



UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Aplicación web para la gestión de regalos

Autora: Susana González Pereira

Tutor: Santiago Alonso Villaverde

Junio 2023

Agradecimientos

Quiero agradecer a mis padres por el ánimo ofrecido a lo largo de mis estudios.

A mis compañeros de carrera, trabajo y máster a los que les espera un futuro brillante.

A todos los profesores del máster, que ponen todo su esfuerzo en elevar la calidad de nuestra profesión. Especialmente a Santiago, por su guía durante la tutorización de este trabajo.

Y a Iván, sin ti no hubiese sido posible.

Resumen

Existen numerosas ocasiones en las que alguien desea realizar un regalo a otra persona: cumpleaños, navidades, ocasiones especiales, amigos invisibles... El objetivo en cualquier caso es acertar con el regalo perfecto pero a la hora de la verdad no es tan fácil.

Para facilitar esta tarea, se ha realizado una web tipo red social que permite a los usuarios guardar listas de regalos deseados que sus círculos de amigos pueden ver. Los usuarios pueden decidir quién puede ver su lista de deseos y pueden limitar los datos que comparten con los demás.

Por otro lado, los demás usuarios pueden reservar un regalo de la lista de regalos, sin que el usuario principal se entere. Además la web incorpora una serie de herramientas que pueden ayudar a los usuarios y han sido definidas en las fases de diseño como:

- Guardar y compartir características de cada usuario como talla de ropa o calzado.
- Mensajería privada entre usuarios de la web.
- Recordatorios de fechas importantes de otros usuarios.
- Creación de sorteos de amigos invisibles.

Para todo ello, se ha realizado una aplicación basada en Angular Ionic, diseñada principalmente para ser accedida a través de un dispositivo móvil pero accesible desde cualquier navegador. Para dar soporte a esta web se ha creado un backend basado en la tecnología Spring y se han utilizado herramientas en la nube como Firebase para añadir funcionalidad a la web.

Además, se ha creado un entorno de integración y despliegue contínuo, que permite que la aplicación web esté disponible para cualquier usuario del mundo.

Todo el proyecto se ha gestionado a través de una metodología ágil, scrum. Se han realizado 4 iteraciones o sprints con las que se ha llegado a una versión avanzada de la aplicación.

Palabras clave

Red social, lista de deseos, Java, Spring, Angular, Firebase

Abstract

There are numerous occasions when someone wants to give a gift to another person: birthdays, Christmas, special occasions, secret Santa exchanges... The goal on every occasion is to choose the perfect gift, but when it comes down to it, it's not that easy.

To make this task easier, a social media-like website has been created. This web allows users to save wish lists with circles of friends. Users can decide who can view their wish lists and can limit the data they share with others.

On the other hand, other users can reserve a gift from the wish list without the main user finding out. Additionally, the website incorporates a set of tools that can assist users and have been defined in the design phases as follows:

- Saving and sharing user-specific characteristics such as clothing and shoe sizes.
- Private messaging between users on the website.
- Reminders of important dates for other users.
- Creation of secret pal drawings.

To achieve all of the above, an application based on Angular Ionic has been developed. The app is primarily designed to be accessed through a mobile device but it is still accessible from any browser. To support this website, a backend based on Spring technology has been created, and cloud tools like Firebase have been used to add functionality to the website.

Furthermore, a continuous integration and deployment environment has been set up, allowing the web application to be available to users worldwide.

The entire project has been managed using an agile methodology, Scrum. Four iterations or sprints have been carried out, resulting in an advanced version of the application.

Keywords

Social network, wish list, Java, Spring, Angular, Firebase

Tabla de contenidos

[Agradecimientos](#)

[Resumen](#)

[Palabras clave](#)

[Abstract](#)

[Keywords](#)

[Tabla de contenidos](#)

[Tabla de Figuras](#)

[Objetivos](#)

[Objetivos secundarios:](#)

[Inception Deck](#)

- [1. ¿Por qué estamos aquí?](#)
- [2. Crear un Elevator Pitch](#)
- [3. Diseñar una Caja de Producto](#)
- [4. Alcance del proyecto](#)
- [5. Conoce a tus vecinos](#)
- [6. Haz ver la solución](#)
- [7. Qué nos quita el sueño](#)
- [8. Tómale las medidas](#)
- [9. Ser claros en lo que vamos a dar](#)
- [10. Muestra lo que te va a costar](#)

[Product Backlog](#)

[Sprints](#)

[Sprint 1](#)

[Sprint planning](#)

[Sprint Review](#)

[Arquitectura Backend:](#)

[Arquitectura Frontend:](#)

[Entornos y CI/CD](#)

[Frontend](#)

[Backend](#)

[Tiempos](#)

[Sprint retrospective](#)

[Sprint 2](#)

[Sprint planning:](#)

[Sprint Review](#)

[Modelo de datos](#)

[Login, registro y perfil de usuario](#)

[Gestión de círculos, listas y amigos](#)

[Gestión de regalos](#)

[Sprint Retrospective](#)

[Sprint 3](#)

[Sprint planning](#)

[Sprint Review](#)

[Modelo de datos](#)

[Test](#)

[Características de usuarios](#)

[Mensajes privados](#)

[Sprint Retrospective](#)

[Sprint 4](#)

[Sprint planning](#)

[Sprint review](#)

[Modelo de datos](#)

[Recordatorios](#)

[Amigo invisible](#)

[Test](#)

[Análisis ZAP OWASP](#)

[Sprint retrospective](#)

[Tiempos del proyecto](#)

[Conclusiones](#)

[Posibles ampliaciones](#)

[Referencias](#)

Tabla de Figuras

<u>Figura 1. Imagen propuesta para la caja de producto</u>
<u>Figura 2. Stack tecnológico</u>
<u>Figura 3. Flujo de despliegue continuo</u>
<u>Figura 4. Diagrama de paquetes</u>
<u>Figura 5. Diagrama de paquetes frontend</u>
<u>Figura 6. Diagrama de paquetes frontend, detalle</u>
<u>Figura 7. Stack tecnológico</u>
<u>Figura 8. Lista de tareas del Sprint 1 en Github Projects</u>
<u>Figura 10. Modelo de datos Sprint 2</u>
<u>Figura 11. Pantalla de registro</u>
<u>Figura 12. Pantalla de usuario</u>
<u>Figura 13. Diagrama de clases frontend, detalle</u>
<u>Figura 14. Diagrama de clases backend, detalle</u>
<u>Figura 15. Entrypoint POST/users</u>
<u>Figura 16. Entrypoint GET /users/{id}</u>
<u>Figura 17. Diagrama de secuencia registro</u>
<u>Figura 18. Diagrama de secuencia inicio de sesión</u>
<u>Figura 19. Diagrama de secuencia visualización de perfil</u>
<u>Figura 20. Pantalla de lista de círculos y modal Nuevo círculo</u>
<u>Figura 21. Detalles de un círculo</u>
<u>Figura 22. Pantalla Mis listas y modal Nueva lista</u>
<u>Figura 23. Pantalla de perfil público</u>
<u>Figura 24. Diagrama de clases Angular, detalle</u>
<u>Figura 25. Diagrama de clases Backend, detalle</u>
<u>Figura 26. Entrypoint GET /circles</u>
<u>Figura 27. Entrypoint POST /circles</u>
<u>Figura 28. Entrypoint GET /circles/{id}</u>
<u>Figura 29. Entrypoint PUT /circles/{id}</u>
<u>Figura 30. Entrypoint PUT /circles/add/{id}</u>
<u>Figura 31. Entrypoint PUT /circles/delete/{id}</u>
<u>Figura 32. Entrypoint DELETE /circles</u>
<u>Figura 33. Entrypoint GET /lists</u>
<u>Figura 34. Entrypoint GET /lists/{id}</u>
<u>Figura 35. Entrypoint POST /lists</u>
<u>Figura 36. Entrypoint PUT /lists/{id}</u>
<u>Figura 37. Entrypoint PUT /lists/visible</u>
<u>Figura 38. Pantalla de lista de mis regalos</u>
<u>Figura 39. Pantalla de detalle de un regalo</u>
<u>Figura 40. Diagrama de clases de Angular</u>
<u>Figura 41. Datos de prueba en Firestore</u>
<u>Figura 42. Lista de tareas de Sprint 3 en Github Project</u>
<u>Figura 43. Modelo de datos Sprint 3</u>
<u>Figura 44. Anotaciones de la clase SeederDev</u>
<u>Figura 45. Estructura de las clases de Test</u>

[Figura 46. Cobertura de los Test](#)

[Figura 47. Características en la pantalla de usuario](#)

[Figura 48. Formulario de creación y edición de características](#)

[Figura 49. Características recomendadas de usuario en detalle de regalo](#)

[Figura 50. Diagrama de clases en Angular](#)

[Figura 51. Endpoints de características](#)

[Figura 52. Endpoints de características visibles. detalle](#)

[Figura 53. Endpoints de mensajes](#)

[Figura 54. Lista de tareas del Sprint 4](#)

[Figura 55. Modelo de datos en Sprint 4](#)

[Figura 56. Pantalla de usuario con recordatorios](#)

[Figura 57. Diagrama de clases en Angular](#)

[Figura 58. Endpoints creados para las historias de recordatorios](#)

[Figura 59. Pantalla detalle de círculos con modal de creación](#)

[Figura 60. Pantalla detalle de círculos con amigo invisible en curso](#)

[Figura 61. Diagrama de clases modificadas en Angular para las historias de amigo invisible](#)

[Figura 62. Entrypoints creados para las historias de amigo invisible](#)

[Figura 63. Cobertura final de los tests](#)

[Figura 64. Advertencia del navegador](#)

[Figura 65. Amenazas detectadas en recorrido manual](#)

[Figura 66. Amenazas detectadas con araña web](#)

[Figura 67. Escaneo en curso](#)

[Figura 68. Tiempo de desarrollo del proyecto backend](#)

[Figura 69. Tiempo de desarrollo del proyecto frontend](#)

Objetivos

- Desarrollo de un frontal web con tecnología Angular que permita gestionar regalos deseados de los usuarios y la gestión de sus amistades.
- Desarrollo de un API Rest que da soporte a las necesidades del portal. Este backend estará programado en Java Spring y su capa de persistencia se basará en PostgreSQL.
- Creación de un entorno de trabajo integrado con herramientas integración y despliegue continuo (Github Actions). Éste, permitirá el despliegue de los artefactos en un entorno productivo en la nube para que esté disponible con una capa gratuita.
- Aplicación y ampliación de los conocimientos adquiridos en el máster: patrones de diseño, buenas prácticas en el desarrollo software, uso de tecnologías concretas (Angular, Spring), gestión de proyectos, etc.

Objetivos secundarios:

- Gestión completa de un proyecto software con una metodología ágil, asegurándonos de respetar el proceso de ingeniería en todo momento.
 - Estudio de métricas del proceso de desarrollo (calidad, tiempo de desarrollo)
- Análisis y mitigación de vulnerabilidades de la seguridad de la web.
- Breve estudio de las opciones de mercado para publicar una web de forma gratuita en la nube.

Inception Deck

“Many Agile projects are doomed to fail before they even begin”

Jonathan Ramusson

La importancia de tener los objetivos y metas de un sistema software claros es vital para la correcta finalización del mismo. A menudo, muchas personas forman parte de un proyecto y no todas las personas comparten la misma visión. Esta disparidad en la visión de objetivos lastra la ejecución del proyecto y el producto final puede decepcionar a alguna de las personas involucradas.

Para unificar la visión del proyecto Jonathan Rasmusson propone la técnica de Inception Deck en sus publicaciones. Se trata de una técnica cuyo objetivo es garantizar que todo el mundo involucrado está en la misma página en relación a los objetivos y metas del proyecto.

La técnica de Inception Deck, creada originalmente en el contexto del desarrollo ágil por Robin Gibson, trata de realizar una serie de diez ejercicios. En su conjunto, estos ejercicios ayudarán a definir las expectativas del proyecto, asegurarse de que todas las personas involucradas están en la misma página, resolver conflictos sobre metas u objetivos y conseguir que las personas relacionadas con el proyecto se involucren y comprometan con el proyecto.

1. ¿Por qué estamos aquí?

Hacer un regalo a un pariente, conocido o ser querido es una forma de ampliar los vínculos entre las personas involucradas.

En el estudio antropológico *Gift Giving in Anthropological Perspective* [1], J.F Sherry establece tres fases que los humanos pasan a la hora de enfrentarse a obsequiar:

1. **Gestation Stage:** La persona piensa en un regalo teniendo en cuenta diferentes factores:
 - a. El motivo del regalo.
 - b. Coste económico y el simbólico del obsequio.
 - c. Relación con la persona a la que se hace el regalo:
 - i. Íntima vs Distante
 - ii. Duración de la relación
 - d. Ideas provocadas por la persona a regalar:
 - i. Intencionales:
 1. Pistas
 2. Peticiones expresas
 3. Regalos recibidos anteriormente de esa persona
 - ii. No intencionales:
 1. Forma de ser
 2. Carisma
 3. Apariencia
2. **Presentation Stage:** La persona entrega el regalo en un momento, lugar y modo adecuado al regalo.

3. **Reformulation Stage:** La persona a la que ofrecen el regalo muestra una reacción al regalo.
 - a. Positivamente:
 - i. Consumiendo el producto.
 - ii. Mostrándolo públicamente.
 - b. Negativamente:
 - i. Almacenando el producto.
 - ii. Intercambiando el producto: devolverlo o dárselo a otra persona.
 - iii. Rechazándolo.

Tras este intercambio es habitual que los roles se inviertan y posteriormente la persona que ha recibido el regalo sea la que haga un nuevo regalo permitiendo que una persona pueda realizar ambas acciones: regalar y recibir un regalo.

El objetivo de este proyecto es facilitar la fase de gestación de los regalos de cualquier persona permitiendo a los receptores dar pistas a sus conocidos sobre regalos que les gustarían de forma fácil y concisa maximizando el éxito de la transacción. Esto será posible gracias a una aplicación web que permitirá a los usuarios gestionar listas de deseos y listas de amigos que podrán ver sus datos. Los usuarios regaladores podrán reservar regalos de manera anónima para la persona que recibe el regalo para evitar duplicidades en los regalos.

El sistema, además, podrá recordar a los usuarios fechas importantes con avisos para que no se olviden de los regalos.

Adicionalmente, el sistema software podrá mostrar ideas de regalo para cada usuario basándose en encuestas en sus perfiles y ofrecerá filtros que permitan al usuario encontrar regalos adecuados en relación a precios, motivo del regalo, pertinencia de la fecha del año etc.

2. Crear un Elevator Pitch

“¿Alguna vez has recibido un regalo que no te ha gustado? ¿Has hecho algún regalo que no ha sido bien recibido? Para todas aquellas personas que dan y reciben regalos la web propuesta facilitará la tarea de idear un regalo exitoso que te ayude a decidir rápidamente en fechas decisivas.

Con la web propuesta podrás acordarte de las fechas importantes de tus conocidos a través de avisos personalizados y conocer los regalos perfectos que esperan las personas de tu alrededor. Acertarás qué talla comprar o qué color le gusta más al usuario. No tendrán que ir a la tienda a descambiarlo. Además, podrás reservar regalos de su lista de deseos para que nadie pise tu regalo perfecto.

Podrás marcar tu propia lista de deseos y compartirla solamente con quien tú quieras gestionando diferentes círculos de amigos. Además podrás organizar y gestionar fácilmente amigos invisibles sin que nadie se olvide de su amigo o se equivoque de regalo.

En definitiva, esta web te permitirá gestionar los regalos de tus familiares y amigos de forma sencilla, privada y eficaz”.

3. Diseñar una Caja de Producto

3 frases importantes

- ¡No más regalos duplicados!
- ¡No más devoluciones!
- Recibirás regalos con valor para ti.

Lema:

- ¡Acierta siempre con tus regalos!

Imagen:



Figura 1. Imagen propuesta para la caja de producto [2]

4. Alcance del proyecto

Es importante dejar claras las expectativas del proyecto para que todos los involucrados estén en la misma página. A continuación se detallan características del sistema en tres

grupos. Características que están dentro del alcance, características que están fuera del alcance y características que podrían estar en el alcance si fuese posible.

Dentro del alcance:

- **Gestión de usuarios:** Cada usuario podrá darse de alta, baja, modificar sus datos personales y consultar sus datos personales.
- **Gestión de círculos de amistades.** Cada usuario podrá crear/consultar/actualizar/borrar N círculos de amistades e incluir vía petición de amistad a otros usuarios de la plataforma. Por defecto los usuarios tendrán un círculo de amistad denominado “amigos” en el que estarán todas las personas con las que haya compartido una petición de amistad.
- **Gestión de listas de deseos:** Cada usuario podrá crear N listas de deseos y compartirlas con N círculos de amistad.
- **Gestión de datos adicionales y privacidad:** Cada usuario podrá publicar datos de interés como tallas de ropa, calzado o colores favoritos. Cada dato adicional podrá ser configurado para visualizarse por círculo de amigos.
- **Gestión de amigos invisibles:** Un usuario podrá crear dentro de un círculo de amigos un juego del amigo invisible. Cada usuario del círculo recibirá el nombre de un amigo y entre todos se harán un regalo. Cada usuario podrá ver en todo momento quién es la persona a la que tienen que regalar y una descripción generada por el usuario que ha creado el juego.
- **Avisos de eventos importantes:** Cada usuario podrá recibir avisos de sus círculos de fechas importantes como cumpleaños. Estos avisos se podrán marcar como leídos.

Fuera del alcance:

- **Recompensas por realizar regalos:** los regalos son incondicionales, no se espera nada a cambio en la aplicación como en sitios como Patreon.
- **Chat en directo entre usuarios:** aunque los usuarios sí podrían compartir mensajes no se trataría de un sistema de chat.

Sin decidir:

- **Mejores precios:** Incorporación de APIs de sitios de venta que comparen los precios de un producto.
- **Regalos en conjunto:** Más de una persona puede compartir los gastos de un regalo a la hora de reservarlo.
- **Sistema de mensajería:** Mensajes entre usuarios y mensajes colectivos entre usuarios de círculos. Este sistema de mensajes es offline, como un foro, no un chat en tiempo real.
- **Mejor amigo:** Un usuario podrá establecer a un mejor amigo que responda dudas sobre la persona principal referentes a un regalo: color, talla, idoneidad...
- **Preguntas anónimas:** Un usuario puede hacerle preguntas anónimas a alguien de su círculo referentes a un regalo.

5. Conoce a tus vecinos

Es importante identificar a todas las personas que estarán involucradas a lo largo de todo el ciclo de vida de la solución.

Equipo de desarrollo:

El equipo de desarrollo estará compuesto por

- Un ingeniero informático senior (arquitecto) que se encargará de diseñar las historias de usuario y de la toma de decisiones técnicas.
- Un desarrollador junior que deberá desarrollar las historias de usuario siguiendo las guías técnicas dispuestas por el arquitecto.

Equipo de diseño:

- Deberá existir una persona especialista en UI/UX encargada de diseñar la aplicación prestando atención a parámetros de usabilidad y experiencia de usuario.

Equipo de sistemas:

- Deberá existir una persona especialista en sistemas que de soporte a la gestión de los sistemas en nube que se van a necesitar en este proyecto. Esta persona deberá velar por la disponibilidad de la web y la realización de copias de seguridad de las bases de datos.

Equipo de marketing:

- Deberá existir una persona especialista en marketing que se encargue de realizar acciones que den a conocer la aplicación a usuarios potenciales. Podrá además recoger feedback de los usuarios y plantearlos al ingeniero encargado del desarrollo.

Equipo de soporte:

- Deberá existir una persona que responda a las dudas de uso de los usuarios, así como prestar atención a las notificaciones de uso indebido de la aplicación.

Usuarios:

- Son los usuarios finales de la aplicación. Podrán necesitar soporte o enviar feedback para la web.
- Adicionalmente en este proyecto pueden estar involucrados voluntarios beta-testers que quieran probar la web como usuarios finales y responder cuestionarios sobre el funcionamiento de la aplicación.

6. Haz ver la solución

La aplicación es un portal web que podrá ser visitado a través de cualquier navegador. El usuario interactúa con un frontal web que se conecta con diferentes entornos backend para poder crear, actualizar o borrar datos.

En la Figura 2 se muestra el stack tecnológico que se va a utilizar en el proyecto. Se incluyen las urls de cada uno de los componentes respectivamente en los entornos de desarrollo ("dev") y producción("prod").

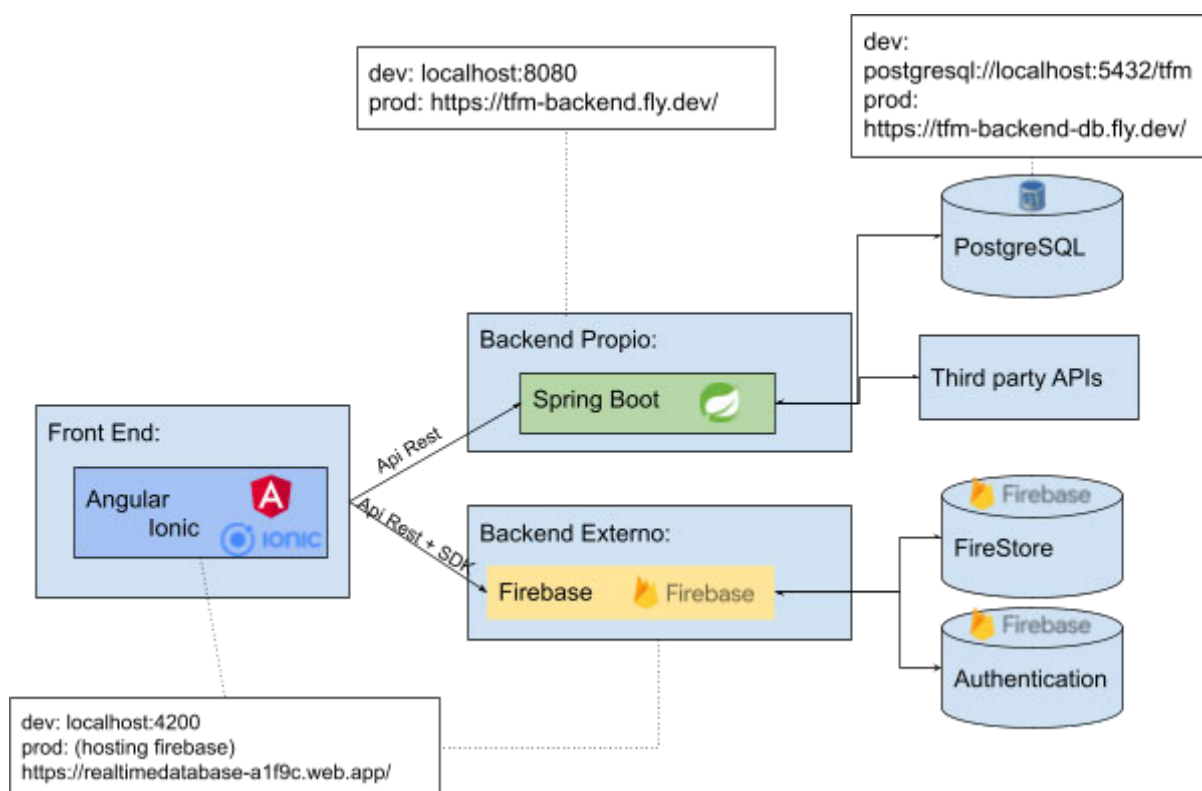


Figura 2. Stack tecnológico

- **Frontend:** Angular[3] (con ionic[4]): Para la web, como tecnología front-end se utilizará Angular (Typescript). Adicionalmente se utilizará Ionic que permite diseñar fácilmente una web adaptativa a cualquier dispositivo.
- **Backend:** Spring Boot[5]. Para dar soporte a las necesidades de la web se creará un backend en Java que utilice el framework Spring. Este backend habilitará un API Rest para permitir la comunicación desde el FrontEnd.
- **Backend:** Firebase [6]: Es una plataforma de desarrollo de aplicaciones móviles y web de Google que proporciona herramientas y servicios en la nube para simplificar la creación de aplicaciones. Se utilizará la plataforma para autenticar a los usuarios, incluyendo un login social y se utilizará una base de datos que permitirá tener sincronizados los datos en tiempo real para todos los usuarios.
- **Bases de datos:**
 - PostgreSQL [7] [8] [9]: Se usará esta base de datos para almacenar datos relativos a las preferencias de usuarios, sus círculos de amigos, mensajes, y avisos.
 - Firebase Cloud Firestore [10]: se usará esta base de datos para almacenar las listas de deseos de los usuarios.

