UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE SOFTWARE

Tema:

SISTEMA MULTIPLATAFORMA DE CÓDIGO ABIERTO PARA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL UTILIZANDO EL FRAMEWORK FLUTTER

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero de Software

ÁREA: Software

LÍNEA DE INVESTIGACIÓN: Desarrollo de Software

AUTOR: Luis Steeven López García

TUTOR: Ing. Victor Hugo Guachimbosa Villalba, PhD.

Ambato - Ecuador

Febrero - 2024

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA MULTIPLATAFORMA DE CÓDIGO ABIERTO PARA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL UTILIZANDO EL FRAMEWORK FLUTTER, desarrollado bajo la modalidad Proyecto de Investigación por el señor Luis Steeven López García, estudiante de la Carrera de Ingeniería de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2024.

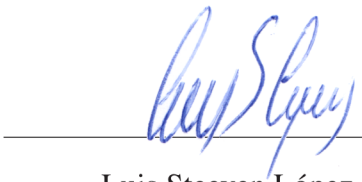
Ing. Víctor Hugo Guachimbosa Villalba, PhD.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA MULTIPLATAFORMA DE CÓDIGO ABIERTO PARA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL UTILIZANDO EL FRAMEWORK FLUTTER es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2024.



Luis Steeven López García.

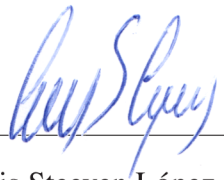
C.C. 1722098199

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2024.



Luis Steeven López García.

C.C. 1722098199

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Luis Steeven López García, estudiante de la Carrera de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA MULTIPLATAFORMA DE CÓDIGO ABIERTO PARA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL UTILIZANDO EL FRAMEWORK FLUTTER, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, febrero 2024.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
DENNIS VINICIO CHICAIZA CASTILLO

Ing. Dennis Chicaiza.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

Ing. Franklin Mayorga, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

Dedico este proyecto a mi querida hermana, Kishanav Nakiara Lopez. Tu haz sido una fuente inagotable de ánimo durante todo este arduo pero gratificante proceso. Esta dedicación refleja mi ferviente deseo de que la pasión y dedicación invertidas en este proyecto sirvan como un faro de aliento en tu propio camino. Que encuentres en estas páginas no solo un testimonio académico, sino también un recordatorio de que con esfuerzo y determinación, los objetivos pueden convertirse en logros tangibles. Gracias por ser mi fuente inquebrantable de motivación. Este logro, sin duda, es compartido contigo, mi fuente constante de aliento.

Steeven Lopez.

AGRADECIMIENTO

Al culminar este proyecto, deseo expresar mi mayor agradecimiento a Dios, cuya guía y fortaleza han sido fundamentales en este viaje académico. Su amor incondicional y sabiduría, han iluminado mi camino, brindándome la fuerza necesaria para superar los desafíos a lo largo del proceso.

Agradezco de manera especial a mi familia, cuyo apoyo inquebrantable ha sido mi mayor bendición, siendo fuentes constantes de inspiración y aliento. Sus sacrificios y paciencia para conmigo han sido la fuerza detrás de cada paso que he dado hacia el cumplimiento de metas.

A mis amigos y compañeros, gracias por las palabras de ánimo, las épicas sesiones de juego y las anécdotas inolvidables que compartimos durante esta emocionante travesía académica.

Steeven Lopez.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN EJECUTIVO

En el presente trabajo se aborda la implantación de una aplicación multiplataforma de código abierto para la resolución de problemas de Programación Lineal de Investigación Operativa utilizando el framework Flutter. Varias aplicaciones especializadas para programación lineal han sido desarrolladas. De estas, destacan PhpSimplex, OpenSolver y Tora, reconocidas por su utilidad, tanto para estudiantes como para profesionales. Algunos de estos programas pueden, por un lado, presentar problemas de compatibilidad con sistemas operativos modernos. Por otro lado, pueden carecer de un modo sin conexión a internet, limitando su accesibilidad y utilidad en situaciones donde la conectividad no es garantizada. Asimismo, la dependencia de paquetes de software de pago suponen una barrera para las usuarios. En primer lugar, la investigación ha realizado el análisis bibliográfico referente a los algoritmos gráfico, algebraico y simplex, evidenciando su aplicabilidad en diferentes escenarios. En segundo lugar, se ha destacado la eficacia y versatilidad del framework Flutter en el desarrollo de aplicaciones multiplataforma, aprovechando su alto rendimiento y rápido ciclo de desarrollo, para generar interfaces de usuario consistentes. Además, la metodología Kanban se ha implementado con éxito, y ha permitido una gestión eficiente del proyecto. En tercer lugar, la aplicación multiplataforma ofrece a los usuarios como una solución autónoma y accesible para abordar problemas de programación lineal en cualquiera de las plataformas más comunes y web sin requerir de conexión a internet o dependencia de software de pago.

Palabras Claves

Aplicación multiplataforma, código abierto, Dart, Flutter, investigación operativa, Kanban, programación lineal, Simplex.

ABSTRACT

In the present work, the implementation of an open-source multiplatform application for solving Linear Programming problems of Operations Research using the Flutter framework is addressed. Several specialized applications for linear programming have been developed. Of these, PhpSimplex, OpenSolver, and Tora stand out for their usefulness, for students and professionals. Some of these programs may, on one hand, present compatibility problems with modern operating systems. Moreover, they may lack an offline mode, limiting their accessibility and usefulness when connectivity is not guaranteed. Likewise, the dependence on paid software packages is a barrier for users. First, analysis of bibliographic material, referring to the graphic, algebraic, and simplex algorithms has been carried out, thus, their applicability in different scenarios is tried. Second, the effectiveness and versatility of the Flutter framework in the development of multiplatform applications has been highlighted, taking advantage of its high performance and rapid development cycle, to generate consistent user interfaces. Furthermore, the Kanban methodology has been successfully implemented and has allowed an efficient management of the project. Third, the cross-platform application provides users with an autonomous and accessible solution for addressing linear programming problems on most of the common platforms and the web, without requiring an internet connection or depending on paid software.

Keywords

Operations research, linear programming, Simplex, Flutter, Dart, custom painter, Riverpod, multi-platform app, Kanban, open source.

CAPITULO I.- MARCO TEÓRICO

1.1 Tema de Investigación

SISTEMA MULTIPLATAFORMA DE CÓDIGO ABIERTO PARA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL UTILIZANDO EL FRAMEWORK FLUTTER.

1.1.1 Planteamiento del problema

El software para ingeniería, ha experimentado una evolución constante en los últimos años, pero persisten problemas que necesitan ser atendidos para mejorar la eficacia y eficiencia del software. La falta de integración entre diferentes herramientas y programas sigue siendo una de las mayores preocupaciones, puesto que dificulta la colaboración y la resolución de problemas complejos.

El desarrollo de software es una disciplina que ha experimentado un gran crecimiento en las últimas décadas [1]. Con la creciente demanda de soluciones informáticas en casi todas las industrias, el software se ha convertido en un componente esencial de la mayoría de los productos y servicios. Sin embargo, esta no es una tarea sencilla y presenta una serie de desafíos y problemas.

Según la literatura, los problemas más comunes en el software incluyen la falta de fiabilidad, seguridad y eficiencia [1]. Además, la falta de estándares y la complejidad de los sistemas también son problemas significativos [2]. Peter Sandborn indica que una significativa causa de la obsolescencia de software son los altos costos relacionados con su mantenimiento [3].

Añadido a lo anterior, la falta de compatibilidad con diferentes sistemas operativos y la variabilidad en la calidad y funcionamiento de los programas son otros problemas recurrentes [4]. Estos problemas tienen un impacto directo en la productividad y eficiencia de los ingenieros.

Existen varios programas de software de optimización lineal para ser utilizados en el campo de la investigación operativa. De estos los más destacados son Phpsimplex, OpenSolver y Tora, los cuales han probado ser programas muy útiles tanto para estudiantes como profesionales [5], pero por variadas razones tales como la falta de compatibilidad con sistemas operativos modernos, ausencia de modo sin conexión a internet [6], o dependencia de software de pago, estos presentan dificultades al momento de utilizarlos.

Otro desafío significativo en la programas de software es la dificultad para mantener y actualizar el software existente. La naturaleza cambiante de la tecnología y los requisitos de los usuarios hacen que el software se vuelva obsoleto, y su mantenimiento y actualización se convierten en una tarea cada vez más costosa y compleja.

1.2 Antecedentes Investigativos

Al realizar una búsqueda de temas referentes al desarrollo de software que tome algoritmos de programación lineal para la optimización, se evidencia la carencia de investigaciones en el sector nacional. Aunque cabe recalcar que se puede encontrar en varios repositorios académicos trabajos que usan una variedad de este tipo de programas, para la optimización de recursos en varios sectores de la industria.

De este tipo de investigaciones realizadas en la Universidad Técnica de Ambato se puede mencionar los siguientes trabajos.

Espinoza [7], implementa un sitio web escrito con el lenguaje de programación PHP, que a través de un modelo de programación lineal permite optimizar dietas alimenticias mediante el cálculo de los valores mínimos utilizando el método de la gran M.

Sánchez [8],demuestra la aplicación de un algoritmo matemático, que soluciona el problema de manejo de inventario, que como consecuencia ha incrementado el costo de los ítems almacenados. Para la creación de este algoritmo, analiza los productos necesarios en la fabricación, el proceso productivo, demanda, capacidad productiva y costos relacionados con el mantenimiento de inventario. Para esto ha hecho uso de la aplicación de software Lingo®, y prueba que ha generado un ahorro para la empresa.

Heredia [9], presenta un modelo de programación lineal entera para la generación automática de horarios de clases. Este algoritmo tiene en cuenta la disponibilidad de profesores, y aulas, además del pensum de los estudiantes. La investigación se aplica en la planificación de horarios en la Facultad de Ciencias de la Escuela Politécnica Nacional.

Artieda [10], desarrolla un modelo de optimización de tarifas de energía eléctrica, en relación con la demanda y costos en cada una de las etapas necesarias para la entrega del servicio a consumidores finales. Como objetivo se tiene realizar el cálculo de tarifas óptimas a ser aplicadas a clientes de distintos estratos.

Villarreal et al. [11], proponen la generación de un modelo de programación lineal de asignación de horarios para una entidad bancaria con la metodología simplex, cuyo objetivo es optimizar recursos. El modelo matemático proporciona el número mínimo requerido de personal para una sucursal con la función de minimización, para el personal de caja de tiempo completo.

Gómez [12], estudia el método simplex y su variante de las dos fases, las utiliza para optimizar varias funciones simultáneamente. Estos análisis se comprueban usando el software Xpress con varios problemas de prueba.

Bernabé et al. [13], presentan el análisis de sensibilidad de un problema de programación lineal continua de asignación de recursos. En este trabajo desarrollan un programa que, de forma iterativa, a través de una serie de modificaciones a la función objetivo y las restricciones puede arrojar un análisis de sensibilidad.

Flores et al. [14], aplican el método simplex de programación lineal, con el propósito de obtener la máxima utilidad en la venta de los productos de una empresa mobiliaria. El modelo matemático se procesa usando el software Solver de Microsoft Excel, y obtiene la solución óptima. En este estudio demuestran que la programación lineal resulta útil para contrastar las hipótesis que contribuyen a una mejor toma de decisiones institucionales, aún cuando son escenarios complejos como el de la crisis de la COVID-19.

Kayode et al. [15], aplican técnicas de programación lineal para determinar la cantidad

de pan que se debe producir en un día, para maximizar las ganancias, sujetándose a limitaciones del proceso de producción. Los datos consisten en los ingredientes mayoritarios usados para la producción del pan. El problema formulado en términos matemáticos se ha solucionado con el programa LIPS.

1.3 Fundamentación teórica

1.3.1 Aplicación multiplataforma

Una aplicación multiplataforma, es creada con el fin de que un solo código fuente, pueda ejecutarse en varias plataformas con cambios mínimos en este código. La plataforma puede hacer referencia al tipo de procesador o hardware sobre el cual se ejecuta la aplicación tales como arquitecturas x86, x64, ARM32 y ARM64; además de este tipo de arquitectura se puede mencionar la de software lo cual hace referencia a sistemas operativos, entornos de programación y máquinas virtuales, tales como Android, Java, Linux, Microsoft Windows, Solaris, iOS entre muchos otros [16]. Por lo tanto, presenta la ventaja de reducir el tiempo de desarrollo y mantenimiento.

1.3.2 Aplicación de código abierto

La interoperabilidad masiva trae beneficios tanto sociales, técnicos y financieros. La participación inclusiva de cualquier agente dispuesto a seguir los protocolos publicados tiene la capacidad de crear algo de amplia usabilidad y valor. Los “estándares abiertos” tienen como objetivo habilitar el potencial de los consumidores y proveedores de tecnología, en la cual se puede invertir sin temor de litigios, obligación de pago, o infringir derechos de autor [17].
Software de código abierto además de acceso al código, debe cumplir con los criterios:

- Libre distribución.
- Trabajo derivado.
- Integridad de código abierto del autor.

- No discriminar personas o grupos.
- No discriminar el campo de esfuerzo.
- Distribución de la licencia.
- Licencia no específica a un producto.
- Licencia no restringe otro software.
- La licencia neutral desde el punto de vista tecnológico [18].

1.3.3 Licencia de código abierto

Una licencia de código abierto define las responsabilidades que toman quienes hacen uso de software bajo esa licencia. Además, brinda protección legal para el autor, y permite tomar acción legal en contra de quienes infringen los términos de esta licencia [19].

En otras palabras, una licencia sirve de contrato entre el autor de una aplicación de software y el usuario final, en esta se definen los derechos y obligaciones de las dos partes. El autor, o a quienes hayan sido cedidos los derechos de explotación, será quien elige, que bajo que licencia se libera el software [20].

Las licencias de software de código abierto se pueden categorizar en: Licencias permisivas, usualmente permiten que los usuarios realicen casi cualquier cosa, sin necesidad de republicar los cambios, o atribuir al autor original. Licencias Copyleft requieren que quien sea que cambie el código fuente, deba liberar el código, y usualmente piden que se atribuya al autor original del código fuente, cuando este se vuelva a publicar [19].

1.3.4 Framework Flutter

Google lanzó Flutter en versión Alpha v0.0.6, el 12 mayo del 2017, que permite a los desarrolladores escribir aplicaciones para Android y iOS utilizando Dart.

Posteriormente el 04 de diciembre lanzó de forma oficial la versión estable 1.0 de Flutter [21].

Flutter es kit de herramientas de interfaz de usuario, que permite desarrollar, probar y desplegar aplicaciones multiplataforma, haciendo uso de un solo código fuente [22]. Flutter, a diferencia de la mayoría de las demás herramientas, construye las interfaces de forma declarativa. Esto significa que el desarrollador no necesita concentrarse en la programación de transiciones entre los diferentes estados de la interfaz, sino que les permite describir el estado posterior de la interfaz, y Flutter se encarga de la transición desde el estado actual [23].

Flutter ha sido diseñado como un sistema extensivo por capas. Existe como una serie de librerías independientes que dependen de la capa inferior. Cada parte de los niveles del framework han sido diseñados para ser opcionales y reemplazables (*Ver figura 1.1*).

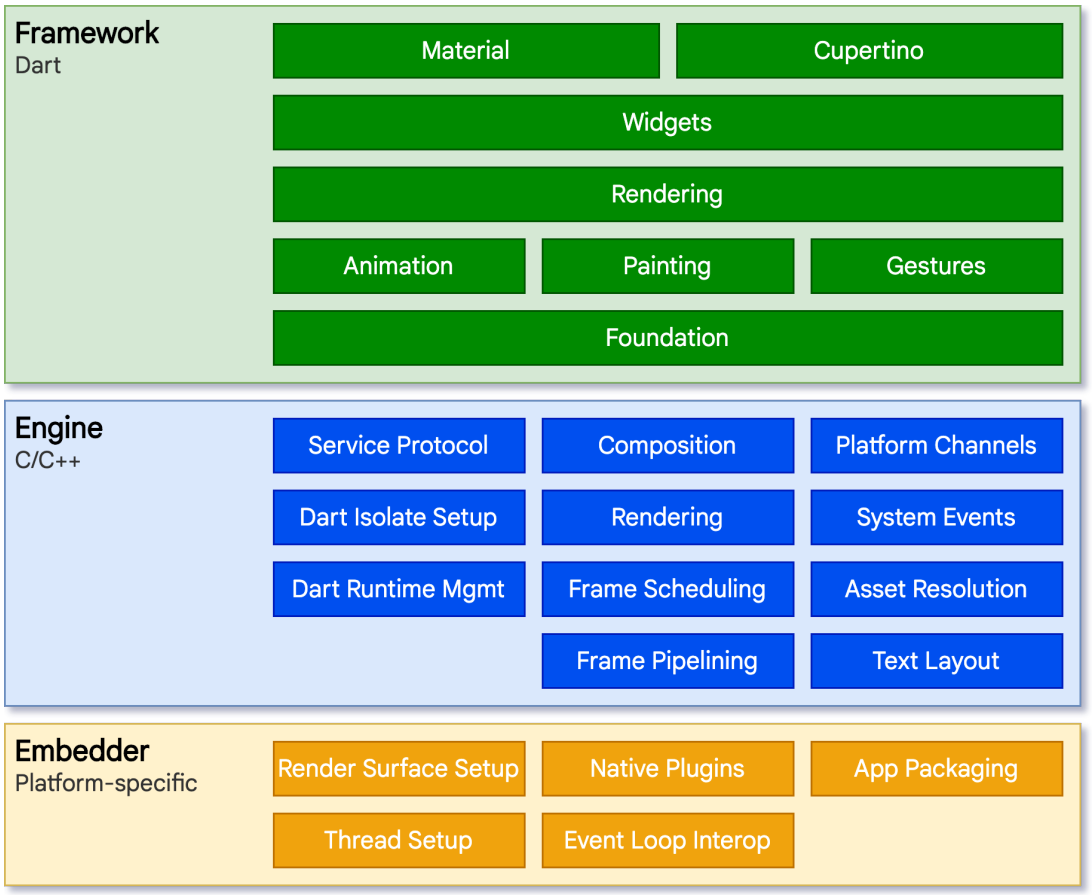


Figura 1.1: Capas de arquitectura Flutter [23].

Para el sistema operativo, las aplicaciones desarrolladas con Flutter aparecen como cualquier otra aplicación nativa. El motor de Flutter, está en su mayoría escrito en C++ y soporta las primitivas necesarias. Este motor es el encargado de rasterizar las escenas compuestas y hace uso del motor de gráficos Skia [23]. Todo esto en conjunto ayuda al obtener el máximo rendimiento del dispositivo sin importar el sistema operativo.

1.3.5 Lenguaje de programación Dart

Dart fue creado por miembros de la versión 8 de JavaScript, esto con el ánimo de solucionar varios de los problemas que JavaScript presentaba, un lenguaje con más de 20 años [24]. Dart es un lenguaje de programación de código abierto, tipado, orientado a objetos, fue lanzado por primera vez en 2011.

Este lenguaje es muy versátil y puede usarse en diferentes aplicaciones como: Aplicaciones web, en servidores [25], pero destaca en aplicaciones en el lado del cliente [21]. Dart combina chequeo estático en tiempo de compilación y de ejecución, para lograr un sistema de tipos completo, garantizando que las expresiones de un tipo no puedan producir un valor de otro tipo [26].

1.3.6 Appwrite

Appwrite inició oficialmente en septiembre de 2019. Inició como un proyecto personal de Eldad Fux. El proyecto fue creado para resolver sus problemas personales como líder tecnológico e ingeniero de software. Desde su origen, Appwrite ha sido diseñado para actuar como una capa de abstracción consistente y predecible sobre las soluciones de la nube tradicionales. Appwrite proporciona a los desarrolladores todas las API fundamentales para construir proyectos de software modernos, basándose en protocolos existentes [27]. Appwrite se construye con la colaboración de la comunidad de código abierto, y se ha convertido en una plataforma de backend para el desarrollo de aplicaciones web, móviles nativas y de Flutter [28].

- **Base de datos:** Almacena, consulta y gestiona el acceso a los datos en tiempo real con una base de datos robusta y escalable.

- **Autenticación:** Gestión a los usuarios con una variedad de métodos de autenticación y proveedores de OAuth.
- **Almacenamiento:** Guarda y sirve archivos con compresión y encriptación incorporadas.
- **Funciones:** Funciones sin servidor que se escalan según la demanda [29].

1.3.7 Google Firebase

Firebase fue inicialmente desarrollada por Evolve en el 2011, posteriormente en 2014 fue adquirido por Google Inc. Google Firebase provee varias herramientas para el desarrollo de aplicaciones [30].

- **Firebase Auth:** Es un servicio que permite autenticar usuarios con correo y contraseña, número telefónico e integrar proveedores de autenticación de terceros como Facebook, Google y varios otros. Facilita que la autenticación suceda de forma segura.
- **Firebase Cloud Firestore:** Es una versión mejorada de la base de datos no relacional de tiempo real de Google, que guarda los datos en forma de pareja "llave y valor" en documentos y colecciones de documentos.
- **Firebase Storage:** Este es el servicio para almacenar archivos en el servidor, acepta archivos de varios formatos como imágenes, videos, archivos. Además, provee un gestor de red para la carga y descarga de los archivos, además puede realizarse de forma programada.
- **Firebase Hosting:** Provee un servicio básico de alojamiento para sitios web estáticos. Este servicio incluye el protocolo Hypertext Transfer Protocol Secure (HTTPS) y Secure Sockets Layer (SSL).

1.3.8 Metodología de desarrollo

a. Extreme Programming

Extreme Programming (XP) fue creado por Kent Beck a finales de 1990 [31]. XP es una metodología de desarrollo de software que se enfoca en la óptima aplicación de técnicas de programación, buena comunicación y trabajo en equipo. Es mejor aplicada por equipos de cualquier tamaño, especialmente cuando los requerimientos no son claros o tienden a cambiar rápidamente.

XP intenta reconciliar humanidad con productividad y tiene como paradigma el mantenerse atento, adaptarse y generar cambio. XP se basa en cinco valores: comunicación, simplicidad, retroalimentación, valentía y respeto [32].

- ***Roles asociados con Extreme Programming:***

  - Cliente – Es el responsable de tomar las decisiones de negocio con respecto al proyecto. Esto incluye las características que debe tener el sistema, los criterios de aceptación, que recursos estarán disponibles y el orden en que se desarrollan las características.

  - Desarrollador – Dada la baja necesidad de un sistema de roles en XP, todos en el equipo a excepción del cliente y los roles listados abajo, son considerados como desarrolladores. Los desarrolladores son los responsables de realizar las características identificadas por el cliente.

  - Tracker – El propósito principal de este rol es dar seguimiento a las métricas que se decidan como necesarias para identificar áreas de mejora y medir el progreso.

  - Coach – Este rol es usualmente asignado a un consultor o alguien fuera del equipo, que tiene experiencia previa con XP. Este rol está diseñado para que sirva de mentor a los demás miembros del equipo en las mejores prácticas de XP [33].

- ***Ciclo de vida del método Extreme Programming***

  Primero, los usuarios definen las historias de usuario, que describen los resultados deseados del proyecto. Posteriormente se estima el tamaño de cada historia. Esto sirve para estimar la prioridad de cada historia de usuario. Si una de las historias no puede ser estimada, porque el equipo carece de conocimientos suficientes, se debe realizar una investigación enfocada para esa historia en particular.

  Consecuentemente todo el equipo debe reunirse para crear un plan razonable. Este plan indicará que historias de usuario que se entregarán, y en que trimestre o lanzamiento. Posteriormente el equipo se dedicará a realizar ciclos semanales, y al inicio de cada ciclo el equipo, se reúne para definir en que historias de usuario se trabajaran esa semana.

  Una vez hecho esto las historias de usuario se descomponen en tareas, que se completarán esa semana. Al fin de la semana el equipo y el cliente se reúnen para decidir si el proyecto debe continuar, o si se ha realizado lo suficiente hasta el momento [33].

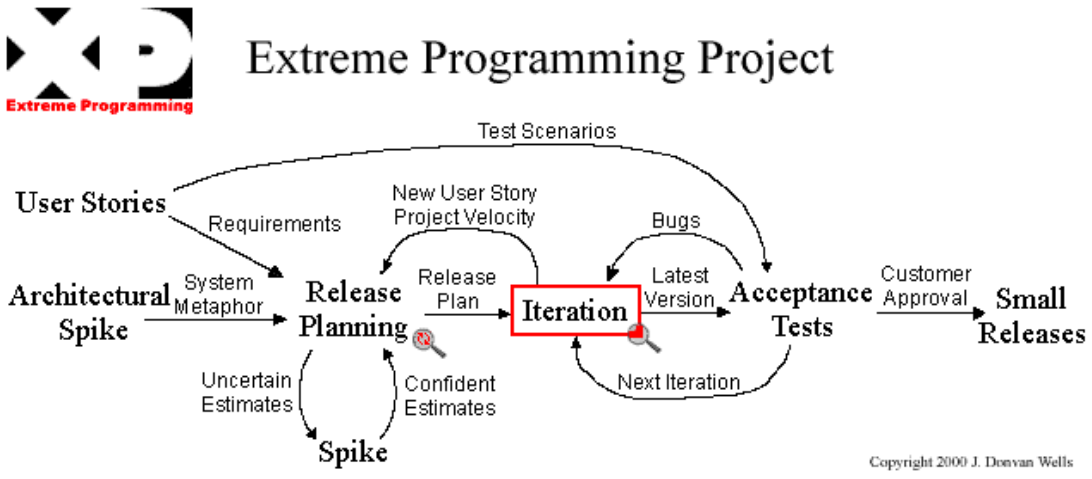


Figura 1.2: Extreme Programming Flow [34].

b. Scrum

La historia de Scrum traza desde 1986, con un artículo publicado en Harvard Bussiness Review [35]. El articulo describe como compañías con Honda y Fuji-Xerox con un enfoque a la escalabilidad y el equipo de desarrollo de productos, es capaz de producir nuevos productos a escala global [36]. Más tarde en 1993, Jeff Sutherland junto a su equipo crearon el proceso Scrum, tomando ideas del artículo previamente mencionado [35].

Scrum es un marco de trabajo con el cual personas pueden abordar problemas complejos, para entregar productos de muy alta calidad, manteniéndose la productividad [37].

Scrum es una metodología de desarrollo ágil usada en el desarrollo de software, que se basa en el proceso iterativo e incremental. Está basado en un conjunto de prácticas y roles, aplicando los 12 principios ágiles [35].

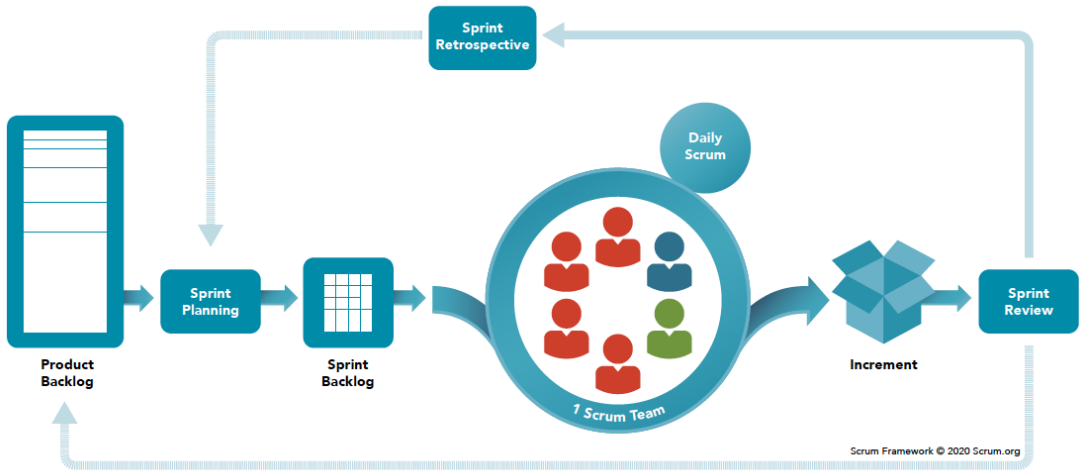


Figura 1.3: Proceso Scrum [37].

La unidad fundamental de Scrum, es un pequeño equipo, este equipo consiste en un Scrum Master, un Product Owner y los desarrolladores [37].

- Scrum Master: Es la persona encargada de guiar al equipo a seguir las reglas y procesos de la metodología. Está encargado de minimizar impedimentos, y entrenamiento a los miembros de su equipo.

- Product Owner: Es el representante, su enfoque está en el lado del negocio del desarrollo. Además de validar las historias de usuario a ser incorporadas.
- Desarrolladores: Es el grupo de profesionales con los conocimientos técnicos para desarrollar el proyecto de forma conjunta [35]. Este consta de 5 a 9 personas [38].

Scrum, usa varios eventos de forma regular, todos estos tienen límite de tiempo. No se puede cambiar la duración de un Sprint una vez que ha comenzado. Los demás eventos terminan una vez que se ha alcanzado el objetivo del evento. Los eventos de Scrum son: Sprint, planeación de sprint, Scrum diario, Revisión de sprint, y Retrospectiva de Scrum [37].

Artefactos de Scrum:

- Product Backlog (PB): Es la lista de todas las necesidades del producto para satisfacer a los usuarios. Ayuda a responder la pregunta, ¿Qué se debe hacer?
- Sprint Backlog: Es un subconjunto de los ítems del PB, seleccionadas por un equipo, para ser realizadas en el transcurso del sprint.
- Incremento: Es la suma de todas las tareas, casos de uso, historias de usuario y cualquier otro elemento que haya sido desarrollado durante el sprint [35].

El marco de trabajo Scrum, es inmutable. “Aunque se implementen ciertas partes de Scrum, el resultado no es Scrum. Scrum existe solo en su totalidad y funciones, así como otras técnicas, metodologías y prácticas” [39].

c. Kanban

El concepto de Kanban fue originalmente parte del sistema de producción Justo a tiempo (JIT) de Toyota en la década de 1950. JIT significa producir "solo lo que se necesita, cuando se necesita y en la cantidad necesaria" [40].

Aunque Kanban fue originalmente creada como una herramienta para maximizar la eficiencia de la producción, ha evolucionado para ser utilizada en diferentes industrias,

y con ciertas adaptaciones se ha convertido en una excelente herramienta de gestión de tareas para equipos de desarrollo de software [41].

Kanban requiere comunicación en tiempo real sobre la capacidad de trabajo. Los elementos de trabajo se representan visualmente en un tablero kanban, lo que permite a los miembros del equipo ver el estado de tarea en cualquier momento [42].

Los equipos de desarrollo de software han encontrado un éxito particular usando la metodología Kanban. Esto se debe en parte a que los equipos de software pueden comenzar a practicar con poco o ningún costo adicional una vez que comprenden los principios básicos de la metodología [42].

La metodología Kanban hace hincapié en planear el trabajo para facilitar la entrega de productos de software justo a tiempo para su implementación. Las organizaciones de todo el mundo están adoptando Kanban e incorporándolo en sus procesos actuales de desarrollo de software para modelar mejor la agilidad empresarial [43].

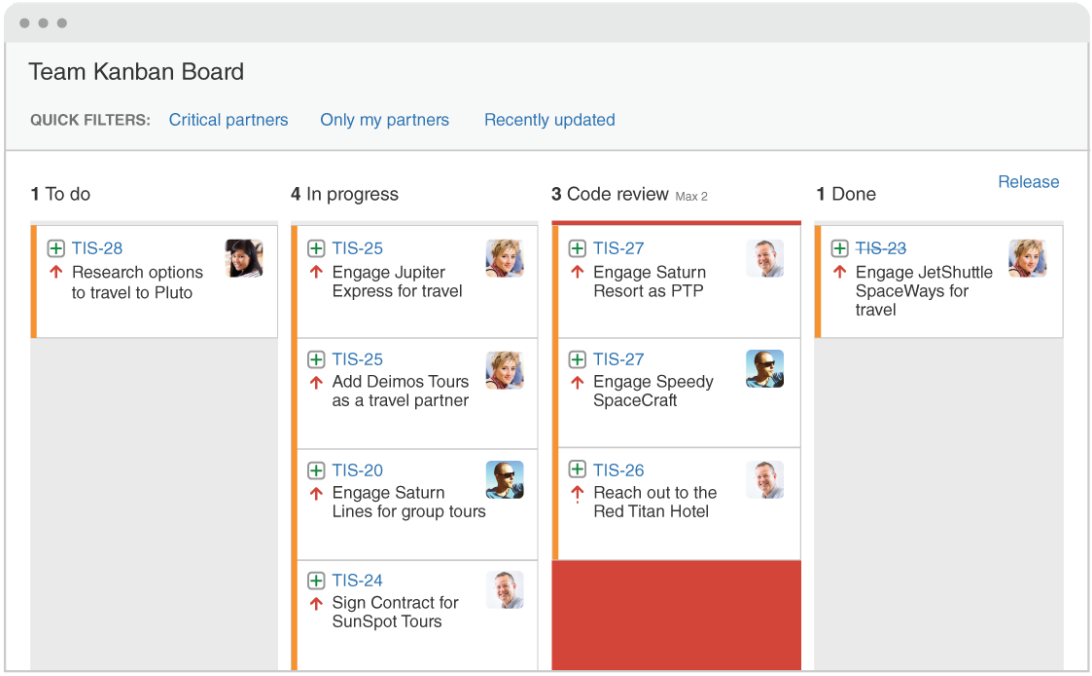


Figura 1.4: Tablero kanban [37].

En la metodología Kanban, existen varios artefactos clave que ayudan en la gestión visual y el seguimiento del flujo de trabajo. Estos artefactos proporcionan transparencia y permiten a los equipos y partes interesadas comprender rápidamente el

estado y progreso de las tareas. A continuación, se describen en detalle los principales artefactos de la metodología Kanban:

- **Tablero kanban:** Es el artefacto principal de Kanban. Consiste en una representación visual del flujo de trabajo en forma de columnas. Cada columna representa una etapa o estado del proceso. El tablero Kanban permite a los miembros del equipo ver de forma panorámica el estado actual de cada tarea y cómo fluye a través del proceso [42].

- **Tarjetas kanban:** Las tarjetas son los elementos individuales de trabajo que se mueven a lo largo del tablero Kanban. Cada tarjeta representa una tarea o una unidad de trabajo específica. Las tarjetas contienen información relevante sobre la tarea, como su descripción, quién ha sido asignado como responsable, la fecha límite y cualquier otro detalle necesario.

- **Límites de columna:** Los límites de columna son restricciones establecidas para limitar la cantidad de tarjetas que pueden estar en una columna en un momento dado. Estos límites evitan la sobrecarga de trabajo y promueven un flujo de trabajo equilibrado. Cuando se alcanzan los límites, se debe resolver o completar el trabajo existente antes de agregar más tarjetas a esa columna.

