalarlo en cualquier computadora personal. De esta forma podrá ser ejecutado en dicho dispositivo sin necesidad de ningún navegador o una conexión a la red. Cabe destacar que, a la hora de realizar el estudio, se ha tenido en cuenta la independencia de los simuladores respecto al sistema operativo en el que se ejecutan, con el fin de evitar mayores limitaciones.

4.2.1. Alvareztech

Aplicación básica y sencilla (ver [figura 4](#)), desarrollada por el usuario “Alvareztech” [7] en Java, que realiza una pequeña simulación de procesos utilizando dos de los algoritmos mencionados anteriormente, FIFO y RR, además de uno extra el cual resulta ser algo menos frecuente de encontrar implementado, *Highest Response Ratio Next*.

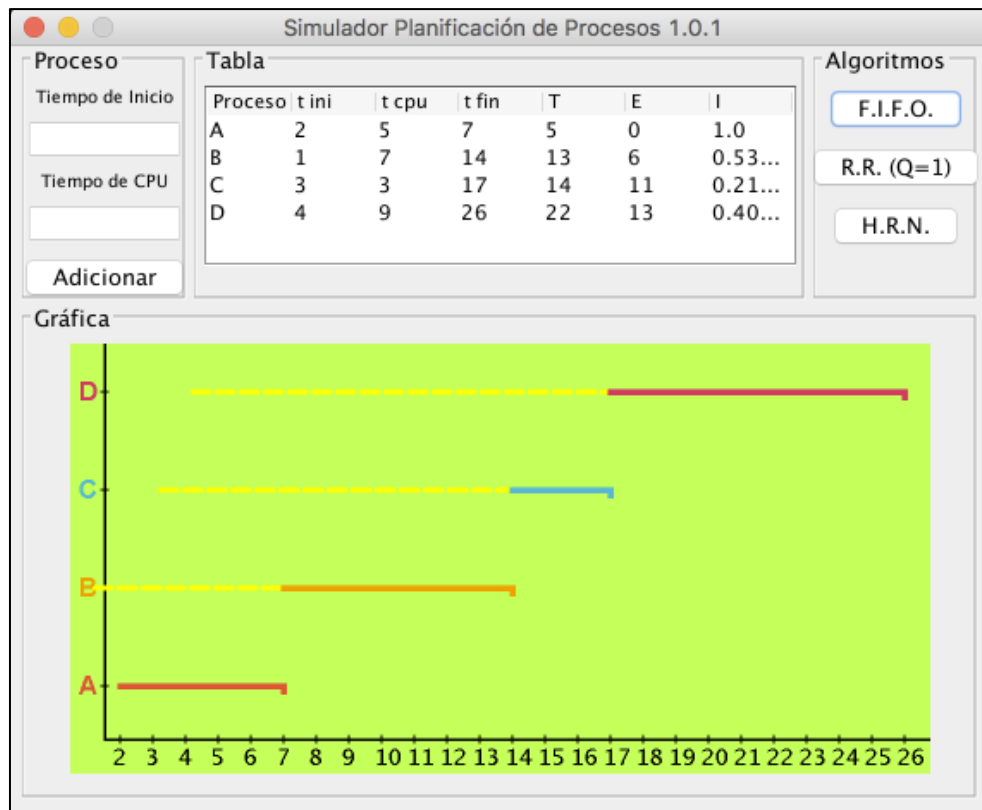


Figura 4. Ejemplo de ejecución del algoritmo FIFO.

Se muestra una lista con las principales ventajas e inconvenientes de la aplicación tras su análisis:

Ventajas:

- Es una herramienta sencilla, usable y fácil de instalar.
- Contiene gráficas que apoyan la visualización de los resultados.

Inconvenientes:

- Los algoritmos implementados son limitados.
- La documentación encontrada es pobre y no contiene desarrollo en el uso de la herramienta, así como en las funcionalidades implementadas.
- No se observa que se pueda indicar la realización de operaciones de E/S.

4.2.2. IncanatoIT

Para este caso, el autor en GitHub IncanatoIT [\[8\]](#) ha desarrollado, una aplicación de uso sencillo con la particularidad de la visualización de los tiempos de cada proceso en diferentes formatos de tablas, como se observa en la [figura 5](#) y [figura 6](#).

.. Planificación CPU-FCFS

PLANIFICACIÓN DE CPU - FIRST COME FIRST SERVED

T Llegada T CPU

Proceso 1 7 9

Proceso 2 5 6

Proceso 3 14 8

Proceso 4 0 5

Ejecutar

Procesos	T Llegada	T CPU	Estado	T Respuesta	T Espera	T Retorno
Proceso 4	0	5	Listo	5	0	5
Proceso 2	5	6	Listo	6	0	6
Proceso 1	7	9	Listo	13	4	13
Proceso 3	14	8	Listo	14	6	14
				9.0	2.0	9.0

Salir

Figura 5. Ingreso de datos para simular los procesos.

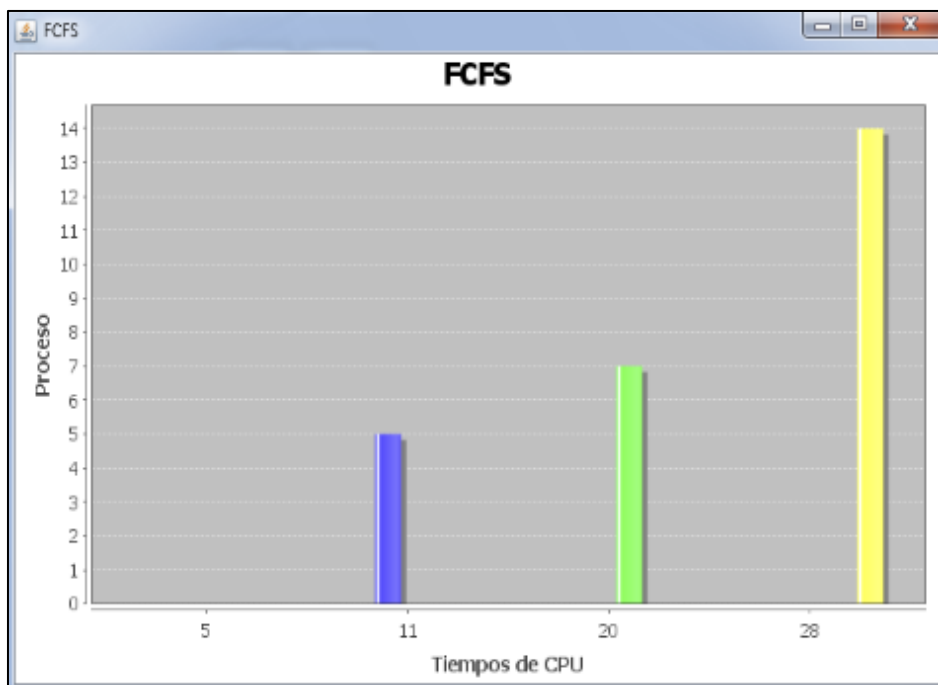


Figura 6. Gráfico resultante de la simulación.

En cuanto a los algoritmos implementados, se puede decir que el simulador hace uso de una variedad interesante de estos, aunque sacrifica la implementación de algunos más básicos en pro de agregar una mayor variedad. Estos se corresponden con los algoritmos FIFO, SJF y RR, así como su variable con prioridades.

A continuación, se detallan las principales ventajas e inconvenientes de la aplicación tras su análisis:

Ventajas:

- Es una herramienta sencilla.
- Se ha encontrado una extensa documentación sobre su funcionamiento junto a unos breves resúmenes de la metodología de cada algoritmo implementado.
- Representación de resultados tanto en tabla como en gráfica.

Inconvenientes:

- La cantidad de algoritmos presentes es algo reducida.
- La interfaz gráfica presenta serias deficiencias estéticas y de usabilidad.
- No se observa que se pueda indicar la realización de operaciones de E/S.

4.2.3. Ifreddyrondon

El desarrollador en GitHub “Ifreddyrondon” [9] ha creado uno de los simuladores más completo de los ejemplos correspondientes a este apartado, el cual implementa los cuatro algoritmos básicos: FIFO, SJF, SRTF y RR. Se dispone de una gráfica que muestra el estado de los procesos en tiempo real, así como las diferentes estadísticas relacionadas con los tiempos (ver [figura 7](#)).



Figura 7. Ejemplo de simulación y selección del algoritmo FIFO.

A continuación, se expone una lista con las principales ventajas e inconvenientes de la aplicación halladas tras su análisis:

Ventajas:

- Facilidad de acceso a la instalación.
- Atractiva y clara representación visual con una extensa información.
- Documentación clara y sencilla respecto al uso de la herramienta.
- Implementación de los algoritmos más frecuentes.

Inconvenientes:

- La usabilidad es algo tosca debido a la interfaz dividida en desplegables.
- No se observa que se pueda indicar la realización de operaciones de E/S.

4.3. Dispositivos Móviles

Como finalización de este capítulo se tratarán algunos ejemplos de simuladores para dispositivos móviles, permitiendo así poder realizar dicha simulación sin necesidad de un ordenador o tener que recurrir a una visualización (probablemente precaria) de simuladores en sitios web mediante el navegador de estos dispositivos. Se han seleccionado las dos aplicaciones más representativas disponibles en *Play Store* de Google.

4.3.1. CPU Simulator (CPU Scheduling)

La aplicación publicada por *Jeico Games* [\[10\]](#) es una aplicación de interfaz sencilla que permite seleccionar el algoritmo, el número de procesos y sus tiempos, tanto manualmente como directamente importando un fichero con estos datos (ver [figura 8](#)), algo ciertamente interesante. Con ello, es capaz de mostrar gráficamente el comportamiento del algoritmo para cada proceso y cómo esto afecta a la CPU en cuestión.

Como característica a destacar, se puede observar que la flexibilidad presente en este simulador rivaliza con el analizado en el apartado anterior, ya que dispone de la implementación de numerosos algoritmos (FIFO, SJF, SRTF, RR, *Priority* y HRRN), considerando esto una clara ventaja respecto a sus iguales, e indicando que no es algo exclusivo de la plataforma anterior.

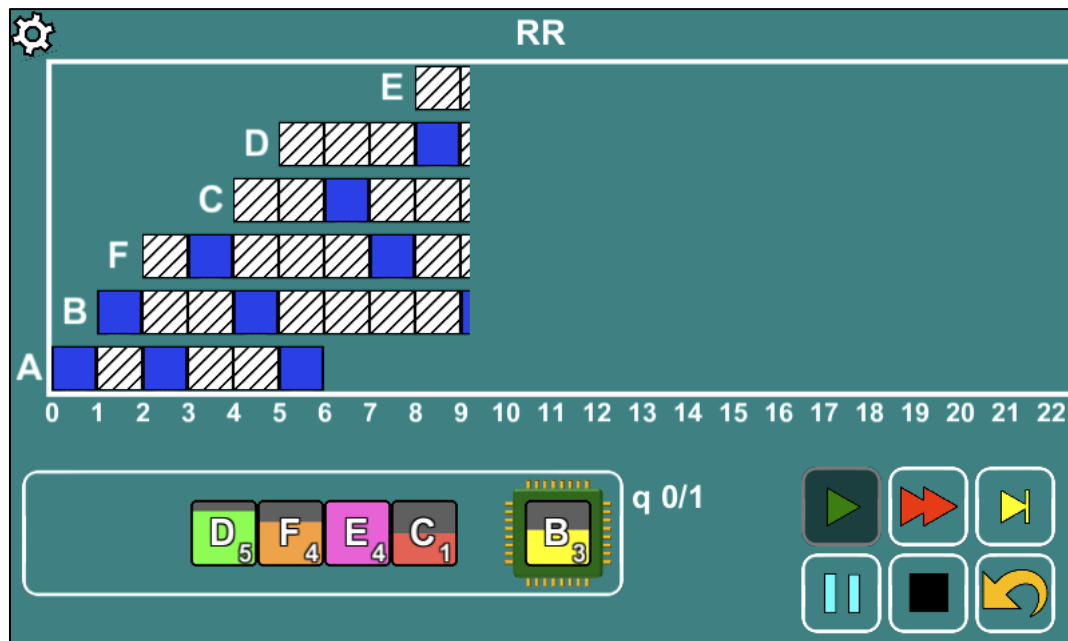


Figura 8. Ejemplo de simulación del algoritmo RR.

A continuación, tras la realización del análisis correspondiente, se detallan las principales ventajas e inconvenientes del simulador:

Ventajas:

- Facilidad de instalación.
- Gran número de algoritmos implementados.
- Exportación / Importación de resultados.

Inconvenientes:

- Representación visual algo confusa debido a la variedad de colores e iconos.
- Falta de documentación o explicación del funcionamiento completo.
- No se observa que se pueda indicar la realización de operaciones de E/S.

4.3.2. Quantum

Esta aplicación [111] (todavía en versiones de prueba) muestra una estética mucho más minimalista y elegante a la anterior (ver [figura 9](#) y [figura 10](#)), haciendo buen uso de la sencillez para tener un uso intuitivo. Se encuentra un simulador que genera un diagrama de estructuras y un gráfico de ejecución de procesos con los datos introducidos.

Los algoritmos implementados en este caso corresponden con aquellos que más frecuencia se han encontrado en estas herramientas, siendo estos: FIFO, SJF, SRTF y RR.

Quantum

Agregar proceso

Llegada CPU Bloqueos Prioridad

+

Proceso

Llegada

CPU

Figura 9. Selección de datos iniciales.

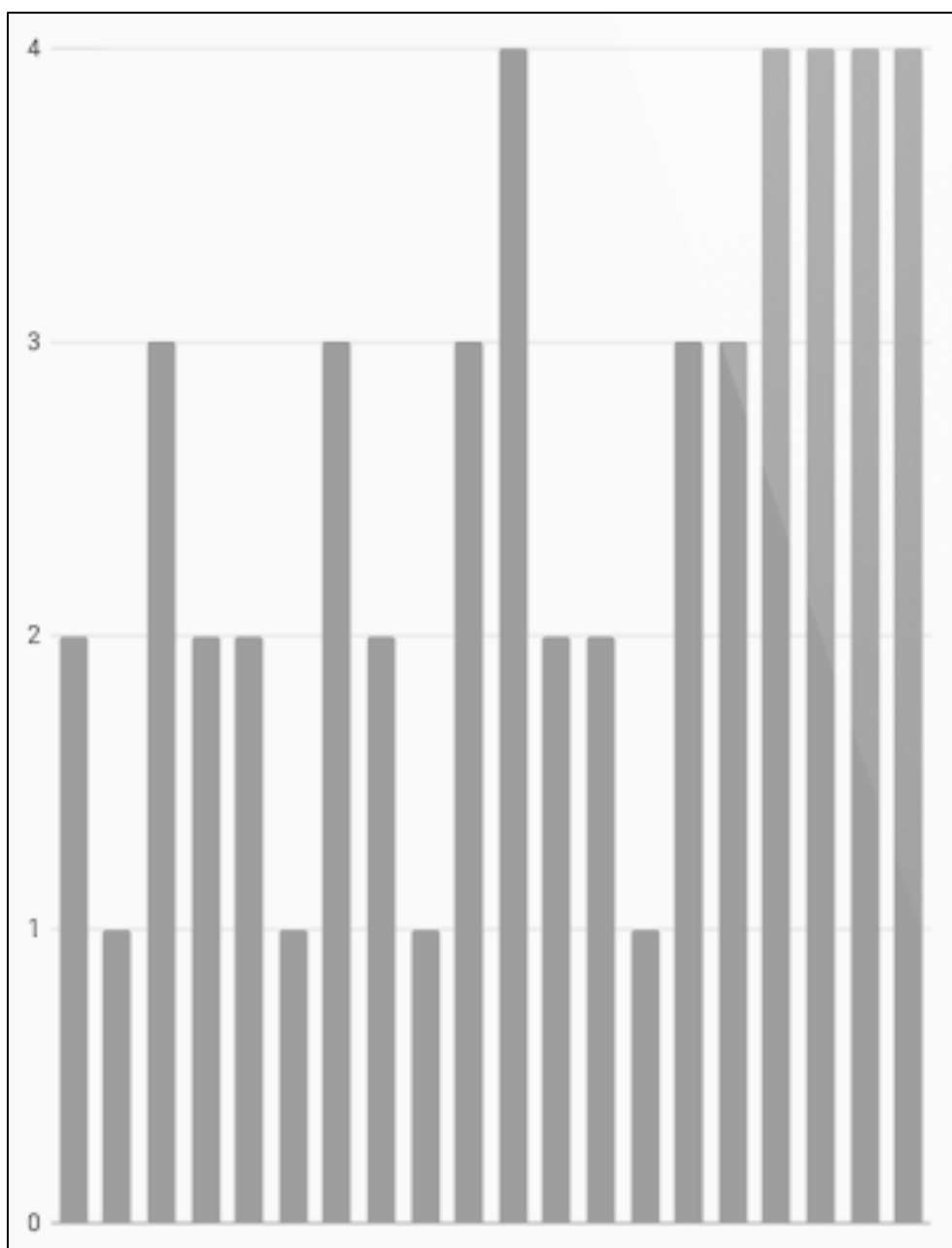


Figura 10. Resultado en gráfico de barras.

