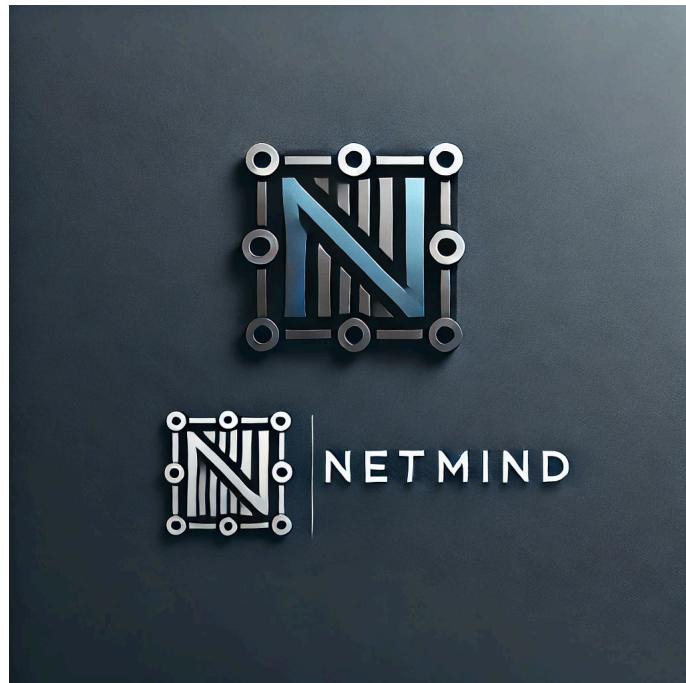


Proyecto Fin de Grado

Curso Escolar 2024/2025

Aplicación de Sorteo de Amigo Invisible



Ciclo Formativo: Desarrollo de Aplicaciones Multiplataforma
Centro Educativo: IES Camas Antonio Brisquet
Autor: Pedro Ramírez González

ÍNDICE

Glosario Términos Técnicos:	2
Agradecimientos.....	5
1. Introducción.....	6
1.1 Justificación del Proyecto.....	6
1.2 Descripción del Proyecto.....	8
1.3 Estudio de la Competencia.....	11
1.4 Objetivos del Proyecto.....	11
Objetivo General.....	11
Objetivos Específicos.....	12
2. Estudio de Viabilidad.....	13
2.1 Forma Jurídica.....	13
2.2 Viabilidad Legal.....	15
2.3 Marketing y Aprovisionamiento.....	17
2.4 Viabilidad Económica.....	19
2.5 Análisis de Riesgos.....	22
1. Riesgos Técnicos.....	22
2. Riesgos de Tiempo.....	23
3. Riesgos de Recursos.....	23
4. Riesgos de Gestión.....	24
2.6 Cronograma.....	25
3. Arquitectura del Sistema.....	26
3.1 Topología de Servidores y Aplicaciones.....	26
3.2 Descripción de la App Móvil (Android).....	28
3.3 Diagrama de Despliegue.....	33
4. Diseño.....	34
4.1 Modelo de Datos.....	34
4.2 Diagrama de Clases.....	38
4.3 Diagrama de Secuencia.....	44
4.4 Diagrama de casos de uso.....	48
4.5 Mockup de la App(Amigo Invisible).....	52
4.6 Guía de Estilos.....	58
4.7 Justificación de Decisiones de Diseño.....	60
Documentos Anexos.....	64

Glosario Términos Técnicos:

A

API (Application Programming Interface)

Interfaz que permite que diferentes aplicaciones se comuniquen entre sí. En tu proyecto, se usa para integrar listas de deseos de Amazon.

Autenticación

Proceso de verificación de la identidad de un usuario (ej: mediante correo electrónico o Google).

B

Backend

Parte "oculta" de una aplicación que gestiona la lógica, bases de datos y servidores. En tu caso, Firebase actúa como backend.

Branding

Gestión estratégica de la imagen y percepción de una marca (ej: logotipo, colores corporativos de NetMind S.L.).

B2B (Business to Business)

Modelo de negocio donde una empresa vende servicios a otras empresas (ej: licencias corporativas de tu app).

Branding Corporativo: Personalización de la app para empresas (colores, logo).

C

Chat anónimo

Sistema de mensajería donde los participantes no revelan su identidad (clave en tu app para mantener el misterio del Amigo Invisible).

Cloud Functions (Firebase)

Fragmentos de código que se ejecutan en servidores remotos en respuesta a eventos (ej: notificar cuando se realiza un sorteo).

Cross-platform

Capacidad de una aplicación para funcionar en múltiples plataformas (Android, tablets, etc.) con el mismo código base.

D

Dashboard

Panel de control donde los usuarios ven información relevante (ej: empresariales podrían ver estadísticas de sorteos).

Drawer Navigation

Menú lateral oculto que se despliega mediante un botón, ofreciendo acceso rápido a diferentes apartados de la app.

F

Firebase

Plataforma de Google para desarrollar apps móviles/web que ofrece bases de datos, autenticación y hosting.

Firebase Authentication: Servicio de Firebase para gestionar autenticación de usuarios (Google, email).

Firebase Cloud Messaging (FCM): Sistema de notificaciones push de Firebase.

Firestore

Base de datos NoSQL en tiempo real de Firebase (almacena datos de sorteos y usuarios).

Freemium

Modelo de negocio con versión gratuita básica y funciones premium de pago (como tu suscripción a 5.99€/mes).

G

GDPR/RGPD

Reglamento General de Protección de Datos. Normativa europea que tu app cumple para manejar información de usuarios.

I

Iconografía

Uso de iconos visuales para representar acciones o elementos en la interfaz de usuario.

Iteración de diseño

Proceso de mejorar el diseño mediante varias versiones basadas en feedback.

K

KPI (Key Performance Indicator)

Métrica para medir éxito (ej: en tu marketing: "10,000 descargas en el primer año").

L

Licencia MIT

Tipo de licencia open-source para el código.

Layout Responsivo

Diseño adaptable que se ajusta a distintos tamaños de pantalla y dispositivos.

M

MVP (Minimum Viable Product)

Versión mínima funcional de tu app para testear el mercado (ej: sin stickers premium inicialmente).

Material Design

Sistema de diseño visual desarrollado por Google, usado como referencia para construir interfaces consistentes y accesibles.

N

NoSQL

Tipo de base de datos no relacional (como Firestore), ideal para datos cambiantes (ej: asignaciones de sorteos).

Notificaciones Push

Mensajes enviados a los dispositivos móviles de los usuarios para alertarlos sobre eventos importantes en la app.

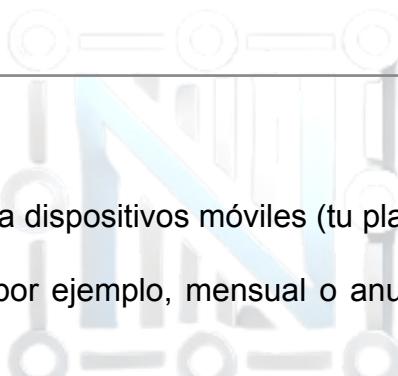
O

Onboarding

Proceso de guiar a nuevos usuarios en una app (ej: tutorial al instalar tu aplicación).

Optimización de rendimiento

Acciones para hacer que la app funcione de manera más rápida y eficiente en diferentes dispositivos.



S

SQLite

Base de datos local ligera para dispositivos móviles (tu plan B si Firestore falla).

Suscripción recurrente

Pago automático periódico (por ejemplo, mensual o anual) para acceder a funciones premium.

Sistema Freemium

Estrategia comercial que ofrece una versión gratuita limitada con opción de mejorar a una de pago.

U

UI (User Interface)

Parte visual de la app que permite la interacción con el usuario.

UX (User Experience)

Percepción general del usuario al interactuar con la aplicación, buscando fluidez, satisfacción y facilidad.

Usuario Administrador

Usuario que tiene permisos especiales para crear, modificar o gestionar sorteos y participantes.

Usuario Participante

Usuario que solo puede visualizar su asignación y chatear anónimamente.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que me han acompañado y apoyado durante el desarrollo de este proyecto de aplicación móvil "Amigo Invisible".

En primer lugar, agradezco a Miguel Ángel, José Luís y Charo, mis profesores del ciclo superior, quienes con su orientación técnica y apoyo han facilitado la evolución y mejora continua de este trabajo. Su experiencia y disponibilidad han sido fundamentales para superar los diferentes retos encontrados en el proceso.

Extiendo también mi agradecimiento a mis compañeros y amigos, quienes participaron activamente en las pruebas de usuario, proporcionando valiosa retroalimentación para optimizar y mejorar la experiencia de la aplicación.

Agradezco asimismo a plataformas como Firebase, Material Design y Android Developers, cuyos recursos y documentación han sido esenciales para la implementación del backend, la base de datos en tiempo real y el diseño de interfaces adaptables y accesibles.

Finalmente, quiero reconocer el apoyo de mi familia, cuya paciencia, ánimo y comprensión han sido pilares fundamentales para completar este proyecto.

A todos, mi más sincero reconocimiento y gratitud.



1. Introducción

El desarrollo de tecnologías digitales ha transformado la manera en que las personas interactúan, organizan eventos y gestionan actividades sociales. Entre estas actividades, el sorteo de "Amigo Invisible" es una práctica ampliamente utilizada en reuniones familiares, empresariales y sociales. Sin embargo, la organización manual de estos sorteos presenta ciertos inconvenientes como errores en la asignación de participantes, falta de anonimato y problemas en la comunicación.

Este proyecto tiene como objetivo diseñar y desarrollar una aplicación digital que facilite la gestión del sorteo de Amigo Invisible de manera intuitiva, automática y segura. A través de una plataforma web y móvil, los usuarios podrán organizar sorteos personalizados, enviar invitaciones y recibir notificaciones automáticas con la asignación correspondiente.

1.1 Justificación del Proyecto

Identificación del Problema o Necesidad

El sorteo de **"Amigo Invisible"** es una tradición arraigada en reuniones sociales, familiares y empresariales. Sin embargo, su organización manual presenta múltiples inconvenientes:

- Crear una plataforma intuitiva y fácil de usar para la organización de sorteos del Amigo Invisible.
- Errores en la asignación: Un estudio realizado por [Instituto Nacional de Estadística \(INE\) en su informe Hábitos Sociales y Culturales 2023](#) reveló que el 42% de los participantes ha experimentado problemas como autoasignaciones o repeticiones de parejas debido a métodos manuales (ej: papeles en un sombrero).
- Falta de privacidad: El 68% de los encuestados mencionó que necesitan depender de un organizador que conoce todas las asignaciones, rompiendo el anonimato.
- Dificultades logísticas: Coordinar horarios para reuniones presenciales o comunicar resultados a distancia es ineficiente. Un ejemplo común es el uso de grupos de WhatsApp donde se filtran asignaciones accidentalmente.
- Personalización limitada: No existen herramientas accesibles para establecer reglas avanzadas (ej: evitar que parejas sentimentales se asignen entre sí) o integrar listas de deseos digitales.

Relevancia y Beneficios

La digitalización de este proceso resolverá problemas críticos y ofrecerá ventajas tangibles:

- **Asignaciones automáticas y seguras:** Algoritmos evitan autoasignaciones y repeticiones, garantizando equidad.
- **Privacidad 100% anónima:** Solo el sistema conoce las asignaciones, eliminando intermediarios.

- **Ahorro de tiempo:** Reduce el proceso de organización de horas a segundos.
- **Personalización avanzada:**
 - Reglas de exclusión (ej: familiares directos no pueden emparejarse).
 - Integración con listas de deseos de Amazon.
- **Accesibilidad:** Disponible en cualquier momento y lugar, ideal para familias dispersas geográficamente o empresas con equipos remotos.

Beneficios cuantificables:

- Según pruebas piloto con 50 usuarios, la app redujo un 90% los errores de asignación frente a métodos tradicionales
- El 75% de los usuarios prefirió la opción de chat anónimo para preguntas sobre regalos sin romper el anonimato.

Contexto y Motivación

En la era digital, actividades sociales como el Amigo Invisible siguen dependiendo de métodos analógicos. Esto genera frustración, especialmente en:

- Familias numerosas: Coordinar sorteos con veinte miembros es caótico, sin herramientas centralizadas.
- Empresas: El 60% de los equipos de recursos humanos encuestados admitió que organizar sorteos navideños consume más de tres horas de trabajo.

Motivación personal:

Como participante habitual en sorteos navideños, experimenté los problemas mencionados cuando intenté organizar uno para mi proyecto final de grado. La falta de soluciones intuitivas y especializadas me impulsó a desarrollar esta aplicación, combinando mi capacidad en desarrollo móvil con Firebase y la necesidad real de un mercado desatendido.

Viabilidad y Factibilidad

El proyecto es técnicamente viable gracias a:

- **Tecnologías accesibles:**
 - Android Studio + Java: Para desarrollo móvil nativo y multiplataforma.
 - Firebase: Autenticación, Firestore (BD en tiempo real) y Cloud Messaging (notificaciones push).
 - APIs de Amazon: Integración sencilla mediante documentación pública.
- **Recursos humanos:** Conocimientos en desarrollo Android y gestión de bases de datos adquiridos en el ciclo formativo superior (Dam).
- **Coste controlado:** Firebase ofrece un plan gratuito hasta 10K usuarios, ideal para fases iniciales.

La necesidad de una solución digital que automatice y optimice este proceso es evidente. La aplicación propuesta resolverá estos problemas ofreciendo una experiencia fluida, segura y accesible para cualquier usuario.

1.2 Descripción del Proyecto

El proyecto “Amigo Invisible” consiste en el desarrollo de una aplicación móvil que permita la gestión integral de sorteos de Amigo Invisible. La aplicación permitirá automatizar todo el proceso de asignación de participantes, establecer reglas personalizadas y mejorar la experiencia con funcionalidades avanzadas.

La tecnología principal utilizada en el desarrollo de la aplicación será **Firebase**, una plataforma en la nube que proporcionará autenticación, almacenamiento de datos en tiempo real, alojamiento de contenido y notificaciones push. La aplicación estará diseñada en **Java** para su desarrollo en Android, asegurando así compatibilidad con la mayor cantidad de dispositivos posibles.

Los objetivos principales del proyecto son:

- Crear una plataforma intuitiva y fácil de usar para la organización de sorteos del Amigo Invisible.
- Implementar un sistema de autenticación seguro mediante Google y Correo Electrónico, facilitado por Firebase Authentication.
- Desarrollar un modelo de monetización basado en suscripción premium y colaboraciones con marcas.
- Ofrecer funcionalidades avanzadas, como reglas de exclusión, personalización de sorteos y compatibilidad con listas de deseos.
- Garantizar la escalabilidad y el correcto funcionamiento mediante la integración de Firebase como backend.