- **Indicadores visuales:** Los indicadores visuales son elementos adicionales que se agregan al tablero Kanban para proporcionar información adicional sobre el estado y las condiciones del flujo de trabajo. Algunos ejemplos comunes incluyen etiquetas de prioridad, señalizadores de bloqueo, gráficos de rendimiento y métricas clave.

- **Reglas de trabajo:** Las reglas de trabajo son pautas y acuerdos establecidos por el equipo, con el fin de garantizar un flujo de trabajo coherente y eficiente. Estas reglas pueden incluir políticas sobre la asignación de tarjetas, la resolución de bloqueos, la priorización de tareas y cualquier otro aspecto relevante para el proceso.

En conjunto, estos artefactos permiten visualizar y controlar el flujo de trabajo de manera efectiva, fomentando la colaboración, transparencia y mejora continua [43].

1.3.9 Investigación Operativa

Las raíces de la Investigación operativa, se puede referenciar a los primeros intentos de usar un método científico en los problemas técnicos y la gestión de organizaciones. Durante la segunda guerra mundial los limitados recursos militares, apremiaron la necesidad de ubicar estos recursos en las diferentes operaciones militares de forma efectiva [44].

Según Gupta, la investigación operativa es "La investigación de operaciones" con el fin de proveer una solución con base científica a quienes toman decisiones [44].

La investigación operativa se puede definir como el uso de métodos cuantitativos para asistir analistas y quienes toman de decisiones en el diseño, análisis, y mejora del rendimiento u operación de sistemas. Los sistemas bajo estudio pueden ser de cualquier tipo, pero se deben poner bajo el escrutinio de las herramientas del método científico [45].

De acuerdo con Taha, el elemento principal de la investigación operativa es el modelado matemático para establecer una base indispensable para la toma de decisiones [46].

Investigación operativa es un método científico, una actividad organizada para atacar nuevos problemas y encontrar soluciones. Es una ciencia aplicada que utiliza todas las técnicas científicas como herramientas para la solución de problemas específicos [47].

Principios del sistema de modelado:
El proceso de construir modelos matemáticos es el núcleo do la práctica de Investigación Operativa. Un modelo es la simplificación, representación de un objeto, proceso o sistema real. Los modelos matemáticos se construyen de estructuras como las ecuaciones, inecuaciones, matrices, funciones, y operadores. Estas estructuras se usan para describir las características sobresalientes de la entidad a ser modelada [45].

Fases del método de Investigación Operativa

a. Formulación del problema.

   Consiste en identificar, definir y especificar la medida de los componentes del

modelo de decisión. El problema debe ser formulado antes de poder ser susceptible a la investigación. Para la formulación se necesita conocer quién tomará la decisión, cuáles son los objetivos, cuáles son las variables controlables e incontrolables [44].

b. Construcción del modelo matemático.
   En esta fase se reformula el problema, para que resulte más fácil la construcción de un modelo matemático que represente al sistema bajo investigación.

c. Extracción de la solución a partir del modelo.
   Esta fase trata de la búsqueda de la solución óptima, sea que maximice o minimice la función objetiva del modelo.

d. Puesta a prueba del modelo y la solución obtenida a partir del modelo.
   Esta fase intenta revelar cualquier error que pueda haberse cometido. La solución encontrada al usar el método de Investigación de operaciones debería dar mejores resultados que otros procedimientos alternativos. Además, la solución debe ser probada, de tal manera que demuestre que es óptima.

e. Implementación y mantenimiento de la solución.
   La fase final es implementar la solución encontrada. Puesto que la condiciones cambian constantemente, es posible que la solución no sea válida por un largo tiempo, por lo tanto, cuando ocurran cambios se debe actualizar tanto el modelo, la solución y su implementación [44].

1.3.10 Programación Lineal

La programación lineal tiene inicios en 1947 cuando G. B. Dantzig diseñó el método Simplex, usualmente el uso de la programación lineal ha incrementado la eficiencia en las operaciones. Con el incremento de la popularidad de la programación lineal, se encontraron nuevas áreas en la que esta puede ser aplicada. En el núcleo de la teoría matemática de programación lineal, está los sistemas de ecuaciones lineales [48].

De acuerdo con Gupta, el término “programación” se refiere al proceso que determina un plan de acción particular, y el término lineal se refiero a que todas las relaciones de un problema en particular son lineales. Estos problemas usualmente incluyen un set de

ecuaciones lineales simultáneas, las cuales representan las condiciones del problema, y otra función que expresa la función objetivo del problema [44].

Programación lineal es una clase de matemáticas especial, donde la función objetivo y las condiciones se expresan como funciones lineales de las variables de decisión. El uso de modelos de programación lineal en sistemas de optimización sucede en una gran variedad de aplicaciones [45].

La programación lineal según Taha produce algoritmos eficientes para la resolución de los problemas, y forma el eje central para otros modelos de investigación operativa. La programación lineal se aplica a modelos de optimización donde las funciones son lineales y tiene variadas aplicaciones en diferentes industrias [46].

1.4 Objetivos

1.4.1 Objetivo general

Implantar una aplicación multiplataforma de código abierto, para la resolución de problemas de Programación Lineal de Investigación Operativa utilizando el framework Flutter.

1.4.2 Objetivos específicos

- Investigar los diferentes algoritmos de resolución del modelo de Programación Lineal de Investigación Operativa.
- Analizar el framework Flutter y sus características como herramienta de desarrollo para la creación de aplicaciones multiplataforma.
- Desarrollar una aplicación multiplataforma, que no dependa de una conexión a internet u otros programas, que permita la resolución de ejercicios de programación lineal.
- Validar la aplicación multiplataforma de código abierto a través de la resolución de ejercicios de programación lineal de fuentes bibliográficas y científicas.

CAPÍTULO II.- METODOLOGÍA

2.1 Materiales

2.1.1 Fichas de Contenidos

Las fichas de contenido representan un elemento fundamental en un proyecto de investigación, sirviendo como material clave para el registro y análisis sistemático de información relevante. Estas fichas, favorecen a la captura de datos específicos de fuentes primarias y secundarias, facilitan la revisión y síntesis de la literatura pertinente al tema de estudio. Además, al ofrecer un formato estandarizado [49], las fichas de contenido promueven la coherencia y claridad en la recopilación de datos, lo que permite a los investigadores gestionar eficazmente la gran cantidad de información obtenida durante el proceso de investigación. Estas contribuyen significativamente a la calidad y credibilidad de los resultados obtenidos.

Según Romero [49], las fichas de contenidos, constan de tres elementos:

1. **Encabezado:** se muestran en orden jerarquico y pueden tener relación directa con el esquema de investigación.
2. **Referencia:** sirve para mostrar los datos que han de permitir reconocer la fuente del contenido de la ficha, en la que pueden constar los apellidos del autor, y páginas consultadas.
3. **Contenido:** se refiere a la información que se ha recopilado de la fuente, y que se considera relevante para el tema de investigación.

En el presente proyecto de investigación, las fichas de contenido son utilizadas como herramientas organizacionales, para registrar información relevante de fuentes bibliográficas. Su propósito principal es facilitar la recopilación sistemática de datos, permitiendo registrar detalles clave de los datos bibliográficos y posibles observaciones pertinentes.

2.2 Métodos

2.2.1 Modalidad de la Investigación

a. Documental-bibliográfica

La investigación del presente documento consiste también en la lectura y análisis de material bibliográfico de fuentes académicas tales como tesis, libros, artículos científicos, y revistas especializadas, para la recopilación de información y datos que asistan en la óptima solución del problema.

b. De campo

Para la aplicación de esta modalidad se emplearon técnicas e instrumentos de recopilación de información, principalmente en el proceso de validación del sistema multiplataforma de código abierto para resolución de problemas de programación lineal utilizando el framework Flutter

c. Investigación aplicada

Es también una investigación aplicada, ya que se usará los conocimientos recopilados en la construcción de una aplicación multiplataforma, tomados de los resultados y conclusiones de investigaciones anteriores, para la resolución de problemas de investigación operativa con programación lineal.

2.2.2 Recolección de Información

Inicialmente, se presentan los hallazgos derivados de la aplicación de la modalidad de investigación documental y bibliográfica. Dichos resultados corresponden al proceso de recopilación de información, y se exponen en las siguientes fichas de contenidos.

FICHA DE CONTENIDOS
TEMA: Método gráfico, programación lineal. *Tipo de fuente:* Libro.
[Fuente: Hillier [50], 2010, p. 29] Según Hillier un problema que tiene dos variables de decisiones tiene por lo tanto solo dos dimensiones. El procedimiento envuelve la construcción de un gráfico de dos dimensiones donde x_1 y x_2 son sus ejes. Para iniciar se debe notar que las condiciones de no negatividad reposan en el lado positivo de los ejes. La región resultante de los puntos permitidos se llama la región factible. El paso final es elegir el punto en la región factible que optimice el valor de la función objetivo.
[Fuente: Carter et al [51], 2018, p. 30, 31, 32] El espacio factible del problema es el conjunto de todos cuyos puntos satisfacen las restricciones del problema. El espacio factible es por lo tanto el conjunto de todas las soluciones factibles. Una solución óptima factible es un punto en el espacio factible que es tan efectivo como cualquier otro punto en alcanzar una meta especifica.

Las solución de problemas de programación lineal, con solo dos variables de decision se pueden ilustrar de forma gráfica. Existen situaciones en las cuales no existe una región factible o tinen varias soluciones óptimas.
[Fuente: Taha [46], 2017, p. 47, 48] La solución gráfica incluye: determinar el espacio de la solución factible, determinar la solución óptima de entre los puntos del espacio factible. La condición de no negatividad restringe a las variables al primer cuadrante. Cada inecuación se debe reemplazar con una ecuación que se gráfica ubicando dos puntos para forma una linea recta. Es necesario usar un procedimiento sistemático para determinar la solución óptima cundo los puntos de solución en la región factible son infinitos.
[Fuente: Stacho [52], 2014, p. 11] Si un programa lineal tiene una solución óptima, entonces también tiene una solución óptima que es un punto de esquina de la región factible La región factible se construye de lineas que vienen de inecuaciones
[Fuente: Garrido, 1993, p. 31, 38, 40] Consiste en obtener geométricamente la solución de programación lineal. La solución es el punto de tangencia entre un hiper plano de la función objetivo un conjunto de oportunidades El hiper plano toma el mayor valor en la solución si el problema es de maximizaron y si es de minimización sera el menor valor.
[Fuente: Flores [14], 2021, p. 23, 24]

Según Flores, citando a otros autores: "En la función objetivo, según Moya (2003, p. 66), debe determinarse la cantidad que se va a optimizar y expresarla como función matemática."
[Fuente: Wayne [53], 2004, p. 56, 58] Cualquier programa linear con solo dos variables se puede resolver gráficamente. Para que un punto (x_1, x_2) sea una solución factible, debe satisfacer todas las restricciones del problema. El valor óptimo de z se puede encontrar sustituyendo los valores de x_1 y x_2 en la función objetivo.
[Fuente: Wayne [53], 2004, p. 132] Cualquier solución básica en la que todas los variables son no negativas, es una solución básica flexible. Cuando una solución básica factible no esta disponible, el método de las dos fases puede ser usado.

Tabla 2.1: Ficha de contenidos - Método Gráfico

FICHA DE CONTENIDOS
TEMA: Método algebraico , programación lineal. *Tipo de fuente:* Libro.
[Fuente: Garrido, 1993, p. 14] Los mismos problemas que se resuelven con el procedimiento gráfico se pueden tratar con operaciones algebraicas. El problema se debe plantear en forma estándar, y se utilizan las variable de holgura para transforma las desigualdades en igualdades.

"La solución necesariamente se tiene que dar en un vértice o en la combinación lineal convexas de varios de ellos."
[Fuente: Stacho, 2014, p. 17, 18] En un problema con n variables y m restricciones, una solución donde por lo menos $(n - m)$ variables son cero es una solución básica. Las soluciones básicas son precisamente los puntos esquina de la región factible.
[Fuente: Hillier [50], 2010, p. 36] La solución óptima, es la solución factible que tiene el valor más favorable de la función objetivo. El método algebraico es el fundamento del algoritmo iterativo Simplex. Los problemas de programación lineal siempre involucra encontrar la mejor *mezcla de niveles de actividad*

Tabla 2.2: Ficha de contenidos - Método Algebraico

FICHA DE CONTENIDOS
TEMA: Método simplex, programación lineal. *Tipo de fuente:* Libro.
[Fuente: Hillier [50], 2010, p. 93, 98] El método simplex es un procedimiento algebraico, aun que sus conceptos subyacentes son geométricos. Lo primero sera convertir las restricciones en forma de inecuaciones a restricciones en forma de igualdad, introduciendo variables de holgura. La forma aumentada del modelo simplex es mas conveniente para la manipulación algebraica.

Una solución básica factible es una soluciona aumentada Punto de esquina factible (*del inglés:* Corner Point Feasible) (CPF)

El numero de variables básicas es igual al numero de restricciones funcionales.

[Fuente: Carter et al [51], 2018, p. 36, 38, 39, 40]
En preparación para el uso del método simplex es necesario es necesario expresar el problema de programación lineal en forma estándar.

Para convertir un problema de minimización a maximizacion se puede simplemente multiplicar la función objetivo por -1, y maximizar esta función.

El lado negativo derecho de la restricción se puede hacer positivo multiplicando la restricción por -1

Las restricciones de ecuaciones no requieren modificaciones

El método simple con algunas modificaciones para eficiencia se ha convertido en el método estándar para resolver problemas de programación muy grandes en computadora.

El método simplex es un algoritmo iterativo que continuamente se mueve a una solución mejor y se detiene cuando no se puede mejorar

El método de las dos fases se resume en: crear una nueva función objetivo que consiste en la suma de las variables artificiales donde simplex minimiza esta función, si se satisface las restricciones originales se inicia la fase dos.

[Fuente: Taha [46], 2017, p. 103, 104, 105, 111]
El método simplex investiga solo unos poco puntos con respecto a todas las soluciones básicas

En aras de estandarizar el algoritmo simplex, este siempre empieza en el origen, donde todas las variables de decisión son cero.

El diseño del método simples no permite incrementara simultáneamente las variables sino que solo una a la vez la variable que se incrementa sera la que tiene la mayor taza de mejora en Z

La solución óptima se alcanza en la iteración donde todos los coeficientes de la fila Z son no negativos.

Tanto para problemas de maximizaron y minimización la variable que sale es la variable básica asociada con la tasa mas pequeña no negativa.

El método de las dos fases inicia intentado encontrar una solución factible básica, si la encuentra, la segunda fase intenta resolver el problema original

[Fuente: Stacho [52], 2014, p. 17, 18]

En un problema con n variables y m restricciones, una solución donde por lo menos $(n - m)$ variables son cero es una solución básica.

Las soluciones básicas son precisamente los puntos esquina de la región factible

[Fuente: Garrido [54], 1993, p. 31, 38, 40]

"Como todo algoritmo iterativo, necesita un punto de partida que es la llama solución factible básica inicial"

Para construir la tabla simplex se colocan las todas las variables en sus columnas de forma ordenada.

La ultima columna recoge los componentes del vector solución.

Los procedimientos pueden corresponder a solución de vértice, solución de arista o falta de solución El método de las fases consiste en descomponer el problema ampliando en dos subproblemas, la primera optimiza una función auxiliar y las segunda el problema original
[Fuente: Wayne [53], 2004, p. 132] Cualquier solución básica en la que todas los variables son no negativas, is una solución básica flexible. Cuando una solución básica factible no esta disponible, el método de las dos fases puede ser usado.

Tabla 2.3: Ficha de contenidos - Método Simplex

En segundo lugar, como consecuencia de la modalidad de investigación de campo, en la que se emplearon técnicas e instrumentos de recopilación de información, se realizó el proceso de validación del sistema multiplataforma de código abierto para resolución de problemas de programación lineal utilizando el framework Flutter, a través de la resolución de ejercicios de programación lineal de fuentes bibliográficas y científicas. Estos resultados se presentan como cumplimiento del cuarto objetivo específico.

2.2.3 Procesamiento y Análisis de Datos

Posterior a la investigación de diversas fuentes bibliográfica pertinentes para el proyecto, se llevó a cabo una lectura crítica para identificar conceptos clave, ideas centrales, conclusiones y aportes de cada fuente en relación con el trabajo realizado.

Se elaboraron fichas de contenido que incluyen información relevante para comprender y contrastar los conceptos que contribuyeron al desarrollo del proyecto. Estas fichas contienen detalles sobre cada autor y aspectos esenciales de sus propuestas, lo que enriquece los conocimientos necesarios para implementar los algoritmos en el

software.

Este proceso permitió obtener un mayor entendimiento y profundidad en el análisis de la información recopilada, destacando lo siguiente:

El método gráfico es un algoritmo, que permite apreciar de forma visual la función objetivo, restricciones y solución óptima de un problema de programación lineal, mediante 2 pasos [46]. Este método es muy útil para el aprendizaje, y se utiliza en problemas de dos variables, pero resulta limitado en problemas con un número mayor de variables [51].

El método algebraico consiste en utilizar ecuaciones y desigualdades para expresar las restricciones y la función objetivo del problema. Este método se utiliza principalmente en la enseñanza de la programación lineal y en problemas de menor complejidad [54].

El método simplex es el algoritmo más utilizado en la resolución de problemas de programación lineal, especialmente en problemas del mundo real, por su eficiencia [50]. Este método utiliza una serie de iteraciones para encontrar la solución óptima del problema, a través de la identificación de variables básicas y no básicas y el cálculo de las variables de decisión óptimas [51].

A partir de lo expuesto se puede afirmar que queda en evidencia la investigación, con respecto a los diversos algoritmos utilizados para la resolución de modelos de Programación Lineal en el campo de la Investigación Operativa.

Además, para comprobar el funcionamiento adecuado del sistema multiplataforma, en la etapa de validación de la aplicación, se seleccionarán varios ejercicios de programación lineal, de fuentes bibliográficas para su posterior resolución. Los datos de cada uno de los problemas se procesaran en el sistema para obtener sus respuestas. Los resultados obtenidos del sistema multiplataforma serán mostrados, junto a las respuestas de los ejercicios de sus respectivas fuentes.

CAPÍTULO III.- RESULTADOS Y DISCUSIÓN

3.1 Algoritmos de resolución del modelo de Programación Lineal

3.1.1 Forma estándar del modelo de programación lineal

Existe una gran variedad de aplicaciones que pueden ser modeladas con programación lineal. El objetivo se establece en minimizar o maximizar una función en la forma [51]:

$$Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n = \sum_{i=1}^{n} c_ix_i$$

Para problemas de programación lineal con n variables y m restricciones, la forma estándar es:

$$\begin{aligned}
\text{maximizar} \quad & Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
\text{sujeto a} \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
& \vdots \qquad\qquad\qquad\qquad \vdots \quad \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m
\end{aligned}$$

3.1.2 Método Gráfico

Los problemas de programación lineal con solo 2 variables se pueden ilustrar de forma gráfica. La región factible del problema es el conjunto de puntos que logran satisfacer las restricciones. Lo que hace que todas las posibles soluciones sean parte de la región factible [51]. Cuando se resuelve uno de estos problemas utilizando el método gráfico, se pueden distinguir tanto la región factible, como las soluciones que la componen.

Para encontrar la solución utilizando el método gráfico se deben seguir 2 pasos [46].

1. Determinar la región factible de la solución.

2. Determinar la solución óptima de entre todos los puntos que confirman la región factible.

Para ilustrar la resolución de un problema de programación lineal de maximización, se toma un ejercicio en [50]: Se desea encontrar los niveles óptimos de dos actividades que compiten por recursos. Ventanas de marcos de madera (x_1), y las de marcos de aluminio (x_2).

La tabla a continuación describe los demás datos del problema.

Recursos por actividad			
Recurso	**Marco de Madera**	**Marco de Aluminio**	**Cant. disponible**
Cristal	6	8	48
Madera	0	1	4
Aluminio	1	0	6
Beneficio c/u	\$180	\$90	

Tabla 3.1: Datos del problema de marcos de ventanas [50]

Para formular el modelo matemático se asigna:

$$x_1 = \text{Cada marco de madera}$$
$$x_2 = \text{Cada marco de aluminio}$$
$$Z = \text{Beneficio total de producir estos productos}$$

Así, x_1 y x_2 son las *variables de decisión* del modelo. Usando la última fila de la Tabla 3.1 se obtiene:

$$Z = 180x_1 + 90x_2$$

El objetivo es encontrar valores para x_1 y x_2 para maximizar $Z = 180x_1 + 90x_2$, sujeta a las restricciones impuestas por los recursos disponibles. Estas restricciones se pueden expresar como:

$$\begin{aligned} 6x_1 + 8x_2 &\leq 48 \\ x_1 &\leq 6 \\ x_2 &\leq 4 \\ \text{y} \quad x_1, x_2 &\geq 0 \end{aligned}$$

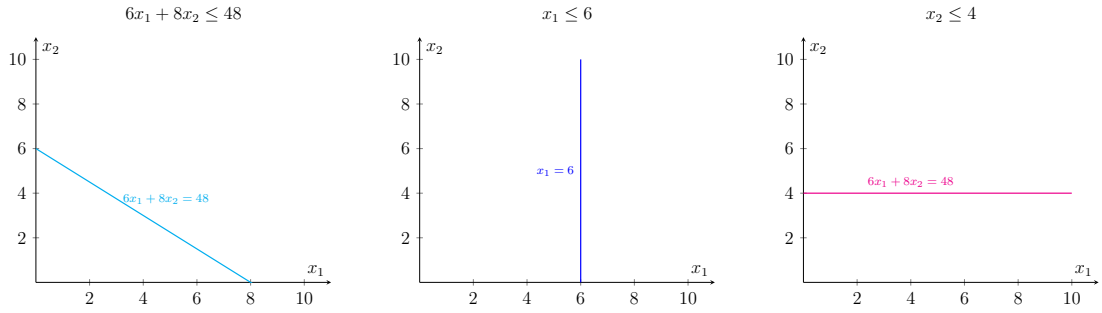


Figura 3.1: Restricciones del problema de marcos de ventanas por separado

Una vez se han realizado los gráficos de las restricciones en el plano cartesiano, se conoce la región factible, sobre la cual se debe encontrar el punto que maximiza el valor de $Z = 180x_1 + 90x_2$. Con esto se conocen las *posibles soluciones*. Una de las formas de encontrar la *solución óptima* es reemplazar los valores de x_1 y x_2 con los valores de las restricciones, donde estas se interceptan en el gráfico, como se muestra en la figura 3.2.

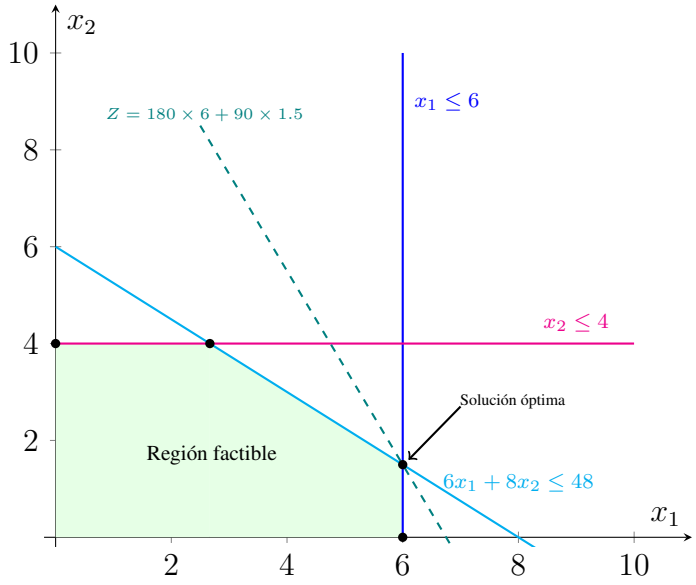


Figura 3.2: Solución óptima del problema de marcos de ventanas

En la figura 3.2 se muestra la región factible, donde se encuentran las posibles soluciones, y la solución óptima, que es el punto donde se maximiza el valor de Z. En este caso, la solución óptima es el punto $(6, 1.5)$, así:

$$Z = 180(6) + 90\left(\frac{3}{2}\right) = 1215 \tag{3.1}$$

Para ilustrar la resolución de un problema de programación lineal de minimización, se toma un ejercicio en [55], con el siguiente enunciado: Una Empresa dedicada a criar caballos, ha establecido que debe suministrar diariamente a cada uno, un mínimo de 200mg. de vitamina A, un mínimo de 160mg. de vitamina B y un mínimo de 150mg. de vitamina C. Los caballos se alimentan con matas de pasto y mineral. Cada mata de pasto cuesta $300 y cada libra de mineral cuesta $500. Se conoce que una mata de pasto contiene 4mg. de vitamina A, 3mg. de vitamina B y 5mg. de vitamina C. Por otro, una libra de mineral contiene 5mg. de vitamina A, 8mg. de vitamina B y 3mg. de vitamina C. ¿Qué cantidad de cada alimento se debe suministrar a cada caballo diariamente para reducir al mínimo el costo de alimentación, manteniendo el suministro de vitaminas descrito?

Alimentos para caballos			
Vitamina	**Pasto**	**Mineral**	**Mínimo requerido**
Vitamina A	4mg	5mg	200mg
Vitamina B	2mg	8mg	160mg
Vitamina C	5mg	3mg	150mg
Costo	$300/mata	$500/libra	

Tabla 3.2: Datos del problema de alimentación de caballos [55]

Para formular el modelo matemático se asigna:

$$x_1 = \text{Matas de pasto}$$

$$x_2 = \text{Libras de mineral}$$

$$Z = \text{Costo mnimo para alimentar a los caballos}$$

Así, x_1 y x_2 son las variables de decisión del modelo. Usando la última fila de la Tabla 3.2 se obtiene:

$$Z = 300x_1 + 500x_2$$

El objetivo es encontrar valores para x_1 y x_2 para minimizar $Z = 300x_1 + 500x_2$, sujeta a las restricciones impuestas por la cantidad mínima de vitaminas para cada caballo. Estas restricciones se pueden expresar como:

$$4x_1 + 5x_2 \geq 200$$

$$2x_1 + 8x_2 \geq 160$$

$$5x_1 + 3x_2 \geq 150$$

y $\quad x_1, x_2 \geq 0$

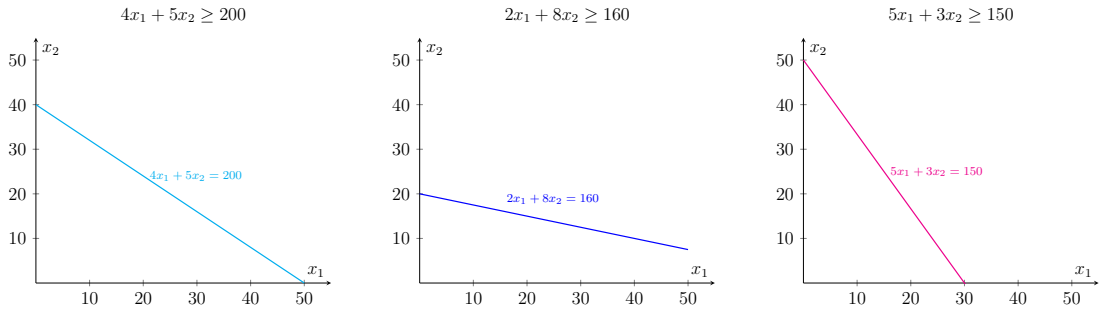


Figura 3.3: Restricciones del problema de alimentación de caballos por separado

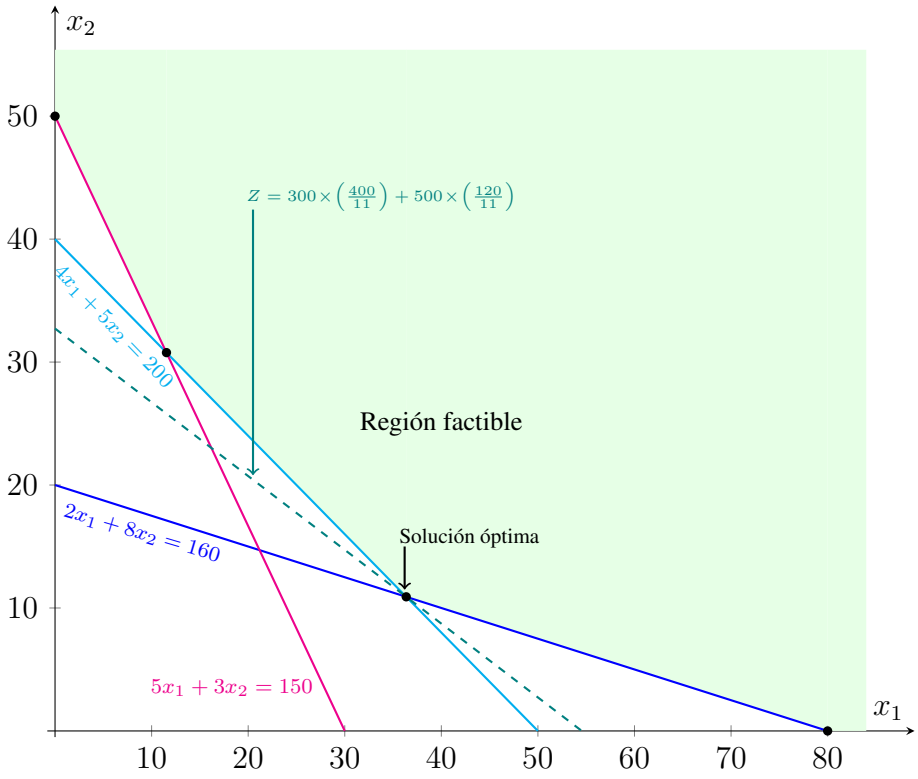


Figura 3.4: Solución óptima del problema de alimentación de caballos

Una vez se han realizado los gráficos de todas las restricciones sobre el plano cartesiano, se conoce la región factible, sobre la cual se debe encontrar el punto que minimiza el valor de $Z = 300x_1 + 500x_2$. La región factible muestra las *posibles soluciones*. Una de las formas de encontrar la *solución óptima* es reemplazar los

valores de x_1 y x_2 donde las restricciones se interceptan, tomando los valores cuando Z es menor, como se muestra en la figura 3.4.

En la figura 3.4 se muestra la región factible, donde se encuentran las posibles soluciones, y la solución óptima, que es el punto donde se minimiza el valor de Z. En este caso, la solución óptima es el punto $(\frac{400}{11}, \frac{120}{11})$, así:

$$Z = 300\left(\frac{400}{11}\right) + 500\left(\frac{120}{11}\right) = \frac{180000}{11} \tag{3.2}$$

3.1.3 Método Algebraico

El método algebraico de es uno de los enfoques matemáticos para resolver problemas de programación lineal. Este implica la aplicación de la teoría de matrices y sistemas lineales. Este algoritmo se enfoca en la manipulación de ecuaciones lineales para determinar la solución óptima. Busca encontrar el valor óptimo de una función lineal, sujeto a un conjunto de restricciones lineales. Para obtener las soluciones factibles del sistema y finalmente la solución óptima.

Para encontrar la solución utilizando el método algebraico se deben seguir los pasos [54].

1. Plantear el problema en forma estándar.
2. Obtener una solución básica; encontrando así un vértice del conjunto de oportunidades.
    - Al tomar una matriz D, formada por m vectores linealmente independientes $(P_1, P_2, \ldots, P_m)$ la solución viene dada por:
    $$P_0 = D^{-1}B$$
    - Donde B es el vector de términos independientes del conjunto de restricciones
3. Resolver las restricciones y la función objetivo donde los valores de las variables no básicas, buscan maximizar o minimizar el valor de la función, según lo exija

el problema. Estas transformaciones también se pueden lograr con operaciones matriciales.

Estos pasos se han de repetir hasta que no sea posible mejorar el valor de la función, con ello se llega a la solución óptima del problema.

Para ilustrar la resolución de un problema de programación lineal, se toma el mismo ejercicio de maximización mostrado en el método gráfico, su datos se muestran en la tabla 3.1.

Una vez se ha formulado la función objetivo y las restricciones, como primer paso del método algebraico se debe pasar el modelo desde la forma canónica a la forma estándar introduciendo una variable de holgura por cada restricción representada por una inecuación, con esto se convierten en igualdades. Los coeficientes nulos se incorporan también en la función objetivo [54].

$$
\begin{array}{lll}
Z = 180x_1 + 90x_2 & & Z = 180x_1 + 90x_2 + 0x_3^h + 0x_4^h + 0x_5^h \\
6x_1 + 8x_2 \leq 48 & & 6x_1 + 8x_2 + x_3^h = 48 \\
x_1 \leq 6 & \rightarrow & x_1 + x_4^h = 6 \\
x_2 \leq 4 & & x_2 + x_5^h = 4 \\
x_1, x_2 \geq 0 & & x_1, x_2, x_3^h, x_4^h, x_5^h \geq 0
\end{array}
$$

Una vez se ha realizado la forma estándar, el problema queda con $n = 5$ variables, $m = 3$ restricciones, $n = 5$ condiciones de no negatividad y $r = n - m = 5 - 3 = 2$. El segundo paso es encontrar el numero de posibles combinaciones, por las que se debe iterar para encontrar la solución óptima.

$$C(n, r) = \frac{n!}{r!(n-r)!}$$

$$C(n, r) = \frac{5!}{2!(5-2)!} = 10 \,\text{Combinaciones}$$

Calculo de la solución óptima.