La siguiente lista presenta las principales ventajas e inconvenientes de la aplicación identificadas tras un análisis:

Ventajas:

- Descarga e instalaciones sencillas.
- Gran facilidad de uso.
- Representación visual sobria y sencilla.
- El conjunto de algoritmos implementados se conforma por los más frecuentes.
- Visualización de resultados en tabla y gráfica.

Inconvenientes:

- Falta de documentación clara del funcionamiento.
- No se observa que se pueda indicar la realización de operaciones de E/S.
- No permite modificación de datos ya introducidos.

4.4. Conclusiones

En última instancia, el proceso de explorar antecedentes ha evolucionado de un mero ejercicio teórico a un objetivo concreto a cumplir en el marco de este proyecto. Al comparar las soluciones existentes con la propuesta en desarrollo, se han identificado aspectos clave que marcan la diferencia y mejoran significativamente la experiencia y utilidad de la aplicación.

Al analizar los antecedentes, se ha destacado la importancia de no solo abordar la implementación técnica de algoritmos de planificación, sino también enfocarse en crear una aplicación atractiva, de alta usabilidad y accesibilidad, con un enfoque didáctico intrínseco. La adaptación de los conceptos teóricos al entorno práctico se ha convertido en un objetivo primordial, permitiendo una interacción significativa y efectiva con los usuarios.

En relación con los rasgos característicos observados en los antecedentes, se ha encontrado una oportunidad clara para marcar una diferencia. La consideración de estados y eventos de entrada/salida en la simulación se alinea directamente con los contenidos de la asignatura de Sistemas Operativos, enriqueciendo el valor educativo de la aplicación. Además, se ha identificado que la adaptabilidad de algunos simuladores web a diferentes tamaños de pantalla es limitada, lo que resalta la importancia de optimizar la presentación en dispositivos móviles.

Una característica adicional que se ha identificado como valiosa y que no todos los simuladores han considerado es la posibilidad de modificar los datos introducidos sin necesidad de eliminar la totalidad de los datos y comenzar desde cero. Este enfoque más flexible permitirá a los usuarios ajustar y refinar los parámetros de manera eficiente, optimizando la experiencia de simulación.

Por tanto, la materialización de un simulador para la asignatura de Sistemas Operativos que no solo aborde los conceptos teóricos, sino que también incorpore elementos prácticos como estados y eventos de E/S, y facilite la modificación ágil de datos, se convierte en una herramienta de apoyo esencial. La implementación en dispositivos Android, además, abre la puerta a su utilización responsable durante las clases teóricas, generando un entorno interactivo y enriquecedor para el aprendizaje.

Estas diferencias distintivas no solo enriquecerán la experiencia del usuario, sino que también elevarán la propuesta por encima de los antecedentes examinados.

Capítulo

5. Restricciones

En el análisis y planificación del proyecto, resulta fundamental identificar y abordar las restricciones que pueden influir en su desarrollo y alcance. Este apartado abordará detalladamente dos aspectos clave: los Factores de Datos y los Factores Estratégicos. Cada uno de estos subapartados explorará las limitaciones y consideraciones que guiarán la toma de decisiones en la ejecución del proyecto.

5.1. Factores Dato

Este subapartado se adentra en las limitaciones fundamentales que influyen directamente en el proyecto en su totalidad. Estas restricciones no modificables tienen un impacto significativo en la planificación y ejecución del proyecto. El análisis exhaustivo de estos factores permitirá un enfoque claro y estratégico para la consecución de los objetivos, considerando cuidadosamente las circunstancias y restricciones inherentes al proyecto en su conjunto.

1. Limitaciones de Personal:

La disponibilidad y capacidad del equipo humano asignado al proyecto son determinantes en su desarrollo. En este caso, se dispondrá únicamente del apoyo del director del proyecto, Juan Carlos Fernández Caballero.

2. Limitaciones de Tiempo:

El tiempo disponible para completar el proyecto es un recurso valioso y limitado. La gestión efectiva de los plazos es esencial para mantener el proyecto en curso y garantizar una implementación exitosa.

Este proyecto tiene una cuantía de horas limitadas para su desarrollo, según se establece en el reglamento de los Trabajos de Fin de Grado de la Escuela Politécnica Superior de Córdoba, por lo que es un elemento importante a tener en cuenta para el desarrollo.

3. Limitaciones de *Hardware*:

Las restricciones en términos de recursos de *hardware* pueden afectar la eficiencia y el rendimiento del proyecto. Evaluar la capacidad de los dispositivos y equipos utilizados es crucial para optimizar la funcionalidad de la aplicación.

Se dispondrán de los recursos aportados por el proyectista, especificados más adelante en el apartado [Recursos Hardware](#).

4. Limitaciones de *Software*:

Las limitaciones de las herramientas y plataformas de *software* utilizadas pueden influir en el desarrollo y funcionalidad del proyecto. Debido a ello y considerando la naturaleza principal del proyecto, se hará uso de *software* libre [18]. Este punto también se detallará en el apartado de [Recursos Software](#).

5. Limitaciones de Rendimiento:

El rendimiento de la aplicación es esencial para brindar una experiencia de usuario óptima. Considerar las limitaciones de rendimiento, como la capacidad de respuesta y velocidad, es fundamental para la satisfacción del usuario final.

Dado el propósito de desarrollar una herramienta que sea accesible desde una amplia gama de dispositivos Android, es imperativo optimizar la aplicación para garantizar una experiencia de usuario fluida y satisfactoria.

5.2. Factores Estratégicos

En este subapartado, se exploran las limitaciones y consideraciones estratégicas que pueden afectar la planificación y ejecución del proyecto en su conjunto. Se trata de evaluar aspectos como los plazos de desarrollo, la asignación de recursos humanos y la integración de diferentes etapas del proceso. Al analizar estos factores, se garantiza una gestión eficaz del proyecto y se minimizan los riesgos potenciales. El entendimiento completo de las limitaciones estratégicas permite una toma de decisiones informada y la implementación exitosa de la aplicación, alineándola con los objetivos y expectativas establecidos.

1. Sistema Operativo:

La elección del sistema operativo influye en la compatibilidad y funcionalidad de la aplicación. La elección se ha inclinado hacia el sistema Android, dada su fácil accesibilidad para la mayoría de los estudiantes y docentes, además de su portabilidad y versatilidad al

no requerir un ordenador de sobremesa o portátil, recurso que no siempre se encuentra a disposición de todos. Esta decisión se ha tomado considerando, también, las limitaciones que supone optar por una página web, que conllevaría el mantenimiento de un dominio y la creación de un diseño adaptable a dispositivos de sobremesa y móviles.

2. Versión de Android:

La versión específica de Android determina las características y capacidades disponibles. Se ha considerado los porcentajes de uso de una de las versiones disponibles proporcionadas oficialmente por los desarrolladores, optando por una versión que se encuentre en un término medio entre el porcentaje de uso y su antigüedad.

3. IDE (Entorno Integrado de Desarrollo):

El IDE utilizado impacta en la eficiencia y calidad del desarrollo.

4. Lenguaje de Programación:

La elección del lenguaje de programación influye en la estructura y funcionalidad del proyecto. Se optará por un lenguaje que potencie la eficacia del desarrollo y el rendimiento de la aplicación.

Dado que se seguirán las nuevas prácticas para el desarrollo de aplicaciones Android, se ha seleccionado el lenguaje Kotlin debido a su gran potencia, soporte y comunidad, lo que implica una excelente ventaja a la hora del desarrollo.

5. Procesador de Textos para el Documento:

El procesador de textos empleado en la documentación es esencial para una presentación clara y profesional.

Se han considerado numerosas opciones que permitan aportar estas características y, finalmente, se ha optado por *Microsoft Word* debido a su potencia y que, pese a ser una herramienta de uso bajo una licencia de pago, la universidad proporciona una licencia oficial a todos sus estudiantes.

6. Control de Versiones con Git:

El control de versiones con Git [\[19\]](#) y GitHub [\[20\]](#) es crucial para el desarrollo colaborativo y la gestión del código fuente. Se implementará para asegurar un seguimiento y manejo efectivo de los cambios en el proyecto.

Capítulo

6. Recursos

En esta sección, se detallan los recursos esenciales que respaldarán la ejecución exitosa del proyecto. Tanto los recursos humanos como los materiales desempeñarán un papel fundamental en la materialización de la propuesta, asegurando la consecución de los objetivos planteados con eficacia y calidad.

6.1. Recursos Humanos

- **Autor:** Julen Pérez Hernández. Alumno de Computación en el Grado en Ingeniería Informática de la Universidad de Córdoba.
- **Director:** Juan Carlos Fernández Caballero. Profesor Titular de la Universidad de Córdoba del Dpto. de Informática y Análisis Numérico y miembro investigador del grupo AYRNA.

6.2. Recursos Materiales

A continuación, se detallan los recursos materiales esenciales que se requerirán para llevar a cabo el desarrollo de la aplicación propuesta.

6.2.1. Recursos *Hardware*

En el caso del proyecto que se presenta, el uso de un ordenador con capacidad suficiente será esencial para la ejecución de las tareas de desarrollo y pruebas de la aplicación, siendo las siguientes las características del ordenador de sobremesa personal a utilizar para el desarrollo:

- **Procesador:** Intel i5-12600, 6 núcleos de alto rendimiento (12 hilos) a 3.30 GHz.
- **Memoria RAM:** 16 GB DDR4 a 3600 MHz.

- **Tarjeta Gráfica:** Nvidia Gforce RTX 3060 12 GB.
- **Almacenamiento:** 1 TB SSD M.2 NVMe a 3500 MB/s + 1 TB HDD.

Además, se hará uso de un dispositivo móvil con sistema operativo Android, ya sea un teléfono o una *tablet*, para realizar algunas de las pruebas correspondientes en la aplicación.

6.2.2. Recursos *Software*

- **Sistema Operativo:** Windows 11.
- **Lenguaje de Programación:** Kotlin.
- **Librerías y *frameworks*:** Se utilizarán todas aquellas librerías que permitan simplificar, facilitar y modularizar el proyecto todo lo posible. Principalmente Jetpack Compose como librería principal en el desarrollo de interfaces.
- **Entorno de desarrollo (IDE) y editores:** *Android Studio*.
- **Documentación:** Se utilizará la herramienta Microsoft Word.

Bloque

II. Análisis del sistema y especificaciones de requisitos

Capítulo

7. Especificación de requisitos

En esta sección, se establecerán las especificaciones de requisitos que servirán como la piedra angular del proceso de desarrollo. Se detallarán de manera precisa y clara los objetivos funcionales y no funcionales que la aplicación deberá cumplir. Estas especificaciones no solo guiarán la implementación técnica, sino que también asegurarán que la aplicación final satisfaga las expectativas y necesidades identificadas en las etapas anteriores del proyecto.

7.1. Requisitos de Información

Sección en la que se establecen los criterios y necesidades relacionadas con la información que la aplicación debe manejar o presentar. Estos requisitos pueden abarcar desde la forma en que se capturan y almacenan los datos, hasta cómo se presentan al usuario final. En esencia, se trata de definir qué tipo de información se requiere para el correcto funcionamiento de la aplicación y cómo debe ser gestionada para cumplir con los objetivos y funcionalidades previamente establecidos.

En el contexto de este proyecto de desarrollo de una aplicación para simular algoritmos de planificación de procesos en dispositivos Android, se incluirán los siguientes:

1. **Datos de Entrada para Simulación:** El usuario tendrá todos los campos necesarios para cada algoritmo disponibles con el fin de introducir todos los datos correspondientes para la simulación, como los tiempos de llegada, duración, quantum, etc.

2. **Resultados de Simulación:** Se presentarán los resultados obtenidos derivados de la simulación. El usuario podrá observar los tiempos de inicio, finalización, estancia o espera, entre otros.
3. **Formato de Presentación:** Tanto los resultados como el listado de elementos datos introducidos se representarán en tablas o diagramas que simplifiquen y faciliten la visualización y entendimiento de toda la información resultante.
4. **Configuración de Parámetros:** A la hora de introducir los datos correspondientes a cada proceso, el usuario tendrá siempre disponible la opción de agregar a dicho proceso un evento de E/S, especificando el momento en el que se deberá producir cada una de estas acciones. Adicionalmente siempre se le permitirá eliminar por completo la información introducida o únicamente uno de los procesos, permitiendo así poder modificar el listado de elementos en función a sus necesidades.
5. **Documentación de Ayuda:** El usuario siempre tendrá acceso a los manuales correspondientes en caso de necesidad de una mayor orientación y guía a través de la aplicación.
6. **Gestión de Errores y Advertencias:** Cada campo donde el usuario pueda ingresar datos estará restringido al formato correspondiente al tipo de dato requerido, reduciendo la posibilidad de introducir datos incorrectos. Al eliminar información, se solicitará una confirmación adicional al usuario. De manera similar, para salir de la aplicación, se exigirá presionar dos veces seguidas el botón de retroceso, añadiendo un nivel de seguridad para evitar cierres accidentales.
7. **Experiencia de Usuario:** Cada aspecto relacionado con la presentación de la interfaz será cuidadosamente abordado para brindar una experiencia de usuario amigable, actualizada, compacta, accesible, modular y de fácil comprensión.

7.2. Requisitos Funcionales

En esta sección, se establecerán los requisitos funcionales que describen las características esenciales y funcionalidades que la aplicación debe poseer. Estos requisitos detallados definen las acciones específicas que la aplicación debe ser capaz de llevar a cabo para cumplir con su propósito principal. Cada requisito se enfoca en la funcionalidad clave que se espera de la aplicación y se especificará de manera precisa para guiar el desarrollo técnico y asegurar que la aplicación logre sus objetivos de manera efectiva.

1. **Selección de Algoritmo:** Los usuarios deben poder elegir entre diferentes algoritmos de planificación para simular.
2. **Ingreso de Datos:** La aplicación debe permitir a los usuarios ingresar los datos relevantes para la simulación, como tiempos de llegada y duración de procesos, tiempo de entrada y salida en caso de haberlos, quantum, etc.
3. **Simulación de Procesos:** La aplicación debe ejecutar la simulación de acuerdo con el algoritmo seleccionado y los datos ingresados. Se pulsará el botón de “Siguiente” tras haber introducido todos los datos deseados para poder ejecutar la simulación.
4. **Visualización de Resultados:** Los resultados de la simulación, como tiempos de inicio y finalización, deben ser presentados de manera clara y comprensible mediante las tablas o diagramas que se consideren necesarias.
5. **Personalización de Parámetros:** Los usuarios deben poder ajustar parámetros de simulación, como la inclusión de eventos de E/S o el quantum.
6. **Gestión de Procesos Agregados:** La aplicación debe permitir la adición y eliminación de procesos para considerar distintos escenarios de simulación.
7. **Visualización de Diagramas de Tiempo:** Los diagramas de Gantt o de tiempo de los procesos deben ser generados y presentados de formas claras y sencillas para una correcta visualización.
8. **Interfaz Intuitiva:** La interfaz de usuario debe ser fácil de usar y navegar, incluso para aquellos que no tienen experiencia técnica.

9. Compatibilidad con Múltiples Dispositivos: La aplicación debe funcionar correctamente en una variedad de dispositivos Android con diferentes tamaños de pantalla y resoluciones.

10. Notificaciones de Error: Informar a los usuarios si hay errores en los datos ingresados o si ocurren problemas durante la simulación.

7.3. Requisitos No Funcionales

En esta sección, se establecerán los requisitos no funcionales que definen los parámetros de calidad y características globales que la aplicación debe cumplir. Estos requisitos se centran en aspectos que no son directamente funcionalidades, pero que son esenciales para garantizar la eficiencia, seguridad, usabilidad y satisfacción general del usuario. Cada requisito no funcional detalla una expectativa específica con respecto a la aplicación y se describirá de manera precisa para orientar su diseño y desarrollo.