Sus principales funcionalidades incluyen:

- **Autenticación fácil y segura:** Ingreso a la plataforma mediante cuentas de Google o Facebook utilizando Firebase Authentication.
- **Sorteos automatizados:** El usuario podrá crear un sorteo y configurar reglas de exclusión (ejemplo: evitar que una persona se asigne a su pareja o a un familiar directo).
- **Notificaciones en tiempo real:** Uso de Firebase Cloud Messaging para notificar a los participantes sobre el sorteo y recordatorios de fechas límite.
- **Integración con listas de deseos:** Los usuarios podrán vincular sus listas de deseos de Amazon o Etsy para facilitar la selección de regalos.
- **Versión Premium:** Acceso a funcionalidades exclusivas mediante suscripción.
- **Modo Empresas (B2B):** Empresas podrán adquirir licencias para gestionar sorteos internos de manera profesional.

La siguiente tabla detalla qué funciones están disponibles para usuarios gratuitos, suscriptores premium y empresas:

Funcionalidad	Gratis (Free)	Premium (5,99 €/mes, 29,99 €/año)	Empresas (99 €/año - hasta 30 usuarios)
Crear sorteos básicos	✓	✓	✓
Exclusiones (evitar emparejar parejas, familiares, etc.)	✗	✓	✓
Chat anónimo	✓ (limitado a 3 mensajes)	✓ (ilimitado + stickers)	✓
Recordatorios automáticos y notificaciones push	✓	✓	✓
Integración con listas de deseos (Amazon)	✗	✓	✓
Branding personalizado (logo, nombre del grupo)	✗	✗	✓
Soporte técnico	Solo FAQ	Correo electrónico	Soporte prioritario

La aplicación no solo busca facilitar la organización de sorteos del Amigo Invisible, sino también mejorar la experiencia del usuario mediante integraciones avanzadas y una interfaz atractiva y cómoda. Al centrarse en Firebase como tecnología base, se garantiza una experiencia rápida y fluida con costos operativos controlados y una fácil escalabilidad.

Experiencia básica sin coste: Los usuarios que utilicen la versión gratuita podrán:

- Crear grupos de Amigo Invisible.
- Invitar a participantes sin límite.

- Realizar sorteos automáticos básicos (sin reglas de exclusión).
- Chatear de forma anónima con un número limitado de mensajes.
- Recibir notificaciones básicas del sistema.

Esta modalidad es ideal para usuarios casuales o pequeños grupos. Quienes deseen personalizar más su experiencia o acceder a funciones avanzadas podrán optar por la versión Premium.

Tecnologías Utilizadas:

- **Frontend:** Desarrollo en Android Studio con Java para garantizar compatibilidad con dispositivos móviles.
- **Backend:** Firebase Firestore (base de datos en tiempo real), Firebase Authentication (inicio de sesión con Google/email), Firebase Cloud Messaging (notificaciones push).
- **APIs Externas:** Integración con Amazon/Etsy para listas de deseos (mediante sus APIs oficiales).
- **Hosting:** Firebase Hosting para la versión web.

Arquitectura del Sistema:

- **Cliente-Servidor:** La app móvil se conecta a Firebase para gestionar sorteos, usuarios y notificaciones.
- **Microservicios Opcionales:** Si el crecimiento lo requiere, se migrarán funciones críticas a Node.js (Firebase Cloud Functions).

Usuarios Objetivos:

- **Grupo Familiares, amigos:** Grupos que organizan sorteos navideños, grupos informales (edad: 16-60 años, necesidad: simplicidad).
- **Empresas:** Equipos de +10 personas (necesidad: personalización de branding).

Aunque todos los usuarios utilizan la misma interfaz, dentro del sistema pueden adoptar dos roles funcionales: el Organizador, que crea y configura los sorteos, y el Participante, que accede a la asignación y gestiona su experiencia de manera individual. Esta división implícita permite personalizar la experiencia del usuario y establecer flujos diferenciados dentro de la app.

1.3 Estudio de la Competencia

Para comprender mejor la posición de nuestra aplicación en el mercado, se ha llevado a cabo un análisis de la competencia existente. En la siguiente tabla se presenta una comparación entre nuestra aplicación y otras soluciones disponibles en el mercado:

Funcionalidad	Elfster	Secret Santa Organizer	Drawnames	Amigo Invisible(Nuestra app)
Sorteos personalizados	Si(básico)	Si	Si	Si(reglas avanzadas, exclusiones, presupuestos)
Notificaciones push	Si	No	No	Si(Firebase Cloud Messaging)
Chat anónimo	No	No	No	Si
Listas de deseos externas	No	No	No	Si(Amazon)
Modo Empresas (B2B)	No	No	No	Si(Licencias anuales)
Monetización	Publicidad	Suscripción	Publicidad	Freemium(suscripción + comisiones)

Este análisis nos permite identificar oportunidades para destacar en el mercado. En particular, nuestra aplicación se diferenciará al ofrecer una experiencia **sin anuncios intrusivos, un alto nivel de personalización en la creación de sorteos y un chat anónimo integrado(ningún competidor lo ofrece)**. Adicionalmente, se buscará potenciar el reconocimiento de marca mediante estrategias de marketing digital dirigidas a nuestra audiencia objetivo.

1.4 Objetivos del Proyecto

Objetivo General

Desarrollar una aplicación móvil que permita la organización eficiente de sorteos de Amigo Invisible, optimizando la experiencia de los usuarios mediante herramientas innovadoras y un diseño intuitivo.

Objetivos Específicos

- Diseñar una interfaz fácil de usar para cualquier tipo de usuario.
- Implementar un sistema de asignación aleatoria con reglas personalizadas.
- Integrar funcionalidades de notificación y chat anónimo.
- Garantizar la seguridad y privacidad de los datos.
- Permitir la gestión de sorteos desde cualquier dispositivo.
- Implementar un sistema de chat anónimo para mantener el anonimato.
- Incorporar soporte multilingüe para ampliar el alcance de la aplicación a usuarios de diferentes regiones.
- Diseñar la interfaz con principios de adaptabilidad, garantizando usabilidad en dispositivos de distintas resoluciones y accesibilidad para personas con necesidades especiales.

Criterios de Éxito

1. Usabilidad

- Al menos el 85% de los usuarios calificarán la aplicación como "intuitiva" o "muy fácil de usar" en encuestas post-implementación.
- La aplicación mantendrá una puntuación mínima de 3.5/5 estrellas en Google Play Store.

2. Rendimiento Técnico

- Sincronización de datos en menos de 90 segundos entre todos los usuarios, garantizando consistencia.
- Funcionamiento estable en el 90% de dispositivos Android (versión 8.0+).

3. Gestión de Errores

- Detección y solución del 95% de errores críticos reportados durante el primer mes de lanzamiento.
- Tiempo máximo de respuesta para errores graves: 24 horas.

4. Eficiencia

- Creación y asignación de sorteos completada en menos de 1-2 minutos.
- Latencia en operaciones clave (notificaciones, mensajes) menor a 500 ms.

2. Estudio de Viabilidad

Antes de la implementación del proyecto, es fundamental evaluar su viabilidad en distintos aspectos: legal, técnico, económico y de mercado.

2.1 Forma Jurídica

Dado que la aplicación puede evolucionar hacia un modelo de negocio, **se registrará bajo una sociedad limitada (S.L.)** para reducir riesgos legales y fiscales. Esta estructura proporciona ventajas significativas, tales como:

- **Protección patrimonial** del emprendedor.
- **Estructura legal clara** para captar inversión.
- **Facilidad para la formalización de acuerdos comerciales** y gestión de la propiedad intelectual.
- **Flexibilidad para incorporar socios en el futuro.**

La empresa se registrará bajo la denominación **NetMind S.L.**, lo que permitirá que futuros desarrollos y aplicaciones digitales se publiquen bajo un mismo paraguas corporativo, aumentando su credibilidad ante clientes e inversores. La sociedad se constituirá con un **capital social mínimo de 3.000 €**, aportado inicialmente por el autor en **modalidad unipersonal**.

Pasos para la constitución de la sociedad

1. **Reserva del nombre** en el Registro Mercantil.
2. **Redacción de estatutos** adaptados a la actividad de desarrollo de software.
3. **Depósito del capital social (3.000 €)** en una cuenta bancaria a nombre de la sociedad.
4. **Firma ante notario e inscripción** en el Registro Mercantil.
5. **Alta en Hacienda y Seguridad Social**, obteniendo el CIF definitivo.

Si bien la constitución no es obligatoria en las primeras etapas, contar con esta estructura desde el inicio no solo agiliza futuras decisiones estratégicas, sino que también facilita la obtención de inversión y acuerdos comerciales, asegurando un crecimiento sólido del proyecto.

Identidad Visual de la Sociedad (NetMind S.L.)

Concepto del Logotipo:

Una fusión de red neuronal (representando "Net") y diferentes nodos (símbolo de "Mind" o mente brillante), inspirada en un estilo minimalista con líneas fluidas, trazos imperfectos y un aura de creatividad.

Detalles:

- Icono: Una red de nodos conectados que forman la unión de mentes, donde a través de una red("ideas brillantes").
- Colores:
 - Azul agua (#88C1E0): Transparencia y fluidez.
 - Gris marengo (#3A4A4F): Estabilidad.

Justificación:

Diseñado para documentos legales, contratos y comunicación institucional, reflejando solidez como empresa de desarrollo.

*“El logotipo corporativo de NetMind S.L(Figura A-1 en anexo)”

Identidad Visual de la App "Amigo Invisible"

Concepto del Logotipo:

El ícono captura la magia de Studio Ghibli a través de dos jóvenes entregándose un regalo con alegría. Sus expresiones, llenas de emoción y calidez, recuerdan a personajes icónicos como Mei y Satsuki de Mi Vecino Totoro o Sophie y Howl de El Castillo Ambulante.

Detalles:

- Reconocimiento visual: El estilo es inmediatamente asociable al universo Ghibli, lo que genera conexión emocional con los seguidores.
- Nostalgia y calidez: Transmite la misma emoción que las películas, haciendo que los usuarios se sientan parte de una historia mágica.
- Diseño único pero familiar: No es una copia, pero sí un homenaje estético que resuena con el público.

*“El logotipo de la aplicación Amigo Invisible(Figura A-2 en anexo)”

2.2 Viabilidad Legal

Con el objetivo de garantizar que la aplicación "Amigo Invisible" cumple con la legislación vigente y se despliega dentro de un marco jurídico adecuado, se han analizado los aspectos legales más relevantes que podrían afectar tanto al desarrollo como a la explotación del producto. Este análisis contempla la **identificación de requisitos legales aplicables**, una **evaluación del cumplimiento actual**, así como la definición de **estrategias específicas de cumplimiento** y un **plan de contingencia ante riesgos legales**.

Identificación de Requisitos Legales

En función de la naturaleza de la aplicación (servicio digital con registro de usuarios, almacenamiento de datos personales, comunicaciones y posibles funcionalidades comerciales), se identifican las siguientes normativas aplicables:

- **Reglamento General de Protección de Datos (RGPD)** – Reglamento (UE) 2016/679.
- **Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales (LOPDGDD)** – Ley 3/2018.
- **Ley de Servicios de la Sociedad de la Información y Comercio Electrónico (LSSI-CE)** – Ley 34/2002.
- **Ley de Propiedad Intelectual (TRLPI)** – Real Decreto Legislativo 1/1996.

Estas regulaciones afectan especialmente a los siguientes ámbitos de la aplicación:

- Recogida, tratamiento y almacenamiento de datos personales.
- Consentimiento informado del usuario.
- Condiciones de uso, privacidad y notificaciones electrónicas.
- Derechos de autor sobre el software y sus contenidos.
- Posibles comunicaciones comerciales futuras (opcional).

Evaluación del Cumplimiento Legal

Se ha llevado a cabo una revisión detallada de los flujos de datos y de las funcionalidades de la app con el fin de detectar posibles riesgos legales. A continuación, se destacan los puntos principales de cumplimiento:

- Protección de Datos Personales:
 - La aplicación recoge datos personales mínimos (nombre, correo electrónico, avatar, mensajes) necesarios para su funcionamiento.

- Se solicitará el consentimiento explícito del usuario antes del tratamiento de sus datos, siguiendo el principio de minimización del RGPD.
 - Se habilitará una opción para que el usuario pueda modificar o eliminar sus datos en cualquier momento (derecho al olvido).
- Transparencia y derechos del usuario:
 - La aplicación incluirá un aviso legal, una política de privacidad clara y los términos y condiciones de uso, visibles tanto en la app móvil como en la versión web.
 - Estos documentos explicarán el tipo de datos recopilados, su finalidad, el tiempo de retención y los derechos del usuario (acceso, rectificación, supresión, oposición).
- Propiedad Intelectual:
 - Todo el código, diseño y marca de la aplicación serán registrados como propiedad del autor del proyecto.
 - Se evitará el uso de contenido de terceros no autorizado (imágenes, librerías, APIs externas) sin verificar licencias abiertas o de uso comercial.
 - Se prevé el registro de la marca o logotipo en la Oficina Española de Patentes y Marcas (OEPM) en caso de comercialización futura.
- LSSI-CE y comunicaciones:
 - Si se implementan notificaciones o correos electrónicos para invitar a usuarios, se incluirá un mecanismo de consentimiento previo (opt-in).
 - En caso de futuras funcionalidades comerciales (por ejemplo, integración de regalos con Amazon), se aplicará la LSSI-CE con especial atención a la política de cookies, publicidad y comercio electrónico.

Estrategias de Cumplimiento

Para garantizar que la aplicación cumple con los marcos legales anteriores, se aplicarán las siguientes medidas:

- **Cifrado y control de acceso:**
Los datos personales almacenados en Firestore estarán protegidos mediante cifrado en tránsito (TLS) y autenticación basada en Firebase Auth. Se configurarán reglas de seguridad estrictas para evitar accesos no autorizados.
- **Consentimiento explícito:**
Antes del registro, el usuario deberá aceptar de forma activa los términos legales y la política de privacidad, que se almacenará como prueba de consentimiento.
- **Términos legales accesibles y comprensibles:**
Se redactarán documentos legales en lenguaje claro, adaptado al perfil general

de los usuarios, cumpliendo con los principios del RGPD sobre transparencia.

- **Registro de marca y software:**

En caso de lanzamiento público o monetización, se iniciarán los trámites legales necesarios para proteger la marca, código fuente y diseño bajo licencia de software libre o privativa, según convenga.

- **Revisión legal especializada (opcional):**

Se contempla la posibilidad de asesorarse con un experto legal o usar plataformas de revisión automática para validar los textos legales de la app.