1. Combinación x_1 y $x_2 = 0$

$$x_3 = 48 \tag{3.3}$$

$$x_4 = 6 \tag{3.4}$$

$$x_5 = 4 \tag{3.5}$$

$$Z = 180(0) + 90(0) = 0$$

2. Combinación x_1 y $x_3 = 0$

$$x_2 = 8 \tag{3.6}$$

$$x_4 = -2 \tag{3.7}$$

$$x_5 = 4 \tag{3.8}$$

$$Z = \text{No factible, en 3.7 el valor es negativo}$$

3. Combinación x_1 y $x_4 = 0$

$$8x_2 + x_3 = 48 \tag{3.9}$$

$$x_3 = indefinido \tag{3.10}$$

$$x_2 + x_5 = 4 \tag{3.11}$$

$$Z = \text{No factible, en 3.10 el valor es indefinido}$$

4. Combinación x_1 y $x_5 = 0$

$$x_2 = 4 \tag{3.12}$$

$$x_3 = 16 \tag{3.13}$$

$$x_4 = 6 \tag{3.14}$$

$$Z = 180(0) + 90(4) = 360$$

5. Combinación x_2 y $x_3 = 0$

$$x_1 = 8 \tag{3.15}$$

$$x_4 = -2 \tag{3.16}$$

$$x_5 = 4 \tag{3.17}$$

$$Z = \text{No factible, en 3.16 el valor es negativo}$$

6. Combinación x_2 y $x_4 = 0$

$$x_1 = 6 \tag{3.18}$$

$$x_3 = 12 \tag{3.19}$$

$$x_5 = 4 \tag{3.20}$$

$$Z = 180(6) + 90(0) \ = \ 1080$$

7. Combinación x_2 y $x_5 = 0$

$$6x_1 + x_3 = 48 \tag{3.21}$$

$$x_3 = indefinido \tag{3.22}$$

$$x_1 + x_4 = 6 \tag{3.23}$$

$$Z = \text{No factible, en 3.22 el valor es indefinido}$$

8. Combinación x_3 y $x_4 = 0$

$$x_1 = 6 \tag{3.24}$$

$$x_2 = \frac{3}{2} \tag{3.25}$$

$$x_5 = \frac{5}{2} \tag{3.26}$$

$$Z = 180(6) + 90\left(\frac{3}{2}\right) = 1215$$

9. Combinación x_3 y $x_5 = 0$

$$x_1 = \frac{8}{3} \tag{3.27}$$

$$x_2 = 4 \tag{3.28}$$

$$x_4 = \frac{10}{3} \tag{3.29}$$

$$Z = 180\left(\frac{8}{3}\right) + 90(4) = 840$$

10. Combinación x_4 y $x_5 = 0$

$$x_1 = 6 \tag{3.30}$$

$$x_2 = 4 \tag{3.31}$$

$$x_3 = -20 \tag{3.32}$$

$$Z = \text{No factible, en 3.32 el valor es negativo}$$

Por último, la solución óptima, es dada por el mayor valor de Z, de entre las 10 combinaciones posibles. Por lo tanto la combinación 8 es la solución óptima, así:

$$Z = 180(6) + 90\left(\frac{3}{2}\right) = 1215 \tag{3.33}$$

Como se indico al inicio del ejercicio, es el mismo mostrado en el método gráfico para maximización, por tanto, la respuesta coincide con la Ecuación 3.1.

Para ilustrar la resolución de un problema de programación lineal de minimización usando el método algebráico, se toma el mismo ejercicio demostrado en el método gráfico, su datos se muestran en la tabla 3.2

$$\begin{aligned} Z =& 300x_1 + 500x_2 \\ 4x_1 &+ 5x_2 \geq 200 \\ 2x_1 &+ 8x_2 \geq 160 \\ 5x_1 &+ 3x_2 \geq 150 \\ x_1, x_2 &\geq 0 \end{aligned} \qquad \rightarrow \qquad \begin{aligned} Z =& 300x_1 + 500x_2 + 0x_3^h + 0x_4^h + 0x_5^h \\ 4x_1 &+ 5x_2 - x_3^h = 200 \\ 2x_1 &+ 8x_2 - x_4^h = 160 \\ 5x_1 &+ 3x_2 - x_5^h = 150 \\ x_1, \quad x_2, \quad x_3^h, \quad x_4^h, \quad x_5^h &\geq 0 \end{aligned}$$

Una vez se ha formulado la función objetivo y las restricciones, como primer paso del método algebraico se debe pasar el modelo, desde la forma canónica a la forma estándar introduciendo una variable de holgura por cada restricción representada por una inecuación, con esto se convierten en igualdades. Los coeficientes nulos se incorporan también en la función objetivo [54]. Una vez se ha realizado la forma estándar, el problema queda con $n = 5$ variables, $m = 3$ restricciones, $n = 5$ condiciones de no negatividad y $r = n - m = 5 - 3 = 2$. El segundo paso es encontrar el número de posibles combinaciones, por las que se debe iterar para encontrar la solución óptima.

$$C(n, r) = \frac{n!}{r!(n-r)!}$$

$$C(n, r) = \frac{5!}{2!(5-2)!} = 10\,\text{Combinaciones}$$

Calculo de la solución óptima.

1. Combinación x_1 y $x_2 = 0$

$$x_3 = -200 \tag{3.34}$$

$$x_4 = -160 \tag{3.35}$$

$$x_5 = -150 \tag{3.36}$$

$$Z = \text{No factible, en 3.34, 3.35 y 3.36 el valor es negativo.}$$

2. Combinación x_1 y $x_3 = 0$

$$x_2 = 40 \tag{3.37}$$

$$x_4 = 160 \tag{3.38}$$

$$x_5 = -30 \tag{3.39}$$

$$Z = \text{No factible, en 3.39 el valor es negativo}$$

3. Combinación x_1 y $x_4 = 0$

$$8x_2 + x_3 = 20 \tag{3.40}$$

$$x_3 = -100 \tag{3.41}$$

$$x_2 + x_5 = -90 \tag{3.42}$$

$$Z = \text{No factible, en 3.41 y 3.42el valor es indefinido}$$

4. Combinación x_1 y $x_5 = 0$

$$x_2 = 50 \tag{3.43}$$

$$x_3 = 50 \tag{3.44}$$

$$x_4 = 240 \tag{3.45}$$

$$Z = 300(0) + 500(50) = 2500$$

5. Combinación x_2 y $x_3 = 0$

$$x_1 = 50 \tag{3.46}$$

$$x_4 = -60 \tag{3.47}$$

$$x_5 = 100 \tag{3.48}$$

$$Z = \text{No factible, en 3.47 el valor es negativo}$$

6. Combinación x_2 y $x_4 = 0$

$$x_1 = 80 \tag{3.49}$$

$$x_3 = 120 \tag{3.50}$$

$$x_5 = 250 \tag{3.51}$$

$$Z = 300(80) + 500(0) = 2400$$

7. Combinación x_2 y $x_5 = 0$

$$6x_1 + x_3 = 30 \tag{3.52}$$

$$x_3 = -80 \tag{3.53}$$

$$x_1 + x_4 = -100 \tag{3.54}$$

$$Z = \text{No factible, en 3.53 y 3.54 el valor es negativo}$$

8. Combinación x_3 y $x_4 = 0$

$$x_1 = \frac{400}{11} \tag{3.55}$$

$$x_2 = \frac{120}{11} \tag{3.56}$$

$$x_5 = \frac{710}{11} \tag{3.57}$$

$$Z = 300\left(\frac{400}{11}\right) + 500\left(\frac{120}{11}\right) = \frac{180000}{11}$$

9. Combinación x_3 y $x_5 = 0$

$$x_1 = \frac{150}{13} \tag{3.58}$$

$$x_2 = \frac{400}{13} \tag{3.59}$$

$$x_4 = \frac{1420}{13} \tag{3.60}$$

$$Z = 300\left(\frac{150}{13}\right) + 500\left(\frac{400}{13}\right) = \frac{245000}{13}$$

10. Combinación x_4 y $x_5 = 0$

$$x_1 = \frac{360}{17} \tag{3.61}$$

$$x_2 = \frac{250}{17} \tag{3.62}$$

$$x_3 = -\frac{710}{17} \tag{3.63}$$

$$Z = \text{No factible, en 3.63 el valor es negativo}$$

Finalmente, la solución óptima, para un problema de minización, es dada por el menor valor de Z, de entre las 10 combinaciones posibles. Por lo tanto, la combinación 8 es la solución óptima en este problema, así:

$$Z = 300\left(\frac{400}{11}\right) + 500\left(\frac{120}{11}\right) = \frac{180000}{11} \tag{3.64}$$

Como se indico al inicio del ejercicio, es el mismo mostrado en el método gráfico para maximización, por tanto, la respuesta coincide con la Ecuación 3.2.

3.1.4 Método Simplex

Este método fue desarrollado por George Dantzig en 1947, capaz de resolver grandes problemas de forma eficiente. Mientras que los subyacentes del método son geométricos, este es un procedimiento algebraico. El desarrollo de los cálculos del método Simplex se facilita imponiendo dos requisitos en el modelo de programación lineal [46].

1. Todas las restricciones han de ser ecuaciones y el lado derecho no puede ser negativo.

2. Ninguna de las variables es negativa.

El método Simplex en vez de enumerar todas las soluciones de un problema de programación lineal, solo investiga "algunas" de estas soluciones [50].

El método Simplex es un algoritmo iterativo que inicia con una solución factible inicial, que progresivamente se mueve hacia una mejor solución, hasta que no puede continuar porque se ha encontrado la solución óptima [51].

El método Simplex de programación lineal, es el más aceptado por su habilidad de modelar problemas complejos de toma de decisiones, y su eficiencia, al lograr resolver problemas en un tiempo adecuado [56].

Para ilustrar la resolución de un problema de programación lineal usando el método Simplex en forma tabular, se hace uso del mismo ejercicio de maximización mostrado

en el método gráfico, su datos se muestran en la tabla 3.1.

Primero se deben pasar la función objetivo y datos del problema a forma tabular. La forma tabular solo mantiene la información más relevante: Los coeficientes de las variables, las contantes al lado derecho de las ecuaciones que representan las restricciones, y las variables básicas que aparecen en cada restricción.

(a) Forma Algebraica	**(b) Forma Tabular**								
	Var. basica	**Eq.**	**Coeficiente**						**Lado der.**
			Z	x_1	x_2	x_3	x_4	x_5	
(0) $Z - 180x_1 - 90x_2 = 0$	Z	(0)	1	-180	-90	0	0	0	0
(1) $6x_1 + 8x_2 + x_3 = 48$	x_3	(1)	0	6	8	1	0	0	48
(2) $x_1 + x_4 = 6$	x_4	(2)	0	1	0	0	1	0	6
(3) $x_2 + x_5 = 4$	x_5	(3)	0	0	1	0	0	1	4

Tabla 3.3: Forma tabular del método simplex.

Var. basica	**Eq.**	**Coeficiente**						**Lado der.**
		Z	x_1	x_2	x_3	x_4	x_5	
Z	(0)	1	-180	-90	0	0	0	0
x_3	(1)	0	6	8	1	0	0	$48 \rightarrow \frac{48}{6} = 8$
x_4	(2)	0	1	0	0	1	0	$6 \rightarrow \frac{6}{1} = 6 \leftarrow \min$
x_5	(3)	0	0	1	0	0	1	4

Tabla 3.4: Aplicación de la prueba de proporción mínima

Iteración	**Var. basica**	**Eq.**	**Coeficiente**						**Lado der.**
			Z	x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-180	-90	0	0	0	0
	x_3	(1)	0	6	8	1	0	0	48
	x_4	(2)	0	1	0	0	1	0	6
	x_5	(3)	0	0	1	0	0	1	4
1	Z	(0)	1	0	-90	0	180	0	1080
	x_3	(1)	0	0	8	1	-6	0	12
	x_1	(2)	0	1	0	0	1	0	6
	x_5	(3)	0	0	1	0	0	1	4

Tabla 3.5: Iteración 1 - Simplex

Iteración	Var. basica	Eq.	Coeficiente						Lado der.
			Z	x_1	x_2	x_3	x_4	x_5	
1	Z	(0)	1	0	-90	0	180	0	1080
	x_3	(1)	0	0	8	1	-6	0	12
	x_1	(2)	0	1	0	0	1	0	6
	x_5	(3)	0	0	1	0	0	1	4
2	Z	(0)	1	0	0	$\frac{45}{4}$	$\frac{225}{2}$	0	1215
	x_2	(1)	0	0	1	$\frac{1}{8}$	$-\frac{3}{4}$	0	$\frac{3}{2}$
	x_1	(2)	0	1	0	0	1	0	6
	x_5	(3)	0	0	0	$-\frac{1}{8}$	$\frac{3}{4}$	1	$\frac{5}{2}$

Tabla 3.6: Iteración 2 - Simplex

La columna de las variables básicas, muestra las variables que aparecen para formar la respuesta, como aparecen x_1 y x_2, ninguna tiene el valor de cero por defecto, sino que toman el valor que les corresponde en la columna de lado derecho. En este caso toman el valor de 6 y $\frac{3}{2}$ respectivamente. La columna de lado derecho también muestra el valor de la respuesta, en la fila de Z. Así:

$$Z = 180(6) + 90\left(\frac{3}{2}\right) = 1215 \tag{3.65}$$

Como se indico al inicio del ejercicio, es el mismo que se muestra en el método gráfico para maximización, por tanto, la respuesta coincide con la Ecuación 3.1.

3.2 Características del framework Flutter

3.2.1 Árboles de widgets, elementos y renderizado

Flutter usa tres árboles para renderizar la interfaz de usuario. Cada uno de estos árboles tiene una responsabilidad diferente pero están conectados entre sí.

El árbol con el que se tiene contacto de forma más frecuente es el árbol de widgets. Esto porque los widgets son los "bloques" para construcción de interfaz gráfica más comunes, y muchos de estos ya están pre-fabricados y vienen incluidos en el framework. [57]. Este árbol contiene los datos de configuración de los widgets, y es

usado para construir el árbol de elementos.

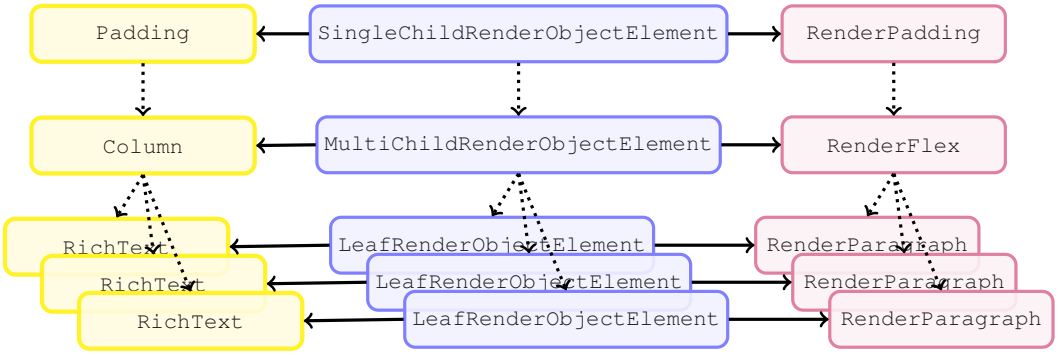


Figura 3.5: Representación de árboles que construyen la interfaz gráfica.

El siguiente, es el árbol de elementos, que es una representación interna de los widgets. Este árbol es usado para realizar la gestión del ciclo de vida de los widgets [58]. Cada elemento de este árbol corresponde a un nodo del árbol de widgets, y contiene una referencia de retorno hacia el widget. [59]

Por último el árbol de renderizado es un subconjunto del árbol de elementos, y está encargado de maquetar y dibujar elementos en pantalla [58]. Este árbol está construido por objetos a partir de clases que heredan de `RenderObject`. Por lo tanto, este árbol sirve de interfaz entre el código de alto nivel y la librería de bajo nivel `dart:UI` [60].

¿Por qué estos árboles se mantienen separados?

- **Rendimiento:** Cuando el maquetado cambia, solo se necesita actualizar el árbol de renderizado, por la composición, el árbol de elementos puede tener muchos mas items que se deberían saltar.
- **Claridad:** Cuando la separación de responsabilidades es clara, los protocolos de widgets y renderizado pueden especializar, y simplificando APIs superficiales. Esto reduce el riesgo de errores y la carga de pruebas.
- **Seguridad de Tipos:** El árbol de renderizado puede garantizar que sus descendientes serán del tipo apropiado. De esta forma los widgets de composición pueden ser al sistema de coordenadas usado en el renderizado [61].

3.2.2 Widgets

La idea de central detrás de los "widgets" en Flutter, es que la interfaz gráfica se construye con estos. Los Widgets describen como luce una vista, dada su configuración y estado actual. Cuando el estado de un widget cambia, se reconstruye su descripción, el cual el framework compara con su descripción anterior para determinar la mínima cantidad de cambios necesarios en el árbol de renderizado para realizar la transición entre estados [62].

Los widgets son una descripción inmutable de parte de la interfaz de usuario, y son además la clase central en la jerarquía de Flutter. Los Widgets pueden inflarse en *elementos*, los cuales controlan el árbol de renderizado [63].

Además los Widgets mismos no poseen un estado que pueda mutar, sino que al usar `StatefulWidget` este crea un objeto de estado en cualquier momento que este se infla como un elemento y se inserta en el árbol [63].

Flutter prefiere composición sobre herencia de clases, de esta manera desarrolladores pueden crear combinaciones de widgets, usando widgets más simples o pequeños para crear unos más complejos o con algún comportamiento específico [60].

3.2.3 Ciclo de vida de los widgets sin estado (`StatelessWidget`)

Un widget sin estado se construye en base a su propia configuración pero su estado no cambia de forma dinámica. El método `build`, encargado de la parte de la interfaz gráfica, puede ser llamado cuando [64]:

- La primera vez que el widget es creado.
- Cuando el widget ancestro cambia.
- Cuando un (widget heredado) `InheritedWidget` cambia.

3.2.4 Ciclo de vida de los widgets con estado (`StatefulWidget`)

Un widget con estado se construye en base a su propia configuración y su estado puede cambiar de forma dinámica. Este tipo de widget se declara con dos clases, la clase `StatefulWidget` y la clase `State` la cual puede mutar, y persistir aunque el widget se tenga que reconstruir; lo que mejora el rendimiento [64].

Métodos del ciclo de vida de un widget con estado:

- `initState()`
  Llamado cuando este objeto se inserta en el árbol.
- `dispose()`
  Se llama cuando el objeto se remueve del árbol permanentemente.
- `didChangeDependencies()`
  Llamado cuando el objeto de estado de este widget cambia.
- `didUpdateWidget(Widget oldWidget)`
  Se llama cuando la configuración del widget cambia. `oldWiget` puede ser usado para verificar si, efectivamente la configuración es diferente a la anterior.
- `deactivate()`
  Llamado cuando el widget es quitado del árbol, pero no es destruido.
- `build(BuildContext context)`
  Puede ser llamado tantas veces como sea necesario para construir la interfaz de usuario. El objeto `BuildContext` controla la ubicación del widget en al árbol.
- `setState()`
  Notifica al framework que el estado de este objeto ha cambiado, y se programa una llamada al método `build`.

3.2.5 Recarga en caliente

Recarga en caliente (Hot Reload) es una característica de Flutter que permite experimentar con cambios en la interfaz de usuario, corrección de errores y cambios

de características, de manera fácil y muy rápida.

Esta característica inyecta el código fuente en la Máquina Virtual de Dart (VM) en ejecución. Una vez que la VM actualiza las clases con sus nuevas versiones, Flutter reconstruye de forma automática el árbol de widgets [65]. Esto permite ver el efecto de los cambios casi de forma instantánea.

Cuando se invoca recarga en caliente se re-compilan las siguientes librerías [65]:

- Cualquiera de las librerías con código modificado.
- La librería principal de la aplicación.
- Las librerías de la librería principal que llevan a las librerías afectadas.

3.2.6 Herramienta de desarrollo multiplataforma

Flutter provee varios instrumentos para desarrolladores, que permiten la creación de aplicaciones multiplataforma, que comparten el mismo código fuente [66].

Características multiplataforma de Dart y Flutter	
Sistemas Operativos	Android iOS Linux macOS Web (Chrome, Edge, Firefox, Safari) Windows
Arquitecturas	armv7, armv8 x86, x64
Compilación	**Justo a tiempo (JIT):** Produce código máquina para cambios rápidos en desarrollo **Antes de tiempo (AOT):** Produce código máquina para producción con máximo rendimiento.

Tabla 3.7: Características multiplataforma [67], [68].

a. Soporte de Entorno de desarrollo integrado (IDE)

El código fuente de una aplicación de Flutter se puede escribir usando cualquier editor de texto. Flutter además soporta completamente los editores Android Studio, IntelliJ y Visual Studio Code [66]. Para estos editores Flutter ofrece extensiones que agregan funcionalidad para:

- Crear nuevas aplicaciones usando un asistente.
- Depurar código fuente con soporte usando puntes de quiebre, y fácil cambio de dispositivos.
- Agregar fragmentos de código, evita escribir código repetitivo.
- Refactorizar código, mejora la organización y acelera la creación e inserción de widgets.
- Mostrar errores de sintaxis y advertencias.

b. Inspeccion del diseño de la interfaz y el estado de la aplicación

El inspector es un instrumento que permite visualizar y explorar el árbol de widgets de Flutter, esto se muestra en la Figura 3.6. Este inspector se puede usar para entender el diseño actual y diagnosticar problemas de diseño [66]. Este tiene una serie de características:

- **Modo selección de widget:** Permite seleccionar un widget desde el dispositivo donde se esté ejecutando la aplicación, y ver sus propiedades en el inspector.
- **Actualizar árbol:** Recarga la información del widget actual en el inspector.
- **Animaciones lentas:** Las animaciones se ejecutan 5 veces más lentamente para ayudar a ajustarlas correctamente.
- **Mostrar lineas directrices:** Lineas directrices superpuestas que asisten en la corrección de problemas de alineación del diseño.

- **Mostrar lineas de base:** Se usan para alinear texto.
- **Resaltar repintados:** Muestra bordes que cambian de color cuando un elemento se repinta.
- **Resaltar imágenes de gran tamaño:** Se voltea e invierte el color de imágenes que utilizan demasiada memoria.

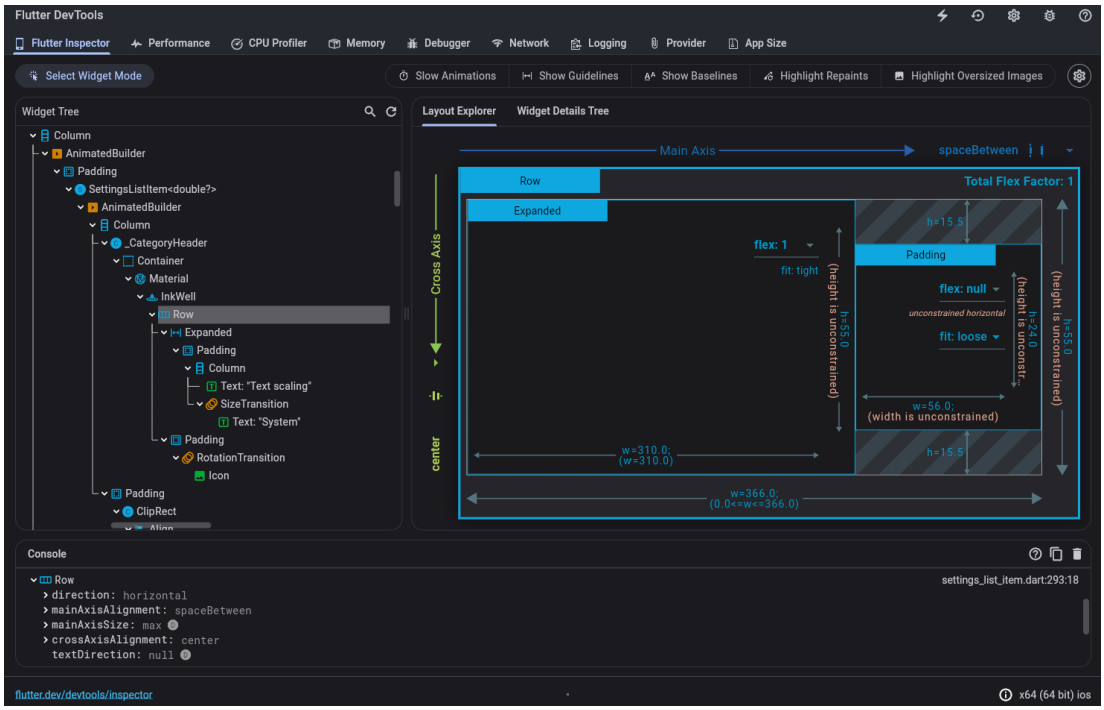


Figura 3.6: Flutter inspector [66].

c. *Diagnóstico de rendimiento de la aplicación*

Esta vista puede ayudar a diagnosticar problemas de rendimiento y bloqueos de procesamiento en el hilo de la interfaz gráfica de la aplicación. Esta vista puede ayudar a identificar las causas del bajo rendimiento de la aplicación [66]. Esta vista tiene las siguientes características:

- **Gráfico de fotogramas:** Contiene la información de los fotogramas de la aplicación, cada barra representa un fotograma, y su color muestra el trabajo realizado para el renderizado el fotograma.

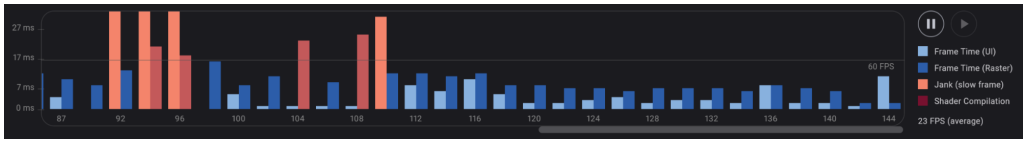


Figura 3.7: Gráfico de fotogramas [66].

- **Análisis de fotograma:** Cuando se selecciona uno de los fotogramas que se muestran en el gráfico de fotogramas 3.7, se muestra el análisis de la información sobre el tiempo que tomó cada una de las fases de interfaz gráfica y renderizado.

- **Estadísticas de rasterizado:** Puede ayudar a diagnosticar, y encontrar el origen de bloqueos de procesamiento en el hilo de rasterizado.

- **Eventos de la línea de tiempo:** Muestra el seguimiento de todos los eventos de la aplicación.Flutter emite eventos de linea del tiempo mientras construye fotogramas, dibuja escenas, y rastrea otras actividades, como peticiones de red. Los desarrolladores tienen la habilidad de emitir sus propios eventos usando la Interfaz de programación de aplicaciones (API) `Timeline` y `TimelineTask` del paquete `dart:developer` de Flutter.

- **Opciones avanzadas de depuración:** Se permite mejorar el seguimiento de los eventos en el tiempo, con respecto a la construcción de widgets 3.8, diseño y renderizado. *Estas opciones pueden ralentizar la velocidad de fotogramas.*

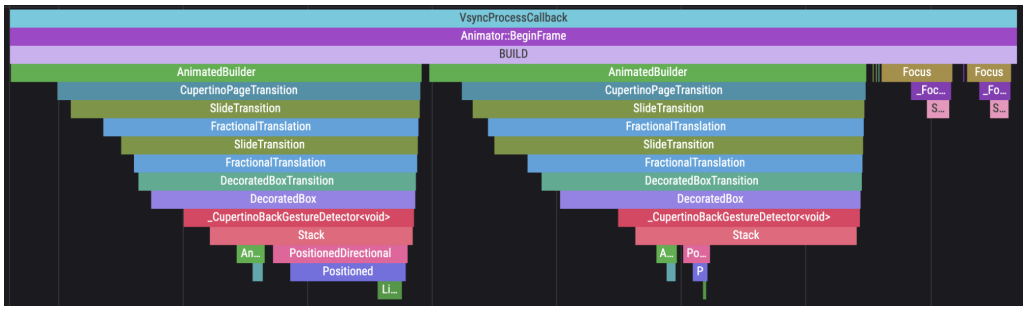


Figura 3.8: Seguimiento de la construcción de Widgets [66].

d. Perfilado del uso de la Unidad de Procesamiento Central (CPU)

El perfilador de CPU permite grabar y perfilar una sesión de una aplicación Dart o Flutter. Puede ayudar a entender la actividad de la aplicación en CPU y solucionar problemas de rendimiento. La VM recopila muestras de CPU y permite la

visualización de los datos. La agregación de estas muestras posibilita la comprensión de cómo se utiliza el tiempo del CPU mayoritariamente [66]. El perfilador puede mostrar esta información de las siguientes maneras:

- **Fondo arriba:** Provee una representación de abajo hacia arriba del perfil del CPU. En esta vista, un método se puede expandir para revelar que métodos lo llaman. Esta vista es útil para encontrar los métodos que consumen la mayor parte del tiempo de la CPU.
- **Árbol de llamadas:** Provee una representación de arriba hacia abajo del perfil del CPU. En esta vista, un método se puede expandir para revelar a que métodos llama. Esta vista es útil para encontrar las rutas que consumen la mayor parte del tiempo de la CPU.
- **Tabla de métodos:** Provee estadísticas para cada uno de los métodos en el perfil del CPU.
- **Gráfico de llama:** Provee una representación gráfica del árbol de llamadas. De arriba hacia abajo del perfil del CPU. En esta vista, el ancho de cada flama representa el tiempo que un método permanece en la pila de llamadas.

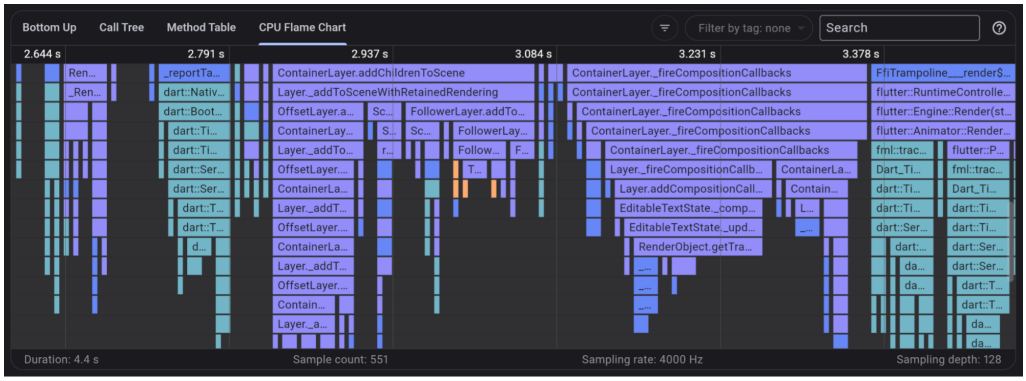


Figura 3.9: Gráfico de llama [66].

e. Vista de uso de la Memoria de Acceso Aleatorio (RAM)

El tiempo de ejecución de Dart incluye un recolector de basura, este componente se encarga de asignar memoria al crear instancias de objetos, y des-asignar memoria cuando estas instancias se vuelven inalcanzables [69].

Por esto, la vista de uso de memoria se usa para una optimización de memoria preventiva o cuando una aplicación experiencia ralentizaciones, errores por falta de memoria, o si se sospecha de una fuga de memoria u otros problemas similares [66].

Un gráfico de series temporales visualiza el estado de la memoria de Flutter en intervalos de tiempo sucesivos. Cada punto en el gráfico corresponde a la marca de tiempo (eje x) de las cantidades medidas (eje y) de la memoria *heap*. Se capturan el uso, la capacidad, la recolección externa de elementos no utilizados, etc ...

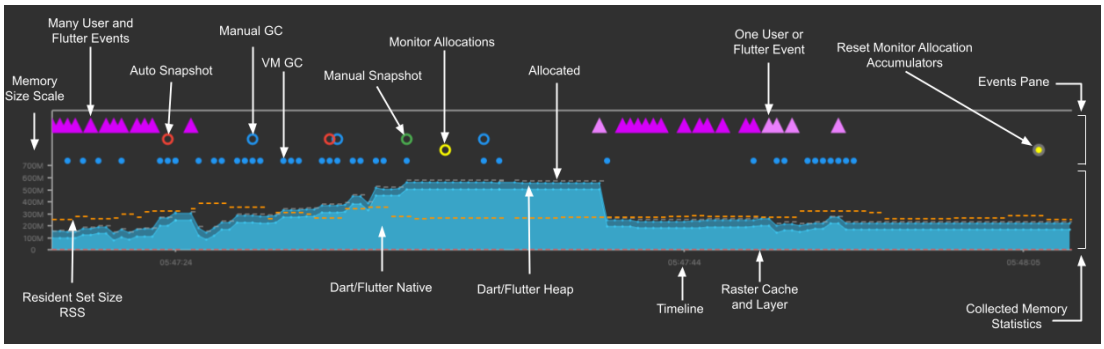


Figura 3.10: Gráfico de memoria [66].

f. Perfilado de Red

Habilita la posibilidad de examinar el tráfico Hypertext Transfer Protocol (HTTP), HTTPS y de sockets web que fluye desde una aplicación Flutter. El tráfico de red se registra de forma automática cuando abre la vista Red, además de poder hacerse de forma manual. Una vez registrados los datos, se puede ver a detalle información general y de tiempos, contenido del encabezado y cuerpo de una solicitud o respuesta de la red [66].

3.2.7 Versión del Framework Flutter seleccionada para desarrollar la aplicación

La versión de Flutter que se utilizará para el desarrollo del presente proyecto es la versión 3.10, esta incluye diversas mejoras de nuevas características, ideales para el desarrollo de aplicaciones multiplataforma modernas de alto rendimiento. Además, se utilizará la versión 3 del lenguaje de programación Dart, el cual se integra de manera

nativa con Flutter y ofrece funcionalidades nuevas, para acelerar tanto el desarrollo y la ejecución de aplicaciones.

Flutter 3.10 incluye muchas mejoras con respecto a web, renderizado y seguridad [70].

- **Mejoras en el soporte de escritorio:** Incluye mejoras para soporte de escritorio Windows y macOS.
- **Mejora de rendimiento:** Se incrementa rendimiento, como la reducción del tiempo de inicio, la reducción del tamaño del código compilado.
- **Mejora de la depuración:** Se moderniza la experiencia de depuración, y mejora la accesibilidad. Reemplaza el visor de seguimiento de la línea de tiempo con *perfetto*, permitiendo manejar conjuntos de datos más grandes.
- **Mejora del soporte web:** Las optimizaciones del soporte web, incluye:
  - Mejora del tiempo de carga de una aplicación web.
  - El tamaño de `CanvasKit` se reduce, a ahora se sirve desde los servidores CDN de Google.
  - Las aplicaciones Flutter web se pueden incrustar en cualquier elemento web, y compartir estado con el sitio web.
  - Aplicaciones Flutter para web, ahora tienen compatibilidad con *shaders*
- **Mejora de seguridad:** Flutter Framework ahora se compila con Niveles de Cadena de Suministro para Artefactos de Software (SLSA) de nivel 1.

3.3 Desarrollo de la aplicación multiplataforma

3.3.1 Análisis y selección de metodología de desarrollo de software.

Kanban, Scrum y XP son metodologías ágiles que pueden ser útiles para desarrollo de proyectos de software. Kanban permite entregas en cualquier momento y cambios de prioridades al vuelo, con un enfoque en el flujo continuo de trabajo. Scrum es adaptable, fomenta la autogestión y permite centrarse en los objetivos del sprint.

XP se enfoca en la calidad del código y las prácticas de ingeniería, favoreciendo la comunicación y el sentido de pertenencia en un equipo.

