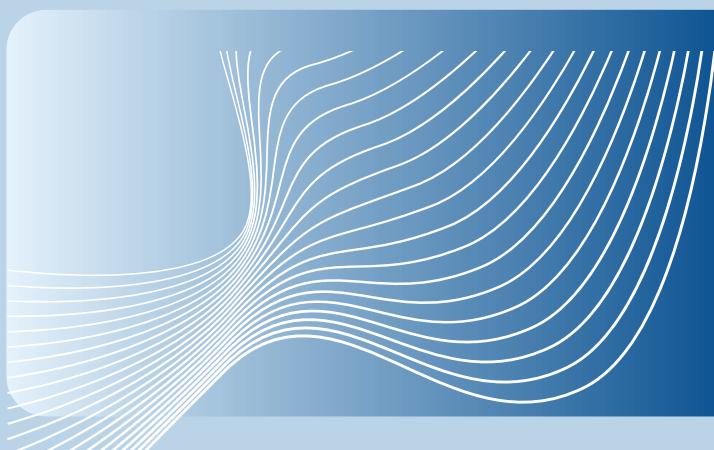


# Proyecto Swapply

Ingeniería del Software - Curso 2025/2026



Especificación de  
requisitos software



# Miembros del equipo

- ➡️ Colomba Andre
- ➡️ Carlos Belmonte Arias
- ➡️ Paula Bonilla Fernández
- ➡️ Carlota Camblor Parrondo
- ➡️ Ricardo Campo Acedo
- ➡️ Carla Cillán Barrajón
- ➡️ Samuel Fernández Gil
- ➡️ Elena Hidalgo Maqueda
- ➡️ Pablo Martín Gutiérrez
- ➡️ Sofía Mato de la Rocha
- ➡️ Diego Pallarés López



# ÍNDICE

## 1. Resumen ejecutivo

|                               |   |
|-------------------------------|---|
| 1.1. Descripción del proyecto | 5 |
| 1.2. Objetivos principales    | 5 |
| 1.3. Alcance general          | 6 |
| 1.4. Cronograma               | 7 |

## 2. Antecedentes y Justificación

|   |    |
|---|----|
| 2.1. Contexto del problema o necesidad              | 8  |
| 2.2. Situación actual                               | 9  |
| 2.3. Impacto esperado (económico, técnico o social) | 10 |

## 3. Objetivos del Proyecto

|                            |    |
|----------------------------|----|
| 3.1. Objetivo general      | 11 |
| 3.2. Objetivos específicos | 11 |

## 4. Alcance del Proyecto

|                                       |    |
|---------------------------------------|----|
| 4.1. Funcionalidades que se incluirán | 13 |
| 4.2. Límites del sistema              | 17 |
| 4.3. Entregables principales          | 19 |

## 5. Requerimientos del Sistema

|                                    |    |
|------------------------------------|----|
| 5.1. Requerimientos funcionales    | 21 |
| 5.2. Requerimientos no funcionales | 22 |
| 5.3. Requerimientos de usuario     | 24 |

## 6. Arquitectura y Diseño del Sistema

|  |    |
|--|----|
| 6.1. Diagrama general de arquitectura                          | 26 |
| 6.2. Definición de subsistemas (por cada subsistema):          | 27 |
| 6.2.1. Diagramas de Casos de Uso                               | 29 |
| 6.2.2. Casos de uso especificados                              | 31 |
| 6.2.3. Dos diagramas de actividad de los casos más importantes | 49 |
| 6.2.4. Prototipado de la herramienta por subsistema            | 51 |

---

## 7. Plan de Trabajo

|   |    |
|---|----|
| 7.1. Fases del proyecto: análisis, diseño, desarrollo, pruebas, implementación, mantenimiento | 56 |
| 7.2. Cronograma: Gantt  | 62 |
| 7.3. Dependencias y hitos clave   | 63 |

---

## 8. Recursos del Proyecto

|   |    |
|---|----|
| 8.1. Equipo de trabajo: roles y responsabilidades | 66 |
| 8.2. Recursos materiales y tecnológico            | 67 |
| 8.3. Presupuesto estimado                         | 72 |

---

## 9. Gestión del Proyecto

|  |    |
|--|----|
| 9.1. Metodología   | 75 |
| 9.2. Gestión de riesgos: identificación, análisis y plan de mitigación.                        | 77 |
| 9.3. Control de calidad: revisiones, pruebas, auditorías                                       | 78 |
| 9.4. Gestión de cambios: procedimiento para manejar modificaciones al alcance o requerimientos | 80 |
| 9.5. Comunicación: reuniones, reportes, herramientas   | 82 |

---

## 10. Plan de Pruebas

|  |    |
|--|----|
| 10.1. Tipos de pruebas                     | 84 |
| 10.2. Estrategia y criterios de validación | 86 |
| 10.3. Herramientas de testing              | 86 |

---



# 1. Introducción

## 1.1. Descripción del Proyecto

Swapply nace como una respuesta a las ineficiencias del mercado actual, posicionándose como una plataforma única que combina la eficiencia tecnológica del comercio electrónico moderno con la filosofía tradicional del trueque. La propuesta central del proyecto es fomentar la economía circular, promoviendo activamente la reutilización tanto de productos físicos como de habilidades personales. A través de este sistema, se busca mitigar el consumismo desmedido y ofrecer una herramienta digital que permita a los usuarios intercambiar valor de manera justa y sostenible.

## 1.2. Objetivos principales

Los objetivos clave de Swapply se definen en tres dimensiones interconectadas: Producto/Desarrollo, Innovación Tecnológica e Impacto Estratégico.

- **Objetivo de Producto y Desarrollo**

Desarrollar una plataforma fácil de usar, segura y estable que permita gestionar de forma autónoma todas las operaciones de trueque, desde que se publica una oferta hasta que el intercambio queda confirmado

- **Objetivo de Innovación Tecnológica**

Integrar de forma completa la moneda virtual SwapCoins. Esta herramienta permite superar las limitaciones del trueque directo, como la necesidad de que ambas partes quieran exactamente lo mismo, y proporciona una referencia de valor más justa para los bienes y servicios intercambiados.

- **Objetivo de Impacto y Estrategia**

Posicionar a Swapply como una herramienta digital de referencia en la región, capaz de ayudar a los estudiantes y las familias a ahorrar dinero y de fomentar una forma de consumo más responsable, colaborativa y consciente.

# 1.3. Alcance general



A continuación se describen de forma más clara y accesible las funcionalidades que se desarrollarán en esta fase del proyecto, así como aquellas que quedan fuera del alcance inicial.

## Funcionalidades incluidas

El sistema permitirá a los usuarios publicar y editar ofertas, tanto de productos físicos como de habilidades o servicios personales. Para demostrar la funcionalidad, se implementará un algoritmo de búsqueda y filtrado avanzado que generará matching entre las ofertas del usuario y los artículos de la base de datos de prueba.

Para facilitar los intercambios simulados, se integrará la moneda virtual SwapCoins. El sistema también contará con herramientas de interacción, como una simulación de chat y un sistema de valoraciones y reputación, diseñados para demostrar la lógica de la confianza dentro de la plataforma.

## Límites del sistema

En esta fase inicial, la plataforma no se encargará de la logística ni del transporte de los bienes intercambiados, ya que esta responsabilidad recaerá exclusivamente en los propios usuarios. Tampoco se incluirán pagos con dinero real ni pasarelas bancarias, por lo que no habrá monetización directa en esta etapa.

Por último, el desarrollo se centrará en una aplicación web adaptable a distintos dispositivos (responsive). La creación de aplicaciones nativas para iOS y Android se contempla para fases posteriores, una vez validado y escalado el proyecto.





## 1.4. Cronograma

La ejecución del proyecto está planificada en seis fases claras, utilizando la metodología ágil Scrum para garantizar la adaptabilidad.

| Fase           | Periodo          | Hito Clave Estratégico  |
|----------------|------------------|---|
| Análisis       | Diciembre        | Cierre del Documento de Requisitos y Casos de Uso (Hito H1).    |
| Diseño         | Enero            | Arquitectura y Prototipos de Interfaz aprobados (Hito H2).      |
| Desarrollo     | Febrero - Abril  | Entrega de la primera versión funcional para pruebas (Hito H3). |
| Pruebas        | Mayo             | Sistema Validado y Estable (Pruebas de Usabilidad y Carga).     |
| Implementación | Junio            | Lanzamiento Público (v1.0) de Swapply al Mercado (Hito H5).     |
| Mantenimiento  | Post-Lanzamiento | Monitorización del rendimiento y soporte continuo.              |

# 2. Antecedentes y Justificación



## 2.1. Contexto del problema y necesidad

El panorama económico actual está dominado por la "economía lineal" (un ciclo de producir, usar y tirar), lo que genera serios desafíos medioambientales y económicos que necesitan una solución digital inmediata.

- **Fomento del Consumo Excesivo:** El modelo actual incentiva la compra constante de bienes nuevos, impulsado por las modas y porque los productos (como la electrónica) están diseñados para dejar de funcionar o quedar anticuados rápidamente. Esto no solo agota los recursos naturales, sino que genera un aumento significativo en el gasto de los usuarios y en el volumen global de residuos.
- **Pérdida de Valor y Objetos Sin Uso:** Existe una gran cantidad de productos funcionales que pierden utilidad o quedan parados en los hogares, quedando almacenados. Esta acumulación representa un desperdicio enorme de recursos. Son bienes que podrían seguir siendo valiosos y útiles para otras personas,
- **La Necesidad del Estudiante:** Este problema es especialmente crítico para estudiantes y jóvenes profesionales que a menudo disponen de presupuestos limitados para adquirir bienes esenciales (libros, material electrónico, ropa). Para este grupo, intercambiar valor por objetos sin usar se convierte en una vía fundamental para solventar sus necesidades económicas sin endeudarse.

## 2.2. Situación actual



Ante el problema del consumo excesivo y el desperdicio de bienes con valor, la solución más eficiente es fomentar la Economía Circular a través del trueque. Sin embargo, el trueque tradicional, llevado a cabo de forma presencial o manual, presenta barreras significativas que limitan su adopción masiva:

### Dificultad para Encontrar el Intercambio

El problema fundamental del trueque tradicional es la "doble coincidencia de deseos". Esto implica que para que una transacción se concrete, la Parte A debe desear el bien ofrecido por la Parte B, y recíprocamente, la Parte B debe desear el bien ofrecido por la Parte A. En un entorno sin tecnología, la búsqueda de esta coincidencia recíproca resulta ser extremadamente rara y frustrante para los usuarios, haciendo que el proceso sea lento y, a menudo, ineficaz.

### Falta de Seguridad y Equidad

El trueque, al ser un intercambio directo entre individuos, carece de la infraestructura necesaria para generar confianza y asegurar el valor de la transacción.

- **Ausencia de valoración justa:** No existe un sistema estandarizado o una unidad de referencia que garantice que el valor percibido del intercambio es equitativo para ambas partes.
- **Riesgo de confianza:** La falta de mecanismos de reputación, valoración o perfiles verificados impide que los usuarios puedan confiar plenamente en la honestidad o fiabilidad de la otra persona antes de comprometerse con la transacción.

### Proceso Poco Práctico

En la sociedad digital actual, el trueque tradicional es percibido como un método anticuado y que requiere demasiado esfuerzo. No es visto como una alternativa de consumo de primera línea, ya que carece de las ventajas que ofrece el comercio electrónico moderno. Por lo tanto, el trueque no logra ser una opción viable, atractiva y eficiente para el usuario medio que valora la rapidez y la comodidad de la compra de artículos nuevos por internet.

## 2.3. Impacto esperado



El proyecto Swapply se justifica al transformar el trueque históricamente ineficiente en una solución digital práctica y viable, generando valor inmediato y estratégico en el mercado.

**Adiós a la Doble Coincidencia:** Superamos la dificultad de encontrar el intercambio perfecto con la implementación de un algoritmo avanzado de búsqueda y filtrado. Adicionalmente, se integra SwapCoins, nuestra moneda virtual, que permite intercambios indirectos (separando la entrega del bien y la adquisición del nuevo bien), lo cual aumenta dramáticamente la probabilidad de transacción exitosa.

### Seguridad y Confianza Garantizadas:

- Equidad de valor: SwapCoins actúa como una unidad de referencia estandarizada y objetiva. Esto es esencial para asegurar que la valoración del intercambio percibido por ambas partes sea justa.
- Riesgo controlado: El Sistema de Reputación y Valoración implementa los mecanismos necesarios para mitigar el riesgo. Esto permite a los usuarios construir y consultar la fiabilidad de las contrapartes, promoviendo un entorno de transacción seguro.

### Triple Impacto de Swapply

La eficiencia operativa de la plataforma se traduce directamente en beneficios estratégicos y medibles:

- **Beneficio Económico:** Fomenta el ahorro directo para la comunidad de usuarios al permitirles conseguir bienes y servicios que necesitan sin incurrir en un gasto monetario, utilizando el valor de objetos que actualmente no utilizan.
- **Beneficio Medioambiental:** Contribuye activamente a la Economía Circular al extender la vida útil de los productos (ropa, electrónica, etc.). De este modo, Swapply participa en la reducción de la necesidad de nueva producción y de la generación masiva de residuos.
- **Beneficio Técnico:** Demostramos que la digitalización eficiente del proceso convierte el trueque en una alternativa moderna y viable que compite directamente con la compra tradicional.

# 3. Objetivos del Proyecto



## 3.1. Objetivo General

El objetivo principal es desarrollar y lanzar Swapply como una plataforma digital de trueques, de alto rendimiento y preparada para el crecimiento futuro, que sirva como un intermediario tecnológico para gestionar de forma segura y eficiente el intercambio de bienes y servicios, posicionando la reutilización como una alternativa viable al consumo tradicional.

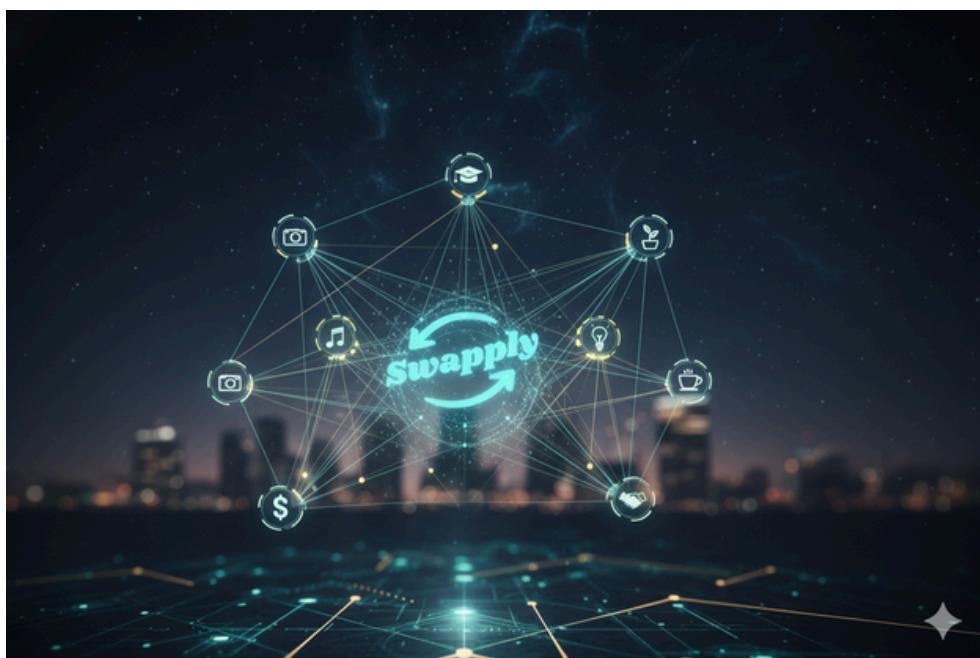
## 3.2. Objetivos Específicos

Para alcanzar el objetivo general del proyecto, se plantean los siguientes objetivos específicos, orientados al desarrollo completo y funcional de la plataforma:

- **Desarrollar la Interfaz de Usuario:** Se creará una interfaz de usuario accesible e intuitiva que garantice una experiencia óptima. Este objetivo incluye el desarrollo de las funcionalidades básicas para que cualquier usuario pueda registrarse, publicar ofertas (bienes y servicios) y gestionar sus solicitudes de intercambio de forma cómoda y eficiente.
- **Implementar el Sistema Central de Trueque y SwapCoins:** Programar la lógica interna del sistema para implementar un algoritmo de búsqueda y filtrado avanzado que optimice el matching entre ofertas y demandas. Conjuntamente, se diseñará la interfaz de SwapCoins que permita consultar un saldo estático o predefinido, demostrando su rol como unidad de referencia en un intercambio simulado..



- **Integrar Módulos de Bienes y Servicios:** Se desarrollarán dos secciones de contenido plenamente integradas en el sistema de trueque: una para el intercambio de bienes físicos y otra para la oferta de habilidades y servicios (como clases o reparaciones). Esto ampliará el valor de la plataforma y el abanico de posibilidades de intercambio.
- **Establecer la Infraestructura de Confianza:** Se implementarán los mecanismos de seguridad y transparencia necesarios. Esto incluye un sistema de registro seguro (con contraseñas cifradas) y un módulo de valoración para que los usuarios puedan calificar las transacciones. Además, se mantendrá un registro detallado de todos los trueques para garantizar la transparencia y facilitar la resolución de posibles conflictos.
- **Entregar la Versión 1.0 Validada:** Finalizar el ciclo de desarrollo, pruebas e implementación con la entrega de un producto estable. Este objetivo asegura que la plataforma está optimizada en rendimiento, es funcionalmente completa (con las características definidas en el alcance) y está libre de errores críticos antes de su despliegue público en el plazo estipulado en el cronograma.