1. Usabilidad:

- ✚ La aplicación debe ser intuitiva y fácil de usar, con una curva de aprendizaje no mayor a 15 minutos para usuarios con conocimientos básicos de tecnología.
- ✚ Los elementos de la interfaz deben ser de tamaño adecuado para su interacción en pantallas táctiles, garantizando una experiencia cómoda para usuarios con diferentes tamaños de dedos.

2. Rendimiento:

- ✚ La aplicación debe ser capaz de manejar la simulación de algoritmos de planificación de procesos con un tiempo de respuesta promedio inferior a 1 segundo, incluso con un alto número de procesos simulados.
- ✚ La velocidad de actualización de los resultados y gráficos durante la simulación no debe generar demoras perceptibles por parte del usuario.

3. Seguridad:

- ✚ Los datos ingresados por los usuarios durante la simulación deben ser almacenados y manejados de manera segura.

4. Compatibilidad:

- ✚ La aplicación debe ser compatible con una amplia variedad de dispositivos Android, abarcando diferentes versiones de sistema operativo y tamaños de pantalla.

5. Accesibilidad:

- ✚ La aplicación debe cumplir con pautas de accesibilidad WCAG [\[21\]](#) para garantizar que usuarios con discapacidades visuales o motoras puedan interactuar y utilizar la aplicación de manera efectiva.

6. Mantenibilidad:

- ✚ El código de la aplicación debe estar bien estructurado y documentado, permitiendo a futuros desarrolladores comprender y mantener el proyecto de manera eficiente.
- ✚ La aplicación debe estar diseñada de manera modular, facilitando la incorporación de nuevos algoritmos de planificación en el futuro sin necesidad de reescribir gran parte del código existente.

7. Rendimiento en Distintos Dispositivos:

- ✚ La aplicación debe ofrecer un rendimiento consistente en dispositivos de diferentes capacidades y especificaciones, evitando retrasos significativos en dispositivos con recursos limitados.

8. Disponibilidad:

- ✚ La aplicación debe estar disponible para su descarga y uso desde la Play Store de Android de manera continua y sin interrupciones.

9. Actualizaciones y Mantenimiento:

- ✚ La aplicación debe permitir la actualización de nuevos algoritmos de planificación y mejoras de manera sencilla, sin requerir reinstalación completa por parte del usuario.
- ✚ Las actualizaciones de la aplicación deben ser notificadas de manera clara al usuario y deben ser descargadas e instaladas sin inconvenientes.

Capítulo

8. Análisis funcional

En esta sección, se llevará a cabo un análisis funcional utilizando el enfoque del UML [22] (Lenguaje de Modelado Unificado) para definir los diferentes aspectos de la interacción del sistema. Se identificarán los usuarios del sistema, se describirán los casos de uso de contexto que representan las interacciones generales y se detallarán los casos de uso específicos que describen las funcionalidades específicas que la aplicación proporcionará. Esta metodología de modelado permitirá una comprensión más completa de cómo los diferentes elementos del sistema interactúan entre sí, sentando las bases para un diseño y desarrollo coherente y efectivo de la aplicación.

A continuación, se presenta la [Tabla 1](#), diseñada para servir como referencia en la representación de los casos de uso en el marco de este proyecto. Asimismo, se ofrece una descripción detallada de los campos que componen esta tabla y cómo se aplicarán en el análisis funcional del sistema.

Campo	Descripción
Caso de uso	Breve descripción del caso de uso.
Identificador	Identificación única del caso de uso.
Descripción	Descripción informal de los objetivos del caso de uso.
Actores	Actores involucrados en la ejecución del caso de uso.
Precondiciones	Condiciones necesarias para que el caso de uso pueda ocurrir.
Flujo principal	Pasos secuenciales que constituyen el flujo normal.
Flujos alternativos	Pasos alternativos que pueden surgir.
Postcondiciones	Estado en que se encuentra el sistema tras el caso de uso.

Tabla 1. Plantilla de Caso de Uso

Este apartado seguirá una estructura organizada. Se comenzará por la presentación y definición de los actores que interactúan con la aplicación. A continuación, se expondrá el caso de uso de contexto, el cual abarca una visión general del sistema junto con los actores involucrados.

Finalmente, se procederá a describir un refinamiento de este caso de uso para cada uno de los módulos funcionales que componen la aplicación. Este enfoque estructurado facilitará una comprensión clara de las interacciones del sistema y permitirá su análisis detallado.

8.1. Identificación de los usuarios objetivo de la app

Los usuarios a los que se dirige la aplicación son estudiantes del Grado de Ingeniería Informática en la Universidad de Córdoba que cursan la asignatura de Sistemas Operativos. Sin embargo, la aplicación estará accesible para cualquier usuario en general, incluyendo tanto estudiantes como docentes, a través de la *Play Store* de Google.

8.2. Casos de uso de contexto

A continuación, se expone el caso de uso de contexto, un caso de uso con la funcionalidad más genérica, correspondiente con el inicio de la aplicación, del cual derivan el resto de los casos de uso. Su especificación se muestra en la [Tabla 2](#).

Campo	Descripción
Caso de uso	Iniciar la aplicación.
Identificador	CUC-00.
Descripción	Acceso a la aplicación. Abre la pantalla de inicio.
Actores	Usuario.
Precondiciones	La aplicación se encuentra instalada en el dispositivo.
Flujo principal	El usuario abre la aplicación. Se inicia la pantalla de carga y se carga la pantalla de inicio.
Flujos alternativos	-
Postcondiciones	La pantalla de inicio ha sido correctamente inicializada.

Tabla 2. CUC-00. Caso de uso de contexto.

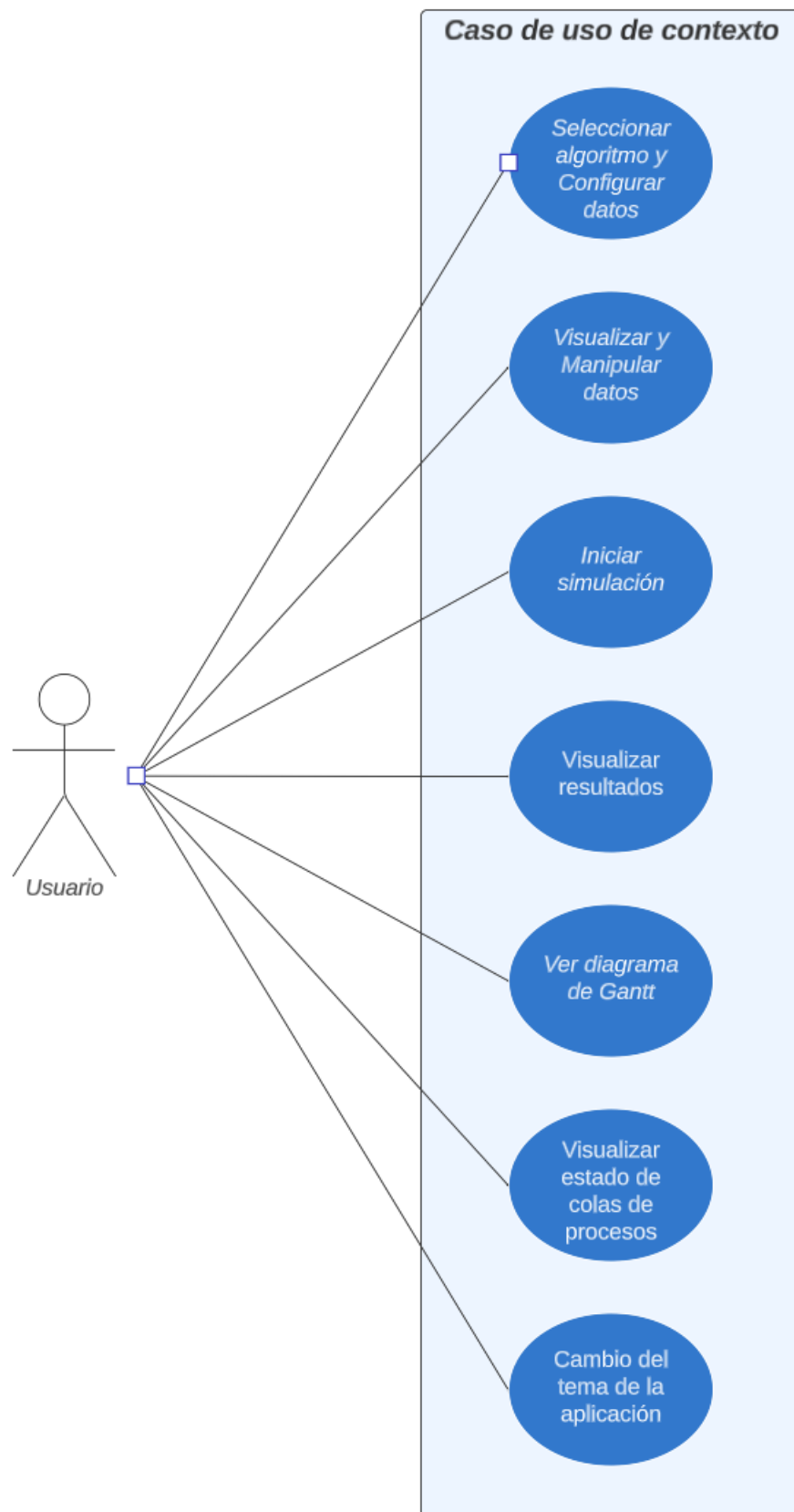


Figura 11. CUC-00.

8.3. Casos de uso

En esta sección, se abordarán los casos de uso específicos que definen las interacciones particulares entre los usuarios y la aplicación. Estos casos de uso detallan las funcionalidades y operaciones específicas que los usuarios pueden llevar a cabo dentro de la aplicación. Cada caso de uso ofrece una perspectiva detallada de cómo los actores interactúan con el sistema para lograr objetivos específicos.

A continuación, se presentarán los casos de uso individuales que componen esta etapa del análisis funcional, brindando una visión detallada de la operación y los resultados esperados en diferentes escenarios de uso.

8.3.1. Seleccionar Algoritmo y Configurar Datos

Campo	Descripción
Caso de uso	Seleccionar Algoritmo y Configurar Datos.
Identificador	CU-01.
Descripción	El usuario inicia la aplicación y accede a la pantalla de inicio. Aquí, puede seleccionar el algoritmo deseado, introducir los datos requeridos para su ejecución y decidir si se desea agregar un evento de E/S.
Actores	Usuario.
Precondiciones	La aplicación se ha iniciado correctamente.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario elige el algoritmo de planificación. 2. El usuario introduce los datos en los campos requeridos. 3. Si es necesario, el usuario agrega un evento de E/S. 4. El usuario agrega el proceso con los datos proporcionados.
Flujos alternativos	El usuario puede cambiar de opinión y seleccionar otro algoritmo o modificar los datos a su gusto.
Postcondiciones	Los datos del proceso se almacenan, se muestran en una tabla y se limpian los campos para poder agregar más procesos.

Tabla 3. CU-01. Selección del algoritmo e Introducción de datos.

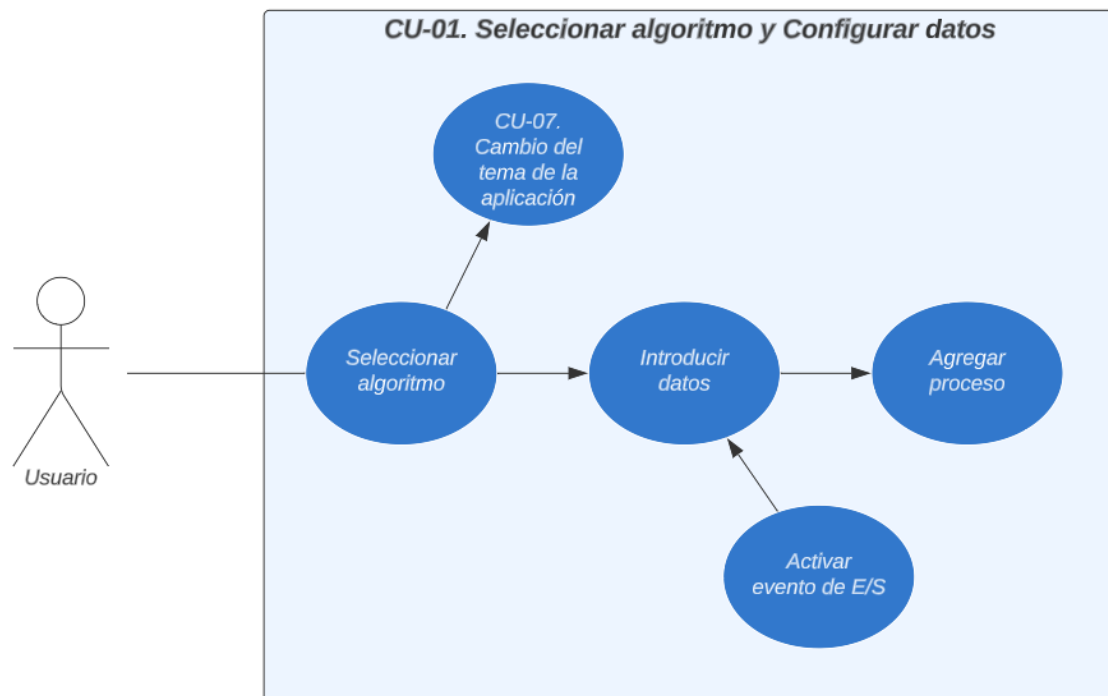


Figura 12. CU-01.

8.3.2. Visualizar y Modificar Datos

Campo	Descripción
Caso de uso	Visualizar y Modificar los Datos.
Identificador	CU-02.
Descripción	El usuario puede ver los datos introducidos en forma de tabla y realizar acciones como eliminar procesos individuales o todos los procesos agregados.
Actores	Usuario.
Precondiciones	Se han introducido al menos un proceso en la pantalla de inicio.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario visualiza la tabla de datos. 2. El usuario selecciona un proceso para eliminarlo. <ol style="list-style-type: none"> a. El usuario confirma la eliminación. b. El usuario cancela la eliminación. 3. El usuario elimina todos los procesos agregados. <ol style="list-style-type: none"> a. El usuario confirma la eliminación. b. El usuario cancela la eliminación.
Flujos alternativos	El usuario puede optar por no eliminar ningún proceso.
Postcondiciones	Los procesos seleccionados se eliminan y la tabla de datos se actualiza.

Tabla 4. CU-02. Visualizar y Modificar los datos.

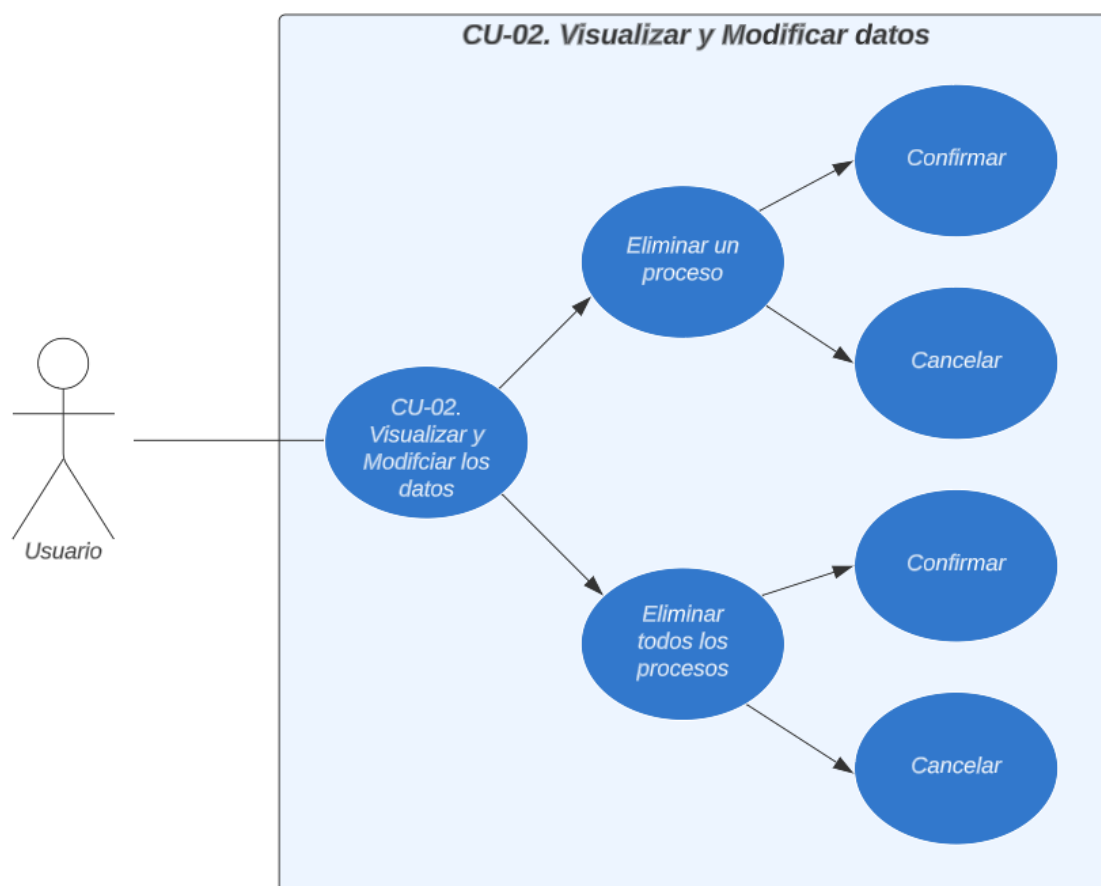


Figura 13. CU-02.