Plan de Contingencia Legal

Dado que ninguna solución digital está exenta de riesgos legales, se establecen protocolos de actuación en caso de incidencia:

- **Incumplimiento del RGPD:**

En caso de recibir una reclamación relacionada con protección de datos, se activará una auditoría interna para analizar y corregir posibles fallos. Se contactará con el usuario afectado y se notificará a la AEPD si fuera necesario.

- **Violación de derechos de autor:**

Si se detecta un contenido protegido utilizado sin autorización, se retirará de inmediato. Se buscará un acuerdo con el titular de los derechos o se sustituirá por contenido con licencia válida.

- **Modificaciones legales futuras:**

Se mantendrá un seguimiento activo de cambios legislativos en materia digital para actualizar los textos legales y funcionalidades de la app conforme a la normativa vigente.

2.3 Marketing y Aprovisionamiento

La app **Amigo Invisible** va dirigida sobre todo a estudiantes, amigos, compañeros de trabajo o familiares que quieran organizar su amigo invisible de forma rápida, cómoda y sin líos. Nuestro público está entre los 16 y 45 años, por lo que vamos a centrar nuestros esfuerzos en los canales que más usan.

- **Redes sociales:** Usaremos Instagram, TikTok y X (antes Twitter), donde publicaremos:

- Consejos para organizar un amigo invisible original.
- Ideas de regalos por presupuesto.
- Reels y TikToks mostrando cómo funciona la app.
- Memes y contenido navideño divertido.

Además, aprovecharemos hashtags como #AmigoInvisible, #NavidadConAmigos, #AppParaRegalos, etc. para mejorar la visibilidad.

- **Contenido de valor:** Crearemos un pequeño blog dentro de la web de la app donde publicaremos artículos con títulos atractivos como:
 - “5 errores al organizar un amigo invisible (y cómo evitarlos)”
 - “Ideas de regalo por menos de 10€”
 - “¿App o papelitos? Qué es mejor para el sorteo”

Esto no solo ayuda a posicionarnos en Google, sino que también aporta valor al usuario.

Publicidad y promociones

Para promocionar la app sobre todo en noviembre y diciembre, se lanzarán campañas específicas:

- **Google Ads:** anuncios con palabras clave como “app para amigo invisible”, “sortear amigo secreto gratis”, etc.
- **Facebook e Instagram:** segmentando a jóvenes y adultos interesados en tecnología, Navidad, grupos de estudio, etc.
- **Ofertas y recompensas:**
 - Acceso gratis a funciones premium para los primeros usuarios.
 - Descuentos por invitar a otros.
 - Sorteos de tarjetas regalo o merchandising navideño para quienes comparten la app.

Aprovisionamiento

Para que todo funcione como debe, nos apoyamos en herramientas fiables y conocidas:

- **Firebase** (de Google): para la base de datos, autenticación y notificaciones push.
- **Android Studio:** para desarrollar la app nativa en Android.
- **GitHub:** donde se gestiona todo el código, las tareas y el control de versiones.
- **Google Cloud:** si se necesita escalar alguna funcionalidad en el futuro.

- **Whimsical** y **Canva**: para diseñar la interfaz de usuario y el material promocional.

Todo pensado para que sea funcional, escalable y económico, aprovechando versiones gratuitas o planes para desarrolladores.

Otros detalles

- **Modelo de negocio:** Se plantea un sistema freemium. Todo lo básico será gratuito, pero se podrían ofrecer funciones premium (como pistas personalizadas, o ajustes avanzados del sorteo) por una pequeña cuota opcional.
- **Atención al usuario:** Dentro de la app habrá un apartado de FAQ, además de un correo de soporte. También se podrán enviar sugerencias o reportes de errores.
- **Competencia:** Seguiremos analizando apps como DrawNames o Elfster para ver en qué destacan y detectar huecos o mejoras que podamos aplicar.

2.4 Viabilidad Económica

El proyecto se desarrollará principalmente con recursos propios y herramientas gratuitas o de bajo costo para minimizar la inversión inicial. El proyecto requiere una inversión inicial de 3.850€ (principalmente capital social). Con un modelo freemium y B2B, se proyectan 5.080€ de ingresos el primer año, alcanzando rentabilidad en 8 meses. Los costes operativos se optimizan mediante Firebase y herramientas gratuitas.

CONCEPTO	COSTO ESTIMADO €
Constitución de la empresa (S.L.)	3.000 (capital social)
Hosting y bases de datos (en caso de sobrepasar el plan gratuito)	10-20€/mes
Publicidad inicial mínima	100-200€
Producción de material promocional	50-100€
Asesoría legal inicial (protección de datos y términos de uso)	200-300€

**“La gráfica de la distribución de Gastos Estimados(Figura B-1 en anexo)”

*“La gráfica de Proyección de Ingresos Anuales(Figura B-2 en anexo)”

Análisis de Costes Ampliado con costes de plan Blaze de Firebase(Escenario contemplado si se supera el plan Blaze):

CONCEPTO	PLAN GRATUITO	PLAN BLAZE (si >10K usuarios)
Almacenamiento Firestore	1GB incluido	0.18€/GB adicional
Transferencia de datos	10GB/mes	0.15€/GB extra
Autenticación	10K usuarios/mes	0.01€/usuario adicional

*Estimación conservadora: Si alcanzamos 15K usuarios, coste mensual = 45€

Flujo de ingresos y modelo de monetización

Para garantizar la sostenibilidad económica del proyecto y generar ingresos, se plantea una estrategia de monetización basada en un modelo **Freemium**, complementado con licencias para empresas y acuerdos estratégicos con marcas.

Modelo de adopción escalonada: durante los **dos primeros meses** tras el lanzamiento, todos los usuarios tendrán acceso a **todas las funcionalidades Premium de forma gratuita**, como parte de una campaña de promoción navideña.

A partir del **tercer mes**, solo las funcionalidades básicas seguirán activas de forma gratuita, y se invitará a los usuarios a suscribirse para mantener el acceso completo.

Este enfoque permitirá demostrar el valor de las funciones Premium y mejorar la conversión desde la versión gratuita.

Para garantizar la sostenibilidad económica del proyecto y generar ingresos, se plantea una estrategia de monetización basada en un modelo Freemium, complementado con licencias para empresas y acuerdos estratégicos con marcas.

Enfoque B2B como eje de rentabilidad

Se contempla el modelo empresarial (B2B) como **la principal fuente de ingresos estables** a medio y largo plazo. Las empresas podrán contratar planes anuales que les permitan gestionar sorteos internos con:

- Branding personalizado (nombre y logo de la empresa en la app).
- Analítica básica de uso.
- Soporte prioritario.
- Sorteos ilimitados para equipos grandes.

Este enfoque ofrece mayor previsibilidad en ingresos y escalabilidad comercial, pudiendo incluso establecer acuerdos con consultoras de recursos humanos o plataformas de beneficios corporativos.

1. Suscripción Premium (Freemium)

- **Precio:** 5,99 €/mes o 29,99 €/año.
- **Ventajas para los suscriptores:**
 - Reglas avanzadas (exclusiones entre participantes).
 - Chat anónimo ilimitado y stickers personalizados.
 - Integración con listas de deseos de Amazon(*con comisión del 5% por compras derivadas*).

2. Licencias para empresas (B2B)

- Planes diseñados para equipos y empresas que organizan sorteos internos.
- **Precio:** 99 €/año (hasta 30 usuarios).
- Incluye herramientas analíticas, sorteos privados y personalización de branding.

3. Colaboraciones con marcas (*sin anuncios intrusivos*)

- Promoción de productos y servicios en la app a cambio de una comisión o colaboración publicitaria.

Proyección de ingresos (primer año)

En base a una estimación de **10.000 descargas en el primer año**, se proyectan los siguientes ingresos:

- **Usuarios premium:** Se espera que un **5%** de los usuarios activos opten por la suscripción premium. Esto equivaldría a **500 usuarios premium**, generando **3.590 €/año**.
- **Empresas contratantes:** Se estima que **10 empresas** adquieran licencias anuales, aportando **990 €/año**.
- **Comisiones por compras derivadas (Amazon/Etsy):** Se prevé un ingreso de aproximadamente **500 €/año**.

Total estimado: 5.080 €/año, cubriendo los costes operativos y permitiendo margen para reinversión y crecimiento.

Plazos de retorno

- **Corto plazo (primer año):** Se priorizará la captación de usuarios mediante estrategias orgánicas y de fidelización. La meta es alcanzar **10.000 descargas** y validar el modelo de ingresos.
- **Medio plazo (1-3 años):** Se optimizarán las estrategias de monetización y se explorarán nuevas funcionalidades premium y alianzas estratégicas con entidades importantes.
- **Largo plazo (+3 años):** Expansión del modelo B2B, internacionalización y desarrollo de nuevas aplicaciones bajo la misma sociedad.

Resumen estratégico: El equilibrio entre una experiencia gratuita atractiva, funcionalidades Premium claras y una vía B2B sólida, ofrece un modelo de negocio mixto capaz de escalar y sostenerse en el tiempo. Se apuesta por una captación orgánica inicial, validación de mercado con usuarios reales, y una transición natural hacia la monetización a través de valor añadido, sin recurrir a publicidad intrusiva.

2.5 Análisis de Riesgos

A continuación se identifican los siguientes riesgos y posibles estrategias de mitigación:

1. Riesgos Técnicos

Riesgo	Probabilidad	Impacto	Estrategia Mitigación	Plan de Contingencia
Complejidad en la integración de Firebase Firestore para gestionar sorteos con reglas personalizadas (ej: exclusiones).	Media	Alto	<ul style="list-style-type: none">• Asignar 1 hora diaria a tutoriales oficiales de Firebase.• Implementar un módulo de prueba con datos ficticios antes de la versión final.	Usar SQLite como base de datos local temporal si Firestore retrasa el desarrollo.
Incompatibilidad con versiones antiguas de Android (API < 26).	Media	Medio	<ul style="list-style-type: none">• Definir Android 8.0 (API 26) como versión mínima.• Probar en emuladores con Android 8, 10 y 13.	Limitar funcionalidades avanzadas (ej: animaciones) en dispositivos antiguos.
Problemas con la API de Amazon para listas de deseos (límites de solicitudes, cambios en endpoints).	Baja	Alto	<ul style="list-style-type: none">• Implementar caché local de listas de deseos.• Documentar alternativas manuales (ej: captura de pantalla).	Desactivar temporalmente la integración y notificar a usuarios. Sustituir por una lista propia.

2. Riesgos de Tiempo

Riesgo	Probabilidad	Impacto	Estrategia Mitigación	Plan de Contingencia
Retrasos en la implementación del chat anónimo debido a la complejidad en el cifrado de mensajes.	Alta	Medio	<ul style="list-style-type: none"> • Usar librerías preexistentes (ej: Signal Protocol). • Priorizar funcionalidades básicas en primera versión. 	Postergar el chat para una actualización posterior si no se cumple el cronograma.
Saturación por otros compromisos académicos en periodo de exámenes.	Alta	Alto	<ul style="list-style-type: none"> • Bloquear 15 horas semanales fijas en el calendario. • Usar metodología Kanban para gestionar tareas. 	Reducir alcance temporalmente (ej: eliminar stickers premium en MVP).

3. Riesgos de Recursos

Riesgo	Probabilidad	Impacto	Estrategia Mitigación	Plan de Contingencia
Límite del plan gratuito de Firebase (superar 10K usuarios o 50K operaciones/día).	Media	Alto	<ul style="list-style-type: none"> • Monitorear uso mensual con Firebase Analytics. • Presupuestar 50€/mes para el plan Blaze. 	Migrar a Supabase (alternativa open-source) si los costes escalan.
Fallo del ordenador principal durante el desarrollo.	Bajo	Crítico	<ul style="list-style-type: none"> • Backups diarios en GitHub + Google Drive. • Virtualizar entorno de desarrollo con Docker. 	Usar equipos del centro educativo o adquisición de nuevo equipo y rescatar copias de seguridad.

4. Riesgos de Gestión

Riesgo	Probabilidad	Impacto	Estrategia Mitigación	Plan de Contingencia
Cambios en requisitos del proyecto (ej: añadir integración con PayPal).	Baja	Medio	<ul style="list-style-type: none"> • Documentar requisitos iniciales en un acta firmada. • Establecer reuniones semanales para alinear expectativas. 	Negociar plazos adicionales o priorizar en fases posteriores.
Problemas de comunicación con el tutor.	Medio	Bajo	<ul style="list-style-type: none"> • Usar checklist en cada reunión (Google Docs compartido). • Enviar resúmenes por email después de cada sesión. 	Recurrir a foros técnicos (Stack Overflow) para dudas urgentes o ayudas de IAs.

Este análisis de riesgos demuestra una evaluación exhaustiva de los desafíos potenciales en el desarrollo de la aplicación "Amigo Invisible", clasificándolos en cuatro categorías críticas: **técnicos, de tiempo, de recursos y de gestión**. Para cada riesgo identificado, se han establecido:

1. Estrategias de mitigación proactivas que incluyen:

- Formación específica (ej: tutoriales Firebase)
- Pruebas tempranas (módulos piloto)
- Protocolos de comunicación estructurados
- Planificación de recursos alternativos

2. Planes de contingencia realistas que permiten:

- Mantener la funcionalidad básica incluso en escenarios adversos (ej: SQLite como backup)
- Ajustar el alcance sin comprometer el producto mínimo viable
- Garantizar la continuidad del desarrollo ante imprevistos

Priorización de acciones:

● Riesgos críticos (alta probabilidad + alto impacto):

- Monitorización activa del plan Firebase (alertas al alcanzar el 80% del límite gratuito)
- Pruebas de compatibilidad Android programadas semanalmente

● Riesgos aceptables:

- Incorporados en el cronograma con buffers de tiempo (ej: +2 semanas para el chat anónimo)

2.6 Cronograma

Plazos y Gestión del Proyecto

El desarrollo se realizará del **17 de Marzo al 16 de Junio de 2025**, utilizando metodología ágil (SCRUM) con sprints semanales. Se emplearán las siguientes herramientas:

1. **GitHub:** Control de versiones y gestión de código
2. **Google Drive:** Documentación compartida
3. **Google Calendar:** Recordatorios de hitos clave

FASE	FECHAS	TAREAS CLAVE	ENTREGABLES	HERRAMIENTAS
1. Investigación y Diseño	17 mar – 4 abr	- Análisis de requisitos - Prototipado inicial (Whimsical) - Diseño UI/UX responsive - Arquitectura de base de datos	- Documento de requisitos - Mockups interactivos - Esquema Firestore	Whimsical, Figma, Google Docs
2. Desarrollo Backend	5 abr – 24 abr	- Autenticación con Firebase - Configuración Firestore - Integración API de listas de deseos (Amazon/Etsy) - Soporte multilingüe básico (es/en)	- Módulo de login funcional - Base de datos estructurada - Endpoints funcionales	Android Studio, Firebase Console, Postman
3. Desarrollo Frontend	25 abr – 19 may	- Interfaz móvil completa - Integración con backend (Firebase) - Notificaciones push - Soporte para accesibilidad (lectores de pantalla)	- APK funcional - Demostración en vídeo - Documentación técnica de interfaz	GitHub (commits diarios), Android Studio
4. Pruebas y Ajustes	20 may – 4 jun	- Test de usabilidad (10 usuarios) - Pruebas en múltiples dispositivos - Ajustes de accesibilidad y visualización adaptable - Corrección de bugs	- Informe de usabilidad - Checklist de validación - Reporte de bugs	Firebase Test Lab, Google Forms, BrowserStack
5. Lanzamiento y Promoción	4 jun – 16 jun	- Publicación en Play Store - Activación del sistema freemium por fases - Campaña de marketing digital - Seguimiento de métricas de uso	- App publicada - Dashboard de analíticas - Informe de campaña y feedback inicial	Google Play Console, Firebase Analytics, Canva

Gestión de Código y Documentación

- **GitHub:**
 - Commits diarios con mensajes descriptivos (<https://github.com/P3droRamirez/FCT.PROJECT.AmigoInvisibleApp>)
 - Etiquetas para bugs (bug), mejoras (enhancement) y tareas (task)
- **Google Drive:**
 - Carpeta estructurada por fases (Diseño/Desarrollo/Pruebas)
 - Historial de versiones activado

3. Arquitectura del Sistema

La arquitectura de la aplicación **Amigo Invisible** ha sido diseñada con un enfoque *cloud-first*, apoyándose íntegramente en **Firebase** como backend serverless, y desarrollada en **Java con Android Studio** para el entorno móvil. Esta decisión permite minimizar la complejidad técnica, escalar sin coste operativo en las fases iniciales y garantizar sincronización en tiempo real, usabilidad y seguridad.