# 4. Alcance del Proyecto



## 4.1. Funcionalidades que se incluirán

La versión desarrollada de Swapply incluirá un conjunto de funcionalidades derivadas de los casos de uso definidos en el apartado Arquitectura y Diseño del Sistema, organizadas en varios módulos principales. Estas funcionalidades representan el comportamiento real que podrá realizar el usuario dentro de la aplicación, considerando que el sistema funcionará de forma local, sin conexión a Internet y usando una base de datos interna.

### 1. Gestión de usuarios

El sistema permitirá a los usuarios crear y administrar sus cuentas dentro de la aplicación local. Las funcionalidades incluidas son:

#### 1.1. Registro de usuario

- Creación de una cuenta mediante correo electrónico y contraseña.
- Verificación de que no existan cuentas duplicadas.

#### 1.2. Inicio de sesión

- Acceso a la aplicación con las credenciales registradas.
- Comprobación local de usuario/contraseña.
- Gestión de errores por credenciales incorrectas.

#### 1.3. Gestión del perfil

- Visualización del perfil propio: nombre, foto y productos publicados.
- Edición de datos personales: nombre, contraseña, fotografía.
- Guardado de cambios en la base de datos local.



#### 1.4. Visualización de perfiles públicos

- Consultar la información básica de otros usuarios.
- Ver los productos publicados por cada usuario.

#### 1.5. Valoración de otros usuarios

- Enviar valoraciones numéricas (0 a 5 estrellas).
- Redactar un comentario sobre la experiencia del intercambio.

### 2. Gestión de productos

El sistema permitirá gestionar artículos para intercambiar dentro de la plataforma local.

#### 2.1. Publicar producto

- Introducir nombre del producto.
- Añadir descripción y estado.
- Añadir imágenes desde archivos locales.
- Registrar el producto en la base de datos.

#### 2.2. Retirar producto

- Eliminar un producto previamente publicado..
- Actualización del listado general de productos disponibles.

#### 2.3. Ver detalles de un producto

- Visualización de fotografías, descripción y estado del artículo.
- Visualización del perfil del vendedor.
- Acceso a acciones relacionadas (ver perfil público, iniciar contacto, guardar en wishlist).

#### 2.4. Wishlist

- Guardar productos favoritos para revisarlos posteriormente.
- Mostrar y gestionar la lista de productos guardados.



### 3. Navegación y búsqueda

La aplicación permitirá explorar los productos disponibles de forma flexible.

#### 3.1. Navegación por categorías

- Acceso a secciones como “Todos los productos” o “Según tus preferencias...”.
- Visualización de listados ordenados

#### 3.2. Búsqueda

- Uso de una barra de búsqueda por nombre o palabra clave.
- Manejo de resultados vacíos con alternativas sugeridas.

#### 3.3. Configuración de filtros de búsqueda

- Guardar intereses o palabras clave.
- Ajustar resultados mostrados en la pantalla de inicio según preferencias.

### 4. Sistema de intercambio (Matching)

Se implementarán las funciones base del proceso de intercambio, sin interacción real entre dispositivos externos.

#### 4.1. Realizar una propuesta de intercambio (Match)

- Selección del producto propio para ofrecer.
- Adjuntar mensaje descriptivo interno.
- Envío de oferta al otro usuario (gestión local).

#### 4.2. Notificación de propuesta recibida

- Registro interno de ofertas recibidas.
- Opción de aceptarlas o rechazarlas.

#### 4.3. Aceptar o rechazar propuesta

- Procesamiento local del estado de la propuesta.
- Cancelación automática si alguno de los usuarios la rechaza.

#### 4.4. Realizar envío/entrega

- Selección del método de entrega (postal o presencial).
- Registro interno del envío.



#### 4.5. Confirmar recepción

- Confirmación manual de recepción de producto.
- Registro del cierre del intercambio.

#### 4.6. Abrir disputa

- Permitir al usuario dejar constancia de un conflicto.
- Registrar una valoración negativa.

### 5. Sistema de notificaciones

Solo se implementará la parte local del sistema de notificaciones.

#### 5.1. Consultar notificaciones

- Acceso al listado de notificaciones generadas dentro de la aplicación.
- Posibilidad de acceder a la pantalla relacionada (producto, oferta, etc.).

#### 5.2. Gestionar preferencias de notificación

- Activar o desactivar tipos de notificaciones internas.
- Guardar ajustes de forma local.

### 6. Sistema de notificaciones

Según el punto Límites del Sistema (Qué no se desarrollará), no se desarrollará un chat real entre usuarios. Sin embargo, se incluye solo lo que sea localmente funcional:

#### 6.1. Consultar mensajes

- Acceso al listado de conversaciones almacenadas localmente.

#### 6.2. Consultar contenido de un mensaje

- Visualizar mensajes registrados (si existen para pruebas o simulaciones).

### 7. Sistema de Wallet

#### 7.1. Consultar saldo y movimientos

- Consultar el “saldo” y los “movimientos” simulados en SwapCoins.

## 4.2. Límites del Sistema



La versión de Swapply desarrollada para este proyecto funcionará de manera local, utilizando una base de datos interna y sin conexión a Internet. Debido a estas restricciones técnicas y al alcance definido en el apartado Funcionalidades que se incluirán, existen ciertas características y comportamientos que no serán implementados en esta versión del sistema, aunque puedan aparecer en los Casos de Uso del apartado Arquitectura y Diseño del Sistema como parte del diseño conceptual general.

A continuación se detallan las funcionalidades que no se desarrollarán:

### 1. Funcionalidades que requieren comunicación entre dispositivos externos

El sistema no permitirá interacción real entre distintos usuarios en diferentes ordenadores. Debido al funcionamiento offline:

- No habrá envío de datos entre dispositivos.
- No habrá sincronización de información entre varias instalaciones de Swapply.
- No se soportará el intercambio real entre dos usuarios remotos.

Cualquier interacción entre “usuarios distintos” se simulará localmente, sin comunicación real.

### 2. Sistema de chat o mensajería en tiempo real

Aunque en los casos de uso se menciona la existencia de un chat, su implementación no es viable sin conexión a Internet. Por tanto, no se desarrollará:

- Enviar mensajes a otro usuario real.
- Recibir mensajes o mantener conversaciones.
- Enviar o recibir códigos QR por chat.
- Notificaciones instantáneas de mensajes.

Únicamente podrá implementarse consulta local de conversaciones simuladas, si fuera necesario para pruebas.

### 3. Sistema de moneda virtual SwapCoin

El sistema de SwapCoins aparece en el diseño general, pero no se desarrollará su lógica real, ya que requiere interacción entre usuarios y validación de intercambios reales. No se implementará:

- Ganar o perder SwapCoins por intercambio.
- Procesar pagos con SwapCoins.
- Transacciones entre usuarios.
- Economía interna basada en saldo.

Solo se implementará consultar un saldo estático o simulado, si así se requiere para pruebas.



#### 4. Integraciones externas

Cualquier función que dependa de un servicio externo queda excluida. Esto excluye:

- Registro o inicio de sesión con cuentas externas (Google, Facebook, Apple...).
- Envío real de correos electrónicos.
- Confirmaciones mediante correo.
- Carga o descarga de recursos desde Internet.
- APIs de terceros.

Todos los procesos se resolverán localmente.

#### 6. Geolocalización y funciones basadas en ubicación

- No se implementará:
- Localización por GPS.
- Mostrar productos “cerca de ti”.
- Integración con Google Maps u otros servicios.

La navegación se basará únicamente en categorías y filtros locales.

#### 7. Confirmación de recepción mediante QR

El diseño conceptual incluye la confirmación mediante escaneo de QR entre usuarios. Esto requiere al menos un dispositivo adicional (teléfono móvil) y/o comunicación entre sistemas, por lo que NO se desarrollará. La confirmación de recepción se realizará de forma manual, registrando la acción localmente.

#### 8. Matching automático avanzado

Aunque el sistema de Matching está definido en los casos de uso, la lógica compleja de:

- Recomendaciones automáticas basadas en aprendizaje,
- Análisis avanzado de preferencias,
- Propuestas automáticas inteligentes,

queda fuera del alcance real. Solo se implementará la parte básica del flujo, de forma local y simplificada.

#### 9. Gestión real de disputas

La disputa se registrará como una simple valoración negativa o comentario, pero no se desarrollará:

- Un sistema de resolución real,
- Intervención de moderadores,
- Comunicación entre partes.

Se limitará a registrar la queja del usuario de manera local.

## 4.3. Entregables principales



A continuación se presentan los entregables previstos del proyecto Swapply, junto con sus fechas estimadas de entrega. Estos entregables están coordinados con el plan de trabajo general del equipo y representan los hitos clave del desarrollo.

### Hito H1 – Documento de Requisitos y Análisis (Final de diciembre)

Entregables:

- Especificación de requisitos funcionales y no funcionales.
- Análisis de riesgos del proyecto.
- Modelo de casos de uso completo.
- Documento de requisitos validado.

### Hito H2 – Diseño del Sistema (Final de enero)

Entregables:

- Documento de arquitectura del sistema.
- Diseño de APIs y esquema de autenticación/gestión de sesiones.
- Modelo de base de datos (DER + esquema lógico).
- Diagramas UML (clases, secuencia, actividades...).
- Prototipos de interfaz de usuario (wireframes o mockups).
- Diseño técnico consolidado y aprobado.

### Hito H3 – Primera Versión Funcional (Mitad de abril)

Entregables:

- Backend implementado (lógica de negocio, APIs, seguridad).
- Frontend implementado e integrado con el backend.
- Funcionalidades clave operativas:
  - Sistema de intercambio de productos/servicios.
  - Chat entre usuarios.
  - Sistema de valoraciones y opiniones.
  - Notificaciones básicas.
- Integración continua configurada.
- Primera versión funcional del sistema (versión alfa).



## Hito H4 – Validación y Pruebas (Mitad de mayo)

Entregables:

- Pruebas unitarias y de integración.
- Informe de pruebas de usabilidad.
- Pruebas de rendimiento/carga y reporte de resultados.
- Corrección de errores críticos y mejora de estabilidad.
- Sistema validado y estable (versión beta).

## Hito H5 – Despliegue y Publicación (Principio de junio)

Entregables:

- Configuración de servidor o infraestructura en la nube.
- Base de datos desplegada en entorno de producción.
- Dominio y certificado SSL configurados.
- Versión v1.0 de Swapply desplegada para uso real (1 junio).
- Publicación oficial de la plataforma.

## Fase de Mantenimiento (Desde publicación)

Entregables:

- Correcciones menores tras el despliegue.
- Mejoras de rendimiento.
- Incorporación de pequeños añadidos (p. ej., verificación de identidad).
- Monitorización de seguridad y actividad.



# 5. Requerimientos del Sistema



Esta sección detalla los requerimientos funcionales, no funcionales y de usuario para la plataforma Swapply. Este documento sirve como la base técnica para el desarrollo, las pruebas y la validación del producto, asegurando a los inversores que la visión del proyecto se traduce en un sistema robusto, escalable y centrado en la retención del usuario.

## Guía de Nomenclatura

Para facilitar la lectura, se utilizan los siguientes códigos en este documento:

- RU (Requerimientos de Usuario): Necesidades expresadas desde la visión del cliente.
- RF (Requerimientos Funcionales): Funciones técnicas que el sistema debe ejecutar.
  - G: Gestión de Usuarios.
  - T: Sistema de Trueque.
  - M: Moneda y Economía.
  - IA: Inteligencia Artificial.
- RNF (Requerimientos No Funcionales): Estándares de calidad y restricciones.
  - P: Performance (Rendimiento).
  - S: Seguridad.
  - U: Usabilidad y Fiabilidad.

## 5.1. Requerimientos de usuario

Describe las expectativas y necesidades del cliente final, validando el propósito de las funciones técnicas.

- **RU-01 (Flexibilidad Económica):** "Como profesor de guitarra, quiero dar clases a cambio de SwapCoins, porque ahora mismo no necesito nada, pero quiero acumular saldo para conseguir una reparación de fontanería en el futuro."



- **RU-02 (Cadenas de Valor):** "Como usuario, quiero conseguir un taladro que tiene Usuario A', aunque él no quiera mis libros. Quiero que el sistema encuentre automáticamente a un 'Usuario B' que quiera mis libros y ofrezca algo que 'Usuario A' sí necesite, cerrando el círculo."
- **RU-03 (Descubrimiento Inteligente):** "Como usuario ocupado, quiero que la IA me recomiende qué intercambiar basándose en mis habilidades y búsquedas pasadas, para no perder tiempo navegando entre ofertas irrelevantes."
- **RU-04 (Confianza y Seguridad):** "Como padre/madre, quiero contratar un servicio de cuidado infantil, pero solo quiero tratar con usuarios que tengan el 'Sello de Confianza' (identidad verificada) y buenas reseñas."
- **RU-05 (Ahorro B2B/B2C):** "Como diseñador gráfico, quiero ofrecer un 'paquete de logos' a cambio de asesoría fiscal, eliminando el intercambio monetario directo."
- **RU-06 Gestión de Oferta (Publicación):** "Como usuario, quiero publicar mi viejo portátil de forma rápida, añadiendo fotos y una descripción breve, para que esté visible en el catálogo de inmediato."

## 5.2. Requerimientos funcionales

Acciones específicas y operaciones que el sistema debe ejecutar.

### Módulo A: Gestión de Usuarios y Confianza (RF-G):

- **RF-G01 (Registro Institucional):** El sistema permitirá el registro y acceso prioritario mediante el correo electrónico oficial de la universidad. También se mantendrán opciones secundarias (Google/Apple) para posibles usuarios externos invitados
- **RF-G02 (Seguridad de Acceso 2FA):** Para el inicio de sesión con cuenta universitaria, el sistema implementará obligatoriamente una autenticación en dos pasos (2FA) similar al estándar utilizado por la universidad (App de autentificación) para garantizar la identidad del alumno.



## Módulo A: Gestión de Usuarios y Confianza (RF-G):

- **RF-G03 (Perfil Dual):** El perfil de usuario distinguirá claramente entre "Habilidades/Servicios" (Intangibles) e "Inventario/Bienes" (Tangibles).
- **RF-G04 (Reputación):** Tras cada transacción, el sistema obligará a una valoración mutua (1-5 estrellas) y reseña escrita.

## Módulo B: Núcleo de trueques y negociación (RF-T):

- **RF-T01 (Motor de Búsqueda):** Buscador con filtros avanzados: Proximidad (Geolocalización), Categoría, Valoración mínima y "Acepta SwapCoins".
- **RF-T02 (Negociación Segura):** Chat interno para acordar detalles sin exponer datos personales (teléfono/email) hasta confirmar el trato.
- **RF-T03 (Ciclo de Vida del Trueque):** El sistema gestionará los estados: Oferta Enviada -> En Negociación -> Acuerdo Aceptado -> En Proceso -> Confirmación Cruzada -> Completado.

## Módulo C: Economía virtual (SwapCoins) (RF-M):

- **RF-M01 (Billetera Virtual):** Cada usuario dispondrá de una Wallet para consultar saldo e historial de transacciones.
- **RF-M02 (Bonificación):** El sistema inyectará SwapCoins como recompensa por completar los primeros 3 trueques con éxito para incentivar el uso.
- **RF-M03 (Pago Híbrido):** El sistema permitirá ofertas mixtas: Bien/Servicio + Diferencia en SwapCoins (ej. "Te cambio mi bici vieja + 20 SwapCoins por tu patinete eléctrico").



## Módulo D: Algoritmos avanzados e IA (RF-IA):

- **RF-IA01 (Trueque Circular A-B-C):** El sistema ejecutará un algoritmo automático para detectar ciclos cerrados de intercambio entre 3 o más usuarios y notificará a los implicados simultáneamente.
- **RF-IA02 (Motor de Recomendación):** Algoritmo inteligente que analiza el historial de trueques y los "Me gusta" para sugerir oportunidades proactivamente en el panel principal.

## 5.3. Requerimientos no funcionales (RNF)

Atributos de calidad, rendimiento y restricciones técnicas.

### Rendimiento y Escalabilidad (RNF-P)

- **RNF-P01 (Velocidad):** El tiempo de carga inicial de la página debe ser inferior a 2.5 segundos en conexiones estándar.
- **RNF-P02 (Respuesta IA):** El sistema de recomendación debe devolver sugerencias en menos de 1.5 segundos.
- **RNF-P03 (Escalabilidad):** La arquitectura del sistema debe ser modular para soportar picos de 10,000 usuarios concurrentes garantizando la estabilidad.
- **RNF-P04 (Eficiencia):** El algoritmo de "Trueques Circulares" se ejecutará en segundo plano durante horarios de baja carga para no ralentizar la navegación de los usuarios activos.

### Seguridad y Datos (RNF-S)

- **RNF-S01 (Cifrado):** Todas las comunicaciones usarán conexión segura (HTTPS) y los datos sensibles (contraseñas, DNI) se guardarán encriptados en la base de datos para máxima seguridad.



## Seguridad y Datos (RNF-S)

- **RNF-S02 (Integridad de Monedas):** Las operaciones con SwapCoins tendrán mecanismos de seguridad transaccional para evitar duplicación, pérdida de saldo o errores en los pagos.
- **RNF-S03 (Privacidad):** Cumplimiento estricto de la normativa europea de protección de datos (GDPR); la ubicación exacta no se comparte hasta cerrar el acuerdo.

## Usabilidad y Fiabilidad (RNF-U)

- **RNF-U01 (Diseño Móvil):** La interfaz estará diseñada para móviles y funcionará de manera fluida en cualquier navegador.
- **RNF-U02 (Regla de los 3 clics):** El proceso de publicación de una oferta no debe superar los 3 pasos o 60 segundos.
- **RNF-U03 (Disponibilidad):** El sistema garantizará que la plataforma esté operativa el 99.9% del tiempo anual.