	Scrum	**XP**	**Kanban**
Principios clave	Comunicación, retroalimentación y simplicidad	Transparencia, inspección y adaptación	Visualización, flujo y límites de trabajo
Enfoque	Marco de trabajo para gestionar proyectos complejos	Mejora continua de calidad y desarrollo rápido	Gestión visual del flujo de trabajo
Roles	Product Owner, Scrum Master, Equipo de Desarrollo	Programador, Cliente, Entrenador	No establece roles específicos
Iteraciones	Sprints de tiempo fijo, generalmente de 1 a 4 semanas	Iteraciones cortas y frecuentes	Flujo continuo sin iteraciones fijas
Artefactos	Product Backlog, Sprint Backlog, Incremento	User Stories, Pair Programming, Tests	Tablero Kanban, Tarjetas de trabajo
Ventajas	Flexibilidad, adaptación al cambio, enfoque colaborativo	Alta calidad, rápida respuesta a cambios, enfoque en la simplicidad	Visualización del trabajo, flujo constante, mejora de procesos
Desafíos	Requiere disciplina y compromiso, dependencia de la comunicación efectiva	Dedicación de recursos para prácticas como Pair Programming, puede ser resistido por equipos	Requiere un flujo de trabajo claro y bien definido, puede ser difícil de implementar en equipos grandes

Tabla 3.8: Tabla comparativa entre XP, Scrum y Kanban.

Se consideraron tres metodologías ampliamente utilizadas en la industria del software: XP, Scrum y Kanban. Las cuales se describen a detalle en la sección Metodologías de desarrollo en el capítulo uno.

Tras una evaluación de las características y principios de cada metodología, se determinó que Kanban se ajustaba de manera óptima a las necesidades específicas de este proyecto. Dado que el equipo de desarrollo consiste en una persona, Kanban proporciona una estructura flexible y adaptable que permite la gestión eficiente de las tareas y prioridades individuales.

Además, las características de flujo continuo, transparencia y visualización de Kanban

resultan especialmente beneficiosas para un desarrollador, porque brinda una visión clara del progreso del proyecto y facilita la gestión de las tareas de manera óptima.

3.3.2 Análisis y selección de tecnología para backend.

Firebase, AppWrite y MongoDB son opciones populares para el desarrollo de aplicaciones con Flutter. Firebase es una plataforma de desarrollo de aplicaciones de Google que se integra con flutter, con su SDK para desarrolladores. Appwrite es un servidor de backend de código abierto diseñado para simplificar el desarrollo del backend y proporciona herramientas útiles para autenticación, almacenamiento de datos y manejo de archivos. MongoDB es una base de datos orientada a documentos, de código abierto y multiplataforma, esta utiliza documentos similares a JSON con esquemas opcionales.

	Firebase	**AppWrite**	**MongoDB**
Modelo de datos	Utiliza un modelo de datos basado en documentos	Implementa un modelo de datos basado en colecciones	Es una base de datos NoSQL con un modelo de documentos flexible
Autenticación	Proporciona servicios de autenticación integrados con soporte para múltiples métodos de inicio de sesión	Incluye funcionalidades de autenticación con soporte para métodos como correo electrónico, OAuth y más	Ofrece opciones de autenticación integradas y personalizables
Gestión de archivos	Permite almacenar y servir archivos estáticos como imágenes, videos y más	Incluye un módulo de almacenamiento de archivos para cargar y servir archivos de manera segura	No incluye funcionalidades nativas de almacenamiento de archivos

Servicios en la nube	Ofrece una amplia gama de servicios en la nube, incluyendo análisis, mensajería, hospedaje, entre otros	Se centra en proporcionar servicios de backend, incluyendo autenticación, almacenamiento y funciones en la nube	Ofrece servicios en la nube para hospedar bases de datos de manera escalable
Escalabilidad	Es altamente escalable y puede manejar aplicaciones con millones de usuarios simultáneos	Está diseñado para ser escalable y puede adaptarse a las necesidades de crecimiento de la aplicación	Es altamente escalable y puede distribuirse en clústeres para manejar grandes volúmenes de datos y tráfico
Compatibilidad de lenguajes	Cuenta con SDKs disponibles para varios lenguajes de programación como Dart, JavaScript, Swift, Java, entre otros	Ofrece SDKs para varios lenguajes incluyendo Dart, JavaScript, Python, PHP, y más	Tiene controladores disponibles para una amplia variedad de lenguajes de programación como JavaScript, Python, Java, entre otros
Licencia	No aplicable (Propietaria)	Apache License 2.0 (Código abierto)	Server Side Public License (Código abierto)

Tabla 3.9: Tabla comparativa entre Firebase, AppWrite, MongoDB.

Se examinaron diversas opciones de tecnologías backend, incluyendo Firebase, Appwrite y MongoDB. Tras una comparación, se determinó que Appwrite es idóneo para la gestión de sesiones de usuario y la base de datos. La flexibilidad y funcionalidades de autenticación y autorización de Appwrite satisfacían los requisitos

específicos del sistema, Además, de que este es un proyecto de código abierto y permite el despliegue por el mismo equipo de desarrollo.

Por otro lado, Firebase se seleccionó para el alojamiento de la versión web de la aplicación, aprovechando su sólida infraestructura y facilidad de despliegue. Se añade a esto que Google le ha optimizado para despliegues de aplicaciones Flutter.

Al combinar ambas soluciones, se logró un backend completo y eficiente para el sistema multiplataforma, que garantiza un manejo seguro de datos y una experiencia de usuario fluida. Esta selección estratégica respalda el desarrollo del sistema; además optimiza su rendimiento y funcionalidad en diferentes plataformas.

3.3.3 Aplicación metodológica.

Para la aplicación de la metodología de desarrollo de software, se configuró Jira como una herramienta para dar seguimiento al proyecto mediante la metodología Kanban, facilitando el seguimieto de las tareas.

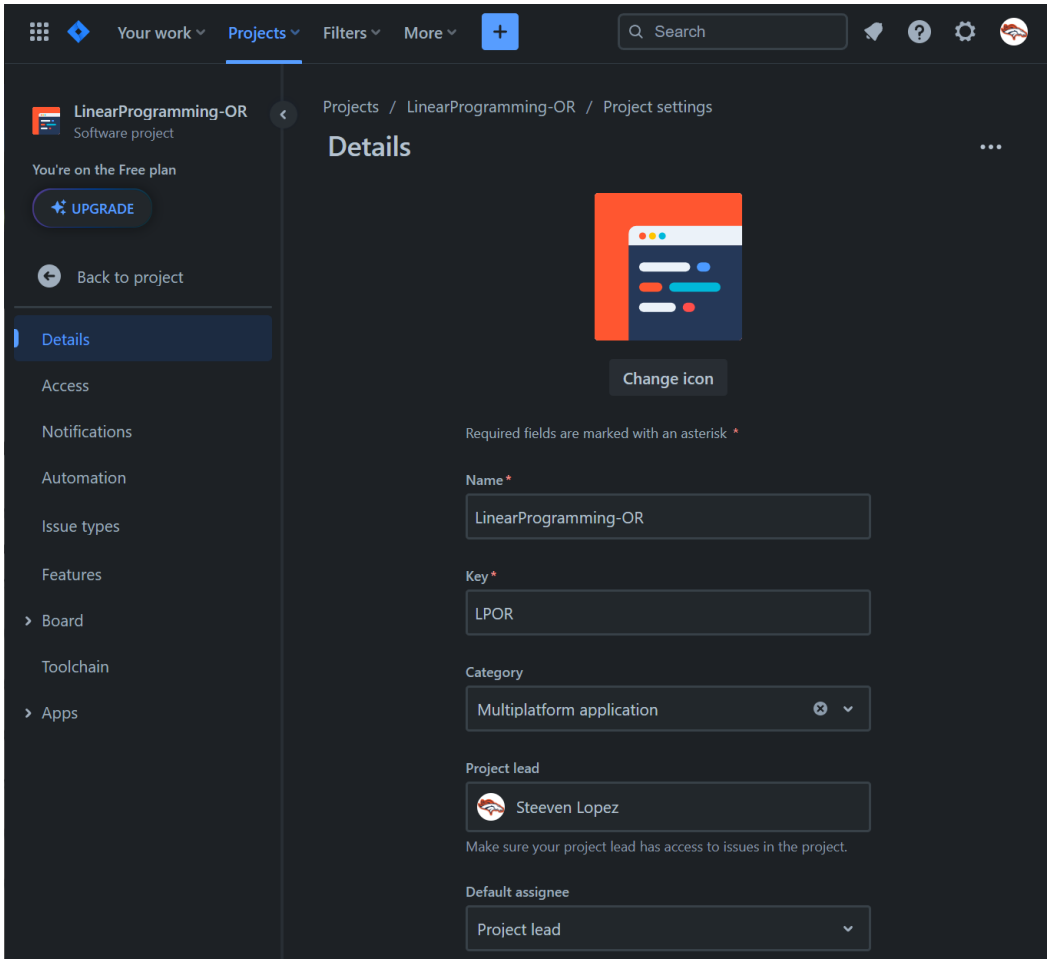


Figura 3.11: Detalles del proyecto en Jira.

Jira permite aplicar la metodología Kanban facilitando la creación de tableros personalizables y tareas que se pueden organizar en columnas, las cuales reflejan el estado de la tarea. Para este proyecto, se optó por usar las columnas que simbolizan los estados de las tareas: *Por hacer*, *En progreso* y *Hecho*.

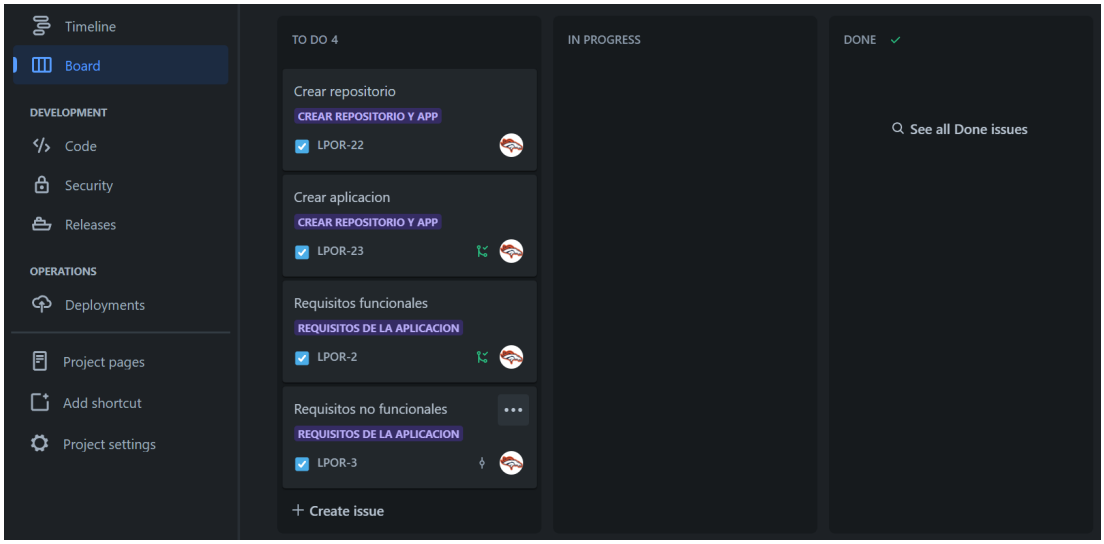


Figura 3.12: Vista inicial de las tareas en el tablero Kanban.

Jira, además, permite visualizar las tareas en el tablero Kanban, en la línea de tiempo, facilitando aún más la planificación y el seguimiento del proyecto.

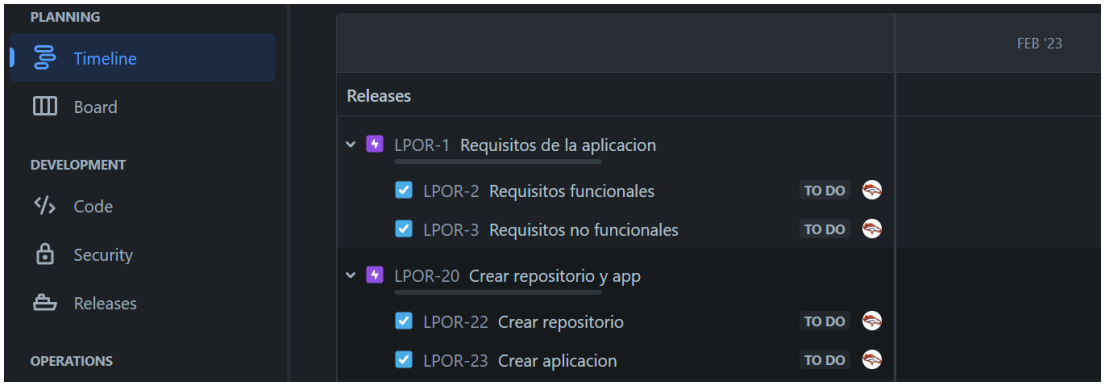


Figura 3.13: Vista de las tareas en la línea de tiempo.

Cada una de las tareas en el tablero Kanban, se pueden visualizar en detalle, revelando información específica de la tarea, como su descripción, subtareas, responsables e incluso los avances de desarrollo atados a esta tarea, usando su identificador en el sistema de control de versiones.

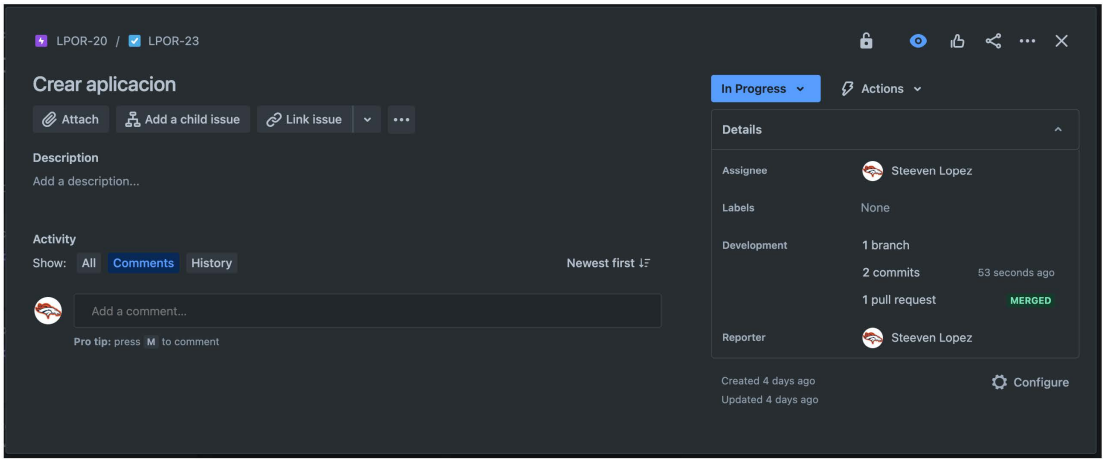


Figura 3.14: Vista de detalle de la tarea de crear aplicación.

3.3.4 Especificación de requisitos funcionales de la aplicación multiplataforma.

- **Acceso a la aplicación sin registro:** La aplicación permitirá a los usuarios resolver problemas de programación lineal sin necesidad de inicio de sesión o registro.
- **Registro de usuarios:** La aplicación permitirá a los usuarios registrarse para acceder a funcionalidades adicionales, como guardar configuraciones personalizadas. El proceso de registro que solicitara información básica, como dirección de correo electrónico y clave.
- **Ingreso de datos del problema:** La aplicación debe permitir a los usuarios ingresar los datos del problema de programación lineal, como coeficientes de variables, restricciones y función objetivo.
- **Validación de datos:** La aplicación debe validar los datos ingresados por los usuarios para asegurarse de que cumplan con los requisitos de formato y consistencia necesarios para la resolución del problema.
- **Solución de problemas de programación lineal:** La aplicación debe permitir la resolución de problemas de programación lineal, permitiendo encontrar soluciones óptimas.
- **Visualización de resultados:** La aplicación debe ser capaz de mostrar los resultados de manera clara y comprensible, ya sea a través de tablas, gráficos

u otras representaciones visuales.

- **Importar datos de problemas:** La aplicación debe permitir a los usuarios, importar los datos de problemas de programación lineal.
- **Exportación de resultados:** La aplicación debe permitir a los usuarios exportar los datos de los problemas, en un formato que puede volver a utilizar en la aplicación.
- **Compatibilidad multiplataforma:** La aplicación debe ser compatible con diferentes plataformas, como sistemas operativos de escritorio y dispositivos móviles, para garantizar su accesibilidad y usabilidad en distintos entornos.

3.3.5 Especificación de requisitos no funcionales de la aplicación multiplataforma.

- **Usabilidad:** La interfaz se diseña para ser intuitiva, fácil de usar para el usuario.
- **Rendimiento:** La aplicación debe ser eficiente en términos de uso de recursos y tiempo de respuesta.
- **Mantenibilidad:** La aplicación debe ser fácil de mantener y actualizar. Siguiendo buenas prácticas de desarrollo de software para facilitar la corrección de errores, la introducción de nuevas funcionalidades y mejoras.
- **Compatibilidad:** La aplicación es compatible con las versiones más recientes de los navegadores web y sistemas operativos móviles y de escritorio.
- **Disponibilidad:** La aplicación está disponible para uso sin conexión a internet, incluso por usuarios no registrados.

3.3.6 Configuración de ambiente de desarrollo.

La configuración del ambiente de desarrollo es una etapa fundamental para asegurar el óptimo desarrollo del sistema multiplataforma. Primero, se realizó la instalación y configuración del entorno de Flutter, garantizando la instalación adecuada de

las dependencias y herramientas necesarias para el desarrollo de la aplicación. A continuación, se configuró Appwrite 3.15 y Firebase Hosting 3.16, estableciendo la conexión y la configuración adecuada para su integración con el sistema.

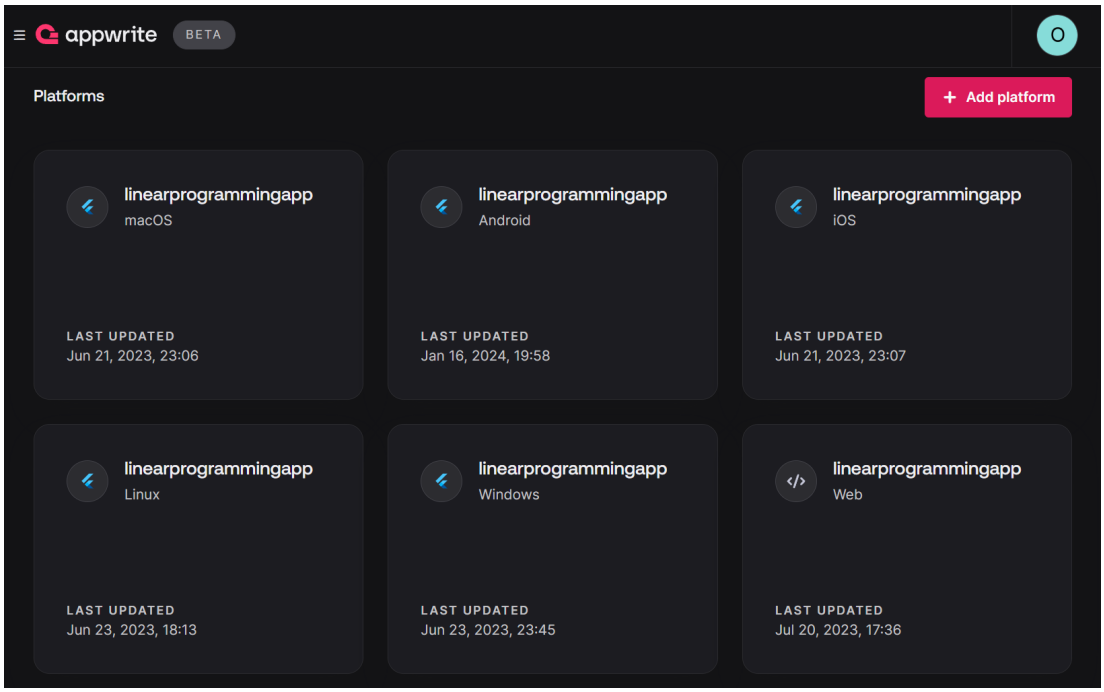


Figura 3.15: AppWrite - vista general de las plataformas configuradas.

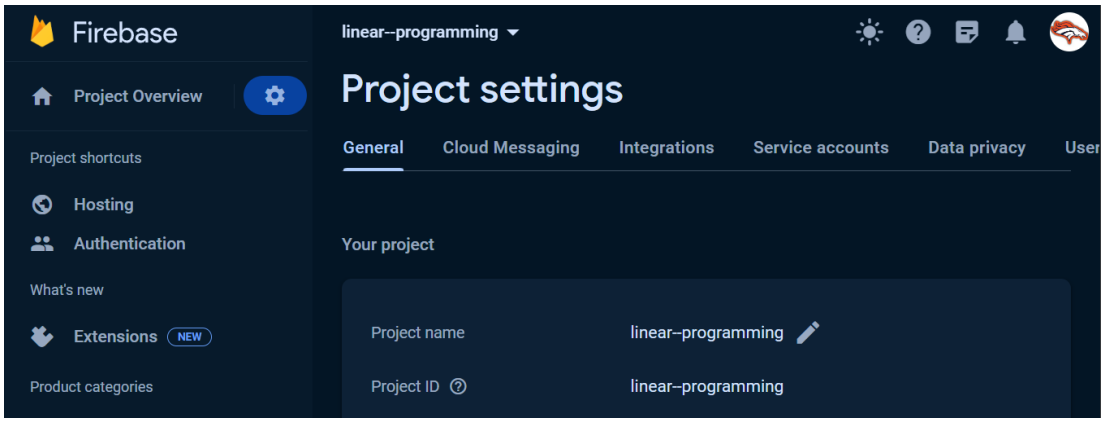


Figura 3.16: Firebase - Vista general del proyecto.

Además, se hace uso del sistema de control de versiones Git a través de la plataforma GitHub, permitiendo un seguimiento preciso de los cambios durante el desarrollo del proyecto 3.17. Esta configuración integral del ambiente de desarrollo garantiza una base sólida y eficiente para el desarrollo del sistema multiplataforma, optimizando la productividad y el flujo de trabajo.

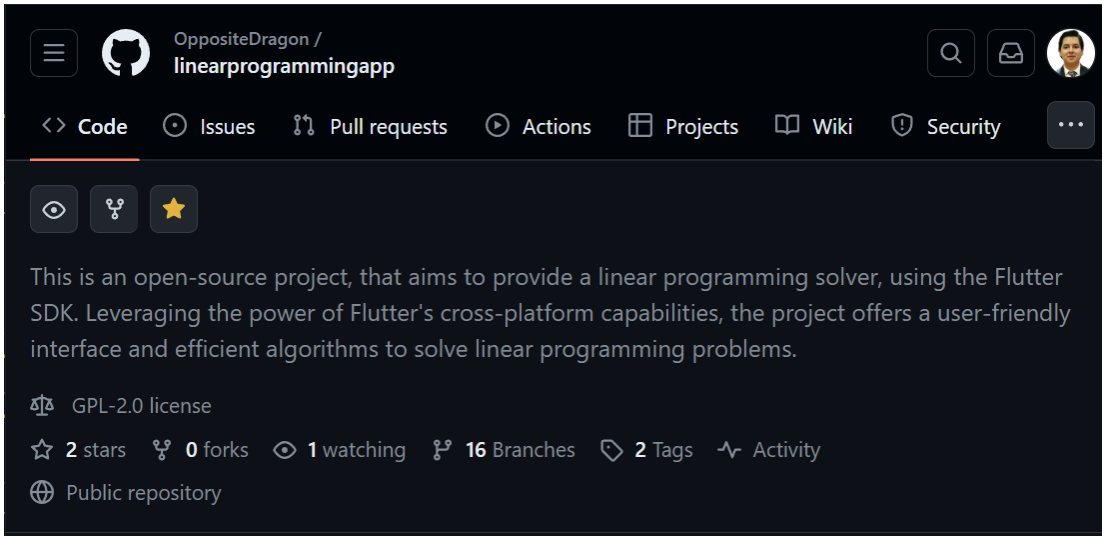


Figura 3.17: Repositorio en GitHub.

3.3.7 Desarrollo de módulos del sistema.

La primera etapa consistió en la creación del repositorio Git y la configuración de las ramas base para el control de versiones. La figura 3.18 muestra las ramas de creación de la aplicación y definición de requerimientos.

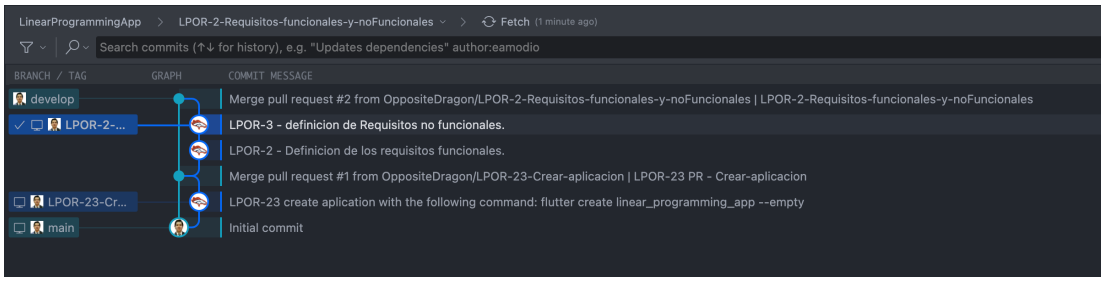


Figura 3.18: Visualización de ramas de creación de app y definición de requerimientos.

Estas acciones fueron expresadas como tareas en el tablero Kanban. La figura 3.19 muestra la vista inicial del tablero Kanban, donde se pueden observar las tareas de creación de la aplicación y definición de requerimientos aún en estado inicial.

Los detalles de cada tarea pueden mostrar el proceso para el desarrollo de la tarea, al ir vinculando ramas, commits y pull requests, usando los códigos que Jira provee para cada tarea. Los vínculos en los detalles de tarea quedan expuestos en la figura 3.14.

A continuación, se procedió a definir los requisitos del sistema, estableciendo una lista clara y detallada de las funcionalidades y características deseadas. Estos requisitos

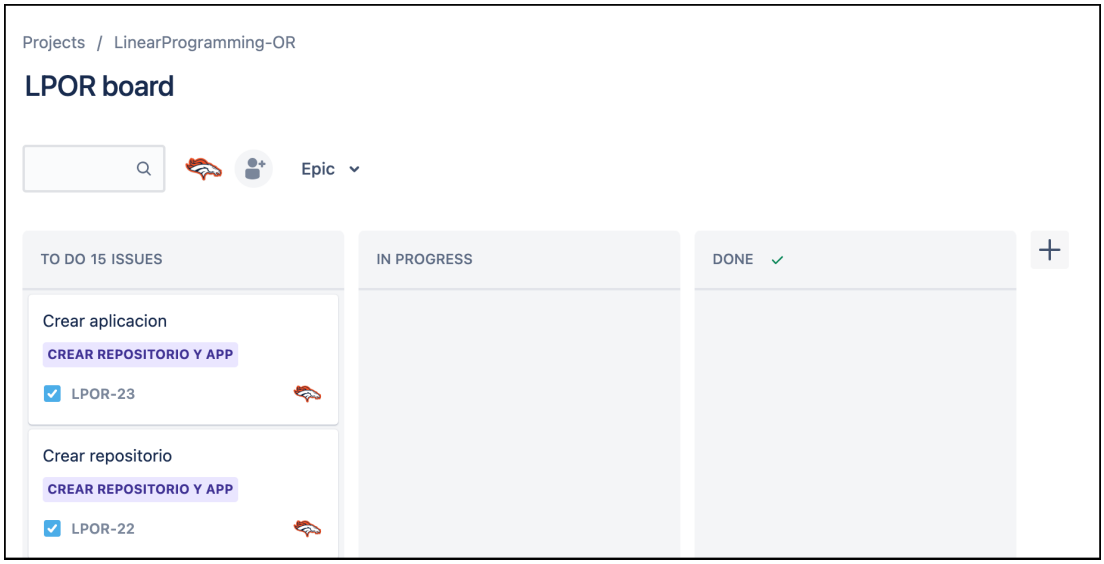


Figura 3.19: Vista inicial del tablero Kanban.

fueron organizados en tarjetas Kanban, asignadas a las respectivas etapas de desarrollo. La metodología Kanban permitió una visualización clara y el seguimiento de cada una de las tareas, facilitando una planificación y asignación eficiente de los recursos.

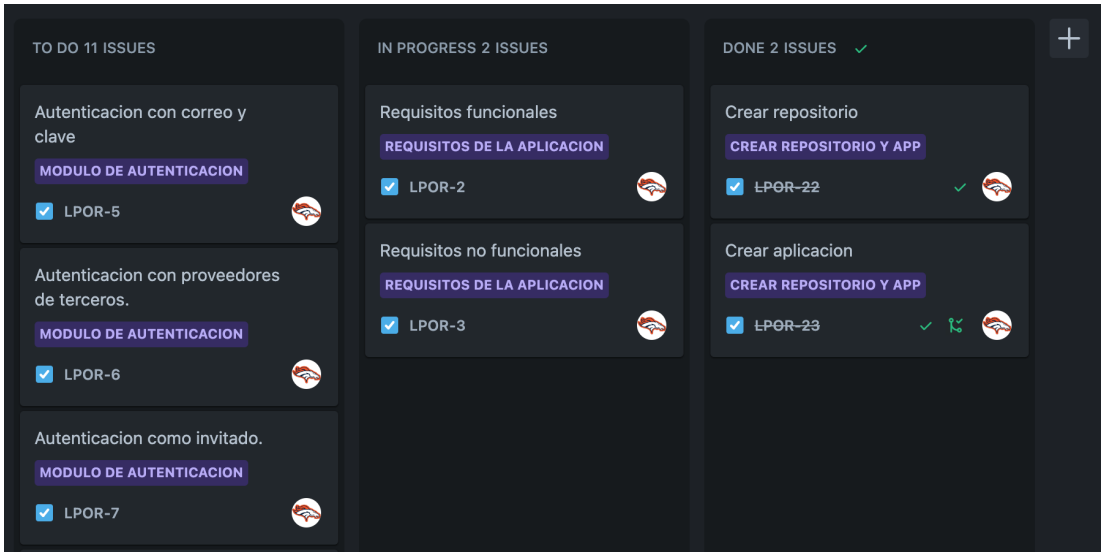


Figura 3.20: Tareas de especificación de requisitos en progreso.

a. Módulo de autenticación

El primer módulo abordado fue el de **autenticación**, el cual se desglosó en submódulos para abarcar distintos métodos de autenticación. Ver figura 3.21.

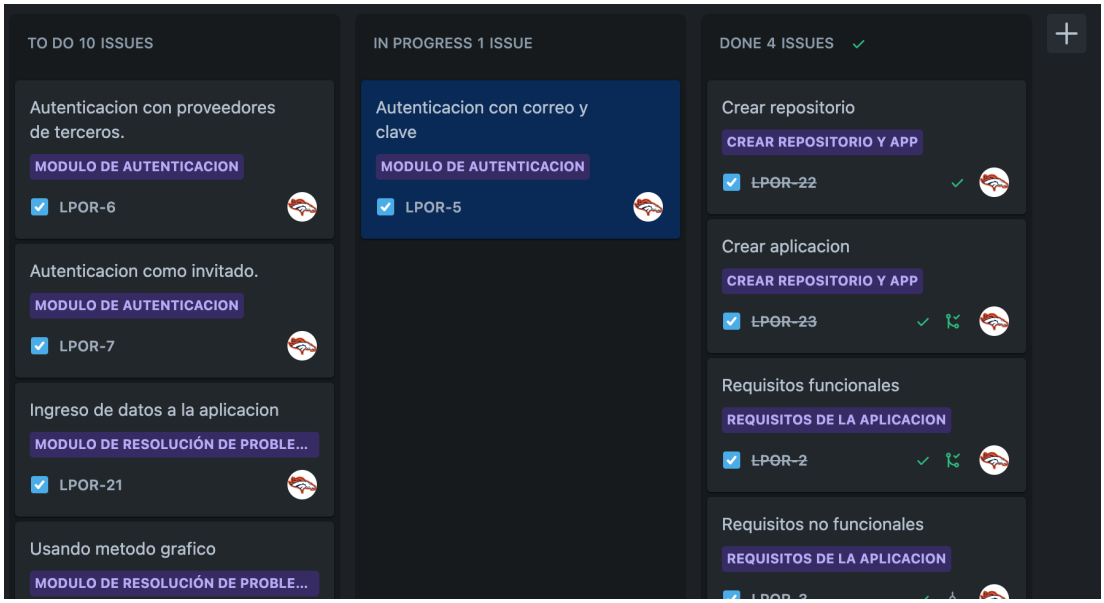


Figura 3.21: Tarea de autenticación con correo y clave en progreso.

En primer lugar, se implementó el submódulo de *autenticación de correo y clave*, permitiendo a los usuarios registrarse y acceder al sistema mediante credenciales convencionales.

```dart
//Método para crear una cuenta con correo y clave
Future<void> createEmailAndPasswordAccount(
String name,
String email,
String password,
) async {
final account = ref.read(accountProvider);
final user = await account.create(
userId: ID.unique(),
name: name,
email: email,
password: password,
);
}

//Método para iniciar sesión con correo y clave
Future<void> logInEmailAndPassword(
String email,
String password,
) async {
final account = ref.read(accountProvider);
final session = await account.createEmailSession(
email: email,
password: password,
);
}
```

A continuación, se desarrolló el submódulo de *autenticación con proveedores de*

terceros, brindando a los usuarios la opción de iniciar sesión con sus cuentas de Google.

```dart
//Método para iniciar sesión con Google, usando OAuth2
Future<void> signInWithGoogle() async {
final account = ref.read(accountProvider);
await account.createOAuth2Session(
provider: 'google',
success: _successCallback(),
failure: _failureCallback(),
);
}
```

Por último, se implementó el submódulo de *autenticación como invitado*, para permitir un acceso temporal sin necesidad de registro. Este enfoque modular y escalonado en el desarrollo del módulo de autenticación, de acuerdo con la metodología Kanban, permitió un progreso ágil y una gestión eficiente de las tareas, asegurando la disponibilidad del sistema. En la figura 3.22 se pueden observar dos de las tareas de autenticación en estado: "en progreso" de forma simultánea.