Para la gestión de versiones de la aplicación se utilizará Git. Se crearán dos repositorios, uno para el proyecto backend y otro para el proyecto angular.

Se utilizará GitHub [11] como repositorio remoto. Para la gestión del proyecto ágil, GitHub proporciona la herramienta "Project"[12] en la que se podrán crear las historias de usuarios y gestionar las diferentes iteraciones.

En cada uno de los proyectos se llevará un flujo ramificado del desarrollo. Por cada historia de usuario o tarea técnica se creará un ticket y una rama. Cuando el desarrollo se considera estable, el código se debe unificar con la rama “master”.

En Github Actions[13], el gestor de integración continua de GitHub, se creará un flujo de integración continua por repositorio. Cada flujo se disparará al unificarse la rama máster y construirá y publicará el código en su respectivo hosting.

En la Figura 3 se muestra el proceso de despliegue continuo de un código que se sube a la rama máster.

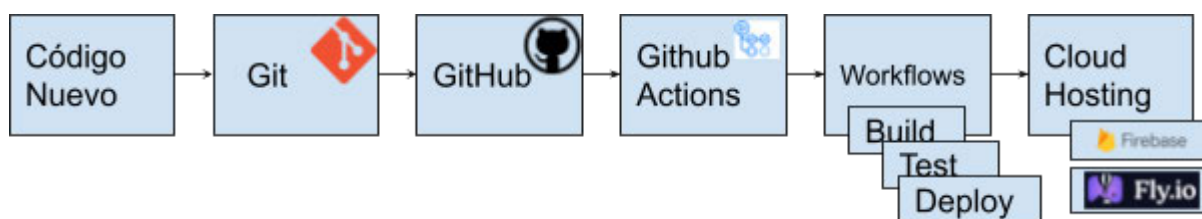


Figura 3. Flujo de despliegue continuo

7. Qué nos quita el sueño

- Riesgos en los que podemos influir
 - Tiempo de desarrollo. El tiempo de desarrollo de este proyecto es limitado: 3 meses. En caso de incurrir en algún bug persistente o si la definición es incorrecta, no habría margen de tiempo para arreglarlo. Se deberán estimar las tareas correctamente y analizar el alcance de cada sprint de forma correcta.
 - Especificación incompleta: Detectar los requisitos de forma completa es una tarea difícil, así mismo, un error en la especificación puede llevar a problemas graves de diseño difíciles de solucionar en iteraciones posteriores.
- Riesgos en los que no podemos influir
 - Cambios en la tecnología de terceros: si se incorpora una API o se usan servicios en la nube, es posible que surjan errores por discontinuidad de la tecnología provista por terceros. Para paliar esta situación, se pueden tener en mente más de un proveedor, pero, especialmente al buscar capas gratuitas de los proveedores, nada garantiza que sigan funcionando.

8. Tómale las medidas

Estimación del tiempo que se va a dedicar a cada tarea y el valor que da la tarea al proyecto. Esto es una estimación, no un compromiso.

Elemento	Tipo	Valor	Esfuerzo
Gestión de BD	Técnico	200	15

Gestión de CI/CD	Técnico	100	10
QA	Técnico	50	10
Gestión del proyecto	Técnico	200	10
Gestión de usuarios	Historia de usuario	300	15
Gestión de círculos	Historia de usuario	300	15
Gestión de listas de deseos	Historia de usuario	500	10
Gestión de datos adicionales y fechas	Historia de usuario	50	10
Gestión de amigos invisibles	Historia de usuario	50	15
Gestión de avisos importantes	Historia de usuario	50	15
Mejores precios	Historia de usuario	30	20
Regalos en conjunto	Historia de usuario	30	8
Sistema de mensajería	Historia de usuario	100	15
Mejor amigo	Historia de usuario	30	15
Preguntas anónimas	Historia de usuario	30	15

9. Ser claros en lo que vamos a dar

Lo ideal en un proyecto es que esté hecho en tiempo, con la máxima calidad, con un presupuesto bajo y con todas las prestaciones posibles. Sin embargo, si las cosas fuesen mal y tuviésemos que descartar alguna de nuestras opciones, ¿en qué estamos dispuestos a negociar?

En la siguiente tabla se exponen los factores clave de un proyecto junto a un factor de flexibilidad. Siendo:

0: Nada flexible, no lo podemos cambiar es esencial para la consecución del proyecto.

10: Muy flexible, cambiemos esto si molesta para la consecución del proyecto.

Factores clave	Flexibilidad	Razón
Tiempo	1	No podemos entregar el TFM tarde.
Alcance	10	Podemos quitar algunas historias del alcance si no añaden valor al proyecto como TFM

Presupuesto	1	El presupuesto del proyecto es muy ajustado
Calidad	0	La calidad del producto es innegociable, siendo el alcance o el tiempo factores que se tendrán que cambiar para no afectar a la calidad
Facilidad de uso	4	Es preferible tener una buena facilidad de uso, pero la funcionalidad está por encima de esta característica
Seguridad	3	Preferiblemente deberíamos tener seguridad máxima pero en este sistema podremos ser un poco flexibles
Rendimiento	2	Preferiblemente deberíamos tener rendimiento y escalabilidad del sistema. Sin embargo este sistema no va a escalar por lo que podríamos ser algo flexibles en este sentido.

10. Muestra lo que te va a costar

El proyecto tiene los siguientes costes estimados:

- **Personal:**

En la siguiente tabla se muestra una estimación de los salarios del personal de cada posición.

Equipo	Posición	Salario bruto anual empleado	Coste bruto mensual empresa	Meses para el proyecto	Coste para el proyecto
Desarrollo	Senior	30.000€	3.330€	2	6.660€
Diseño	Especialista UX	20.000€	2.220€	0.5	1.110€
Sistemas	Técnico de sistemas	18.000€	2.000€	0.5	1.000€
				TOTAL	8.770€

La columna “Coste Bruto Mensual Empresa” incluye los impuestos calculados por la herramienta de la empresa FactorialHR [14] especializada en gestión de recursos humanos.

No todos los roles son necesarios en todas las etapas del proyecto, por ello en la columna “Meses para el proyecto” estipula los meses en los que se va a necesitar la posición.

Los desarrolladores trabajarán durante todo el proyecto. El especialista UX y el técnico de sistemas deberán trabajar un mes al principio del proyecto. El desarrollador trabajará después del diseño.

Tras la entrega del proyecto, es esperable que solamente se necesite un desarrollador “mid” que lleve a cabo las tareas de mantenimiento. No será necesario el especialista UX y sólo puntualmente se necesitará al técnico de sistemas.

A cambio, existirán los perfiles encargados de soporte y marketing deberán estar presentes tras la entrega del proyecto, pero prácticamente no tendrán esfuerzos durante el desarrollo.

- **Infraestructura y máquinas:**

Item	Precio/mes	Cantidad	Total
Ordenador alto rendimiento	60€	3	180€
		TOTAL:	180 €

Los precios de los ordenadores son precios estándar de alquiler obtenidos en la plataforma Grover [15].

- **Totales para tres meses:**

Item	Precio
Recursos Humanos	8.770 €
Infraestructuras	180 €
TOTAL 3 meses	8.950 €

Product Backlog

El product backlog incluye la lista de historias de usuario e historias técnicas que son necesarias para el producto software. A lo largo del proyecto el product backlog puede cambiar: pueden aparecer nuevas historias de usuario o cambiar su especificación o prioridad.

Identificador	Título
TS01	Crear el proyecto Spring
TS02	Crear workflow para integración continua para Spring
TS03	Crear proyecto Angular
TS04	Crear workflow para integración continua para Angular
TS05	Crear autenticación en Firebase
TS06	Crear base de datos Cloud FireStore
TS07	Crear base de datos PostgreSQL
TS09	Análisis ZAP OWASP
US01	Registro de usuarios
US02	Inicio y cierre de sesión de usuarios
US03	Consultar datos de usuario
US04	Crear deseo de regalo
US05	Consultar deseo de regalo
US06	Actualizar deseo de regalo
US07	Borrar deseo de regalo
US08	Consultar lista de deseos
US09	Reservar un regalo
US10	Deshacer reserva de regalo
US11	Crear círculo
US12	Consultar detalles de círculo
US13	Actualizar detalles de círculo
US14	Borrar círculo

US15	Añadir persona a círculo
US16	Consultar personas de círculo
US17	Borrar persona de círculo
US18	Consultar detalles de persona
US19	Añadir característica de usuario
US20	Consultar características de usuario
US21	Actualizar característica de usuario
US22	Borrar característica de usuario
US23	Consultar recordatorios
US24	Descartar recordatorio
US25	Crear mensaje
US26	Consultar mensajes recibidos
US27	Leer mensaje
US28	Responder mensaje
US29	Hacer una pregunta anónima
US30	Consultar preguntas recibidas
US31	Responder preguntas
US32	Asignar mejor amigo
US33	Crear amigo invisible
US34	Consultar amigo invisible
US35	Finalizar amigo invisible

A continuación se muestran los datos de cada una de las historias técnicas o de usuario con su descripción, el valor y la prioridad que tienen en el proyecto.

Identificador	TS01
Título	Crear el proyecto Spring
Descripción	Como desarrolladora quiero configurar un proyecto base con Java Spring para poder añadir funcionalidad fácilmente
Valor	2 SP

Prioridad	Muy alta
-----------	----------

Identificador	TS02
Título	Crear workflow para integración continua para Spring
Descripción	Como desarrolladora quiero configurar un workflow de Github Actions que se dispare cada vez que subo código a la rama máster y que despliegue el proyecto en fly.io para que el proceso de despliegue sea automático
Valor	3 SP
Prioridad	Alta

Identificador	TS03
Título	Crear el proyecto en Angular
Descripción	Como desarrolladora quiero configurar un proyecto base con las tecnologías Angular e Ionic para poder evolucionarlo posteriormente
Valor	2 SP
Prioridad	Muy alta

Identificador	TS04
Título	Crear workflow para integración continua para Angular
Descripción	Como desarrolladora quiero configurar un workflow en GitHub Actions que me permita desplegar automáticamente la aplicación a Firebase para que mis despliegues sean automáticos
Valor	3 SP
Prioridad	Alta

Identificador	TS05
Título	Crear autenticación en Firebase

Descripción	Como desarrolladora quiero configurar un proyecto en Firebase Authentication y conectarlo con el proyecto de Angular para poder tener login social
Valor	2 SP
Prioridad	Alta

Identificador	TS06
Título	Crear base de datos Cloud FireStore
Descripción	Como desarrolladora quiero configurar un proyecto en Firebase Cloud FireStore y conectarlo con el proyecto de Angular para poder guardar datos de la aplicación
Valor	3 SP
Prioridad	Muy Alta

Identificador	TS07
Título	Crear base de datos PostgreSQL
Descripción	Como desarrolladora quiero configurar una base de datos PostgreSQL y conectarla con el proyecto de Spring para poder guardar datos de la aplicación
Valor	3 SP
Prioridad	Muy Alta

Identificador	TS09
Título	Análisis ZAP OWASP
Descripción	Como desarrolladora quiero realizar un análisis de seguridad de la solución para poder mitigarlos en el futuro
Valor	5 SP
Prioridad	Muy Baja

Identificador	US01
---------------	------

Título	Registro de usuarios
Descripción	Como usuario quiero poder crear una cuenta para poder acceder a la web
Valor	3 SP
Prioridad	Alta

Identificador	US02
Título	Inicio y cierre de sesión de usuarios
Descripción	Como usuario quiero poder iniciar y cerrar sesión en la web para poder entrar a mi cuenta
Valor	3 SP
Prioridad	Alta

Identificador	US03
Título	Consultar datos de usuario
Descripción	Como usuario quiero poder ver mis datos de usuario en la aplicación
Valor	2 SP
Prioridad	Alta

Identificador	US04
Título	Crear deseo de regalo
Descripción	Como usuario quiero poder crear un deseo de regalo para que mis amigos vean lo que deseo.
Valor	3 SP
Prioridad	Alta

Identificador	US05
Título	Consultar deseo de regalo
Descripción	Como usuario quiero poder ver mis deseos

	de regalo para poder realizar acciones sobre él
Valor	2 SP
Prioridad	Alta

Identificador	US06
Título	Actualizar deseo de regalo
Descripción	Como usuario quiero poder actualizar mis deseos de regalo para mantenerlos con la última información disponible.
Valor	2 SP
Prioridad	Media

Identificador	US07
Título	Borrar deseo de regalo
Descripción	Como usuario quiero poder borrar mis deseos de regalo para poder mantener mis deseos actualizados.
Valor	1 SP
Prioridad	Baja

Identificador	US08
Título	Consultar lista de deseos
Descripción	Como usuario quiero poder consultar listas de deseos mías o de mis amigos para poder elegir un regalo
Valor	3 SP
Prioridad	Alta

Identificador	US09
Título	Reservar un regalo
Descripción	Como usuario quiero poder reservar un regalo para que nadie compre el mismo

	regalo
Valor	1 SP
Prioridad	Alta

Identificador	US10
Título	Deshacer reserva de regalo
Descripción	Como usuario quiero poder cancelar la reserva de un regalo para que otra persona pueda reservarlo
Valor	1 SP
Prioridad	Media

Identificador	US11
Título	Crear círculo
Descripción	Como usuario quiero poder crear círculos de amigos para poder compartir listas de deseos de forma organizada
Valor	5 SP
Prioridad	Alta

Identificador	US12
Título	Consultar detalles de círculo
Descripción	Como usuario quiero poder consultar los detalles de un círculo para conocer los datos
Valor	2 SP
Prioridad	Alta

Identificador	US13
Título	Actualizar detalles de círculo
Descripción	Como usuario quiero poder actualizar los detalles de mis círculos para mantenerlos al día

Valor	2 SP
Prioridad	Alta

Identificador	US14
Título	Borrar círculo
Descripción	Como usuario quiero poder borrar los círculos que he creado para eliminar los que ya no tengan sentido
Valor	3 SP
Prioridad	Alta

Identificador	US15
Título	Añadir persona a círculo
Descripción	Como usuario quiero poder añadir a otra persona a mis círculos para que pueda ver las listas compartidas con ese círculo
Valor	2 SP
Prioridad	Alta

Identificador	US16
Título	Consultar personas de círculo
Descripción	Como usuario quiero poder ver qué personas están en un círculo
Valor	2 SP
Prioridad	Alta

Identificador	US17
Título	Borrar persona de círculo
Descripción	Como usuario quiero poder eliminar a personas de un círculo para mantener los círculos ordenados
Valor	2 SP

Prioridad	Media
-----------	-------

Identificador	US18
Título	Consultar detalles de persona
Descripción	Como usuario quiero poder consultar detalles de las personas que están en mis círculos para obtener más información relevante sobre el regalo
Valor	2 SP
Prioridad	Media

Identificador	US19
Título	Añadir característica de usuario
Descripción	Como usuario quiero poder añadir características y fechas a mi perfil para que los demás usuarios tengan más contexto sobre qué y cuándo regalarmelo
Valor	5 SP
Prioridad	Media

Identificador	US20
Título	Consultar características de usuario
Descripción	Como usuario quiero poder consultar mis características de usuario para poder comprobar que son correctas
Valor	2 SP
Prioridad	Media

Identificador	US21
Título	Actualizar característica de usuario
Descripción	Como usuario quiero poder actualizar mis características y fechas de mi perfil para que reflejen los datos más actuales
Valor	2 SP

Prioridad	Media
-----------	-------

Identificador	US22
Título	Borrar característica de usuario
Descripción	Como usuario quiero poder borrar características y fechas de mi perfil para que mi perfil represente mis preferencias
Valor	2 SP
Prioridad	Media

Identificador	US23
Título	Consultar recordatorios
Descripción	Como usuario quiero poder consultar recordatorios de regalos y fechas de mis amigos para que no se me pasen
Valor	5 SP
Prioridad	Media

Identificador	US24
Título	Descartar recordatorio
Descripción	Como usuario quiero descartar recordatorios que no me interesan para mantener mi lista de recordatorios ordenada
Valor	1 SP
Prioridad	Baja

Identificador	US25
Título	Crear mensaje
Descripción	Como usuario quiero poder crear mensajes para otras personas
Valor	2 SP

Prioridad	Baja
-----------	------

Identificador	US26
Título	Consultar mensajes recibidos
Descripción	Como usuario quiero poder consultar una lista de mensajes recibidos para gestionarlos
Valor	2 SP
Prioridad	Baja

Identificador	US27
Título	Leer mensajes
Descripción	Como usuario quiero poder leer un mensaje recibido
Valor	1 SP
Prioridad	Baja

Identificador	US28
Título	Responder mensaje
Descripción	Como usuario quiero poder responder a un mensaje recibido
Valor	1 SP
Prioridad	Baja

Identificador	US29
Título	Hacer una pregunta anónima
Descripción	Como usuario quiero poder hacer una pregunta a un usuario anónimamente para obtener información respecto a un regalo
Valor	1 SP
Prioridad	Baja

Identificador	US30
Título	Consultar preguntas recibidas
Descripción	Como usuario quiero ver una lista de personas anónimas recibidas
Valor	1 SP
Prioridad	Baja

Identificador	US31
Título	Responder preguntas
Descripción	Como usuario quiero poder contestar a las preguntas anónimas recibidas
Valor	1 SP
Prioridad	Baja

Identificador	US32
Título	Asignar mejor amigo
Descripción	Como usuario quiero poder asignar a una persona para que reciba preguntas sobre mis gustos para que yo no sospeche sobre el regalo que voy a recibir
Valor	1 SP
Prioridad	Baja

Identificador	US33
Título	Crear amigo invisible
Descripción	Como usuario quiero poder crear un juego de amigo invisible entre todos los miembros de un grupo
Valor	5 SP
Prioridad	Baja

Identificador	US34
Título	Consultar amigo invisible
Descripción	Como usuario quiero poder ver si hay un amigo invisible en un círculo y qué usuario es mi amigo invisible
Valor	2 SP
Prioridad	Baja

Identificador	US35
Título	Finalizar amigo invisible
Descripción	Como usuario quiero poder finalizar amigos invisibles que haya iniciado.
Valor	2 SP
Prioridad	Baja

Sprints

Los costes de un cambio en los requisitos escalan a medida que el proyecto avanza [16].

Las metodologías ágiles intentan paliar los costes de los cambios de requisitos entregando software usable de manera periódica que permite la mejor comprensión del producto por parte del cliente y ayuda a detectar cambios deseados a lo largo del desarrollo.

En este proyecto se realizarán cuatro entregas. Cada entrega es el resultado de un “Sprint”: un periodo de dos semanas de desarrollo en el que se priorizan las historias más relevantes del proyecto atendiendo a la capacidad de desarrollo del equipo. Tras cada Sprint se obtendrá un producto software mínimo que se despliega a producción.

Sprint 1

Sprint planning

Del 17 al 30 de abril de 2023

En el primer sprint se desarrollarán las tareas técnicas que condicionan gran parte del desarrollo del proyecto.

Identificador	Título	SP
TS01	Crear el proyecto Spring	2
TS02	Crear workflow para integración continua para Spring	3
TS03	Crear proyecto Angular	2
TS04	Crear workflow para integración continua para Angular	3
TS05	Crear autenticación en Firebase	2
TS06	Crear base de datos Cloud Firestore	3
TS07	Crear base de datos PostgreSQL	3

Sprint Review

En este Sprint se han definido las bases de los proyectos junto a la arquitectura principal así como la configuración de entornos.

Arquitectura Backend:

El backend es una aplicación Java Spring, que utiliza maven como control de dependencias. Con el inicializador que ofrece Spring Boot, Spring Initializer, se ha generado un proyecto básico. En el fichero pom.xml se han configurado las dependencias básicas que se necesitan, pudiendo ampliarse en futuros Sprints. Entre ellas destacan:

- Dependencias para trabajar fácilmente con la base de datos postgresQL:
 - spring-boot-starter-data-jpa
 - postgresql
 - jakarta.persistence-api
- Dependencias para poder realizar programación reactiva:
 - projectreactor
 - spring-boot-starter-webflux
- Dependencias que facilitan el desarrollo:
 - lombok: simplifica la creación de los modelos de datos a través de anotaciones

La arquitectura del proyecto está basada en tres capas. En la figura 4 se muestra un diagrama de paquetes de la solución.

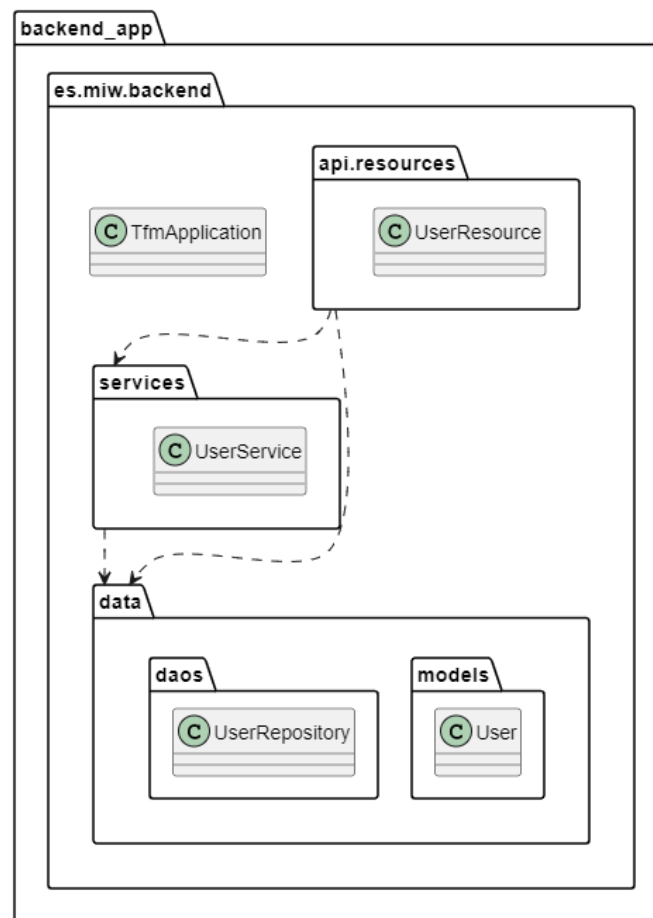


Figura 4. Diagrama de paquetes

En el paquete *api.resources* se encuentran las clases que definen las operaciones de la API Rest. Este paquete es en esencia un paquete de vistas: interfaces con las que actuarán los usuarios finales, que en este proyecto será el frontend.