### SYSTEM REQUIREMENTS



# 6. Arquitectura y Diseño del Sistema



## 6.1. Arquitectura general

El sistema Swapply está basado en una Arquitectura Orientada a Servicios (SOA) y dirigida por eventos (Event-Driven Architecture). Esta estructura responde a la necesidad de separar la lógica transaccional estándar (gestión de la información correspondiente a los usuarios y productos) de la lógica computacional compleja (cálculo de grafos y cadenas de trueque).

Por este mismo motivo, Swapply está diseñado sobre un patrón híbrido en su capa de persistencia. Por un lado, Swapply cuenta con una Base de Datos Relacional que asegura la integridad de los datos financieros (SwapCoins), inventario y usuarios. De forma contigua, del modelado de relaciones complejas entre deseos y pertenencias (Matching Multilateral) se encarga una segunda Base de Datos, esta, orientada a Grafos.

Para la comunicación entre subsistemas Swapply cuenta con un bus de eventos que, de forma asíncrona, garantiza la escalabilidad y reduce significativamente la posibilidad de bloqueo de procesos críticos.



## 6.2. Definición de Subsistemas



La estructura interna de Swapply se compone de seis subsistemas, los cuales tienen una responsabilidad única y delimitada. Estos subsistemas son:

- **6.2.1 Subsistema de Identidad y Reputación**

Este módulo es el encargado de administrar la gestión de acceso, seguridad y confianza entre pares (usuarios que realizan los intercambios). Dentro de las responsabilidades acotadas en este subsistema se encuentran la autenticación y autorización (registro/login), la gestión de los perfiles de usuario y el algoritmo encargado de calcular la ponderación de las valoraciones recibidas de cada usuario para determinar el nivel de confianza que se asigna al mismo (crítico para la prioridad dada en las cadenas de trueque).

- **6.2.2 Subsistema de inventario y Wishlist**

Se trata del repositorio central de los inputs (lo que se tiene) y los outputs (lo que se quiere) de la aplicación, individuales para cada usuario. Este modulo es el encargado de gestionar el ciclo de vida de los productos dentro de Swapply (Publicación - Pausa - Retiro), además de la categorización y etiquetado de ítems para facilitar y optimizar el proceso de búsqueda por parte de los usuarios. Junto a esto se añade la gestión de las Wishlist correspondientes a cada usuario para mejorar las recomendaciones personalizadas de la aplicación, puesto que se trata de la fuente de datos primaria para el motor de matching.

- **6.2.3 Subsistema de Matching Inteligente**

Este modulo se trata del cerebro matemático del sistema. Se trata de un servicio aislado optimizado para el recorrido de grafos.

La funcionalidad principal es la de realizar las conexiones en base a los datos del inventario de productos de la base de datos y las Wishlist de cada usuario. De este modo, este algoritmo es capaz de detectar ciclos en tiempo real para así cerrar cadenas de intercambio entre usuarios, generando así de forma óptima posibles ofertas de intercambio.



- **6.2.4 Subsistema de Gestión de Transacciones**

Este módulo se compondría de una máquina de estados finitos que orquesta los procesos de intercambio.

Las tareas que este módulo ejerce son: la verificación de que todas las partes del intercambio estén de acuerdo en realizarlo, el bloqueo de stock de los productos y las SwapCoins implicadas en el intercambio para evitar problemas en el intercambio, gestionar los distintos estados del intercambio (Propuesto > Aceptado > Enviado > Recibido > Finalizado) y de la gestión de posibles disputas y cancelaciones.

- **6.2.5 Subsistema Económico**

Este módulo está dedicado expresamente a la gestión, procesamiento y registro de las SwapCoins.

Se trata de una base de datos dedicada cuya tarea es tramitar todos los pagos en los que haya implicadas SwapCoins, ya sean exclusivamente con SwapCoins o pagos mixtos (Producto + SwapCoins). Además de la gestión de monedas correspondientes a cada usuario. Sumado a todo esto, se mantendrá un historial de todos los movimientos financieros y los usuarios implicados en ellos con la finalidad de aportar la mayor transparencia posible a dichas transacciones.

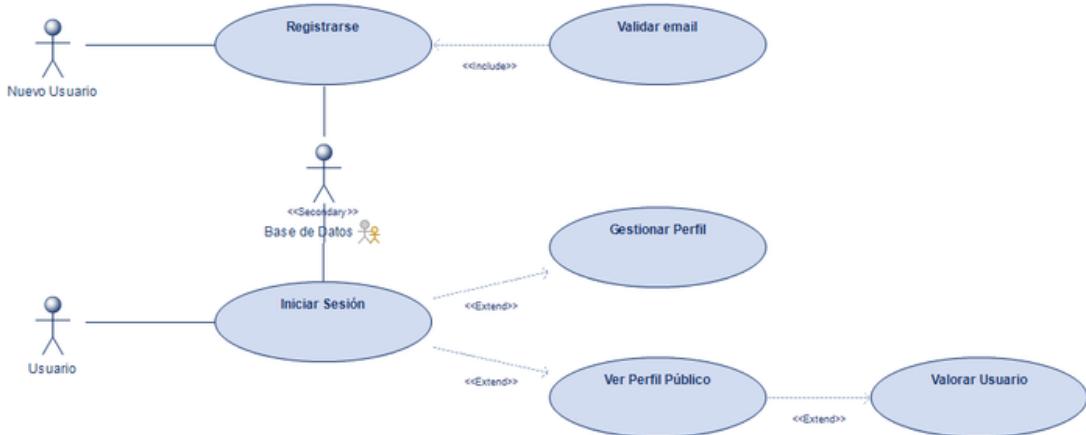
- **6.2.6 Subsistema de Notificaciones y Comunicación**

Esta es la capa de interacción entre los usuarios y el sistema. Este módulo está encargado exclusivamente de enviar a los usuarios notificaciones o emails respecto a nuevas ofertas de intercambio o el estado de pedidos.

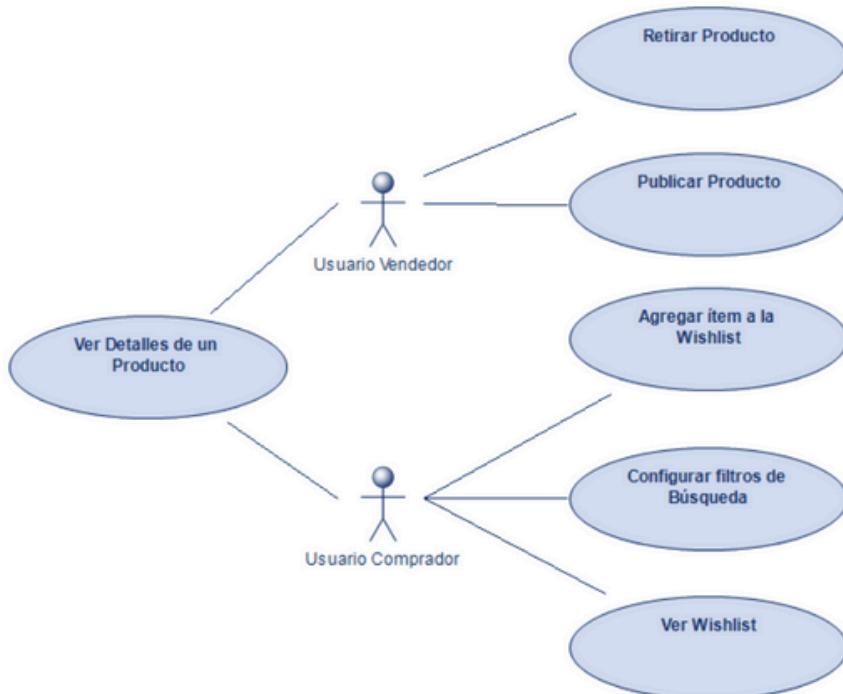


## 6.2.1 Diagramas de casos de uso

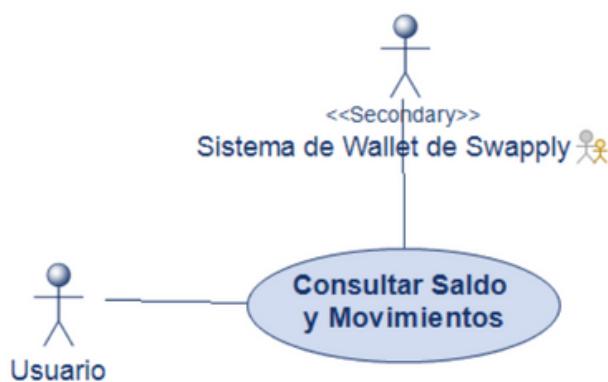
- Subsistema de Identidad y Reputación



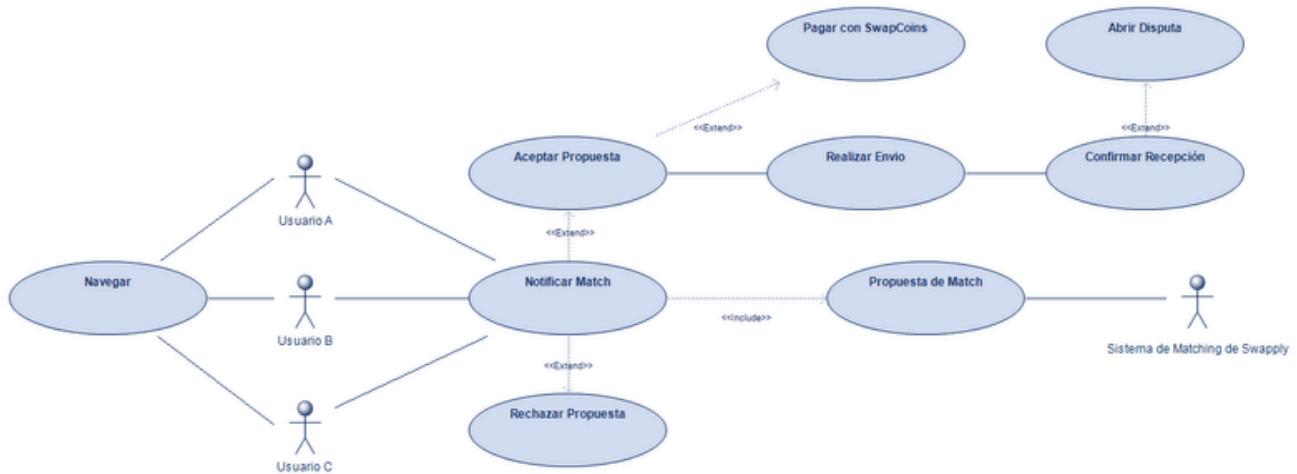
- Subsistema de Inventory y Wishlist



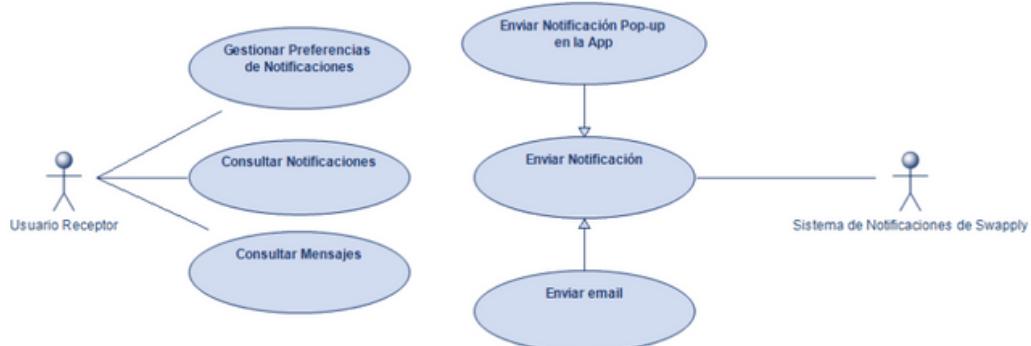
- Subsistema Económico



- Subsistema de Transacciones y Matching



- Subsistema de Notificaciones y Comunicación



## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Identidad y Reputación

| Autor            | Carlos Belmonte Arias   |  |
|------------------|---|--|
| [CU-IR01]        | Gestionar Perfil  |  |
| Descripción      | El usuario navega a su perfil para revisar su información y su historial de publicaciones y realizar los cambios oportunos. |  |
| Actores          | Principal: Usuario  |  |
| Pre condiciones  | El usuario hace clic en el ícono de su perfil en la navegación.   |  |
| Post condiciones | El usuario ve su perfil y edita la información en él a su gusto.  |  |
| Secuencia Normal | #   | Acción   |
|                  | 1   | El sistema muestra la foto de perfil, nombre y productos publicados del usuario.   |
|                  | 2   | El usuario hace clic en el botón de gestión de su perfil y se le redirige a la pestaña de edición de su perfil.  |
|                  | 3   | El usuario cambia la información a su gusto. <ul style="list-style-type: none"> <li>• 3.1 El usuario cambia su nombre en la aplicación.</li> <li>• 3.2 El usuario cambia su contraseña.</li> <li>• 3.3 El usuario añade o modifica su foto de perfil.</li> </ul> |
|                  | 4   | El usuario guarda los cambios.   |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Identidad y Reputación

|                         |  |  |
|-------------------------|--|--|
| <b>Autor</b>            | Carlos Belmonte Arias  |  |
| <b>[CU-IR02]</b>        | Iniciar sesión   |  |
| <b>Descripción</b>      | El usuario utilizará sus datos para acceder a su cuenta en la aplicación (creada anteriormente). |  |
| <b>Actores</b>          | Principal: Usuario   |  |
| <b>Pre condiciones</b>  | El usuario se encuentra registrado en la aplicación.   |  |
| <b>Post condiciones</b> | El usuario accede a Swapply con su cuenta.   |  |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>  |
|                         | 1  | El usuario elige el método por el que iniciar sesión.  |
|                         |  | <ul style="list-style-type: none"> <li>1.1 El usuario inicia sesión con cuentas externas (Cuenta UCM, Facebook, Gmail o Apple) y es redirigido a sus respectivas páginas de inicio de sesión.</li> <li>1.2 El usuario inicia sesión con un email.</li> </ul> |
|                         | 2  | El usuario introduce sus credenciales y el sistema le da acceso a la aplicación.   |
| <b>Excepciones</b>      | #  | <b>Acción (actor)</b>  |
|                         | 2-a.   | Si el correo o la contraseña son inválidos se le solicitará al usuario que pruebe de nuevo con otras credenciales o se registre en la aplicación.  |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Identidad y Reputación

|                         |  |   |
|-------------------------|--|---|
| <b>Autor</b>            | Carlos Belmonte Arias  |   |
| <b>[ CU-IR03 ]</b>      | Registrarse  |   |
| <b>Descripción</b>      | El usuario que accede por primera vez a Swapply se registra para así poder hacer uso de la aplicación sin restricciones (realizar ofertas de intercambio o subir productos propios). |   |
| <b>Actores</b>          | Principal: Nuevo Usuario<br>Secundario: Base de Datos de Swapply   |   |
| <b>Pre condiciones</b>  | El usuario abre la aplicación por primera vez.   |   |
| <b>Post condiciones</b> | El usuario queda registrado en la base de datos con una cuenta asociada.   |   |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>   |
|                         | 1  | El usuario elige el método por el que registrarse.  |
|                         |  | <ul style="list-style-type: none"> <li>1.1 El usuario elige registrarse con cuentas externas (Cuenta UCM, Facebook, Gmail o Apple) y es redirigido a sus respectivas páginas de inicio de sesión.</li> <li>1.2 El usuario se registra con un email y una contraseña.</li> </ul> |
|                         | 2  | El usuario crea su cuenta con su correo y contraseña y estos se almacenan cifrados en la base de datos.   |
|                         | 3  | El usuario valida su cuenta con un código enviado al correo proporcionado.  |
|                         |  |   |
| <b>Excepciones</b>      | #  | <b>Acción (actor)</b>   |
|                         | 2-a.   | Si ya existe una cuenta con ese correo el sistema avisará al usuario de ello y le pedirá que utilice otro correo o inicie sesión con él.  |
|                         | 3-a.   | Si no se verifica la cuenta en 72 horas, es eliminada de la base de datos.  |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Identidad y Reputación

|                         |  |   |
|-------------------------|--|---|
| <b>Autor</b>            | Carlos Belmonte Arias  |   |
| <b>[ CU-IR04]</b>       | Valorar Usuario  |   |
| <b>Descripción</b>      | El usuario A deja una valoración en el perfil del usuario B.                                     |   |
| <b>Actores</b>          | Principal: Usuario   |   |
| <b>Pre condiciones</b>  | El usuario hace clic en el botón de “Valorar Usuario” en el perfil de otro usuario.              |   |
| <b>Post condiciones</b> | El usuario deja una valoración para que otros usuarios puedan conocer mejor al usuario valorado. |   |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>   |
|                         | 1  | El usuario escribe una valoración sobre su interacción con el usuario valorado y muestra su índice de conformidad de forma visual puntuando de 0 a 5 estrellas. |

|                         |  |  |
|-------------------------|--|--|
| <b>Autor</b>            | Carlos Belmonte Arias  |  |
| <b>[CU-IR05]</b>        | Ver Perfil Público   |  |
| <b>Descripción</b>      | El usuario A navega por el perfil de otro usuario B para revisar su información. |  |
| <b>Actores</b>          | Principal: Usuario   |  |
| <b>Pre condiciones</b>  | El usuario hace clic en el ícono del perfil de otro usuario.                     |  |
| <b>Post condiciones</b> | El usuario ve la información disponible del perfil del otro usuario.             |  |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>  |
|                         | 1  | El sistema muestra la foto de perfil, nombre y productos publicados del usuario. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Inventario y Wishlist

|                         |   |  |
|-------------------------|---|--|
| <b>Autor</b>            | Carlos Belmonte Arias   |  |
| <b>[CU-IW01]</b>        | Agregar ítem a la Wishlist  |  |
| <b>Descripción</b>      | El usuario guarda en su Wishlist un producto o servicio para poder volver a él más tarde. |  |
| <b>Actores</b>          | Principal: Usuario  |  |
| <b>Pre condiciones</b>  | El usuario hace clic en el ícono de “Agregar a la Wishlist”.                              |  |
| <b>Post condiciones</b> | El producto queda guardado en la Wishlist del usuario.                                    |  |
| <b>Secuencia Normal</b> | #   | <b>Acción</b>  |
|                         | 1   | El sistema guarda el producto o servicio en la Wishlist del usuario. |