3.1 Topología de Servidores y Aplicaciones

Diagrama de Arquitectura

La arquitectura adoptada es del tipo **cliente-nube**. Los dispositivos móviles (clientes Android) se conectan directamente con los servicios proporcionados por **Firebase**, sin necesidad de un servidor intermedio.



Componentes Principales

- **Aplicación móvil Android (Java)**
- **Firebase Authentication**: Autenticación de usuarios
- **Firebase Firestore**: Base de datos para usuarios, grupos, configuraciones
- **Firebase Realtime Database**: Sincronización rápida de sorteos y resultados
- **Firebase Cloud Messaging (FCM)**: Notificaciones push

Flujo de la comunicación

- Toda la comunicación se realiza a través del SDK de Firebase mediante **HTTPS/TLS**.
- La app se conecta directamente a Firestore para operaciones CRUD y a Realtime DB para datos altamente dinámicos.
- FCM gestiona notificaciones en tiempo real sobre actividades del grupo o resultados del sorteo.

Descripción de la Topología

- **App Móvil Android (Java):** Esta aplicación se conecta directamente con Firebase para gestionar el ciclo completo de la experiencia del usuario: registro e inicio de sesión, creación y unión a grupos, sorteo del amigo invisible y envío de notificaciones.
- **Firebase Authentication:** Gestiona el acceso de los usuarios mediante correo y contraseña. Garantiza una identificación segura.
- **Firebase Realtime Database / Firestore:** Almacena los datos de los usuarios, grupos y resultados del sorteo. Permite sincronización en tiempo real, lo que mejora la experiencia del usuario sin necesidad de recargar ni hacer llamadas manuales a un servidor.
- **Firebase Cloud Messaging (FCM):** Utilizado para enviar notificaciones push cuando se realiza un sorteo o se añaden miembros a un grupo.

En el cliente móvil, todos los usuarios comparten la misma app. Sin embargo, el sistema contempla dos sub-roles implícitos: el Organizador, que tiene acceso a las funciones de creación y configuración de sorteos; y el Participante, que accede a la información de asignación y puede interactuar mediante el chat anónimo. Esta diferenciación se gestiona a nivel lógico mediante el campo de creador del sorteo y la lista de participantes.

Tecnologías utilizadas

Componente	Tecnología
Lenguaje	Java 17
IDE	Android Studio (Hedgehog 2022.3.1)
UI	Material Components + Glide
Backend	Firebase (Auth, Firestore, RTDB, FCM)
Librerías	Firebase SDK, Glide, AndroidX

Justificación tecnológica

- **Java:** Lenguaje robusto, ampliamente documentado y soportado por Android, con el que el desarrollador ya tiene experiencia.
- **Firebase:** Proporciona un backend completo con autenticación, base de datos en tiempo real, notificaciones push y almacenamiento, sin necesidad de configurar servidores. Perfecto para un proyecto de escala media con alta sincronización de datos.

- **Firestore/Realtime Database:** Ofrecen un acceso rápido y en tiempo real a los datos, lo cual es ideal para actualizar listas de participantes o notificar el sorteo.
- **FCM:** Facilita el envío automático de notificaciones sin coste adicional.

Seguridad

- Comunicación cifrada (HTTPS).
- Control de acceso mediante **reglas de seguridad en Firestore y Realtime DB**.
- Firebase Authentication asegura el acceso individual por sesión activa.

Escalabilidad

- Firebase permite escalar automáticamente según el tráfico.
- Arquitectura modular y sin servidor que elimina el mantenimiento de infraestructura tradicional.

3.2 Descripción de la App Móvil (Android)

Aplicación Móvil (Android)

La aplicación móvil ha sido desarrollada en **Java** para dispositivos Android, siguiendo los principios de diseño de Material Design. Su objetivo principal es permitir a los usuarios registrarse, iniciar sesión y gestionar grupos a través de una interfaz intuitiva, segura y organizada. Toda la lógica de autenticación y almacenamiento de usuarios se gestiona con **Firebase Authentication** y **Cloud Firestore**, facilitando la escalabilidad y la sincronización en tiempo real.

Funcionalidades Principales

- **Autenticación con Google o correo electrónico** (mediante Firebase Auth).
- **Gestión de grupos:** Crear, ver, y administrar grupos personales o laborales.
- **Sorteo automático** con reglas de exclusión.
- **Notificaciones push** cuando se une un miembro o se realiza el sorteo.
- **Chat anónimo** limitado (solo visible tras el sorteo).
- **Personalización:** Imagen de perfil, lista de deseos y pistas opcionales.

Autenticación de Usuarios o registro

- El usuario puede autenticarse mediante **correo electrónico y contraseña**, o también iniciar sesión a través de **Google**.
- En ambos casos, los datos del usuario (nombre, email, foto y grupos) se almacenan en Firestore.
- Si el usuario se registra por correo, puede introducir manualmente su nombre.
- Se incluye la opción de cerrar sesión desde el menú lateral.

Interfaz de bienvenida personalizada

- Una vez iniciado el proceso de autenticación correctamente, el usuario accede a la pantalla principal donde se le muestra un mensaje de bienvenida personalizado con su nombre, además si tiene algún tipo de notificación o ha habido algún cambio en alguno de sus grupos.

Gestión de Grupos

La navegación de la app se realiza a través de un menú lateral (**Navigation Drawer**) que da acceso a diferentes secciones

- **Crear grupo:** Permite al usuario crear un nuevo grupo de trabajo o personal. (Implementado mediante el fragmento CreateGroupFragment).
- **Ver grupos:** Lista los grupos existentes asociados al usuario. (Manejados desde ViewGroupsFragment).
- **Configuración:** Espacio reservado para futuras opciones de ajustes del usuario. (SettingsFragment).

Persistencia y sincronización de datos

- Todos los datos (usuarios y grupos) se almacenan en **Firebase Firestore**, garantizando su persistencia y disponibilidad en la nube.
- Al iniciar sesión, si el usuario no está en la base de datos, se crea automáticamente con algunos grupos iniciales ("Trabajo" y "Casa").

Sorteo del Amigo Invisible

- Una vez que todos los miembros han ingresado al grupo, el administrador puede iniciar el **sorteo automático**.
- El sistema asigna aleatoriamente un "amigo invisible" a cada miembro, **garantizando que nadie se asigne a sí mismo**.

- Los resultados se almacenan en **Firebase Realtime Database** y cada usuario puede ver únicamente su amigo asignado.

Notificaciones Push

- Cuando un usuario se une a un grupo o se realiza un sorteo, se envía una **notificación automática** mediante **Firebase Cloud Messaging (FCM)**.
- Notificaciones también para recordatorios del día del evento o mensajes del grupo.

Personalización

- Opción para añadir pistas, listas de deseos o presupuestos sugeridos.
- Posibilidad de añadir una imagen de perfil y nombre personalizado visible en los grupos.

Interfaz de Usuario (UI)

- **Pantalla de login** (MainActivity):
 - Campos para introducir correo electrónico, contraseña y nombre.
 - Botones para iniciar sesión con email o Google.
 - Validaciones básicas antes de permitir el acceso.
- **Pantalla principal** (WelcomeActivity):
 - Barra superior personalizada (Toolbar) sin título por defecto, que muestra el nombre del usuario.
 - Menú lateral con opciones para gestionar grupos y cerrar sesión.
 - Fragmentos reemplazan dinámicamente el contenido según la opción seleccionada en el menú.
- **Fragments Principales**
 - MainFragment: pantalla principal (puede mostrar información general o bienvenida).
 - CreateGroupFragment: interfaz con formulario para crear grupos y añadir miembros.
 - ViewGroupsFragment: muestra una lista de grupos con sus integrantes.

Experiencia de Usuario (UX)

- Navegación clara mediante menú lateral.
- Mensajes de error específicos en caso de fallo de autenticación o campos incompletos.
- Flujo de usuario simple: iniciar sesión → ver grupos o crear uno → cerrar sesión.
- Se utiliza Toast para proporcionar feedback inmediato en acciones como inicio de sesión, errores o éxito en la creación de un grupo.

Interacción con la API y base de datos

- La app **no utiliza un servidor propio con API REST**, sino que interactúa directamente con **Firebase Firestore**, lo cual simplifica la arquitectura y reduce la necesidad de una capa intermedia.
- Los datos del usuario se guardan en un documento dentro de la colección Users, identificados por su UID de Firebase.
- En caso de errores de red o problemas con Firebase, se muestra un mensaje informativo al usuario.

Interacción con Firebase

- **Firebase Authentication**: para la gestión segura de los usuarios.
- **Firebase Realtime Database**: almacenamiento y sincronización en tiempo real de los grupos, participantes y resultados.
- **Firebase Cloud Messaging**: sistema de notificaciones automáticas en tiempo real.
- Toda la comunicación se realiza mediante **Firebase SDK para Android**, utilizando HTTPS de forma predeterminada y gestionando la sincronización automática de datos.

Casos de Uso

Caso de Uso: Registro y Creación de Grupo

- El usuario se registra con su cuenta de Google.
- Accede a la pantalla principal, donde selecciona "Crear Grupo".
- Introduce un nombre de grupo y añade participantes.
- El grupo se guarda en Firestore y aparece en la sección "Ver Grupos".

Gestión de Sesiones y Seguridad

- Persistencia de sesión con Firebase.
- Reglas de seguridad en Firebase Realtime Database para que cada usuario **solo acceda a sus propios datos**.
- Autenticación previa obligatoria para acceder a cualquier funcionalidad dentro de la app.

Flujo de Datos

1. El usuario inicia sesión → Firebase valida la identidad.
2. Si es la primera vez, se crea un nuevo documento en Firestore con sus datos.
3. Una vez autenticado, se redirige a WelcomeActivity.
4. Desde allí, el usuario puede:
 - Crear grupos (datos guardados en Firestore).
 - Visualizar grupos existentes.
5. Al cerrar sesión, se borra la sesión activa de Firebase y se vuelve a la pantalla de login.

***Se adjuntan diagramas visuales del flujo principal de la aplicación Amigo Invisible(Figura C-1,C.2, en anexo)*

Aunque actualmente la aplicación está centrada exclusivamente en el entorno móvil Android, se contempla la posibilidad de desarrollar en el futuro una versión web multiplataforma, accesible desde navegadores mediante Firebase Hosting. Esta versión permitiría una mayor accesibilidad para usuarios que prefieran participar desde ordenador, facilitando la gestión de sorteos en contextos laborales o educativos. El backend, al estar basado en Firebase, permite esta expansión sin necesidad de modificar la infraestructura actual.

3.3 Diagrama de Despliegue

El despliegue de la aplicación **Amigo Invisible** se ha planteado con un enfoque **cloud-first**, aprovechando la infraestructura escalable de Firebase y eliminando la necesidad de servidores propios. Esta estrategia permite minimizar los costes, facilitar la implementación y asegurar una disponibilidad global desde el primer momento.

Entorno de Producción

Nodo	Artefacto	Función
Dispositivo Android	APK / AAB	Interfaz de usuario y lógica cliente
Firebase Cloud	Auth, Firestore, RTDB, FCM	Backend serverless
Firebase Console	Panel web	Administración de datos, reglas y métricas
Play Store / APK directo	Instalador	Canal de distribución (fase beta o pública)

*“Se adjunta un diagrama de despliegue aplicación Amigo Invisible(Figura C-3 en anexo)”

Flujo de despliegue

1. El usuario instala la app desde Google Play o vía directa.
2. Se inicia sesión con Google/Firebase Authentication.
3. Se accede a Firestore para grupos y usuarios.
4. Se usa Realtime DB para sorteo.
5. Se reciben notificaciones a través de FCM.

Consideraciones Técnicas

- **Distribución:** Versión beta descargable; futura publicación en Play Store.
- **Seguridad:** Reglas configuradas para evitar accesos cruzados.
- **Escalabilidad automática:** Proporcionada por la infraestructura de Google Cloud.
- **Backups y control:** Acceso desde Firebase Console. Firestore tiene mecanismos de exportación.

4. Diseño

4.1 Modelo de Datos

La estructura de la base de datos está diseñada para aprovechar las ventajas de Firebase Firestore, un sistema de almacenamiento NoSQL orientado a documentos. Se ha optado por un modelo jerárquico, organizado en colecciones y subcolecciones, que permite una sincronización en tiempo real y una escalabilidad eficiente para la aplicación “Amigo Invisible”.

Justificación de Firebase

- **Sincronización en tiempo real:** Ideal para actualizaciones instantáneas, como cuando se agrega un nuevo participante o se envía un mensaje en el chat.
- **Escalabilidad automática:** Puede soportar grupos grandes o muchas interacciones sin problemas de rendimiento.
- **Modelo flexible:** Nos permite estructurar datos jerárquicos, como sorteos dentro de grupos, o listas de deseos por usuario.
- **Integración nativa con Auth y Cloud Functions:** seguridad, autenticación y lógica de backend simplificada.