8.3.3. Iniciar Simulación

Campo	Descripción
Caso de uso	Iniciar Simulación.
Identificador	CU-03.
Descripción	El usuario avanza a la siguiente etapa de la aplicación al pulsar el botón "Siguiente".
Actores	Usuario.
Precondiciones	Se ha introducido al menos un proceso y se ha avanzado a la siguiente página.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario ha introducido al menos un proceso con el que trabajar. 2. Se pulsa el botón "Siguiente".
Flujos alternativos	-
Postcondiciones	La simulación se inicia y se avanza a la página de resultados.

Tabla 5. CU-03. Iniciar simulación.

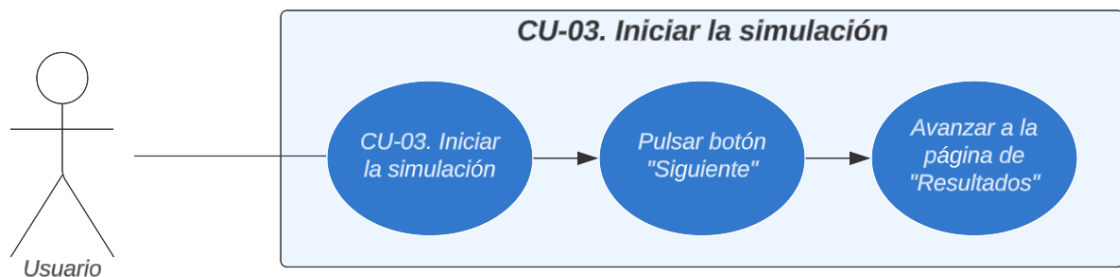


Figura 14. CU-03.

8.3.4. Visualizar Resultados

Campo	Descripción
Caso de uso	Visualizar Resultados.
Identificador	CU-04.
Descripción	El usuario accede a la página de resultados donde se muestra una tabla con los tiempos obtenidos por la simulación del algoritmo seleccionado.
Actores	Usuario.
Precondiciones	La simulación ha sido iniciada y se ha avanzado a la página de resultados.
Flujo principal	1. El usuario visualiza la tabla de resultados con los tiempos obtenidos por la simulación.
Flujos alternativos	-
Postcondiciones	El usuario obtiene una visión detallada de los resultados de la simulación en la tabla.

Tabla 6. CU-04. Visualizar los resultados.

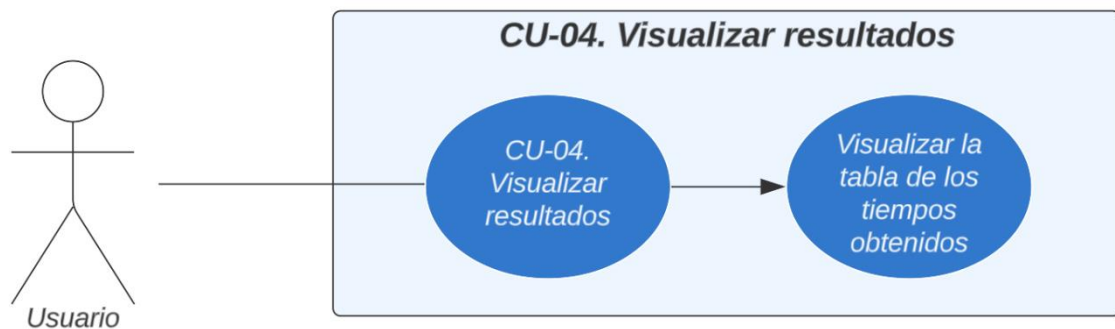


Figura 15. CU-04.

8.3.5. Ver Diagrama de Gantt

Campo	Descripción
Caso de uso	Ver Diagrama de Gantt.
Identificador	CU-05.
Descripción	El usuario accede a la página que muestra una tabla en forma de diagrama de Gantt que representa los tiempos obtenidos y los estados en los que se encuentra cada proceso en momentos determinados de la simulación.
Actores	Usuario.
Precondiciones	La simulación ha sido iniciada y se ha avanzado a la página del diagrama de Gantt.
Flujo principal	1. El usuario visualiza el diagrama de Gantt con la representación gráfica de los tiempos y estados de los procesos.
Flujos alternativos	-
Postcondiciones	El usuario obtiene una representación visual clara de la evolución de los procesos a lo largo del tiempo.

Tabla 7. CU-05. Visualizar el diagrama de Gantt.

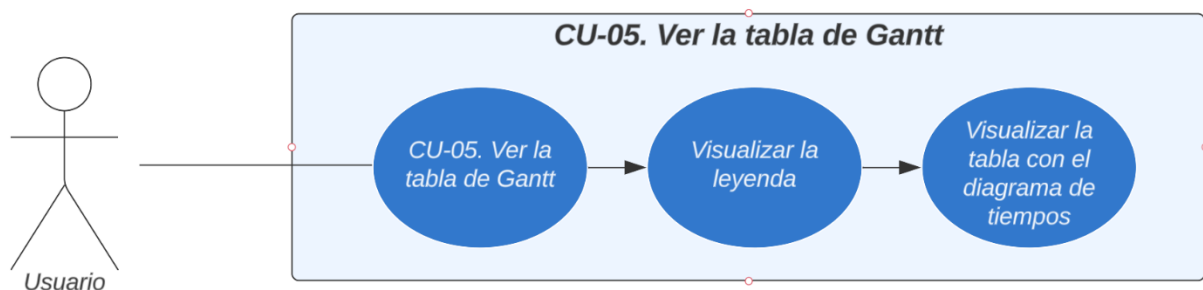


Figura 16. CU-05.

8.3.6. Visualizar Estado de Colas de Procesos

Campo	Descripción
Caso de uso	Visualizar Estado de Colas de Procesos.
Identificador	CU-06.
Descripción	El usuario accede a la página que muestra en tablas el estado de las colas de procesos en diferentes momentos de la simulación.
Actores	Usuario.
Precondiciones	La simulación ha sido iniciada y se ha avanzado a la página del estado de las colas de procesos.
Flujo principal	1. El usuario visualiza las tablas que representan el estado de las colas de procesos en distintos momentos de la simulación.
Flujos alternativos	-
Postcondiciones	El usuario obtiene una visión detallada de cómo los procesos se agrupan y evolucionan en las colas a lo largo de la simulación.

Tabla 8. CU-06. Visualiza las colas de estados de los procesos.

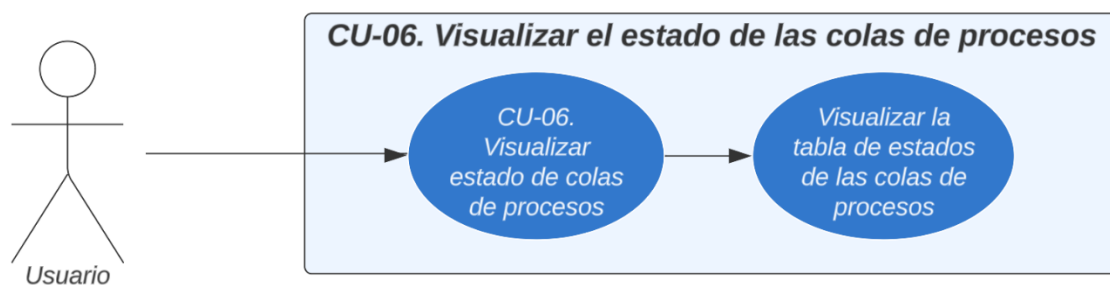


Figura 17. CU-06.

8.3.7. Cambiar Tema de la Aplicación

Campo	Descripción
Caso de uso	Cambiar Tema de la Aplicación.
Identificador	CU-07.
Descripción	El usuario cambia el tema de la aplicación entre oscuro y claro utilizando el botón correspondiente en la barra superior.
Actores	Usuario.
Precondiciones	El usuario se encuentra en cualquier página de la aplicación.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario pulsa el botón de cambio de tema en la barra superior. 2. La aplicación alterna su apariencia entre el tema actual y el opuesto.
Flujos alternativos	-
Postcondiciones	La aplicación adopta el tema oscuro o claro según la preferencia del usuario, proporcionando una experiencia visual adaptada a su elección.

Tabla 9. CU-07. Cambio de tema de la aplicación.

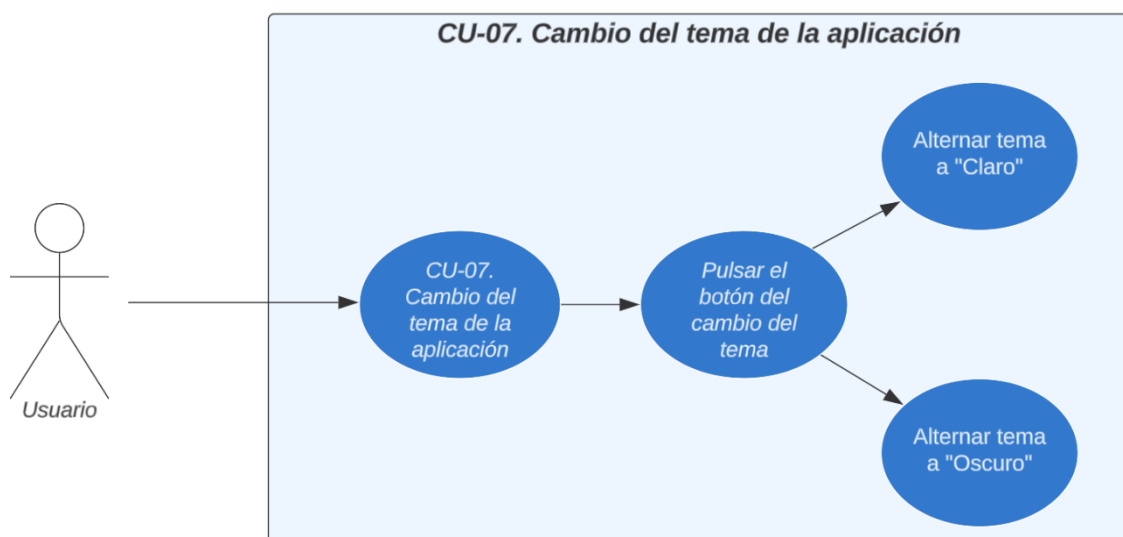


Figura 18. CU-07.

Bloque
III. Diseño del sistema

Capítulo

9. Arquitectura del sistema

La arquitectura de la aplicación se ha diseñado siguiendo un enfoque modular, lo que facilita la escalabilidad y permite la adición de nuevas funcionalidades en el futuro de manera eficiente. En este proyecto, el desarrollo se ha centrado en el lado cliente, ya que no se requiere un servidor para cumplir con los objetivos de la aplicación.

El lado cliente es la parte visible y con la que interactúa el usuario de la aplicación. Está compuesto por varios tipos de ficheros que trabajan en conjunto para proporcionar la funcionalidad y la interfaz gráfica de la aplicación:

1. **Archivos de Código Lógico:** Estos archivos contienen el código que define las acciones y tareas que debe llevar a cabo cada actividad de la aplicación. Los archivos de código lógico están escritos en Kotlin tal y como se ha especificado en apartados anteriores.
2. **Archivos de Interfaz:** Los archivos de interfaz definen la parte gráfica de la aplicación y están implementados utilizando Jetpack Compose, un *framework* de interfaz de usuario moderno y declarativo de Kotlin. Esto permite la creación de interfaces de usuario dinámicas y atractivas.
3. **Modelos:** Estos ficheros definen los objetos que almacenan y representan la información en la aplicación. Están escritos en Kotlin y se encargan de estructurar y gestionar los datos utilizados en la aplicación.
4. **Recursos Multimedia:** En esta sección se incluyen todos los recursos multimedia que forman parte de la aplicación, como imágenes, iconos, sonidos, etc. Estos recursos se utilizan para enriquecer la experiencia del usuario.

La arquitectura de la aplicación se ha concebido con un enfoque modular que proporciona una base sólida para el desarrollo y la expansión futura. Esto significa que la aplicación se divide en componentes independientes, cada uno de los cuales cumple una función específica. Cada componente se desarrolla de manera autónoma y se puede mejorar o ampliar sin afectar a los demás, lo que simplifica enormemente el proceso de desarrollo y mantenimiento.

Un elemento clave de esta arquitectura es la elección de Kotlin como lenguaje de programación principal, como se ha mencionado en previos apartados. Kotlin es un lenguaje moderno y versátil que ofrece numerosas ventajas, como una mayor concisión de código, una sintaxis más legible y un mejor soporte para la programación orientada a objetos y funcional. Además, Kotlin se integra perfectamente con las herramientas de desarrollo de Android, lo que facilita la creación de aplicaciones Android de alta calidad.

La interfaz de usuario de la aplicación se implementa utilizando Jetpack Compose, un framework de IU declarativo desarrollado por Google. Jetpack Compose simplifica la creación de interfaces de usuario dinámicas y atractivas al permitir a los desarrolladores definir la interfaz de usuario mediante código en lugar de archivos XML. Esto no solo agiliza el proceso de diseño, sino que también hace que la interfaz de usuario sea más flexible y fácil de mantener.

Otro aspecto importante de la arquitectura es la gestión de datos y modelos. Los modelos, escritos en Kotlin, definen cómo se estructuran y almacenan los datos en la aplicación. Esto asegura que la información se organice de manera coherente y sea fácilmente accesible para su procesamiento y visualización en la interfaz de usuario.

Por último, los recursos multimedia, como imágenes e iconos, se incorporan a la aplicación para mejorar la experiencia del usuario. Estos recursos se gestionan de manera eficiente, lo que garantiza que la aplicación sea receptiva y no consuma recursos innecesarios.

En resumen, la arquitectura modular de la aplicación, la elección de Kotlin y Jetpack Compose, y la gestión eficiente de recursos multimedia son pilares fundamentales que garantizan un desarrollo eficiente y sostenible de la aplicación. Esta estructura proporciona la flexibilidad necesaria para futuras expansiones y mejoras, lo que permitirá que la aplicación pueda seguir evolucionando mediante la implicación de futuros estudiantes.

Capítulo

10. Diseño de la interfaz

El diseño de la interfaz de usuario es un componente fundamental en el desarrollo de cualquier aplicación móvil exitosa, ya que desempeña un papel crucial en la experiencia del usuario. En este apartado, se detallará minuciosamente la estructura y el aspecto de la interfaz de la aplicación, centrándose en cada uno de los módulos que la componen. Cada módulo ha sido cuidadosamente diseñado con el propósito de ofrecer una experiencia de usuario intuitiva, atractiva y eficaz.

A lo largo de este apartado, se destacarán los subapartados que describen de manera exhaustiva cada uno de los módulos de la interfaz, resaltando sus características distintivas y su integración en la aplicación. Cada módulo ha sido concebido con especial énfasis en la usabilidad, la accesibilidad y la estética, con el objetivo primordial de proporcionar a los usuarios una experiencia fluida y satisfactoria al interactuar con la aplicación.

Cabe mencionar que el esquema de colores empleado en el diseño de la interfaz se ajusta a las directrices específicas proporcionadas por la universidad, garantizando así una coherencia visual con la identidad institucional y contribuyendo a consolidar la imagen de la aplicación como una herramienta de calidad y confiabilidad.