|                         |  |   |
|-------------------------|--|---|
| <b>Autor</b>            | Carlos Belmonte Arias  |   |
| <b>[ CU-IW02]</b>       | Configurar filtros de Búsqueda   |   |
| <b>Descripción</b>      | El usuario modificará los filtros de búsqueda de su algoritmo para que se le muestren resultados acordes a sus preferencias. |   |
| <b>Actores</b>          | Principal: Usuario   |   |
| <b>Pre condiciones</b>  | El usuario hace clic en el cuadro de “Intereses” en su perfil.   |   |
| <b>Post condiciones</b> | Los intereses del usuario quedan almacenados para el algoritmo de Swapply.   |   |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>   |
|                         | 1  | El usuario introduce palabras clave en el cuadro de texto.  |
|                         | 2  | El algoritmo guarda las palabras clave para mostrar resultados relacionados en la página de inicio. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Inventario y Wishlist

|                         |   |  |
|-------------------------|---|--|
| <b>Autor</b>            | Carlos Belmonte Arias   |  |
| [ CU-IW03]              | Publicar Producto   |  |
| <b>Descripción</b>      | El usuario subirá a la base de datos de la aplicación el producto o servicio que quiera ofrecer al resto de usuarios para intercambiar. |  |
| <b>Actores</b>          | Principal: Usuario  |  |
| <b>Pre condiciones</b>  | El usuario hace clic en el botón “Subir producto” y se le lleva a la página para introducir la información de este.                     |  |
| <b>Post condiciones</b> | El producto o servicio queda registrado en la aplicación para que otros usuarios puedan ofrecer algo a cambio.                          |  |
| <b>Secuencia Normal</b> | #   | <b>Acción</b>  |
|                         | 1   | El usuario introduce el nombre con el que desea subir el producto o servicio.  |
|                         | 2   | El usuario detalla una breve descripción del producto o servicio y el estado en el que se encuentra.                           |
|                         | 3   | El usuario sube una o varias fotos del producto o resultados del servicio que ofrece.  |
| <b>Excepciones</b>      | #   | <b>Acción (actor)</b>  |
|                         | 1-a.  | El usuario no introduce un nombre para el producto o servicio, la aplicación le informa y no le permite seguir con el proceso. |
|                         | 2-a.  | El usuario no introduce una descripción para el producto o servicio.   |
|                         | 3-a.  | El usuario no sube ninguna foto del producto o servicio.   |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Inventario y Wishlist

|                  |   |   |
|------------------|---|---|
| Autor            | Carlos Belmonte Arias   |   |
| [ CU-IW04]       | Retirar Producto  |   |
| Descripción      | El usuario retirará un producto o servicio previamente publicado que ya no quiera ofrecer al resto de usuarios para intercambiar. |   |
| Actores          | Principal: Usuario  |   |
| Pre condiciones  | El usuario hace clic en el botón “Retirar producto”.  |   |
| Post condiciones | El producto o servicio se retira de la aplicación   |   |
| Secuencia Normal | #   | Acción  |
|                  | 1   | El sistema elimina el producto o servicio de la lista de productos disponibles. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Inventario y Wishlist

|                         |   |  |
|-------------------------|---|--|
| <b>Autor</b>            | Carlos Belmonte Arias   |  |
| [ CU-IW05]              | Ver Detalles de un Producto   |  |
| <b>Descripción</b>      | El usuario selecciona un producto o servicio de la lista de resultados para ver sus fotos, descripción, precio, estado y detalles del vendedor. |  |
| <b>Actores</b>          | Principal: Usuario  |  |
| <b>Pre condiciones</b>  | El usuario hace clic sobre un producto o servicio en la pestaña de navegación principal.  |  |
| <b>Post condiciones</b> | El usuario ha terminado de revisar la información del producto o servicio.  |  |
| <b>Secuencia Normal</b> | #   | <b>Acción</b>  |
|                         | 1   | El sistema carga y muestra la pestaña con todos los detalles del artículo o servicio.  |
|                         | 2   | El usuario revisa el carrusel de imágenes, la descripción y la condición del producto o servicio.  |
|                         | 3   | El usuario puede realizar acciones relacionadas con el producto o servicio y su vendedor.  |
|                         |   | <ul style="list-style-type: none"> <li>• 3.1 El usuario manda un mensaje al vendedor.</li> <li>• 3.2 El usuario envía una oferta al vendedor.</li> </ul>     |
| <b>Excepciones</b>      | #   | <b>Acción (actor)</b>  |
|                         | 1-a.  | El sistema no puede cargar la información porque el producto o servicio ya ha sido retirado o vendido y muestra un mensaje que informa al usuario del error. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Inventario y Wishlist

|                         |   |  |
|-------------------------|---|--|
| <b>Autor</b>            | Carlos Belmonte Arias   |  |
| [ CU-IW06]              | Ver Wishlist  |  |
| <b>Descripción</b>      | El usuario consulta los productos o servicios guardados en su Wishlist. |  |
| <b>Actores</b>          | Principal: Usuario  |  |
| <b>Pre condiciones</b>  | El usuario hace clic el ícono de la Wishlist.                           |  |
| <b>Post condiciones</b> | El usuario ha terminado de revisar su Wishlist.                         |  |
| <b>Secuencia Normal</b> | #   | <b>Acción</b>  |
|                         | 1   | El sistema carga y muestra la pestaña con la Wishlist del Usuario.   |
|                         | 2   | El usuario revisa los productos y servicios guardados.   |
|                         | 3   | El usuario puede consultar los detalles de los ítems guardados.  |
| <b>Excepciones</b>      | #   | <b>Acción (actor)</b>  |
|                         | 1-a.  | El sistema no puede cargar la información porque el producto o servicio ya ha sido retirado o vendido y muestra un mensaje que informa al usuario del error. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

|                  |  |   |
|------------------|--|---|
| Autor            | Carlos Belmonte Arias  |   |
| [CU-TM01]        | Propuesta de Match   |   |
| Descripción      | El sistema o usuario comprador ofrecerá una oferta de intercambio a otro usuario vendedor por un producto o servicio propio.   |   |
| Actores          | Principal: Usuario/s comprador/es, Sistema de Matching de Swapply<br>Secundario: Usuario vendedor  |   |
| Pre condiciones  | El usuario hace clic en el botón de "Realizar oferta de intercambio" en un producto/servicio o el algoritmo de Swapply manda la oferta de intercambio a todas las partes implicadas. |   |
| Post condiciones | Se informa al usuario vendedor de la oferta del usuario comprador.   |   |
| Secuencia Normal | #  | Acción  |
|                  | 1  | Los usuarios compradores adjuntan el producto que ofrecen para el intercambio   |
|                  | 3  | Los usuarios compradores envían la oferta al vendedor y este recibe un mensaje en su bandeja de notificaciones con la oferta y el mensaje de los compradores (si lo hubiera). |
| Excepciones      | #  | Acción (actor)  |
|                  | 1-a.   | El comprador no adjunta ningún producto o servicio, la aplicación no le permite avanzar.  |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

|                  |   |  |
|------------------|---|--|
| Autor            | Carlos Belmonte Arias   |  |
| [ CU-TM02]       | Notificar Match   |  |
| Descripción      | El sistema notificara a los usuarios implicados en un intercambio para que estos lo acepten o rechacen. |  |
| Actores          | Principal: Usuarios participes del intercambio  |  |
| Pre condiciones  | El sistema hace match y la aplicación manda la oferta de intercambio a todas las partes implicadas.     |  |
| Post condiciones | Se informa a los usuarios implicados y estos aceptan o rechazan la oferta.                              |  |
| Secuencia Normal | #   | Acción   |
|                  | 1   | Los usuarios presionan un botón para decidir qué hacer.  |
|                  |   | <ul style="list-style-type: none"> <li>• 1.1 El usuario presiona el botón de “Aceptar Propuesta”.</li> <li>• 1.2 El usuario presiona el botón de “Rechazar Propuesta”</li> </ul> |
|                  | 2   | El sistema procesa las respuestas de los usuarios y si alguno de ellos ha rechazado la propuesta se cancela la transacción.  |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

|                         |  |   |
|-------------------------|--|---|
| <b>Autor</b>            | Carlos Belmonte Arias  |   |
| [ CU-TM03]              | Navegar  |   |
| <b>Descripción</b>      | El usuario podrá desplazarse por la página principal de la aplicación donde se le muestran los productos y servicios disponibles por categorías. |   |
| <b>Actores</b>          | Principal: Usuario   |   |
| <b>Pre condiciones</b>  | El usuario hace clic en el botón de “Inicio” o realiza una búsqueda de un tipo de producto o servicio específico.                                |   |
| <b>Post condiciones</b> | Se le muestran al usuario distintos productos o servicio que le puedan interesar.  |   |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>   |
|                         | 1  | El usuario selecciona la página por la que quiere navegar.  |
|                         |  | <ul style="list-style-type: none"> <li>• 1.1 El usuario selecciona la pestaña de productos “Según tus preferencias...”.</li> <li>• 1.2 El usuario selecciona la pestaña de productos “Cerca de ti...”.</li> <li>• 1.3 El usuario selecciona la pestaña de productos “Todos los productos”.</li> <li>• 1.4 El usuario utiliza la barra de búsqueda para buscar un tipo de producto en específico.</li> </ul> |
|                         | 2  | El usuario explora las opciones ofertadas y puede hacer clic en una para ver su descripción detallada.  |
| <b>Excepciones</b>      | #  | <b>Acción (actor)</b>   |
|                         | 1.4-a.   | No existen resultados para el producto buscado y el sistema le muestra al usuario productos similares o relacionados con su tipo (herramientas, hogar, electrónica...)  |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

| Autor            | Carlos Belmonte Arias  |  |
|------------------|--|--|
| [ CU-TM04]       | Realizar Envío   |  |
| Descripción      | Los usuarios realizan el envío de sus productos por correo postal o de intercambiándolos en persona o realizando el servicio ofrecido. |  |
| Actores          | Principal: Usuario comprador, Usuario vendedor   |  |
| Pre condiciones  | Los usuarios han aceptado la propuesta de intercambio.   |  |
| Post condiciones | Los productos se envían con éxito.   |  |
| Secuencia Normal | #  | Acción   |
|                  | 1  | Los usuarios eligen el método de envío.  |
|                  |  | <ul style="list-style-type: none"><li>• 1.1 Envío postal</li><li>• 1.2 Entrega en persona</li><li>• 1.3 Realización del servicio</li></ul> |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

|                  |  |   |
|------------------|--|---|
| Autor            | Carlos Belmonte Arias  |   |
| [ CU-TM05]       | Confirmar Recepción  |   |
| Descripción      | Los usuarios confirman que los productos han sido recibidos o el servicio realizado.       |   |
| Actores          | Principal: Usuarios implicados   |   |
| Pre condiciones  | Los usuarios han enviado o se han reunido presencialmente para intercambiar sus productos. |   |
| Post condiciones | Los usuarios confirman la recepción de sus productos.                                      |   |
| Secuencia Normal | #  | Acción  |
|                  | 1  | Los usuarios reciben los productos. <ul style="list-style-type: none"> <li>• 1.1 Por envío postal</li> <li>• 1.2 De forma presencial</li> </ul> |
|                  | 2  | Cada uno escanea el código QR en la aplicación de Swapply con su teléfono móvil y cierran el intercambio.                                       |
| Excepciones      | #  | Acción (actor)  |
|                  | 2-a.   | Si el envío se realiza por correo postal los usuarios se pueden enviar su código QR por email o impreso junto al paquete.                       |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Transacciones y Matching

|                  |  |   |
|------------------|--|---|
| Autor            | Carlos Belmonte Arias  |   |
| [ CU-TM06]       | Abrir Disputa  |   |
| Descripción      | Un usuario descontento notifica en la aplicación las molestias ocasionadas por otro usuario. |   |
| Actores          | Principal: Usuario   |   |
| Pre condiciones  | El usuario ha intercambiado un producto o servicio con otro y ha quedado descontento.        |   |
| Post condiciones | El usuario reporta su disconformidad en la aplicación.                                       |   |
| Secuencia Normal | #  | Acción  |
|                  | 1  | El usuario entra al perfil del vendedor.  |
|                  | 2  | El usuario deja una valoración negativa sobre los problemas que le han ocasionado durante el intercambio. |

|                  |  |  |
|------------------|--|--|
| Autor            | Carlos Belmonte Arias  |  |
| [ CU-TM07]       | Reclamar SwapCoins de Intercambio  |  |
| Descripción      | El usuario reclama las monedas que se le deben por un intercambio.                         |  |
| Actores          | Principal: Usuario   |  |
| Pre condiciones  | El usuario ha intercambiado un producto o servicio con otro usuario a cambio de SwapCoins. |  |
| Post condiciones | El usuario recibe sus SwapCoins.   |  |
| Secuencia Normal | #  | Acción   |
|                  | 1  | El usuario hace clic en el botón de "Reclamar SwapCoins".              |
|                  | 2  | El usuario recibe las SwapCoins y se actualiza el saldo de su cartera. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Económico

|                         |   |  |
|-------------------------|---|--|
| <b>Autor</b>            | Carlos Belmonte Arias   |  |
| [ CU-E01]               | Consultar Saldo y Movimientos   |  |
| <b>Descripción</b>      | El usuario consulta su saldo de SwapCoins y sus movimientos recientes.                |  |
| <b>Actores</b>          | Principal: Usuario<br>Secundario: Sistema de Wallet de Swapply.                       |  |
| <b>Pre condiciones</b>  | El usuario hace clic en el ícono de “SwapCoins”.                                      |  |
| <b>Post condiciones</b> | El usuario queda informado de sus últimos movimientos y su saldo en SwapCoins actual. |  |
| <b>Secuencia Normal</b> | #   | <b>Acción</b>  |
|                         | 1   | El usuario navega por la pestaña para consultar sus últimos movimientos. |
|                         | 2   | El sistema le muestra en la parte superior de la pestaña su saldo.       |

- Subsistema de Notificaciones y Comunicación

|                         |  |  |
|-------------------------|--|--|
| <b>Autor</b>            | Carlos Belmonte Arias  |  |
| [ CU-NC01]              | Gestionar Preferencias de Notificaciones   |  |
| <b>Descripción</b>      | El usuario indica a la aplicación las notificaciones que quiere o no quiere recibir. |  |
| <b>Actores</b>          | Principal: Usuario   |  |
| <b>Pre condiciones</b>  | El usuario entra a la configuración de las notificaciones.                           |  |
| <b>Post condiciones</b> | Swapply guarda las preferencias del usuario.   |  |
| <b>Secuencia Normal</b> | #  | <b>Acción</b>  |
|                         | 1  | El usuario indica que notificaciones quiere o no quiere recibir. |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema



- Subsistema de Notificaciones y Comunicación

|                  |   |   |
|------------------|---|---|
| Autor            | Carlos Belmonte Arias   |   |
| [ CU-NC02]       | Consultar Notificaciones  |   |
| Descripción      | El usuario accede a una lista cronológica de eventos relevantes para su cuenta y productos. |   |
| Actores          | Principal: Usuario  |   |
| Pre condiciones  | El usuario hace clic en el ícono de "Notificaciones" en la navegación principal.            |   |
| Post condiciones | El usuario ha revisado sus alertas y ha interactuado con una de ellas.                      |   |
| Secuencia Normal | #   | Acción  |
|                  | 1   | El sistema carga y muestra una lista de notificaciones pendientes y anteriores, ordenadas por fecha.                        |
|                  | 2   | El usuario selecciona una notificación específica (ej. "Nueva Oferta").   |
|                  | 3   | El sistema dirige al usuario a la pantalla o producto relacionado con esa alerta (ej. a la pantalla de la oferta recibida). |

## 6.2.2 Especificación de los Casos de Uso por cada Subsistema

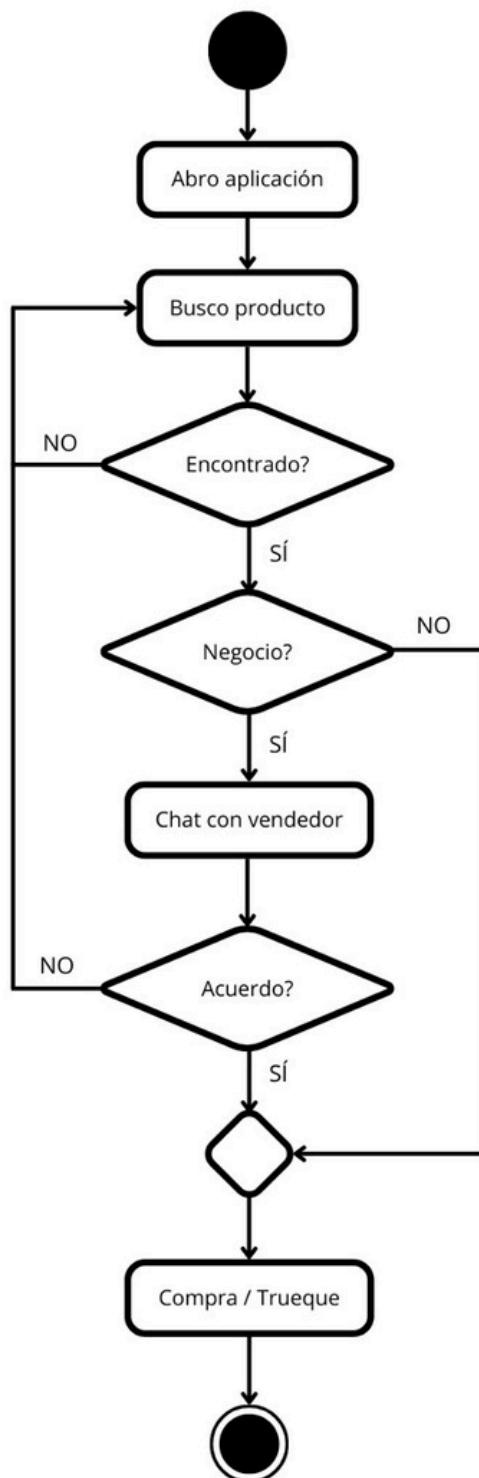


- Subsistema de Notificaciones y Comunicación

|                  |  |  |
|------------------|--|--|
| Autor            | Carlos Belmonte Arias  |  |
| [ CU-NC03]       | Enviar Notificación  |  |
| Descripción      | El sistema envía una notificación al usuario en función de sus preferencias. |  |
| Actores          | Principal: Sistema de Notificaciones de Swapply                              |  |
| Pre condiciones  | Existe una notificación pendiente para un usuario.                           |  |
| Post condiciones | El sistema envía la notificación al usuario.                                 |  |
| Secuencia Normal | #  | Acción   |
|                  | 1  | El sistema envía la notificación.  |
|                  |  | <ul style="list-style-type: none"> <li>• 1.1 Por correo electrónico.</li> <li>• 1.2 Por medio de una notificación pop-up en la aplicación de Swapply.</li> </ul> |
| Excepciones      | #  | Acción (actor)   |
|                  | 1.1-a.   | El correo electrónico ha sido eliminado y no se puede enviar el mensaje.   |
|                  | 1.2-a.   | El usuario ha eliminado la aplicación y no se puede enviar el mensaje.   |