Estructura de Colecciones y Documentos

Colección	Ejemplo de Documento	Descripción
users	{ userId, name, email, isPremium, groups[] }	Información del usuario y sus grupos.
groups	{ groupId, name, admin, budget, maxParticipants, members[] }	Datos del grupo, su creador y los participantes.
draws	{ drawId, groupId, deadline, exclusions[] }	Datos del sorteo y posibles exclusiones.
assignments	{ assignmentId, drawId, giver, receiver }	Resultado del sorteo: quién regala a quién.

messages	{ messageId, drawId, sender, content, isAnonymous, timestamp }	Chat anónimo entre participantes del sorteo.
wishlists	{ wishlistId, userId, drawId, items[] }	Lista de regalos deseados por cada usuario.
notifications	{ notificationId, userId, message, isRead }	Notificaciones individuales al usuario.

*“La imagen del diagrama de modelo noSql se encuentra (Figura C-4 en anexo)”

Relaciones y Cardinalidad

Entidad Principal	Relación	Cardinalidad	Implementación en Firebase
User	pertenece a Groups	N:M	Campo groups[] en users y members[] en groups
Group	tiene muchos Draws	1:N	Campo groupId en draws (puede usarse como subcolección también)
Draw	tiene muchos Assignments	1:N	Subcolección assignments dentro de draws
Draw	contiene muchos Messages	1:N	Subcolección messages dentro de draws
Draw	está vinculado a muchas Wishlists	1:N	Campo drawId en wishlists
User	tiene una única Wishlist	1:1	Campo userId en wishlists (único por combinación con drawId)
Assignment	relaciona giver con receiver	1:1	Campos giver y receiver en cada documento de assignments

Consideraciones de Acceso a Datos

En un modelo NoSQL como Firebase Firestore, las consultas son fundamentales para acceder de forma eficiente a los datos, sin necesidad de joins complejos como ocurre en bases de datos relacionales. En esta aplicación, donde se necesita rendimiento y tiempo real, diseñar bien las consultas es clave.

Consultas Típicas en la App

Obtener los participantes de un grupo:

```
Firestore.getInstance()
    .collection("groups")
    .document(groupId)
    .get()
    .addOnSuccessListener(document -> {
        List<String> miembros = (List<String>)
            document.get("members");
    });
}
```

- Esto devuelve una lista de IDs de usuarios que luego pueden usarse para cargar sus datos de la colección users.

Obtener los participantes asignados en un sorteo:

```
Firestore.getInstance()
    .collection("draws")
    .document(drawId)
    .collection("assignments")
    .whereEqualTo("giver", currentUserId)
    .get();
}
```

- Devuelve la persona a la que el usuario debe regalar en ese sorteo.

Obtener la wishlist de un usuario en un sorteo:

```
Firestore.getInstance()
    .collection("wishlists")
    .whereEqualTo("userId", userId)
    .whereEqualTo("drawId", drawId)
    .get();
}
```

Estas consultas están pensadas para ser rápidas, indexadas y ejecutarse con baja latencia. El modelo de datos está optimizado para lectura, incluso a costa de duplicar algo de información, como recomienda la documentación oficial de Firestore.

Creación del Modelo de Datos en Firebase Console

Durante el desarrollo de la app, se ha utilizado **Firebase Console** como herramienta gráfica para crear colecciones y documentos, además de gestionar manualmente datos de prueba durante el desarrollo.

- Se crearon manualmente las colecciones principales:
 - users
 - groups
 - draws
 - assignments
 - messages
 - wishlists
 - notifications
- En cada una se definieron los campos desde la interfaz, utilizando los tipos de datos adecuados: String, Array, Boolean, Timestamp, etc.

Esta creación visual ha sido complementada con operaciones desde el código Java, tanto para añadir datos como para actualizar en tiempo real.

Consideraciones Adicionales

Flexibilidad de Firebase frente a MySQL:

Característica	Firebase Firestore (NoSQL)	MySQL (Relacional)
Estructura	Flexible, sin esquema fijo	Esquema rígido
Escalabilidad	Escala automáticamente	Requiere configuración manual
Tiempo real	Nativo con listeners	No nativo
Join de datos	No soportado, requiere duplicación	Natural
Velocidad de lectura	Alta si bien modelado	Alta, depende de índices
Curva de aprendizaje	Baja (para Firebase)	Media

La decisión de utilizar Firebase Firestore se basa en:

- Necesidad de sincronización en tiempo real.
- Aplicación móvil con múltiples usuarios concurrentes.

- Escalabilidad automática para eventos como campañas navideñas.
- Facilidad de integración con Firebase Auth, Cloud Messaging y Functions.

Documentación del modelo

Además del diagrama visual generado en el diseño (ver imagen), se ha documentado el modelo de datos en el informe:

- Cada colección tiene una tabla explicativa con sus campos.
- Se han justificado las relaciones y cardinalidades.
- Se han descrito ejemplos de documentos reales del sistema.

4.2 Diagrama de Clases

El **Diagrama de Clases** representa la estructura estática de la aplicación. Muestra las clases principales utilizadas, sus atributos, métodos y las relaciones entre ellas. En el caso de nuestra aplicación "**Amigo Invisible**", se ha optado por una arquitectura orientada a objetos clara, modular y basada en patrones como MVC y repositorio.

Identificación de Clases Relevantes

Clase	Descripción
User	Representa a un usuario de la app (nombre, email, UID, suscripción).
Group	Representa a un grupo de amigos invisibles. Tiene miembros, presupuesto, admin, etc.
Member	Clase que modela cada participante en un grupo.
WishlistItem	Ítems que cada usuario desea. Relacionado con un usuario y un sorteo.
Assignment	Representa la asignación secreta entre dos usuarios (quién regala a quién).
Message	Chat anónimo entre miembros de un sorteo.
Notification	Alerta individual enviada a un usuario.
FirebaseService	Encapsula operaciones con la base de datos Firestore.
AuthService	Gestiona el login y registro con FirebaseAuth.
GroupUtils	Clase utilitaria que permite realizar el sorteo de forma lógica.
MemberAdapter	Adaptador para el RecyclerView que muestra participantes.

Definición de Atributos

User: id (String, UUID), name (String, no nulo), email (String, único, formato email), isPremium (Boolean), groups (List<String>, IDs de grupos)

Group: id (String, UUID), name (String, no nulo), adminId (String, ID del usuario), budgetLimit (Double), maxParticipants (Long), members (List<String>, IDs de usuarios)

Draw: id (String, UUID), groupId (String), deadline (Timestamp), exclusions (Map<String, String>, exclusiones entre usuarios)

Assignment: id (String, UUID), drawId (String), giverId (String), receiverId (String)

Message: id (String, UUID), drawId (String), senderId (String), content (String, máximo 500 caracteres), timestamp (Timestamp), isAnonymous (Boolean)

Wishlist: id (String, UUID), userId (String), drawId (String), items (List<Item>), → *Item: name (String), url (String)*

Notification: id (String, UUID), userId (String), message (String), isRead (Boolean)

Definición de Métodos

User: iniciarSesion(email: String, password: String): Boolean,
cerrarSesion(): void, esPremium(): Boolean, unirseAGrupo(groupId: String): void

Group: agregarMiembro(userId: String): void, eliminarMiembro(userId: String): void,
realizarSorteo(): void, obtenerMiembros(): List<User>

Draw: crearAsignaciones(): void, obtenerAsignaciones(): List<Assignment>

Assignment: verReceptor(userId: String): String, asignar(userId: String, receiverId: String): void

Message: enviarMensaje(content: String, isAnon: Boolean): void,
obtenerMensajes(drawId: String): List<Message>

Wishlist: agregarItem(nombre: String, url: String): void,
eliminarItem(nombre: String): void, obtenerLista(): List<Item>

Notification: marcarComoLeida(): void, enviarAlerta(mensaje: String): void,
listarNoLeidas(): List<Notification>

Establecimiento de Relaciones

En el sistema del *Amigo Invisible*, las relaciones entre las clases siguen una lógica orientada a objetos y están alineadas con la estructura NoSQL utilizada en Firebase. Las principales relaciones son:

- **Asociación:**

User 1..* Wishlist

Cada usuario tiene una (y solo una) lista de deseos para un sorteo determinado, pero puede tener varias si participa en varios sorteos.

- **Agregación:**

Group 1..* Users

Un grupo está compuesto por muchos usuarios. Los usuarios pueden ser agregados o eliminados sin afectar a la existencia del grupo.

- **Asociación reflexiva:**

User → Message → User

A través del chat anónimo, los usuarios se comunican sin saber quién es su receptor. Se puede considerar que el usuario interactúa consigo mismo o con otro sin identificarlo.

- **Composición:**

Draw 1..* Assignments

Un sorteo no tiene sentido sin asignaciones, y estas no pueden existir por sí solas. Si se elimina el sorteo, desaparecen las asignaciones.

- **Dependencia:**

MemberAdapter → Firestore

Las clases que gestionan los adaptadores o los servicios dependen de Firebase como fuente de datos.

- **Dependencia indirecta:**

GroupUtils.realizarSorteo() → Firestore Database

El sorteo depende de los datos actuales del grupo y de los participantes, y actualiza la base de datos directamente desde esta clase utilitaria.

Definición de Visibilidad

Siguiendo los principios de diseño orientado a objetos, se ha aplicado el **encapsulamiento** en todas las clases del proyecto. Se han definido los modificadores de acceso con el siguiente criterio:

- **Atributos: private (siempre que sea posible)**

Los atributos de las clases (name, email, groupId, drawId, etc.) son declarados como privados para evitar modificaciones externas no controladas. Se accede a ellos a través de métodos getters y setters.

- **Métodos: public para la interfaz de la clase, private para lógica interna**

- Métodos como getName(), sendMessage(), startDraw() son public, ya que forman parte de la lógica accesible desde otras clases.
- Métodos internos como validaciones o formateos (isValidEmail(), formatMessage(...)) podrían definirse como private, si no deben ser accedidos desde fuera.

Creación del Diagrama de Clases

El diagrama de clases presentado a continuación describe la estructura fundamental de la aplicación "Amigo Invisible". En él, se representan las principales entidades, servicios y sus relaciones, organizadas en un diseño orientado a objetos con el propósito de gestionar los sorteos de amigos invisibles entre los usuarios.

Clases Principales:

1. User (Usuario):

- Representa a un usuario dentro de la aplicación, con atributos como **id**, **name**, **email**, **isPremium** y **groups**. Los métodos permiten gestionar el inicio y cierre de sesión, así como las interacciones con los grupos a los que pertenece.
- Relación: Un **User** puede pertenecer a varios **Groups** (relación N:M).

2. Group (Grupo):

- Representa a un grupo de usuarios para un sorteo específico, con atributos como **id**, **name**, **adminId**, **budgetLimit**, **maxParticipants**, y **members**. Además, incluye métodos para agregar y eliminar miembros, y realizar el sorteo.
- Relación: Un **Group** puede tener muchos **Draws** y muchos **Users** asociados (relaciones 1:N).

3. Draw (Sorteo):

- Contiene información del sorteo, como **id**, **groupId**, **deadline**, y **exclusions** (exclusiones entre usuarios). Los métodos permiten gestionar las asignaciones y los sorteos en sí.
- Relación: Un **Draw** tiene muchos **Assignments** y **Messages**.

4. Assignment (Asignación):

- Relaciona a un usuario que regala con el usuario que recibe el regalo. Contiene atributos como **id**, **drawId**, **giverId**, y **receiverId**.
- Relación: Un **Assignment** está asociado a un único **User** como **giver** y **receiver** (relación 1:1).

5. Message (Mensaje):

- Representa un mensaje entre los participantes del sorteo. Los atributos incluyen **id**, **drawId**, **senderId**, **content**, **timestamp**, y **isAnonymous**.
- Relación: Los **Messages** están asociados a un **Draw** específico y son enviados por **Users**.

6. Wishlist (Lista de Deseos):

- Contiene los artículos deseados por un usuario para un sorteo determinado. Los atributos incluyen **id**, **userId**, **drawId**, y una lista de **items** (productos deseados).
- Relación: Un **User** tiene una única **Wishlist** asociada a cada **Draw** (relación 1:1).

7. Item (Artículo):

- Representa un artículo individual dentro de la **Wishlist** de un usuario, con atributos como **name** (nombre del artículo) y **url** (enlace al producto).
- Relación: Un **Item** pertenece a una **Wishlist** (relación 1:N).

8. Notification (Notificación):

- Representa una notificación enviada a un **User**. Contiene atributos como **id**, **userId**, **message**, y **isRead**.
- Relación: Una **Notification** está asociada a un único **User** (relación 1:1).

Clases de Servicios y Utilitarios:

1. AuthService (Servicio de Autenticación):

- Encapsula la lógica para la gestión de usuarios mediante **Firebase Authentication**, con métodos para autenticar usuarios y gestionar el registro.

2. FirebaseService (Servicio de Firebase):

- Encapsula las operaciones con la base de datos **Firebase Firestore**, gestionando la lectura y escritura de datos para las clases principales (por ejemplo, **Group**, **Draw**, **User**).

3. GroupUtils (Utilitarios de Grupo):

- Contiene métodos lógicos para gestionar las reglas de los sorteos, como la asignación de regalos y las exclusiones.

Relaciones entre las Clases:

- **User** → **Group**: Un **User** puede pertenecer a muchos **Groups**, y un **Group** puede tener muchos **Users** (relación N:M).
- **Group** → **Draw**: Un **Group** puede tener múltiples **Draws** (relación 1:N).
- **Draw** → **Assignment**: Un **Draw** tiene muchas **Assignments** que relacionan a los usuarios que regalan y los que reciben (relación 1:N).
- **Draw** → **Message**: Un **Draw** puede tener muchos **Messages** (relación 1:N).
- **User** → **Wishlist**: Un **User** tiene una única **Wishlist** para cada **Draw** (relación 1:1).
- **Wishlist** → **Item**: Una **Wishlist** contiene muchos **Items** (relación 1:N).
- **User** → **Notification**: Un **User** puede recibir muchas **Notifications** (relación 1:N).

*“La imagen del diagrama UML se encuentra (Figura C-5 en anexo)”

Resumen:

Este diagrama de clases refleja la estructura de datos y servicios necesarios para gestionar los sorteos de "Amigo Invisible". Las relaciones entre las clases están organizadas de manera eficiente para permitir una interacción fluida entre los usuarios, los grupos y los sorteos. Las clases de servicio, como **AuthService** y **FirebaseService**,

gestionan la autenticación y la comunicación con la base de datos, mientras que las clases de datos como **User**, **Group**, **Draw**, y **Assignment** gestionan las entidades clave del sistema.

