I.E.S LAS SALINAS

PROYECTO FINAL DE GRADO

CURSO 2024-2025

**ROBANK**

CICLO FORMATIVO GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

AUTOR: Tymur Kulivar Shymanskyi

TUTOR: Manuel Antonio Benito Mora

*Índice*

[1. Justificación 5](#_Toc199027283)

[2. Introducción 5](#_Toc199027284)

[3. Objetivos 6](#_Toc199027285)

[3.1. Desarrollo de un Sistema Multiplataforma de Gestión Financiera 6](#_Toc199027286)

[3.2. Implementación de Múltiples Métodos de Autenticación 6](#_Toc199027287)

[3.3. Visualización Interactiva de Datos Financieros 6](#_Toc199027288)

[3.4. Escalabilidad 6](#_Toc199027289)

[4. Plan de empresa 7](#_Toc199027290)

[4.1. ¿Cómo será nuestra empresa? 7](#_Toc199027291)

[4.1.1. Concreción de la idea 7](#_Toc199027292)

[4.1.1.1. Actividad: Servicio de gestión financiera personal 7](#_Toc199027293)

[4.1.1.2. ¿Por qué este negocio? 7](#_Toc199027294)

[4.1.1.3. Nombre de la empresa 8](#_Toc199027295)

[4.1.1.4. Logotipo de la empresa 8](#_Toc199027296)

[4.1.1.5. Misión, visión y valores 9](#_Toc199027297)

[4.2. Análisis del mercado 10](#_Toc199027298)

[4.2.1 Tamaño del mercado 10](#_Toc199027299)

[4.2.2. Estructura del mercado 11](#_Toc199027300)

[4.2.3. Nuestros clientes 11](#_Toc199027301)

[4.2.4. La competencia 11](#_Toc199027302)

[4.2.5 DAFO 12](#_Toc199027303)

[5. Arquitectura del Sistema 13](#_Toc199027304)

[5.1. Visión General de Componentes 13](#_Toc199027305)

[5.2. Flujo de Datos entre Plataformas 13](#_Toc199027306)

[5.3. Seguridad 14](#_Toc199027307)

[5.3.1. Integración con Firebase Authentication 14](#_Toc199027308)

[5.3.2. Protección de Endpoints 14](#_Toc199027309)

[6. Backend 15](#_Toc199027310)

[6.1. Conceptos importantes. 15](#_Toc199027311)

[6.2. Configuración del proyecto 16](#_Toc199027312)

[6.3. Modelo de Datos 17](#_Toc199027313)

[6.4. Repositorios 18](#_Toc199027314)

[6.5. Servicios 18](#_Toc199027315)

[6.6. Controladores 19](#_Toc199027316)

[6.7. Flujo 20](#_Toc199027317)

[7. Android 21](#_Toc199027318)

[7.1. Conceptos importantes 21](#_Toc199027319)

[7.2. Configuración del proyecto 22](#_Toc199027320)

[7.3. Gestión de la autenticación 23](#_Toc199027321)

[7.4. Comunicación con la API 23](#_Toc199027322)

[7.5. Gestión del estado 24](#_Toc199027323)

[7.6. Interfaz de usuario 24](#_Toc199027324)

[7.7. Localización 25](#_Toc199027325)

[8. Frontend 26](#_Toc199027326)

[8.1. Conceptos importantes. 26](#_Toc199027327)

[8.2. Estructura del proyecto 27](#_Toc199027328)

[8.3. Autenticación y seguridad 27](#_Toc199027329)

[8.4. Gestión de preferencias y estado 29](#_Toc199027330)

[8.5. Componentes 30](#_Toc199027331)

[8.5.1. Página de inicio 30](#_Toc199027332)

[8.5.2. Página Error 404 31](#_Toc199027333)

[8.5.3. Barra de navegación y pie de página 31](#_Toc199027334)

[9. Mejoras Futuras 32](#_Toc199027335)

[10. Conclusión 33](#_Toc199027336)

[11. Bibliografía 34](#_Toc199027337)

[12. Anexos 35](#_Toc199027338)

*Índice de figuras*

[Ilustración 1 Logotipo con el nombre de la empresa 8](#_Toc199088583)

[Ilustración 2 Logotipo pequeño 8](#_Toc199088584)

[Ilustración 3 Logotipo Grande 8](#_Toc199088585)

[Ilustración 4 Misión, Visión y valores 9](#_Toc199088586)

[Ilustración 5 Inversiones en fintech 10](#_Toc199088587)

[Ilustración 6 Uso de la banca digital en Europa. 10](#_Toc199088588)

[Ilustración 7 Matriz DAFO 12](#_Toc199088589)

[Ilustración 8 Diagrama simplificado del flujo de datos en Robank 13](#_Toc199088590)

[Ilustración 9 Consola de Firebase Authentication 14](#_Toc199088591)

[Ilustración 10 Configuración de Spring Security 14](#_Toc199088592)

[Ilustración 11 Logotipo PostgreSql 15](#_Toc199088593)