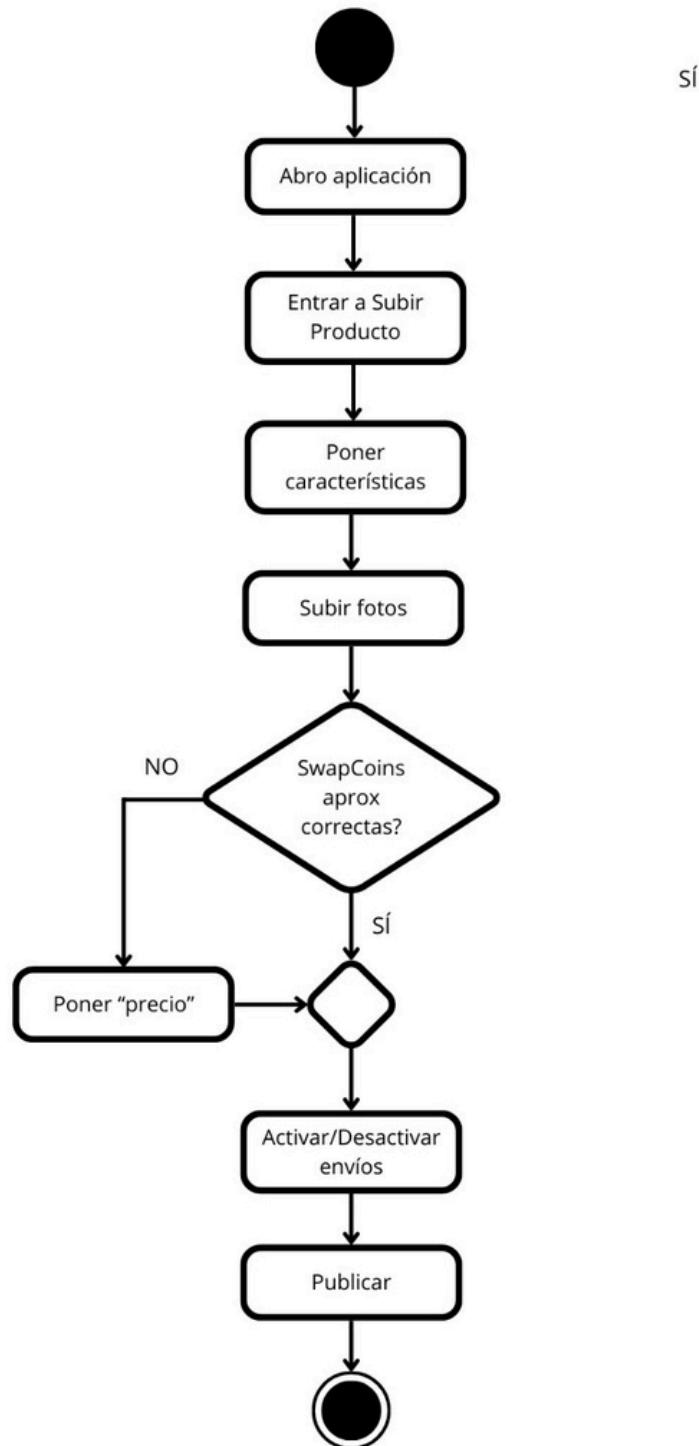


## 6.2.3 Dos diagramas de actividad de los casos más importantes





## 6.2.3 Dos diagramas de actividad de los casos más importantes



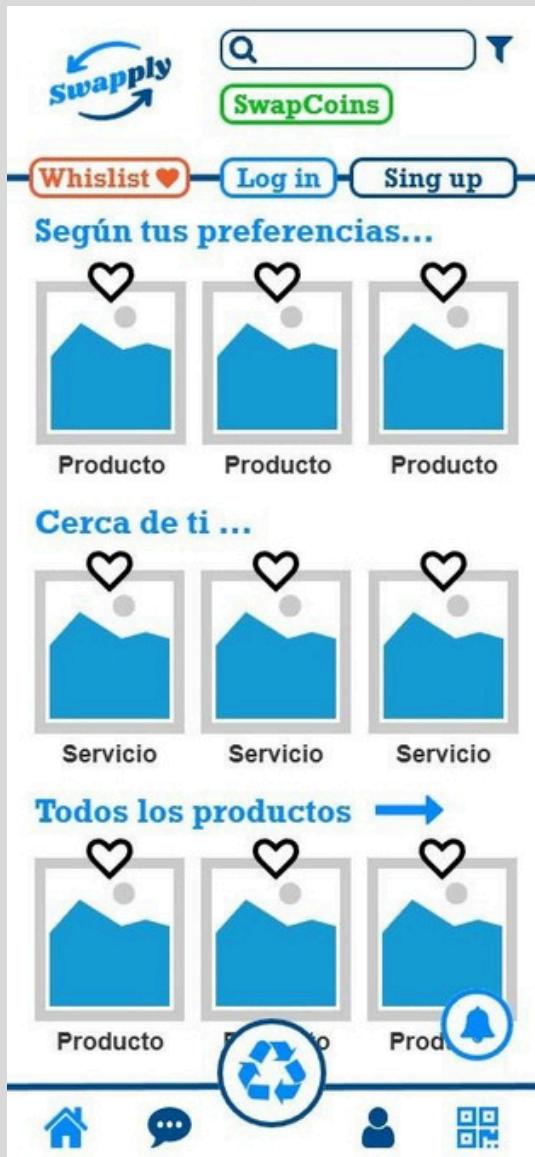
## 6.5. Prototipado de la herramienta por subsistema



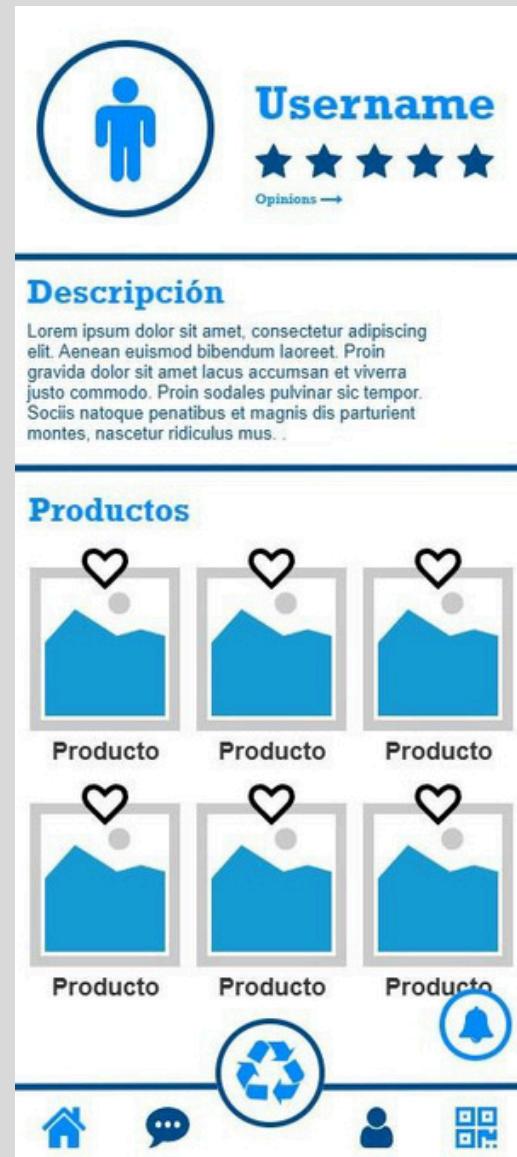
Página de registro



Página de inicio de sesión



Página de inicio



Perfil de usuario



**SwapCoins** 00000

**Product name**

Placeholder text: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo. Proin sodales pulvinar sic tempor. Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. . .

**Tag** **Tag** **Tag** **Tag**

**Swap!**

Página de producto

**Título:**

**Producto** **Servicio**

**SwapCoins:** **Aprox** **Your choice**

**Categoría** **Marca** **Modelo** **Tamaño**  
**Condición** **Talla** **Color**

**Envíos:** **Sí** **No**

**Swap!**

Página para subir productos



## Wishlist ❤

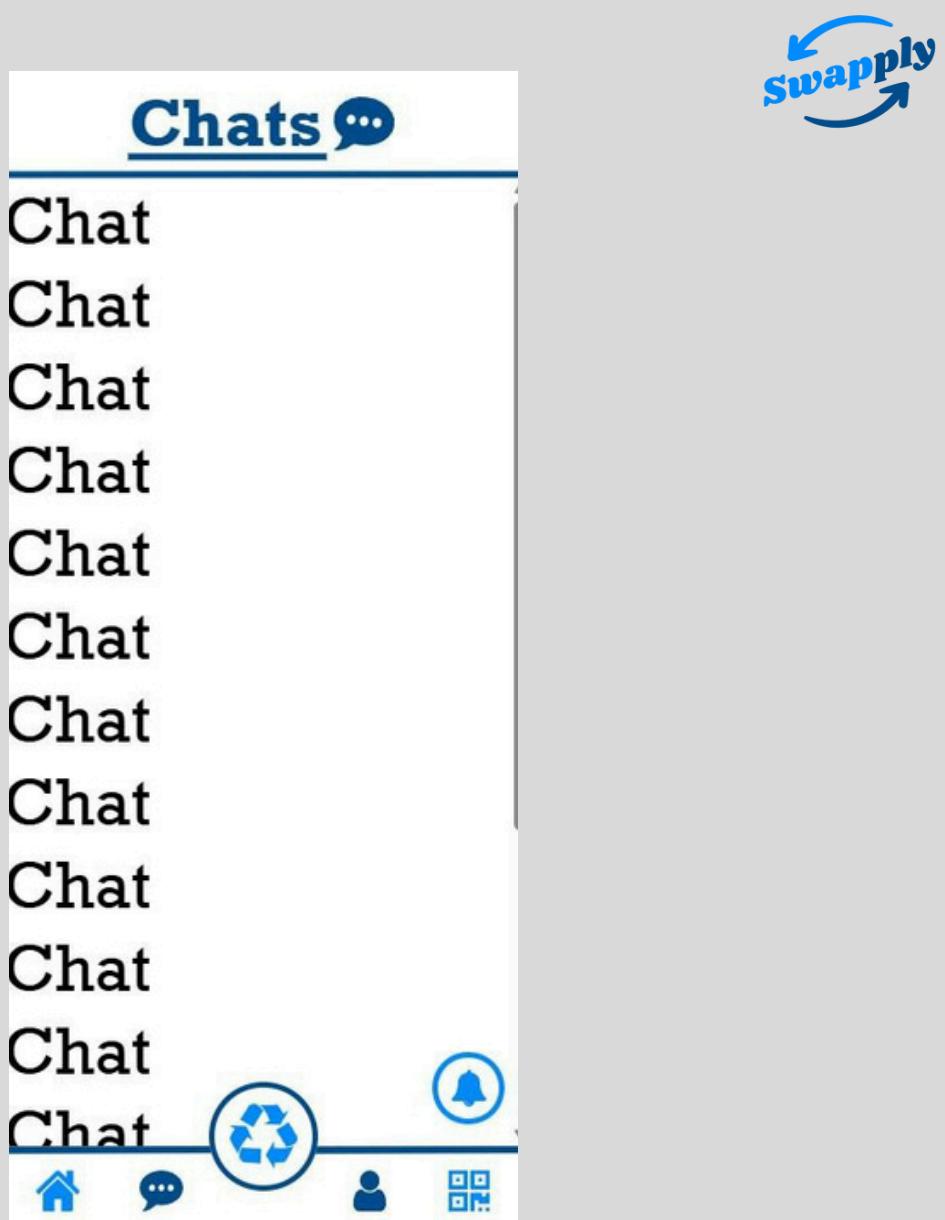
Producto <3  
Produc <3

Página de la wishlist

## Notifications 🎵

Notification  
Notific n

Página de notificaciones



Página de chats

Para más detalles pulse aquí



# 7. Plan de Trabajo



## 7.1. Fases del Proyecto

El proyecto Swapply se estructura en 6 fases principales que se distribuyen desde principios de noviembre de 2025 hasta el 1 de junio de 2026. Cada fase tiene objetivos claros, actividades definidas y entregables específicos.

### **Análisis (Principios de noviembre – mediados de diciembre)**

Durante esta fase se recopila y define toda la información necesaria para construir Swapply

#### **Actividades principales:**

- **Noviembre (hasta el 30 de noviembre):**

- Descripción del proyecto y contexto general.
- Especificación de roles y responsabilidades del equipo.
- Recopilación de requisitos funcionales: registro de usuarios, solicitud de intercambios, chat interno, notificaciones.
- Definición de requisitos no funcionales: seguridad (protección de datos personal), rendimiento, escalabilidad, fiabilidad y usabilidad.
- Gestión de riesgos: análisis de riesgos técnicos como fraudes, sobrecarga del sistema, fallos de pago si existieran servicios premium.

- **Diciembre (hasta el 14 de diciembre):**

- Especificación detallada de casos de uso que describen cómo los usuarios interactuarán con la plataforma.

- **Diciembre (hasta el 29 de diciembre):**

- Cierre del documento de requisitos y especificación funcional completa.

**Entregable:** Documento de requisitos, especificación de casos de uso y análisis de riesgos completados.

**Hito H1 (29 de diciembre):** Documento de requisitos y casos de uso aprobado.



## Diseño (Mediados de diciembre – finales de enero)

Se define la arquitectura técnica y la estructura de Swapply, basándose en los requisitos validados.

### Actividades principales:

- **Diciembre – Enero (hasta el 26 de enero):**

- Elaboración de diagramas UML: diagramas de clases, secuencia y componentes que modelan el sistema.
- Elaboración de diagramas de actividad para los casos de uso más importantes.
- Diseño de la base de datos: definición de tablas para usuarios, productos, categorías, solicitudes de intercambio, valoraciones y mensajes.
- Validación de la arquitectura general: estructura cliente-servidor, uso de APIs, gestión de sesiones, y control de acceso.

- **Enero – Febrero (hasta el 8 de febrero):**

- Diseño de la interfaz de usuario (UI): creación de maquetas para las pantallas principales (inicio, perfil, bandeja de intercambios, chat, notificaciones).
- Definición del flujo de usuario (UX) para asegurar una navegación intuitiva y coherente.

**Entregables:** Arquitectura definida, modelo de datos completo, diagramas UML y prototipos de interfaz.

**Hito H2 (26 de enero):** Arquitectura, modelo de datos y prototipos de interfaz validados.



## Desarrollo (Febrero – mediados de abril)

Se implementa Swapply como producto real, dividiendo el trabajo entre módulos del servidor e interfaz de usuario.

### Actividades principales:

- **Febrero:**

- Implementación de la lógica del lado del servidor (backend):
- Desarrollo de APIs para gestionar usuarios, productos e intercambios.
- Implementación de mecanismos de seguridad.
- Lógica del sistema de intercambio (validación de solicitudes, estados de intercambio, confirmación).

**Nota:** Los "módulos del servidor" son componentes que administran recursos, almacenan archivos y datos, y procesan las solicitudes de los clientes a través de una red.

- **Febrero – Marzo:**

- Implementación de la interfaz de usuario (frontend):
- Desarrollo de las vistas basadas en los prototipos de diseño.
- Integración con las APIs del servidor para mostrar información en tiempo real.
- Implementación de aspectos visuales: estilos, navegación, componentes interactivos.

- **Marzo (hasta el 30 de marzo):**

- Integración de funcionalidades clave:
- Sistema de chat entre usuarios (mensajería en tiempo real).
- Sistema de valoraciones y reputación.
- Notificaciones de intercambios y eventos.
- Integración continua mediante GitHub.

- **Mediados de abril (hasta el 17 de abril):**

- Completar funcionalidades pendientes y ajustes finales.
- Primera integración completa de backend y frontend.
- Verificación de que todos los requisitos críticos se han implementado.

**Entregables:** Primera versión funcional de Swapply con backend y frontend integrados.

**Hito H3 (17 de abril):** Primera versión funcional de Swapply completada.



## Pruebas (Periódicamente)

Se ejecutan pruebas exhaustivas para garantizar la calidad, estabilidad y usabilidad de la plataforma.