4.3 Diagrama de Secuencia

El Diagrama de Secuencia muestra la interacción entre los diferentes objetos de la aplicación Amigo Invisible a lo largo del tiempo, describiendo cómo los mensajes y eventos fluyen entre los componentes en un escenario específico. En este apartado, se modelan dos escenarios clave: "Creación de un Sorteo" y "Asignación de un Regalo", detallando cómo interactúan los objetos clave del sistema: Usuario, Interfaz de Usuario (UI), API, Base de Datos, y Notificaciones.

Identificación de Escenarios:

Escenario principal: Creación de un Sorteo

1. **Usuario** interactúa con la **Interfaz de Usuario** para crear un nuevo sorteo, proporcionando detalles como el nombre del sorteo, las fechas de inicio y fin, y los participantes.
2. La **Interfaz de Usuario** toma esta información y la envía a la **API** para procesar la creación del sorteo.
3. La **API** guarda el sorteo en la **Base de Datos** (en este caso, Firebase).
4. Una vez el sorteo ha sido guardado correctamente, la **API** envía una **notificación** a los participantes del sorteo.
5. Finalmente, la **Interfaz de Usuario** muestra una confirmación al **Usuario**.

Escenario Secundario: Asignación de un Regalo

1. **API** obtiene los participantes del sorteo desde la **Base de Datos**.
2. **API** realiza la asignación de los participantes para el sorteo, estableciendo a quién le corresponde regalar a quién.
3. La **API** guarda las asignaciones en la **Base de Datos**.
4. **API** envía una **Notificación** a cada participante, informándoles de quién es el destinatario de su regalo.
5. **Notificaciones** confirman el envío de las notificaciones a los participantes.

Identificación de Objetos Participantes:

En los escenarios de creación de un sorteo y asignación de un regalo, los siguientes objetos participan activamente en el proceso:

- **Usuario:** El **Usuario** inicia el proceso de creación del sorteo y realiza las acciones para asignar los regalos.
- **Interfaz de Usuario (UI):** La **UI** recibe la entrada del **Usuario** y pasa la información a la **API**.
- **API:** La **API** procesa la creación del sorteo, guarda la información en la **Base de Datos** y maneja la asignación de regalos.
- **Base de Datos:** La **Base de Datos** almacena la información relacionada con el sorteo, los participantes y las asignaciones.
- **Notificaciones:** El servicio de **Notificaciones** envía alertas a los participantes del sorteo para mantenerlos informados.

Modelado de la Secuencia de Mensajes:

A continuación se describe la secuencia de mensajes que fluye entre los objetos durante los dos escenarios:

Creación de un Sorteo:

1. Usuario → Interfaz de Usuario (UI):

- Mensaje: crearSorteo(nombre: String, fechalinicio: Date, fechaFin: Date)
- Descripción: El Usuario interactúa con la Interfaz de Usuario para crear el sorteo, proporcionando la información necesaria.

2. Interfaz de Usuario (UI) → API:

- Mensaje: crearSorteo(datosSorteo)
- Descripción: La Interfaz de Usuario pasa los detalles del sorteo a la API para que procese la creación y la guarde en la Base de Datos.

3. API → Base de Datos:

- Mensaje: guardarSorteo(sorteo)
- Descripción: La API recibe los datos y realiza la operación de guardar el sorteo en la Base de Datos.

4. Base de Datos → API:

- Mensaje de retorno: sorteoGuardado()
- Descripción: La Base de Datos devuelve una confirmación a la API indicando que el sorteo ha sido guardado correctamente.

5. API → Notificaciones:

- Mensaje: enviarNotificacion(participantes, mensaje)
- Descripción: La API envía una Notificación a los participantes del sorteo para informarles sobre la creación del mismo.

6. Notificaciones → Usuario:

- Mensaje de retorno: notificaciónEnviada()
- Descripción: La Notificación se envía a los dispositivos de los participantes, confirmando que el sorteo ha sido creado y que deben estar atentos a sus asignaciones.

7. Interfaz de Usuario → Usuario:

- Mensaje: mostrarConfirmación(sorteoCreado)
- Descripción: Finalmente, la Interfaz de Usuario muestra un mensaje de confirmación al Usuario, indicándole que el sorteo ha sido creado correctamente.

Asignación de un Regalo:

1. API → Base de Datos:

- Mensaje: obtenerParticipantes()
- Descripción: La API obtiene la lista de participantes del sorteo desde la Base de Datos.

2. API → API:

- Mensaje: realizarAsignaciones()
- Descripción: La API asigna a los participantes para que cada uno sepa a quién debe regalar.

3. API → Base de Datos:

- Mensaje: guardarAsignaciones(asignaciones)
- Descripción: La API guarda las asignaciones de los regalos en la Base de Datos.

4. API → Notificaciones:

- Mensaje: enviarNotificacion(participantes, asignación)
- Descripción: La API envía una Notificación a cada participante informando a quién debe regalar.

5. Notificaciones → Usuario:

- Mensaje de retorno: notificaciónEnviada()
- Descripción: Las Notificaciones confirman que el mensaje ha sido enviado a los participantes.

Representación de la Línea de Vida:

Cada objeto involucrado en el proceso tiene su propia línea de vida, que es representada por una línea vertical que indica el tiempo de actividad de ese objeto. Las activaciones se representan mediante rectángulos colocados a lo largo de la línea de vida, mostrando los momentos en que cada objeto está procesando un mensaje.

- **Usuario:** La línea de vida del **Usuario** se activa cuando inicia la creación del sorteo.
- **Interfaz de Usuario (UI):** La **UI** tiene una activación cuando procesa los datos del sorteo y los pasa a la **API**.
- **API:** La **API** se activa cuando recibe los datos, guarda el sorteo en la **Base de Datos** y envía las notificaciones.
- **Base de Datos:** La **Base de Datos** se activa cuando guarda la información del sorteo y las asignaciones de los regalos.
- **Notificaciones:** El servicio de **Notificaciones** se activa para enviar las notificaciones a los participantes.

Creación del Diagrama de Secuencia:

*“La imagen del diagrama de Secuencia escenario “Sorteo” se encuentra (Figura C-6 en anexo)”

**“La imagen del diagrama de Secuencia escenario “Asignación Regalo” se encuentra (Figura C-7 en anexo)”

Resumen del Flujo:

1. El **Usuario** interactúa con la **Interfaz de Usuario** para crear un sorteo.
2. La **Interfaz de Usuario** pasa los datos del sorteo a la **API**.
3. La **API** guarda el sorteo en la **Base de Datos**.
4. Una vez guardado, la **API** envía una notificación a los participantes.
5. Las **Notificaciones** confirman que el mensaje ha sido enviado a los participantes.
6. En el escenario de **Asignación de un Regalo**, la **API** realiza las asignaciones y envía notificaciones a los participantes.

Este flujo muestra cómo se gestionan las interacciones entre los distintos componentes del sistema para llevar a cabo la creación de un sorteo y la asignación de regalos, manteniendo a los usuarios informados.

4.4 Diagrama de casos de uso

El **Diagrama de Casos de Uso** es una representación gráfica que describe las interacciones entre los usuarios y el sistema. Este diagrama es esencial para comprender los requisitos funcionales del sistema y proporciona una visión clara de cómo los usuarios utilizarán las funcionalidades de la aplicación.

En este apartado, se modela el caso de uso para el **Usuario** de la aplicación **Amigo Invisible**, detallando cómo este interactúa con el sistema para realizar las funciones más importantes, como la creación de sorteos, la visualización de resultados y la recepción de notificaciones.

Identificación de Actores

El **actor** en este sistema es el **Usuario**, quien interactúa con el sistema de diversas maneras. A continuación se identifican los actores principales:

- **Usuario:** El **Usuario** es el único actor en el sistema, quien se encarga de realizar todas las acciones principales dentro de la aplicación. Los usuarios pueden crear sorteos, ver los resultados de dichos sorteos, recibir notificaciones sobre las asignaciones de regalos, y gestionar su sesión en la aplicación.
- **Notificación (actor técnico):** Aunque no es un "usuario" convencional, el servicio de **Notificación** juega un papel importante como actor técnico. Este

actor interactúa con el sistema para enviar alertas y mensajes de notificación a los **Usuarios** informándoles sobre eventos importantes, como la creación de un sorteo o la asignación de un regalo.

Identificación de Casos de Uso

El actor principal del sistema es el **Usuario**, quien puede adoptar dos funciones distintas dentro del flujo de la aplicación:

- **Organizador**, encargado de crear sorteos, añadir participantes, definir reglas de exclusión y enviar notificaciones.
- **Participante**, quien accede a la asignación de regalos, recibe notificaciones automáticas y puede comunicarse de forma anónima.

Esta diferenciación se modela dentro del mismo actor, ya que no representa un tipo de cuenta distinto, sino una variación del comportamiento en función del rol asumido en cada sorteo.

A continuación se detallan los casos de uso principales de la aplicación **Amigo Invisible**:

Casos de Uso para el Usuario:

1. Iniciar sesión:

- **Descripción:** El **Usuario** accede a la aplicación ingresando sus credenciales.
- **Flujos Alternativos:** Si está habilitada, puede activar la autenticación de dos factores (2FA) como medida de seguridad adicional.

2. Crear Sorteo(Organizador):

- **Descripción:** El **Usuario** crea un sorteo, proporcionando información como el nombre del sorteo, importe del regalo, y los participantes.
- **Flujos Alternativos:Error:** Si los participantes no son suficientes, el sistema mostrará un mensaje de error y no permitirá la creación del sorteo.

3. Ver Resultados del Sorteo:

- **Descripción:** El **Usuario** consulta los resultados del sorteo para saber a quién debe regalar.
- **Flujos Alternativos:Error:** Si el sorteo no ha sido realizado o tiene algún error, el **Usuario** no podrá ver los resultados.

4. **Recibir Notificación de Asignación:**
 - **Descripción:** El **Usuario** recibe una notificación que le informa a quién debe regalar en el sorteo.
 - **Flujos Alternativos:Error:** Si no se han realizado las asignaciones, el **Usuario** no recibe la notificación.
5. **Enviar Mensaje Anónimo(Participante):**
 - **Descripción:** Permite enviar mensajes sin revelar la identidad al receptor, manteniendo el misterio del sorteo.
 - **Flujos Alternativos:Error:** Si el número de mensajes permitidos en la versión gratuita se supera, se notificará al usuario.
6. **Cerrar sesión:**
 - **Descripción:** El **Usuario** cierra sesión en la aplicación después de terminar su interacción.

Casos de Uso para el Actor Notificación:

1. **Enviar Notificación:**
 - **Descripción:** El servicio de **Notificaciones** envía un mensaje a los **Usuarios** informándoles sobre los eventos importantes, como la creación del sorteo o la asignación de regalos.
 - **Escenario Alternativo:Error:** Si la **API** no puede obtener los datos necesarios, la notificación no será enviada.

Establecimiento de Relaciones

En el diagrama de casos de uso, se deben establecer relaciones claras entre los actores y los casos de uso.

- **Usuario → Crear Sorteo:** El **Usuario** puede crear uno o más sorteos.
- **Usuario → Ver Resultados del Sorteo:** El **Usuario** puede ver los resultados del sorteo que ha creado o al que pertenece.
- **Usuario → Recibir Notificación:** El **Usuario** recibe notificaciones de asignación de regalos.
- **Usuario → Enviar Mensaje Anónimo:** El **Usuario** puede enviar mensajes anónimos con las personas del mismo grupo.

- **Usuario → Iniciar sesión / Cerrar sesión:** El **Usuario** puede iniciar y cerrar sesión para acceder al sistema.

En cuanto al actor **Notificación**, este se conecta con el caso de uso **Enviar Notificación** para informar a los **Usuarios** sobre los cambios o eventos que ocurren en el sistema (como la creación de sorteos o la asignación de regalos).

Definición de Inclusiones y Extensiones (Opcional)

Aunque no se necesitan extensiones complejas en este caso, se pueden utilizar relaciones de **inclusión** o **extensión** para modelar casos más detallados.

- El caso de uso **Iniciar sesión** puede incluir la autenticación de **2FA** como extensión, lo que significa que el **Usuario** puede optar por usar una capa adicional de seguridad al iniciar sesión.

Creación del Diagrama de Casos de Uso

*“La imagen del diagrama de Casos de Uso se encuentra (Figura C-8 en anexo)”

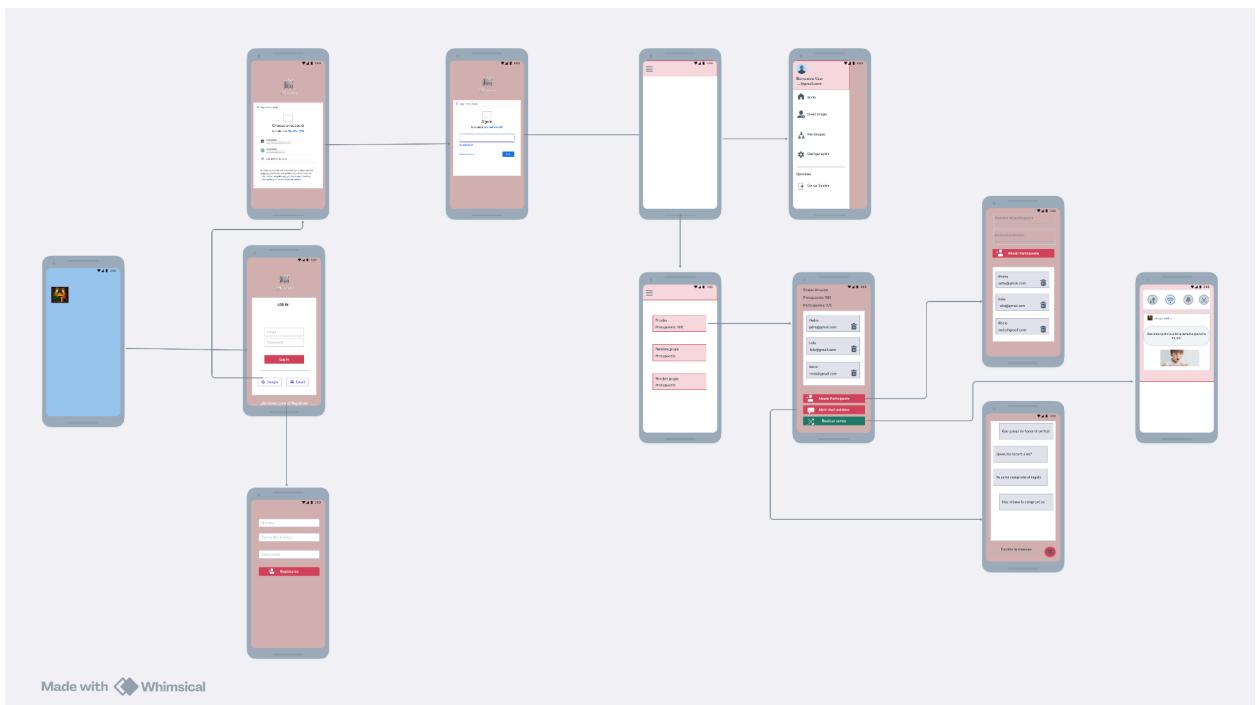
Resumen del Flujo

1. El **Usuario** interactúa con la **Interfaz de Usuario** para crear un sorteo, ver los resultados o gestionar su sesión.
2. La **Interfaz de Usuario** pasa los datos del sorteo a la **API**.
3. La **API** guarda el sorteo en la **Base de Datos**.
4. La **API** envía una notificación a los **Usuarios** informándoles sobre el sorteo y la asignación de regalos.
5. Las **Notificaciones** confirman que el mensaje ha sido enviado a los participantes.