En el proceso de definición y desarrollo de la interfaz de usuario para la aplicación, se han considerado varias metodologías, cada una con sus propias ventajas y desafíos. Estas metodologías incluyen enfoques "Top-Down" y "Bottom-Up", así como el enfoque "Iterativo e Incremental". A continuación, se discutirá cómo se ha abordado la definición de la interfaz de usuario en este proyecto y por qué se ha optado por un enfoque iterativo e incremental.

- **Metodología Top-Down (Arriba-Abajo):** Inicialmente, se consideró la posibilidad de adoptar un enfoque "Top-Down" para definir la estructura general de la interfaz de usuario. Esto habría implicado comenzar con una visión panorámica de la aplicación y luego desglosarla en componentes más específicos. Si bien este enfoque es valioso para establecer una estructura coherente, se consideró que no se adaptaba completamente a la naturaleza modular de la aplicación y a la necesidad de ajustar los detalles de manera flexible a medida que se avanzaba.
- **Metodología Bottom-Up (Abajo-Arriba):** También se consideró el enfoque "Bottom-Up", que se centra en la construcción de componentes individuales y su posterior integración en la interfaz completa. Si bien este enfoque es efectivo para abordar componentes aislados y funcionales, se encontró que podría resultar en desafíos en la integración y coherencia general de la interfaz, especialmente cuando se trata de un proyecto complejo con múltiples módulos interdependientes.
- **Metodología Iterativa e Incremental:** Dado el enfoque modular del proyecto y la necesidad de una interfaz de usuario bien definida y cohesionada, se ha optado por seguir una metodología "Iterativa e Incremental". Este enfoque permite dividir el desarrollo en pequeñas iteraciones o fases, cada una de las cuales produce incrementos funcionales y mejoras en la interfaz.

Cada iteración se enfoca en un módulo específico de la aplicación, desarrollando y refinando su interfaz y funcionalidad de manera aislada. Esto permite obtener retroalimentación temprana de usuarios y realizar ajustes según sea necesario. Al seguir un enfoque iterativo, es posible asegurar que cada parte de la interfaz funcione de manera eficaz antes de avanzar a la siguiente.

Este enfoque es particularmente adecuado para proyectos de desarrollo de software como este, donde la flexibilidad y la capacidad de adaptación son esenciales. También se alinea bien con la naturaleza educativa de esta aplicación, ya que permite incorporar características y mejoras en función de la retroalimentación y las necesidades cambiantes del entorno educativo.

En resumen, se ha adoptado una metodología "Iterativa e Incremental" para el diseño de la interfaz de usuario de la aplicación, ya que se considera que es la más adecuada para el proyecto actual y el estilo de trabajo seguido. Este enfoque permite abordar de manera

efectiva los desafíos de diseño y desarrollo, asegurando que la interfaz sea coherente, funcional y adaptable a medida que se avanza en el proceso de desarrollo.

10.1. Pantalla de carga

La pantalla de carga es el primer punto de contacto entre la aplicación y el usuario. Se presenta al iniciar la aplicación. Su propósito principal es proporcionar una breve animación de entrada y salida del icono de la aplicación, lo que agrega un toque estético a la experiencia de inicio.

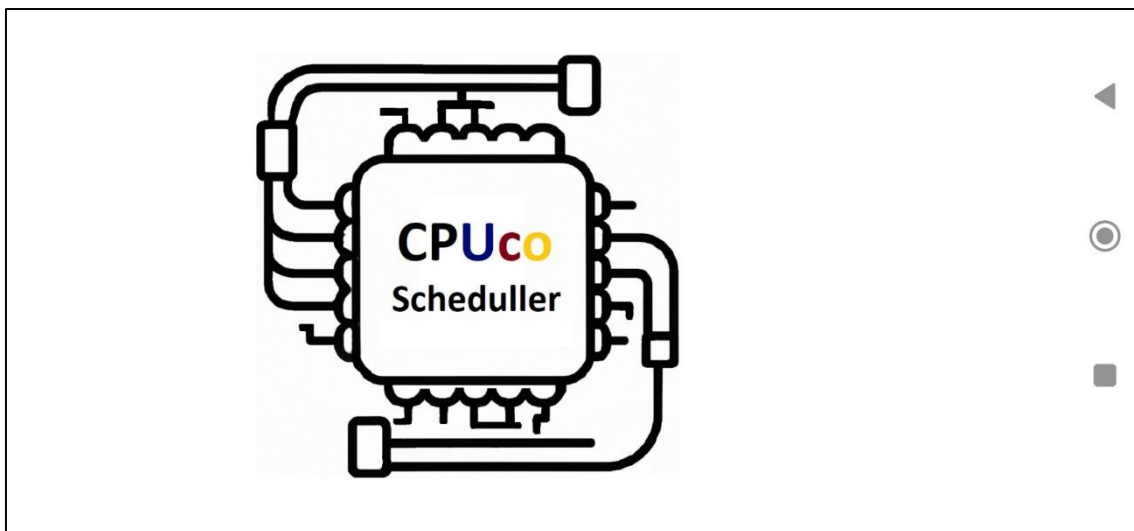


Figura 19. Pantalla de carga.

La pantalla de carga tiene una funcionalidad dual. En primer lugar, mejora la experiencia visual del usuario al agregar un elemento de diseño atractivo durante la transición inicial. Esto contribuye a una impresión positiva desde el primer momento.

En segundo lugar, la pantalla de carga desempeña un papel fundamental en el rendimiento de la aplicación. Mientras se muestra esta animación, la aplicación aprovecha ese tiempo para llevar a cabo tareas de inicialización y precarga de recursos necesarios para el funcionamiento de la aplicación. Esto significa que cuando el usuario llega a la pantalla principal, la aplicación está lista para funcionar de manera fluida y eficiente, sin demoras ni interrupciones notables.

10.2. Página principal

La pantalla principal de la aplicación ha sido diseñada cuidadosamente para brindar una experiencia de usuario intuitiva y eficiente. En esta sección, se describirá en detalle cada uno de los elementos que componen esta interfaz principal.

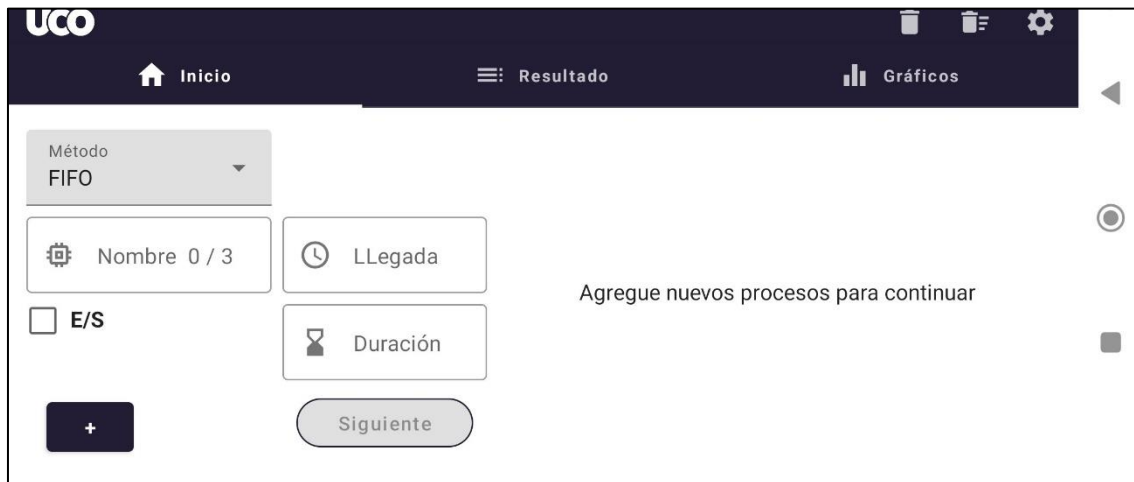


Figura 20. Página principal.

10.2.1. Barra Principal

En la parte superior de la pantalla, se encuentra la barra principal de la aplicación, que presenta las siglas de la universidad y tres iconos importantes. El primero es un icono de eliminación de procesos, que permite a los usuarios eliminar procesos individualmente. El segundo, de forma similar al primero, es un botón para eliminar los procesos, pero considerando la eliminación de todo el listado introducido hasta el momento. El tercero es un botón para cambiar el tema de la aplicación, lo que permite alternar entre los modos oscuro y claro según las preferencias del usuario.

10.2.2. Barra de Navegación

Justo debajo de la barra principal, se encuentra la barra de navegación. Esta barra muestra los títulos de las distintas páginas o módulos de la aplicación, lo que facilita la navegación. Los títulos también funcionan como marcadores para acceder rápidamente a cada página pulsando en ellos o deslizando lateralmente desde la propia pantalla.

10.2.3. Selector de Algoritmo

Debajo, a la izquierda de la pantalla principal, se ha ubicado un selector desplegable que permite a los usuarios elegir el algoritmo de planificación que desean aplicar. Por defecto, se selecciona el algoritmo FIFO (First-In, First-Out). Este selector proporciona una forma clara y accesible de personalizar la simulación según las preferencias del usuario.

10.2.4. Formulario de Datos del Proceso

Justo debajo del selector de algoritmo, se encontrarán los campos correspondientes para introducir los datos necesarios para agregar un proceso al listado de simulación. Cada campo se ha diseñado cuidadosamente y está sujeto a restricciones para garantizar que los datos introducidos sean válidos y no provoquen errores importantes en la simulación. Además, se ha incluido una casilla de verificación que permite al usuario agregar eventos de entrada y salida al proceso, lo que brinda flexibilidad para modelar escenarios más complejos.



The screenshot shows the main interface of the UCO simulation software. At the top, there is a dark blue header with the 'UCO' logo on the left and navigation icons (home, list, bar chart) on the right. Below the header, there are three tabs: 'Inicio' (selected), 'Resultado', and 'Gráficos'. The main content area is divided into two sections. On the left, there is a 'Método' dropdown menu set to 'FIFO'. Below it, there are three input fields: 'Nombre' (with a value of '0 / 3'), 'Llegada' (with a clock icon), and 'Duración' (with a checkmark icon). There are also two buttons labeled 'In' and 'Out', and a 'Siguiete' button. On the right, there is a table with three columns: 'Nombre', 'Llegada', and 'Duración'. The table contains one row with the values 'A', '2', and '3' respectively.

Nombre	Llegada	Duración
A	2	3

Figura 21. Página principal con proceso y E/S seleccionado.

10.2.5. Botones de Acción

Bajo el formulario de datos del proceso, se han colocado dos botones importantes. El primero, con el símbolo "+" (más), permite a los usuarios agregar el proceso al listado de trabajo después de completar los campos requeridos. Este botón también notificará los errores si se encuentran problemas en los datos introducidos.

El segundo botón, etiquetado como "Siguiete", desencadena la ejecución de la simulación y lleva al usuario a la página de resultados. Esta función permite a los usuarios avanzar con facilidad una vez que han preparado todos los procesos para la simulación.

10.2.6. Tabla de Procesos

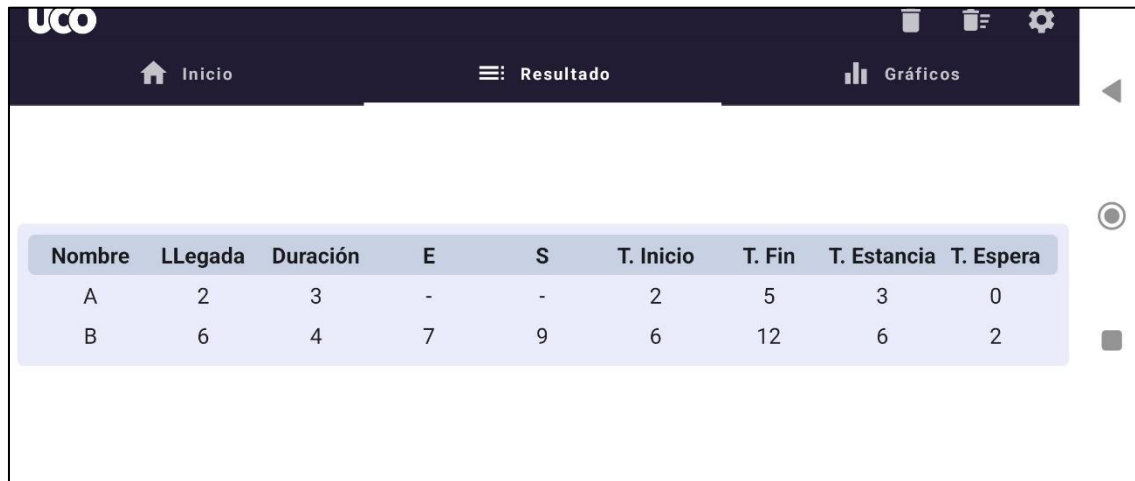
En la parte derecha de la pantalla principal, se encuentra una tabla que muestra de manera clara y organizada los procesos agregados junto con la información correspondiente. Esta tabla proporciona una visión general de los procesos creados y sus detalles, lo que facilita la supervisión y la verificación de la información introducida por el usuario.



Figura 22. Página principal con procesos con E/S.

10.3. Página de resultados temporales

La página de "Resultados Temporales" es una sección esencial de la aplicación que proporciona información detallada sobre los resultados obtenidos después de ejecutar una simulación. Esta página mantiene la misma parte superior que la pantalla principal para garantizar la coherencia en el diseño y la navegación de la aplicación.



Nombre	Llegada	Duración	E	S	T. Inicio	T. Fin	T. Estancia	T. Espera
A	2	3	-	-	2	5	3	0
B	6	4	7	9	6	12	6	2

Figura 23. Página de resultados.

En la parte central de la página, se encuentra una tabla que muestra los valores numéricos correspondientes a los tiempos resultantes de la simulación. Estos tiempos incluyen información crucial, como el tiempo de inicio, el tiempo de finalización, el tiempo de espera, entre otros. Cada fila de la tabla representa un proceso simulado, y cada columna muestra un valor específico asociado a ese proceso.

Esta página permite al usuario obtener una visión detallada de cómo se han comportado los procesos durante la simulación. Los valores numéricos proporcionan información crucial sobre el rendimiento y la eficiencia del algoritmo seleccionado. Los usuarios pueden analizar estos resultados para evaluar cómo se han distribuido los tiempos y tomar decisiones informadas en función de los datos mostrados.

Desde esta página, los usuarios pueden navegar fácilmente a otras secciones de la aplicación utilizando la barra de navegación superior, lo que les brinda la flexibilidad de acceder a diferentes funcionalidades y explorar más a fondo los resultados.

10.4. Página de diagrama de Gantt

La sección del "Diagrama de Gantt" o "gráfica de tiempos" en la aplicación presenta un diseño coherente con el resto de los módulos, lo que garantiza una experiencia de usuario consistente y familiar.

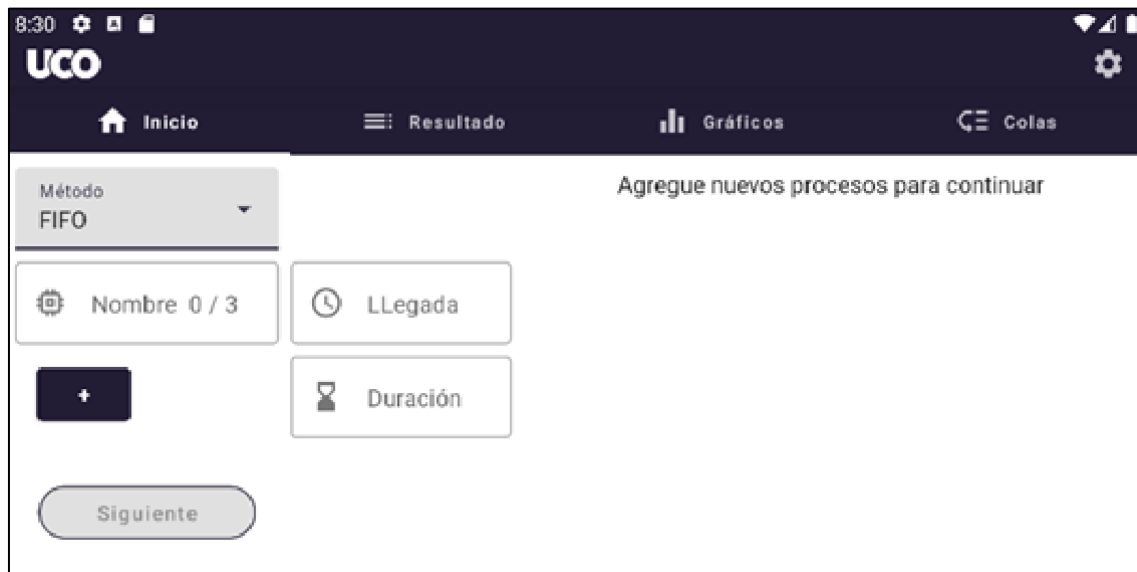


Figura 24. Página de gráfica de Gantt.