Actividades principales:

- **Enero – Enero (hasta el 31 de enero):**

- Pruebas de integración continua tempranas: validación de que los primeros prototipos de subsistemas se comunican correctamente, comprobando interfaces y flujos básicos entre módulos de usuarios, productos y base de datos.

- **Marzo (hasta el 16 de marzo):**

- Segundo ciclo de pruebas de integración: validación de que la lógica del servidor e interfaz de usuario funcionan correctamente en conjunto.

- **Mediados de abril (hasta el 30 de abril):**

- Pruebas unitarias 1 y 2: validación de funciones, clases y componentes de forma aislada en backend (APIs, lógica de negocio) y frontend.

**Nota:** Las pruebas unitarias dependen de que exista código implementado, es decir, de la implementación de la lógica del servidor y de la interfaz, porque se prueban funciones, clases o componentes concretos ya desarrollados.

- **Abril (hasta el 12 de abril):**

- Pruebas de usabilidad 1: evaluación de la facilidad de uso de Swapply con usuarios de prueba (compañeros del equipo o usuarios externos), validando que los flujos principales (registro, solicitar intercambio, chat) son intuitivos.

- **Abril (hasta el 16 de abril):**

- Subsanación de errores tras pruebas: corrección de los errores detectados en las pruebas unitarias y de usabilidad más importantes, priorizando los que afectan a los flujos principales (registro, intercambio, chat, etc.).

- **Mayo (hasta el 2 de mayo):**

- Pruebas de usabilidad 2: validación adicional de flujos secundarios y refinamientos tras correcciones.



- **Mayo (hasta el 11 de mayo):**

- Pruebas de rendimiento para verificar la estabilidad del servidor.
- Corrección final de errores detectados.

**Entregables:** Validación del sistema, garantía de funcionamiento estable, documentación de pruebas.

**Hito H4 (11 de mayo):** Sistema validado y estable, listo para implementación.

## Implementación (Mediados – finales de mayo)

Se despliega Swapply para su uso real en producción.

Actividades principales:

- **Mediados de mayo:**

- Configuración final del servidor.
- Instalación y configuración de la base de datos en producción.
- Configuración de dominio y certificado SSL para comunicación segura.

- **Finales de mayo (hasta el 1 de junio):**

- Despliegue de la aplicación web.
- Publicación de la versión v.1.0 para acceso público.
- Monitorización inicial del funcionamiento en producción.

**Entregables:** Swapply disponible públicamente para los usuarios.

**Hito H5 (1 de junio):** Publicación de Swapply v1.0.



## Mantenimiento(Desde publicación)

Se realiza monitorización y mejora continua de la plataforma después del lanzamiento.

### Actividades principales:

- Corrección de errores que aparecen con el uso real.
- Actualización de funcionalidades existentes basada en feedback de usuarios.
- Implementación de mejoras y nuevas características.
- Mejora del rendimiento del servidor según métricas observadas.
- Monitorización continua de seguridad y protección de datos personales.

**Entregables:** Plataforma funcionando de manera estable y mejoras implementadas con el tiempo.

### Definiciones Clave:

**API (Interfaz de Programación de Aplicaciones):** intermediario de software que conecta información y funcionalidades con los requerimientos de la aplicación.

**Gestión de sesiones:** proceso de administrar la interacción continua de un usuario con una aplicación para mantener su estado y datos seguros y eficientes.

**UX:** Camino que sigue un usuario a través de una plataforma digital para completar una tarea específica, como registrarse o comprar un producto.

**Backend:** lógica de negocio de Swapply (gestión de usuarios, seguridad, anuncios, intercambios, reputación, notificaciones) y el acceso a la base de datos.

**Frontend:** Interfaz web que ven los usuarios (pantallas de inicio, crear anuncio, perfil, bandeja de intercambios).

**SSL:** protocolo de seguridad que cifra la conexión entre un servidor web y un navegador para proteger la privacidad y la integridad de los datos transmitidos, como información personal o financiera.

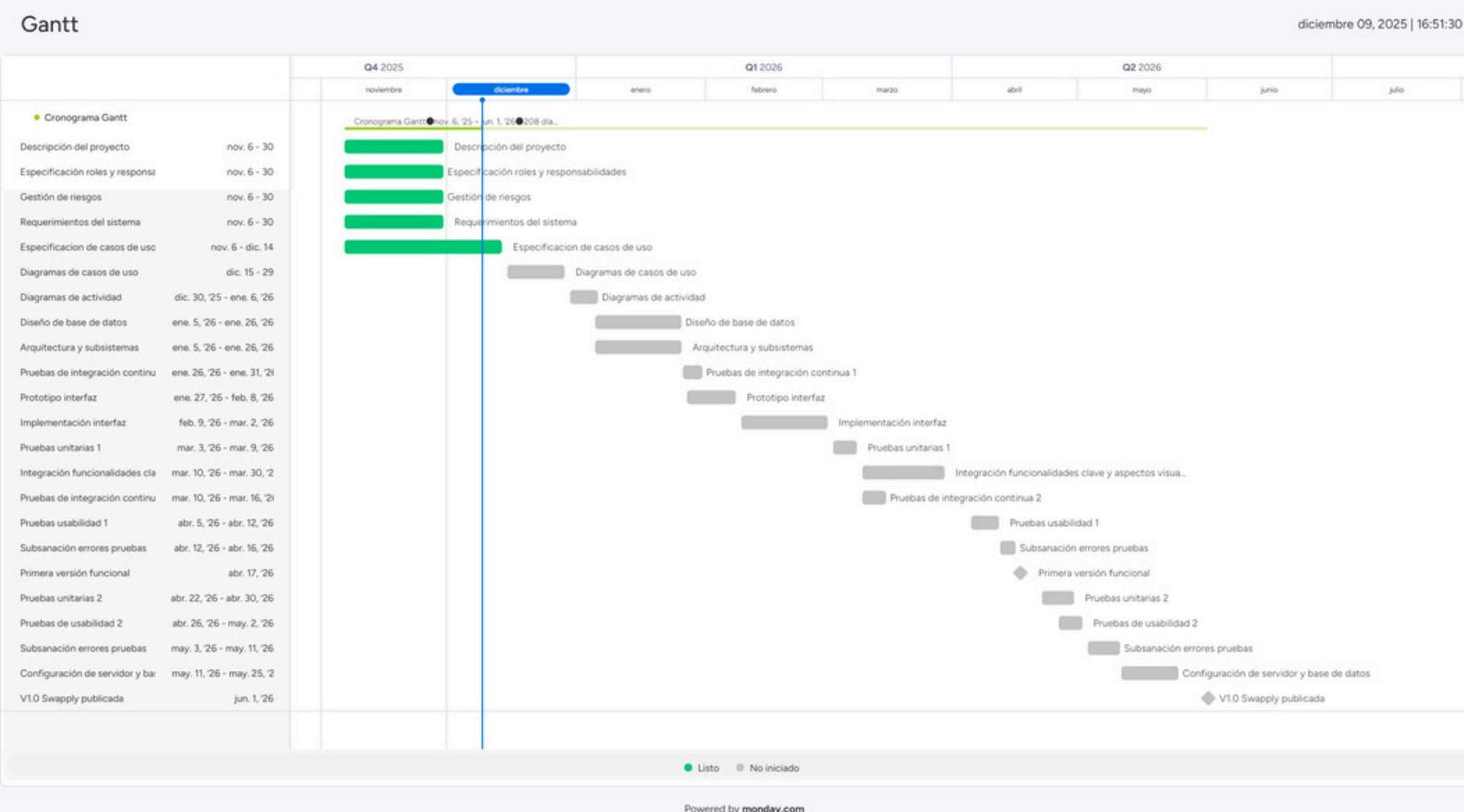


## 7.2. Cronograma

El cronograma Gantt refleja la distribución temporal de todas las fases y tareas del proyecto desde el inicio de noviembre de 2025 hasta el 1 de junio de 2026, un período de aproximadamente 6 meses.

### Principales características del Gantt:

- Las barras de color representan el rango temporal de cada actividad.
- Las dependencias se indican mediante flechas o líneas de conexión entre tareas.
- Los hitos (milestones) se marcan como puntos clave de decisión o entrega.
- Los solapamientos entre fases indican trabajo paralelo (ej.: diseño y primeras pruebas de integración conceptual, desarrollo e integración continua).



Hacer zoom

Para ver el cronograma con más detalle pulse aquí



# 7.3. Dependencias e Hitos Clave

## DEPENDENCIAS DEL PROYECTO

Las siguientes dependencias aseguran una secuencia lógica y ordenada del trabajo:

### 1. Diseño depende de Análisis:

El diseño de arquitectura, base de datos e interfaz requiere que los requisitos y casos de uso estén completamente definidos y aprobados (Hito H1).

### 2. Desarrollo depende de Diseño:

La implementación de los módulos del servidor y la interfaz depende de que la arquitectura, y los prototipos de interfaz estén validados (Hito H2). Los desarrolladores usan estos documentos como guía.

### 3. Pruebas de integración tempranas dependen de arquitectura:

Las primeras pruebas de integración (enero) validan que los subsistemas diseñados pueden comunicarse, lo que depende de tener la arquitectura definida.

### 4. Pruebas unitarias dependen de desarrollo:

Las pruebas unitarias requieren código implementado en los módulos del servidor e interfaz. No pueden ejecutarse sin que exista lógica desarrollada y componentes creados.

### 5. Pruebas de integración de segundo ciclo dependen de pruebas unitarias:

Una vez validados los módulos de forma individual, se prueban en conjunto para asegurar que interactúan correctamente.

### 6. Pruebas de usabilidad dependen de interfaz implementada:

Las pruebas de usabilidad requieren que la interfaz esté completa y funcional (Hito H3) para ser evaluada por los usuarios.

### 7. Implementación depende de pruebas aprobadas:

Solo se despliega a producción cuando se han superado las pruebas y el sistema se ha validado como estable (Hito H4).

### 8. Mantenimiento depende de publicación:

El mantenimiento comienza después de que la v1.0 está disponible públicamente (Hito H5).



## Camino crítico:

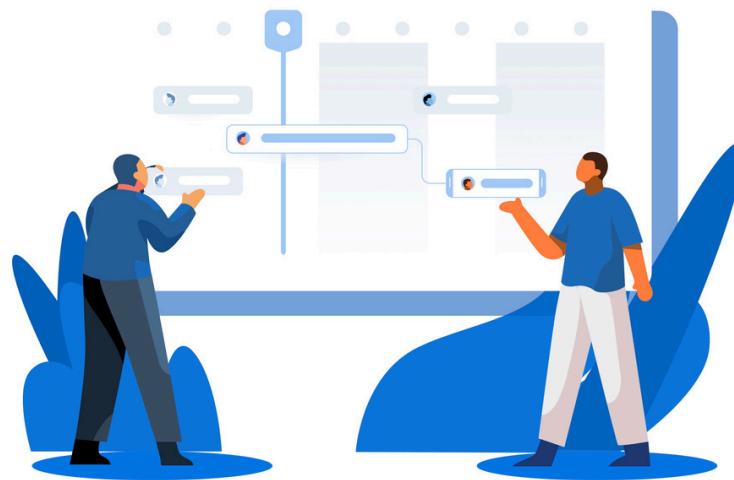
- El camino crítico es una herramienta y concepto utilizado en la gestión de proyectos para identificar la secuencia de tareas o actividades que determinan la duración total del proyecto. Esto permite tomar decisiones informadas para evitar retrasos en la finalización del proyecto.
- Las actividades en el camino crítico no tienen **holgura**, lo que significa que no pueden retrasarse sin afectar a la duración total del proyecto.
- En nuestro proyecto, si alguna de estas tareas se retrasa, el proyecto entero se desplaza en el tiempo:
  - **Especificación de casos de uso y diagramas de casos de uso / actividad** (sin esto no se podría diseñar BD ni arquitectura).
  - **Diseño de base de datos y arquitectura y subsistemas** (si se retrasan, empieza más tarde todo el desarrollo, APIs, interfaz).
  - **Prototipo de interfaz e implementación de la interfaz** (sin interfaz lista no se podrían hacer pruebas de usabilidad ni integrar bien las funcionalidades clave).
  - **Implementación de la lógica del servidor e integración de funcionalidades clave** (chat, reputación, notificaciones): si no están, no hay primera versión funcional.
  - **Pruebas unitarias e integración**: si se alargan, se retrasa la detección y corrección de errores y, por tanto, la v1.0.
  - **Pruebas de usabilidad + subsanación de errores antes de la v1 funcional**: sin cerrar estos puntos, no se podría declarar el Hito H3 ni pasar a la fase final.
  - **Configuración de servidor y base de datos en producción y publicación Swapply v1.0**: cualquier retraso aquí mueve directamente la fecha de publicación. (Hito H5)



## HITOS PRINCIPALES ("MILESTONES")

Se definen 5 hitos principales que representan puntos de decisión críticos y entregas importantes:

|    | Fecha             | Descripción  |
|----|-------------------|--|
| H1 | 29 Diciembre 2025 | Documento de requisitos y casos de uso aprobado                  |
| H2 | 26 Enero 2026     | Arquitectura, modelo de datos y prototipos de interfaz validados |
| H3 | 17 Abril 2026     | Primera versión funcional de Swapply completada                  |
| H4 | 11 Mayo 2026      | Sistema validado y estable, pruebas superadas                    |
| H5 | 1 Junio 2026      | Publicación de Swapply v1.0                                      |



# 8. Recursos del Proyecto



## 8.1. Equipo de trabajo

Para poder llevar a cabo el desarrollo de la aplicación Swapply de una manera organizada y profesional, hemos asignado diferentes roles específicos a cada uno de los 11 miembros del equipo de trabajo. Esta estructura nos permite cubrir todas las fases del proyecto, desde el análisis inicial hasta el despliegue final, asegurando el cumplimiento de los plazos y la calidad del producto.

El equipo está estructurado en los siguientes roles clave:

### 1. Project Manager (CEO) y Gestor de Producto

- **Responsable:** Sofía Mato de la Rocha
- **Responsabilidades:** Planificar tareas y controlar el calendario (Hitos), coordinar a todo el equipo asegurando el cumplimiento de la metodología, y gestionar los problemas que puedan surgir (cambios de requisitos o retrasos). Además, define la visión estratégica y las prioridades de producto.

### 2. Scrum Master y Responsable de Calidad (QA)

- **Responsables:** Carlos Belmonte Arias, Paula Bonilla Fernández
- **Responsabilidades:** Scrum Master: Liderar las reuniones (Daily Stand-ups), asegurar que el equipo siga la metodología ágil, y eliminar los impedimentos que afecten al desarrollo. QA: Realizar pruebas continuas (bugs), verificar el cumplimiento de todos los requisitos funcionales y no funcionales, y revisar la usabilidad de la aplicación.

### 3. Desarrollador Principal (Arquitectura y Datos)

- **Responsables:** Pablo Martín Gutiérrez, Carla Cillán Barrajón
- **Responsabilidades:** Diseñar la base de datos (PostgreSQL/MongoDB) para almacenar usuarios, productos y el historial de trueques. Programar la lógica del negocio en el servidor (backend), incluyendo la validación de intercambios y el procesamiento de SwapCoins. Asegurar la conexión segura (SSL) y la encriptación de los datos personales.



#### 4. Desarrollador de la Interfaz Web y Móvil (Frontend y UX)

- **Responsables:** Diego Pallarés López, Carlota Cambor Parrondo, Samuel Fernández Gil, Ricardo Campo Acedo
- **Responsabilidades:** Crear las pantallas de la aplicación (registro, inicio, perfil, etc.) para que sean intuitivas y se ajusten a los diseños previos (Hito H2). Programar la funcionalidad visual (frontend) asegurando que la aplicación sea responsive en diferentes dispositivos. Implementar funcionalidades que utilicen el hardware del móvil (cámara, códigos QR).

#### 5. Especialista en Algoritmos Avanzados y Rendimiento

- **Responsable:** Elena Hidalgo Maqueda, Colomba Andre
- **Responsabilidades:** Diseñar e implementar los algoritmos más complejos, como el Motor de Recomendación y la lógica del Trueque Circular (RF-A01). Liderar las pruebas de rendimiento y carga (RNF-P03) para garantizar la estabilidad del servidor con 10.000 usuarios concurrentes.

## 8.2. Recursos materiales y tecnológicos

En esta sección se describen los recursos materiales y tecnológicos necesarios para el desarrollo, despliegue y operación de Swapply, una plataforma de intercambio de productos y servicios accesible vía web y aplicación móvil.

### • INFRAESTRUCTURA HARDWARE Y SERVIDORES

Dado que Swapply es una plataforma online con potencial crecimiento en número de usuarios, se opta por una infraestructura basada en servicios en la nube en lugar de servidores físicos propios. Esto reduce la inversión inicial y facilita la posibilidad de hacer crecer la empresa.

#### → SERVIDORES LÓGICOS (EN LA NUBE)

Se contemplan, como mínimo, los siguientes servidores lógicos:

- **Servidor de aplicaciones**
  - a. Función: alojar la API que gestiona la lógica de negocio (registro de usuarios, publicación de anuncios, sistema de intercambio, mensajería, etc.).
  - b. Requisitos aproximados (fase inicial): 2 vCPU, 4 - 8 GB RAM, 50 - 100 GB de almacenamiento, sistema operativo distribución Linux



- **Servidor web**
  - a. Función: servir la aplicación web
  - b. Puede estar integrado con el servidor de aplicaciones o separado detrás de un balanceador de carga.
- **Servidor de base de datos**
  - a. Función: almacenar de forma persistente la información de usuarios, productos/servicios publicados, valoraciones, mensajes, transacciones de intercambio, etc.
  - b. Motor de base de datos: Opción relacional y tamaño inicial 100 - 200 GB escalable.
- **Servidor de ficheros / almacenamiento de objetos**
  - a. Función: almacenar imágenes de productos, fotos de perfil y otros documentos.
  - b. Se recomienda un servicio de almacenamiento de objetos en la nube.
- **Servidor de preproducción / pruebas**
  - a. Réplica reducida del entorno de producción donde se prueban nuevas versiones antes de su despliegue.
  - b. Es esencial para garantizar calidad y estabilidad.