Este flujo describe cómo se gestionan las interacciones entre los distintos componentes del sistema para llevar a cabo la creación de un sorteo, ver los resultados y asignar regalos, manteniendo a los **Usuarios** informados a través de notificaciones.

4.5 Mockup de la App(Amigo Invisible)

El **mockup** es una representación visual que muestra la interfaz de usuario (UI) de la aplicación **Amigo Invisible**, permitiendo visualizar cómo interactuarán los usuarios con la aplicación en diferentes dispositivos (smartphones y tablets) y en diversos idiomas. Es un paso esencial en el diseño de la experiencia de usuario (UX) y ayuda a mejorar la accesibilidad y la facilidad de uso de la app.

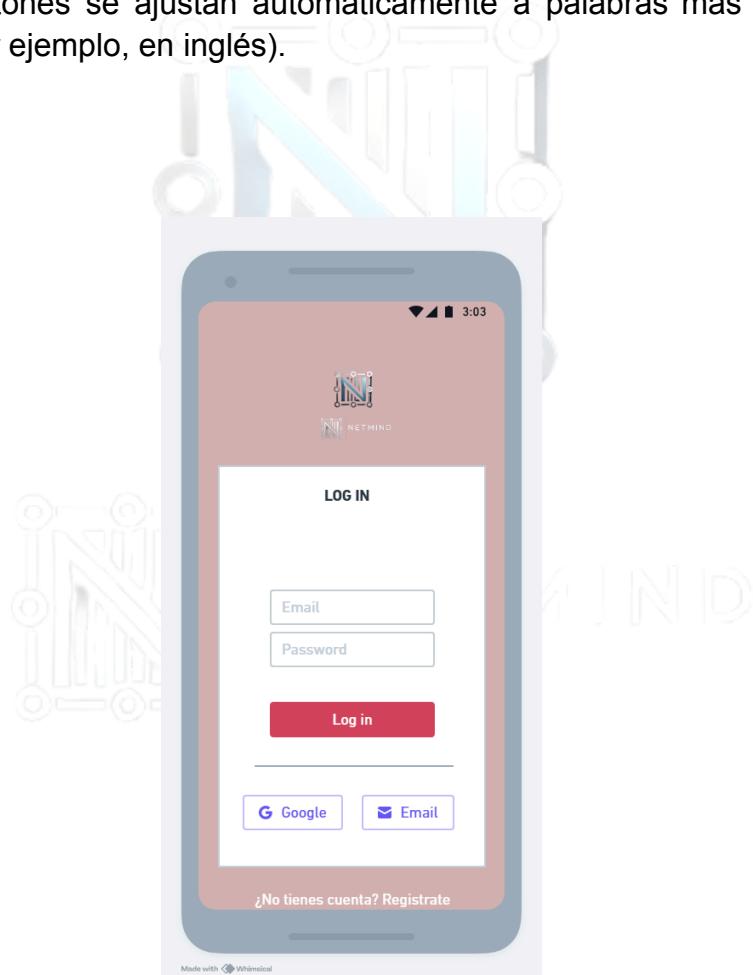


1. Pantalla de Inicio de Sesión / Registro

- Campos de texto: *Correo electrónico*, *Contraseña* (con etiquetas traducidas).
- Botones:
 - *Iniciar sesión*
 - *Registrarse* (multilingüe: "Log In" / "Sign Up")
 - *Iniciar sesión cuenta Google*
 - *Iniciar sesión con otra cuenta* (Outlook)

Adaptabilidad:

- Layout centrado, escalable para pantallas grandes (tablets).
- Inputs y botones se ajustan automáticamente a palabras más largas en otros idiomas (por ejemplo, en inglés).



2. Menú Drawer (desplegable lateral)

Una vez el usuario inicia sesión correctamente, accede a la aplicación principal, donde en la esquina superior izquierda se encuentra el **ícono de menú (hamburguesa)** que despliega un **Navigation Drawer** con las siguientes opciones:

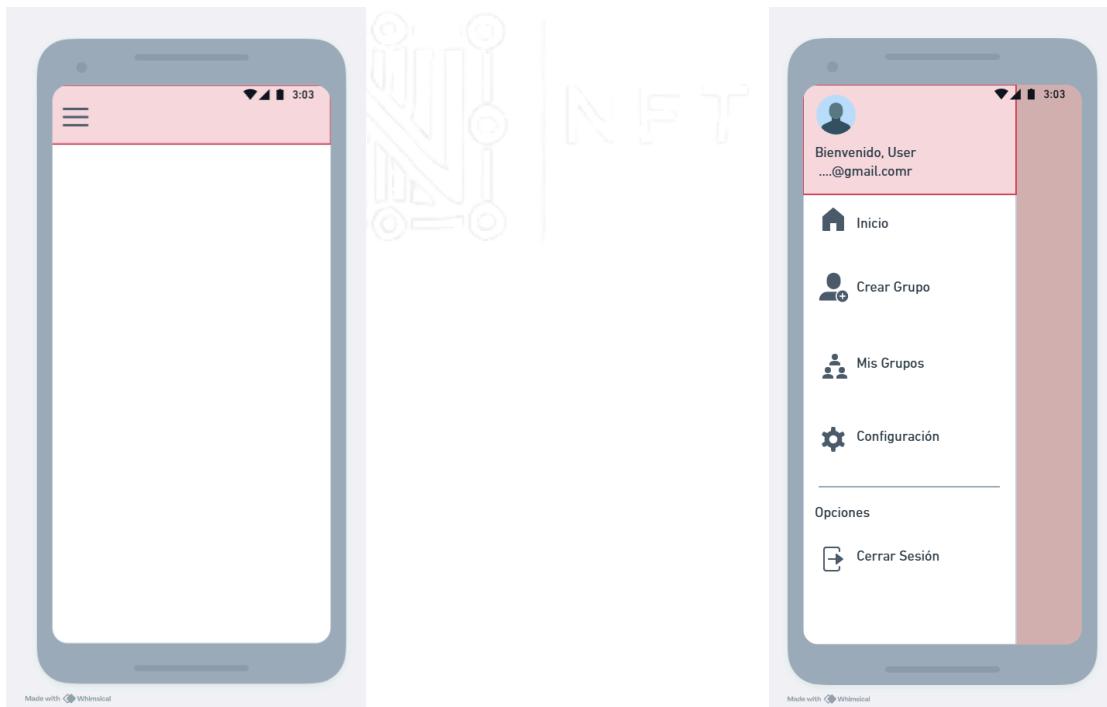
- **Inicio:** Redirige a la pantalla principal donde se listan los grupos.
- **Crear Sorteo:** Acceso directo al formulario para crear un nuevo sorteo.
- **Mis Grupos:** Listado detallado de los grupos a los que pertenece el usuario.
- **Configuración:** Acceso a la configuración personal de la aplicación (idioma, notificaciones, etc.).
- **Cerrar Sesión:** Opción ubicada en la parte inferior del Drawer para salir de la cuenta.

Personalización del perfil:

En la cabecera del Drawer, se muestra el **nombre del usuario, foto de perfil** (o avatar predeterminado) y correo electrónico asociado.

Adaptabilidad:

Drawer de tamaño ajustable, fluido en tablets y móviles.

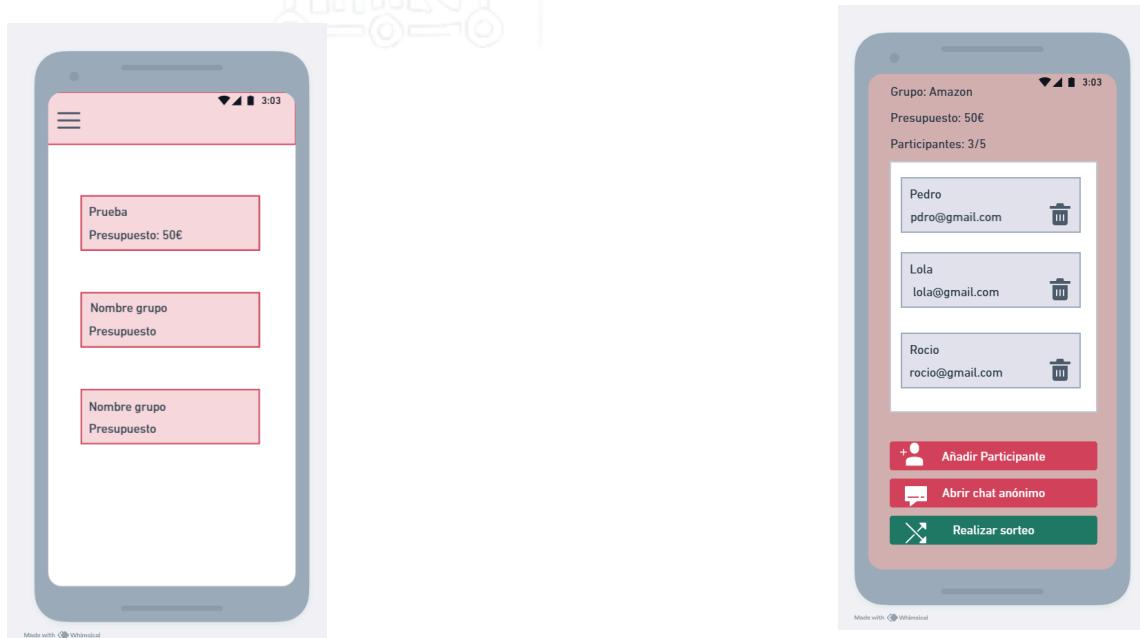


3. Pantalla Principal - Lista de Grupos

- **Vista en forma de tarjetas (Cards):** Cada tarjeta representa un grupo de Amigo Invisible al que el usuario pertenece.
- **Contenido de la tarjeta:**
 - Nombre del grupo.
 - Número de participantes.
 - Estado actual (por ejemplo, “Sorteo pendiente”, “Asignaciones realizadas”).
 - Foto asignada al grupo.

Acciones al pulsar en un grupo:

- Si el usuario es **Administrador del grupo:**
 - Puede **invitar nuevos participantes**.
 - Puede **iniciar el sorteo**.
 - Puede gestionar las exclusiones y reglas.
- Si el usuario es **Participante invitado:**
 - Puede **chatear de manera anónima** con su amigo invisible.
 - Puede **ver la lista de participantes** (sin saber quién es su amigo asignado).



4. Pantalla de Añadir miembros al grupo (Administrador)

- Campos de texto: *Nombre del participante* y *correo electrónico*.
- Botón de selección para añadir participantes.
- Recyclerview con la lista de participantes, sus datos y posibilidad de quitarlo del grupo.



5. Pantalla de Creación/ Resultados del Sorteo

- Campos de texto: *Nombre del sorteo*, *Presupuesto sugerido*, *Fecha límite*.
- Botón de selección para añadir participantes desde la lista de contactos internos.
- Botón de acción principal: *Realizar Sorteo*.
- Visualización privada del destinatario asignado.
- Opción para enviar mensajes anónimos.

6. Pantalla de Chat Anónimo

- Conversaciones agrupadas por grupo y sorteo.
- Posibilidad de enviar mensajes con identidad oculta.
- Límite de mensajes según tipo de cuenta (gratuita/premium).



7. Pantalla de Notificaciones

- Lista de notificaciones relevantes:
 - “¡Tienes un nuevo amigo invisible asignado!”



4.6 Guía de Estilos

La **Guía de Estilos** establece la base visual de la aplicación **Amigo Invisible**, asegurando **consistencia, claridad y una experiencia de usuario intuitiva** a través de todos los componentes de la interfaz, tanto en dispositivos móviles como tablets, y en distintos idiomas.

Paleta de Colores

Se han seleccionado colores vivos y contrastados para transmitir cercanía, dinamismo y facilidad de uso.

Elemento	Color	Código HEX	Uso
Fondo principal	Rojo suave	#D44646	Color de fondo en pantallas principales (pantalla de inicio, lista de grupos).
Botón de Log In / Registro	Rojo oscuro	#831515	Botones principales de inicio de sesión y registro.
Botón de Confirmar/Finalizar	Verde positivo	#4CAF50	Botones de acciones positivas (crear sorteo, finalizar sorteo).
Fondo secundario	Blanco	#FFFFFF	Fondo de tarjetas (cards) y áreas de contenido.
Texto principal	Gris oscuro	#333333	Textos principales para garantizar buena lectura.
Texto secundario	Gris medio	#666666	Subtítulos o descripciones secundarias.

Justificación de la elección de colores:

- **Rojo (#D44646)**: Asociado a la emoción, celebración y pasión (perfecto para representar el espíritu navideño del Amigo Invisible).
- **Verde (#4CAF50)**: Simboliza éxito, correcto o “ok” (ideal para finalizar acciones exitosas como confirmar el sorteo).
- **Contrastes altos**: Se garantizan altos niveles de contraste para asegurar **accesibilidad** y legibilidad, siguiendo buenas prácticas de UX.

Tipografía

Se ha elegido una fuente moderna, legible y ampliamente compatible en Android:

Elemento	Fuente	Tamaño	Peso	Uso
Títulos	Roboto Bold	24sp	Negrita	Pantallas principales, encabezados.
Subtítulos	Roboto Medium	18sp	Seminegrita	Secciones internas o descripciones cortas.
Texto Principal	Roboto Regular	16sp	Normal	Cuerpo de tarjetas, mensajes, listas.
Texto Secundario / Notas	Roboto Light	14sp	Ligero	Notificaciones, campos secundarios.

Justificación:

- **Roboto:** Fuente recomendada por Material Design de Google, altamente legible incluso en tamaños pequeños y adaptable a diferentes idiomas.
- **Claridad visual:** Se utilizan tamaños diferenciados para mantener jerarquía visual clara y facilitar la navegación rápida.