Como en todas las páginas de la aplicación, la parte superior de esta pantalla contiene la barra principal con las siglas de la universidad y los iconos para eliminar procesos y cambiar el tema de la aplicación. Justo debajo se encuentra la barra de navegación, que permite al usuario desplazarse fácilmente entre las distintas páginas de la aplicación.

El contenido principal de la página se divide en dos partes clave: una leyenda y una tabla de representación visual. La leyenda se encuentra directamente bajo la barra de navegación superior y proporciona una explicación detallada de los símbolos y códigos utilizados en el diagrama de Gantt. Esta información es esencial para que el usuario comprenda la representación visual de los procesos y sus estados en el diagrama.

Justo debajo de la leyenda, se presenta la tabla de representación visual del diagrama de Gantt. Esta tabla muestra los tiempos de ejecución de los procesos y su estado en diferentes momentos de la simulación. Cada fila de la tabla representa un proceso simulado, mientras que las columnas representan los momentos o intervalos de tiempo durante la simulación.

En el diagrama de Gantt, los procesos se muestran como barras de colores con una longitud proporcional a su tiempo de ejecución. Los diferentes colores y símbolos utilizados en las barras reflejan los estados en los que se encuentran los procesos en cada momento.

La tabla de representación visual del diagrama de Gantt permite a los usuarios obtener una visión clara y detallada de cómo se desarrolla la simulación en función del tiempo. Pueden identificar fácilmente los momentos en los que se inician y finalizan los procesos, así como su estado en diferentes etapas de la simulación. Esta información es valiosa para comprender el comportamiento de los procesos y evaluar el rendimiento del algoritmo seleccionado.

En resumen, esta página proporciona a los usuarios una representación visual clara de los tiempos y estados de los procesos simulados durante la ejecución. La combinación de la leyenda y la tabla visual facilita la comprensión y el análisis de los resultados de la simulación. Los usuarios pueden utilizar esta información para tomar decisiones adecuadas y mejorar sus conocimientos sobre la planificación de procesos.

10.5. Página de colas de procesos

En la sección de "Colas de Procesos" de la aplicación, se sigue la misma estructura de diseño que en las otras páginas, asegurando coherencia en la experiencia del usuario. En la parte superior de esta pantalla, se encuentra la barra principal con las siglas de la universidad y los iconos para eliminar procesos y cambiar el tema de la aplicación. Justo debajo, se encuentra la barra de navegación que permite al usuario acceder fácilmente a las distintas páginas de la aplicación.

El contenido principal de la página se centra en representar de manera clara y visual la información relacionada con las colas de procesos durante la simulación. Esta información se presenta a través de una o más tablas, donde cada tabla representa una cola específica.

En estas tablas, las filas corresponden al orden o posición de cada proceso en la cola, mientras que las columnas representan los momentos durante la simulación. Cada celda de la tabla contiene información relevante sobre el estado de un proceso en un momento determinado.

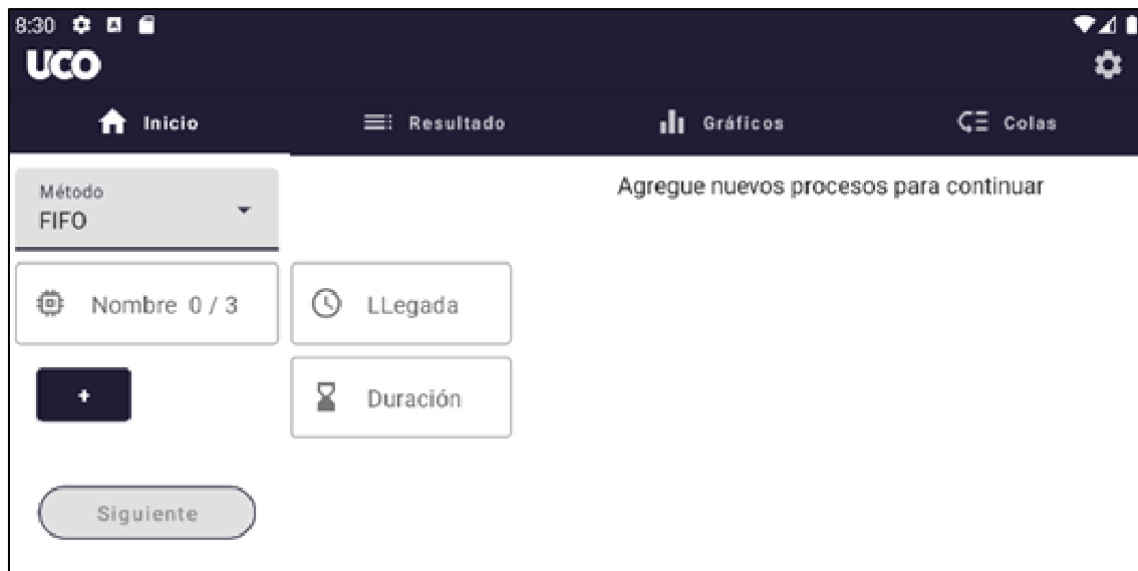


Figura 25. Página de colas de procesos.

Los datos en estas tablas se actualizan al finalizar la simulación, lo que permite a los usuarios seguir el progreso de los procesos en cada una de las ejecuciones que realicen. Pueden ver cómo cambian los estados de los procesos y cómo se modifican a través de las colas a lo largo del tiempo.

Esta representación visual de las colas de procesos proporciona a los usuarios una visión detallada y fácil de entender de cómo se gestionan y ejecutan los procesos durante la simulación. Pueden identificar rápidamente el orden de ejecución, los tiempos de espera y otros detalles importantes relacionados con las colas de procesos.

Bloque IV. Pruebas

Capítulo

11. Pruebas

La fase de pruebas desempeña un papel crucial en el ciclo de desarrollo de este proyecto, donde se someten los componentes de la aplicación a diversas ejecuciones y evaluaciones exhaustivas para verificar su cumplimiento de los requisitos establecidos. A diferencia de una fase aislada, las pruebas se integran de manera continua a lo largo del desarrollo, trabajando en paralelo con la codificación de la aplicación. Sin embargo, al concluir la etapa de desarrollo, se dedica un tiempo específico para llevar a cabo pruebas rigurosas tanto a nivel individual de los módulos como a nivel integral del sistema.

Para asegurar la fiabilidad del sistema, se inicia con pruebas unitarias que evalúan minuciosamente cada módulo de funcionalidad, garantizando que cada función se comporte conforme a las expectativas. Estas pruebas unitarias son un paso fundamental antes de avanzar hacia las pruebas del conjunto de módulos que componen la aplicación.

A continuación, se explorarán en detalle los principios y objetivos de las pruebas a las que se someterá el software, así como algunas de las pruebas específicas que se han implementado en este proyecto.

11.1. Estrategia de pruebas

Diseñar una estrategia de pruebas sólida es un elemento crucial para garantizar la calidad y confiabilidad de la aplicación, evitando posibles errores y asegurando que sea completamente funcional, segura y robusta. La estrategia de pruebas es una parte esencial de cualquier proyecto de desarrollo de software, y en el contexto de este proyecto de TFG, se centra en asegurar que la aplicación cumple con los requisitos y expectativas establecidos.

La estrategia de pruebas se compone de varios tipos de casos de prueba, cada uno con un propósito específico:

1. **Pruebas Unitarias:** Estas pruebas se centran en verificar el funcionamiento individual de cada componente o módulo de la aplicación. Para lograr esto, se dividirán en dos categorías:
 - a. **Pruebas de Caja Negra:** Evaluarán la funcionalidad de los componentes sin conocer su implementación interna. Esto garantiza que los resultados sean coherentes con los requisitos funcionales y las expectativas del usuario.
 - b. **Pruebas de Caja Blanca:** Examinarán la lógica interna de los componentes, asegurando que los caminos de ejecución, las estructuras de control y las variables cumplan con las especificaciones técnicas y contribuyan al correcto funcionamiento del sistema.
2. **Pruebas de Integración:** Estas pruebas se centran en evaluar la interacción entre diferentes componentes o módulos de la aplicación. El objetivo es garantizar que se comuniquen y colaboren de manera efectiva para lograr la funcionalidad completa de la aplicación.
3. **Pruebas de Sistema:** Se realizan para evaluar el comportamiento y el rendimiento del sistema en su conjunto. Esto incluye probar las funcionalidades globales, la navegación entre pantallas y la respuesta del sistema a situaciones límite. Se verificará que la aplicación se comporte correctamente en diferentes dispositivos Android.
4. **Pruebas de Usabilidad:** Las pruebas de usabilidad implican que los usuarios reales prueben la aplicación y brinden comentarios sobre su facilidad de uso. Esto puede ser valioso para evaluar si la interfaz de usuario es intuitiva y amigable para estudiantes y docentes.
5. **Pruebas de Aceptación:** Estas pruebas implican la validación de la aplicación por parte del usuario final o del docente. Se asegura de que la aplicación cumple con los requisitos iniciales y es satisfactoria para su uso en el contexto educativo.

En este documento, se detallarán los casos de prueba más relevantes de cada una de estas categorías. Dado que la aplicación puede contener múltiples funciones similares, se mostrarán aquellos casos de prueba que mejor ejemplifican el enfoque de la estrategia de pruebas. En caso de que los resultados de las pruebas no cumplan con los estándares mínimos de calidad, se procederá a modificar y optimizar el código correspondiente para garantizar un rendimiento óptimo y una experiencia de usuario de alta calidad.

11.2. Pruebas de caja blanca

Dentro del proceso de desarrollo de software, es fundamental garantizar la confiabilidad y la robustez de una aplicación. Las pruebas de caja blanca, también conocidas como pruebas estructurales, son un enfoque esencial para lograr este objetivo. En estas pruebas, se examina y valida la estructura interna del código fuente de la aplicación. En esencia, se trata de abrir la "caja blanca" del software y revelar sus secretos internos.

Estas pruebas se basan en el conocimiento de cómo funciona el código, lo que incluye la lógica interna, las rutas de ejecución, las condiciones lógicas y los bucles. Al comprender estas características, los probadores pueden diseñar casos de prueba que evalúen exhaustivamente el código y sus posibles caminos de ejecución. Esto permite identificar posibles errores de programación, garantizar una mayor cobertura del código y mejorar la calidad general del software.

En este subapartado, se explorarán en detalle las pruebas de caja blanca realizadas como parte de la estrategia de pruebas de este proyecto. Se describirán los enfoques utilizados, los criterios de prueba, las herramientas empleadas y, lo que es más importante, los resultados obtenidos. Estas pruebas ofrecen una visión profunda del funcionamiento interno de la aplicación, brindando la confianza necesaria en su integridad y capacidad de respuesta.

A continuación, se presentarán las pruebas de caja blanca realizadas como parte de esta estrategia de pruebas y se discutirá cómo estas pruebas han contribuido a garantizar la calidad y la fiabilidad de la aplicación.

11.2.1. CU-01. Seleccionar Algoritmo y Configurar Datos

A continuación, se presenta la [tabla 10](#), que resume las pruebas realizadas en el desarrollo de la aplicación. Cada prueba se ha diseñado para evaluar diferentes aspectos de la funcionalidad y el rendimiento de la aplicación, y se han definido los resultados esperados para cada una.

A través de esta tabla, se documentan los resultados obtenidos y cualquier acción correctiva tomada en caso de que el resultado no cumpla con las expectativas. Esta información proporciona una visión general de la calidad y el estado de la aplicación después de las pruebas.

Descripción	
1	El menú desplegable selecciona correctamente el algoritmo deseado.
2	El formulario actualiza correctamente los campos en función al algoritmo.
3	Se almacenan correctamente la información de los campos.
4	Se consideran adecuadamente los campos correspondientes al evento de E/S (cuando se selecciona).
5	Se agrega correctamente un nuevo proceso al pulsar el botón “+”.

Tabla 10. Pruebas de caja blanca para CU-01.

Las pruebas 1, 2 y 5 arrojaron resultados satisfactorios tras realizar ajustes en el enfoque seguido.

Sin embargo, las pruebas restantes (pruebas 4 y 5) inicialmente presentaron errores de ejecución, los cuales se subsanaron en iteraciones posteriores hasta alcanzar los resultados deseados.

Con relación a la prueba 3, los fallos detectados no eran críticos, sino que generaban resultados no deseados. Precisamente, se observó que la información en los campos se almacenaba correctamente, pero era imperativo restringir la casuística de caracteres válidos para evitar la inserción de datos no permitidos. Esta limitación se logró mediante el empleo de expresiones regulares, que controlaron tanto la cantidad como el tipo de caracteres permitidos en cada campo. Adicionalmente, se adaptó el tipo de teclado (numérico o estándar) en función del campo en el que se encontraba el usuario, y se implementaron notificaciones de error que impiden la ejecución de la simulación en caso de campos vacíos o valores no permitidos.

En cuanto a la prueba 4, se identificó un error por el cual los campos del evento de E/S se consideraban activos en todo momento, independientemente de si se seleccionaban o no. La corrección de este fallo incluyó la incorporación de una variable de confirmación de selección al agregar el proceso correspondiente.

11.2.2. CU-02. Visualizar y Modificar Datos

A continuación, se presenta la [tabla 11](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Los procesos que se agregan se añaden correctamente a la tabla.
2	Se visualiza correctamente la información cuando se tiene un evento de E/S.
3	Se elimina correctamente el proceso seleccionado.
4	Se eliminan correctamente todos los procesos.

Tabla 11. Pruebas de caja blanca para CU-02.

Las pruebas 1, 2 y 4 han arrojado resultados esperados sin requerir modificaciones significativas.

No obstante, la prueba número 3 presentó un problema en la eliminación adecuada del proceso deseado. La solución adoptada implicó la necesidad de introducir una variable única como identificador para cada proceso, permitiendo su selección y eliminación precisa del listado.

11.2.3. CU-03. Iniciar Simulación.

A continuación, se presenta la [tabla 12](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	El botón “Siguiente” ejecuta la llamada al algoritmo seleccionado.
2	El botón “Siguiente” avanza automáticamente a la ventana de Resultados.

Tabla 12. Pruebas de caja blanca para CU-03.

En este caso, todas las pruebas han sido satisfactorias y no se han encontrado errores de ejecución críticos.

11.2.4. CU-04. Visualizar Resultados.

A continuación, se presenta la [tabla 13](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	La tabla de resultados muestra correctamente los valores obtenidos.
2	La tabla de resultados muestra adecuadamente los resultados al existir más procesos de los que entran en pantalla.

Tabla 13. Pruebas de caja blanca para CU-04.

La prueba número 1 ha obtenido el resultado esperado.

En contraste, la prueba número 2 no desplegaba los procesos correspondientes cuando la tabla superaba el tamaño de la pantalla del dispositivo. Para abordar esta cuestión, se implementó una solución consistente en emplear un nuevo formato de tabla con funcionalidad de desplazamiento vertical. Con esta modificación, el contenido que excede la capacidad de visualización de la pantalla no se muestra hasta que se encuentra dentro de la pantalla visible.

11.2.5. CU-05. Ver Diagrama de Gantt

A continuación, se presenta la [tabla 14](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Se visualiza correctamente la leyenda en la parte superior.
2	Los símbolos se colocan correctamente en las celdas correspondientes.

Tabla 14. Pruebas de caja blanca para CU-05.

Todas las pruebas realizadas han arrojado resultados satisfactorios, y no se han identificado errores de ejecución que impliquen un carácter crítico o que puedan comprometer el funcionamiento adecuado de la aplicación.

11.2.6. CU-06. Visualizar Estados de Colas de Procesos

A continuación, se presenta la [tabla 15](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	La tabla correspondiente a la cola de procesos se visualiza correctamente.
2	Las colas de procesos se muestran adecuadamente en cada momento de la ejecución.

Tabla 15. Pruebas de caja blanca para CU-06.

En este caso, es importante destacar que todas las pruebas realizadas han arrojado resultados satisfactorios, y no se han identificado errores de ejecución que revistan carácter crítico o que puedan comprometer el funcionamiento adecuado de la aplicación.

11.2.7. CU-07. Cambiar el Tema de la Aplicación

A continuación, se presenta la [tabla 16](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	El menú de ajustes despliega correctamente el <i>switch</i> del tema de la app.
2	La animación funciona de manera óptima.
3	Al pulsar el botón, se alterna entre los temas “Claro” y “Oscuro”.