## → INFRAESTRUCTURA DE RED Y COMUNICACIONES

Aunque el proveedor cloud gestiona el hardware físico, es necesario configurar:

- Balanceador de carga para repartir tráfico entre instancias del servidor de aplicaciones cuando la carga aumente.
- VPN o redes privadas virtuales entre los distintos servidores para aumentar la seguridad.
- Firewall y Security Groups para restringir el acceso a puertos y servicios.
- CDN (Content Delivery Network) para acelerar la entrega de recursos estáticos (imágenes, scripts) y mejorar el rendimiento global.

## → EQUIPOS DE DESARROLLO

Para el equipo de desarrollo y operación se requieren:

- Ordenadores personales (portátiles o sobremesa) con:
  - CPU de al menos 4 núcleos.
  - 16 GB de RAM recomendados.
  - 512 GB SSD.
  - Dispositivos móviles (Android e iOS) de gama media para pruebas de la app.
  - Conectividad a Internet estable de alta velocidad.



## • RECURSOS SOFTWARE PARA EL DESARROLLO

Estos recursos se utilizan internamente por el equipo para construir y mantener Swapply.

### → LENGUAJES Y FRAMEWORKS

- Backend
  - a. Lenguaje: por ejemplo, Java (Spring Boot) o Python
  - b. Framework: se seleccionará uno que facilite la construcción de APIs REST y el acceso a la base de datos.
- Frontend web
- Aplicación móvil

### → GESTIÓN DE CÓDIGO Y COLABORACIÓN

- Sistema de control de versiones: Git.
- Plataforma de repositorios: GitHub, GitLab o Bitbucket.
- Herramientas de gestión de proyectos: Jira, Trello o GitHub Projects para planificación de sprints, tareas y seguimiento de incidencias.
- Comunicación interna: Slack, Microsoft Teams o similar.

### → INTEGRACIÓN CONTINUA Y DESPLIEGUE CONTINUO (CI/CD)

- Servidor o servicio de CI/CD: GitHub Actions, GitLab CI, Jenkins o similar.
- Pipelines configurados para:
  - a. Ejecutar tests automatizados.
  - b. Construir imágenes de Docker si fuese necesario.
  - c. Desplegar automáticamente en entornos de pruebas y producción.

### → BASES DE DATOS Y HERRAMIENTAS ASOCIADAS

- Gestor de base de datos: PostgreSQL/MySQL.
- Herramientas de administración y diseño: PgAdmin, MySQL Workbench o DBeaver.
- Herramientas de migración de esquemas: Flyway, Liquibase u otras integradas en el framework backend.

## • RECURSOS SOFTWARE PARA LA OPERACIÓN Y EL MANTENIMIENTO

Una vez en producción, Swapply requiere un conjunto de herramientas para monitorizar, asegurar y optimizar la plataforma.



### → MONITORIZACIÓN Y LOGGING

- Sistemas de monitorización: Prometheus, Grafana, CloudWatch u otros equivalentes según el proveedor.
- Gestión de logs: ELK Stack (Elasticsearch, Logstash, Kibana) o servicios de logging gestionados.
- Alertas: Integración con email, Slack o SMS cuando se detecten errores o caídas.

### → SEGURIDAD

- Certificados SSL/TLS para cifrado de comunicaciones .
- Web Application Firewall para proteger la plataforma de ataques comunes.
- Herramientas de análisis de vulnerabilidades: Dependabot, Snyk, OWASP ZAP o similares.

### → ANALÍTICA Y MÉTRICAS DE USO

- Herramientas de analítica web: Google Analytics, Matomo o similar para medir visitas, comportamiento, retención.
- Métricas de negocio: Número de intercambios realizados, productos publicados, usuarios activos, etc., integrados en paneles internos.

## • LICENCIAS Y SUSCRIPCIONES

Para el funcionamiento de Swapply será necesario gestionar los siguientes tipos de licencias y suscripciones:

### → LICENCIAS DE SOFTWARE

- Software libre / open source.
- Muchos componentes (frameworks, librerías, motores de base de datos) se basan en licencias como MIT, Apache 2.0, GPL, etc.
- Es necesario llevar un control de estas licencias para garantizar el cumplimiento (por ejemplo, mención de copyright cuando sea necesario).
- Software propietario.
- Posibles licencias de IDEs de pago si se utilizan versiones comerciales.
- Herramientas de gestión de proyectos o comunicación en planes de pago si se requiere.



## → SUSCRIPCIONES A SERVICIOS EN LA NUBE

- Proveedor de infraestructura cloud.
- Coste asociado a: Máquinas virtuales (servidores de aplicaciones y base de datos), almacenamiento de objetos (imágenes de productos), tráfico de red y balanceador de carga, servicios gestionados (monitorización, logging, etc.).
- Servicios de terceros: Dependiendo de las funcionalidades de Swapply, pueden emplearse:
  - a. Servicios de envío de correos transaccionales (por ejemplo, para confirmación de registro, notificaciones de intercambio).
  - b. Servicios de SMS/push notifications.
  - c. Gateway de pago.

## → DOMINIOS Y CERTIFICADOS

- Registro de dominio: dominio principal del proyecto (por ejemplo, swapply.com o swapply.es).
- Certificados digitales: certificados SSL/TLS emitidos por una autoridad de certificación

En resumen, en conjunto, los recursos materiales y tecnológicos de Swapply se basan en:

- Una infraestructura cloud escalable (servidores de aplicaciones, base de datos, almacenamiento de ficheros, balanceador de carga).
- Equipos de desarrollo adecuados (ordenadores, dispositivos móviles de prueba).
- Un stack software moderno (frameworks web y móviles, bases de datos, herramientas de CI/CD, monitorización y seguridad).
- Licencias y suscripciones necesarias (software de desarrollo, servicios cloud, registro de dominio y certificados).

Esta combinación permite desarrollar y operar Swapply de forma eficiente, segura y con capacidad de crecimiento según aumente el número de usuarios y el volumen de intercambios.

## 8.3. Presupuesto Estimado



A continuación se presenta una estimación de los costes asociados al desarrollo y primer año de operación de la plataforma Swapply. Las cifras son aproximadas y se basan en supuestos razonables para un entorno profesional en España.

### • Hipótesis de estimación

Para poder calcular el presupuesto, se asumen las siguientes hipótesis, reflejando el compromiso de remunerar a los 11 miembros del equipo:

- Duración de la fase de desarrollo inicial : 6 meses.
- Dedicación del Equipo: Los sueldos se basan en dedicación a tiempo completo (100%) o a tiempo parcial (50% o 75%), según la responsabilidad.
- Coste Laboral: Las cifras reflejan el **Coste Bruto Total para la Empresa** (salario bruto anual del empleado + cotizaciones a la Seguridad Social y otros costes a cargo de la empresa).
- Se incluye el primer año de operación (infraestructura cloud, licencias...).
- Todos los importes están expresados en euros (€).

### • Coste de Personal

| Perfil                           | Cantidad | Dedicación | Duración | Coste mensual estimado   | Coste total aproximado |
|----------------------------------|----------|------------|----------|--------------------------|------------------------|
| Jefe/a de proyecto               | 1        | 75%        | 6 meses  | 3.375€<br>(4.500€ x 75%) | 20.250€                |
| Scrum Mater/QA                   | 2        | 75%        | 6 meses  | 2.625€<br>(3.500€ x 75%) | 31.500€                |
| Desarrollador/a backend          | 1        | 100%       | 6 meses  | 4.000€                   | 24.000€                |
|                                  | 1        | 75%        | 6 meses  | 3.000€<br>(4.000€ x 75%) | 18.000€                |
| Desarrollador Fronted/ UX        | 4        | 50%        | 6 meses  | 1.750€<br>(3.500€ x 50%) | 42.000€                |
| Especialista en Algoritmos       | 2        | 100%       | 6 meses  | 4.500€                   | 54.000€                |
| Coste Total estimado de Personal | 11       |            |          |                          | 189.750€               |



- **Costes de infraestructura tecnológica**

Incluyen servidores, almacenamiento y servicios asociados en la nube para desarrollo y producción durante el primer año.

| Concepto   | Detalle           | Importe anual aproximado |
|--|-------------------|--------------------------|
| Servidores de aplicaciones y base de datos (cloud) | 300€/mes          | 3.600€                   |
| Almacenamiento de ficheros (backups)               | 50€/mes           | 600€                     |
| Ancho de banda, balanceador de carga, otros        | Estimación global | 1.000€                   |

**Coste total estimado de infraestructura cloud (primer año) :**

$$3.600\text{€} + 600\text{€} + 1.000\text{€} = 5.200\text{€}$$

- **Costes de Software, licencias y servicios**

| Concepto   | Detalle                           | Importe aproximado |
|--|-----------------------------------|--------------------|
| Licencias de IDE o herramientas de desarrollo (opcional) | 2 licencias de IDE de pago        | 400€               |
| Herramientas de gestión de proyectos                     | 50€/mes                           | 600€               |
| Herramientas de comunicación interna                     | Versión básica o plan económico   | 200€               |
| Servicios de email transaccional/notificaciones          | Alta y consumo básico             | 300€               |
| Dominio y certificados SSL/TLS                           | Registro de dominio + certificado | 150€               |

**Coste total estimado de software y servicios (primer año) :**

$$400\text{€} + 600\text{€} + 200\text{€} + 300\text{€} + 150\text{€} = 1650\text{€}$$



- **Costes de Equipamiento**

En caso de que el equipo no disponga de equipos previos, se podría contemplar una partida para equipamiento mínimo:

| Concepto                          | Cantidad | Coste unitario aprox. | Coste total aprox. |
|-----------------------------------|----------|-----------------------|--------------------|
| Portátiles para desarrollo        | 3        | 2.000€                | 6.000€             |
| Dispositivos móviles para pruebas | 2        | 500€                  | 1.000€             |

**Coste total estimado de equipamiento: 6.000€ + 1.000€ = 7.000€**

- **Resumen de Presupuesto**

- Costes de personal (desarrollo, 6 meses) : 189.750€ aprox.
- Infraestructura cloud (primer año) : 5.200€ aprox.
- Software, licencias y servicios (primer año) : 1.650€ aprox.
- Equipamiento (si es necesario) : 7.000€ aprox.
- Total estimado (sin contingencia) : 203.600€

- **Margen de Contingencia**

Se añade un pequeño margen al presupuesto, teniendo en cuenta la posibilidad de que suceda algún imprevisto. Si aplicamos un 10% sobre el total estimado : 10% de 203.600€ = 20.360€

Por tanto el presupuesto final aproximado que tomaremos será : 223.960€



# 9. Gestión de Proyecto



## 9.1. Metodología

Para el desarrollo de la plataforma Swapply, se ha adoptado una metodología de trabajo híbrida. Esta combina el rigor en la documentación y arquitectura de un modelo de Ciclo de Vida de Desarrollo de Software (SDLC) Secuencial con la adaptabilidad y eficiencia en la ejecución diaria de la metodología Scrum/Ágil.

Esta combinación asegura la solidez técnica del producto sin perder la capacidad de respuesta y la gestión organizada del equipo de 11 personas.

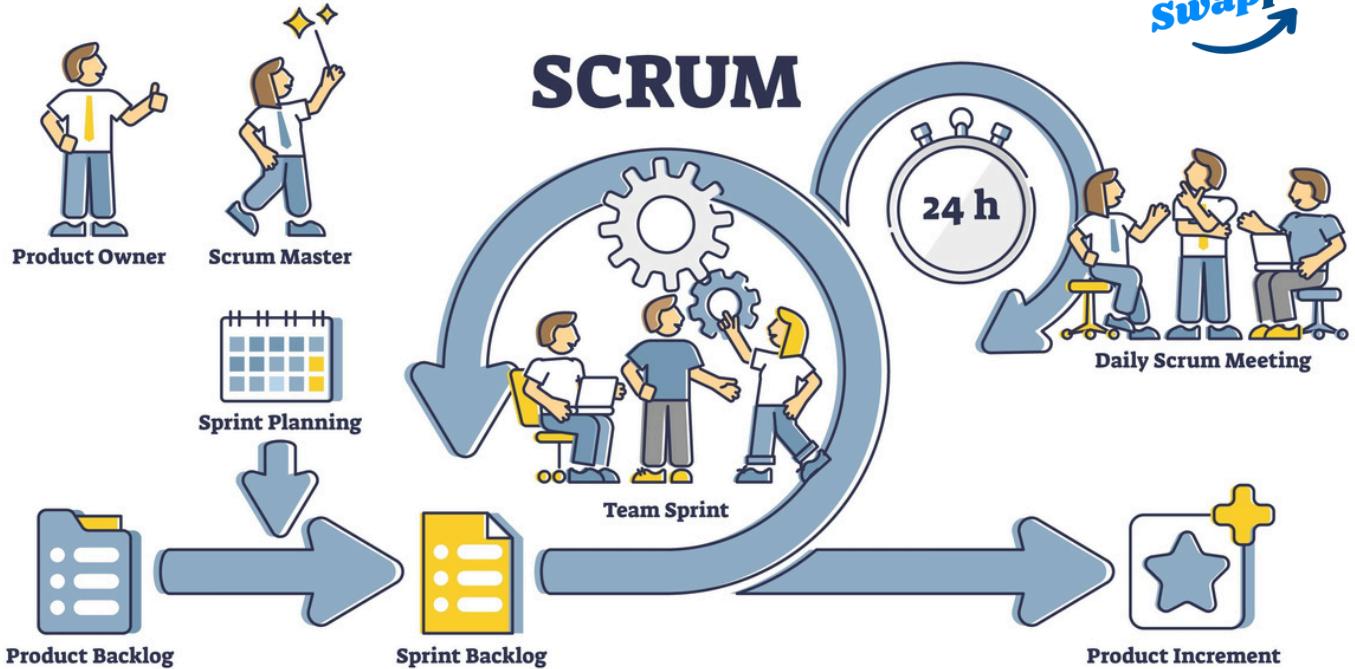
### Estructura Secuencial del Proyecto (SDLC)

El proyecto respeta una estructura de ingeniería secuencial para garantizar que las bases conceptuales y técnicas (Análisis y Diseño) sean validadas antes de la implementación masiva (coherente con el Punto 7.1). El desarrollo avanza a través de las siguientes fases con entregables (Hitos) claros:

1. **Análisis:** Definición exhaustiva de requisitos funcionales y no funcionales, identificación de actores y análisis de riesgos técnicos (Hito H1).
1. **Diseño:** Elaboración de la arquitectura del sistema, modelado de la base de datos, diagramas UML y diseño de la experiencia de usuario (UX/UI) (Hito H2).
1. **Desarrollo:** Implementación iterativa del backend y frontend. Esta fase es gestionada mediante Sprints.
1. **Pruebas:** Ejecución de pruebas unitarias, de integración, de usabilidad y de rendimiento para asegurar la calidad antes del despliegue (Hito H4).
1. **Implementación y Mantenimiento:** Despliegue en el entorno real de producción y monitorización continua (Hito H5).

### Gestión Operativa Ágil (Scrum)

La fase de Desarrollo, Pruebas y parte del Mantenimiento se gestionan mediante Scrum para maximizar la eficiencia y la colaboración de un equipo grande.



| Elemento Scrum                   | Descripción y Responsabilidad  |
|----------------------------------|--|
| <b>Sprints</b>                   | Ciclos de trabajo fijos de dos semanas para la entrega incremental de funcionalidades específicas y probadas.  |
| <b>Backlog del Producto</b>      | Lista priorizada de funcionalidades y requisitos (Historias de Usuario) que mantiene el Project Manager (Gestor de Producto).  |
| <b>Daily Scrum</b>               | Reunión diaria de 15 minutos (Daily Stand-ups) liderada por el Scrum Master para revisar el progreso, sincronizar tareas y eliminar impedimentos.  |
| <b>Integración Continua (CI)</b> | El código se integra y prueba varias veces al día a través de repositorios como GitHub, asegurando que los Desarrolladores trabajen siempre sobre una base estable.                          |
| <b>Herramienta de Gestión</b>    | Se utiliza GitHub Projects o una herramienta similar (Jira/Trello) para mantener un tablero Kanban digital del Sprint Backlog, asegurando total transparencia sobre el estado de las tareas. |

## 9.2. Gestión de riesgos



La gestión de riesgos en Swapply se aborda desde una perspectiva preventiva, identificando amenazas potenciales tanto a nivel técnico como de gestión del proyecto. A continuación, se detallan los riesgos principales detectados durante la fase de análisis y las estrategias de mitigación implementadas en el diseño del sistema.

### Riesgos Técnicos

- **Riesgo de Confianza y Fraude entre Usuarios:** Dado que Swapply se basa en el intercambio entre pares, existe el riesgo inherente de usuarios malintencionados o perfiles falsos.
  - *Estrategia de Mitigación:* Se ha definido un Subsistema de Identidad y Reputación robusto. A nivel funcional, se impone el Registro Institucional para garantizar la identidad real de los alumnos. Además, el sistema hace obligatoria la valoración mutua tras cada transacción para construir un historial de reputación transparente.
- **Riesgo de Rendimiento y Sobrecarga (Cuellos de Botella):** El cálculo de cadenas de trueque (A->B->C) implica una alta complejidad computacional que podría bloquear el sistema si aumenta el número de usuarios.
  - *Estrategia de Mitigación:* La arquitectura del sistema ha sido diseñada específicamente para contrarrestar esto mediante un enfoque Orientado a Servicios (SOA) y Dirigido por Eventos. El uso de un bus de eventos asíncrono desacopla los procesos, evitando que los cálculos complejos bloqueen la experiencia del usuario.
- **Riesgo de Integridad en el Matching Multilateral:** La complejidad de relacionar deseos y pertenencias entre múltiples usuarios podría generar errores en los datos o "ciclos infinitos".
  - *Estrategia de Mitigación:* Se ha adoptado un modelo de persistencia híbrido. Mientras los datos transaccionales residen en una base de datos relacional estándar, la lógica de emparejamiento se delega a una Base de Datos Orientada a Grafos, optimizada para manejar estas relaciones complejas de forma eficiente y segura.