En el paquete *services* se encuentran las clases que se encargan de realizar los procesamiento necesarios. Estas clases son los controladores de la app.

En el paquete *data* se encuentran los paquetes *daos* y *models*.

- En *daos* se encontrarán las interfaces necesarias para comunicarse con la base de datos. Aquí se implementarán los métodos necesarios para las búsquedas sobre la base de datos.
- En *models* se encontrarán las clases que representan la estructura de los datos.

En definitiva, esta arquitectura es una arquitectura Modelo-Vista-Controlador. En la arquitectura típica MVC las vistas dependen de los controladores y los controladores dependen de los modelos. De esta manera los cambios en las vistas serán muy simples pero los cambios en los modelos serán más costosos.

Esta arquitectura crecerá a medida que se realicen entregas. Idealmente los modelos serán estables y si se necesitasen cambios en las clases ya programadas se haría utilizando composición.

Arquitectura Frontend:

El frontend es una aplicación realizada en Angular. Se ha añadido la dependencia de Ionic para fácilmente diseñar una interfaz de usuario estándar.

La arquitectura en el frontal es la recomendada por Angular. Agrupada por componentes.

En la Figura 5 se muestra un diagrama de paquetes de la arquitectura angular.

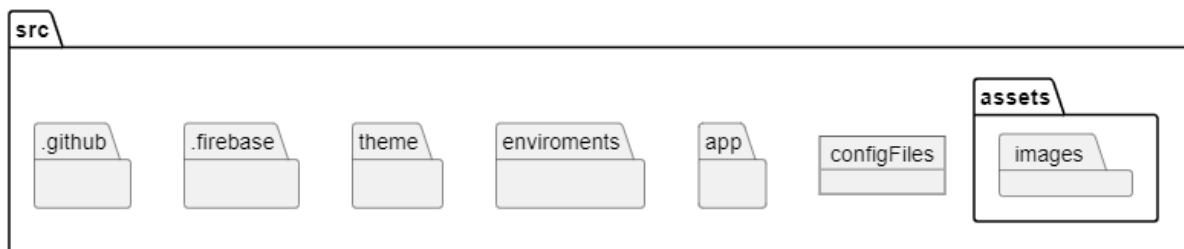


Figura 5. Diagrama de paquetes frontend

En *.github* y *.firebase* se coloca la configuración de sendas herramientas.

En *theme* hay ficheros de estilos globales.

En *environments* hay dos ficheros de configuración con las variables de entorno necesarias.

En *assets* se colocarán todos recursos que se necesiten como imágenes o audios.

Además habrá diferentes ficheros de configuración en la raíz del proyecto siendo *package.json* el documento en el que se configuran las dependencias principales del proyecto.

En *apps*, se coloca el grueso del código de la app. En la siguiente figura se muestra el detalle del contenido.

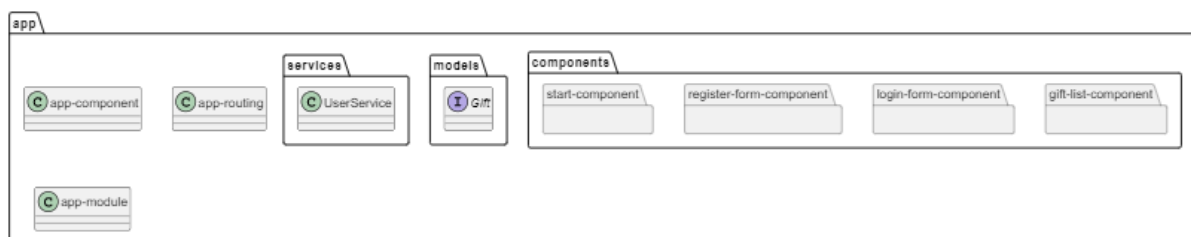


Figura 6. Diagrama de paquetes frontend, detalle

Los ficheros más importantes de la carpeta app son los siguientes:

- *app.component.**: el componente raíz. En él se configuran los elementos de interfaz comunes y la plantilla general de la app.
- *app-routing.module.ts*: el componente que gestiona el enrutado de las páginas.
- *app.module.ts*: el componente en el que se declaran e importan las dependencias y componentes del proyecto.

Adicionalmente existen tres directorios que marcan la arquitectura:

- *components*: contiene carpetas de componentes de cada una de las pantallas de la web. Cada componente tendrá una vista, un controlador y los estilos necesarios.

- *models*: interfaces de los datos.
- *services*: clases que interactúan con sistemas externos: APIs, bases de datos... Normalmente serán conexiones al backend.

Entornos y CI/CD

Una de las tareas importantes de este Sprint es preparar y configurar dos entornos de desarrollo: uno para desarrollo y otro más estable, enfocado a producción.

El entorno de desarrollo utiliza recursos locales, sin embargo los entornos remotos utilizan recursos que estarán en diferentes hosting.

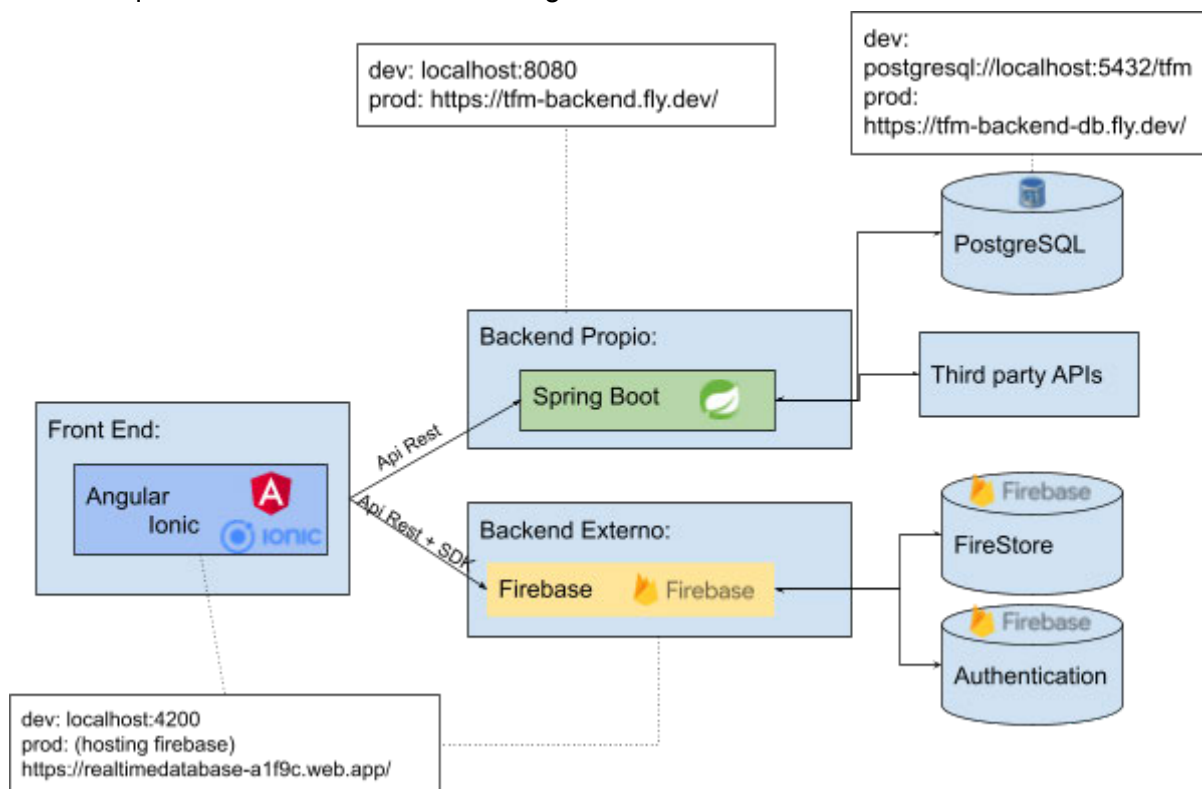


Figura 7. Stack tecnológico

Frontend

Para el Frontend se utilizará Firebase[17]. Utilizar Firebase como hosting de Angular es trivial pues ambas tecnologías son de Google y tienen amplio soporte. Además, Firebase tiene una capa de uso gratuita que permite usar tanto el servicio de hosting como sus tecnologías Cloud Firestore y Authentication.

Inicialmente no estaba planteado utilizar Cloud Firestore, sino que se planeaba usar RealTime Database. Ambas bases de datos proporcionadas por Firebase son bases de datos no relacionales que garantizan el sincronismo de los datos de las diferentes conexiones de usuarios. En la siguiente tabla se muestra una pequeña comparativa de las características entre las dos bases de datos de acuerdo a Firestore [18]:

Real Time Database	Cloud Firestore	Necesidades del proyecto
Sincronización de datos con filtros sencillos	Sincronización de datos con filtros avanzados	Cloud FireStore
Pocos GB de datos que cambian frecuentemente	Sincronización de muchos datos que se leen mucho más a menudo de lo que se cambian	Real Time Database
Datos estructurados como un solo árbol JSON	Datos estructurados en colecciones.	Cloud FireStore
Garantía de disponibilidad del 99.95%	Garantía de disponibilidad del 99.999%	Ambas
Mala solución para usuarios que se conectan desde dispositivos con conectividad limitada	Mejor solución para dispositivos con conectividad limitada	No usaremos esta características
Varias bases de datos	Una sola base de datos	Cloud FireStore

Cloud Firestore es una base de datos más robusta y escalable que RealTime Database. Firestore ofrece mayor flexibilidad a la hora de filtrar resultados por lo que es la mejor opción para el proyecto.

Una vez están claras todas las tecnologías a usar, realizar un workflow de github que soporte la construcción y despliegue de la app de Angular a Firebase es trivial.

Backend

Para el backend, en cambio, no estaba clara la plataforma elegida para el despliegue. Idealmente se hubiese usado Heroku, una de las plataformas más utilizadas para desplegar apps en la nube. Lamentablemente, el 28 de noviembre de 2022 [19], Heroku dejó de ofrecer una capa gratuita de sus servicios, por lo que se han buscado otras alternativas.

Las plataformas Cloud líderes en mercado son AWS y Google Cloud (App Engine). Ambas plataformas ofrecen gran capacidad de cómputo y disponibilidad, disponiendo [20] [21] ambas de una instancia en España lo cual aseguraría una latencia muy baja.

Ambas plataformas disponen de una política “*pay as you go*” que permite pagar solamente por los recursos consumidos. Además disponen de una capa de prueba gratuita.

- AWS: 12 meses de prueba gratis en la que se puede configurar una máquina compatible (EC2) con nuestras necesidades.
- Google Cloud: 90 días de prueba y 300\$ en crédito para probar la plataforma.

Sin embargo no se apuesta por ninguna de las dos opciones. La prueba gratuita de Google podría poner en peligro la disponibilidad de la aplicación en el momento de la presentación

del presente TFM y anteriormente, la prueba gratuita de AWS fue consumida en un trabajo de fin de grado.

Además, ambos servicios dejarían de ser gratuitos eventualmente tras la entrega del proyecto. Por ello se estudia otra alternativa menos presente en el mercado pero que soluciona nuestras necesidades.

fly.io [22] es una plataforma Platform as a Service(PaaS) con una capa gratuita sin límite en el tiempo que permite la correcta ejecución de nuestro backend así como la creación y hosting de una base de datos PostgreSQL:

La solución permite la ejecución de tres máquinas virtuales de 256MB de memoria, 3GB de almacenamiento y 160GB de transferencia de datos. Además, ofrece 5 USD gratuitamente al mes, por lo que podemos ampliar la memoria de las máquinas virtuales a 512MB sin coste, suficiente para nuestro backend.

Elegir una PaaS que no es líder en mercado conlleva el riesgo de un soporte de la comunidad limitado que se asume.

Se ha creado un workflow de github que soporta la construcción y despliegue de la app Spring a una máquina virtual. Adicionalmente se ha creado la configuración de la conexión a la base de datos base de datos en el archivo `application.properties` para el entorno de producción.

Tiempos

Con la herramienta Wakatime, se puede observar cuánto tiempo se ha dedicado a la programación en cada proyecto.

Backend: 6 horas

Frontend: 11 horas

Habitualmente al finalizar un sprint con la metodología scrum se utilizan los gráficos burn down, que representan en el eje "x" el tiempo en días del sprint y en el eje "y" las historias de usuario por finalizar. Idealmente el gráfico burn down debería tener una línea recta descendiente desde el primer día, que estará en el máximo al día de entrega, que estará en 0.

La línea real que muestra el desarrollo de acuerdo a las tareas finalizadas por día permite ver si el desarrollo está en tiempo o si hay alguna desviación importante.

Sin embargo, para este proyecto no se generan estos gráficos ya que el tiempo dedicado a este proyecto no es el mismo para cada día del sprint y ha habido días en los que se ha avanzado mucho más que otros, simplemente porque se ha dedicado más tiempo en el proyecto. De esta manera el gráfico burn down tendría días "picos" en los que se programaría mucho y días en los que no se programaría. Estos picos no representan ni adelantos ni retrasos en el proyecto y el gráfico burn down perdería parte de su significado.

Sprint retrospective

Problemas detectados:

- La herramienta Wakatime no ha podido identificar correctamente el tiempo que se ha invertido en el Sprint, puesto que la mayor parte de las tareas técnicas de este Sprint son de configuración y se realizan fuera de los ficheros de la aplicación. Además el reporte por fichero se pierde a los 14 días, por lo que debe descargarse en ese plazo de tiempo.
- No se ha llevado bien el flujo ramificado ya que la configuración de los entornos necesita que se trabaje en la rama master.
- No se han creado test del código que existe.

Plan de mejora:

- Utilizar wakatime adecuadamente, sacando una copia de reportes cada 14 días.
- Utilizar el flujo de trabajo ramificado de manera más estricta.
- Crear test para el nuevo código.

Sprint 2

Sprint planning:

Del 1 al 14 de mayo de 2023

En este Sprint se abordarán las siguientes tareas:

Identificador	Título	SP
US01	Registro de usuarios	3
US02	Inicio y cierre de sesión de usuarios	3
US03	Consultar datos de usuario	2
US04	Crear deseo de regalo	3
US05	Consultar deseo de regalo	2
US06	Actualizar deseo de regalo	2
US07	Borrar deseo de regalo	1
US08	Consultar lista de deseos	3
US09	Reservar un regalo	1
US10	Deshacer reserva de regalo	1
US11	Crear círculo	5
US12	Consultar detalles de círculo	2
US13	Actualizar detalles de círculo	2
US14	Borrar círculo	3
US15	Añadir persona a círculo	2
US16	Consultar personas de círculo	2
US17	Borrar persona de círculo	2
US18	Consultar detalles de persona	2

En este sprint se pretende crear la gestión de la cuenta de usuarios, la gestión de los deseos de regalos y la gestión de círculos de amigos.

Sprint Review

Title	Milestone	Tipo	SP	Time consumed
9 US01 Registro de usuarios #9	Sprint 2	User Story	3 Points	3
10 US02 Inicio y cierre de sesión de usuarios #10	Sprint 2	User Story	3 Points	4.33
11 US03 Consultar datos de usuario #11	Sprint 2	User Story	2 Points	1
12 US11 Crear círculo #19	Sprint 2	User Story	5 Points	4.1
13 US12 Consultar detalles de círculo #20	Sprint 2	User Story	2 Points	3
14 US13 Actualizar detalles de círculo #21	Sprint 2	User Story	2 Points	1.1
15 US16 Consultar personas de círculo #24	Sprint 2	User Story	2 Points	0
16 US04 Crear deseo de regalo #12	Sprint 2	User Story	3 Points	2
17 US05 Consultar deseo de regalo #13	Sprint 2	User Story	2 Points	3
18 US18 Consultar detalles de persona #26	Sprint 2	User Story	2 Points	1.25
19 US06 Actualizar deseo de regalo #14	Sprint 2	User Story	2 Points	0.5
20 US07 Borrar deseo de regalo #15	Sprint 2	User Story	1 Point	0.5
21 US08 Consultar lista de deseos #16	Sprint 2	User Story	3 Points	2
22 US09 Reservar un regalo #17	Sprint 2	User Story	1 Point	0.5
23 US14 Borrar círculo #22	Sprint 2	User Story	3 Points	1
24 US10 Deshacer reserva de regalo #18	Sprint 2	User Story	1 Point	0.6
25 US15 Añadir persona a círculo #23	Sprint 2	User Story	2 Points	2

Figura 8. Lista de tareas del Sprint 1 en Github Projects

La suma del tiempo de todas las tareas de desarrollo 27 horas. Sin embargo, un pequeño *bug* inesperado hizo que el tiempo de configuración se incrementase un poco.

Modelo de datos

En la Figura 9 se muestra un diagrama de clases que implementa el modelo de datos generado para este Spring en la base de datos PostgreSQL. Este modelo evolucionará en futuros Sprint de acuerdo a las necesidades del proyecto.

En este Sprint se han generado 3 clases que tienen relaciones entre ellas: User, Circle y List.

Circle y User comparten dos relaciones. Por un lado la relación 1:N representa que un círculo tiene un creador del tipo usuario lo que repercute en un atributo de tipo User llamado *Creator* en la clase Circle. Por otro lado existe una relación N:M que representa que un círculo tiene una lista de usuarios que están dentro de ese círculo lo que repercute en un atributo de tipo array en círculo, *userList*, que representa la lista de usuarios que contiene el círculo.

List y Circle comparten una relación N:M que representa que una lista se puede compartir con N círculos y además un círculo puede estar en varias listas. Esto repercute en un campo de tipo array en Lists, *circles*, que representa la lista de círculos con la que la lista ha sido compartida.

Por último, List y User tienen una relación 1:N. Una lista tendrá un usuario creador, lo que repercute que la clase List tiene un atributo *creator* del tipo User.

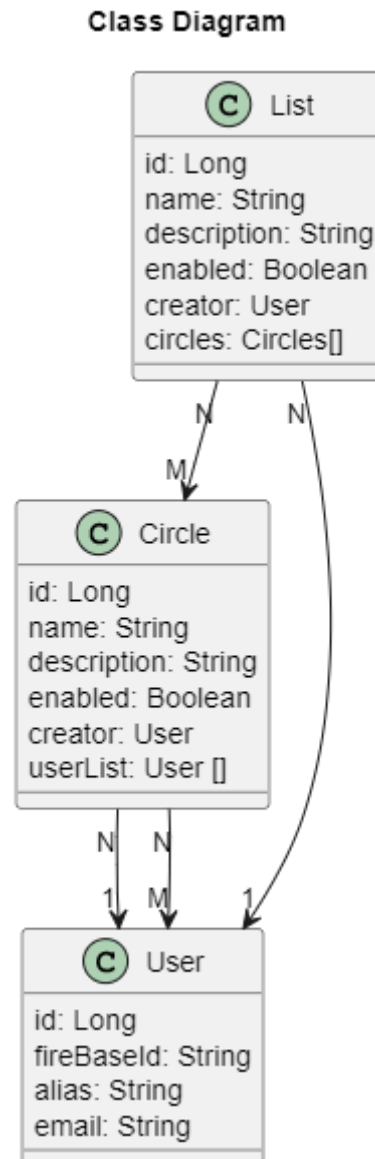


Figura 9. Modelo de datos Sprint 2

Login, registro y perfil de usuario

En este Sprint se han afrontado las tareas necesarias para que el registro e inicio de sesión sea posible para un usuario. Para el registro e inicio de sesión se utiliza el servicio en nube de Firebase, llamado Firebase Authentication. Con este servicio será sencillo implementar un login con usuario y contraseña y además se puede iniciar sesión con Google.

Pantallas:

Para soportar estas características se proponen las siguientes pantallas.

Pantalla 1, login, que dispone de un simple formulario para introducir las credenciales. Además se incluye un botón para el inicio de sesión con una cuenta de Google y un botón para acceder a la pantalla del registro.

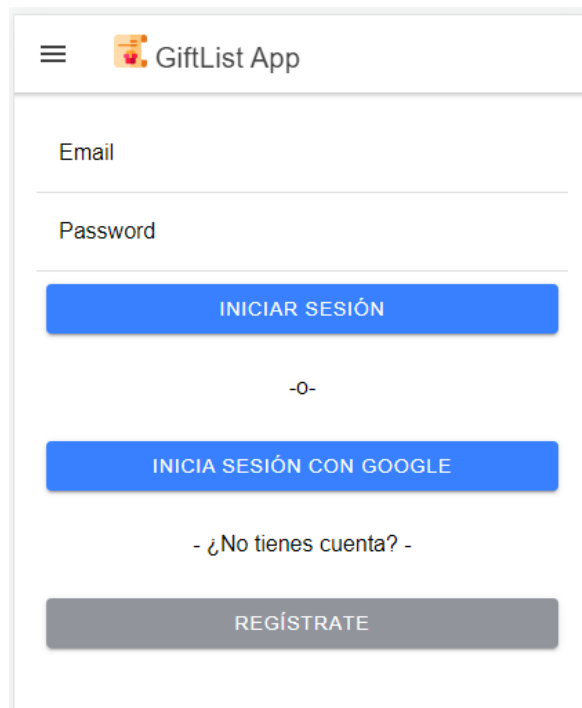


Figura 10. Pantalla de login

Pantalla 2, registro, que dispone de un simple formulario de registro.

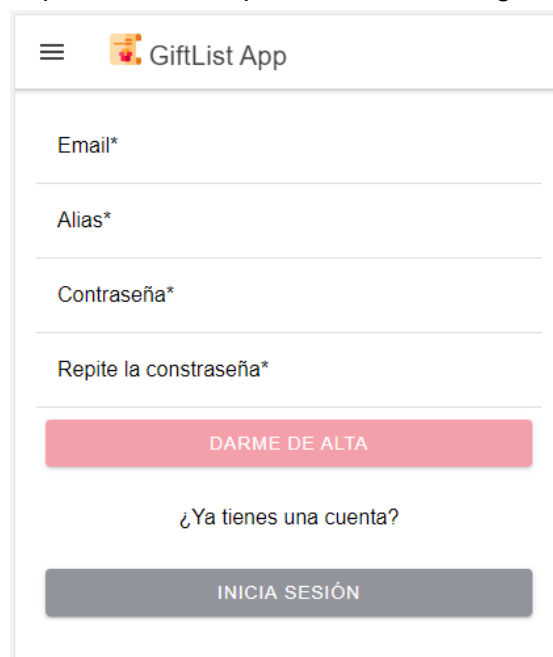


Figura 11. Pantalla de registro

Pantalla 3, perfil de usuario, es una pantalla en la que se muestran datos básicos que evolucionará en futuros sprints.