Tabla 16. Pruebas de caja blanca para CU-07.

Las pruebas número 1 y 3 han arrojado resultados adecuados.

En el caso de la prueba número 2, se identificó un problema en la animación de transición entre los temas. Tras la primera pulsación, la animación quedaba bloqueada en dicho estado. Para solventar esta incidencia, se ha implementado una funcionalidad de retorno al estado inicial en la animación cuando el botón es nuevamente pulsado.

11.3. Pruebas de caja negra

A continuación, se presenta el apartado de Pruebas de Caja Negra, una metodología de evaluación que se centra en verificar la funcionalidad de un sistema sin necesidad de conocer su estructura interna o su código fuente. Este enfoque se basa en observar el comportamiento externo de la aplicación y verificar si se cumplen los requisitos especificados previamente.

A lo largo de esta sección, se detallarán las pruebas de caja negra realizadas en el proyecto, evaluando su efectividad y sus resultados.

11.3.1. CU-01. Seleccionar Algoritmo y Configurar Datos

A continuación, se presenta la [tabla 17](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Asegurar que los botones de navegación en la pantalla de inicio, como los marcadores de página funcionen correctamente y permitan acceder a las páginas correspondientes.
2	Confirmar que el selector desplegable de selección de algoritmo muestra todas las opciones disponibles y permite elegir un algoritmo sin problemas.
3	Verificar que los campos de entrada de datos acepten datos válidos y rechacen datos inválidos, como caracteres no numéricos en campos numéricos.
4	Probar la funcionalidad de agregar un proceso con datos válidos.
5	Verificar que se muestran notificaciones de error claras cuando el usuario intenta realizar acciones incorrectas, como agregar un proceso sin completar todos los campos obligatorios.
6	Comprobar que el desplazamiento lateral desde la pantalla de inicio a otras páginas se realice sin problemas y que todas las páginas sean accesibles.
7	Comprobar que el botón de navegación de regreso notifica al usuario la salida de la aplicación y la necesidad de realizar una nueva pulsación para confirmar.

Tabla 17. Pruebas de caja negra para CU-01.

El caso de prueba número 5 experimentó un error debido a que no se realizaba una verificación adecuada de todos los campos. Esto ocasionaba que se pudieran agregar procesos incluso cuando ciertos campos estaban vacíos o contenían valores no válidos.

La solución a este problema se encontró mediante la adición de una capa adicional de control y seguridad. Esta comprobación extra ahora se aplica a todos los campos, complementando las propiedades de validación existentes para garantizar un nivel más alto de integridad y control en la aplicación.

11.3.2. CU-02. Visualizar y Modificar Datos

A continuación, se presenta la [tabla 18](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que se muestren correctamente todos los datos previamente ingresados.
2	Comprobar que los procesos sean eliminables y que, al realizar cambios en ellos, los datos se actualicen de manera precisa y reflejen las modificaciones realizadas.
3	Tras realizar cambios, confirmar que los datos se muestren actualizados en la tabla.

Tabla 18. Pruebas de caja negra para CU-02.

Todas las pruebas de este módulo específico han concluido con éxito, sin necesidad de correcciones relevantes. Esto respalda la calidad y el cumplimiento de los requisitos de este componente del proyecto.

11.3.3. CU-03. Iniciar Simulación.

A continuación, se presenta la [tabla 19](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que, al pulsar el botón "Siguiente", el proceso de simulación se inicie correctamente y que los procesos se ejecuten según el algoritmo seleccionado.
2	Antes de iniciar la simulación, asegurarse de que se realice una validación adecuada de los datos ingresados por el usuario en la pantalla principal.
3	Evaluar cómo se manejan situaciones límite, como simular una gran cantidad de procesos, para asegurarse de que la aplicación no presente problemas de rendimiento o errores.

Tabla 19. Pruebas de caja negra para CU-03.

El caso de prueba número 3 arrojó resultados no deseados. En un principio, no se consideraron limitaciones al ejecutar la simulación con numerosos procesos, lo que afectó negativamente a dispositivos con especificaciones más limitadas. Estos dispositivos, en situaciones extremas, experimentaron bloqueos de la aplicación o interrupciones debido a la alta demanda de recursos.

Como solución, dado que el propósito es proporcionar ejemplos didácticos simplificados en lugar de ejecutar simulaciones realistas, se ha establecido un límite en la cantidad máxima de procesos a agregar y en el número máximo de unidades temporales en CPU por proceso, fijándolos en 99 como valores máximos.

11.3.4. CU-04. Visualizar Resultados.

A continuación, se presenta la [tabla 20](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que la página de resultados muestra todos los resultados esperados de la simulación, como tiempos de inicio, finalización y espera de cada proceso.
2	Confirmar que los valores numéricos mostrados en la página de resultados son precisos y coinciden con los cálculos esperados.
3	Asegurarse de que, si la tabla de resultados es larga y no cabe en la pantalla, se pueda desplazar verticalmente para ver todos los datos.
4	Probar la página de resultados en varios dispositivos con diferentes tamaños de pantalla para asegurarse de que se vea correctamente en todos ellos.

Tabla 20. Pruebas de caja negra para CU-04.

Todas las pruebas de este módulo específico han concluido con éxito, sin necesidad de correcciones significativas.

11.3.5. CU-05. Ver Diagrama de Gantt

A continuación, se presenta la [tabla 21](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que, al seleccionar la página correspondiente en la aplicación, se accede a la página del diagrama de tiempos sin problemas.
2	Verificar que los iconos y colores utilizados en la leyenda sean comprensibles y proporcionen una representación clara de los estados.
3	Asegurarse de que el diagrama se muestra correctamente en la pantalla, con una representación visual clara de los procesos y sus estados en el tiempo.

Tabla 21. Pruebas de caja negra para CU-05.

En el caso de prueba número 2, se obtuvieron resultados no deseados en la representación de iconos y colores en la leyenda y el diagrama posterior. Se solucionó al reasignar el esquema de colores personalizado de la app en lugar de utilizar los colores predeterminados del IDE (Android Studio).

11.3.6. CU-06. Visualizar Estados de Colas de Procesos

A continuación, se presenta la [tabla 22](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que, al seleccionar la página correspondiente, se accede a la página para la visualización de las colas de procesos.
2	Verificar que las colas muestran la información adecuada correspondiente.

Tabla 22. Pruebas de caja negra para CU-06.

En este apartado, se verificó que el caso de prueba número 2 generaba resultados no deseados en la representación de las colas de procesos, ya que los procesos se dispusieron horizontalmente en lugar de verticalmente, que era la disposición deseada. La solución implicó procesar la lista de procesos de cada momento de manera individual, carácter a carácter, para disponerlos correctamente en cada celda de la tabla.

11.3.7. CU-07. Cambiar el Tema de la Aplicación

A continuación, se presenta la [tabla 23](#) con las pruebas realizadas en la aplicación, sus resultados esperados y las acciones correctivas tomadas en caso de que sea necesario.

Descripción	
1	Verificar que, al seleccionar la opción de cambio de tema en la aplicación, el tema de la interfaz cambie de acuerdo con la elección del usuario, ya sea de modo claro a oscuro o viceversa.
2	Comprobar que el tema seleccionado por el usuario se mantenga constante en todas las pantallas de la aplicación después de haber realizado el cambio.
3	Verificar que la opción de cambio de tema esté disponible y funcione correctamente desde cualquiera de las páginas de la aplicación.
4	Verificar que, al abrir la aplicación por primera vez, se aplique el tema predeterminado del sistema y que el usuario tenga la opción de cambiarlo.

Tabla 23. Pruebas de caja negra para CU-07.

En este caso, todas las pruebas de este módulo específico han concluido con éxito, sin necesidad de correcciones adicionales.

11.4. Pruebas de integración

Las pruebas de integración juegan un papel esencial en el proceso de desarrollo de *software*, ya que permiten verificar que los diferentes módulos y componentes de una aplicación funcionen de manera armoniosa cuando se combinan. En este subapartado, se explorarán en detalle las pruebas de integración que se han llevado a cabo en el contexto de este proyecto de Trabajo de Fin de Grado (TFG).

Es importante señalar que, aunque los distintos módulos y páginas de la aplicación están interconectados, la comunicación se centra en la página principal, que actúa como punto central de entrada y control. Recibe y gestiona los datos ingresados por el usuario, y luego el resto de las páginas se encargan de presentar y procesar estos datos en diferentes formatos y representaciones.

En este sentido, las pruebas de integración se centran en asegurar que la página principal pueda comunicarse eficazmente con los demás módulos, garantizando que los datos se transmitan y procesen correctamente a medida que avanzan a través de la aplicación.

Esto es esencial para mantener la coherencia y el funcionamiento sin problemas de la aplicación en su conjunto, a pesar de que la comunicación entre las páginas secundarias es relativamente limitada. En las siguientes secciones, se describirán las pruebas específicas

diseñadas para evaluar esta integración clave entre la página principal y los demás componentes.

11.4.1. Caso de prueba 1: Paso del listado de procesos

Como se mencionó previamente, la página principal desempeña un papel crucial al solicitar y almacenar los datos del usuario en un listado de procesos, que es un componente esencial para el funcionamiento de la aplicación. Dado que todos los demás módulos dependen de esta información, es vital que tengan acceso a dicho listado.

Afortunadamente, esta prueba transcurrió sin problemas, y todos los módulos pudieron acceder correctamente a la información contenida en el listado de procesos.

11.4.2. Caso de prueba 2: Paso de los resultados

Después de la ejecución de la simulación, el listado de procesos se actualiza con los resultados obtenidos, lo que incluye la información relevante para cada proceso. Además, se crea un nuevo listado que contiene los estados de los procesos a lo largo de la simulación. En este punto, estos dos listados se vuelven esenciales para los módulos posteriores.

Dado que los próximos módulos se centran en la representación gráfica de estos resultados, es crucial que tengan un acceso adecuado a la información almacenada en ambos listados mencionados.

Afortunadamente, esta fase de pruebas tampoco presentó errores significativos.

11.5. Pruebas de sistema

En el contexto de desarrollo de *software*, las pruebas de sistema desempeñan un papel fundamental. Estas pruebas tienen como objetivo evaluar el sistema en su conjunto, verificando que todas las partes individuales funcionen de manera cohesiva y cumplan con los requisitos y especificaciones definidos. Es en esta fase donde se pone a prueba la aplicación en su totalidad, antes de su lanzamiento, para identificar posibles problemas o incompatibilidades que puedan surgir en un entorno de uso real.

En este apartado, se detallarán las pruebas de sistema realizadas en el proyecto, abarcando diversos aspectos de la aplicación. Estas pruebas buscan confirmar que la aplicación funcione según lo previsto (ver [Capítulo 7. Especificación de Requisitos](#)) y que los diferentes módulos y componentes interactúen adecuadamente entre sí.

A continuación, se describirán en detalle las pruebas de sistema llevadas a cabo para garantizar un rendimiento óptimo y la total funcionalidad de la aplicación en su conjunto.

11.5.1. Pruebas de Interfaz de Usuario (UI)

Durante esta prueba, se evaluó que todas las páginas de la aplicación mantuvieran su apariencia esperada y que todos los elementos estuvieran correctamente dispuestos en cada una de ellas. Se verificó que esta disposición fuera coherente en diversas resoluciones de pantalla y dispositivos.

Asimismo, se examinó la fluidez en la navegación entre las pantallas y se evaluó el tiempo de respuesta de las interacciones con la interfaz, asegurando que estuviera dentro de los límites aceptables. También se comprobó que todos los elementos interactivos funcionaran sin problemas.

11.5.2. Pruebas de Rendimiento

Se realizó una evaluación exhaustiva del rendimiento de la aplicación al ejecutar simulaciones con un elevado número de procesos o eventos de entrada/salida. El objetivo principal era asegurarse de que la aplicación mantuviera un nivel de respuesta adecuado incluso bajo condiciones de alta carga de trabajo. Para lograr esto, se llevaron a cabo simulaciones que implicaban la manipulación de un gran número de procesos y eventos simultáneos. Durante estas pruebas, se supervisó de cerca cómo se comportaba la aplicación en términos de velocidad y estabilidad, verificando que continuara funcionando sin problemas incluso bajo cargas intensas de trabajo.

Adicionalmente, se efectuó un seguimiento del consumo de recursos del sistema mientras la aplicación estaba en ejecución. Esto incluyó el monitoreo de la memoria y la unidad central de procesamiento (CPU). El objetivo era asegurarse de que la aplicación no agotara excesivamente los recursos disponibles en el dispositivo del usuario. Se registraron y analizaron los datos de uso de recursos para garantizar que la aplicación operara de manera eficiente en términos de *hardware* y no causara problemas de sobrecarga en el dispositivo del usuario.

11.5.3. Pruebas de Navegación

Se llevó a cabo una revisión exhaustiva para confirmar que la navegación entre las diferentes páginas de la aplicación fuera coherente y sin errores. Esto implicó examinar cada transición entre páginas y verificar que todas las interacciones de navegación funcionaran

correctamente. El objetivo era garantizar que el flujo de la aplicación fuera intuitivo y que los usuarios pudieran moverse entre las diferentes secciones de manera fluida y sin problemas. Se comprobaron en detalle los enlaces, botones y gestos utilizados para la navegación, y se aseguró que no hubiera discrepancias ni problemas de funcionamiento.

Adicionalmente, se llevaron a cabo pruebas para evaluar la capacidad de retorno a la página principal desde cualquier punto de la aplicación. Esto implicó probar todas las rutas posibles para volver a la pantalla de inicio, independientemente de en qué página se encontrara el usuario en un momento dado. Se confirmó que todas las opciones de retorno funcionaran correctamente y que los usuarios pudieran regresar a la pantalla principal de manera efectiva en cualquier punto de su interacción con la aplicación.

11.5.4. Pruebas de Estrés

Se sometió la aplicación a situaciones de carga extrema con el objetivo de evaluar su estabilidad y capacidad de recuperación. Estas pruebas se realizaron para determinar cómo se comportaba la aplicación cuando se le exigía al máximo de su capacidad. Se simularon escenarios en los que un gran número de procesos con eventos de E/S se ejecutaban simultáneamente, lo que podría generar una carga significativa en la aplicación.

Estas pruebas permitieron evaluar la respuesta de la aplicación bajo condiciones extremas y confirmar si seguía funcionando de manera estable y sin bloqueos críticos.

Es importante destacar que estas pruebas de carga extrema se llevaron a cabo en conjunto con las pruebas de rendimiento mencionadas anteriormente, lo que proporcionó una evaluación integral del comportamiento de la aplicación en situaciones exigentes.

11.6. Pruebas de aceptación

Estas pruebas se centran en asegurar que la aplicación cumpla con las expectativas y requisitos del cliente o usuario final antes de su implementación. Son la última barrera antes del lanzamiento oficial de la aplicación y se realizan para validar que el *software* es apto para su uso en situaciones reales.

En este apartado, se presentarán las pruebas de aceptación llevadas a cabo en el proyecto. Estas pruebas están diseñadas para verificar que la aplicación satisface plenamente los criterios y necesidades del usuario final, asegurando que todas las funcionalidades respondan adecuadamente a las demandas planteadas. Se describirán

detalladamente las pruebas de aceptación realizadas, destacando cómo se han llevado a cabo y qué resultados se han obtenido.

11.6.1. Validación por el director del TFG

En este caso, se considera la evaluación por parte del director del TFG de la aplicación con el objetivo de identificar posibles errores y evaluar las mejoras técnicas realizadas durante el desarrollo.

El director del TFG desempeña un papel crucial en la validación y revisión de la aplicación, ya que su experiencia y conocimiento son fundamentales para garantizar la calidad y la coherencia del proyecto. A través de esta evaluación, se busca obtener retroalimentación valiosa que contribuya a perfeccionar la aplicación y asegurar que cumple con los estándares requeridos.

A lo largo de las diferentes reuniones, se han identificado diversas mejoras y cambios que han contribuido a la evolución de la aplicación. Algunas de estas mejoras incluyen:

- Cambio a una disposición horizontal para toda la aplicación, lo que ha mejorado la usabilidad y la experiencia del usuario.
- Implementación de una disposición en formato paginado para las distintas pantallas de la aplicación, lo que facilita la navegación y la visualización de la información.
- Agregación del módulo de las colas de procesos, lo que ha enriquecido la funcionalidad y la capacidad de representación de datos en la aplicación.