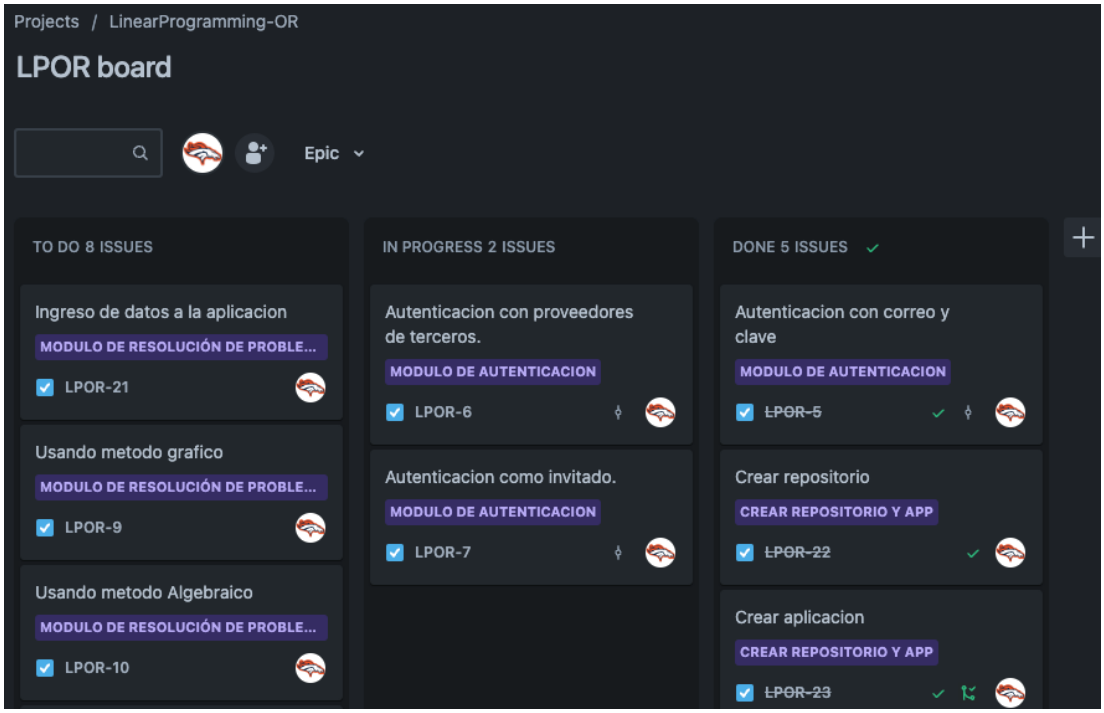


Figura 3.22: Otras tareas de autenticación en progreso.

El uso de git como control de cambios, permite ver los cambios, conforme se realizan sobre la aplicación, a medida que avanza el desarrollo del proyecto. Cada mensaje de commit tiene un código, que le vincula con una tarea o subtarea en Jira, y se sincroniza con el tablero Kanban. Esto queda demostrado en la figura 3.23.

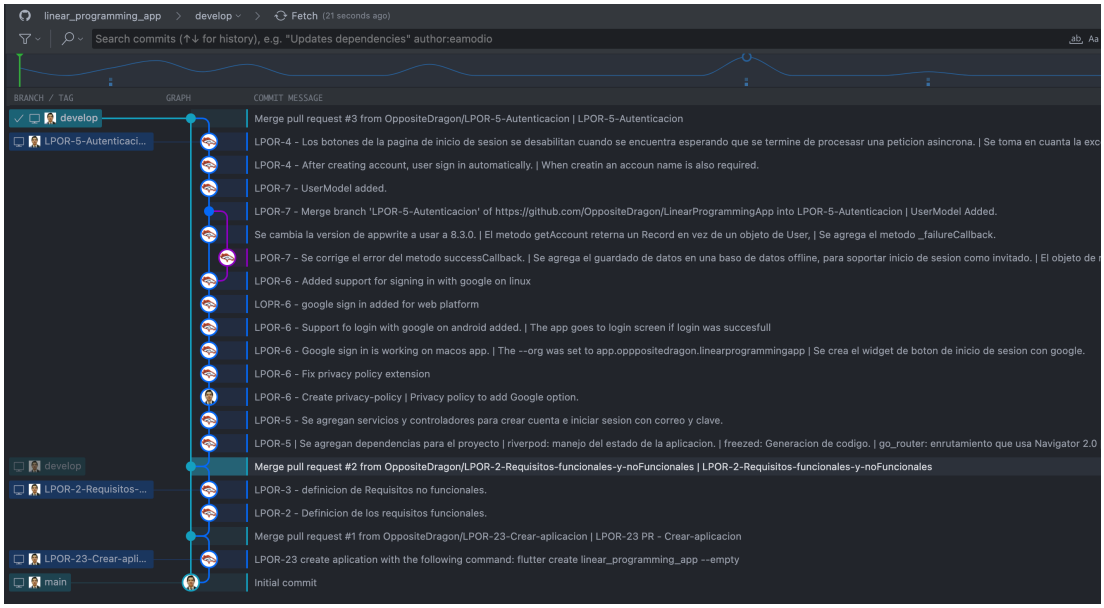


Figura 3.23: Commits del avance del desarrollo hasta las tareas de autenticación.

b. Módulo de recolección de datos

Figura 3.24: Fases 1 y 2 de la entrada de datos.

A continuación se procedió con el desarrollo del módulo de recolección de datos. Este módulo consta de dos fases. En la primera se recopilan las *dimensiones* del problema, y el método por el cual el usuario desea resolver el ejercicio. La seleccion de métodos, solo está disponible si el número de variables es 2, de otra manera, simplex

se selecciona por defecto.

```
//Establece la cantidad de variables
y verifica si mostrar la seleccion de procesos.
void setVariables(String source) {
if (int.tryParse(source) == null) return;
final value = int.parse(source);
if (state.variables == value) return;
state = state.copyWith(
variables: value,
showProcess: value == 2,
);
if (state.variables != 2) {
 ref.read(processTypeControllerProvider.notifier)
 .setProcess(ProcessTypes.simplex);
 }
}
```

Con esto, se continúa a la siguinte fase, donde el usuario puede ingresar los datos mismos del problema. Estos datos deben cumplir las dimensiones ingresadas en la primera fase. El controlador se encarga de tomar los datos de las dimensiones para crear la lista de variables, y la matriz de restricciones.

```
@riverpod
class DataEntryController extends _$DataEntryController {
@override
DataEntryModel build() {
 //Toma los datos de cantidad de variables y restricciones
 final variables = ref.watch(entrySizeControllerProvider
 .select((value) => value.variables));
 final constraints = ref.watch(entrySizeControllerProvider
 .select((value) => value.constraints));
 //Genera estructuras de datos, para las variables y restricciones
 final DataEntryModel dataEntryState = DataEntryModel(
 objectiveFunction: List.generate(variables, (i) => 0.0),
 constraints: List.generate(constraints, (i) =>
 List.generate(variables + 1, (j) => 0.0)),
 operators: List.generate(constraints, (i) => Operators.leq),
 objective: Objectives.max,
 );
 return dataEntryState;
 }

//Métodos para mutar el estado del Notifier
...
}
```

En la vista para la entrada de datos, se usa una única ruta, pero las interfaces de cada fase se muestran por separado, dentro de `PageView` que permite indicar al usuario la continuidad y relación de una fase con la otra. El usuario tiene el control para avanzar o retroceder entre fases, pero no mediante "scroll", como `PageView` funciona

normalmente, sino que mediante acciones deliberadas como presionar un botón, o retroceder. La interfaz de usuario para estas fases está ordenada, como se muestra en 3.24.

c. Módulo de método algebráico

El método algebráico usa sistemas de ecuaciones para encontrar la solución. "equations" es una librería escrita en Dart que tiene como una de sus utilidades, la solución de sistemas de ecuaciones [71].

```
//El parametro `matrix` es un sistema de ecuaciones
//de una de las combinaciones posibles.
final List<double> solutions = LUSolver(
matrix: RealMatrix.fromData(
rows: constraints.length,
columns: constraints.length,
data: matrix,
),
knownValues: righSide)
.solve();
```

El método algebráico, presenta la respuesta, en tres secciones, la conversión a forma estándar junto a la fórmula para encontrar el número de combinaciones 3.25, todas las combinaciones posibles 3.26 y la respuesta en forma de texto, donde se indica al usuario la combinación que contiene la respuesta óptima 3.27.

$$\begin{aligned} +6x_1 +8x_2 +x_3 &= 48 \\ +x_1 \quad -x_4 &= 6 \\ +x_2 \quad +x_5 &= 4 \end{aligned}$$

$$x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0$$

$$C(n,r) = \frac{5!}{2!(5-2)!} = 10$$

Figura 3.25: Forma estándar y combinaciones.

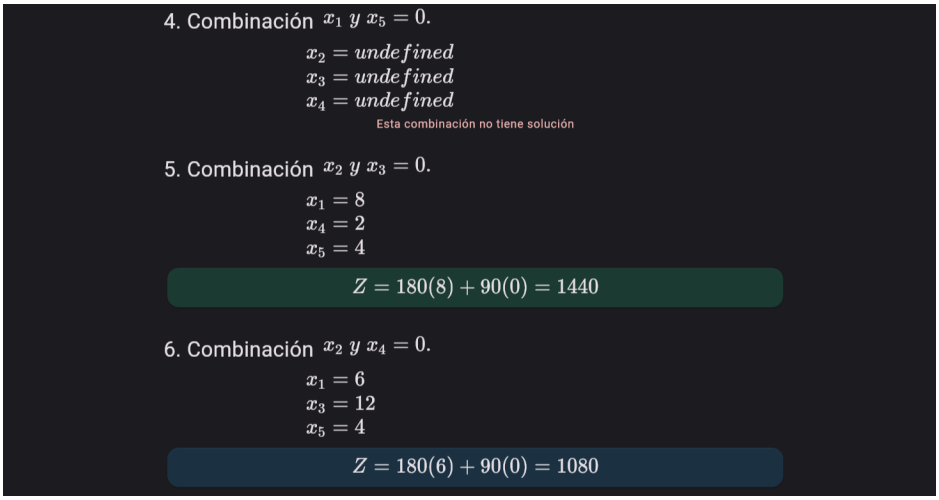


Figura 3.26: Combinaciones 4, 5 y 6.

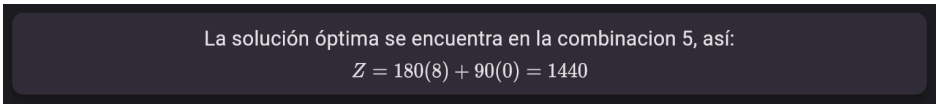


Figura 3.27: Respuesta en forma de texto.

d. Módulo de método gráfico

Para la representación de la respuesta en método gráfico, se extiende la clase `CustomPainter`, que provee del método `Paint(Canvas canvas, Size size)` [72], y entrega acceso al "lienzo" sobre el cuál se puede trazar cualquier diseño.

```
class GraphicProcessPainter extends CustomPainter {
GraphicProcessPainter({required this.theme, required
 ↪  this.answerData});
final ThemeData theme;
//GraphicDataModel tiene todos los datos necesarios
//para presentar la respuesta.
final GraphicDataModel answerData;

@override
void paint(Canvas canvas, Size size) {
//Lógica para trazar diseño
}
}
```

El controlador para el modo gráfico, es el encargado de hacer todos los cálculos necesarios, para hallar puntos de intersección entre restricciones y con el primer cuadrante del plano cartesiano, soluciones posibles y solución óptima. El objeto

resultante, se entrega a `GraphicProcessPainter` el cuál es responsable únicamente de trazar el gráfico.

```
@riverpod
class GraphicController extends _$GraphicController {
@override
GraphicDataModel build() {
 //Se obtinen todos los datos para construir `GraphicDataModel`
 ...
 //Objeto que se utilizará por `GraphicProcessPainter`
 return GraphicDataModel(
 xLimit: roundedLimits.x,
 yLimit: roundedLimits.y,
 answer: answerPoint,
 objectiveFunctionIntersections: answerLine,
 restrictions: restrictions,
 feasibleRegionMatrixPoints: feasibleRegion,
 compliantIntersections: compliantIntersections,
 );
 }
}
```

La vista que presenta la respuesta del proceso gráfico, muestra el gráfico mismo dentro de un visor interactivo, y por debajo de este, la respuesta en formato de texto como muestra la figura 3.28.

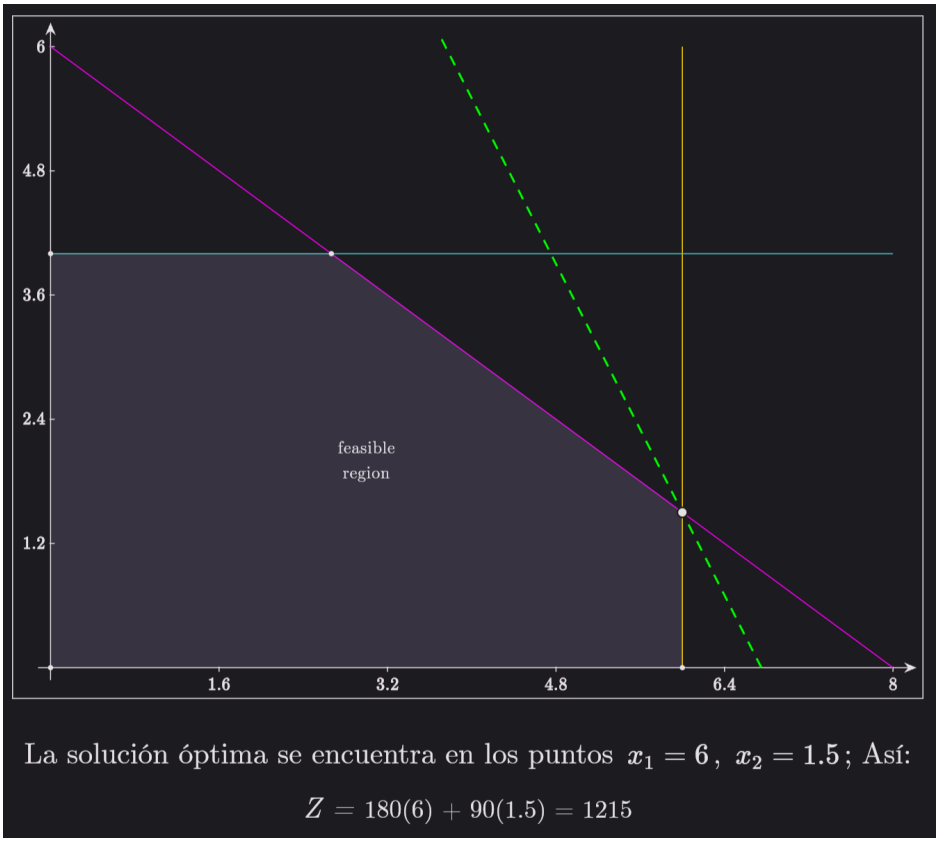


Figura 3.28: Respuesta método gráfico.

e. Módulo de método simplex

El método Simplex, hace uso de las 2 fases cuando por lo menos una de las restricciones no se define con el operador "≤". Cada fase tiene un método para computar la respuesta, y el controlador se encarga de decidir si se necesita una o dos fases, y de llamar a los métodos correspondientes.

```
@riverpod
class SimplexController extends _$SimplexController {
//Variables privadas, distinguen datos entre fases.
List<TabularFormInformation> _followingTableaus = [];
List<TabularFormInformation> _firstPhaseTableaus = [];
List<int> _artificialVariablesIndexes = [];
List<Point<int>> _pivotsCoordinates = [];
List<Point<int>> _firstPhasePivotsCoordinates = [];

@override
SimplexDataModel build() {
final (tabularFormInfo, artificialVariablesIndexes)
= ref.watch(toTabularFormProvider);
_followingTableaus = [tabularFormInfo];
_artificialVariablesIndexes = artificialVariablesIndexes;
return computeAnswer();
}

SimplexDataModel computeAnswer() {
//verifica si necesita 2 fases y calcula la respuesta
//teniendo en cuenta el objetivo.
...
}
}
```

La vista para el método simplex, muestra la respuesta en una lista de tablas, que muestran el proceso completo para alcanzar la respuesta, e incluye las 2 fases cuando existen 3.29. Las tablas tienen resaltadas las filas y columnas de pivote, para cada iteración. La última tabla en cambio resalta los valores que deben tomar las variables para alcanzar la solución óptima y la respuesta misma 3.31. Además, se presenta al final, la respuesta en formato de texto, como se muestra en la figura 3.30.

Fase 1

Iteración 1

	x_1	x_2	x_3	x_4	x_5	x_6	Lado derecho
Z	0	-1	0	0	1	0	-4
x_3	6	8	1	0	0	0	48
x_4	1	0	0	1	0	0	6
x_6	0	1	0	0	-1	1	4

Iteración 2

	x_1	x_2	x_3	x_4	x_5	x_6	Lado derecho
Z	180	90	0	0	0	1	0
x_3	6	0	1	0	8	-8	16
x_4	1	0	0	1	0	0	6
x_2	0	1	0	0	-1	1	4

Figura 3.29: Representación tabular de la fase 1.

Fase 2

Iteración 1

	x_1	x_2	x_3	x_4	x_5	Lado derecho
Z	-180	0	0	0	-90	360
x_3	6	0	1	0	8	16
x_4	1	0	0	1	0	6
x_2	0	1	0	0	-1	4

Iteración 2

	x_1	x_2	x_3	x_4	x_5	Lado derecho
Z	0	0	30	0	150	840
x_1	1	0	0.167	0	1.333	2.667
x_4	0	0	-0.167	1	-1.333	3.333
x_2	0	1	0	0	-1	4

Figura 3.30: Representación tabular de la fase 2.

La solución óptima se encuentra en los puntos $x_1 = 8\,,\ x_2 = 0\,$; Así:

$$Z = 180(8) + 90(0) = 1440$$

Figura 3.31: Respuesta método simplex.

f. Importar los datos de un problema

Para poder importar los datos de un ejercicio, el usuario, lo puede hacer, cargando de un archivo de texto, con extensión "`txt`" o "`json`". Los contenidos del archivo deben estar en formato `JSON` y debe contener los datos de un problema de programación lineal, según se muestra en la sección de ayuda 3.32.

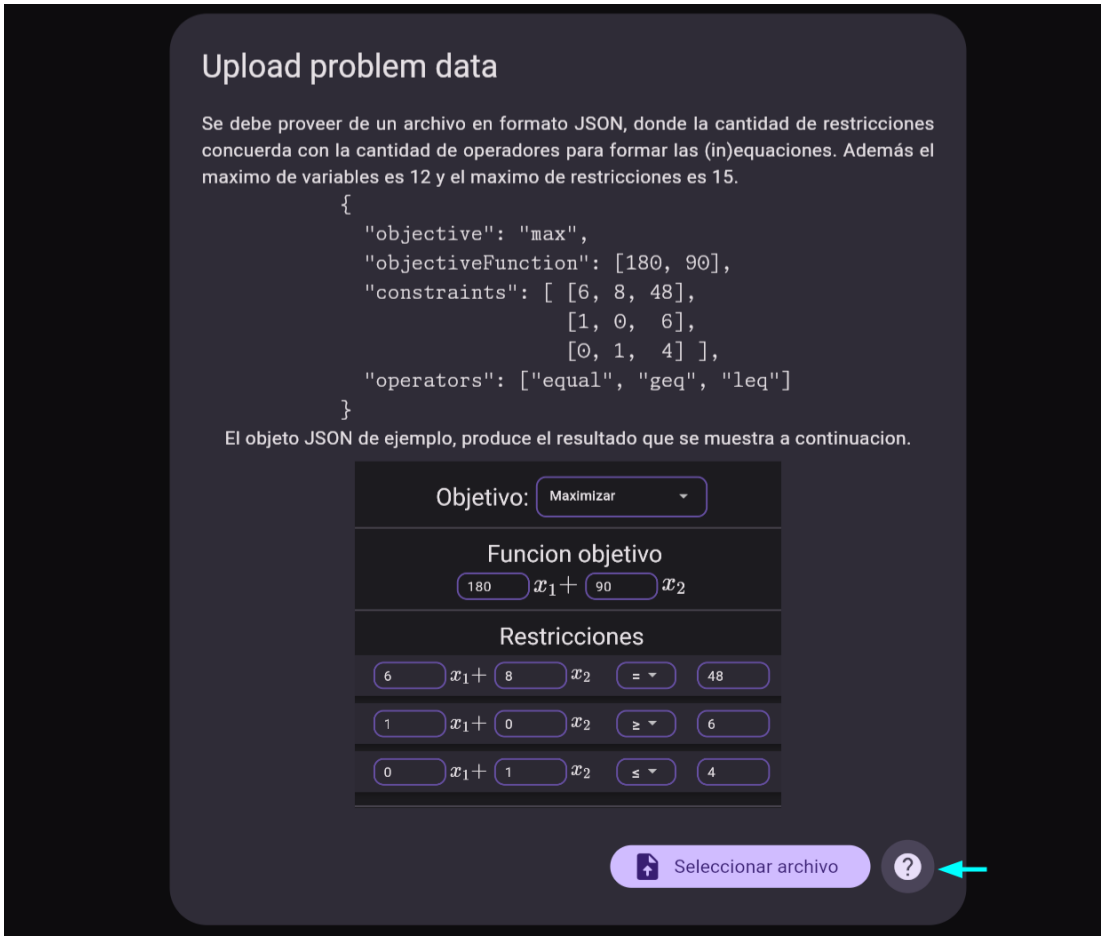


Figura 3.32: Diálogo para importar datos.

Otra forma de importar datos, es usar la barra de direcciones del navegador web, cuando se usa la versión web de la aplicación. En la ruta "`data-entry`", la dirección soporta el parámetro de búsqueda "`data`", al que se puede asignar el cuerpo del problema en el mismo formato `JSON` que se usa para importar desde un archivo.

```
https://linear--programming.web.app/data-entry?
data={"objective":"max","objectiveFunction":[180,90]
```

,"constraints":[[6,8,48],[1,0,6],[0,1,4]],"operators":
["equal","geq","leq"]}

g. Compartir los datos de un problema

Para compartir los datos de un problema, una vez que está resuelto, se provee al usuario la opción de compartir en la barra de título de la aplicación. Este botón copia un enlace al portapapeles, el enlace contiene los datos del problema.

```
share() async {
    try {
        const String path = '/$routeDataEntry';
        final String query =
        ↪  jsonEncode(ref.read(dataEntryControllerProvider).toJson());
        final uri = Uri.https(authority, path, {'data': query});
        //Copia al portapapeles el vínculo codificado como texto
        await Clipboard.setData(ClipboardData(text:
        ↪  uri.toString()));
        //Muestra un mensaje de éxito
        ...
      } catch (e) {
        ...
      }
  }
```

De esta forma el enlace puede ser guardado como favorito, o compartido con otros usuarios. El enlace es codificado con caracteres válidos para un vínculo web, de tal manera que el usuario no experimente inconsistencia. Por ejemplo:

https://linear--programming.web.app/data-entry?data=%7B%
22objective%22%3A%22max%22%2C%22objectiveFunction%22%
3A%5B180%2C90%5D%2C%22operators%22%3A%5B%22equal%22%2C%
22geq%22%2C%22leq%22%5D%2C%22constraints%22%3A%5B%5B6%
2C8%2C48%5D%2C%5B1%2C0%2C6%5D%2C%5B0%2C1%2C4%5D%5D%7D

3.3.8 Desarrollo de requisitos no funcionales.

a. Usabilidad.

La interfaz se diseña con los componentes de Material 3, que sigue principios de accesibilidad como jerarquías visuales, paletas de colores que promueven la

legibilidad, agrupación y orden de enfoque. Se prestan todas las opciones de navegación para cada plataforma, por ejemplo en Windows, Linux y demás sistemas operativos de que hacen uso de teclado físico, se hacer uso de teclas como `Tab` y "`Shift + Tab`" para navegar entre elementos interactivos, y `Enter` o `Espacio` para "Aceptar".

En dispositivos móviles, se hace uso del teclado virtual, y de gestos como `Tap` y `Swipe` para interactuar con los elementos. Donde corresponde, el teclado virtual muestra el tipo de teclado adecuado para el tipo de dato que se espera del usuario, por ejemplo, cambio a teclado numérico, o el cambio de la tecla `Enter` por la tecla `Next` o `Done`.

Figura 3.33: Teclado virtual numérico, con tecla "next".

b. Rendimiento.

El framework Flutter, es muy eficiente en el uso de recursos, como se menciona en la sección 3.2.1 en la que se explica el renderizado y los árboles de widgets. Además, en el desarrollo de la aplicación, se hace uso del paquete `riverpod`, que permite hacer "cache" y no tener que volver a calcular datos que no han cambiado. Por otro lado, solo se reconstruye los widgets que han cambiado.

```dart
class EntrySizeWidget extends ConsumerStatefulWidget {
    @override
    Widget build(BuildContext context, WidgetRef ref) {
        //Solo se reconstruye el widget cuando cambia el valor
        //de la variable dataEntrySizeState
        final dataEntrySizeState =
        ↪  ref.watch(entrySizeControllerProvider);
        return Center(
        //Se usa la variable dataEntrySizeState
        child: ...
        )
      }
  }
```

c. Mantenibilidad.

La aplicación está escrita en Dart, que es un lenguaje de programación orientado a objetos, que permite la reutilización de código, y la organización de este en clases y paquetes.

En el desarrollo de la aplicación, se hace uso de paquetes de terceros, que proveen de funcionalidades específicas, como el paquete `equations` que permite resolver sistemas de ecuaciones, lo que disminuye la carga de trabajo del desarrollador.

Además, se hace uso de la metodología de desarrollo Kanban, que permite una organización eficiente de las tareas, y un seguimiento claro del progreso del proyecto.

Por otro lado, se hace uso de la arquetectura de software MVC, que permite separar la lógica de la interfaz de usuario, de la lógica de negocio, y de la lógica de acceso a datos. Esto permite que el código sea más fácil de mantener, ya que se puede modificar una parte del código, sin afectar a las otras. Esto se puede evidenciar aún en la forma de la estructura del directorio del proyecto 3.34.

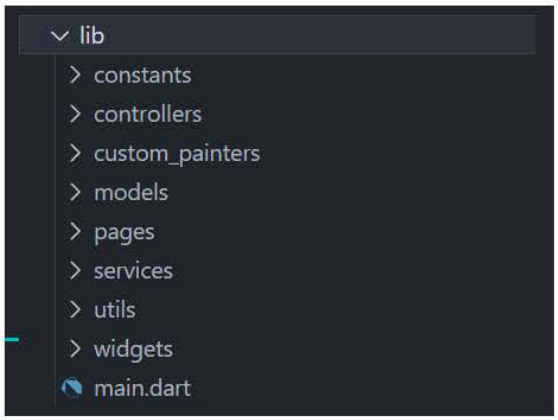


Figura 3.34: Estructura del directorio "lib" del proyecto.

d. Compatibilidad.

La aplicación es compatible con navegadores web y sistemas operativos móviles y de escritorio. Se hace uso de versiones recientes de Flutter y Dart.

e. Disponibilidad.

El sistema una vez ha sido descargado, no requiere de conexión a internet para su uso. Además, el usuario puede usar la aplicación sin necesidad de registrarse, y puede guardar los datos de un problema. Si para un usuario le fuera difícil instalar la aplicación, puede usar la versión web, que no requiere de instalación, y puede ser usada desde cualquier navegador web.

3.3.9 Realizar pruebas de la aplicación.

Se realizaron varias pruebas unitarias para los diferentes procesos de resolución de problemas, con estos se intenta cubrir la mayor cantidad de casos posibles, para garantizar que la aplicación funcione correctamente. Se usó el paquete `test` de Dart, que permite escribir varios tipos de pruebas.

Para realizar las pruebas de los "`Providers`" generados por Riverpod, manteniéndoles en aislamiento, se crearon "Mocks". Estas son clases que permiten simular el comportamiento de los "`Providers`" reales, pero con datos controlados.

Por ejemplo, se puede simular el comportamiento de `EntrySizeController` con el siguiente "Mock", donde además se provee un valor inicial para el estado del "`Provider`". Para este ejemplo se usa para el valor inicial, 2 variables, 3 restricciones y por lo tanto se muestra la selección del proceso de resolución.

```
class MockSizeEntry2Var3Con extends EntrySizeController {
    @override
    EntrySizeState build() => const EntrySizeState(
    variables: 2,
    constraints: 3,
    showProcess: true,
    );
  }
```

Otro ejemplo es el siguiente "Mock" para `DataEntryController`, donde se provee un valor inicial para el estado del "`Provider`". Para este Mock se tiene en cuenta que el usuario ya ha ingresado los datos de las dimensiones del problema, y se ha movido a la siguiente fase, donde se ingresan los datos del problema.

```
class MockDataMaximize2Var3Con extends DataEntryController {
@override
DataEntryModel build() {
return const DataEntryModel(
objective: Objectives.max,
objectiveFunction: [180, 90],
operators: [
Operators.leq,
Operators.leq,
Operators.leq,
],
constraints: [
[6, 8, 48],
[1, 0, 6],
[0, 1, 4],
],
);
}
}
```

Las pruebas de "Providers" requiere de un objeto `ProviderContainer`, que es el encargado de proveer los "Mocks" como una lista de "`Overrides`".

```
ProviderContainer createProviderContainer({
ProviderContainer? parent,
List<Override> overrides = const [],
List<ProviderObserver>? observers,
}) {
final container = ProviderContainer(
parent: parent,
overrides: overrides,
```

```
observers: observers,
);
addTearDown(container.dispose);
return container;
}
```

Para las pruebas unitarias se usan tanto los "Mocks" como el objeto de tipo `ProviderContainer`. En el ejemplo que se muestra a continuación, se verifica, en las pruebas del proceso algebráico:

- Que la transformación de los datos del problema a forma estándar, se haga correctamente.
- Que la transformación de los datos del problema a llevar variables de ajuste, se haga correctamente.
- Que la lista de variables del lado derecho de las restricciones, se genere correctamente.
- Que los datos del problema se muestren correctamente en forma de texto.

```
test('toStandardForm minimize', () {
final overrides = <Override>[
 dataEntryControllerProvider.overrideWith(
 MockDataMinimize2Var3Con.new,
 ),
 ];
final container = createProviderContainer(
overrides: overrides,
);
final algebraicData = container
.read(algebraicControllerProvider.notifier)
.toStandardForm();
expect(algebraicData.standardForm, [
 [0.3, 0.1, 1.0, 0.0, 0.0, 2.7],
 [0.5, 0.5, 0.0, 1.0, 0.0, 6.0],
 [0.6, 0.4, 0.0, 0.0, -1.0, 6.0],
 ]);
expect(algebraicData.constraintWithSlack, [
 [0.3, 0.1, 1.0, 0.0, 0.0],
 [0.5, 0.5, 0.0, 1.0, 0.0],
 [0.6, 0.4, 0.0, 0.0, -1.0]
 ]);
expect(algebraicData.rightSide, [2.7, 6, 6]);
expect(
algebraicData.rightSideString,
['\\;2.7', '\\;6', '\\;6'],
);
expect(algebraicData.constraintsString, [
```

```
['+0.3x_{1}', '+0.5x_{1}', '+0.6x_{1}'],
['+0.1x_{2}', '+0.5x_{2}', '+0.4x_{2}'],
['+x_{3}', '\\;', '\\;'],
['\\;', '+x_{4}', '\\;'],
['\\;', '\\;', '-x_{5}'],
['\\;=', '\\;=', '\\;='],
]);
expect(
algebraicData.combinationsEquation,
'C(n,r) = \\frac{0!}{0!(0-0)!} = 1',
);
});
```

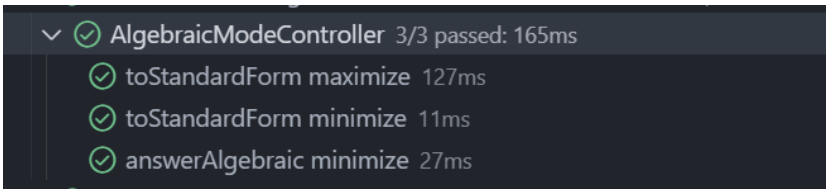


Figura 3.35: Resultado pruebas proceso algebráico.

Las pruebas unitarias para el proceso gráfico, se enfocan en verificar que los datos del problema se transformen correctamente en datos para el gráfico. En el ejemplo que se muestra a continuación, se verifica que las intersecciones entre restricciones se calculen correctamente.

```
test('getIntersectionsBetweenCostraints', () {
final overrides = <Override>[
 getBiggestIntersectionsOnAxesProvider
 .overrideWithValue(
 const Point<double>(6, 8),
 ),
 getIntersectionsOnAxesProvider.overrideWithValue(
 const [
 Point<double>(8.0, 0.0),
 Point<double>(0.0, 6.0),
 Point<double>(6.0, 0.0),
 Point<double>(0.0, 4.0),
 ],
 ),
 ];
final container = createProviderContainer(overrides: overrides);
final getIntersectionsBetweenCostraints = container.read(
getIntersectionsBetweenCostraintsProvider,
);
expect(
getIntersectionsBetweenCostraints,
const [
 Point(8.0, 0.0),
 Point(0.0, 6.0),
 Point(6.0, 0.0),
 Point(0.0, 4.0),
```

```
 ],
);
});
```

Otro ejemplo relevante, es verificar que se encuentre la solución óptima correctamente, asimismo las interserccines de ésta, con el primer cuadrante del plano cartesiano.

```
test('getOptimalAnswer maximize', () {
final overrides = <Override>[
 dataEntryControllerProvider
 .overrideWith(MockDataMaximize2Var7Con.new),
 getBiggestIntersectionsOnAxesProvider.overrideWithValue(
 const Point<double>(10, 9),
 ),
 getCompliantIntersectionsProvider.overrideWithValue(
 const [
 Point<double>(5.0, 3.0),
 Point<double>(3.0, 4.2),
 ],
 ),
 ];
final container = createProviderContainer(overrides: overrides);
final getOptimalAnswer = container.read(getOptimalAnswerProvider);
expect(getOptimalAnswer, const Point<double>(5, 3));
});
```

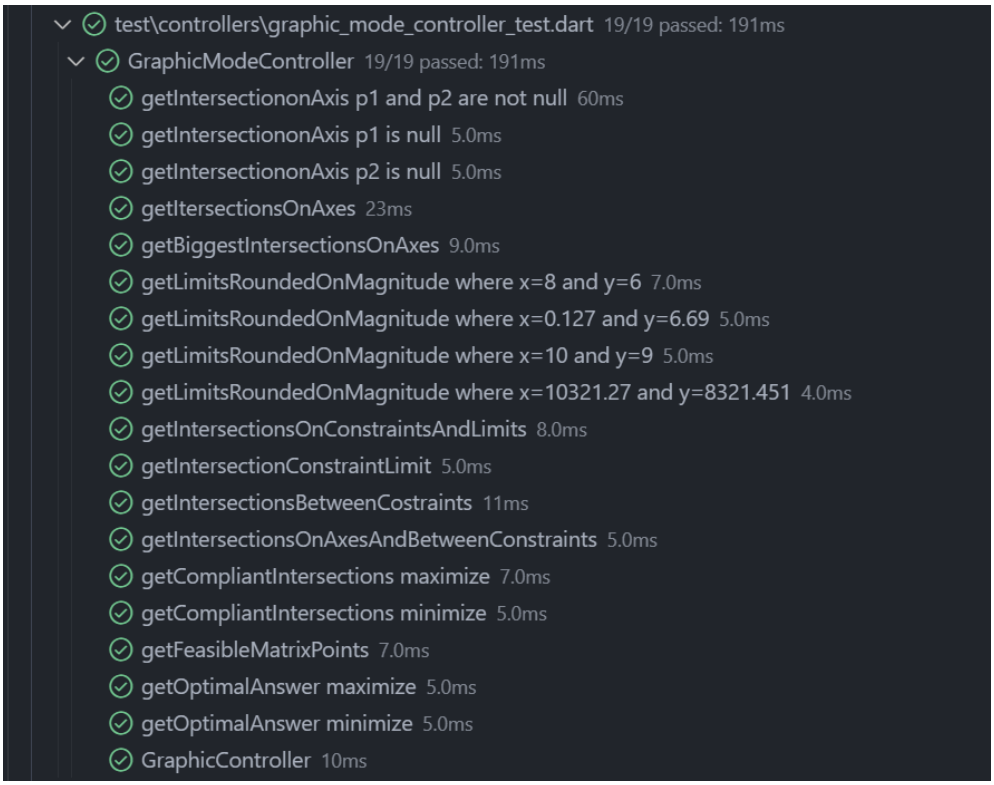


Figura 3.36: Resultado pruebas proceso gráfico.