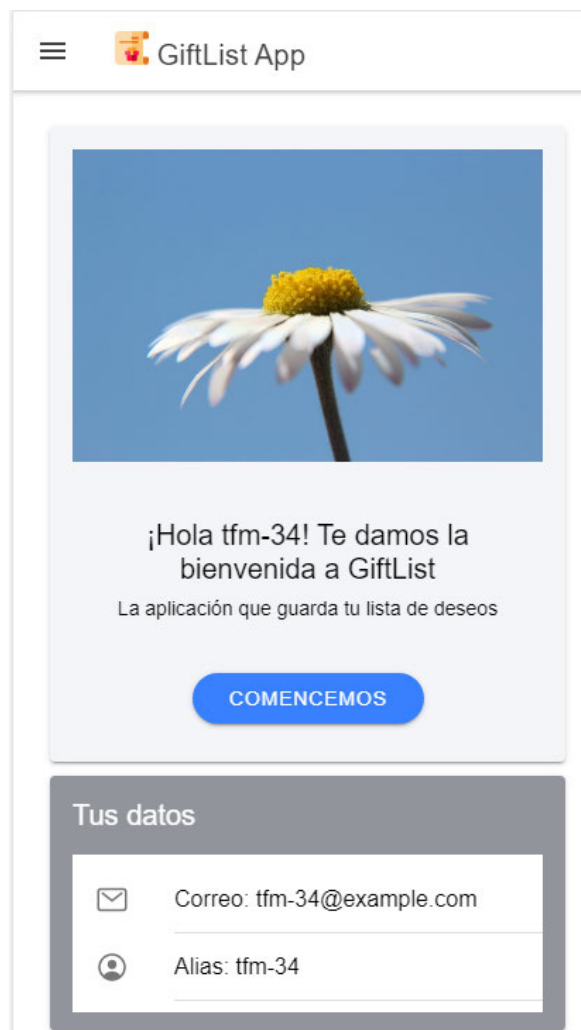


Figura 12. Pantalla de usuario

El flujo habitual será que el usuario se registre una única vez, inicie sesión y finalmente llegue a su perfil de usuario.

Frontend:

Para dar soporte a estas características en el front end se han evolucionado o creado tres componentes, dos servicios y un modelo:

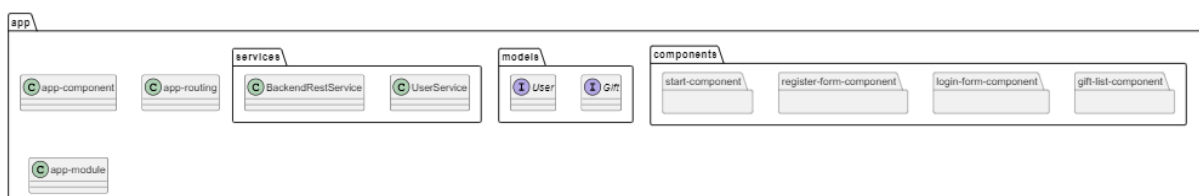


Figura 13. Diagrama de clases frontend, detalle

En app.components.* se han evolucionado:

- login-form-component, que controla la pantalla de inicio de sesión.
- register-form-component, que controla la pantalla de registro.
- start-component, que controla a la pantalla del perfil de usuario.

Todos estos directorios contienen un elemento .html y .css que contienen los detalles de las vistas y un componente .ts que se encarga de realizar las llamadas necesarias a los servicios.

Se ha creado el modelo User, que contiene el modelo de datos.

Estos componentes utilizan dos servicios que son los encargados de llamar a los backends requeridos. En la carpeta services están los dos servicios que se necesitan:

- UserService: Es un servicio que realiza las llamadas necesarias a Firebase, utilizando su SDK. En concreto se realizan llamadas a:
 - *createUserWithEmailAndPassword*, utilizado para el registro.
 - *signInWithEmailAndPassword*, utilizado para el login
 - *signInWithPopup*, utilizado para el login social con Google
 - *signOut*, utilizado para el logout
- BackendRestService: Es un servicio que realiza las llamadas necesarias al backend spring.

Backend:

Para ampliar las funcionalidades en el backend se han evolucionado el recurso, servicio, repositorio y modelo de la clase User para dar soporte a los nuevos endpoints creados.

Se han añadido clases en el paquete `api.https_errors` para la gestión correcta de los errores, tomando inspiración en la gestión de errores que se ha hecho en la asignatura de Spring [23].

Así mismo se ha añadido en el paquete `dtos` (data transfer objects) una clases que representan los datos entrantes a través de las llamadas API que tendrán una equivalencia con los modelos de datos que se guardarán finalmente en la base de datos.

La arquitectura sigue siendo una arquitectura por capas:

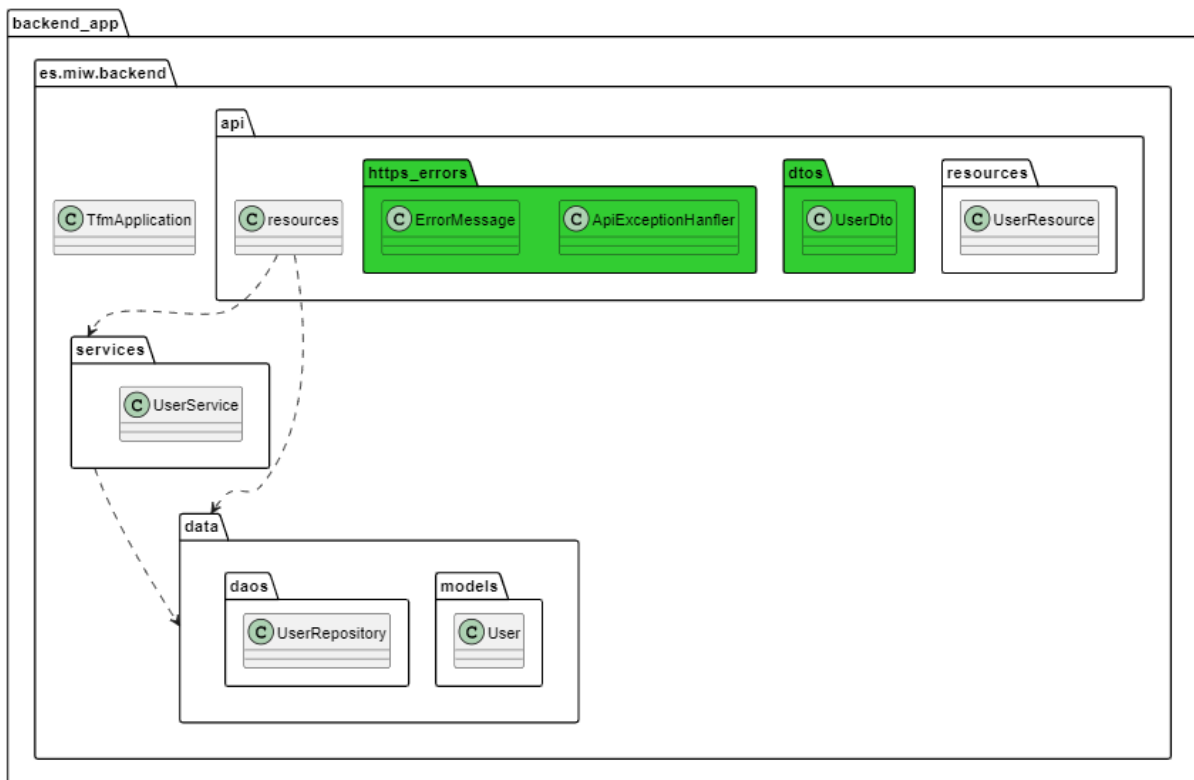


Figura 14. Diagrama de clases backend, detalle

En el backend se han habilitado los siguientes endpoints para el registro y el login.

POST /users

Endpoint que permite crear nuevos usuarios en el sistema. Este endpoint no necesita autorización para realizarse ya que cualquier usuario puede registrarse. Es necesario enviar en el cuerpo de la petición los datos bases del usuario siendo el id el id del mismo usuario en Firebase.



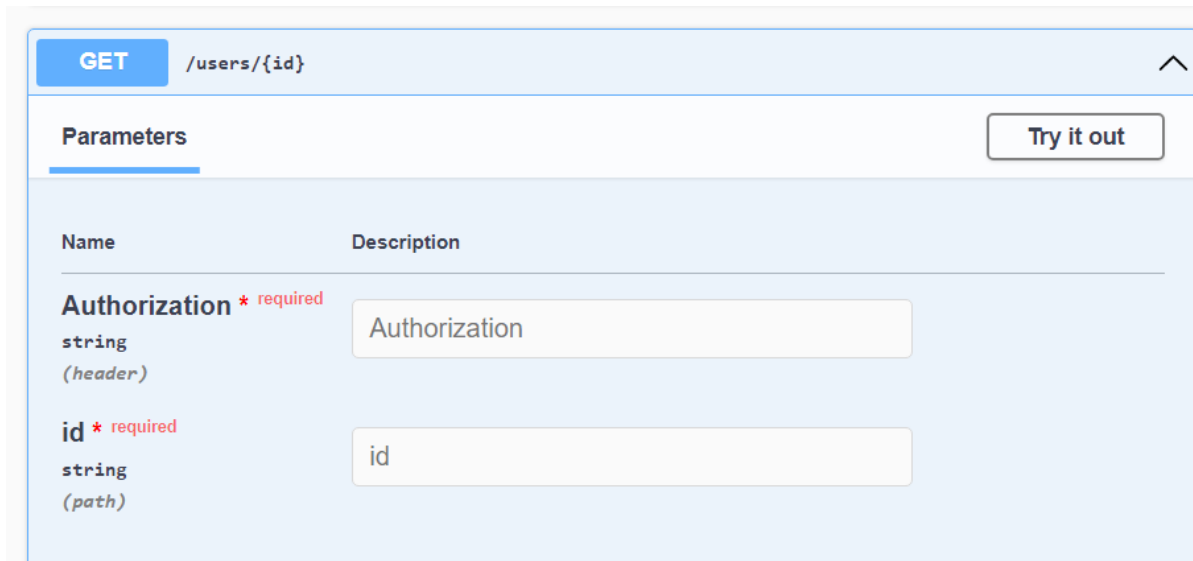
Figura 15. Entrypoint POST/users

GET /user/{id}

Endpoint que permite traer los datos de un usuario con un id de Firebase concreto. Se necesita un token de autenticación de Firebase válido para poder ejecutar la operación.

En caso de que el usuario que realice la operación sea el mismo que se está consultando, se enviarán todos los datos del usuario. En caso contrario una serie de datos personales

como el email del usuario no se enviará.



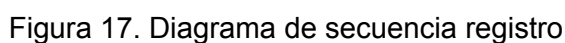
Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id

Figura 16. Entrypoint GET /users/{id}

Flujos:

Para aclarar el proceso de registro y login, ya que intervienen en ellos muchos componentes diferentes se han realizado los siguientes diagramas de secuencia para aclarar detalladamente el flujo completo de las dos historias.

Registro: El usuario solicita mediante la vista de registro registrarse con unos datos, el componente de la vista envía una petición al UserService que ejecuta una llamada a Firebase para dar de alta al usuario. Si todo en firebase es correcto se envía una petición al backend de spring para crear los datos adicionales del usuario y vincularlos con el id asignado en firebase. En caso de éxito el usuario es redirigido a la pantalla de login. En caso de error el usuario visualiza un mensaje de error.



49

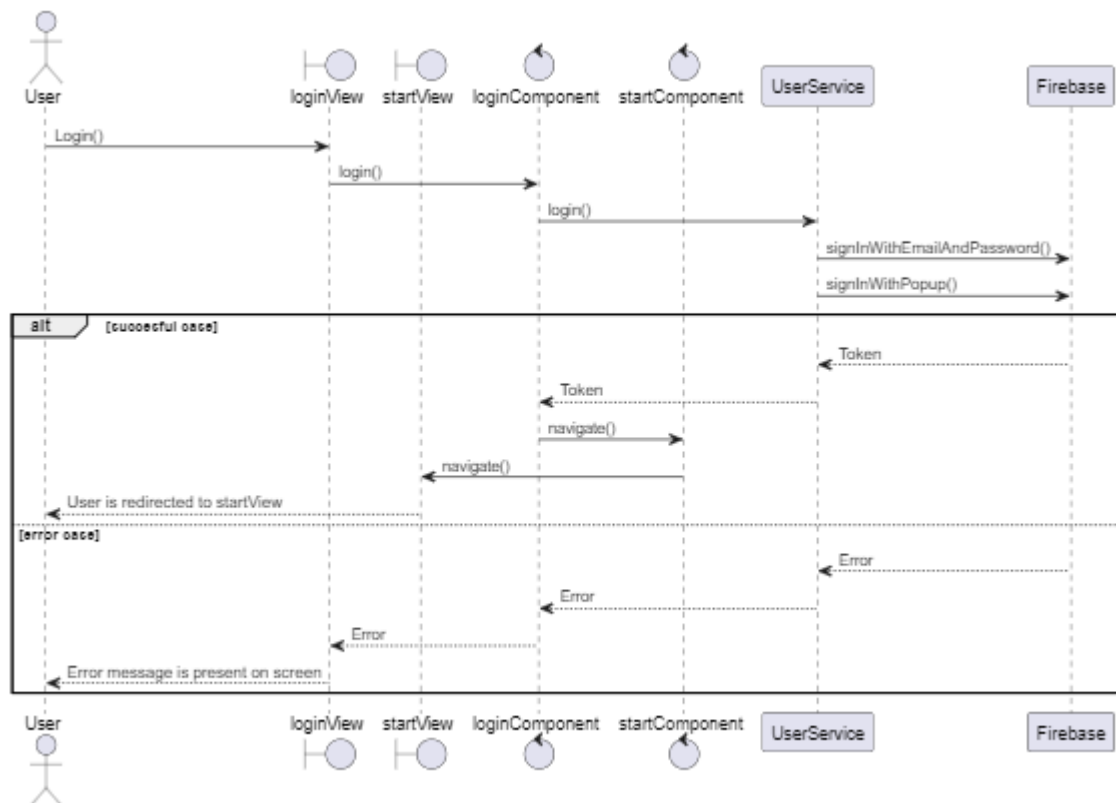


Figura 18. Diagrama de secuencia inicio de sesión

Perfil de Usuario, vista start.

Una vez el usuario ha realizado login y ha sido redirigido al componente start, el componente ejecuta una llamada al backend. Todas las llamadas al backend que necesitan autorización se realizan de manera análoga, por lo que este diagrama de secuencia representa cualquier llamada al backend.

En general una petición podrá ser iniciada por la carga de una pantalla o por la acción de un usuario en la misma, por ejemplo pulsar un botón. El componente encargado de la pantalla recoge la instrucción y pide al servicio de backend que realice una llamada rest. Será el componente el que se suscriba esperando el resultado asíncrono. El servicio de backend hace uso del SDK de Firebase para crear una cabecera con una autenticación JWT y ejecuta la llamada.

La parte backend, transparente para el servicio Rest, da soporte a la llamada con un Resource, que mapea y escucha las llamadas al endpoint. Este resource además realiza una llamada a UserService(Spring) para verificar que la firma del JWT es correcta. Si la petición REST fuese incorrecta por estar mal formada, o el JWT fuese incorrecto, el Resource devolvería el correspondiente mensaje de error con el código HTTP homólogo.

Si la petición es correcta, el Resource llamará al Servicio de la clase de datos que esté procesando, en este caso al tratarse de la clase User, se utiliza UserService. El servicio será el encargado de procesar adecuadamente la petición y a través de un Repository, una

clase que extiende de JpaRepository y permite de forma simple realizar consultas a la base de datos, recupera los datos de la base de datos.

Los datos son enviados de vuelta al resource y son devueltos por la llamada REST, que avisará al componente que está suscrito a la llamada. El componente de Angular finalmente procesa los datos y los carga en la pantalla.

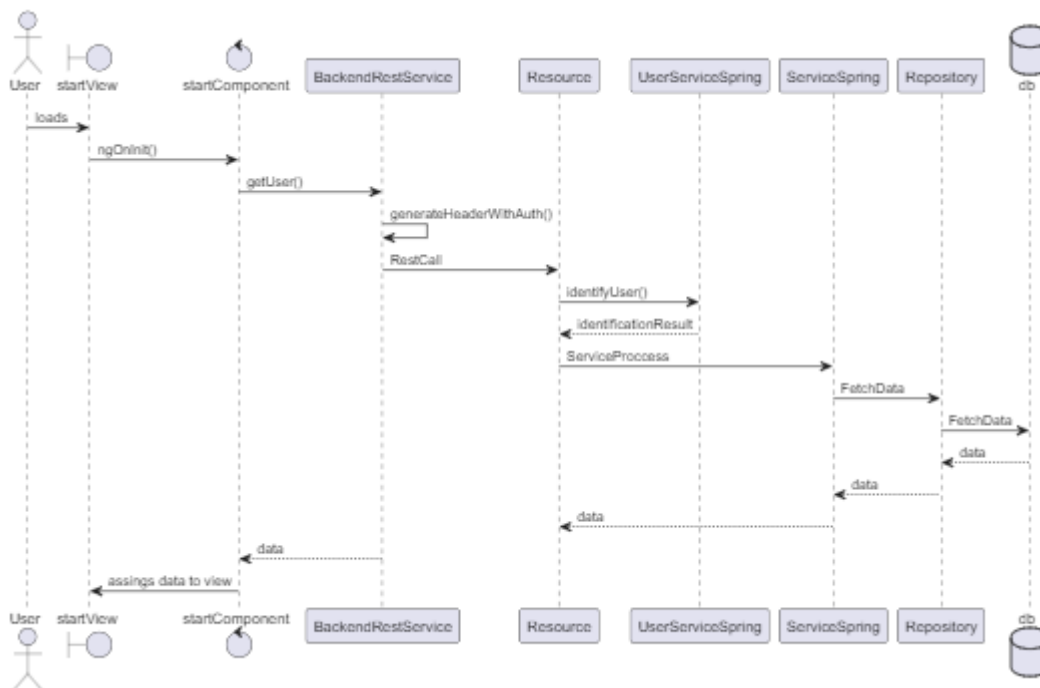


Figura 19. Diagrama de secuencia visualización de perfil

Gestión de círculos, listas y amigos

En este sprint además se han abordado las historias necesarias para la gestión de círculos de amigos, amigos que están en dichos círculos y listas de regalos que permitirán vincular y dar visibilidad a los regalos a los diferentes círculos.

Pantallas:

circles

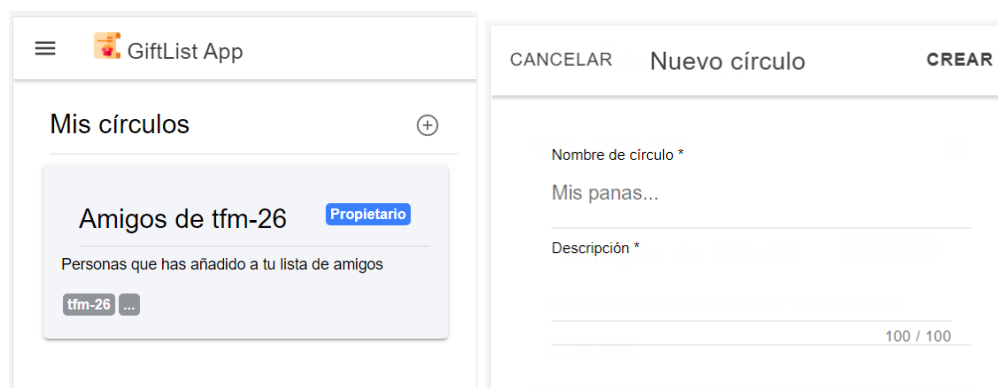


Figura 20. Pantalla de lista de círculos y modal Nuevo círculo

circles-detail

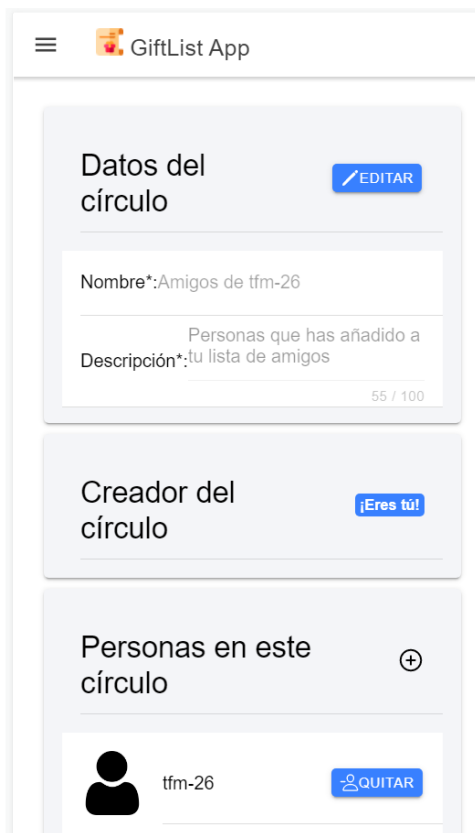


Figura 21. Pantalla detalles de un círculo

lists

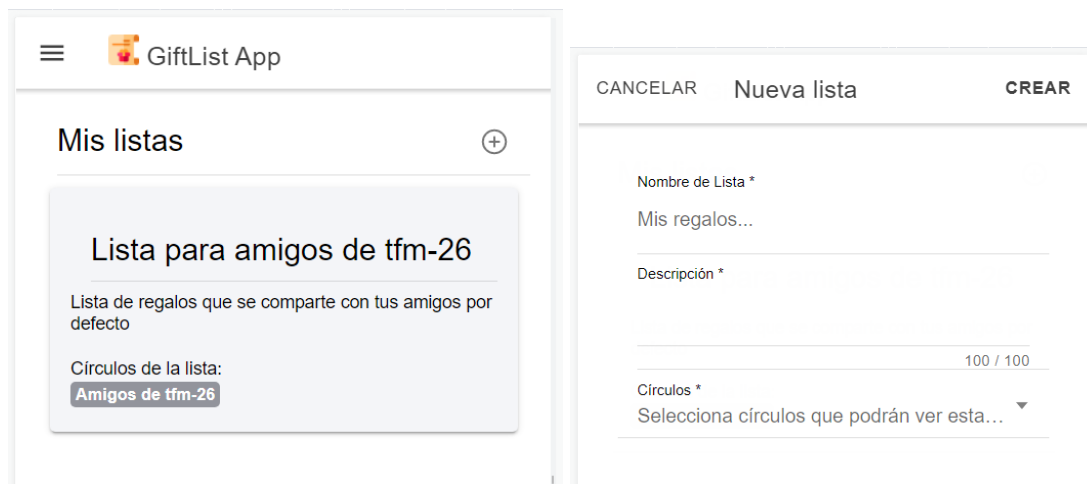


Figura 22. Pantalla Mis listas y modal Nueva lista

user-public-detail

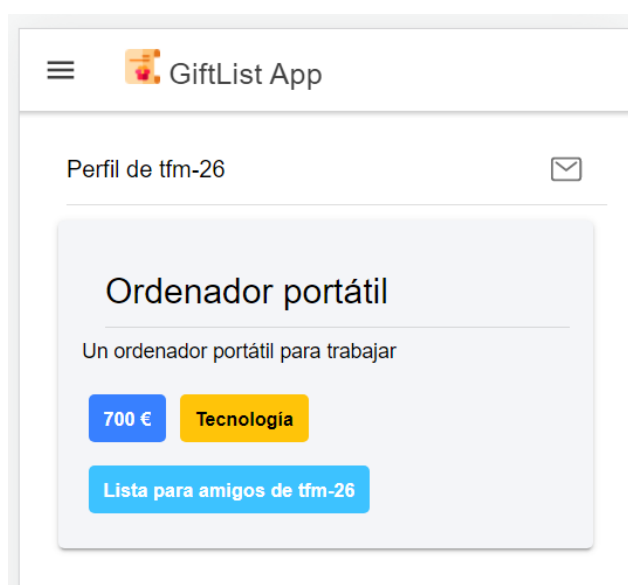


Figura 23. Pantalla de perfil público

Frontend:

Para dar soporte a estas pantallas se han añadido los componentes necesarios al proyecto de Angular.

- circles: Una pantalla que permite ver la lista de los círculos que puede ver un usuario. Permite crear un círculo a través de un modal.
- circles-detail: Una pantalla que permite ver el detalle de un círculo y si el usuario que tiene sesión iniciada es el propietario permite su edición.
- lists: Una pantalla que permite ver las listas de un usuario. Permite crear y editar una lista.
- user-public-detail: Una pantalla que muestra el detalle público de un usuario.