Estos cambios y mejoras se han incorporado de acuerdo con las necesidades y los requisitos del proyecto, y su evaluación por parte del director del TFG es esencial para validar su efectividad y relevancia en el contexto de la aplicación.

11.6.2. Pruebas de Escenario Básico

Durante las pruebas de aceptación, se evaluó exhaustivamente el escenario básico de la aplicación para asegurar su funcionamiento sin inconvenientes. En primer lugar, se confirmó que los usuarios pudieran abrir la aplicación en dispositivos Android sin experimentar ningún error en el proceso de inicio. Esto es esencial para garantizar que la

aplicación sea accesible y utilizable en una variedad de dispositivos Android de manera confiable.

En cuanto a la selección de algoritmos de planificación, se comprobó que los usuarios pudieran realizar esta tarea de manera sencilla y sin enfrentar obstáculos. La elección del algoritmo es crucial para la ejecución exitosa de la simulación, y asegurarse de que esta funcionalidad esté disponible sin contratiempos es esencial.

Otro aspecto importante evaluado fue la capacidad de los usuarios para introducir datos con precisión, tanto para agregar procesos como para incluir eventos de E/S. Se verificó que los campos de entrada funcionaran correctamente y que las validaciones impidieran la introducción de datos incorrectos o no válidos.

Además, se confirmó que los usuarios pudieran visualizar los datos ingresados de manera clara y comprensible en la interfaz de usuario. La presentación adecuada de la información es fundamental para que los usuarios puedan revisar y verificar los datos que han introducido, lo que contribuye a una experiencia de usuario positiva.

Finalmente, se verificó que los usuarios pudieran ejecutar la simulación sin problemas y que la aplicación respondiera de manera eficaz a esta acción. Dado que la simulación es la característica central de la aplicación, es esencial que esta función se realice sin errores y que los resultados sean coherentes con los datos de entrada.

Estas pruebas de escenario básico aseguraron que la aplicación cumpliera con sus funcionalidades esenciales y que los usuarios pudieran interactuar con ella de manera efectiva.

11.6.3. Pruebas de Interacción con la Interfaz de Usuario

Se prestó especial atención a la evaluación de la facilidad de uso de la aplicación. Esto implicó asegurarse de que los usuarios pudieran navegar de manera intuitiva por las diferentes páginas de la aplicación sin encontrar obstáculos significativos. La navegación es un aspecto crítico para la experiencia del usuario, por lo que se verificó que los elementos de la interfaz de usuario estén dispuestos de manera lógica y que los usuarios pudieran acceder a las funciones clave de manera clara y sencilla.

Además de la navegación, se evaluó la capacidad de los usuarios para realizar acciones importantes, como eliminar procesos o cambiar el tema de la aplicación, de manera eficiente y sin confusiones. Estas acciones son parte integral de la funcionalidad de la aplicación, y es crucial que los usuarios puedan llevarlas a cabo sin dificultad. Se aseguró que los botones y elementos de control estén ubicados de manera que los usuarios puedan identificarlos fácilmente y que las acciones sean intuitivas.

En resumen, estas pruebas se centraron en la experiencia del usuario y la usabilidad de la aplicación. Se garantizó que los usuarios pudieran interactuar con la aplicación de manera fluida y que las acciones comunes sean accesibles y comprensibles, lo que contribuye a una experiencia positiva en general.

11.6.4. Pruebas de Resultados

Se comprobó que los usuarios pudieran ver los resultados de la simulación de manera clara y comprensible. Los resultados de la simulación son un componente esencial de la aplicación, y es fundamental que los usuarios puedan acceder a estos datos de forma sencilla. Se verificó que la presentación de los resultados esté diseñada de manera que sea fácil de entender, con una disposición lógica y una representación visual efectiva.

Además de la claridad en la presentación de los resultados, se confirmó que los tiempos de espera, tiempos de retorno y otros valores relacionados con la simulación sean correctos y coincidan con los resultados esperados. Estos datos son cruciales para comprender el funcionamiento de los algoritmos de planificación y la eficiencia del sistema en general. Se aseguró que los cálculos y representaciones sean precisos, lo que contribuye a la confiabilidad de la aplicación.

11.6.5. Pruebas de Usabilidad

Se llevaron a cabo evaluaciones con usuarios reales para evaluar la facilidad de uso y la experiencia del usuario. Se prestaron especial atención a la interacción con la interfaz de usuario, la navegación entre páginas y la ejecución de acciones como eliminar procesos o cambiar el tema de la aplicación.

El objetivo principal fue identificar posibles áreas de mejora en la usabilidad de la aplicación. Se recopilaron comentarios y retroalimentación de los usuarios para comprender sus necesidades y expectativas. Estos datos ayudaron a refinar la interfaz de usuario y hacer ajustes que mejoren la experiencia del usuario en general.

Bloque

V. Conclusiones y futuras mejoras

Capítulo

12. Conclusiones

Este proyecto de Trabajo de Fin de Grado representa un hito importante para el alumno, siendo el colofón de su carrera universitaria. A lo largo de esta ardua pero apasionante travesía, se han alcanzado objetivos fundamentales. Se ha diseñado, desarrollado y probado una aplicación de simulación de algoritmos de planificación de procesos para dispositivos Android. Este logro no solo demuestra las habilidades técnicas, sino también la capacidad para abordar problemas complejos y transformarlos en soluciones prácticas.

Durante este viaje, se han enfrentado desafíos que han obligaron a repensar enfoques y adoptar soluciones creativas. Se ha aprendido a gestionar el tiempo, los recursos y las expectativas. Pero lo más importante, se ha adquirido una comprensión más profunda de la importancia de la planificación y la organización en el desarrollo de *software*.

En este apartado, se presentarán las conclusiones clave que emergen de este proyecto. Se hará una reflexión sobre los hitos alcanzados, los obstáculos superados y las lecciones valiosas aprendidas a lo largo de este proceso.

12.1. Objetivos operacionales alcanzados

En el desarrollo de este proyecto, se han logrado cumplir de manera satisfactoria los objetivos operacionales establecidos (ver [Capítulo 3. Objetivos](#)). Estos objetivos operacionales, que detallan las acciones y pasos específicos necesarios para alcanzar los objetivos generales del proyecto, han guiado de manera efectiva el proceso de desarrollo de la aplicación.

El objetivo principal del proyecto, que se centraba en la concepción y desarrollo integral de una aplicación dirigida a dispositivos móviles Android para respaldar la enseñanza de conceptos relacionados con la planificación de procesos, ha sido plenamente alcanzado. La aplicación ha sido diseñada con la capacidad de simular eventos de E/S y la implementación de un algoritmo de planificación, así como una base robusta y modular para futuros desarrolladores que deseen aumentar o pulir todas aquellas funcionalidades que lo requieran.

Además, los objetivos específicos que abordaron aspectos como el diseño de la interfaz de usuario, la navegación entre módulos, la selección de algoritmos, la gestión de datos, la visualización de resultados, la eliminación de procesos y la representación gráfica de tiempos y estados de colas también se han cumplido de manera exitosa.

La aplicación presenta una interfaz compacta y accesible, una navegación clara entre sus módulos, una selección sencilla de algoritmos, una especificación clara de datos, una gestión eficiente de eventos de E/S, una visualización organizada de datos ingresados, una eliminación flexible de procesos y una representación clara de resultados numéricos y gráficos.

En resumen, el proyecto ha logrado cumplir con los objetivos operacionales establecidos, asegurando el desarrollo de una aplicación efectiva y funcional que cumple con su propósito educativo y técnico.

12.2. Objetivos formales alcanzados

Los objetivos formales establecidos para este proyecto han sido alcanzados con éxito, lo que demuestra un sólido logro en varios aspectos clave:

1. **Adquisición de Conocimientos de Desarrollo:** Durante el desarrollo de esta aplicación, se ha logrado una comprensión profunda y detallada de todas las etapas implicadas en la creación de aplicaciones, lo que ha permitido abordar los desafíos de manera efectiva y respaldar el diseño y desarrollo integrales de la aplicación.
2. **Manejo de Android Studio:** Se ha adquirido una experiencia sólida en el uso de Android Studio, la plataforma principal para el desarrollo de aplicaciones Android.

3. **Iniciación en el Lenguaje de Programación Kotlin:** Se ha alcanzado un nivel adecuado en el lenguaje de programación Kotlin, lo que ha facilitado la implementación de la lógica de la aplicación y la manipulación de datos.
4. **Aprendizaje en Jetpack Compose:** Se ha logrado familiarizarse con Jetpack Compose, la biblioteca de diseño de interfaces de última generación. Esto ha permitido el desarrollo de una interfaz de usuario intuitiva y moderna que mejora la experiencia del usuario.
5. **Análisis y Resolución de Problemas:** Se ha perfeccionado la capacidad de analizar y resolver problemas complejos que surgieron durante el desarrollo. Esto ha garantizado la estabilidad del producto final.
6. **Toma de Decisiones Estratégicas para la Experiencia de Usuario:** Las decisiones estratégicas tomadas durante el proceso de diseño y desarrollo se basaron en una comprensión profunda de las necesidades del usuario y los principios de diseño. Esto ha mejorado significativamente la experiencia general del usuario.
7. **Optimización de Rendimiento:** Se han implementado estrategias avanzadas de optimización de rendimiento que garantizan un funcionamiento adecuado de la aplicación en diversos escenarios de uso, incluida la ejecución de simulaciones con un gran número de procesos.
8. **Generación de Documentación Completa:** Se ha elaborado una documentación completa y detallada que abarca todas las fases del proyecto. Esto asegura la transparencia y la replicabilidad del proceso, facilitando futuras mejoras y desarrollos.

En conjunto, estos logros demuestran un compromiso sólido con el aprendizaje, la resolución de problemas y la entrega de una aplicación de calidad que cumple con los objetivos formales del proyecto.

Capítulo

13. Futuras mejoras

Un proyecto de *software* nunca es estático, siempre existe margen para la mejora continua. A medida que esta aplicación de simulación de algoritmos de planificación de procesos toma forma, es imperativo considerar cómo evolucionará y se adaptará en el futuro. La informática es un campo en constante evolución, y esta aplicación debe mantenerse al día con las últimas tendencias y requisitos de los usuarios.

En este apartado, se explorarán las áreas donde esta aplicación podría beneficiarse de futuras mejoras y expansiones. A medida que se identifican las limitaciones actuales, también se observan oportunidades para un mayor desarrollo. Se considerará cómo incorporar nuevas características, mejorar la experiencia del usuario y ampliar la utilidad de la aplicación. Con una mirada hacia el futuro, esta sección sienta las bases para que este proyecto siga siendo relevante y útil en un entorno tecnológico en constante cambio. Algunas de las posibles mejoras incluyen:

- **Ampliar los algoritmos implementados:** Se considera la incorporación de un mayor número de algoritmos de planificación, lo que enriquecería la aplicación y proporcionaría a los usuarios una gama más amplia de opciones y recursos para su aprendizaje.

- **Ajustar los campos que requieran los nuevos algoritmos:** Con la incorporación de nuevos algoritmos, es esencial ajustar los campos de entrada de datos de manera efectiva para que reflejen los requisitos específicos de cada algoritmo. Esto podría lograrse mediante una ventana emergente o un asistente de configuración, lo que simplificaría la experiencia del usuario al proporcionar orientación contextual.
- **Modificar procesos:** Actualmente, la aplicación permite la eliminación de procesos, pero se podría mejorar permitiendo la modificación de procesos existentes. Esto brindaría a los usuarios un mayor control sobre los datos de entrada y permitiría realizar ajustes sin necesidad de eliminar y volver a ingresar información.
- **Ajustar automáticamente la cantidad de colas de procesos para visualizar:** En la versión actual, la cantidad de colas de procesos se define de antemano. Sería beneficioso implementar una funcionalidad que ajuste automáticamente la cantidad de colas de procesos que se muestran en función del algoritmo seleccionado. Esto mejoraría la eficiencia y la capacidad de visualización de datos.
- **Señalar el proceso que entra y sale de las colas:** Para una comprensión más clara de la simulación, se podría agregar una función que destaque o señale el proceso que se está moviendo en cada momento dentro de las colas de procesos. Esto ayudaría a los usuarios a seguir el progreso de manera más efectiva.
- **Traducir la aplicación a otros idiomas:** Para ampliar el alcance y la accesibilidad de la aplicación, sería beneficioso traducirla a varios idiomas adicionales. Esto permitiría a usuarios de diferentes regiones y con diferentes idiomas nativos utilizar la aplicación de manera efectiva. Para ello, se debería incorporar un selector de idiomas que permita a los usuarios elegir su preferencia de idioma al iniciar la aplicación. Esto contribuiría a una experiencia de usuario más inclusiva y global.

Estas mejoras potenciales permitirían que la aplicación siga evolucionando y mejorando su utilidad para la enseñanza de la planificación de procesos en el contexto de los sistemas operativos. Además, podrían contribuir a una experiencia de usuario aún más enriquecedora y eficiente.

Bibliografía

- [1] Revista Virtual Pro. Virtual Pro. Obtenido de <https://www.virtualpro.co/noticias/que-es-la-planificacion-de-procesos-de-un-sistema-operativo> (Consultado el 20 de abril de 2023).
- [2] Stallings, W. (2005). Sistemas Operativos, Aspectos internos y principios de diseño. Prentice Hall.
- [3] A. McIver, I. M. (2011). Sistemas operativos, 6a edición. Cengage Learning.
- [4] Esquemas de colores de la UCO: https://www.uco.es/servicios/actualidad/images/documentos/identidad-corporativa/Manual_UCO_definitivo.pdf (Consultado el 27 de mayo de 2023).
- [5] *Clean code*: <https://www.freecodecamp.org/news/clean-coding-for-beginners/> (Consultado el 10 de junio de 2023).
- [6] Android Studio: <https://developer.android.com/studio/intro?hl=es-419> (Consultado el 26 de junio de 2023).
- [7] Kotlin: <https://developer.android.com/kotlin?hl=es-419> (Consultado el 15 de junio de 2023).
- [8] Jetpack Compose: <https://developer.android.com/jetpack/compose?hl=es-419> (Consultado el 06 de mayo de 2023).

- [9] Diagrama de Gantt: <https://asana.com/es/resources/gantt-chart-basics> (Consultado el 12 de junio de 2023).
- [10] Boonsuen. (s.f.). *Boonsuen (GitHub)*. Obtenido de <https://boonsuen.com/process-scheduling-solver> (Consultado el 18 de enero de 2023).
- [11] Gregor Koytainy, P. D.-I. (2014). *AnimOS CPU-Scheduling*. Obtenido de <https://ess.cs.tu-dortmund.de/Software/AnimOS/CPU-Scheduling/> (Consultado el 18 de enero de 2023).
- [12] Cooray, H. (2020). *CPU Scheduling Simulator*. Obtenido de <https://cpu-scheduling-sim.netlify.app/> (Consultado el 18 de enero de 2023).
- [13] Alvareztech. (s.f.). *Alvareztech (GitHub)*. Obtenido de <https://github.com/alvareztech/ProcessSimulatorJava> (Consultado el 21 de enero de 2023).
- [14] IncanatoIT. (2014). *IncanatoIT, Desarrollando Software!* Obtenido de <https://www.incanatoit.com/2014/06/simulador-planificacion-de-cpu-memoria.html> (Consultado el 20 de enero de 2023).
- [15] Ifreddyrondon. (s.f.). *Ifreddyrondon (GitHub)*. Obtenido de https://github.com/ifreddyrondon/ula_so-scheduler (Consultado el 21 de enero de 2023).
- [16] Jeico Games. (s.f.). *CPU Simulator (CPU Scheduling)*. Obtenido de https://play.google.com/store/apps/details?id=com.jeicogames.cpusimulator&hl=es_NI&pli=1 (Consultado el 22 de enero de 2023).
- [17] Itmarck. (s.f.). *Quantum*. Obtenido de <https://play.google.com/store/apps/details?id=software.marcx.quantum> (Consultado el 23 de enero de 2023).
- [18] Software libre: <https://www.gnu.org/philosophy/free-sw.es.html> (Consultado del 15 de julio de 2023).
- [19] Git: <https://git-scm.com/> (Consultado el 12 de junio de 2023).
- [20] Github: <https://github.com/github> (Consultado el 12 de junio de 2023).

- [21] Aaccessibilidad WCAG: https://formiux.com/que-es-wcag/#Que_significa_WCAG (Consultado el 17 de julio de 2023).
- [22] UML: <https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml> (Consultado el 16 de julio de 2023).