[Ilustración 12 Logotipo Spring 15](#_Toc199088594)

[Ilustración 13 Logotipo Java 15](#_Toc199088595)

[Ilustración 14 build.gradle 16](#_Toc199088596)

[Ilustración 15 application.properties 16](#_Toc199088597)

[Ilustración 16 Dockerfile 16](#_Toc199088598)

[Ilustración 17 Diagrama relacional 17](#_Toc199088599)

[Ilustración 18 Directorio entities 17](#_Toc199088600)

[Ilustración 19 Entidad preferences 17](#_Toc199088601)

[Ilustración 20 Repositorio BillRepository 18](#_Toc199088602)

[Ilustración 21 Directorio repositories 18](#_Toc199088603)

[Ilustración 22 Servicio BillService 18](#_Toc199088604)

[Ilustración 23 Directorio services 18](#_Toc199088605)

[Ilustración 24 Controlador UserController 19](#_Toc199088606)

[Ilustración 25 Flujo de datos en el backend 20](#_Toc199088607)

[Ilustración 26 Logotipo Jetpack Compose 21](#_Toc199088608)

[Ilustración 27 Logotipo Kotlin 21](#_Toc199088609)

[Ilustración 28 build.gradle.kts 22](#_Toc199088610)

[Ilustración 29 local.properties 22](#_Toc199088611)

[Ilustración 30 AndroidMatifest.xml 22](#_Toc199088612)

[Ilustración 31 AccountManager.kt 23](#_Toc199088613)

[Ilustración 32 RobankApiService.kt 23](#_Toc199088614)

[Ilustración 33 BillManager.kt 24](#_Toc199088615)

[Ilustración 34 Colors.kt 24](#_Toc199088616)

[Ilustración 35 Pantalla principal 25](#_Toc199088617)

[Ilustración 36 Pantalla de login 25](#_Toc199088618)

[Ilustración 37 directorio de localización 25](#_Toc199088619)

[Ilustración 38 Logotipo Angular 26](#_Toc199088620)

[Ilustración 39 Logotipo TypeScript 26](#_Toc199088621)

[Ilustración 40 Logotipo HTML 26](#_Toc199088622)

[Ilustración 41 Estructura simplificada del frontend 27](#_Toc199088623)

[Ilustración 42 plantilla de entorno de Angular 27](#_Toc199088624)

[Ilustración 43 Guardia de rutas 28](#_Toc199088625)

[Ilustración 44 Email de verificación de Robank 28](#_Toc199088626)

[Ilustración 45 Local storage 29](#_Toc199088627)

[Ilustración 46 Página se ajustes 29](#_Toc199088628)

[Ilustración 47 Página de inicio 30](#_Toc199088629)

[Ilustración 48 Página de error 404 31](#_Toc199088630)

[Ilustración 49 Barra de navegación en computadora 31](#_Toc199088631)

[Ilustración 50 Barra de navegación en móvil 31](#_Toc199088632)

[Ilustración 51 Pie de página 31](#_Toc199088633)

[Ilustración 52 Mejoras futuras 32](#_Toc199088634)

1. Justificación

Controlar las finanzas personales es complicado. Muchas personas gastan sin saber exactamente en qué, pierden de vista sus objetivos de ahorro o simplemente no tienen una visión clara de su situación económica.

Las apps bancarias tradicionales muestran movimientos, pero no ayudan a entender patrones de gasto. Las herramientas especializadas suelen ser demasiado complejas, caras, o no funcionan bien en diferentes dispositivos.

Decidimos crear Robank para resolver estos problemas: una aplicación sencilla que funcione tanto en el móvil como en el ordenador, donde la gente pueda ver sus gastos de forma clara, crear categorías personalizadas y establecer metas de ahorro realistas.

2. Introducción

Robank es un sistema multiplataforma para gestionar finanzas personales. Permite a los usuarios controlar sus gastos, establecer objetivos financieros y visualizar su progreso económico desde cualquier dispositivo.

El sistema está compuesto por tres componentes principales: un backend que gestiona los datos y la autenticación, una aplicación web, y una aplicación nativa para Android. Todos los componentes trabajan de forma unificada para ofrecer una experiencia coherente independientemente del dispositivo utilizado.

La aplicación nació de la necesidad de tener una herramienta que combine simplicidad de uso con funcionalidades potentes, ofreciendo una alternativa equilibrada entre las aplicaciones básicas y las opciones más complicadas que existen actualmente.

3. Objetivos

3.1. Desarrollo de un Sistema Multiplataforma de Gestión Financiera

Crear un ecosistema completo de gestión financiera que funcione de manera integrada en web y Android, manteniendo una experiencia de usuario consistente y permitiendo la sincronización de datos entre dispositivos.

3.2. Implementación de Múltiples Métodos de Autenticación

Desarrollar un sistema de autenticación seguro y flexible que ofrezca diferentes opciones de acceso (email/contraseña y Google) mientras garantiza la protección de los datos financieros personales.