Adicionalmente se han añadido los modelos Circle y List que corresponden con el modelo de datos.

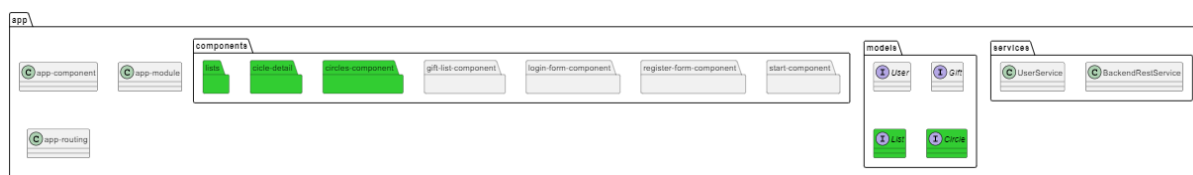


Figura 24. Diagrama de clases Angular, detalle

En el servicio de BackendRestService se han implementado las llamadas necesarias a los nuevos endpoints detallados en la siguiente sección.

Backend:

Para dar soporte a los endpoints necesarios para la historia de usuario se han creado nuevas clases en el backend.

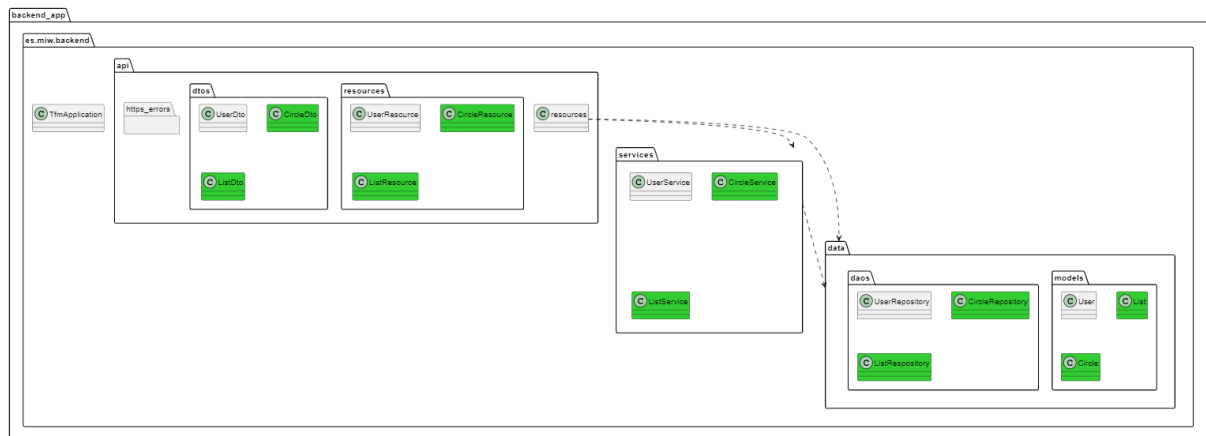


Figura 25. Diagrama de clases Backend, detalle

- DTOs necesarios.
- Recursos para círculo y listas
- Servicios para círculo y listas
- Modelos para círculo y listas
- Repositorios para círculo y listas

Además se han habilitado los siguientes endpoints:

GET /circles

Endpoint que permite obtener los círculos que puede ver un usuario. Un usuario podrá ver un círculo si es suyo o alguien le ha agregado. El endpoint necesita un token de autenticación

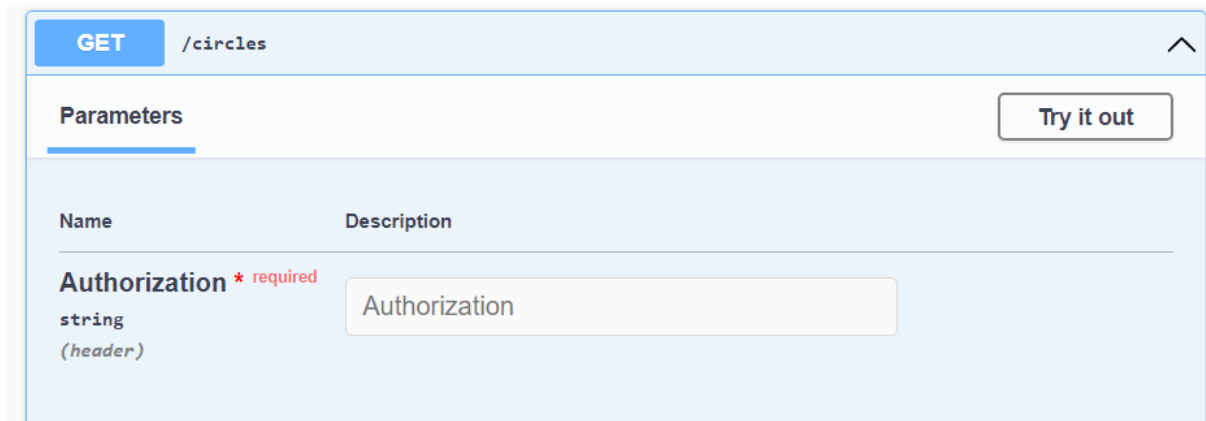
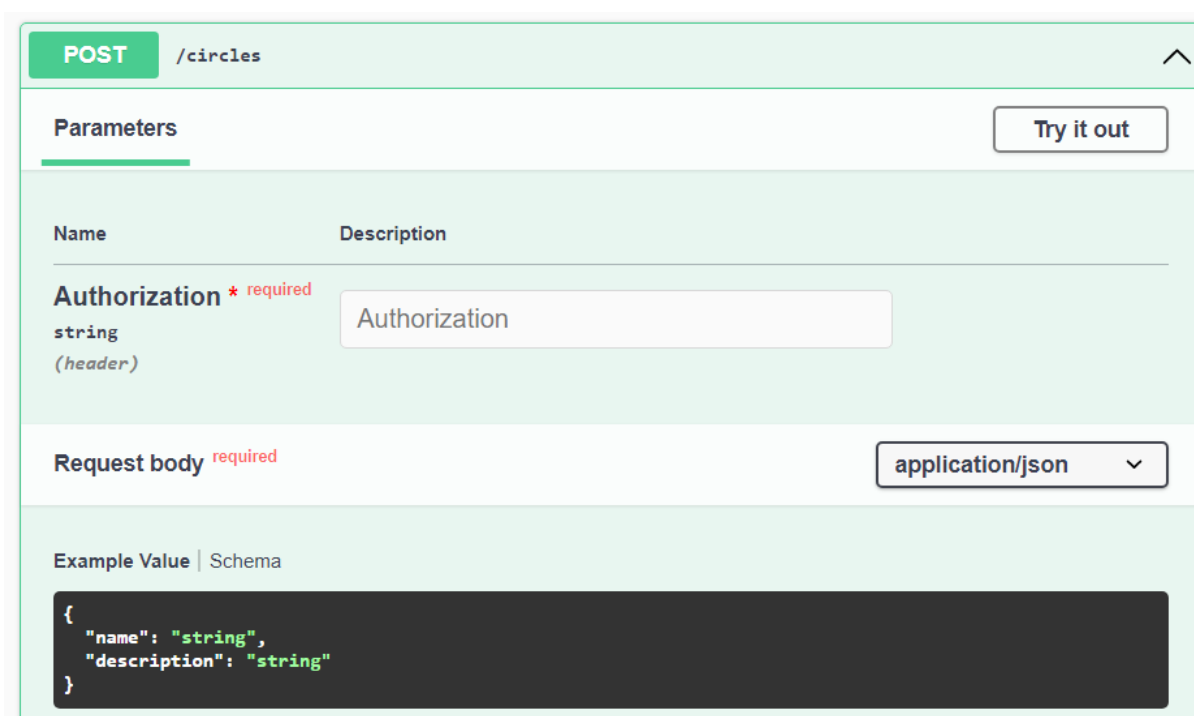


Figura 26. Endpoint GET /circles

POST /circles

Endpoint que permite la creación de un círculo. Necesita un token de autorización. El dueño del círculo será el usuario que realiza la petición.



POST /circles

Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization

Request body required application/json

Example Value | **Schema**

```
{
  "name": "string",
  "description": "string"
}
```

Figura 27. Entrypoint POST /circles

GET /circles/{id}

Entrypoint que permite obtener los detalles de un círculo. Solo devolverá valores si el usuario es dueño o está en el círculo.



GET /circles/{id}

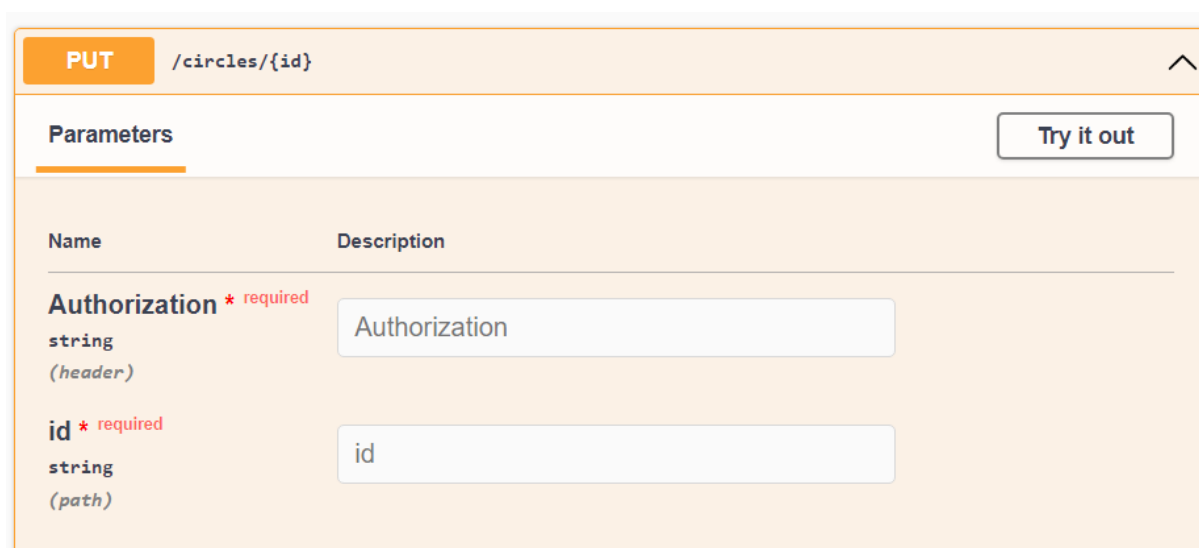
Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id

Figura 28. Entrypoint GET /circles/{id}

PUT /circles/{id}

Entrypoint que permite actualizar un círculo propio.



PUT /circles/{id}

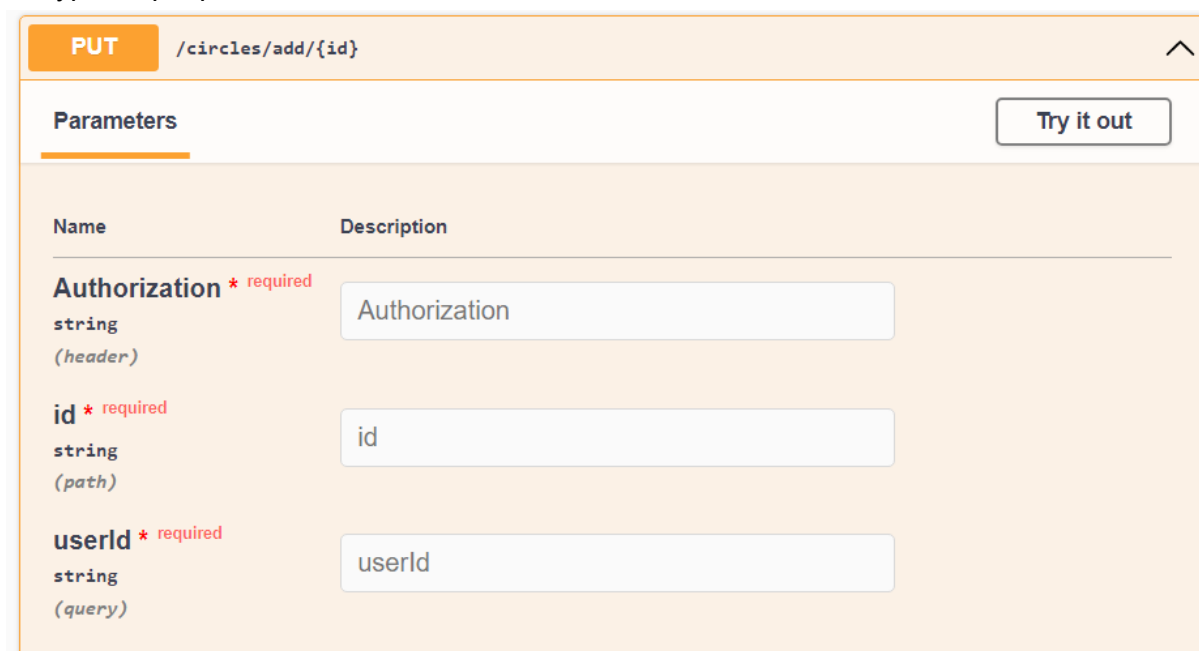
Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id

Figura 29. Entrypoint PUT /circles/{id}

PUT /circles/add/{id}

Entrypoint que permite añadir un usuario a un círculo.



PUT /circles/add/{id}

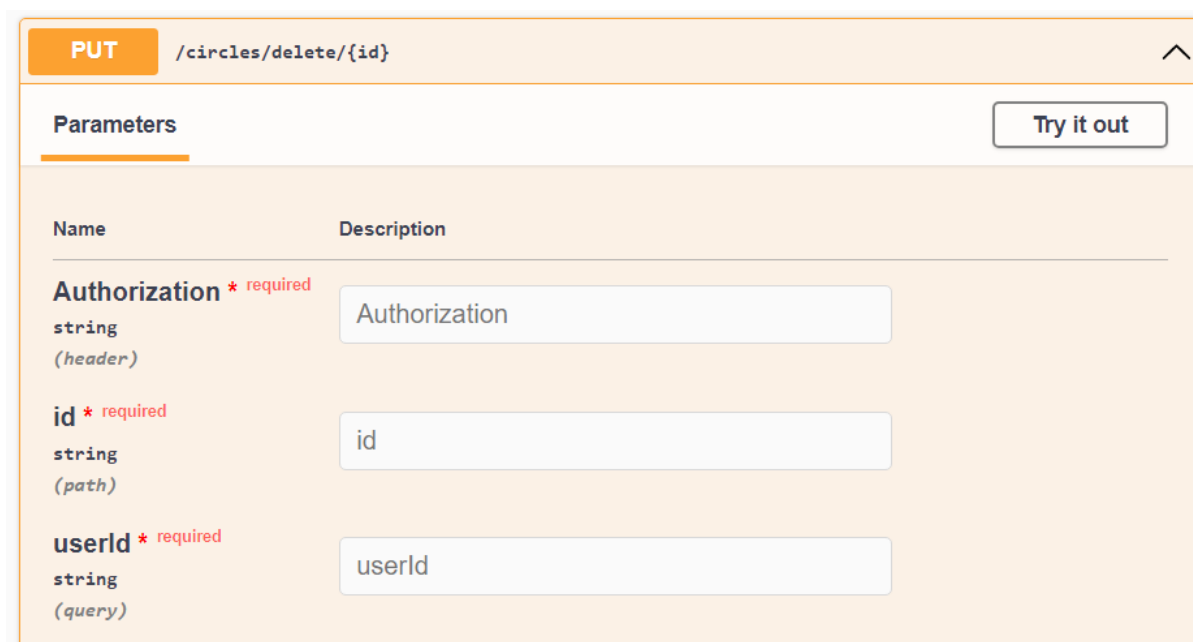
Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id
userId * required string (query)	userId

Figura 30. Entrypoint PUT /circles/add/{id}

PUT /circles/delete/{id}

Entrypoint que permite eliminar un usuario de un círculo.



PUT /circles/delete/{id}

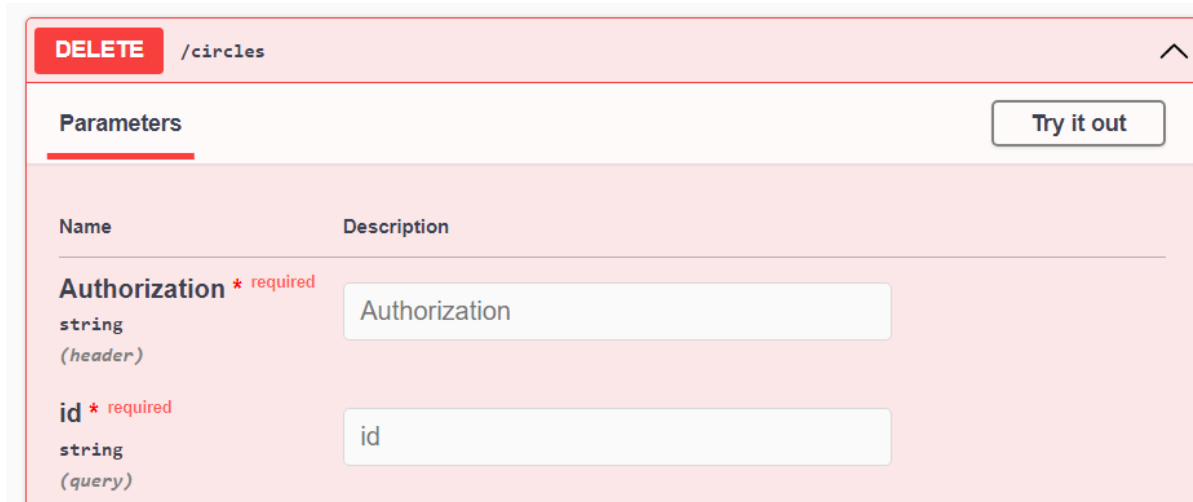
Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id
userId * required string (query)	userId

Figura 31. Entrypoint PUT /circles/delete/{id}

DELETE /circles/{id}

Entrypoint que permite eliminar un círculo del usuario que envía la petición. El borrado del círculo es lógico, pues pone el valor del círculo “enabled” a “false”.



DELETE /circles

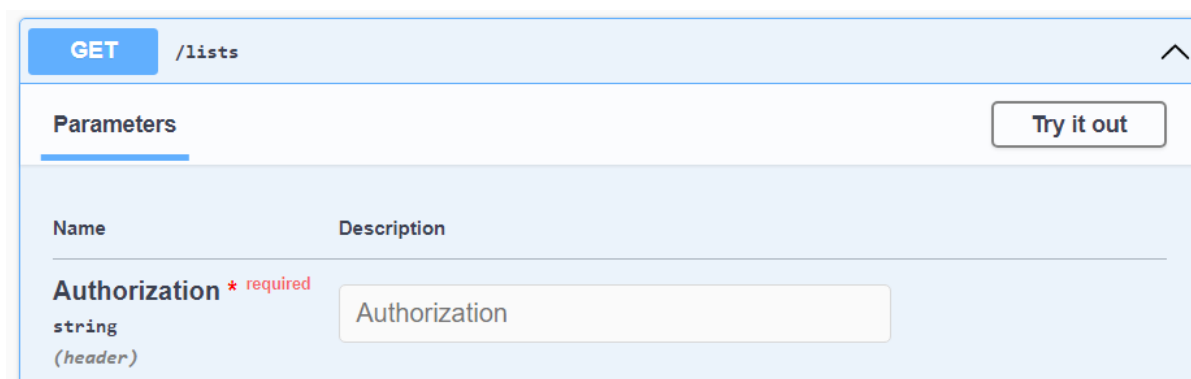
Parameters Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (query)	id

Figura 32. Entrypoint DELETE /circles

GET /lists

Entrypoint que permite obtener todas las listas del usuario que las solicita.



GET /lists

Parameters

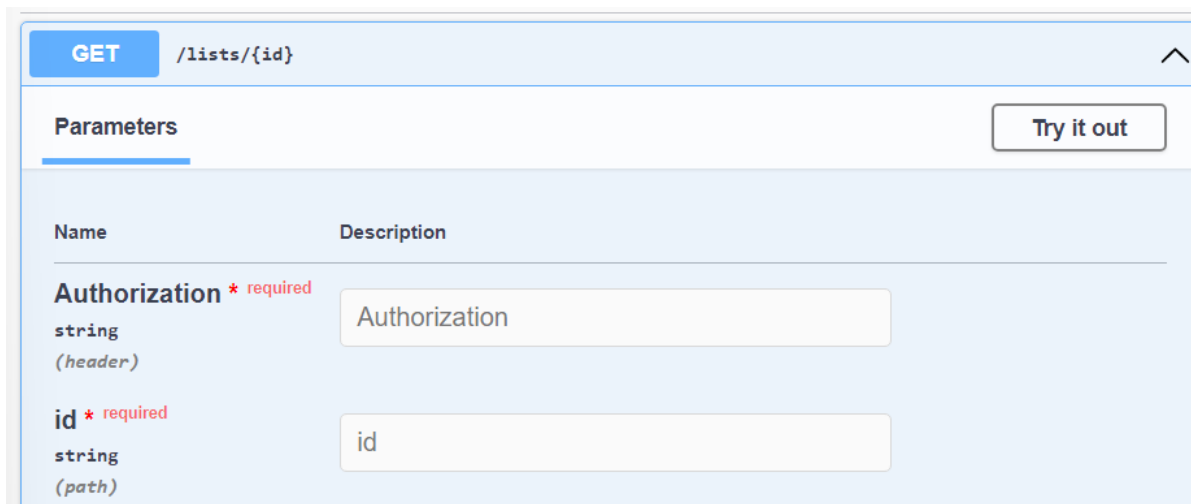
Try it out

Name	Description
Authorization * required string (header)	Authorization

Figura 33. Entrypoint GET /lists

GET /lists/{id}

Entrypoint que permite obtener los detalles de una lista de acuerdo a su id.



GET /lists/{id}

Parameters

Try it out

Name	Description
Authorization * required string (header)	Authorization
id * required string (path)	id

Figura 34. Entrypoint GET /lists/{id}

POST /lists

Este endpoint permite crear listas.

POST

/lists

^

Parameters

Try it out

Name	Description
Authorization * required string (header)	<div>Authorization</div>

Request body required

application/json

▼

Example Value

Schema

```

{
  "name": "string",
  "description": "string",
  "circles": [
    {
      "id": 0,
      "name": "string",
      "description": "string",
      "creator": {
        "id": 0,
        "fireBaseId": "string",
        "alias": "string",
        "email": "string",
        "lastGeneratedRemindersTime": "2023-05-29T16:49:43.240Z"
      },
      "userList": [
        {
          "id": 0,
          "fireBaseId": "string",
          "alias": "string",
          "email": "string",
          "lastGeneratedRemindersTime": "2023-05-29T16:49:43.240Z"
        }
      ],
      "enabled": true
    }
  ]
}

```

Figura 35. Entrypoint POST /lists

PUT /lists/{id}

Este endpoint permite actualizar una lista.