Por otro lado, las pruebas unitarias para el proceso simplex, se enfocan en verificar que

los datos del problema se transformen correctamente en datos para las tablas, y que las tablas se generen correctamente. A continuación, se muestra cómo se verifica que se calculen las dimensiones de la tabla.

```
test('calculateSize 3Variables 3Constraints', () {
final overrides = <Override>[
 entrySizeControllerProvider.overrideWith(
 MockSizeEntry3Var3Con.new,
 ),
 ];
final container = createProviderContainer(
overrides: overrides,
);
final result = container.read(calculateSizeProvider);
expect(result.x, 4);
expect(result.y, 7);
});
```

En el ejemplo que se muestra a continuación, se verifica que en la tabla que se encuentre la respuesta, y que las tablas para presentar en la interfaz se hayan generado correctamente.

```
test('simplexController max 2Var3Con', () {
final overrides = <Override>[
 calculateSizeProvider
 .overrideWithValue(const Point(4, 6)),
 dataEntryControllerProvider
 .overrideWith(MockDataMaximize2Var3Con.new),
 ];
final container = createProviderContainer(
overrides: overrides,
);
final simplexData = container
.read(simplexControllerProvider.notifier)
.computeAnswer();
expect(
simplexData.pivotsCoordinates,
[const Point<int>(0, 2), const Point<int>(1, 1)],
);
expect(
simplexData.tableaus,
[
 [
 ['Z', '-180', '-90', '0', '0', '0', '0'],
 ['x_3', '6', '8', '1', '0', '0', '48'],
 ['x_4', '1', '0', '0', '1', '0', '6'],
 ['x_5', '0', '1', '0', '0', '1', '4']
 ],
 [
 ['Z', '0', '-90', '0', '180', '0', '1080'],
 ['x_3', '0', '8', '1', '-6', '0', '12'],
 ['x_1', '1', '0', '0', '1', '0', '6'],
 ['x_5', '0', '1', '0', '0', '1', '4']
 ],
```

```
  [
  ['Z', '0', '0', '11.25', '112.5', '0', '1215'],
  ['x_2', '0', '1', '0.125', '-0.75', '0', '1.5'],
  ['x_1', '1', '0', '0', '1', '0', '6'],
  ['x_5', '0', '0', '-0.125', '0.75', '1', '2.5']
  ]
  ],
);
});
```

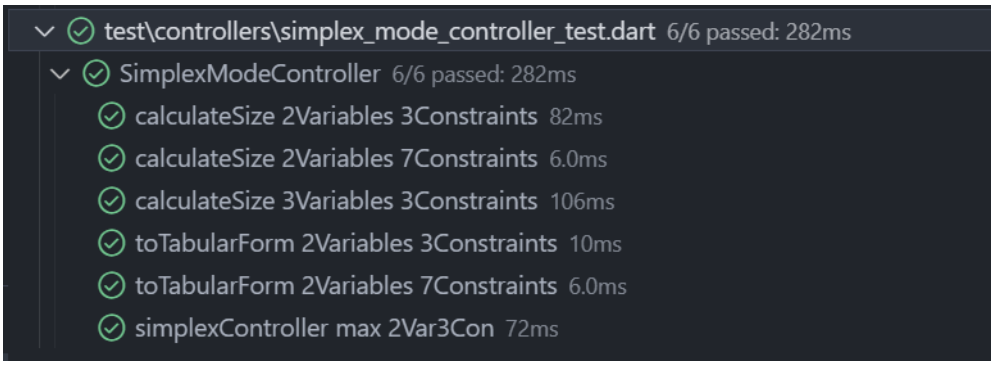


Figura 3.37: Resultado pruebas proceso simplex.

3.3.10 Implantar la aplicación multiplataforma.

Con el propósito de implantar la aplicación multiplataforma, se procedió a configurar el despliegue del sistema, para las principales plataformas. Para ello, cada plataforma requiere de un proceso de configuración y puesta en producción específico, de lo cual se detalla a continuación lo mas relevante de cada proceso.

a. Despliegue de la plataforma web.

Para el despliegue de la plataforma web, se procedió a configurar Firebase Hosting, el cual permite alojar y desplegar aplicaciones web de forma sencilla y eficiente. Esto tiene como requisito, tener instalado Node.js, npm y Firebase CLI. Para instalar Firebase CLI, se ejecutó el comando `npm install -g firebase-tools` en la terminal. A continuación, se procedió a configurar el proyecto de Firebase, ejecutando el comando `firebase init` en la terminal. Este comando, permite seleccionar las opciones de configuración del proyecto, como se muestra a en 3.39.

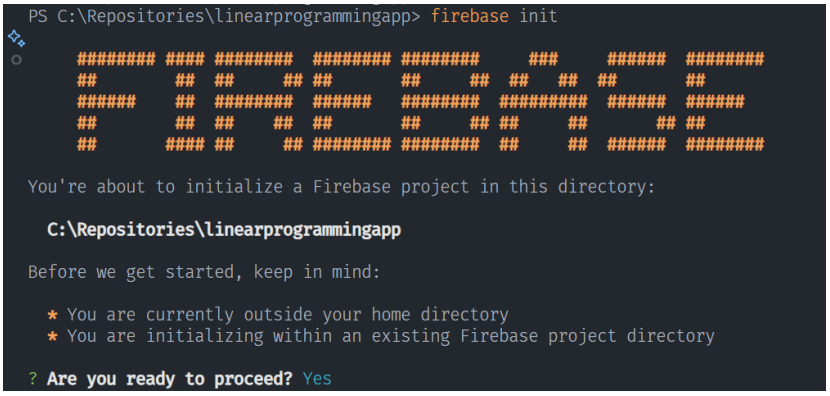


Figura 3.38: Inicialización de firebase.

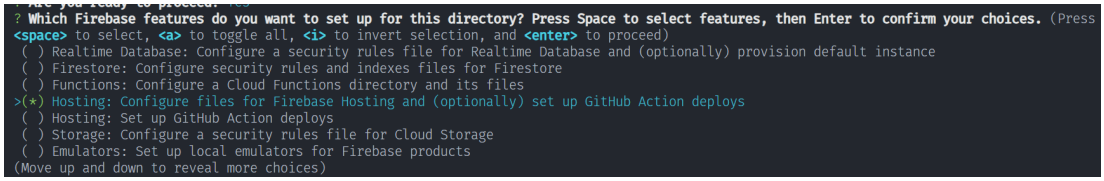


Figura 3.39: Selección de características de firebase.

Ya que el presente proyecto es creado con flutter y usa la estrategia de ruta para la barra de direcciones en el navegador web, es necesario seleccionar la configuración de "single-page app" 3.40. Esto permite que firebase redirija todas las rutas a `index.html`, dejando que flutter se encargue de manejar las rutas.

Figura 3.40: Inicialización de firebase completa.

Antes de desplegar la aplicación, se procedió a compilar el proyecto, ejecutando el comando `flutter build web --release` en la terminal, en la raíz del proyecto. Por último, se procedió a desplegar la aplicación web, ejecutando el comando `firebase deploy` en la terminal. Una vez desplegada la aplicación, firebase devuelve la URL de la aplicación, como se muestra en 3.41, la cual es pública y accesible desde cualquier navegador web, tal como se muestra en 3.42.

```
PS C:\Repositories\linearprogrammingapp> firebase deploy

=== Deploying to 'linear--programming'...

i  deploying hosting
i  hosting[linear--programming]: beginning deploy...
i  hosting[linear--programming]: found 53 files in build/web/
+  hosting[linear--programming]: file upload complete
i  hosting[linear--programming]: finalizing version...
+  hosting[linear--programming]: version finalized
i  hosting[linear--programming]: releasing new version...
+  hosting[linear--programming]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/linear--programming/overview
Hosting URL: https://linear--programming.web.app
PS C:\Repositories\linearprogrammingapp>
```

Figura 3.41: Despliegue con firebase.

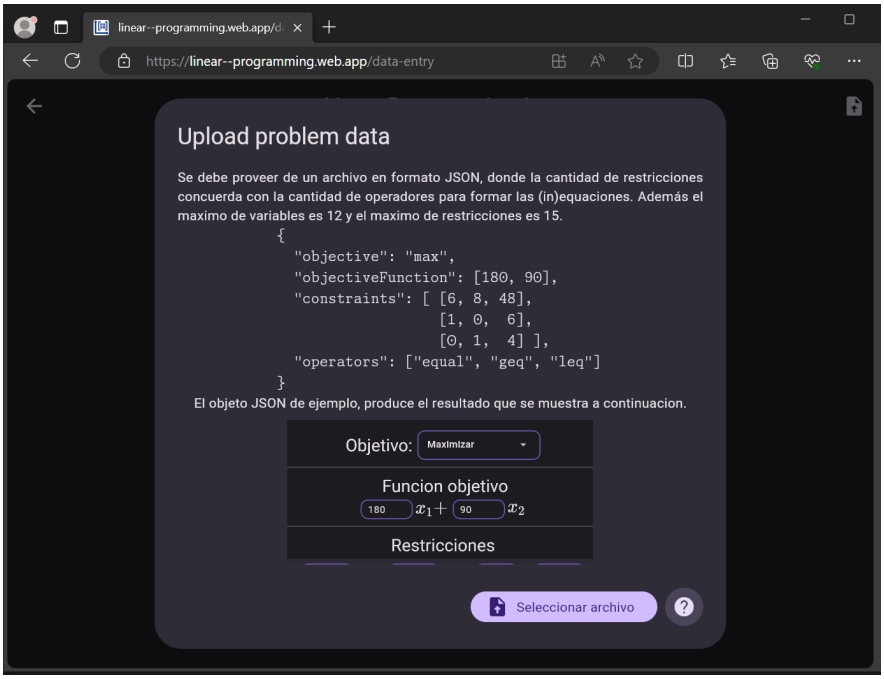


Figura 3.42: Aplicación en la web.

b. Despliegue de la plataforma Android.

Para el despliegue de la plataforma Android, se debe primeramente "firmar" la aplicación, para esto es necesario generar un "keystore" de Android.

```
keytool -genkey -v -keystore %userprofile%/upload-keystore.jks ^
-storetype JKS -keyalg RSA -keysize 2048 -validity 10000 ^
-alias upload
```

Los detalles de este paso se deben referenciar en el arcrhivo `./android/app/build.gradle`

```
signingConfigs {
release {
keyAlias keystoreProperties['keyAlias']
keyPassword keystoreProperties['keyPassword']
storeFile keystoreProperties['storeFile'] ?
 ↪  file(keystoreProperties['storeFile']) : null
storePassword keystoreProperties['storePassword']
}
}
buildTypes {
release {
signingConfig signingConfigs.release
}
}
```

Para compilar la aplicación para producción, se ejecuta el comando `flutter build appbundle --release`

Con esto, en Play Console, se procede a crear un nuevo lanzamiento, y se sube el archivo `.aab` generado en el paso anterior, cual debe tener el mismo "package name" que el proyecto en Play Console. A continuación, se muestran imágenes del proceso de despliegue en Play Console.

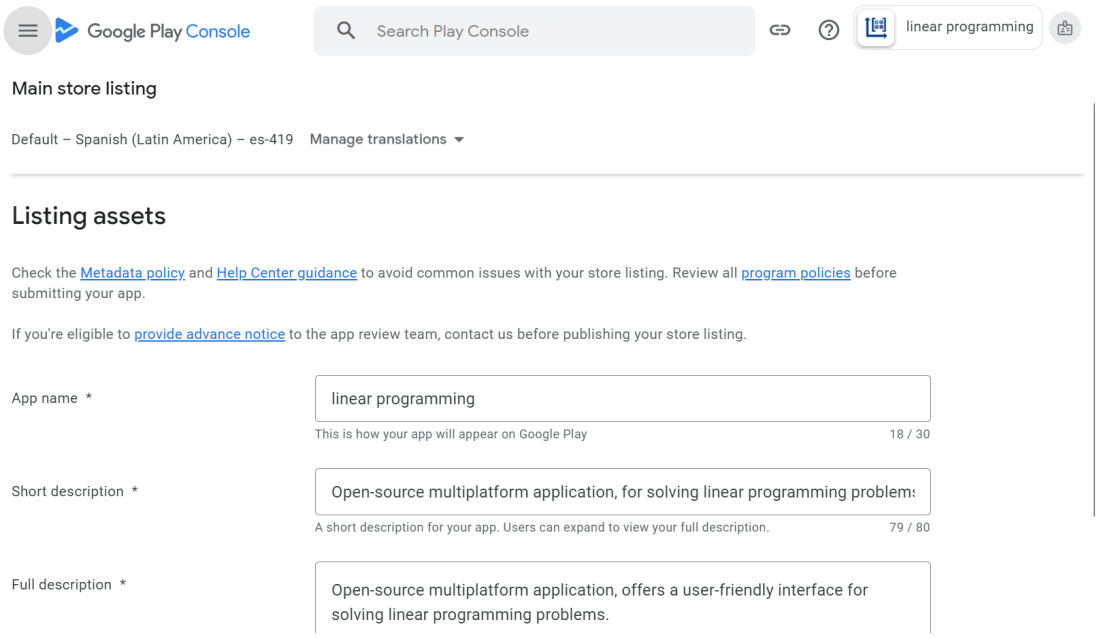


Figura 3.43: Información de la aplicación.

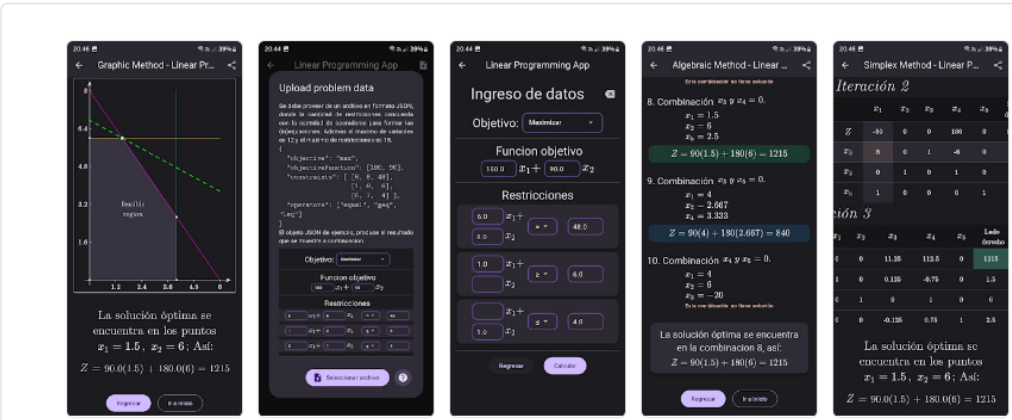
Graphics

Manage your app icon, screenshots, and videos to promote your app on Google Play. Review the content guidelines before uploading new graphics. If you add translations for your store listing without localized graphics, we will use the graphics from your default language.

App icon *

Your app icon must be a PNG or JPEG, up to 1 MB, 512 px by 512 px, and meet our design specifications and metadata policy

Figura 3.44: Icono de la aplicación.



Upload 2-8 phone screenshots. Screenshots must be PNG or JPEG, up to 8 MB each, 16:9 or 9:16 aspect ratio, with each side between 320 px and 3,840 px

Figura 3.45: Capturas de pantalla de la aplicación.

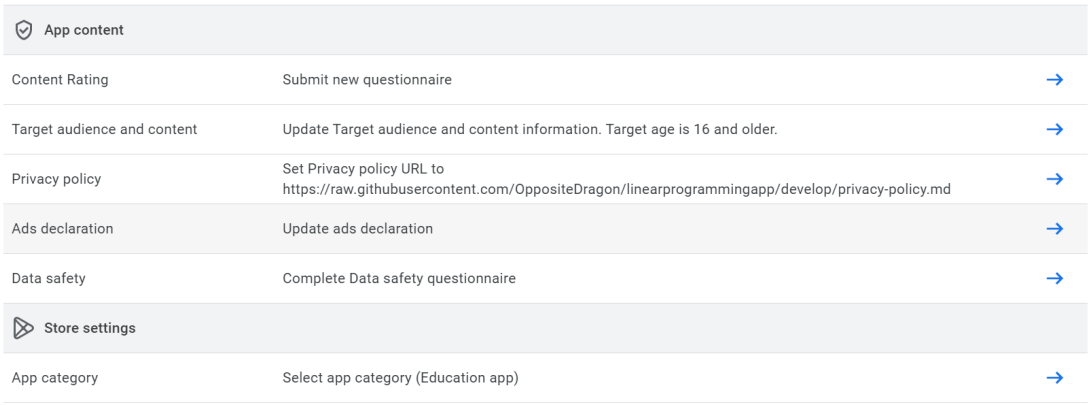


Figura 3.46: Contenido de la aplicación.

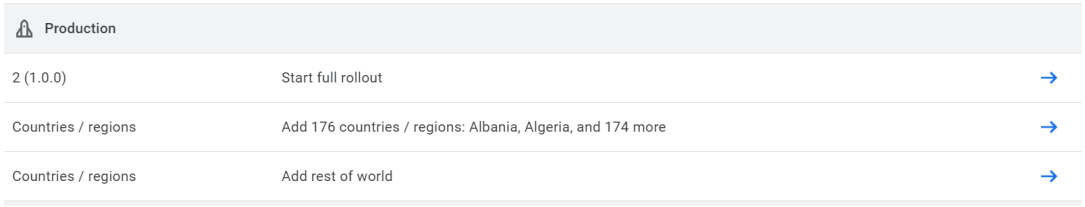


Figura 3.47: Puesta en producción de la aplicación.

Una vez que la aplicación ha pasado la certificación de Google, es decir que cumple con las reglas impuestas por la tienda, esta se publica en la tienda "Google Play", para que los usuarios puedan descargarla. A continuación, se muestra la presencia de la aplicación en la tienda de Google.

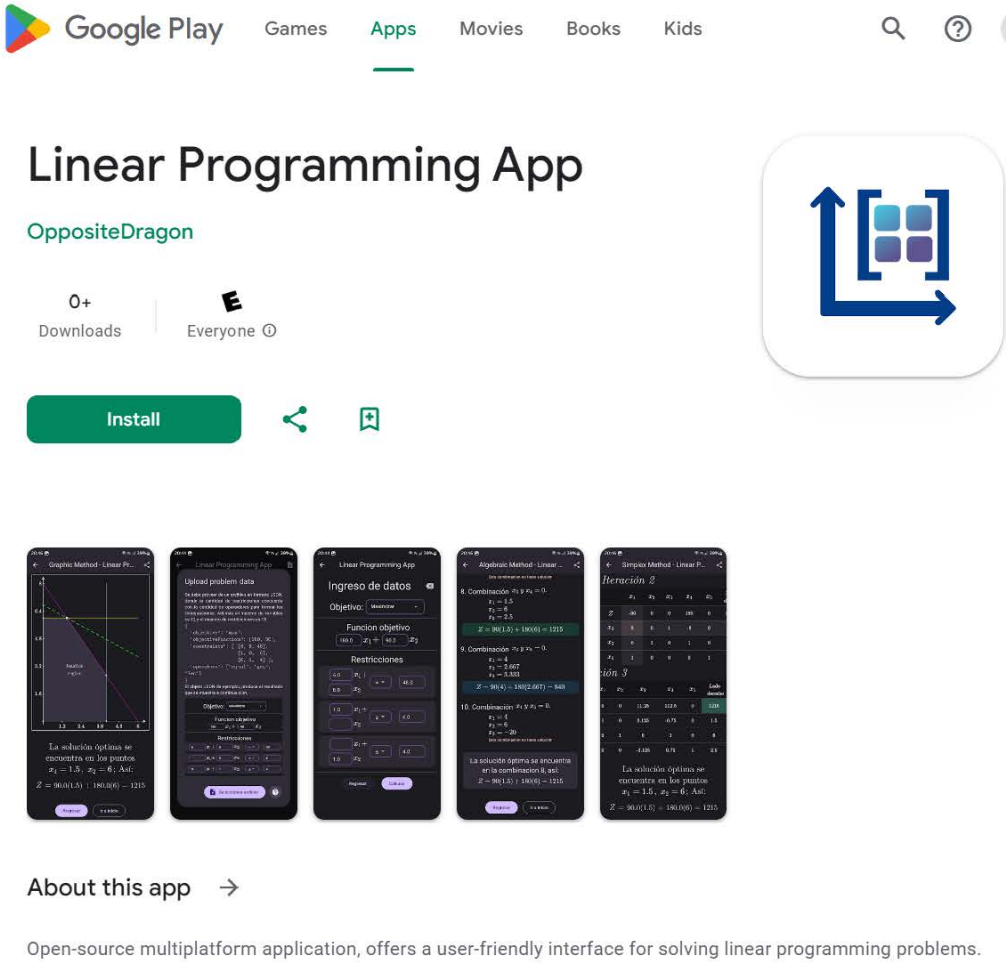


Figura 3.48: Aplicación en la tienda de Google.

c. Despliegue de otras plataformas.

Para el despliegue de la aplicación en la tienda de Windows, se puede generar un paquete `.msix`, con la ayuda del paquete msix [73], el cual se agrega como dependencia a la aplicación con el comando `flutter pub add --dev msix`. Para poder generar un paquete válido, es necesario proveerle de datos de identidad desde el portal de desarrolladores de Microsoft Partner Center.

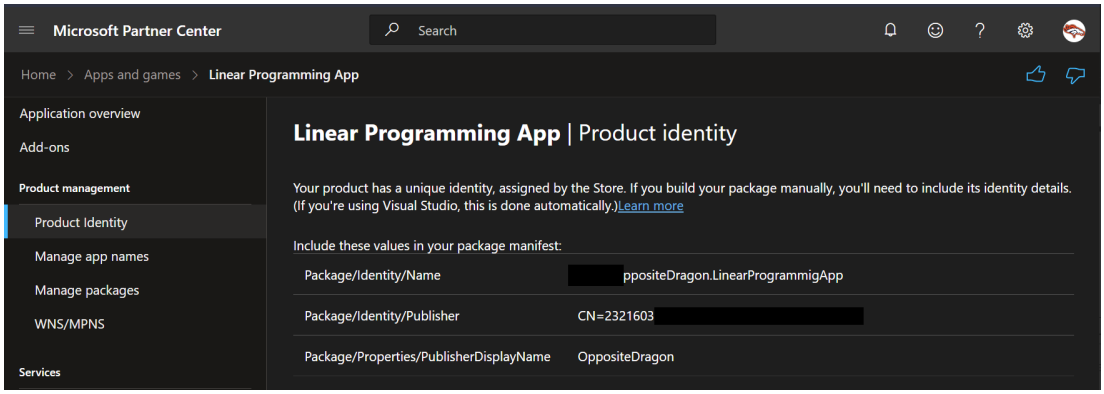


Figura 3.49: Identidad del producto.

Estos datos se agregan junto a los demás datos de configuración de msix, en el archivo `pubspec.yaml` en la raíz del proyecto, como se muestra a continuación.

```
msix_config:
display_name: Linear Programming App
publisher_display_name: OppositeDragon
identity_name: XXXXXXppositeDragon.LinearProgrammigApp
publisher: CN=2321603XXXXXXXXXXXXXXXXXXXXXXXXXXXX
msix_version: 1.0.0.0
languages: es-es, en-us
capabilities: internetClient
store: true
logo_path: C:/img/app_icon.png
```

Con estos datos, se procede a generar el paquete `.msix`, ejecutando el comando `dart run msix:create`, en la raíz del proyecto. Este comando genera el paquete `.msix` en la carpeta `.build/windows/x64/runner/Release`. A continuación, se muestran imágenes del proceso de despliegue en Microsoft Partner Center.

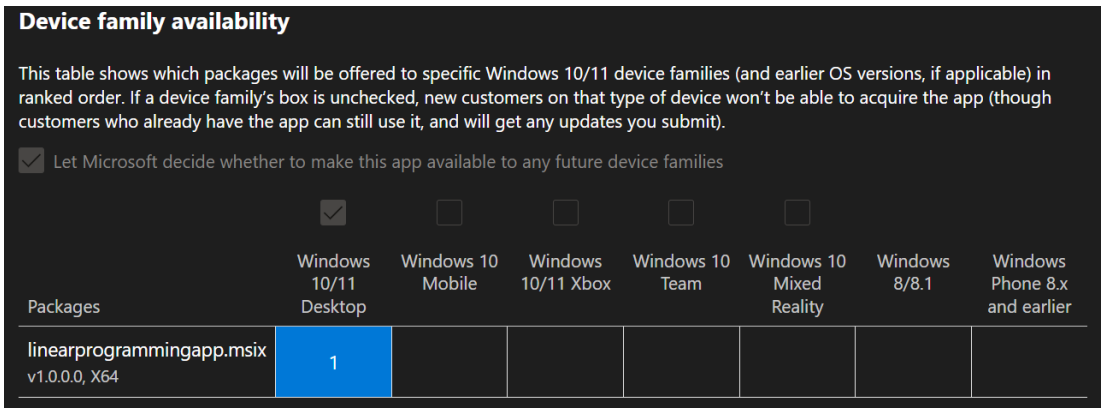


Figura 3.50: Carga del paquete msix.

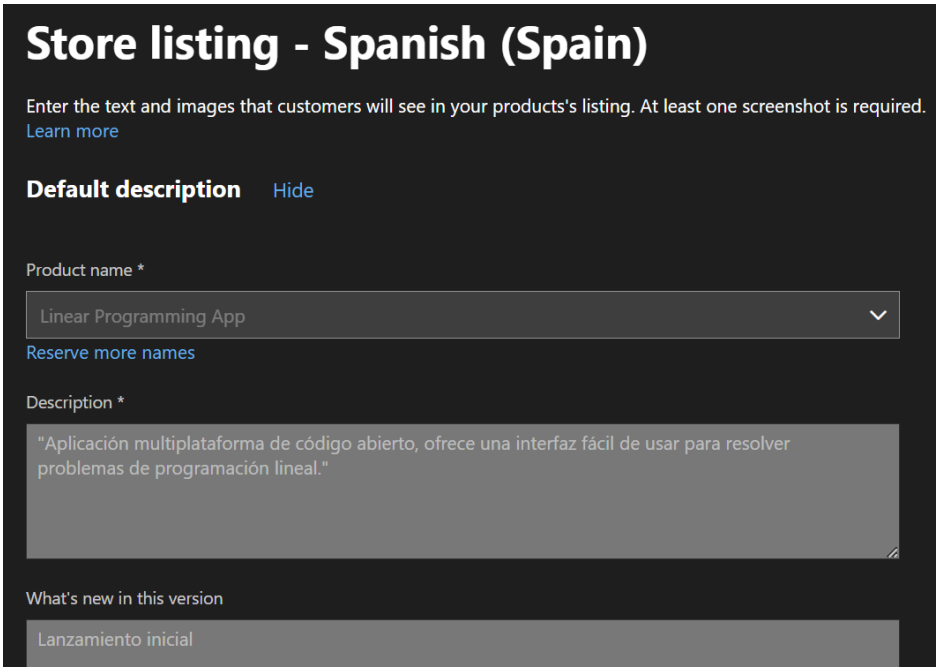


Figura 3.51: Datos del producto para la tienda.

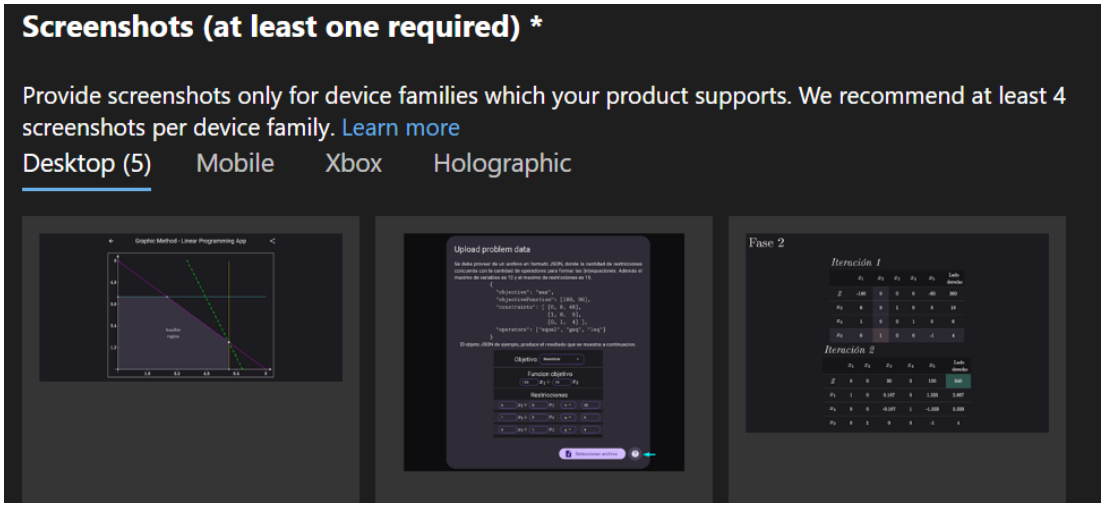


Figura 3.52: Capturas del producto para la tienda.

Una vez que Microsoft certifica que la aplicación que cumple con las reglas impuestas por la tienda, esta se publica en la tienda, para que los usuarios puedan descargarla. A continuación, se muestra la presencia de la aplicación en la tienda de Microsoft.

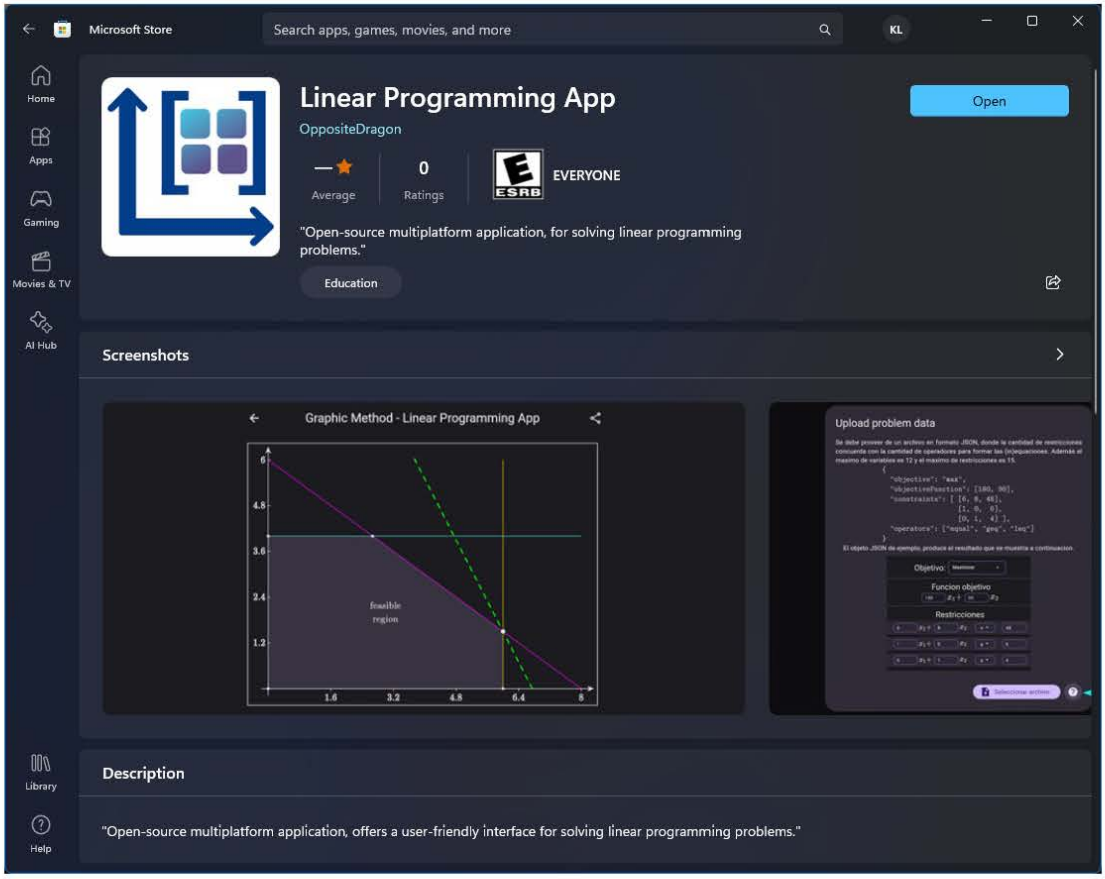


Figura 3.53: Aplicación en la tienda de Microsoft.

Para el despliegue de la aplicación en la plataforma **Linux**, se optó por distribuir directamente el ejecutable de la aplicación como un empaquetado `AppImage`, esto porque Linux siendo un sistema operativo mas abierto, no requiere de un proceso de certificación para distribuir aplicaciones, y sus usuarios están mas acostumbrados a instalar aplicaciones de esta forma.

Para generar el binario de la aplicación, se ejecuta el comando `flutter build linux --release` en la raíz del proyecto. Este comando genera el ejecutable en la carpeta `.build/linux/x64/release/bundle`.

Los contenidos de esta carpeta se copian al directorio `/LinearProgrammingApp.AppDir`, y allí se crean los archivos `AppRun` 3.54 y `.desktop` 3.55.

```
$ AppRun
#!/bin/sh
cd "$(dirname "$0")"
exec ./linearprogrammingapp

```

Figura 3.54: Contenidos de AppRun.

```
.desktop
[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Name=Linear Programming App
Comment=Open-source multiplatform application, for solving linear programming problems.
Exec=linearprogrammingapp %u
Icon=app_icon
Categories=Education;
```

Figura 3.55: Contenidos de ".desktop".