## Riesgos del Proyecto

- **Desviación del Alcance (Scope Creep):** Existe el riesgo de que la incorporación continua de nuevas ideas o requisitos a lo largo del desarrollo retrase la entrega final del prototipo (Hito H5).
  - *Estrategia de Mitigación:* Se aplicará rigurosamente el procedimiento de Control de Cambios definido en el apartado 9.4. El uso de la metodología Scrum/Kanban (Punto 9.1) permite al Project Manager visualizar claramente si el trabajo en curso excede la capacidad del equipo, facilitando la toma de decisiones para rechazar o posponer nuevas funcionalidades.
- **Pérdida de Datos o Trabajo:** Debido a la participación de 11 integrantes, existe un riesgo elevado de fallos en la coordinación del código o la pérdida de código fuente.
  - *Estrategia de Mitigación:* Se impone el uso obligatorio de Integración Continua y control de versiones mediante GitHub. Esto asegura que siempre exista una versión estable y centralizada del proyecto disponible en la nube, y automatiza la ejecución de pruebas cada vez que se sube un nuevo código.

## Riesgos Económicos

- **Fallos o Inconsistencias en Pagos Mixtos (SwapCoins):** Existe el riesgo de que una transacción falle (por ejemplo, se descuenten monedas, pero no se registre el trueque).
  - *Estrategia de Mitigación:* Implementación de transacciones ACID en el Subsistema Económico para garantizar la atomicidad. Si una parte del intercambio falla (ej. stock no disponible), se revierten automáticamente todos los cambios financieros (rollback), asegurando que ningún usuario pierda sus monedas.

## 9.3. Control de calidad

El objetivo principal de la estrategia de control de calidad es garantizar que el prototipo desarrollado cumple con los requisitos funcionales definidos y transmite de forma sólida la propuesta de valor de Swapply. Para ello, se han establecido cuatro actividades de aseguramiento de la calidad (QA).



## Verificación y Validación de Casos de Uso

Se realizará una revisión técnica exhaustiva de todos los Casos de Uso definidos en la Arquitectura (Sección 6.3) para asegurar su trazabilidad y completitud.

- **Consistencia Lógica:** Se verificará que no existan contradicciones en la lógica de negocio, especialmente en procesos complejos como el Matching y la Gestión de Transacciones.
- **Cobertura de Escenarios:** Se comprobará que los casos de uso cubren tanto el "camino feliz" (flujo ideal) como los flujos alternativos y de excepción (ej. gestión de errores al publicar un producto o fallos en la validación local).
- **Alineación con Requisitos:** Se validará que cada caso de uso implementado responde fielmente a los Requisitos Funcionales (RF) establecidos en la fase de análisis.

## Pruebas de Usabilidad (UX)

Dada la naturaleza social de la aplicación, es crítico evaluar la interacción del usuario con el sistema. Se ejecutarán pruebas básicas con usuarios externos al equipo de desarrollo para evaluar:

- **Claridad en Tareas Críticas:** Se medirá la facilidad con la que un usuario nuevo puede completar tareas clave: Publicar un producto, Configurar filtros de búsqueda y Proponer un intercambio.
- **Comprendión del Modelo:** Se verificará si el usuario entiende conceptos propios de Swapply, como la distinción del "Perfil Dual" (Bienes vs. Servicios) o el funcionamiento de las valoraciones.

**Auditoría de Flujos de Navegación.** Se controlará la integridad del flujo de navegación entre pantallas para asegurar que la experiencia sea intuitiva y sin callejones sin salida.

- **Transiciones:** Verificación del paso correcto entre estados (ej. de Lista de Productos a Detalle y de ahí a Solicitud de Match).
- **Accesibilidad de Funciones:** Asegurar que las funcionalidades transversales, como el acceso a la Wallet de SwapCoins o al perfil de usuario, sean accesibles desde los puntos adecuados de la navegación sin interrumpir el flujo principal de trueque.



## Auditoría Estética y de Uniformidad (UI)

Además, se realizará una revisión del estilo visual para garantizar una presentación profesional, coherente y atractiva de cara a posibles inversores.

- **Coherencia Visual:** Se revisará que la paleta de colores, tipografías e iconografía se mantengan uniformes en todos los módulos (Gestión de Usuarios, Inventario, Notificaciones), reforzando la identidad de marca de Swapply.
- **Fidelidad al Prototipo:** Se comprobará que la implementación final respeta los diseños de interfaz (UI) aprobados en la fase de diseño.

## Pruebas Técnicas y de Rendimiento

Finalmente, más allá de la calidad funcional, se asegurará la robustez del código mediante la ejecución de las pruebas definidas en el Plan de Trabajo:

- **Pruebas Unitarias y de Integración:** Se verificará el correcto funcionamiento de los componentes críticos del backend, asegurando que la comunicación entre los subsistemas (ej. del Bus de Eventos a la Base de Datos de Grafos) no genera errores silenciosos.
- **Pruebas de Carga:** Para mitigar el "Riesgo de Rendimiento" identificado en el apartado 9.2, se simularán escenarios de alta concurrencia en el algoritmo de matching para garantizar que el sistema responde adecuadamente y no se bloquea ante múltiples cadenas de trueque simultáneas.

## 9.4. Gestión de Cambios

Dada la naturaleza evolutiva del desarrollo de software y el tamaño del equipo (11 integrantes), es previsible que surjan nuevos requerimientos o modificaciones al alcance inicial durante la construcción del prototipo. Para mitigar el riesgo de Scope Creep (alcance no controlado) y garantizar la coherencia arquitectónica de la plataforma, se ha definido el siguiente **Procedimiento Formal de Control de Cambios**:



**1. Identificación y Propuesta.** Cualquier modificación sugerida (ya sea una nueva funcionalidad, una alteración en la base de datos o un ajuste en la interfaz) debe ser formalizada antes de su implementación.

- **Solicitud:** El cambio se debe proponer claramente, identificando si se trata de una corrección de error (bugfix), una mejora (enhancement) o una nueva característica.
- **Contexto:** Se debe especificar qué componente del sistema se ve afectado (ej. Subsistema de Matching, Módulo de Autenticación).

**2. Análisis de Impacto y Viabilidad.** Antes de aprobar cualquier solicitud, el equipo evaluará las consecuencias de introducir el cambio en el ecosistema de Swapply:

- **Impacto Técnico:** ¿Afecta la modificación a la integridad de la base de datos híbrida o a la comunicación asíncrona entre servicios?
- **Coste Temporal:** Se estimará el tiempo de desarrollo necesario y si dicha inversión pone en riesgo los plazos de entrega definidos en el cronograma.

**3. Toma de Decisiones y Resolución.** Basándose en el análisis anterior, la decisión final recaerá en el Comité de Control de Cambios (Project Manager, Scrum Master y Desarrollador Principal).

- **Aprobar:** Si el cambio aporta valor significativo a la propuesta de trueque y es viable técnica y temporalmente.
- **Rechazar o Posponer:** Si el cambio desvirtúa el objetivo principal o compromete la estabilidad del prototipo.

**4. Documentación y Trazabilidad.** Para evitar confusiones durante el desarrollo distribuido, se mantendrá un registro estricto de las decisiones:

- **Registro de Cambios:** Se documentarán las modificaciones aprobadas, su justificación y el responsable asignado.
- **Control de Versiones:** Se utilizará GitHub para vincular los commits de código con las tareas o issues correspondientes, asegurando una trazabilidad completa desde la propuesta hasta el código final.

**5. Verificación (Pruebas de Regresión).** Una vez implementado el cambio, se realizarán pruebas específicas para garantizar que la nueva funcionalidad opera correctamente y, crucialmente, que no ha introducido errores en módulos preexistentes (efecto dominó).

Este procedimiento asegura que Swapply evolucione de manera ordenada, protegiendo la calidad del código y garantizando que el equipo se mantenga alineado con los objetivos estratégicos del proyecto.

## 9.5. Comunicación



La gestión de la comunicación es un factor crítico en Swapply debido al tamaño del equipo de desarrollo (11 integrantes) y la interdependencia entre los distintos subsistemas (Matching, Economía, Identidad). Para garantizar la fluidez informativa y evitar la formación de silos de conocimiento, se ha establecido un plan de comunicación estructurado en tres niveles jerárquicos.

### Comunicación Interna y Gestión de Tareas (Nivel Operativo)

El núcleo de la comunicación diaria del equipo se centraliza en la plataforma GitHub (coherente con el Punto 8.2), que actúa como la "fuente única de verdad" del proyecto.

- **Tablero Kanban Digital:** Toda actualización de estado sobre una tarea (desde Backlog hasta Hecho) debe reflejarse inmediatamente en el tablero de GitHub Projects (Punto 9.1). Esto elimina la necesidad de comunicaciones constantes, ya que el estado del proyecto es visible en tiempo real para todos los miembros.
- **Trazabilidad en Issues y Pull Requests:** Las discusiones técnicas, dudas sobre requisitos o reportes de bugs se gestionan a través de la herramienta Issues y comentarios en los Pull Requests. Esto asegura que las decisiones queden documentadas en el contexto del código y sean trazables (Punto 9.4).

### Coordinación y Seguimiento (Nivel Táctico)

Dada la metodología híbrida adoptada, se establecen puntos de control síncronos y formales para alinear al equipo y mitigar riesgos, bajo la dirección del Scrum Master.

- **Reuniones Síncronas Clave (Modelo Scrum):**
  - **Daily Scrum (Stand-up):** Sesiones diarias de 15 minutos para revisar el progreso y reportar impedimentos.
  - **Reunión de Planificación de Sprint:** Al inicio de cada ciclo de trabajo, para seleccionar las tareas prioritarias del Backlog.
  - **Reunión de Retrospectiva:** Al final del Sprint, para analizar la eficiencia y proponer mejoras al proceso.
- **Gestión de Conflictos:** Se fomenta una comunicación abierta para resolver discrepancias de diseño, especialmente en puntos críticos como la integración de la base de datos híbrida o la definición de las reglas de negocio del trueque, asegurando la intervención del Project Manager en caso de escalada.



## Comunicación Externa y Entregables (Nivel Estratégico)

Este nivel abarca la interacción con los stakeholders principales (profesorado/evaluadores) y la presentación del producto final.

- **Repositorio de Documentación:** Se mantiene una estructura de carpetas organizada en la nube (con secciones claras como Alcance, Arquitectura, Plan de Trabajo) para asegurar que la documentación entregada sea siempre la versión más reciente y coherente con el código desarrollado.
- **Presentación de Resultados:** La comunicación del valor del producto se apoyará en los prototipos de alta fidelidad y en la demostración de los flujos principales (Registro, Match, Wallet), asegurando una narrativa visual sólida para la defensa del proyecto.



# 10. Plan de Pruebas



Este plan ha sido diseñado para acompañar al ciclo de vida del desarrollo, comenzando con validaciones tempranas en enero y terminando con una fase intensiva de aseguramiento de calidad entre mediados de abril y principios de mayo. La ejecución será responsabilidad del Scrum Master/Responsable de Calidad (QA), asegurándose de que el sistema llegue al Hito H4 que se da el 11 de mayo, totalmente validado y listo para su implementación, y tener la versión v1 lista para el 1 de junio.

## 10.1. Tipos de Pruebas

Se realizarán diferentes pruebas divididas en tres categorías técnicas diferentes para asegurar la calidad del código, el uso correcto de la app y la estabilidad del sistema:

- **Pruebas unitarias**
  - **Objetivo:** Verificar que todas las partes del código funcionan correctamente de forma aislada, como son los métodos y las clases. Estas pruebas se estarán llevando a cabo hasta el 30 de abril.
  - **Enfoque en Swapply:** Confirmar que la lógica utiliza el backend, concretamente la parte de los algoritmos que crean las diferentes parejas y los cálculos de las valoraciones, para poder asegurarnos de que no existen problemas dentro de nuestro servidor.
- **Pruebas de Integración Continua:**
  - **Objetivo:** Comprobar que los diferentes módulos y subsistemas interactúan y se comunican correctamente entre ellos. Esto incluye la ejecución de pruebas de regresión para asegurar que los cambios introducidos (Punto 9.4) no rompan funcionalidades preexistentes.
  - **Enfoque en Swapply:** Se validará la comunicación entre la API del servidor y la aplicación. También se probará la correcta integración con servicios externos, como la opción de hacer una foto del producto con el que se pretende realizar un intercambio de servicios.



- **Pruebas de Usabilidad y Funcionalidad:**

- **Objetivo:** Validar el comportamiento completo del sistema, asegurando que se cumplen con todos los requisitos funcionales (RF) y que la experiencia del usuario (UX) es fluida e intuitiva.
- **Ejecución:** Se realizará una primera ronda centrada en los flujos principales (hasta el 12 de abril) y una segunda ronda de validación final (hasta el 2 de mayo).
- **Enfoque en Swapply:** Se simularán flujos completos de uso (End-to-End), desde el registro de una cuenta hasta la finalización exitosa de un trueque, verificando la claridad de tareas críticas (Publicar un producto, Proponer un intercambio).

- **Pruebas de Lógica de Negocio y Algoritmos**

- **Objetivo:** Asegurar que las reglas de negocio más complejas se implementan correctamente, más allá del código.
- **Enfoque en Swapply:** Se realizarán pruebas específicas para:
  - **Trueque Circular (RF-A01):** Validar que el algoritmo encuentra correctamente las cadenas A → B → C y que la transacción se cierra correctamente en la base de datos de grafos.
  - **Sistema de Valoraciones:** Verificar que la reputación de un usuario se actualiza correctamente y afecta su visibilidad.

- **Pruebas de Seguridad y Vulnerabilidad**

- **Objetivo:** Garantizar la integridad y confidencialidad de los datos de los usuarios (RNF-S01) antes del lanzamiento.
- **Enfoque en Swapply:** Se realizarán escaneos de vulnerabilidades para verificar la correcta implementación del cifrado SSL/TLS y que no existan puertas traseras (SQL Injection, XSS) en el manejo de la identidad y la Wallet de SwapCoins.

- **Pruebas de Rendimiento y Carga (Mitigación de Riesgos)**

- **Objetivo:** Mitigar directamente el Riesgo de Rendimiento y Sobrecarga (Punto 9.2), asegurando la estabilidad del sistema ante un alto volumen de usuarios y transacciones.
- **Ejecución:** Estas pruebas se realizarán en una fase final intensiva entre el 1 de mayo y el 11 de mayo y serán lideradas por el Especialista en Algoritmos.



## 10.2. Estrategia y Criterios de Validación

Nuestra estrategia se centra en encontrar errores lo antes posible. Usaremos el método de pruebas automáticas y revisiones constantes cada vez que se añada código al proyecto.

- 1. Prioridad:** Empezaremos con las pruebas pequeñas (pruebas de cada función del código) y terminaremos con las pruebas grandes (pruebas de uso final de la aplicación), ya que esto ahorra tiempo y dinero.
- 2. Integración Continua:** Cada cambio de código se revisa automáticamente para asegurar que no ha roto nada del sistema antes de aceptarlo.

**Criterios de Validación para el lanzamiento:** Para que nuestra versión v1.0 esté lista para ser publicada y entre al mercado, debe de cumplir de manera muy estricta los siguientes requisitos de calidad:

- 0 Fallos Críticos:** Antes de poder publicar la aplicación, no puede existir ningún tipo de error bloqueante que impida el registro de una cuenta, poder finalizar un trueque o comprometer los datos personales de los clientes registrados.
- Cobertura del Código:** Al menos el 80% del código, tanto el backend como el frontend, deben de estar cubiertos y validados por las pruebas unitarias automáticas que se habrán realizado con anterioridad.
- Rendimiento de la aplicación:** La carga de las pantallas principales de la página y el mapa de geolocalización para que los usuarios puedan comunicarse con gente que esté cerca de ellos no deben de superar el tiempo de 2 segundos bajo las condiciones normales de la red.
- Compatibilidad:** La aplicación debe de instalarse, abrirse y funcionar de manera correcta en los dispositivos.

## 10.3. Herramientas de Testing

Para poder ejecutar nuestro plan de pruebas y garantizar una cobertura de calidad total, nuestro equipo utilizará un stack tecnológico especializado que es completamente coherente con nuestro entorno de desarrollo.



Utilizaremos **JUnit** como herramienta principal para la ejecución automatizada de pruebas unitarias y de integración en el backend (servidor), asegurando que la lógica de los algoritmos y la API funcionan correctamente. Para el frontend (la parte de la interfaz de usuario en la web), utilizaremos frameworks ligeros como **Jest** o **Mocha**, estándar en JavaScript, para realizar las pruebas unitarias.

En la capa de interacción, emplearemos **Selenium** con el fin de automatizar las pruebas de interfaz, simulando la interacción humana con todos los botones y formularios, tanto en la web como en los diferentes dispositivos móviles.

Para las pruebas de la API y de integración manual, utilizaremos **Postman**, que nos permite probar de forma directa los endpoints del servidor y asegurarnos de que este responde de forma correcta y adecuada a las peticiones de la aplicación.

Finalmente, para mitigar los riesgos de rendimiento, usaremos **JMeter**. Con esta herramienta podremos realizar diferentes pruebas de carga y estrés en el servidor, simulando múltiples usuarios que se conectan a la vez para así prevenir cualquier caída en nuestro sistema. Adicionalmente, utilizaremos **SonarQube** para un análisis estático del código, lo que nos ayudará a detectar automáticamente vulnerabilidades de seguridad, deuda técnica y asegurar la limpieza y comprensión del código.