PUT

/lists/{id}

Parameters

Try it out

Name	Description
Authorization * required string (header)	<input type="text" value="Authorization"/>
id * required string (path)	<input type="text" value="id"/>

Request body required

application/json

Example Value | Schema

```

{
  "name": "string",
  "description": "string",
  "circles": [
    {
      "id": 0,
      "name": "string",
      "description": "string",
      "creator": {
        "id": 0,
        "fireBaseId": "string",
        "alias": "string",
        "email": "string",
        "lastGeneratedRemindersTime": "2023-05-29T16:54:05.172Z"
      },
      "userList": [
        {
          "id": 0,
          "fireBaseId": "string",
          "alias": "string",
          "email": "string",
          "lastGeneratedRemindersTime": "2023-05-29T16:54:05.172Z"
        }
      ],
      "enabled": true
    }
  ]
}

```

Figura 36. Entrypoint PUT /lists/{id}

GET /lists/visible

Este endpoint recupera las listas que un usuario puede ver, ya sea porque es su dueño o están compartidas con un círculo al que pertenece.

GET

/lists/visible

Parameters

Try it out

Name	Description
Authorization * required string (header)	<input type="text" value="Authorization"/>

Figura 37. Entrypoint PUT /lists/visible

Gestión de regalos

A diferencia de los demás recursos, los regalos no se guardan en la base de datos proporcionada por el backend de spring, sino que son guardados en la base de datos de firestore.

Pantallas:

Se han creado las pantallas necesarias para gestionar esta capacidad.

gift-list:



Figura 38. Pantalla de lista de mis regalos

gift-detail:

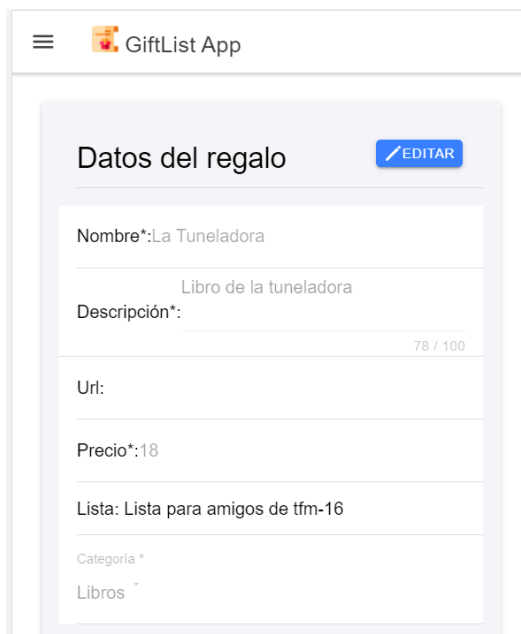


Figura 39. Pantalla de detalle de un regalo

Adicionalmente en la pantalla ya existente del usuario (*user-public-detail*) se mostrarán los regalos que estén en listas que el usuario consultante puede ver. Si el usuario consulta un perfil que no es el suyo, podrá reservar el regalo.

FrontEnd:

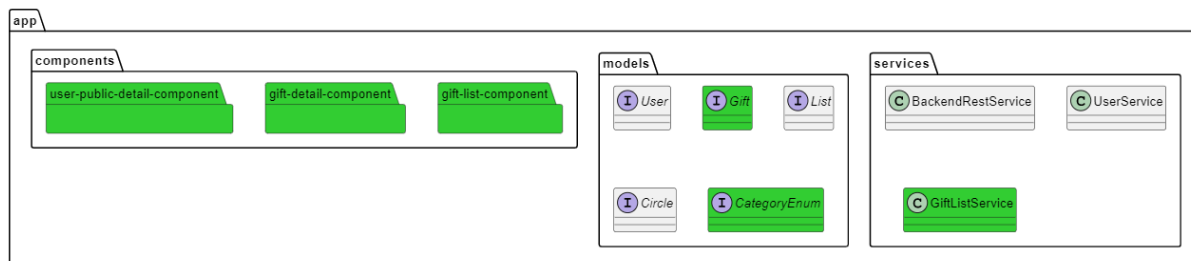


Figura 40. Diagrama de clases de Angular

Se han creado o actualizado los siguientes componentes para dar soporte a las vistas.

- *user-public-detail-component*
- *gift-detail-component*
- *gift-list-component*

Se han creado los modelos *Gift* y *CategoryEnum* que representan el regalo y la categoría del regalo del modelo de datos.

Modelo de datos:

En Firestore los regalos se almacenarán de la siguiente manera:

- Por cada usuario se genera una colección cuyo identificador es el id de firebase del usuario.
- Se añaden los regalos como documentos, que tendrán unos datos básicos como nombre, categoría, descripción, lista a la que pertenecen o url. Al tratarse de una base de datos noSQL el formato del modelo de datos es flexible y diferentes regalos podrán tener más datos, por ejemplo, un regalo podrá tener o no url.

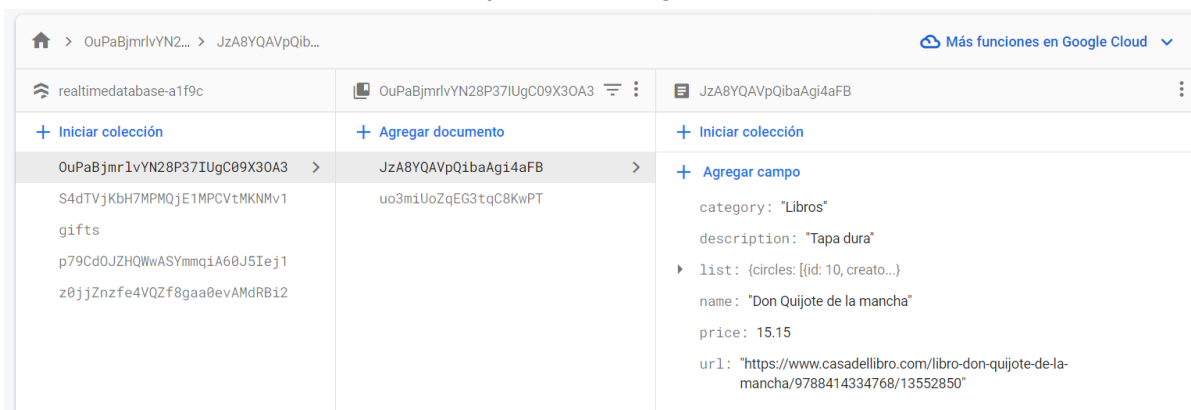


Figura 41. Datos de prueba en Firestore

Se ha creado un nuevo servicio que utiliza el SDK de Firestore (*@angular/fire/firestore*) para realizar las llamadas necesarias a Firestore para recuperar los datos de los regalos. En concreto se han desarrollado los siguientes métodos.

- `addGiftToList(gift: Gift)`. Añade para el usuario que tiene sesión iniciada.
- `getGifts(userIdPath: string): Observable<Gift[]>`. Obtiene los regalos de un path dado, normalmente el id del propio usuario.
- `getUsersGifts(userIdPath: string, validLists: number[])`. Obtiene los regalos de un usuario, normalmente un usuario distinto al que tiene sesión iniciada, en base a los ids de las listas que el usuario peticionario puede ver.
- `getGiftById(userIdPath: string, giftId: string, validLists: number[])`. Obtiene un regalo por id, si el usuario peticionario tiene suficientes permisos para ello.

Sprint Retrospective

Problemas detectados:

- El flujo de autodespliegue ha fallado durante este sprint ya que se necesitó actualizar la versión de las dependencias. La solución llevó un par de horas de investigación y quedó fuera del tiempo planificado.
- Los tests no tienen una cobertura de código extensa.

Plan de mejora:

- Se crea un ticket técnico de deuda técnica para afrontar la escasa cobertura del código, siendo el deseable un 80%.

Sprint 3

Sprint planning

Del 15 al 28 de mayo de 2023.

En este Sprint se van a realizar las siguientes tareas:

Identificador	Título	SP
TS08	TS08 Debt Tech, ensure 80% of test coverage	5 SP
US19	Añadir característica de usuario	5 SP
US20	Consultar características de usuario	2 SP
US21	Actualizar característica de usuario	2 SP
US22	Borrar característica de usuario	2 SP
US23	Consultar recordatorios	5 SP
US24	Descartar recordatorio	1 SP
US25	Crear mensaje	2 SP
US26	Consultar mensajes recibidos	2 SP
US27	Leer mensajes	1 SP
US28	Responder mensaje	1 SP

Sprint Review

Title	...	Mil...	...	Tipo	...	SP	...	Time consumed	...
27	✓ US19 Añadir característica de usuario #27	Sprin...	...	User Story	...	5 Points	...	4	...
28	✓ US20 Consultar características de usuario #28	Sprin...	...	User Story	...	2 Points	...	2	...
29	✓ US21 Actualizar característica de usuario #29	Sprin...	...	User Story	...	2 Points	...	2	...
30	✓ US22 Borrar característica de usuario #30	Sprin...	...	User Story	...	2 Points	...	1	...
31	✓ US23 Consultar recordatorios #31	Sprin...	...	User Story	...	5 Points
32	✓ US24 Descartar recordatorio #32	Sprin...	...	User Story	...	1 Point
33	✓ US25 Crear mensaje #33	Sprin...	...	User Story	...	2 Points	...	2	...
34	✓ US27 Leer mensajes #35	Sprin...	...	User Story	...	1 Point	...	1	...
35	✓ US26 Consultar mensajes recibidos #34	Sprin...	...	User Story	...	2 Points	...	3	...
36	✓ US28 Responder mensaje #36	Sprin...	...	User Story	...	1 Point	...	1	...
37	✓ TS08 Debt Tech, ensure 80% of test coverage #37	Sprin...	...	Technical	...	5 Points	...	7	...

Figura 42. Lista de tareas de Sprint 3 en Github Project

En este Sprint se han afrontado las funcionalidades de gestionar características de usuario, crear mensajes privados entre usuarios, gestionar recordatorios y una tarea técnica para mejorar los tests existentes en la aplicación.

En total se han gastado 23 horas de desarrollo para las tareas finalizadas, sin embargo, al finalizar el Sprint no se han podido finalizar las tareas relacionadas con la gestión de recordatorios. Por ello, de acuerdo a la metodología no se deben tomar en cuenta y devolverlas al backlog.

Modelo de datos

Para dar soporte a las nuevas historias de usuario se han generado nuevas tablas en la base de datos. En la Figura 43 se muestra un diagrama de clases que refleja el modelo de datos. En este Sprint se han creado 3 nuevas clases: Trait, Message, ThreadMessage y tienen relaciones entre ellas o entre clases ya existentes.

Trait, característica, tiene una relación N:M con List. Una característica se puede compartir con N lista y una lista puede tener compartida más de una característica. En la clase Trait existe en consecuencia un atributo lists de tipo array. Adicionalmente Trait tiene una relación 1:N con User, que representa que una característica tiene un dueño.

ThreadMessage es una clase que representa un hilo de conversación. Esta clase tiene una relación N:M con user. Un hilo tendrá dos o más usuarios, aunque en este Sprint permitirá la creación de un hilo entre dos usuarios solamente. Un usuario podrá estar en varios hilos.

Message es una clase que representa un mensaje individual de un hilo, por ello el mensaje tiene una relación con 1:N con ThreadMessage. Además un mensaje es creado por un usuario por lo que Message tiene una relación 1:N con User.

Test

Los test deberían entregarse a medida que el código es entregado. Sin embargo, la cobertura de los test existentes tras la finalización del Sprint 2 era escasa, así que se ha propuesto la historia técnica de asegurar el 80% de cobertura.

Para ello se han realizado principalmente dos tareas:

1. Crear un juego de datos para los tests que se crean y destruyen en cada ejecución solamente cuando se utiliza un perfil de desarrollo. Este juego de datos está definido en la clase SeederDev.

```
@Repository
@Profile("dev")
public class SeederDev {
```

Figura 44. Anotaciones de la clase SeederDev

2. Se ha creado una serie de test unitarios que prueben cada una de las clases. Es importante que los test cubran la mayor parte del código especialmente el código de los servicios, pues son las piezas de datos más críticas del sistema.

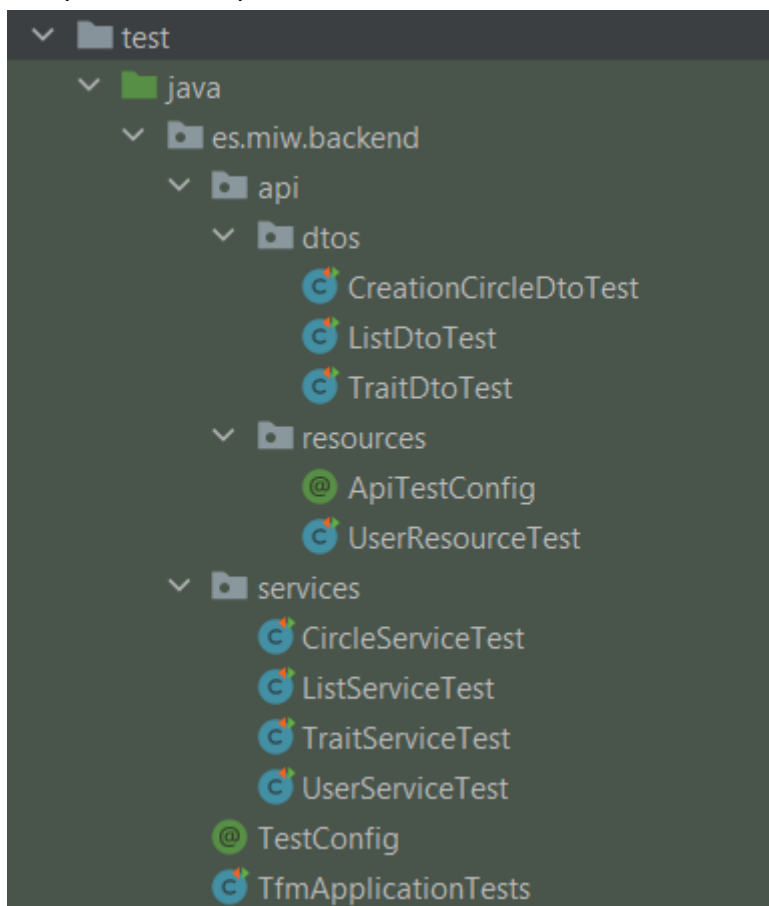


Figura 45. Estructura de las clases de Test

Si se ejecutan los tests en un IDE como IntelliJ, se puede especificar que se calcule la cobertura de los mismos. Tras programar y mejorar los tests, el resultado de la cobertura es de un 80% en el global de las clases, siendo la cobertura prácticamente de 100% en los servicios:

Element	Class, %	Method, %	Line, %
es	92% (23/25)	73% (85/115)	81% (213/261)
miw	92% (23/25)	73% (85/115)	81% (213/261)
backend	92% (23/25)	73% (85/115)	81% (213/261)
api	90% (10/11)	49% (25/51)	51% (42/82)
dtos	100% (6/6)	65% (17/26)	73% (25/34)
CircleDto	100% (2/2)	50% (4/8)	50% (4/8)
ListDto	100% (2/2)	44% (4/9)	61% (8/13)
UserDto	100% (2/2)	100% (9/9)	100% (13/13)
http_errors	50% (1/2)	0% (0/7)	8% (1/12)
ApiExceptionHandler	100% (1/1)	0% (0/2)	25% (1/4)
ErrorMessage	0% (0/1)	0% (0/5)	0% (0/8)
resources	100% (3/3)	44% (8/18)	44% (16/36)
CircleResource	100% (1/1)	12% (1/8)	17% (3/17)
ListResource	100% (1/1)	25% (1/4)	33% (3/9)
UserResource	100% (1/1)	100% (6/6)	100% (10/10)
configurations	100% (1/1)	100% (1/1)	81% (9/11)
FirebaseInitializer	100% (1/1)	100% (1/1)	81% (9/11)
data	87% (7/8)	92% (37/40)	96% (84/87)
daos	100% (1/1)	100% (4/4)	100% (47/47)
CircleRepository	100% (0/0)	100% (0/0)	100% (0/0)
ListRepository	100% (0/0)	100% (0/0)	100% (0/0)
SeederDev	100% (1/1)	100% (4/4)	100% (47/47)
UserRepository	100% (0/0)	100% (0/0)	100% (0/0)
models	85% (6/7)	91% (33/36)	92% (37/40)
validations	0% (0/1)	0% (0/2)	0% (0/2)
Circle	100% (2/2)	100% (14/14)	100% (17/17)
List	100% (2/2)	90% (9/10)	90% (9/10)
User	100% (2/2)	100% (10/10)	100% (11/11)
services	100% (4/4)	100% (22/22)	97% (77/79)
exceptions	100% (1/1)	100% (1/1)	100% (1/1)
ForbiddenException	100% (1/1)	100% (1/1)	100% (1/1)
CircleService	100% (1/1)	100% (9/9)	100% (33/33)
ListService	100% (1/1)	100% (5/5)	100% (8/8)
UserService	100% (1/1)	100% (7/7)	94% (35/37)
TfmApplication	100% (1/1)	0% (0/1)	50% (1/2)

Figura 46. Cobertura de los Test

Características de usuarios

En este Sprint se han afrontado las tareas necesarias para guardar características de un usuario. Estas características se podrán mostrar en la pantalla de detalles de un regalo y se utilizarán para generar recordatorios para un usuario.

Pantallas

start-component

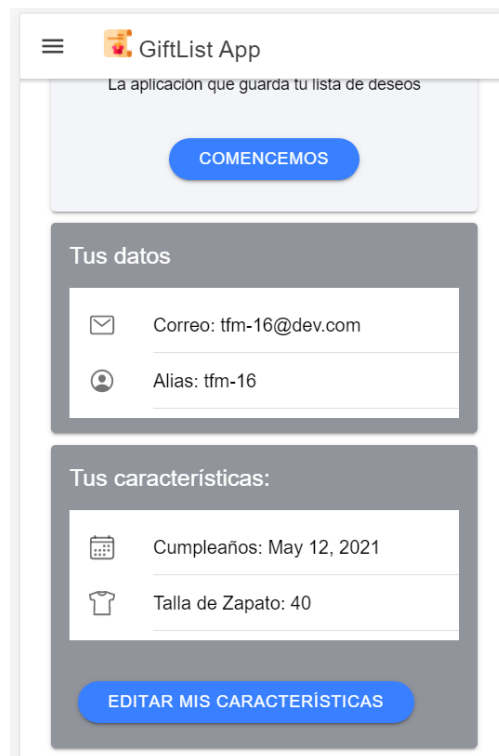


Figura 47. Características en la pantalla de usuario

trait-swipper

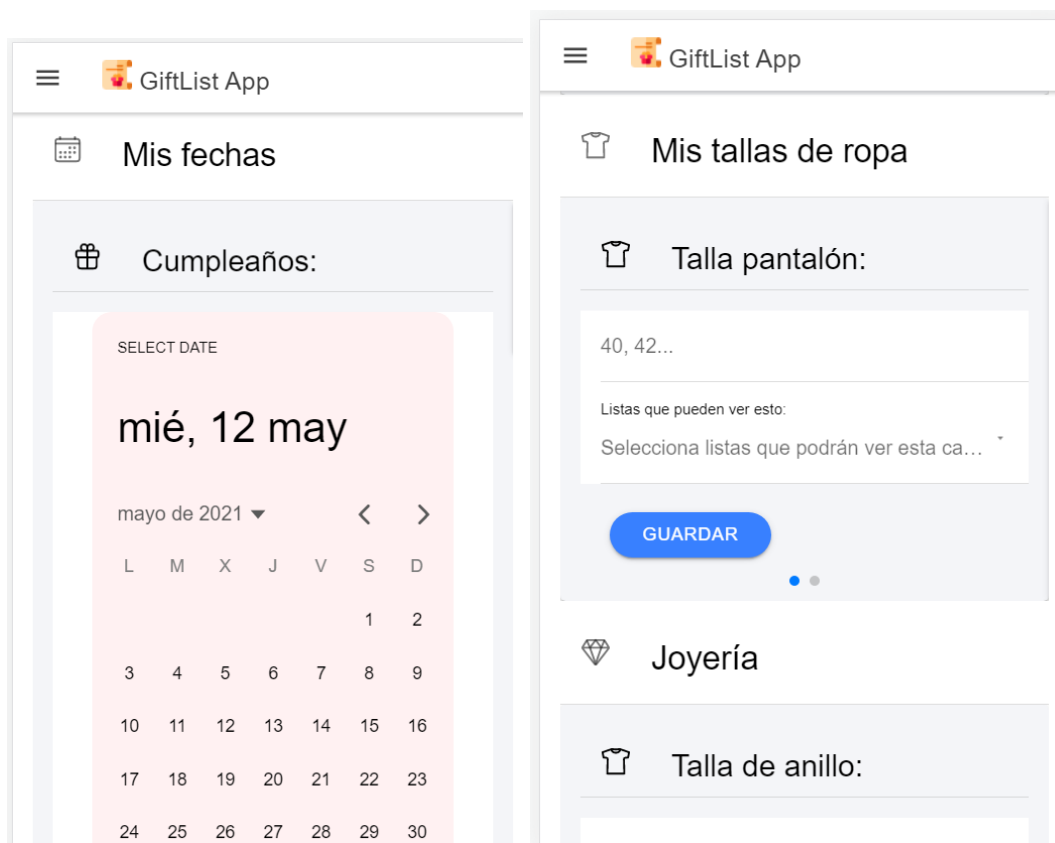


Figura 48. Formulario de creación y edición de características

gift-detail

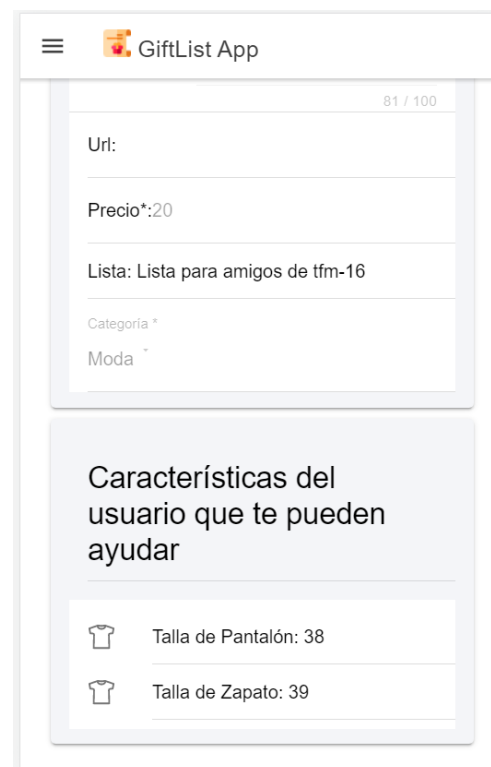


Figura 49. Características recomendadas de usuario en detalle de regalo

Frontend:

Se han actualizado o creado las siguientes clases y componentes:

- trait-swipper: un componente que contiene un formulario para añadir y editar características.
- startl-component: este componente se ha actualizado para mostrar las características del usuario en su perfil
- gift-detail-component: este componente se ha actualizado para mostrar las características del usuario relevantes para el tipo de regalo.
- Trait: se ha creado esta interfaz como modelo de datos.
- BackendRestService: se ha actualizado el servicio para que realice llamadas necesarias al backend.

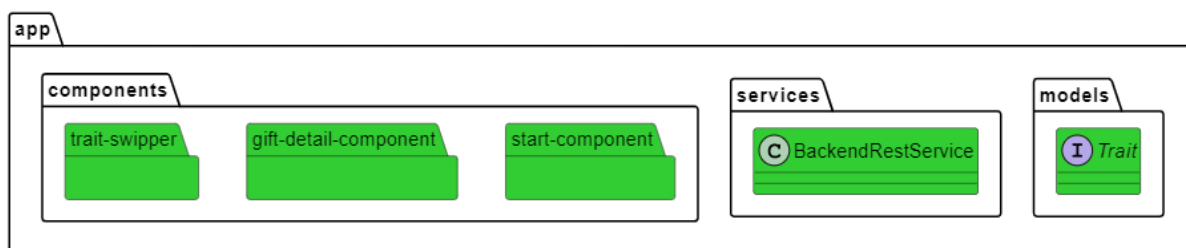


Figura 50. Diagrama de clases en Angular

Backend:

Siguiendo la misma lógica que en el Sprint 1 se han creado los siguientes endpoints.