Para crear el archivo `.AppImage`, se ejecuta el comando `appimagetool-x86_64.AppImage /LinearProgrammingApp.AppDir linearprogrammingapp.AppImage`, que tomará los archivos en el directorio especificado, y prodicira el empaquetado paquetado que puede ser distribuido directamente.

```
pop@pop-os:/LinearProgrammingApp.AppDir$ ls -a
.  ..  app_icon.png  AppRun  data  .desktop  .DirIcon  lib  linearprogrammingapp  linearprogrammingapp.AppImage
```

Figura 3.56: Contenidos de "/LinearProgrammingApp.AppDir".

Como una opcion alternativa a las tiendas de aplicaciones de Windows y Android, y para facilitar la distribución del binario de la aplicación para la plataforma Linux, se libera el conjunto de archivos `.AppImage`, `.msix` y `.apk`, en la sección de lanzamientos del repositorio de GitHub.

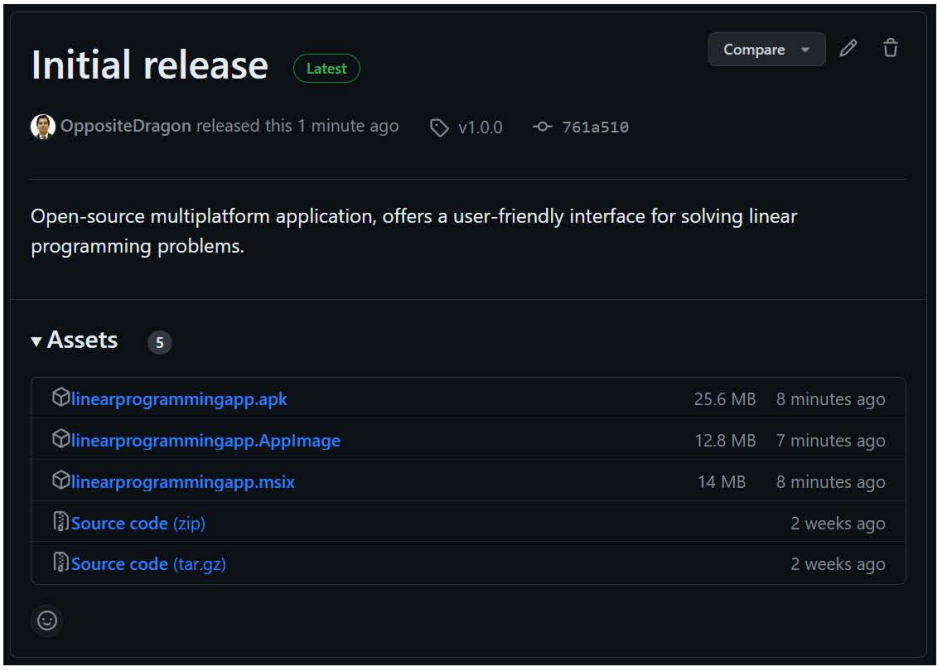


Figura 3.57: Lanzamiento inicial en GitHub.

3.4 Validación de la aplicación multiplataforma.

La validación del sistema multiplataforma de código abierto se resuelve con la resolución de ejercicios de programación lineal procedentes de fuentes bibliográficas y científicas reconocidas. Este procedimiento se orienta a asegurar la precisión y eficacia del sistema al enfrentar problemas reales extraídos de la literatura especializada en Investigación Operativa. Se seleccionarán casos de estudio representativos de manera cuidadosa con el fin de evaluar la capacidad de la aplicación para resolver con precisión diversos escenarios de programación lineal. La comparación de los resultados obtenidos por la aplicación con las soluciones propuestas en la literatura permitirá verificar la validez del sistema desarrollado, respaldando así su aplicabilidad y utilidad en entornos tanto académicos como prácticos.

Dr. Gupta en su libro **Operations Research** [74], en el capítulo 4, página 107, expone el problema de producción de medicinas, que intenta maximizar la ganancia de la empresa, cuyo modelo de programación linear completo, se muestra en la figura 3.58.

Hence the linear programming problem of the given problem is as follows :

$$\text{Max. } Z = 8x_1 + 7x_2$$

subject to constraints

$$3x_1 + x_2 \leq 66000$$
$$x_1 + x_2 \leq 45000$$
$$x_1 \leq 20000$$
$$x_2 \leq 40000$$

and $x_1 \geq 0, x_2 \geq 0$

Figura 3.58: Modelo de problema medicina [74].

En el libro indica como llegar a la solución óptima usando el método algebráico, y la respuesta se muestra en la figura 3.59, donde $Z = 325500$.

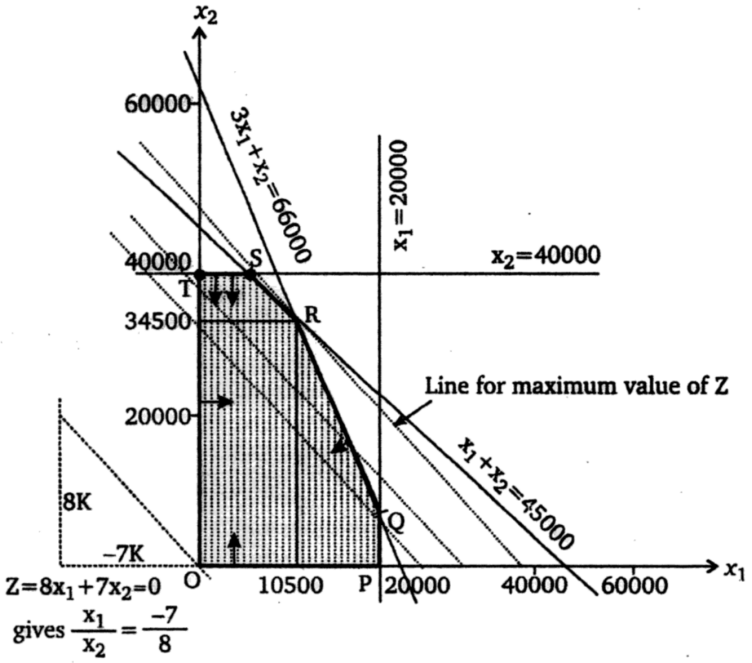


Figura 3.59: Respuesta gráfica de problema medicina [74].

Para demostrar las capacidades de la aplicación, se muestra la entrada de datos en la aplicación, para resolver este problema 3.58 y posterior a ello se muestra la respuesta, usando los métodos gráfico, algebráico y simplex.

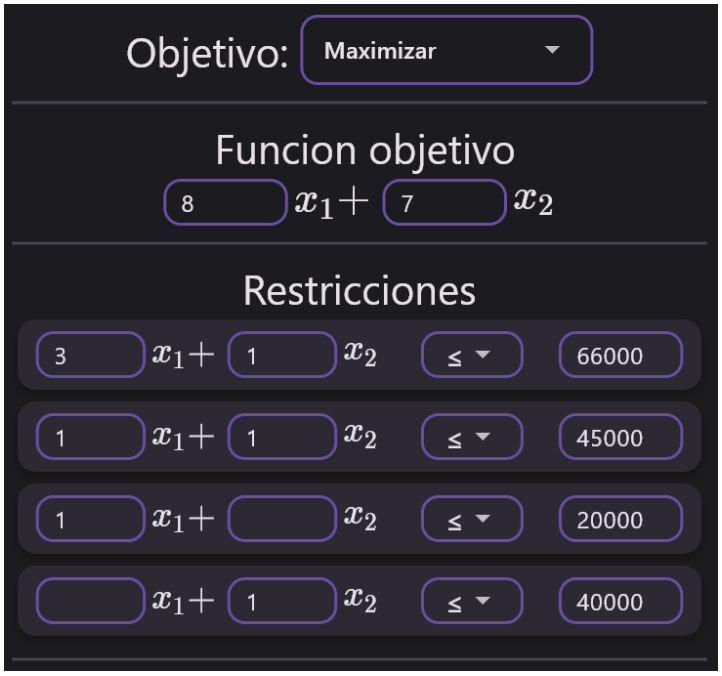


Figura 3.60: Datos del problema de medicinas en la aplicación.

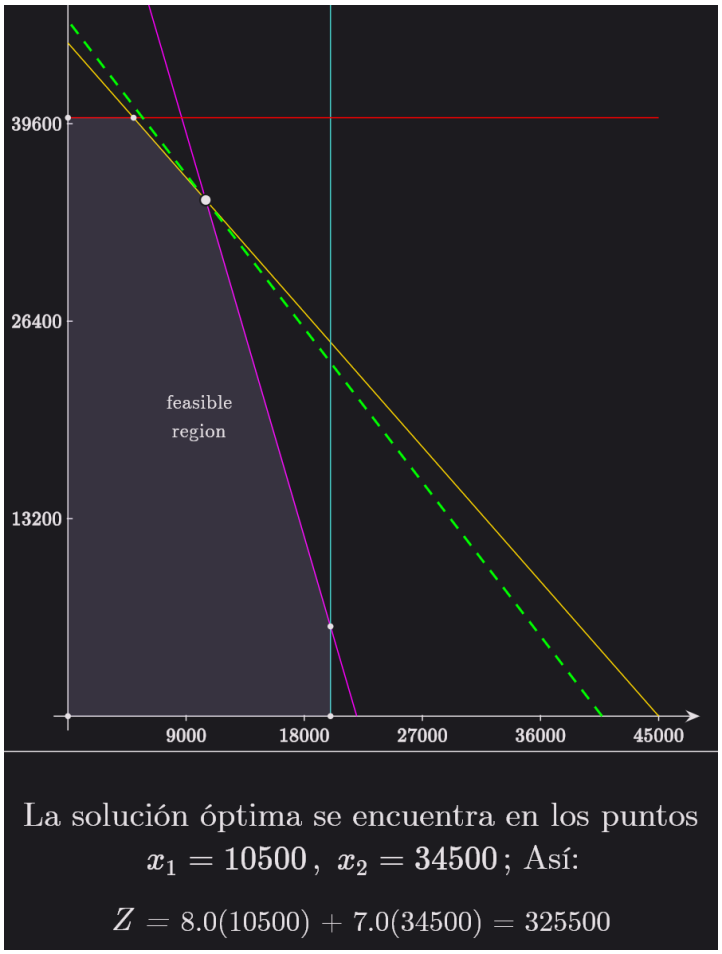


Figura 3.61: Resultados del problema de medicinas método gráfico.

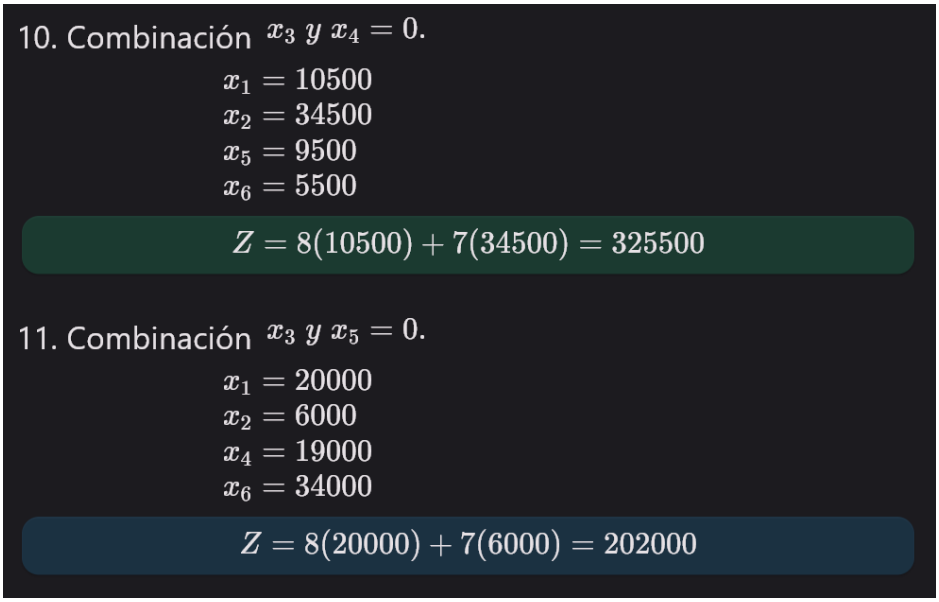


Figura 3.62: Resultados del problema de medicinas método algebráico, pasos 10-11.

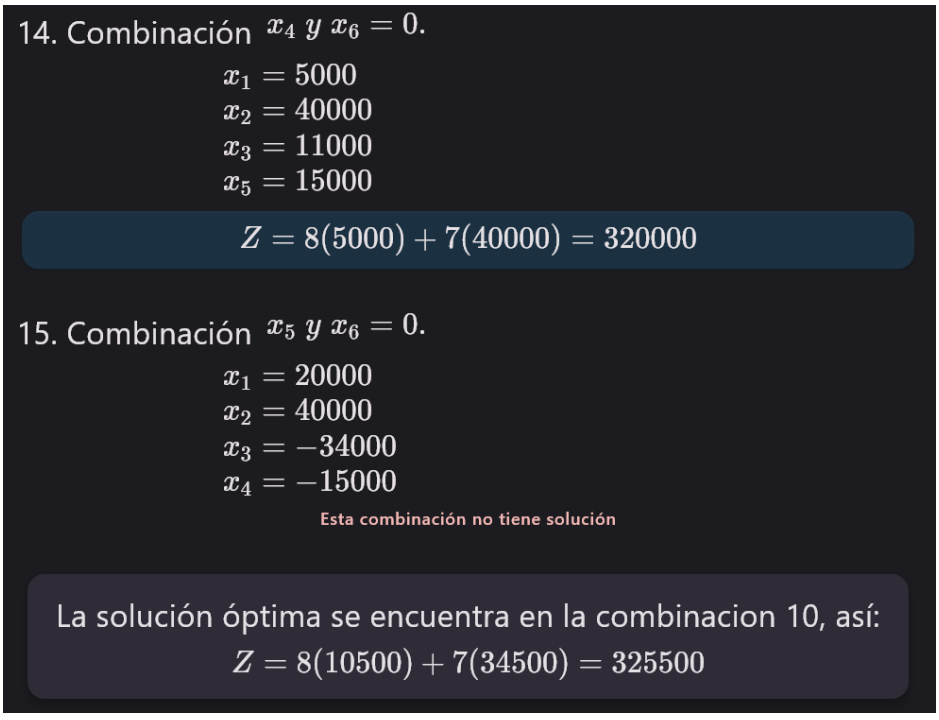


Figura 3.63: Resultados del problema de medicinas método algebráico, pasos 14-15.

Figura 3.64: Resultados del problema de medicinas método simplex.

Todos los métodos que soporta la aplicación llegan a la misma respuesta presentada en la figura 3.59, esto su corrobora con las figuras 3.61, 3.62, 3.63 y 3.64.

Hillier en su libro **Introducción a Investigación de Operaciones** [50], en el capítulo 3, página 57, expone el problema del horario de personal, cuyo modelo de programación linear completo, se muestra en la figura 3.65.

Minimize $Z = 170x_1 + 160x_2 + 175x_3 + 180x_4 + 195x_5$,

subject to

x_1		≥ 48	(6–8 A.M.)
$x_1 + x_2$		≥ 79	(8–10 A.M.)
$x_1 + x_2$		≥ 65	(10 A.M. to noon)
$x_1 + x_2 + x_3$		≥ 87	(Noon–2 P.M.)
$x_2 + x_3$		≥ 64	(2–4 P.M.)
$x_3 + x_4$		≥ 73	(4–6 P.M.)
$x_3 + x_4$		≥ 82	(6–8 P.M.)
x_4		≥ 43	(8–10 P.M.)
$x_4 + x_5$		≥ 52	(10 P.M.–midnight)
x_5		≥ 15	(Midnight–6 A.M.)

and

$x_j \geq 0$, for $j = 1, 2, 3, 4, 5$.

Figura 3.65: Modelo de problema horario de personal [50].

En el libro indica que la solución óptima para este problema es:

$$Z = 30610 \tag{3.66}$$

A continuación, se muestra la entrada de datos en la aplicación 3.66 para resolver este problema y posterior a ello se muestra la última iteración de la segunda fase del método simplex 3.67, la cual demuestra que la aplicación llega a la misma respuesta presentada en la ecuación 3.66.

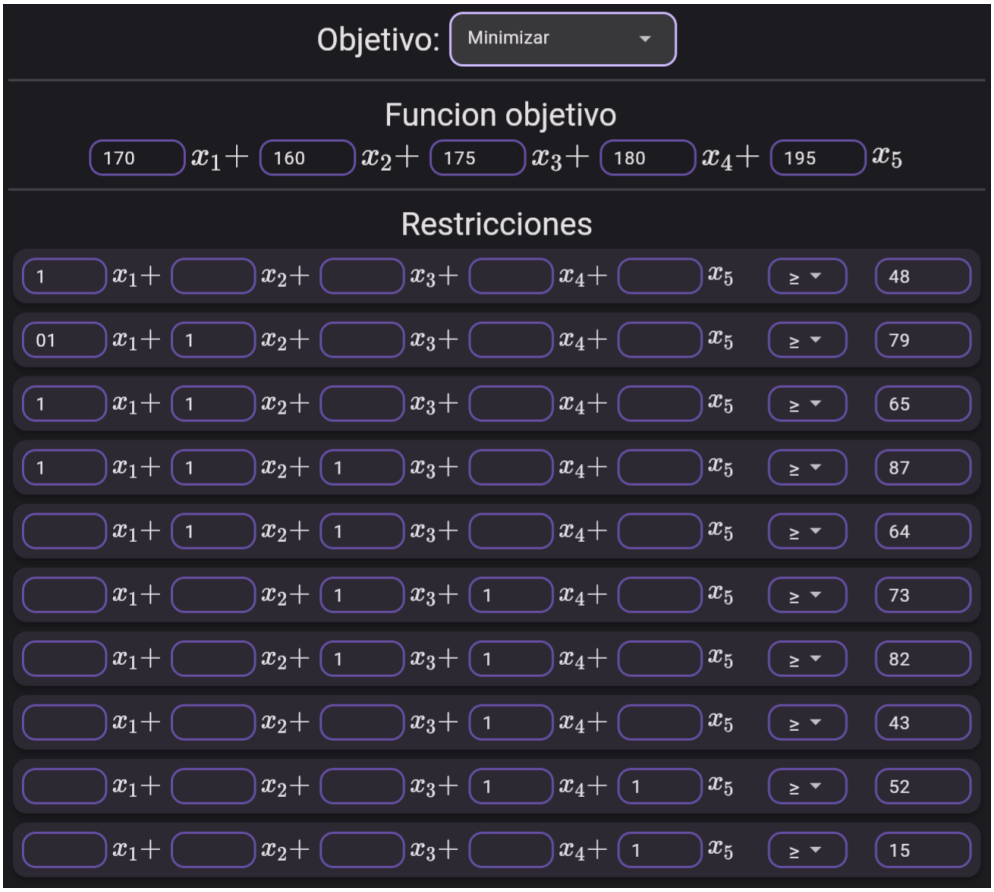


Figura 3.66: Datos del problema de personal en la aplicación.

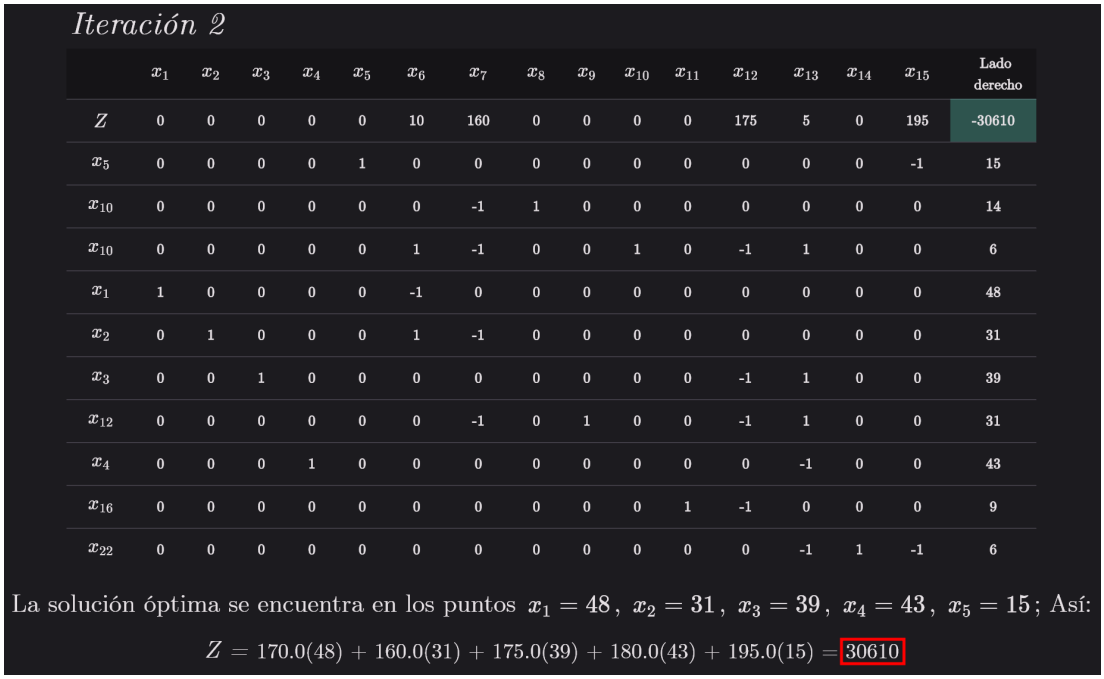


Figura 3.67: Resultados del problema de personal en la aplicación.

Taha en su libro **Introducción a Investigación de Operaciones** [46], en el capítulo 3, página 129, expone el problema de ensamble de vehiculos, cuyo modelo de programación linear completo, se muestra en la figura 3.68.

$$\text{Maximinze } z = 3x_1 + 2x_2 + 5x_3$$

subject to

$$\begin{aligned} x_1 + 2x_2 + x_3 &\leq 430 \text{ (Operation 1)} \\ 3x_1 + 2x_3 &\leq 460 \text{ (Operation 2)} \\ x_1 + 4x_2 &\leq 420 \text{ (Operation 3)} \end{aligned}$$

$$x_1, x_2, x_3 \geq 0$$

Figura 3.68: Modelo de problema ensamble [46].

En el libro indica como llegar a la solución óptima usando el método simplex, y la respuesta se muestra en la figura 3.69, donde $Z = 1350$.

Basic	x_1	x_2	x_3	x_4	x_5	x_6	Solution
z	4	0	0	1	2	0	1350
x_2	$-\frac{1}{4}$	1	0	$\frac{1}{2}$	$-\frac{1}{4}$	0	100
x_3	$\frac{3}{2}$	0	1	0	$\frac{1}{2}$	0	230
x_6	2	0	0	−2	1	1	20

The solution recommends manufacturing 100 trucks and 230 cars but no trains. The associated revenue is $1350.

Figura 3.69: Respuesta simplex de problema ensamble [46].

A continuación, se muestra la entrada de datos en la aplicación 3.70 para resolver este problema y posterior a ello se muestran las iteraciones del método simplex 3.71, la cual demuestra que la aplicación llega a la misma respuesta presentada en la figura 3.69.

Figura 3.70: Datos del problema de ensamble en la aplicación.

	x_1	x_2	x_3	x_4	x_5	x_6	Lado derecho
Z	-3	-2	-5	0	0	0	0
x_4	1	2	1	1	0	0	430
x_5	3	0	2	0	1	0	460
x_6	1	4	0	0	0	1	420

	x_1	x_2	x_3	x_4	x_5	x_6	Lado derecho
Z	4.5	-2	0	0	2.5	0	1150
x_4	-0.5	2	0	1	-0.5	0	200
x_3	1.5	0	1	0	0.5	0	230
x_6	1	4	0	0	0	1	420

	x_1	x_2	x_3	x_4	x_5	x_6	Lado derecho
Z	4	0	0	1	2	0	1350
x_2	-0.25	1	0	0.5	-0.25	0	100
x_3	1.5	0	1	0	0.5	0	230
x_6	2	0	0	-2	1	1	20

Figura 3.71: Resultados del problema de ensamble con método simplex.

3.4.1 Validación del rendimiento.

Para valorar la rapidez con la que la aplicación puede resolver problemas de programación lineal, se realizó una comparación de los tiempos de ejecución, para resolver problemas de diferentes dimensiones y datos. Para tener un punto de referencia, estos ejercicios se resuelven además usando PhpSimplex.

Los resultados de PhpSimplex se toman de la pestaña de "Red" en las herramientas de desarrollador de un navegador basado en Chromium, y se toma el tiempo de la solicitud de la página, que es el tiempo que tarda en resolver el problema.

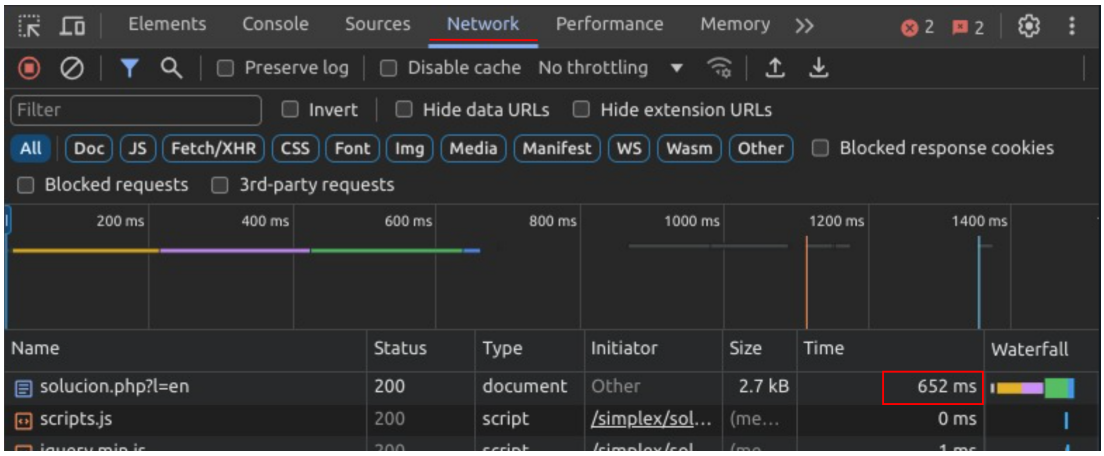


Figura 3.72: Tiempo de ejecución de PhpSimplex.

Los resultados de la aplicación Flutter se toman de la consola de depuración de Flutter. Para estas pruebas se compila la aplicación en modo de perfilado. Para una mayor relevancia de la comparacion de resultados, se ejecuta la version web de la aplicación en un navegador basado en Chromium.

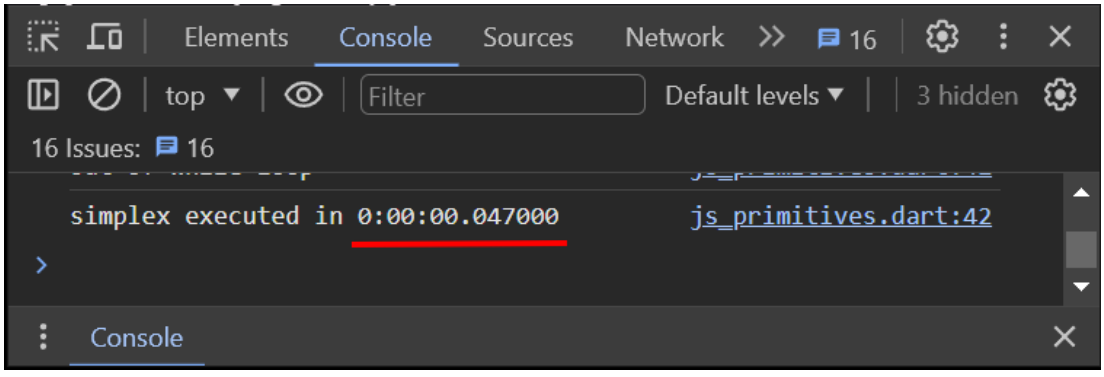


Figura 3.73: Tiempo de ejecución por consola.

A continuación, se muestra una tabla con los resultados obtenidos.

	PHPSimplex	**LinearProgrammingApp**
	629	52.5
	634	56.6
	687	53.3
	684	59.19
	675	48.9
	598	48.7
	597	57.6
	605	48.4
	655	57.6
	689	47
$\bar{x}$	645.3	52.9

Tabla 3.10: Tiempo promedio de ejecución para obtener respuesta.

Los datos presentados en la tabla muestran el tiempo promedio de ejecución para obtener respuesta en los sistemas, PHPSimplex y LinearProgrammingApp. Se observa que, en promedio, el sistema PHPSimplex tiene un tiempo de ejecución mayor en comparación con la aplicación desarrollada.

CAPITULO IV.- CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones.

La investigación y análisis de bibliografía sobre los algoritmos de programación lineal presenta de forma clara sus aplicaciones en diversos escenarios. Se demostró que el algoritmo gráfico es una herramienta visual efectiva para problemas de dos variables, mientras que el algoritmo algebraico proporciona una solución más generalizada para problemas sin mayor complejidad. El algoritmo simplex es altamente eficiente y versátil en la resolución de problemas de cualquier escala.

El framework Flutter se presenta como una alternativa viable para el desarrollo de aplicaciones multiplataforma con un enfoque en el alto rendimiento, habiendo tomado la ruta de compilación a código máquina para cada una de las plataformas que soporta. A esto se suma un rápido ciclo de desarrollo, que permite iterar velozmente, la creación de prototipos y la experimentar nuevas funcionalidades. Este framework posibilita para crear interfaces de usuario consistentes en diferentes plataformas con facilidad.

La adopción de la metodología Kanban en el desarrollo de este sistema multiplataforma ha demostrado ser altamente beneficiosa, gracias a su capacidad para facilitar un seguimiento claro y visual de las tareas, así como para brindar una supervisión más eficiente del proyecto en su totalidad. Esta experiencia positiva resalta la eficacia y transparencia inherentes a Kanban, lo que sugiere su aplicabilidad en proyectos futuros de desarrollo de software, tanto en entornos colaborativos como en contextos de trabajo individual.

La aplicación proporciona a los usuarios una solución versátil para abordar problemas de programación lineal mediante los métodos gráfico, algebraico y simplex. Disponible en múltiples plataformas: Android, Windows, GNULinux y Web, ofrece a los usuarios una herramienta accesible para la toma de decisiones fundamentada en modelos matemáticos. Se destaca por ser de código abierto, lo que permite su acceso y modificación por parte de la comunidad, y además, tiene la capacidad de resolver ejercicios sin necesidad de conexión a internet y sin costo alguno.

4.2 Recomendaciones.

Se sugiere que, en pos de enriquecer el alcance del proyecto, se considere la exploración de las demás áreas de investigación en el campo de la Programación Lineal. En consecuencia, se podría explorar la implementación de otros algoritmos relevantes en la Programación Lineal, como el algoritmo del transporte. Este algoritmo es importante para la resolución de problemas de asignación y distribución de recursos. Su inclusión, en futuras etapas del desarrollo, podría contribuir significativamente a la utilidad del proyecto.

La actualización de las dependencias, y el framework mismo a nuevas versiones, supone la mejora de la estabilidad, rendimiento y seguridad, por lo que se recomienda que se vaya actualizando la aplicación a medida que se vayan liberando nuevas versiones, especialmente en el caso del framework Flutter, que se encuentra en constante desarrollo y promete mejoras significativas, con la inclusión de soporte mejorado para WASM y "shaders" personalizados.

Se recomienda considerar la implementación de la metodología de desarrollo Kanban dada la naturaleza clara y visual que facilita el seguimiento de tareas y el progreso del proyecto, tanto para equipos de trabajo como para desarrolladores individuales. Por lo tanto, se alienta a explorar y adoptar activamente esta metodología en futuros proyectos para maximizar la eficiencia y el éxito del desarrollo de software.

Se recomienda, aprovechar la naturaleza de código abierto del proyecto para fomentar la colaboración y la participación de la comunidad de desarrolladores y expertos en Investigación de Operaciones. Al abrir el proyecto a contribuciones externas, retroalimentación y aportes, se posibilita una mejora continua.

Además, se invita a que se incluya la aplicación como recurso didáctico en clases, talleres prácticos y sesiones de capacitación en aulas y laboratorios de la FISEI, para temas relacionades a la Programación Lineal de Investigación Operativa. Se puede incluso incorporar como una herramienta para el aprendizaje y la resolución de problemas, aplicando sus funcionalidades en contextos académicos pertinentes dentro de la Universidad Técnica de Ambato.

Referencias Bibliográficas.

[1] M. Cusumano, A. MacCormack, C. Kemerer, and B. Crandall, "Ieee software," *Software development worldwide: the state of the practice.*, vol. 20, pp. 28–34, 2003.

[2] D. J. W., *The reliability of software*. Commun ACM, 1965.

[3] D. R. A., L. R. J, and F. G. Sayward, *Hints on test data selection: Help for the practicing programmer*. IEEE Comput., 1978.

[4] P. Sandborn, "Software obsolescence – complicating the part and technology obsolescence management problem," *IEEE Trans on Components and Packaging Technologies*, vol. 30, pp. 886–888, 2007.

[5] M. Harman, "Engineering, the current state and future of search based software," *Future of Software Engineering (FOSE '07)*, 2007.

[6] F. Ndayiragije, "The limitations of tora software in solving linear programming problems: Case of the unrevised simplex method," *International Journal of Science and Engineering Investigations*, vol. 6, pp. 49–51, 2017 2017.

[7] M. A. Espinosa Carvajal, "Aplicación web utilizando el modelo de programación lineal para la optimización de dietas alimenticias nutricionales," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial, 2020.

[8] S. P. Sánchez Sánchez, "Control de inventarios mediante programación lineal en la empresa la fortaleza cia. ltda.," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial, 2015.

[9] M. B. Heredia Guzmán, "Modelo de programación lineal entera para la generación de horarios de clase en la universidad," B.S. thesis, Quito, 2014., 2014.

[10] A. F. Artieda Cadena, "Optimización de tarifas de la energía eléctrica para una respuesta a la demanda por medio de programación lineal," B.S. thesis, Universidad Politécnica Salesiana. Carrera de Ingeniería Eléctrica. Sede Quito, 2017.

[11] J. E. Núñez Ribadeneira, D. F. López Jiménez, F. L. Villarreal Satama, D. I. Montenegro Gálvez, H. P. Márquez Fuentes, and S. A. Ullauri Betancourt, "Programación lineal en la asignación de planeación de mano de obra de entidades bancarias," *Ciencia Latina-Revista Multidisciplinar*, 2021.

[12] D. del Río Gómez *et al.*, "Un método símplex en programación lineal multiobjetivo," *UVaDOC*, 2021.

[13] M. B. B. Loranca, L. A. C. Cruz, R. G. Velázquez, and G. M. Guzmán, "Análisis de sensibilidad mediante software en programación lineal," *INNOVACIONES TECNOLÓGICAS EN LAS CIENCIAS COMPUTACIONALES*, p. 128, 2019.