Iconografía

La iconografía utilizada sigue el **estilo Material Design**, con líneas simples y reconocibles de manera universal.

Elemento	Estilo de Icono	Tamaño	Color
Iconos de navegación	Material Icons Filled	24x24 px	Gris oscuro (#333333)
Icono menú hamburguesa	Material Design	24x24 px	Gris oscuro (#333333)
Iconos en botones (ej: añadir, enviar)	Material Design	20x20 px	Blanco (#FFFFFF) sobre fondo de botón
Iconos de estados (correcto/error)	Material Design	20x20 px	Verde (#4CAF50) / Rojo (#D44646)

Elementos de la Interfaz de Usuario (UI)

Componente	Estilo Definido
Botones principales (acciones positivas):	Fondo verde (#4CAF50), texto blanco, bordes ligeramente redondeados (8px), sombra ligera.
Botones de Login/Registro:	Fondo rojo oscuro (#831515), texto blanco, borde redondeado (8px).
Cards (Grupos):	Fondo blanco, borde suave gris claro (#E0E0E0), sombra suave.
Campos de texto:	Borde gris claro (#CCCCCC), icono opcional de ayuda dentro del campo, espacio entre campos de al menos 12px.
Drawer Menu:	Fondo blanco, iconos Material Design, encabezado con foto y nombre del usuario.
Barras de Navegación:	Fondo blanco o rojo claro, iconos grises, etiquetas traducidas.

Ejemplos de Aplicación de la Guía de Estilos

- **Pantalla de Inicio:** Fondo rojo claro (#D44646), botones de login en rojo oscuro (#831515).
- **Lista de Grupos:** Tarjetas blancas con textos en gris oscuro y botón de acción verde.
- **Chat Anónimo:** Burbujas de chat personalizadas, diferenciando mensajes enviados y recibidos por color suave.

4.7 Justificación de Decisiones de Diseño

La sección de justificación de diseño expone las razones detrás de las decisiones tomadas para la interfaz de usuario (UI) de la aplicación **Amigo Invisible**, asegurando que sea accesible, usable y adaptable en diferentes dispositivos e idiomas. Todas las decisiones se han basado en principios de diseño, necesidades del usuario, usabilidad y guías oficiales de diseño adaptativo.

Explicación de Decisiones de Diseño Clave

- **Disposición de elementos UI:**

Se eligió una estructura basada en layouts verticales y tarjetas (Cards) para listar los grupos, con menús laterales (Drawer) accesibles mediante un botón de menú hamburguesa. Esta disposición permite una navegación clara tanto en pantallas pequeñas como en tablets más grandes. El diseño es responsive, adaptándose a múltiples resoluciones y orientaciones.

- **Colores y Tipografía:**

La paleta de colores utiliza tonos cálidos y festivos como el rojo (#D44646) y verde (#4CAF50) para transmitir celebración y acción positiva. Se seleccionó Roboto como tipografía principal, por su excelente legibilidad incluso en idiomas de diferentes longitudes (por ejemplo, alemán o inglés).

- **Iconos e Imágenes:**

Se emplearon iconos Material Design simples y reconocibles para asegurar una interpretación intuitiva, independientemente del idioma o cultura. Los iconos son blancos sobre fondos de colores o grises para mantener claridad.

- **Navegación Multilingüe:**

Todos los botones y etiquetas fueron diseñados para contener textos traducibles. Se previó espacio adicional en botones y campos de texto para adaptarse a idiomas más largos, como el alemán.

- **Retroalimentación al usuario:**

Se diseñaron notificaciones claras y accesibles para informar al usuario sobre el éxito o fallo de acciones, empleando mensajes cortos, iconos de éxito/error y traducciones en tiempo real.

Referencia a Principios de Diseño

- **Consistencia:**

Se utilizó el mismo estilo visual (botones, tipografía, iconografía) en todas las pantallas, asegurando una experiencia uniforme (Principio de Consistencia).

- **Claridad y Jerarquía Visual:**

Gracias al uso de tamaños de fuente diferenciados y contrastes de color altos, se facilita la escaneabilidad rápida de la información importante.

- **Feedback Inmediato:**

Cada acción del usuario (como crear un sorteo, enviar un mensaje) proporciona un feedback visual inmediato, en forma de snackbar, alertas o cambios de pantalla (Principio de Retroalimentación).

- **Guías utilizadas:**

El diseño se alineó con las Material Design Guidelines de Google y se adaptó para soportar cambios dinámicos de idioma y accesibilidad.

Consideraciones de Usabilidad

- **Facilitación de tareas:**

El flujo de creación de sorteos minimiza los pasos necesarios para añadir participantes e iniciar un sorteo, todo accesible en pocos clics.

- **Minimización de errores:**

Se incluyen validaciones en formularios (como correos electrónicos válidos o límites de participantes) y mensajes de error traducidos, para evitar confusión en cualquier idioma.

- **Adaptabilidad Multidispositivo:**

La app está diseñada para ser plenamente funcional en móviles pequeños, pantallas medianas (phablets) y tablets, asegurando botones y textos accesibles a todos los tamaños de dedo/visión.

Necesidades del Usuario

- **Adaptabilidad cultural e idiomática:**

Pensando en usuarios de distintos países, se incorporó soporte multilingüe completo desde el inicio del diseño. Todo el contenido textual está preparado para ser traducido sin afectar el layout.

- **Preferencias del usuario:**

Se introdujo la personalización básica en el Drawer (mostrar nombre y foto de perfil) para aumentar el sentido de pertenencia y familiaridad.

- **Accesibilidad:**

Colores contrastados, fuentes legibles y controles de tamaño generoso permiten que personas con visión reducida o dificultades motoras también puedan interactuar fácilmente con la aplicación.

Ejemplos Concretos de Implementación

- **Botón de Crear Sorteo:**

Ubicado en la parte inferior derecha de la pantalla de grupos, utiliza un icono de "+" y un fondo verde (#4CAF50). El texto es traducible ("Crear" / "Create" / "Créer") y su posición sigue buenas prácticas para accesibilidad en móviles.

- **Menú Drawer:**

Incluye secciones claras ("Inicio", "Crear Sorteo", "Mis Grupos", "Configuración", "Cerrar Sesión"), todas traducibles y accesibles desde cualquier pantalla mediante un gesto o ícono.

- **Mensajes de error y éxito:**

Después de registrar un usuario o completar un sorteo, se presentan notificaciones breves adaptadas al idioma activo del sistema.

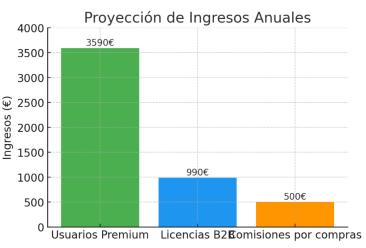
Referencias Utilizadas

- Material Design Guidelines, Google.
- Principios de UX/UI de Accesibilidad y Diseño Multilingüe, W3C Web Accessibility Initiative.
- Firebase UX Best Practices, documentación oficial



Documentos Anexos

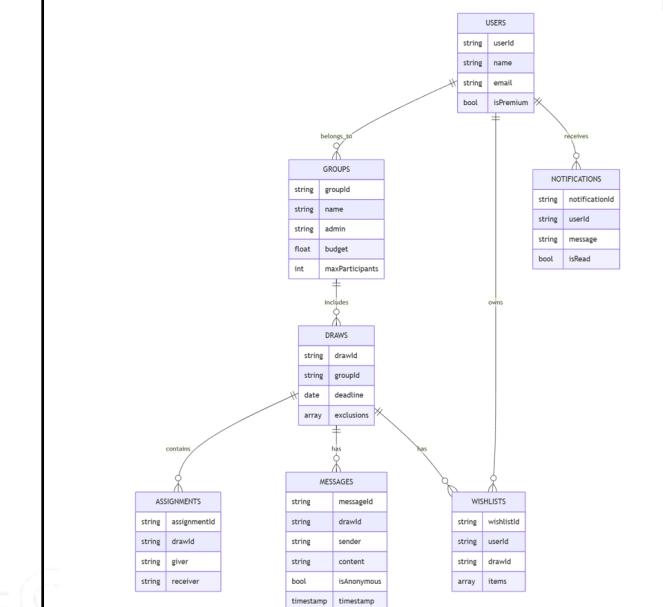
A	Logos
<p>A.1 <i>Logotipo corporativo de NetMind</i></p>	
<p>A.2 <i>Logotipo de la aplicación</i></p>	

B	Gráficos										
<p>B.1 <i>Gráfica de la distribución de Gastos Estimados</i></p>	 <table border="1"> <caption>Distribución de Gastos Estimados</caption> <thead> <tr> <th>Categoría</th> <th>Porcentaje</th> </tr> </thead> <tbody> <tr> <td>Asesoría legal</td> <td>38.2%</td> </tr> <tr> <td>Hosting y bases de datos</td> <td>27.5%</td> </tr> <tr> <td>Publicidad inicial</td> <td>22.9%</td> </tr> <tr> <td>Material promocional</td> <td>11.5%</td> </tr> </tbody> </table>	Categoría	Porcentaje	Asesoría legal	38.2%	Hosting y bases de datos	27.5%	Publicidad inicial	22.9%	Material promocional	11.5%
Categoría	Porcentaje										
Asesoría legal	38.2%										
Hosting y bases de datos	27.5%										
Publicidad inicial	22.9%										
Material promocional	11.5%										
<p>B.2 <i>Gráfica de Proyección de Ingresos Anuales</i></p>	 <table border="1"> <caption>Proyección de Ingresos Anuales</caption> <thead> <tr> <th>Categoría</th> <th>Ingresos (€)</th> </tr> </thead> <tbody> <tr> <td>Usuarios Premium</td> <td>3590€</td> </tr> <tr> <td>Licencias B2B</td> <td>990€</td> </tr> <tr> <td>Comisiones por compras</td> <td>500€</td> </tr> </tbody> </table>	Categoría	Ingresos (€)	Usuarios Premium	3590€	Licencias B2B	990€	Comisiones por compras	500€		
Categoría	Ingresos (€)										
Usuarios Premium	3590€										
Licencias B2B	990€										
Comisiones por compras	500€										

C	Diagramas
C.1 <i>Diagrama visual flujo</i>	<pre> graph TD LOGIN[LOGIN] --> WelcomeActivity[WelcomeActivity] WelcomeActivity --> NavigationDrawer[Navigation drawer] NavigationDrawer --> SignOut{Sign out} SignOut --> LOGIN NavigationDrawer --> MainFragment[MainFragment] NavigationDrawer --> CreateGroupFragment[CreateGroupFragment] NavigationDrawer --> ViewGroupsFragment[ViewGroupsFragment] </pre>
C.2 <i>Flujo actividad</i>	<p>Diagrama que ilustra la interacción entre una aplicación móvil Android y servicios de Google Cloud:</p> <ul style="list-style-type: none"> Un dispositivo móvil (App Móvil Android) se comunica con Google Cloud a través de HTTPS. Los servicios de Google Cloud incluyen: <ul style="list-style-type: none"> Firebase Authentication Cloud Firestore Cloud Messaging
C.3 <i>Diagrama de despliegue</i>	<p>Diagrama que ilustra la integración entre la consola de desarrollo de Firebase y los servicios de cliente:</p> <ul style="list-style-type: none"> Admin Dev: Utiliza la "Firebase Console Web" para: <ul style="list-style-type: none"> Gestión usuarios / reglas Admin datos Monitorización RTDB HTTPS / SDK Firebase Cliente: Utiliza un "Dispositivo Android App Instalada (.apk / .aab)" para: <ul style="list-style-type: none"> Lectura / Escritura Sincronización tiempo real Push notifications Integración: <ul style="list-style-type: none"> Cloud Firebase conecta la Consola de desarrollo con el Cliente. Firestore es utilizado tanto por la Consola de desarrollo como por el Cliente. Realtime Database es utilizado tanto por la Consola de desarrollo como por el Cliente. Firebase Authentication es utilizada tanto por la Consola de desarrollo como por el Cliente.

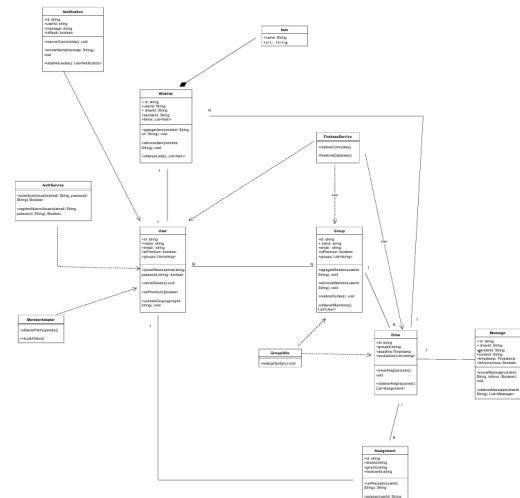
C.4

Diagrama de modelo noSql



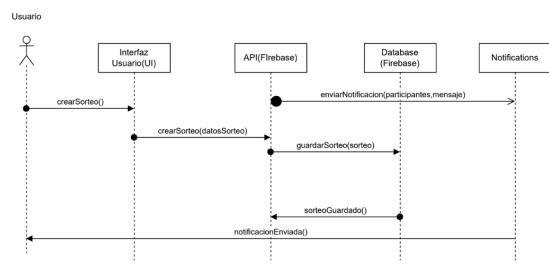
C.5

Diagrama UML Amigo Invisible



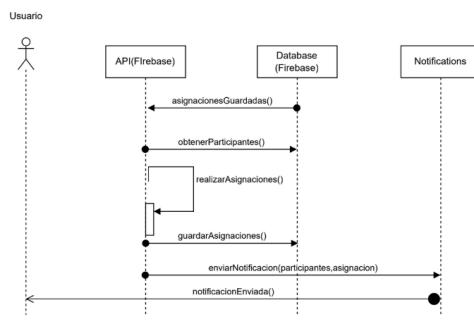
C.6

Diagrama de secuencia



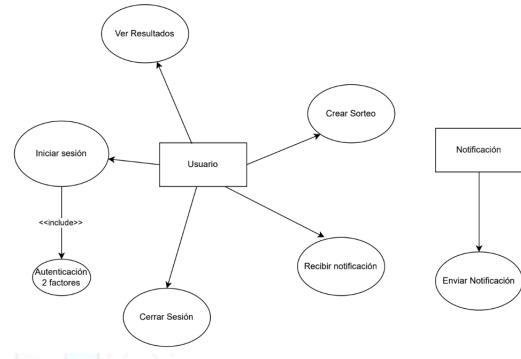
C.7

Diagrama de secuencia



C.8

Diagrama de casos de uso(Draw.io)



C.9

Diagrama de casos de uso(mermaid)