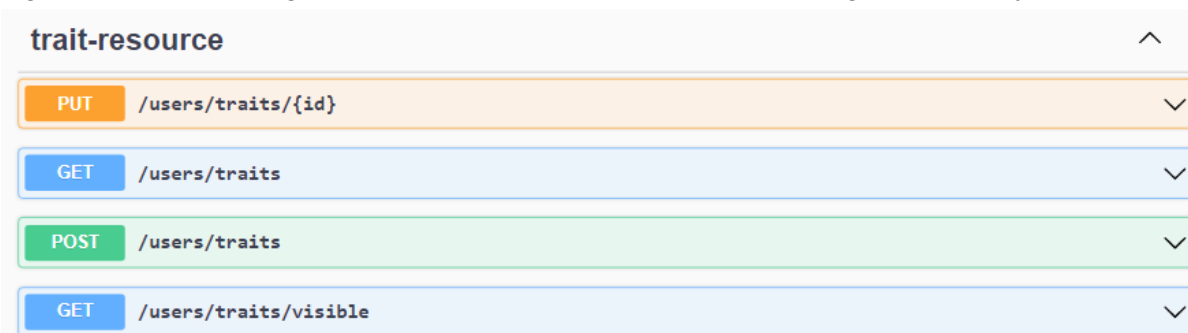


Figura 51. Endpoints de características

POST /user/traits

Permite crear una característica

GET /user/traits

Permite obtener características del usuario que realiza la petición

GET /user/traits/visible

Permite obtener las características de otro usuario que el usuario peticionario puede ver según tipo de regalo e id de la lista en la que está el regalo.

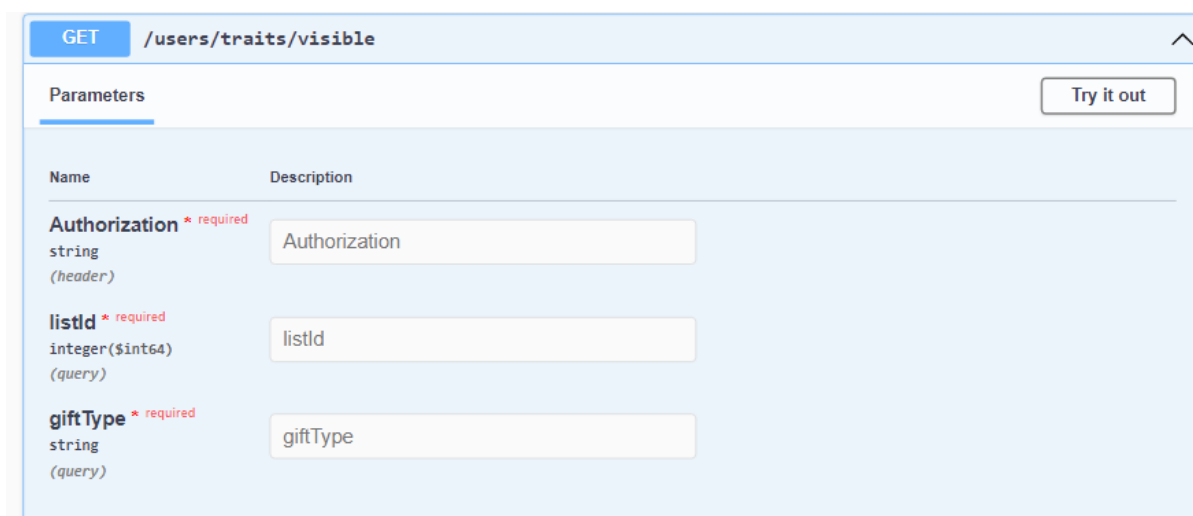


Figura 52. Endpoints de características visibles, detalle

PUT /user/traits/{id}

Permite actualizar el valor de una característica.

Mensajes privados

Los usuarios pueden compartir mensajes en la app.

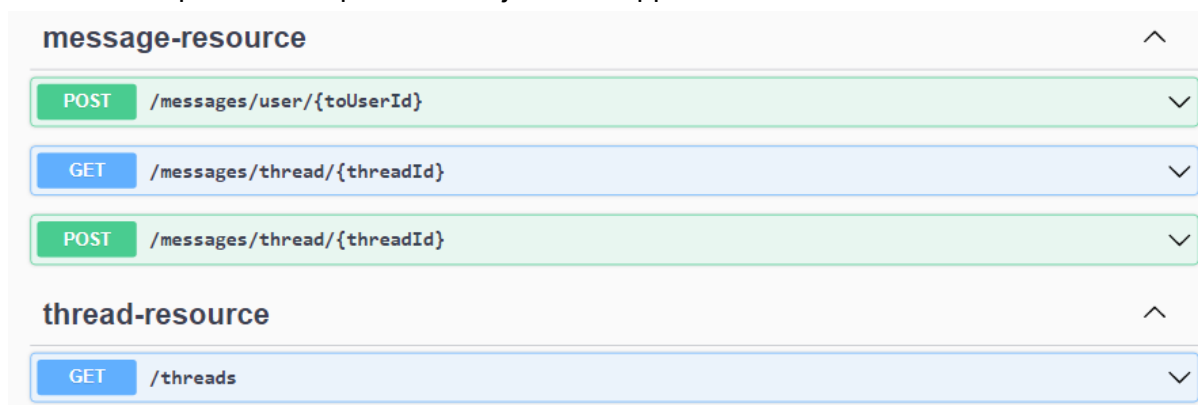


Figura 53. Endpoints de mensajes

Sprint Retrospective

Seguimiento del plan de mejora:

- Se han incorporado tests automáticos para elevar la cobertura al 80%.

Problemas detectados:

- Se planifican demasiadas tareas para el Sprint y eso hace que no se consigan todas ellas. En este Sprint unos pequeños bugs en la tarea de recordatorios hizo que se retrasara hasta la siguiente entrega

Plan de mejora:

- Planificar menos tareas por Sprint.

Sprint 4

Sprint planning

Del 29 de mayo al 12 de junio de 2023

Identificador	Título	SP
US23	Consultar recordatorios	5 SP
US24	Descartar recordatorio	1 SP
US33	Crear amigo invisible	5 SP
US34	Consultar amigo invisible	2 SP
US35	Finalizar amigo invisible	2 SP
TS09	Análisis OWASP	5 SP

Sprint review



















Title	Milestone	Tipo	SP	Time consumed
2  US33 Consultar amigo invisible #44	Sprint 4	 User Story	 2 Points	2
32  US23 Consultar recordatorios #31	Sprint 4	 User Story	 5 Points	4
33  US24 Descartar recordatorio #32	Sprint 4	 User Story	 1 Point	1
34  US33 Crear amigo invisible #42	Sprint 4	 User Story	 5 Points	4
35  US35 Finalizar amigo invisible #43	Sprint 4	 User Story	 2 Points	1
41  TS09 Análisis OWASP #40	Sprint 4	 User Story	 5 Points	5

Figura 54. Lista de tareas del Sprint 4

En el Sprint 4 se han afrontado las historias de usuarios para gestionar los recordatorios y generar juegos de amigos invisibles, así como un breve análisis de la seguridad de la web.

Se han utilizado unas 12 horas de puro desarrollo, sumadas a 5 horas de análisis de la herramienta.

Modelo de datos

En la Figura 54 se puede ver el diagrama de clases que refleja el modelo de datos del Sprint 4. Como novedad respecto el Sprint 3, se le han añadido 3 nuevas clases: Reminder, que corresponde al mensaje sobre un usuario que otro usuario recibe para recordarle sobre una fecha importante, PalDrawing que representa la instancia de un juego invisible y PalPair que representa las parejas asignadas en el juego del amigo invisible.

Reminder se relaciona dos veces con usuario con una relación 1:N. Esto es porque por un lado un usuario será el dueño del recordatorio y por otro lado un usuario será al que le afecta el recordatorio.

PalDrawing también tiene una relación 1:N con User, representando que un evento de amigo invisible tiene un creador. Además tiene relación 1:N con Circle porque un evento de amigo invisible se realiza para los usuarios de un círculo.

PalPair es la pareja de usuarios resultante del sorteo de amigo invisible. Por tanto tiene una relación 1:N doble con User, una para determinar el dueño, es decir, la persona que deberá hacer el regalo y su pareja, la persona a la que deberá hacer el regalo. Adicionalmente PalPair tiene una relación N:1 con PalDrawing pues las parejas solamente son válidas para un juego de amigo invisible concreto.

Adicionalmente, en la clase User se ha añadido un campo, *lastGeneratedRemindersTime*, que muestra la última vez que se han calculado los recordatorios de un usuario.

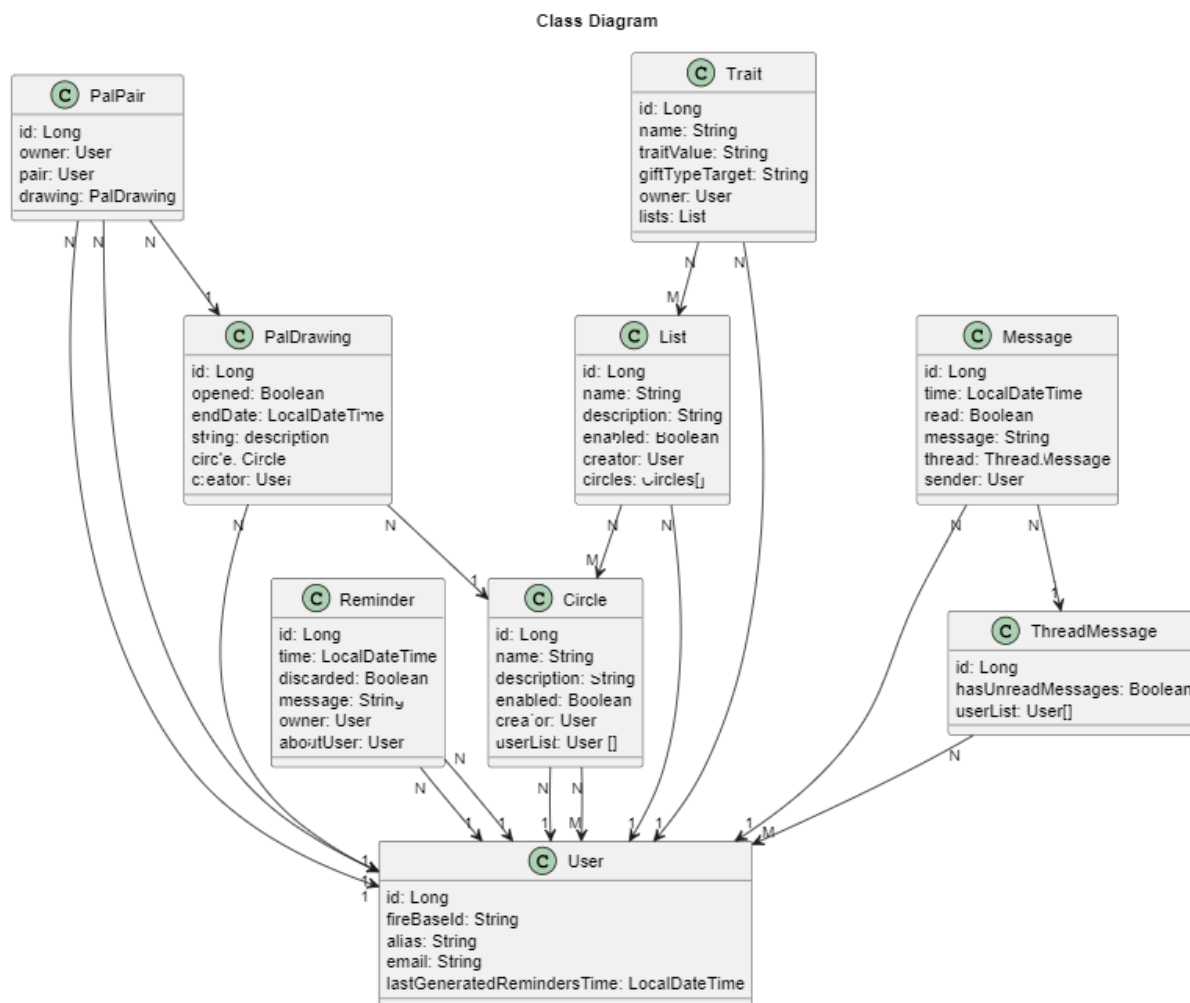


Figura 55. Modelo de datos en Sprint 4

Recordatorios

En este Sprint se han afrontado las tareas necesarias para generar recordatorios a partir de las características de un usuario. Estos recordatorios se podrán ver en la pantalla principal del usuario, como se ve en la Figura 55.

En concreto, se generan recordatorios basados en la fecha de cumpleaños de usuarios que están en los círculos del usuario, siempre y cuando esa fecha de cumpleaños se haya compartido adecuadamente.

Cuando un usuario accede a la pantalla principal, una vez al día, se calculan los cumpleaños que están a menos de un mes de distancia y no han generado todavía un aviso.

Los recordatorios se pueden descartar y el recordatorio se ocultará.

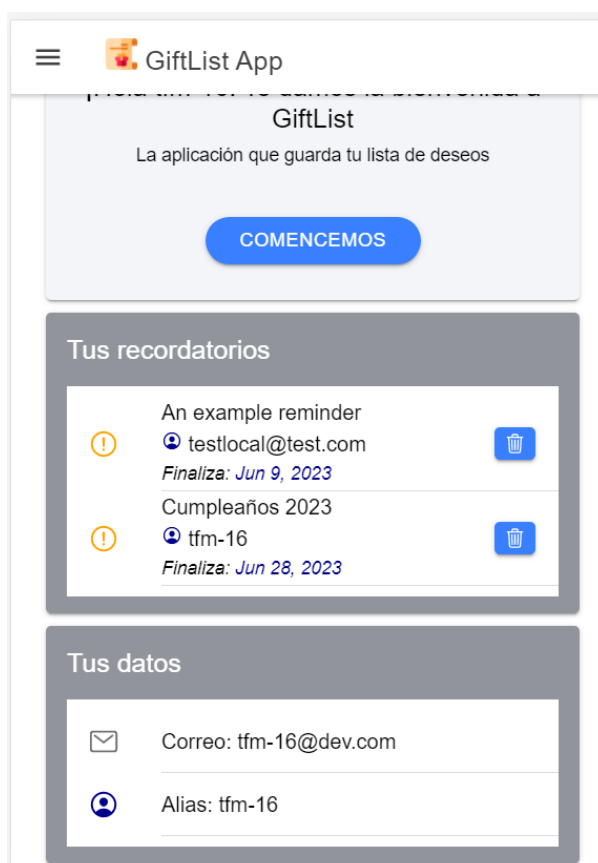


Figura 56. Pantalla de usuario con recordatorios

Para dar soporte a estas historias de usuario se han modificado los siguientes componentes ya existentes.

- start-component: Se crea un apartado de recordatorios en la vista, se solicita al servicio los recordatorios y se permite solicitar al servicio descartar un recordatorio concreto.

- BackendRestService: Se crean los métodos para realizar las llamadas a los endpoints correspondientes.
- Reminder: Modelo de datos utilizado para recuperar los datos del backend.

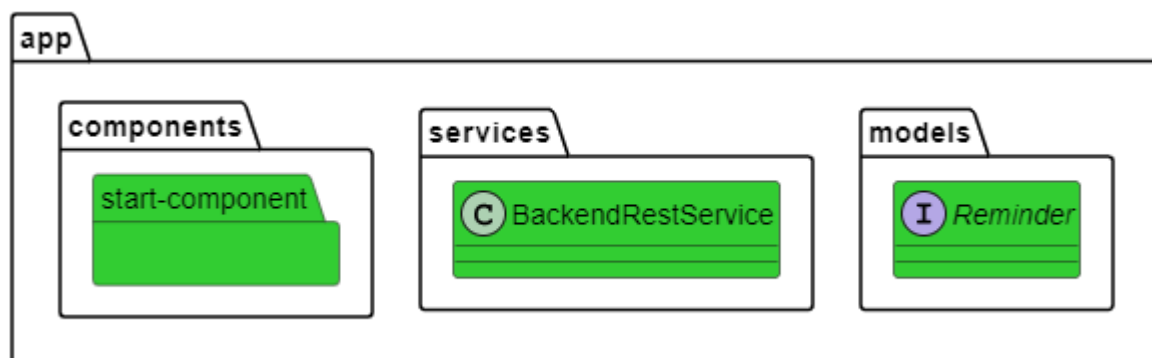


Figura 57. Diagrama de componentes modificados en Angular

Para dar soporte a las necesidades de las historias de usuarios se han generado los siguientes entypoints.

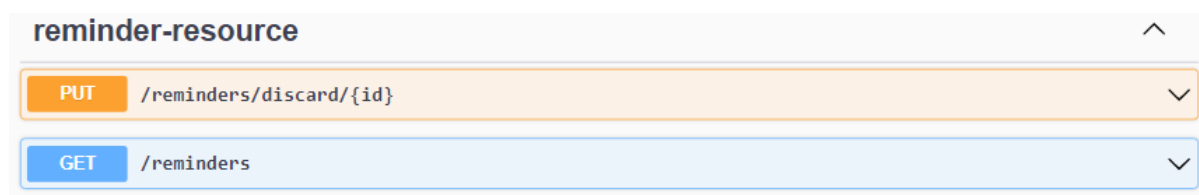


Figura 58. Endpoints creados para las historias de recordatorios

GET /reminders

Obtiene los recordatorios del usuario solicitante. Si es la primera vez en el día que se consultan los recordatorios se hacen las consultas necesarias para generarlos y se devuelven todos los recordatorios. Las sucesivas veces que se ejecuta el método, se devuelven los recordatorios almacenados del usuario solicitante.

PUT /reminders/discard/{id}

Se marca como descartado el recordatorio con el id pasado por parámetro.

Amigo invisible

En este Sprint se han afrontado las tareas necesarias para generar juegos de amigo invisible. En el juego del amigo invisible, se realiza un sorteo entre personas de un grupo (círculo). Cada persona recibe el nombre de otra persona del grupo y acuerdan en una fecha concreta intercambiarse regalos sin saber quién es la persona que realiza el regalo.

En la web, se habilita una opción en la pantalla de detalles del círculo. El creador del círculo podrá iniciar el juego, seleccionando en un menú la fecha y la descripción deseada para el juego, Figura 58.

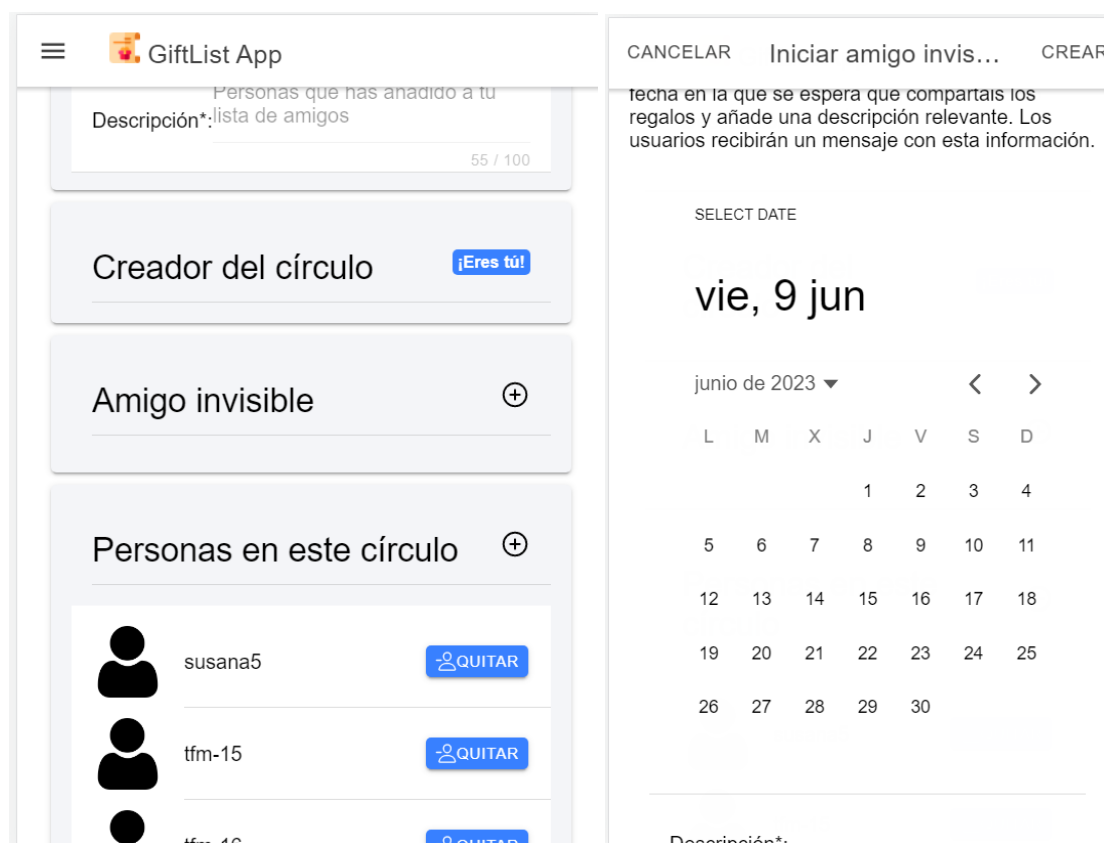


Figura 59. Pantalla detalle de círculos con modal de creación

Una vez generado el amigo invisible los usuarios podrán consultar en la pantalla de detalles del juego en la misma pantalla, Figura 59.

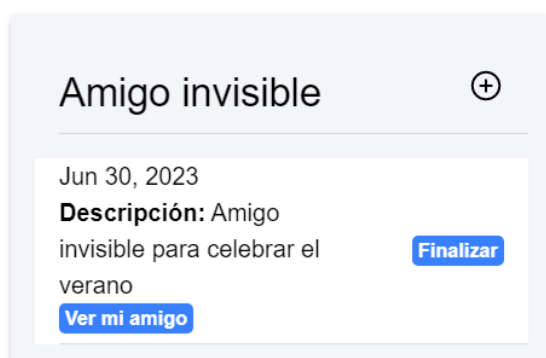


Figura 60. Pantalla detalle de círculos con amigo invisible en curso

Para dar soporte a estas pantallas se han modificado los siguientes componentes del proyecto Angular:

- circle-detail-component: se crean las vistas y se solicita al servicio los datos requeridos para mostrar los datos.
- BackendRestService: Se crean los métodos para llamar a los Endpoints creados para la gestión de juegos de amigo invisible.
- PalDrawing: Modelo de datos de un juego de amigo invisible.