[14] C. E. Flores-Tapia, K. L. Flores Cevallos, *et al.*, "Método simplex de programación lineal aplicado a una empresa distribuidora de mobiliario," *Revista entorno*, 2021.

[15] K. O. Oluwaseyi, A. Elizabeth, and O. E. Olaoluwa, "Profit maximization in a product mix bakery using linear programming technique," *Journal of investment and Management*, vol. 9, no. 1, pp. 27–30, 2020.

[16] L. Alegsa, "Definición de multiplataforma (informática)," 2018.

[17] O. S. Initiative, "Open standards requirements for software – rationale," Sep 2022.

[18] O. S. Initiative, "The open source definition," Sep 2022.

[19] F. E. Team, "How to choose the right open source license," Oct 2022.

[20] R. M. Gómez Labrador, "Tipos de licencias de software," *SOLFA-US*, p. 6, 2005.

[21] J. Choo, "The art of flutter: Hello world," Dec 2019.

[22] Flutter, "Flutter | multi-platform."

[23] Flutter, “Flutter | docs.”

[24] R. Amadeo, “Google’s dart language on android aims for java-free, 120 fps apps,” *Ars Technica*, May 2015.

[25] V. Diví, “What is the dart programming language?,” Oct 2020.

[26] D. Team, “Dart faq,” Jun 2019.

[27] AppWrite, “About us - appwrite.” [Accessed: Jun. 28, 2023.]. Available: `https://appwrite.io/company/about`, Jun 2023.

[28] AppWrite, “Appwrite - open-source end-to-end backend server.” [Accessed: May. 23, 2023.]. Available: `https://appwrite.io/`, May 2023.

[29] AppWrite, “Docs - appwrite.” [Accessed: Jun. 28, 2023.]. Available: `https://appwrite.io/docs`, Jun 2023.

[30] P. Patel, “Introduction to google firebase,” Sep 2019.

[31] J. T. Bell, “Extreme programming,” 2017.

[32] K. Beck, *Extreme programming explained: embrace change*. Addison Wesley Professional, 2 ed., 2004.

[33] A. Alliance, “What is extreme programming (xp)?.”

[34] D. J. Wells, “Extreme programming.” Available online at `http://www.extremeprogramming.org/map/project.html`, 2000.

[35] “What is scrum methodology? & scrum project management,” Mar 2023.

[36] T. Hirotaka and N. Ikujiro, “The new product development game,” Aug 2014.

[37] Scrum.org, “What is scrum?.”

[38] M. Gallego, A. Trigas, and C. Domingo, “Metodología scrum. gestión de proyectos informáticos,” 2012.

[39] K. Schwaber and J. Sutherland, “The scrum guide,” *Scrum Alliance*, vol. 21, 2011.

[40] N. Kirovska and S. Koceski, "Usage of kanban methodology at software development teams," *Journal of applied economics and business*, vol. 3, no. 3, pp. 25–34, 2015.

[41] P. M. Inc., "What is kanban?." [Accessed: May. 29, 2023.]. Available: `https://www.projectmanager.com/guides/kanban`, 2023.

[42] D. RADIGAN, "Kanban - a brief introduction." [Accessed: May. 23, 2023.]. Available: `www.atlassian.com/agile/kanban`, 2023.

[43] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical study of agile software development methodologies: A comparative analysis," *SIGSOFT Softw. Eng. Notes*, vol. 40, pp. 1–6, feb 2015.

[44] D. R. K. Gupta, *Introduction to Operations Research*. Krishna Prakashan Media, 1992.

[45] M. W. Carter, C. C. Price, and G. Rabadi, *Operations Research: A Practical Introduction*. Boca Raton: CRC Press, 2nd edition ed., 2001.

[46] H. A. Taha, *Operations Research: An Introduction*, vol. 7. Pearson Education, tenth ed., 2017.

[47] P. M. Morse, G. E. Kimball, and S. I. Gass, *Methods of Operations Research*. Dover Publications, Inc., 2012.

[48] V. Chvatal, V. Chvatal, *et al.*, *Linear programming*. Macmillan, 17 ed., 1983.

[49] A. Romero Medina, "Organización de los documentos conseguidos." [Accessed: Feb. 08, 2024.]. Available: `https://www.um.es/docencia/agustinr/docum/docum3.htm`, 2010.

[50] F. S. Hillier, *Introduction to Operations Research*. McGraw-Hill Education, 10th ed., 2015.

[51] M. Carter, C. C. Price, and G. Rabadi, *Operations research: a practical introduction*. CRC Press, 2018.

[52] J. Stacho, *Introducction to operation research, deterministic models*. Department of Industrial Engineering and Operations Research, 2014.

[53] W. L. Winston, *Operations research: applications and algorithms*. Thomson Learning, Inc., 2004.

[54] R. S. Garrido and A. M. M, *Programación Lineal, Metodología Y Problemas*. Editorial Tébar Flores, 1993.

[55] H. G. Salas, *Programación lineal aplicada*. Ecoe Ediciones, 3 ed., 2022.

[56] B. S. Moktar, J. J. J., and S. H. D., *Linear Programming and Network Flows*. Wiley, 2010.

[57] T. Tran, "Flutter native performance and expressive ui/ux," *Metropolia University of Applied Sciences*, p. 34, April 2020.

[58] A. Fitz Gibbon and M. Sullivan, "How flutter renders widgets." Accessed: Apr. 18, 2023. [Online video]. Available: `https://youtu.be/996ZgFRENMs`, Nov 2019.

[59] A. Vichare, "Flutter widgets lifecycle, widget tree and element tree," Apr 2021.

[60] E. Windmill, *Flutter in action*. Simon and Schuster, 2020.

[61] Flutter, "Inside flutter." [Accessed: Apr. 18, 2023.]. Available: `https://docs.flutter.dev/resources/inside-flutter`, Nov 2021.

[62] Flutter, "ntroduction to widgets." [Accessed: Apr. 29, 2023.]. Available: `https://docs.flutter.dev/ui/widgets-intro`, Apr 2023.

[63] Flutter, "Widget class." [Accessed: Apr. 24, 2023.]. Available: `https://api.flutter.dev/flutter/widgets/Widget-class.html`, Oct 2022.

[64] M. L. Napoli, *Beginning Flutter: A hands on guide to app development*. John Wiley & Sons, 2020.

[65] Flutter, "Widget class." [Accessed: Apr. 29, 2023.]. Available: `https://docs.flutter.dev/tools/hot-reload`, Apr 2023.

[66] Flutter, “Devtools.” [Accessed: May. 05, 2023.]. Available: https://docs.flutter.dev/tools/devtools, Apr. 2023.

[67] Flutter, “Supported deployment platforms.” [Accessed: May. 05, 2023.]. Available: https://docs.flutter.dev/reference/supported-platforms, Mar 2023.

[68] Dart, “Dart overview.” [Accessed: May. 05, 2023.]. Available: https://dart.dev/overview, Mar 2023.

[69] M. Sillivan, “Flutter: Don’t fear the garbage collector.” [Accessed: May. 06, 2023.]. Available: https://medium.com/flutter/flutter-dont-fear-the-garbage-collector-d69b3ff1ca30, Jan. 2019.

[70] K. Chisholm, “What’s new in flutter 3.10.” [Accessed: May. 12, 2023.]. Available: https://medium.com/flutter/whats-new-in-flutter-3-10-b21db2c38c73, May 2023.

[71] fluttercompletereference.com, “equations.” [Accessed: Jan. 05, 2024.]. Available: https://pub.dev/packages/equations, Sep 2023.

[72] Flutter, “Custompainter class.” [Accessed: Jan. 05, 2024.]. Available: https://api.flutter.dev/flutter/rendering/CustomPainter-class.html, Dec 2023.

[73] kremer.software, “msix.” [Accessed: Jan. 12, 2024.]. Available: https://pub.dev/packages/msix, Nov 2023.

[74] R. Gupta, *Operations research*. Krishna Prakashan, 1992.

Anexos.

Acrónimos

AOT Antes de tiempo. 63

API Interfaz de programación de aplicaciones. 23, 66

CPF Punto de esquina factible (*del inglés:* Corner Point Feasible). 40

CPU Unidad de Procesamiento Central. 66, 67

HTTP Hypertext Transfer Protocol. 68

HTTPS Hypertext Transfer Protocol Secure. 24, 68

IDE Entorno de desarrollo integrado. 64

JIT Justo a tiempo. 28, 63

PB Product Backlog. 28

RAM Memoria de Acceso Aleatorio. 67

SLSA Niveles de Cadena de Suministro para Artefactos de Software. 69

SSL Secure Sockets Layer. 24

VM Máquina Virtual de Dart. 63, 66

XP Extreme Programming. 25, 69, 70

Manual de usuario

Contenidos

1. Introducción

Bienvenido al manual de usuario de esta aplicación, que permite encontrar la solución a problemas de programación lineal mediante los métodos gráfico, algebraico y simplex. Este manual proporciona una guía detallada por secciones, con puntos de referencia numerados. Estos ayudan al usuario a comprender cómo utilizar cada funcionalidad de la aplicación, indicando cada uno de los controles en cada pantalla de manera ordenada. Este manual contempla temas que van desde el inicio de sesión, carga de datos y obtención de soluciones a los problemas ingresados.

2. Inicio de sesión

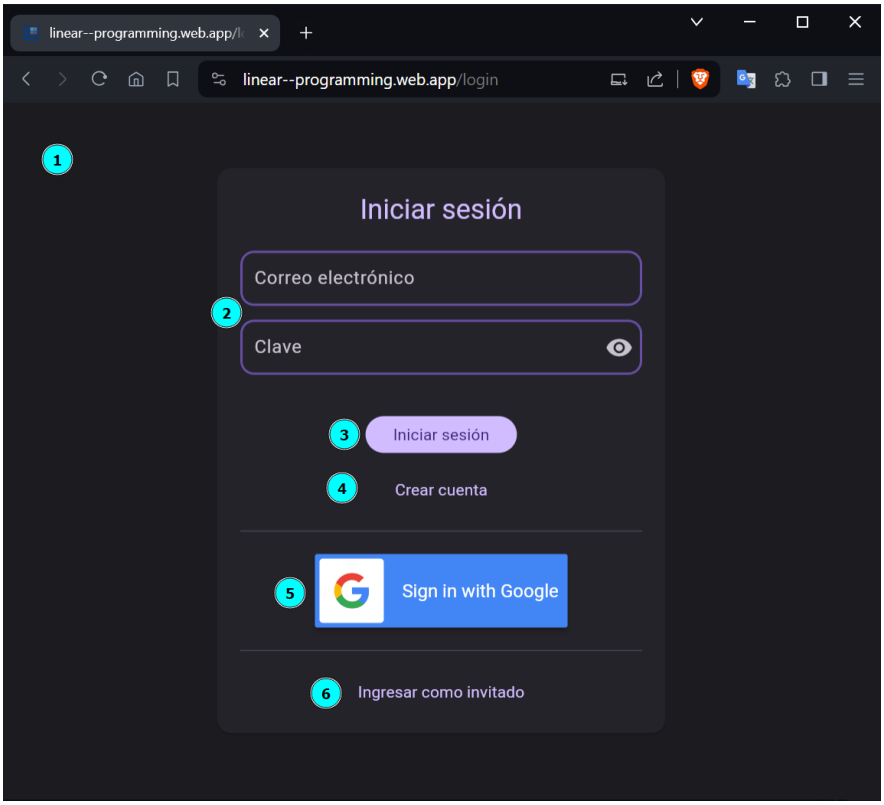


Figura 4.1: Inicio de sesión

1. Pantalla de inicio, donde el usuario tiene varias opciones de inicio de sesión o registro de cuenta de usuario.
2. Ingreso de un usuario registrado, con autenticación por correo electrónico y clave.
3. Botón para iniciar sesión.
4. Botón para registro de un nuevo usuario 4.2.
5. Botón para autenticación de usuario, con cuenta de Google.
6. Botón que permite al usuario, ingresar al sistema sin necesidad de registro.

3. Registro de usuario

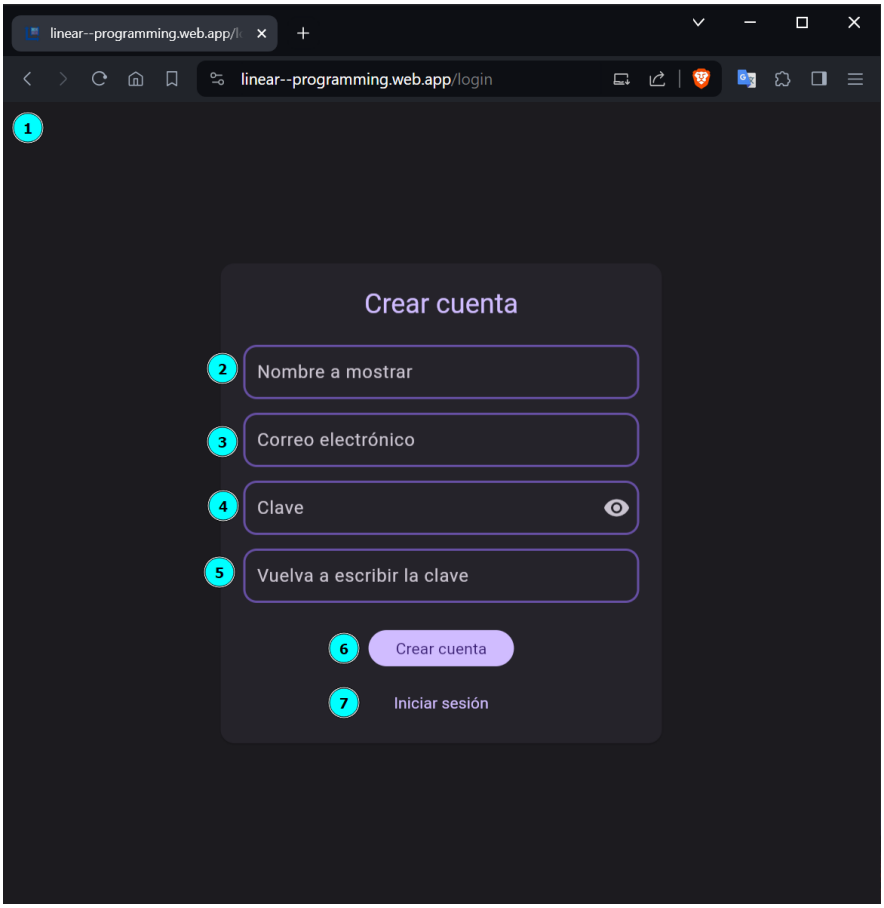


Figura 4.2: Manual de registro

1. Pantalla de creación de cuenta, con los controles para proveer información de inicio de sesión.

2. Nombre del usuario, que se mostrara cuando inicie sesión.

3. Correo electrónico del usuario, que debe ser único entre los usuarios registrados.

4. Clave que el usuario desea usar, con la opción de desvelar o no los caracteres ingresados.

5. Botón para registrar al usuario con los datos ingresados, y si el proceso es exitoso, automáticamente autentica al usuario con los mismos datos.

6. Botón que permite al usuario, regresar a la pantalla de inicio de sesión 4.1.

4. Pantalla principal

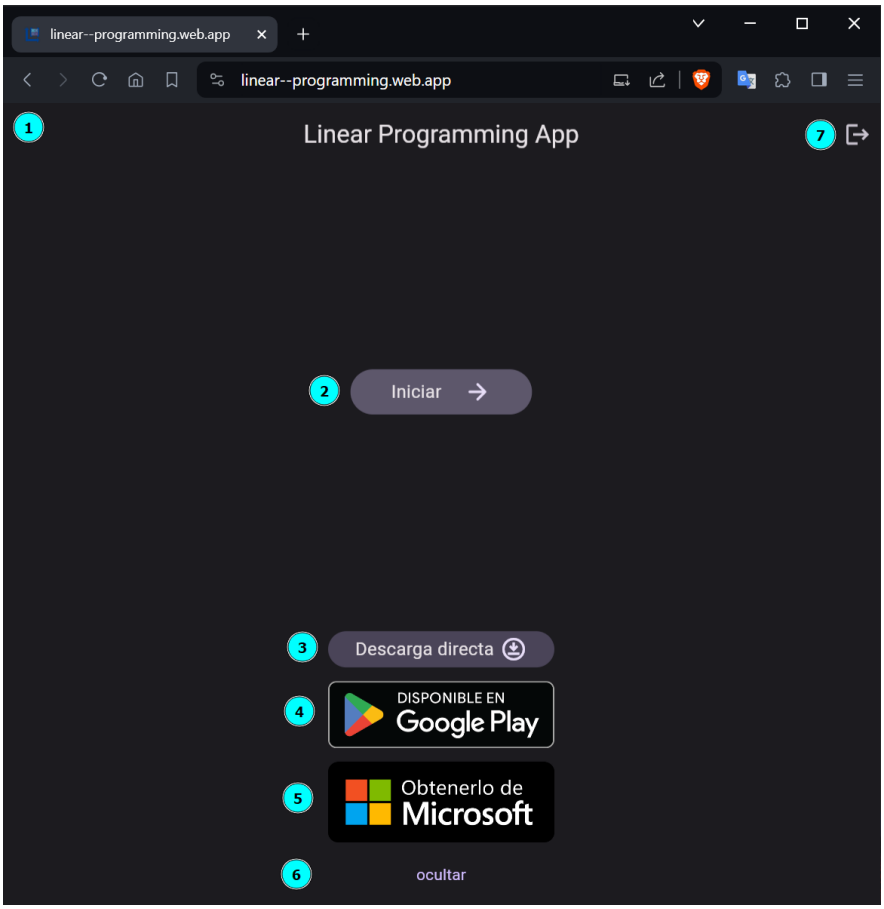


Figura 4.3: Pantalla principal

1. Pantalla principal donde se encuentra la opción para iniciar la solución de problemas, las opciones de descarga de la aplicación y la opción de cierre de sesión.
2. Botón para iniciar el ingreso de datos del problema.
3. Botón de descarga directa que proporciona los instaladores para las distintas plataformas disponibles.
4. Botón para descargar la aplicación desde la tienda de Android.
5. Botón para descargar la aplicación desde la tienda de Microsoft.
6. Botón para ocultar las opciones de descarga.
7. Botón para cerrar la sesión en curso.

5. Pantalla ingreso de dimensión del problema

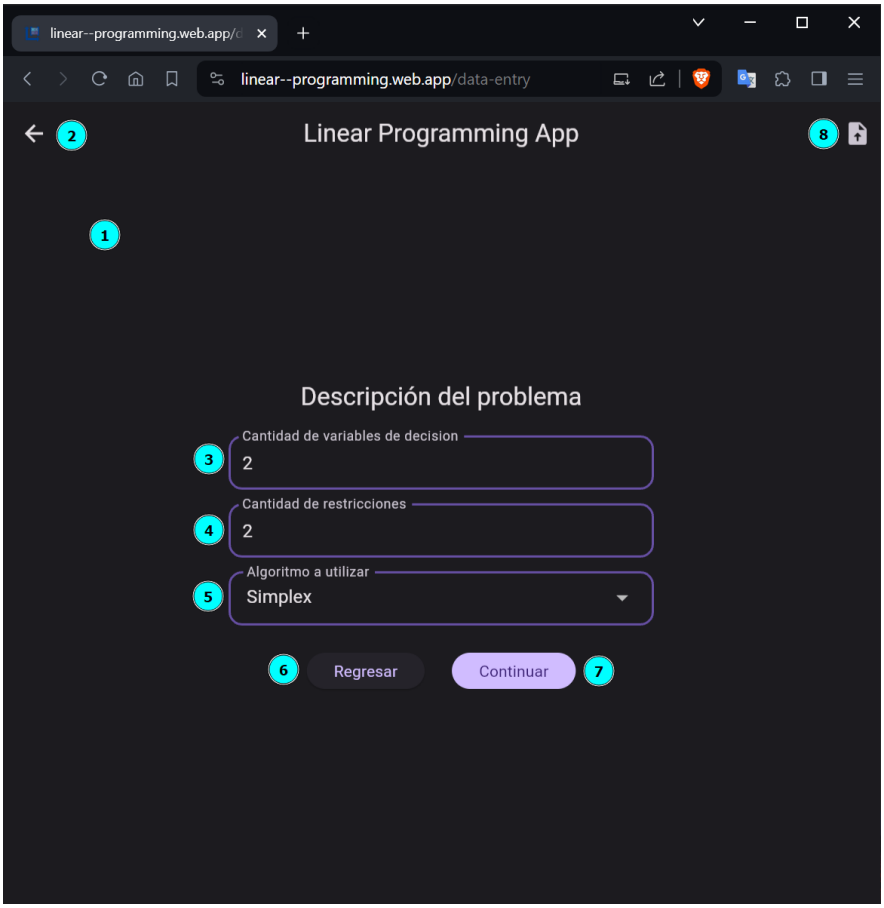


Figura 4.4: Pantalla ingreso de dimensión del problema

1. Pantalla donde se encuentran los campos correspondientes para el ingreso de la dimensión del problema a resolver, la carga externa de la dimensión y las opciones para continuar con el proceso o regresar a la pantalla anterior.

2. Botón para navegar a la pantalla principal 4.3.

3. Campo para el ingreso de la cantidad de variables de decisión.

4. Campo para el ingreso de la cantidad de restricciones.

5. Campo de tipo selección para establecer el algoritmo a utilizar cuando el numero de variables es igual a dos.

6. Botón para navegar a la pantalla principal 4.3.

7. Botón para continuar con el ingreso de datos del problema.

8. Botón para cargar las dimensiones desde un archivo externo.

6. Pantalla ingreso de datos del problema

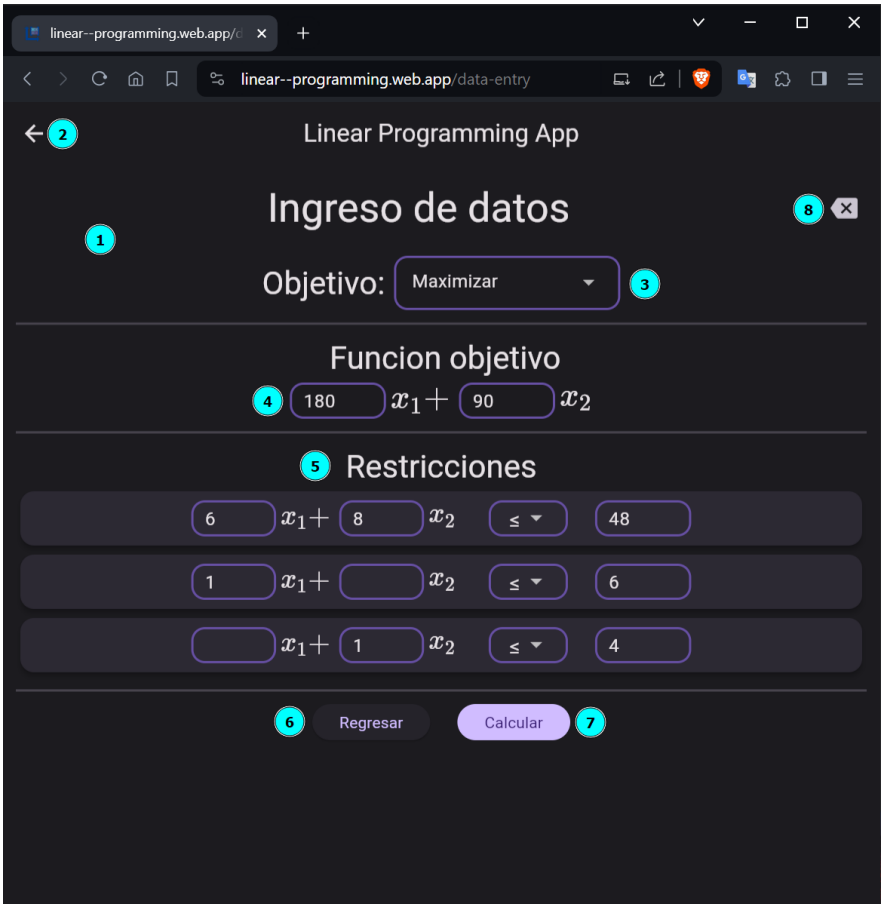


Figura 4.5: Pantalla ingreso de datos del problema

1. Pantalla donde se encuentran los campos correspondientes para indicar la función objetivo, los datos de la función objetivo, las variables, las opciones para navegar a la pantalla previa y la opción para calcular el ejercicio.

2. Botón para navegar a la pantalla 4.4 de ingreso de dimensión del problema.

3. Campo de tipo selección para establecer el tipo de función objetivo.

4. Campos para ingresar los valores de la función objetivo.

5. campos para ingresar los valores de las restricciones.

6. Botón para navegar a la pantalla 4.4 de ingreso de dimensión del problema .

7. Botón para calcular el ejercicio.

7. Pantalla de solución con el método simplex

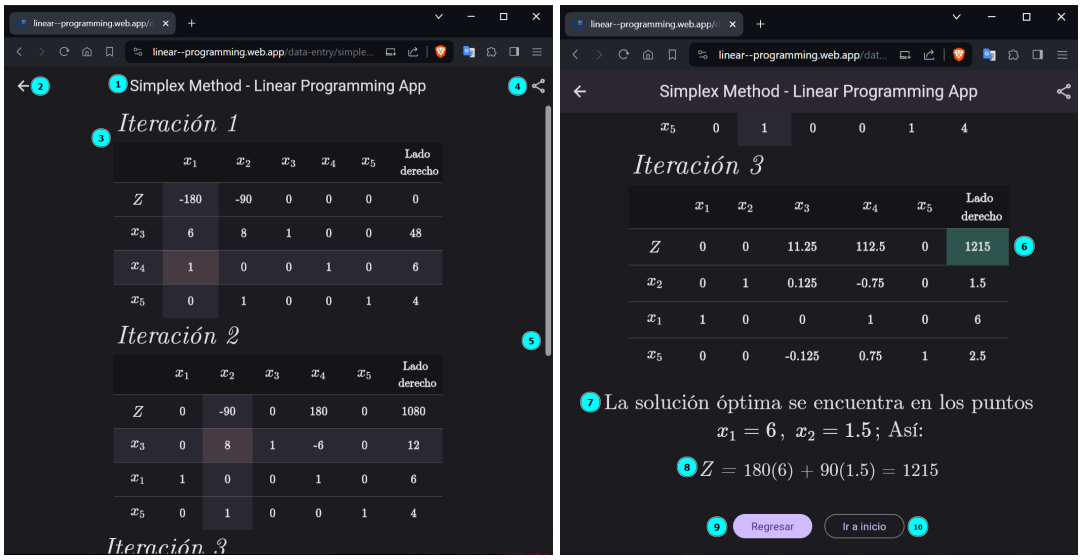


Figura 4.6: Pantalla de solución con el método simplex

1. Pantalla donde se muestra la solución encontrada mediante el método simplex.

2. Botón para navegar a la pantalla 4.5 de ingreso de datos.

3. Se muestra las iteraciones que el algoritmo sigue para dar con la solución.

4. Botón para compartir la solución encontrada.

5. Barra de desplazamiento que aparece automáticamente cuando el contenido de la solución desborda de la ventana.

6. Se muestra la solución como resultado de los cálculos de la iteración final.

7. Se muestran los valores que toman las variables para dar con la solución.

8. Solución en términos de la función objetivo.

9. Botón para navegar a la pantalla 4.5 de ingreso de datos.

10. Botón para navegar a la pantalla 4.3 principal.

8. Pantalla de solución con el método algebraico

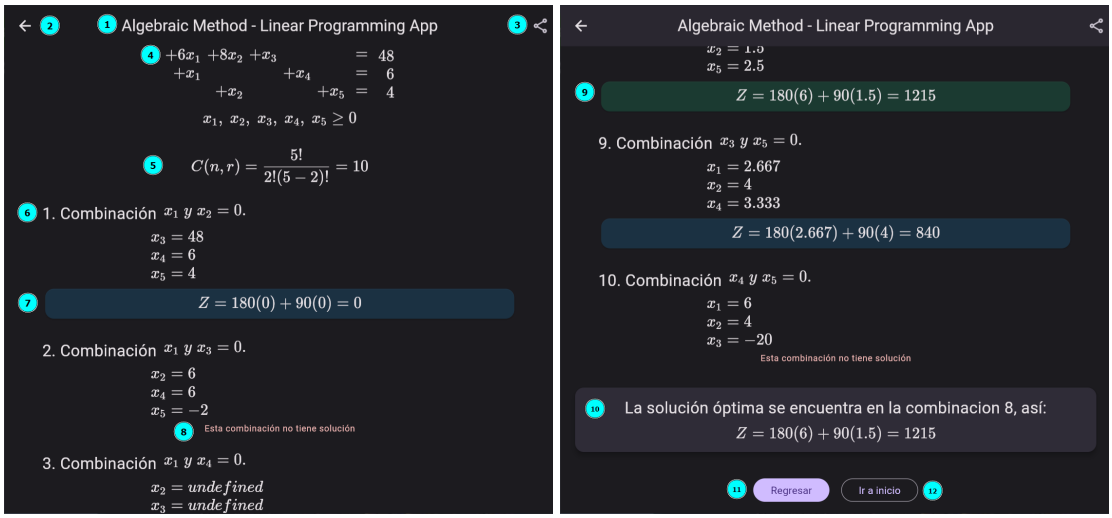


Figura 4.7: Pantalla de solución con el método algebraico

1. Pantalla donde se muestra la solución encontrada mediante el método algebraico, botones de navegación y botón para compartir la solución.
2. Botón para navegar a la pantalla 4.5 de ingreso de datos.
3. Botón para compartir los resultados
4. Forma estándar de las restricciones y condición de no negatividad de las variables.
5. Formula que se aplica para obtener la cantidad de combinaciones.
6. Combinaciones entre las cuales se halla la solución óptima.
7. Una de las soluciones factibles.
8. Combinación sin solución.
9. Solución óptima en linea con la combinación en la que aparece.
10. Se muestra la solución en términos de la función objetivo.
11. Botón para navegar a la pantalla 4.5 de ingreso de datos.
12. Botón para navegar a la pantalla 4.3 de inicio.

9. Pantalla de solución con el método gráfico

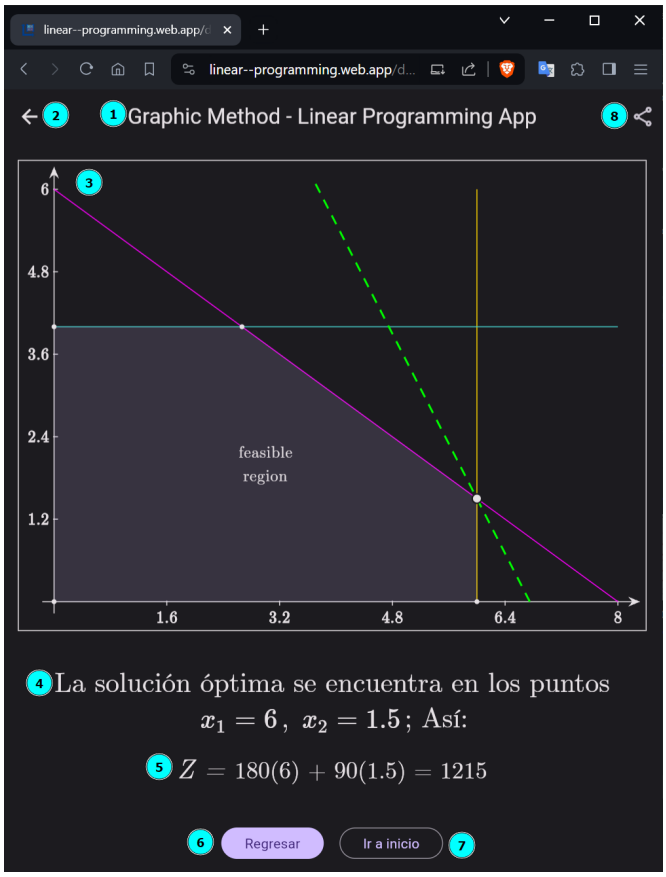


Figura 4.8: Pantalla de solución con el método gráfico

1. Pantalla donde se muestra la solución encontrada mediante el método algebraico, botones de navegación y botón para compartir la solución.
2. Botón para navegar a la pantalla 4.5 de ingreso de datos.
3. Canvas donde se dibuja la solución de forma gráfica, incluye las rectas de las variables, región factible, y puntos de las soluciones.
4. Se muestra los valores que toman las variables para dar con la solución.
5. Se muestra la solución en términos de la función objetivo.
6. Botón para navegar a la pantalla 4.5 de ingreso de datos.
7. Botón para navegar a la pantalla 4.3 de inicio.
8. Botón para compartir la solución encontrada.

10. Carga de datos

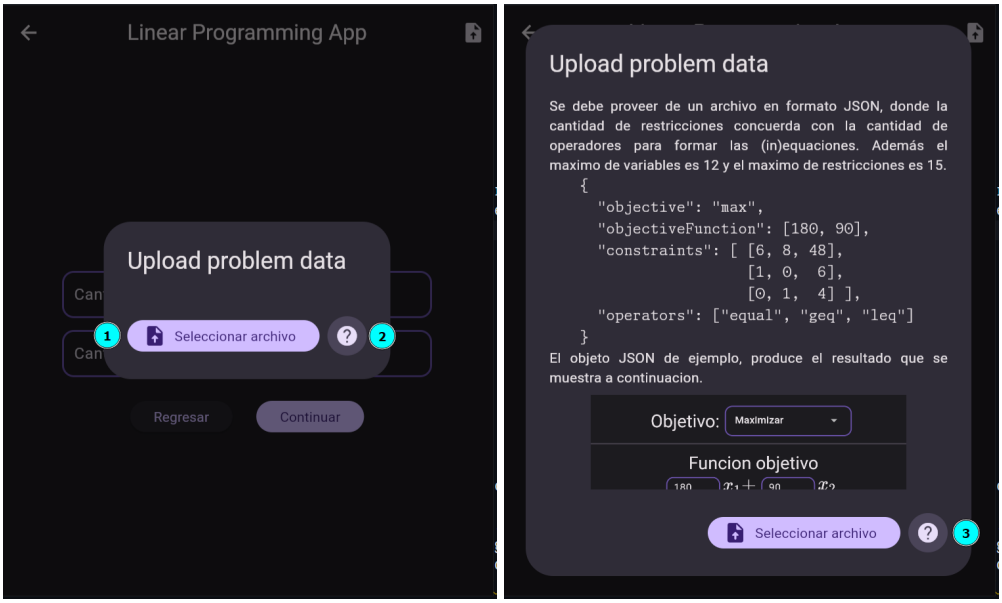


Figura 4.9: Carga de datos mediante archivo externo

1. Cargar de datos mediante un archivo externo de texto plano o formato JSON.

2. Botón para seleccionar un archivo.

3. Botón para desplegar la ayuda.

4. Botón para contraer la ayuda.