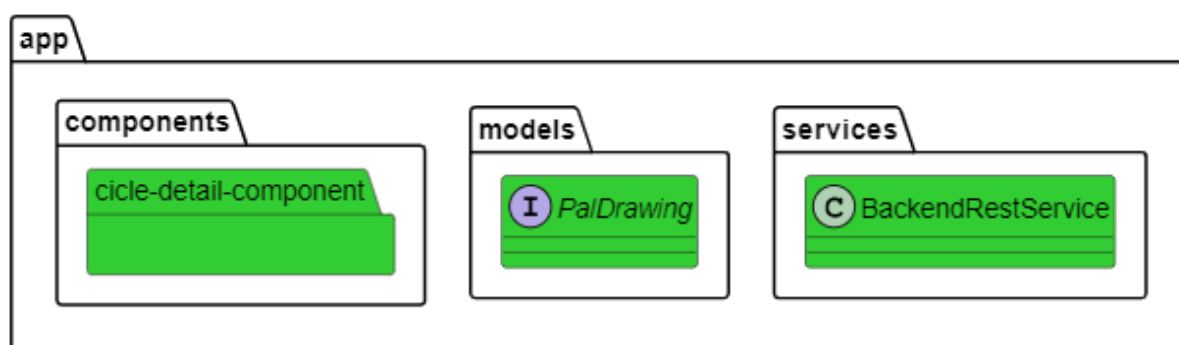


Figura 61. Diagrama de clases modificadas en Angular para las historias de amigo invisible

En el backend, en contrapartida, se han habilitado los siguientes endpoints.

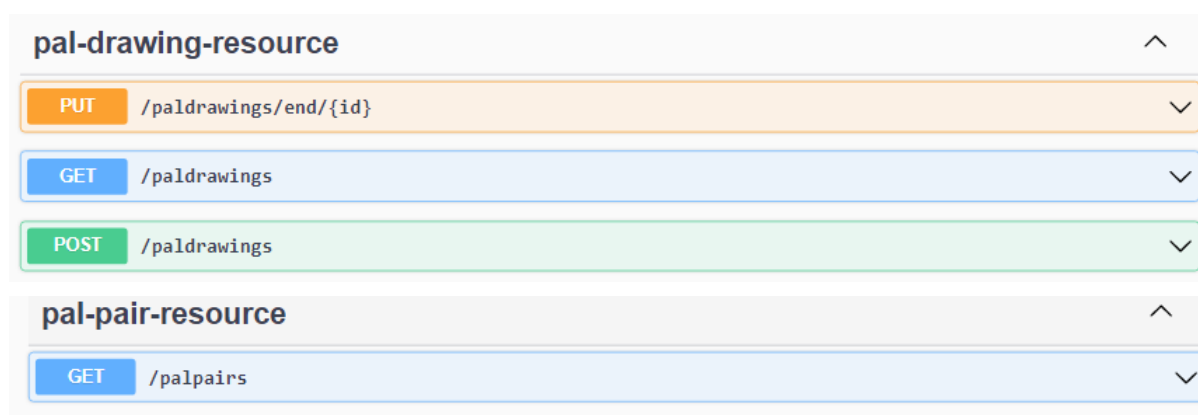


Figura 62. Endpoints creados para las historias de amigo invisible

POST /paldrawings

Endpoint que permite crear un juego de amigo invisible. A partir de un círculo, un usuario creador y una fecha de fin, se genera un sorteo para encontrar parejas de jugadores. Un usuario nunca será pareja de sí mismo, cada usuario tendrá una persona a la que hacer un regalo y cada usuario será la pareja de alguien.

GET /paldrawings

Endpoint que permite obtener los juegos de amigo invisible en curso para un círculo.

PUT /paldrawings/end/{id}

Endpoint que permite finalizar un juego de amigo invisible. Al finalizar el juego, este no será devuelto por el método GET /paldrawings

GET /palpairs

Obtiene la pareja del usuario solicitante dado un id de círculo y un id de juego invisible.

Test

Como es habitual, los test automáticos para garantizar la calidad del código se desarrollan a medida que se desarrolla el código. En este Sprint se ha obtenido un 80% de líneas cubiertas por, al menos, un test.

Element	Class, %	Method, %	Line, %
es	90% (49/54)	68% (181/264)	80% (559/692)
miw	90% (49/54)	68% (181/264)	80% (559/692)
backend	90% (49/54)	68% (181/264)	80% (559/692)
TfmApplication	100% (1/1)	0% (0/1)	50% (1/2)
services	100% (10/10)	100% (57/57)	98% (248/253)
UserService	100% (1/1)	100% (7/7)	92% (35/38)
TraitService	100% (1/1)	100% (6/6)	100% (14/14)
ThreadService	100% (1/1)	100% (5/5)	100% (15/15)
ReminderService	100% (1/1)	100% (6/6)	95% (45/47)
PalPairService	100% (1/1)	100% (3/3)	100% (11/11)
PalDrawingService	100% (1/1)	100% (7/7)	100% (44/44)
MessageService	100% (1/1)	100% (4/4)	100% (33/33)
ListService	100% (1/1)	100% (7/7)	100% (15/15)
CircleService	100% (1/1)	100% (11/11)	100% (35/35)
exceptions	100% (1/1)	100% (1/1)	100% (1/1)
data	100% (19/19)	87% (88/101)	94% (230/243)
models	100% (18/18)	86% (84/97)	87% (92/105)
User	100% (2/2)	100% (11/11)	100% (12/12)
Trait	100% (2/2)	100% (11/11)	100% (11/11)
ThreadMessage	100% (2/2)	87% (7/8)	87% (7/8)
Reminder	100% (2/2)	72% (8/11)	72% (8/11)
PalPair	100% (2/2)	66% (6/9)	66% (6/9)
PalDrawing	100% (2/2)	81% (9/11)	81% (9/11)
Message	100% (2/2)	63% (7/11)	63% (7/11)
List	100% (2/2)	100% (11/11)	100% (14/14)
Circle	100% (2/2)	100% (14/14)	100% (18/18)
daos	100% (1/1)	100% (4/4)	100% (138/138)
configurations	100% (1/1)	100% (1/1)	81% (9/11)
api	78% (18/23)	33% (35/104)	38% (71/183)
resources	100% (9/9)	34% (14/41)	39% (35/89)
http_errors	50% (1/2)	0% (0/7)	8% (1/12)
dtos	66% (8/12)	37% (21/56)	42% (35/82)

Figura 63. Cobertura final de los tests

Como ya se ha indicado en anteriores Sprints, es importante que la cobertura de los servicios sea muy elevada. En este caso hay un 100% en clases y métodos y un 98% en líneas.

Análisis ZAP OWASP

OWASP (Open Worldwide Application Security Project) [25] es un proyecto de código abierto para la ciberseguridad que se utiliza como referente para las auditorías de seguridad informática. La Fundación OWASP es un organismo sin ánimo de lucro que gestiona la infraestructura y los diferentes proyectos que mejoran la seguridad del Software.

OWASP ZAP (Zed Attack Proxy) [26] es una herramienta de código abierto gestionada por OWASP que permite analizar las vulnerabilidades de una web a través del análisis dinámico de código, es decir, se analiza la ejecución de la web. Este tipo de análisis no analiza el código fuente, sino el producto de la ejecución, se trata entonces de un análisis de caja negra.

Para encontrar las vulnerabilidades de este proyecto primeramente se realiza un análisis pasivo y posteriormente uno activo.

El análisis pasivo consiste en *escuchar* las llamadas a la web utilizando ZAP en modo proxy. ZAP intercepta los datos y analiza si las vulnerabilidades de su base de datos se encuentran presentes.

Para realizar este análisis, se coloca a ZAP en modo proxy, de manera que inmediatamente el navegador detecta que hay un problema con el certificado. Firebase proporciona un certificado de “Google Trust Services LLC” gratuito. Este certificado garantiza que las comunicaciones están cifradas. Fly.io proporciona similarmente un certificado de “Let’s Encrypt”.

Al intentar acceder tanto a la url del frontend como la del backend, el navegador muestra una advertencia indicando que puede estar ocurriendo un ataque.



Un software que impide a Firefox conectarse de forma segura a este sitio

tfm-backend.fly.dev probablemente es un sitio seguro, pero no se ha podido establecer una conexión segura. Este problema está causado por **OWASP Zed Attack Proxy Root CA**, que es un programa en su ordenador o en su red.

¿Qué puede hacer al respecto?

- Si su antivirus incluye una función que escanea conexiones cifradas (normalmente llamada “escáner web” o “escáner https”), puede desactivar esa función. Si eso no funciona, puede eliminar y volver a instalar el programa antivirus.
- Si está en una red corporativa, puede ponerse en contacto con su departamento de informática.
- Si no está familiarizado con **OWASP Zed Attack Proxy Root CA**, entonces esto puede ser un ataque y no debería acceder al sitio.

[Más información...](#)

Retroceder (recomendado)

Avanzado...

Figura 64. Advertencia del navegador

Por otro lado, todas las peticiones a Firebase fallan, pues detecta que el origen no es legítimo. De esta manera no se pueden realizar peticiones de registro, ni peticiones autenticadas al backend, ya que el token de firebase nunca se solicita.

Desactivando el proxy podemos bypassar el login, pero al activarlo de nuevo, las peticiones al backend fallan, se pueden ver las diferentes pantallas, pero no obtener datos. Se realiza un recorrido de la aplicación para que pasivamente ZAP realice el análisis.

Tras el recorrido se han detectado las siguientes amenazas.

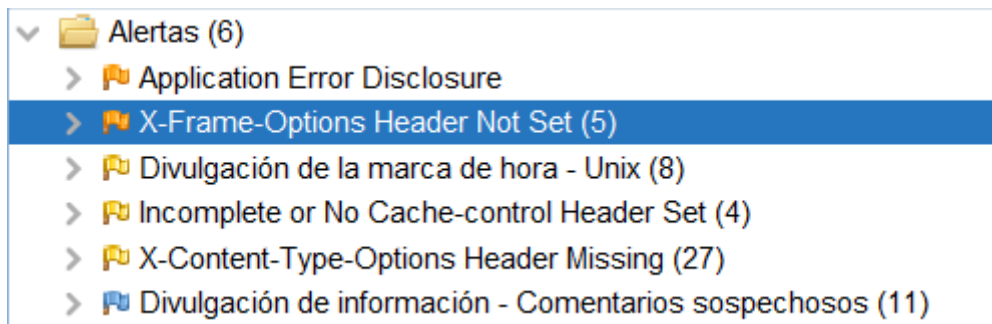


Figura 65. Amenazas detectadas en recorrido manual

- Application Error Disclosure. Riesgo medio. Tras realizar un login fallido se muestra al usuario un error de aplicación, en este caso la denegación por parte de Firebase. Presentar esta información podría facilitar ataques.
Solución propuesta: no mostrar mensajes de error al usuario.
- X-Frame-Options Header Not Set. Riesgo medio. La web puede insertarse en otros sitios web sin límites, lo que podría ocasionar que un atacante malicioso crease una web maliciosa transparente e insertase la web original debajo para engañar a los usuarios y robar su información.
Solución propuesta: habilitar X-Frame-Options en la configuración de firebase.
- Divulgación de la marca de hora: Se trata de un falso positivo, detecta números como marcas de hora (fechas de 1970).
- Incomplete or No Cache-control Header Set. Riesgo bajo. Al no estar habilitada esta cabecera, diferentes navegadores o proxies pueden guardar datos de caché.
Solución propuesta: si es posible habilitar esta cabecera.
- X-Content-Type-Options Header Missing: Para evitar ataques de tipo MIME sniffing, se recomienda indicar el tipo de contenido de los ficheros.
Solución propuesta: Indicar esta cabecera como no-sniff y siempre que sea posible especificar el tipo de contenido del fichero.
- Comentarios sospechosos: Riesgo indefinido. El error de Firebase se está mostrando al usuario, lo que puede ser utilizado por un posible atacante.
Solución propuesta: No revelar el error de Firebase al usuario.

El análisis activo consiste en intentar atacar la web, replicando ataques conocidos para explotar vulnerabilidades con el objetivo de probar si la vulnerabilidad está presente en la web.

Para ello primeramente se intenta obtener la mayor información posible sobre la aplicación creando una araña web con la herramienta. Tras la creación de la araña, se detectan nuevas amenazas del mismo tipo que las obtenidas con el recorrido manual.

La araña web ha podido detectar algunas rutas más de la app.

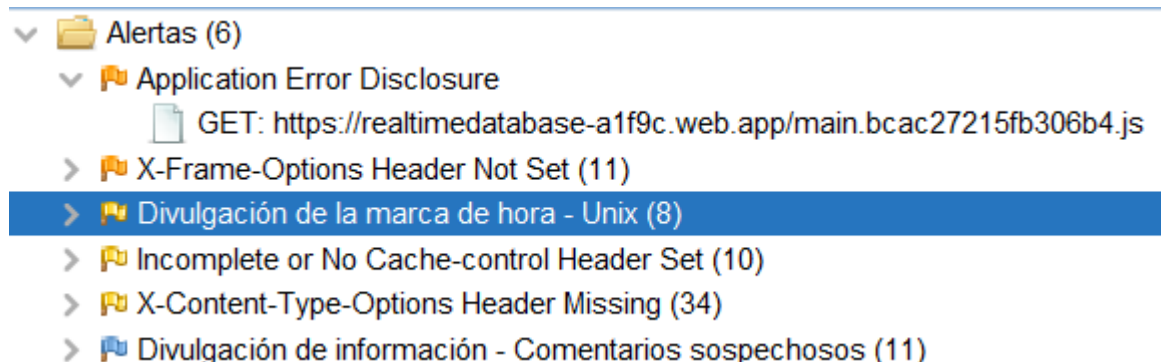


Figura 66. Amenazas detectadas con araña web

Una vez se detecta toda la información posible, el siguiente paso es atacar la web con un test de penetración. La herramienta permite atacar automáticamente la aplicación intentando vulnerar la seguridad con técnicas conocidas.

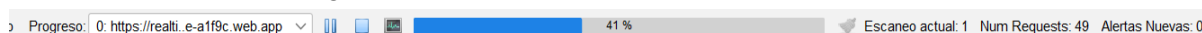


Figura 67. Escaneo en curso

Se ha ejecutado un ataque de pentesting para el frontal y otro para el backend. El resultado de los test sin embargo no arroja nuevas vulnerabilidades a las ya encontradas por el escaneo pasivo.

En definitiva, se han estudiado las vulnerabilidades de la web y se han encontrado posibles acciones para mitigarlas.

Sprint retrospective

Seguimiento de los planes de mejora:

- Al añadir menos tareas en este Sprint, se han completado todas las tareas del Sprint.

Problemas detectados:

- La integración continua ha dado algunos problemas porque se ha tenido que escalar el tamaño de memoria de la máquina del servidor.
- Se han encontrado diferentes amenazas en la web en producción que se pueden mitigar. Se debe crear una serie de historias técnicas que apliquen la solución propuesta.

Tiempos del proyecto

A lo largo del proyecto se ha utilizado la herramienta Wakatime[24], que permite medir el número de horas dedicado a programar un proyecto. Wakatime mide el tiempo que el IDE está activo en primer plano y ofrece una estimación del tiempo total utilizado para el desarrollo.

Este tiempo es una estimación y no tiene en cuenta el tiempo de diseño sobre herramientas, redacción y revisión de documentación ni investigación sobre las tecnologías. Sin embargo, estos datos pueden ser de utilidad ya que dan una visión global del esfuerzo dedicado sobre cada proyecto.

A lo largo del proyecto se han incorporado los tiempos utilizados en cada tarea teniendo en cuenta los datos de Wakatime, figuras 8, 42 y 53.

El tiempo utilizado para los proyectos es ligeramente superior a la suma de todas las tareas ya que algunos errores de configuración de la integración continua o pruebas de concepto se han realizado fuera de las ramas de cada tarea del proyecto.

Para el backend, en total, se han utilizado 45 horas y 53 minutos de desarrollo.

Projects • backend

total 45 hrs 53 mins



Figura 68. Tiempo de desarrollo del proyecto backend

Para el frontend, en total, se han utilizado 55 horas y 27 minutos de desarrollo.

Projects • tfm-angular

total 55 hrs 27 mins



Figura 69. Tiempo de desarrollo del proyecto frontend

El seguimiento de estos datos ha ayudado a tomar consciencia del tiempo necesario para cada una de las tareas realizadas y arroja un dato imprevisto. Las tareas de frontend, subestimadas en el proyecto, en total han durado 10 horas más que las de backend para las que se ha reservado más tiempo.

Conclusiones

Durante este Trabajo de Fin de Máster se ha desarrollado una web que permite la gestión de los deseos de regalo de los usuarios que además pueden ser compartidos con otros usuarios.

La tecnología Angular ha permitido que el desarrollo del frontal haya sido más ordenado que una web sin esta tecnología y el uso de Ionic ha facilitado y unificado la experiencia del usuario.

Para el backend se ha creado la API Rest en Spring que permite acceder a los datos almacenados en PostgreSQL. Adicionalmente se utiliza Firestore para almacenar datos de regalos en una base de datos NoSQL.

Durante el proyecto, se ha creado y utilizado un entorno de trabajo integrado en Github Actions que ha permitido la integración y despliegue continuo de los artefactos en la nube con Fly.io y Firebase, obteniendo una aplicación disponible online de manera gratuita.

La gestión del proyecto a través de una metodología ágil ha permitido adaptar las necesidades del proyecto al tiempo disponible para realizar el desarrollo.

En conclusión, se han aplicado todos los conocimientos adquiridos a lo largo del máster y se han ampliado los mismos para poder resolver los retos que este TFM ha presentado.

Posibles ampliaciones

Entre las posibles ampliaciones están todas las tareas del product backlog no realizadas, además de incorporar nuevas tareas del punto 4 del Inception Deck.

- Preguntas anónimas
- Asignación de mejor amigo
- Inclusión de APIs de terceros para incluir recomendaciones de precios para un producto.
- Tareas técnicas para mitigar las amenazas detectadas por OWASP
- Regalos en conjunto
- Ampliar el sistema de mensajería. Hilos de mensajes con más de dos usuarios.
- Mejora del entorno CI/CD, inclusión de más herramientas en el flujo de despliegue

Adicionalmente, existe la posibilidad de crear un sistema de recomendación de regalos en base a los datos recopilados por la propia app. Para ello sería necesario crear un proyecto de big data que a partir de los datos insertados por los usuarios, se generen diferentes recomendaciones. Por ejemplo, si un usuario tiene varios deseos de regalo (A,B, C) que otros usuarios, es posible que un nuevo usuario con el deseo de regalo A pueda desear también B o C.

Referencias

- [1] John F. Sherry, Jr. <<Gift-Giving-in-Anthropological-Perspective>> [En línea] Disponible: https://www.researchgate.net/profile/John-Sherry/publication/24098336_Gift_Giving_in_Anthropological_Perspective/links/540764690cf2bba34c1f0fc9/Gift-Giving-in-Anthropological-Perspective.pdf [Último acceso: 1 de junio 2023]
- [2] Imagen <<Enorme mano dando regalo de navidad>> de pch.vector en Freepik [En línea] Disponible: https://www.freepik.es/vector-gratis/enorme-mano-dando-regalo-navidad-ilustracion-vector-plano-chica-chica-feliz-pie-cerca-arbol-vacaciones-disfrutando-nieve-celebrando-concepto-sorpresa-banner-diseno-sitio-web-o-pagina-web-inicio_26877096.htm#query=happy%20present&position=3&from_view=search&track=sph [Último acceso: 1 de junio 2023]
- [3] Angular [En línea] Disponible: <https://angular.io/> [Último acceso: 1 de junio de 2023]
- [4] Documentación Ionic [En línea] Disponible: <https://ionicframework.com/docs/> [Último acceso: 1 de junio de 2023]
- [5] Documentación Spring Boot [En línea] Disponible: <https://spring.io/projects/spring-boot> [Último acceso: 1 de junio de 2023]
- [6] Firebase [En línea] Disponible: <https://firebase.google.com/?hl=es-419> [Último acceso: 1 de junio de 2023]
- [7] bezcoder <<Spring Boot + Angular 15 + PostgreSQL example: CRUD App>> [En línea] Disponible: <https://www.bezkoder.com/spring-boot-angular-15-postgresql/> [Último acceso: 1 de junio de 2023]
- [8] Andreas Chandra. <<Setup Free PostgreSQL on Fly.io and Import Database>> [En línea] Disponible: <https://medium.com/data-folks-indonesia/setup-free-postgresql-on-fly-io-and-import-database-3f8f891cbc71> [Último acceso: 1 de junio de 2023]
- [9] Renan Franca, <<Deploy jhipster monolithic (angular + spring boot) at fly.io>> [En línea] Disponible: https://renanfranca.github.io/deploy-jhipster-monolithic-angularjs-and-spring-boot-at-fly.io.html?utm_source=dev.to&utm_medium=social&utm_campaign=promote-blog [Último acceso: 1 de junio de 2023]
- [10] Firebase Firestore [En línea] Disponible: <https://firebase.google.com/docs/firestore?hl=es-419> [Último acceso: 1 de junio de 2023]
- [11] Github [En línea] Disponible: <https://github.com/> [Último acceso: 1 de junio de 2023]

- [12] About Github Projects [En línea] Disponible:
<https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects> [Último acceso: 1 de junio de 2023]
- [13] Github Actions [En línea] Disponible: <https://github.com/features/actions> [Último acceso: 1 de junio de 2023]
- [14] Calculadora de costes de recursos humanos [En línea] Disponible:
<https://factorialhr.es/calculadora-coste-trabajador> [Último acceso: 1 de junio de 2023]
- [15] Aplicación de alquiler de tecnología <<Groover>> [En línea] Disponible:
<https://www.groover.com/es-es> [Último acceso: 1 de junio de 2023]
- [16] Bushra Sharif, Dr. Shoab A. Khan, Muhammad Wasim Bhatti <<Measuring the Impact of Changing Requirements on Software Project Cost: An Empirical Investigation>> [En línea] Disponible:
https://www.researchgate.net/publication/268423183_Measuring_the_Impact_of_Changing_Requirements_on_Software_Project_Cost_An_Empirical_Investigation [Último acceso: 1 de junio de 2023]
- [17] Firebase [En línea] Disponible: <https://firebase.google.com/?hl=es-419> [Último acceso: 1 de junio de 2023]
- [18] <<Cloud Firestore o Realtime Database>> [En línea] Disponible:
<https://firebase.google.com/docs/database/rtdb-vs-firestore> [Último acceso: 1 de junio de 2023]
- [19] Heroku, <<Removal of Heroku Free Product Plans FAQ>> [En línea] Disponible:
<https://help.heroku.com/RSBRUH58/removal-of-heroku-free-product-plans-faq#:~:text=Free%20databases%20that%20belong%20to,resources%20before%20November%2028%2C%202022> [Último acceso: 1 de junio de 2023]
- [20] Infraestructura Global AWS [En línea] Disponible:
https://aws.amazon.com/es/about-aws/global-infrastructure/regions_az/ [Último acceso: 1 de junio de 2023]
- [21] Ubicaciones de Google Cloud [En línea] Disponible:
<https://cloud.google.com/about/locations?hl=es> [Último acceso: 1 de junio de 2023]
- [22] Docs Fly.io [En línea] Disponible: <https://fly.io/docs/> [Último acceso: 1 de junio de 2023]
- [23] J. Bernal, <<TPV Core>> [En línea] Disponible:
<https://github.com/miw-upm/betca-tpv-core> [Último acceso: 1 de junio de 2023]
- [24] Wakatime [En línea] Disponible: <https://wakatime.com/> [Último acceso: 10 de junio de 2023]

[25] OWASP methodology [En línea] Disponible: <https://owasp.org/> [Último acceso: 10 de junio de 2023]

[26] OWASP ZAP [En línea] Disponible: <https://owasp.org/www-project-zap/> [Último acceso: 10 de junio de 2023]