3.3. Visualización Interactiva de Datos Financieros

Transformar datos numéricos en información visual comprensible mediante gráficos interactivos, permitiendo a los usuarios identificar patrones de gasto y realizar un seguimiento efectivo de sus metas financieras.

3.4. Escalabilidad

Diseñar una arquitectura que soporte el crecimiento futuro tanto en número de usuarios como en funcionalidades. Esto incluye implementar una *base de código*[[1]](#footnote-1) mantenible, una estructura de base de datos optimizada para consultas frecuentes, y una *API REST*[[2]](#footnote-2) que pueda manejar un aumento progresivo de peticiones sin comprometer el rendimiento del sistema.

4. Plan de empresa

En este apartado vamos a hablar sobre la viabilidad empresarial del producto que hemos creado. Hablaremos más en detalle sobre la identidad de nuestra empresa, analizaremos nuestra competencia y explicaremos nuestro modelo de negocio.

4.1. ¿Cómo será nuestra empresa?

Nuestra empresa se centrará en el desarrollo de la aplicación, su gestión y el mantenimiento continuo. Somos un equipo pequeño enfocado en crear una solución sencilla pero potente para la gestión de finanzas personales.

4.1.1. Concreción de la idea

4.1.1.1. Actividad: Servicio de gestión financiera personal

Robank es un servicio multiplataforma que ayuda a controlar gastos, establecer metas de ahorro y visualizar el progreso económico. Ofrecemos herramientas digitales accesibles que no requieren conocimientos financieros previos.

Nos centramos en tres pilares: **categorización de gastos**, v**isualización clara de datos**, y **seguimiento de objetivos personales**. Priorizamos la simplicidad sin perder funcionalidad.

4.1.1.2. ¿Por qué este negocio?

Vimos que mucha gente sigue sin tener una herramienta clara para entender y mejorar su situación financiera. Las apps de los bancos muestran datos, pero no ayudan a interpretarlos. Y las aplicaciones más avanzadas suelen ser confusas o de pago.

Pensamos que hacía falta algo en el medio: fácil de usar, pero útil de verdad. Algo que permita ver en qué se gasta, marcar objetivos y seguirlos sin complicarse. Además, notamos que pocas apps funcionan bien tanto en móvil como en ordenador, algo que para nosotros era fundamental.

Por eso creamos Robank: una solución sencilla para un problema que todos tenemos.

4.1.1.3. Nombre de la empresa

El nombre "Robank" combina "Ro" (derivado de "Robot" sugiriendo automatización e inteligencia) con "bank" (banco, relacionado con finanzas). Esta combinación transmite modernidad tecnológica aplicada al sector financiero, manteniendo familiaridad con el usuario final.

El nombre es corto, fácil de recordar, y funciona bien tanto en español como en inglés, lo que facilita una posible expansión internacional futura.

4.1.1.4. Logotipo de la empresa

Nuestro logotipo muestra a un personaje simpático con barba blanca y sombrero, que refleja cercanía y confianza. Hemos optado por un estilo ilustrado y colores morados que transmiten creatividad y modernidad.

La versión completa incluye un billete, reforzando la idea de finanzas personales, mientras que la versión reducida mantiene la identidad en espacios pequeños. Ambos formatos funcionan bien tanto en entornos digitales como impresos, asegurando coherencia visual en cualquier aplicación.

****

Ilustración Logotipo con el nombre de la empresa

Ilustración Logotipo pequeño

Ilustración Logotipo Grande

4.1.1.5. Misión, visión y valores

**Misión**

Simplificar la gestión financiera personal mediante herramientas tecnológicas intuitivas que empoderan a los usuarios para tomar decisiones económicas informadas y alcanzar sus objetivos financieros, sin importar su experiencia previa.

**Visión**

Convertirnos en la plataforma líder para la gestión de finanzas personales a nivel nacional, expandiendo posteriormente a mercados internacionales y democratizando el acceso a herramientas financieras de calidad.

**Valores**

* **Simplicidad**. Creamos interfaces intuitivas y procesos claros que cualquier persona puede usar sin frustración, porque la tecnología financiera debe ser accesible para todos.
* **Modernidad**. Utilizamos tecnologías actuales y diseños contemporáneos para crear experiencias digitales fluidas y atractivas que se mantienen al día con las expectativas de los usuarios.
* **Seguridad**. Protegemos los datos financieros con los más altos estándares de seguridad digital, dando a los usuarios control total sobre su información.
* **Accesibilidad**. Desarrollamos soluciones que funcionan en diferentes dispositivos y se adaptan a diversas necesidades, promoviendo la inclusión financiera digital.

****

Ilustración Misión, Visión y valores

4.2. Análisis del mercado

4.2.1 Tamaño del mercado

Aunque la inversión en fintech[[3]](#footnote-3) bajó en 2024 hasta los 95 600 millones de dólares (su mínimo en siete años), mostradas en la Ilustración 5, la adopción de pagos digitales siguió creciendo entre los consumidores.

En España las cosas van bien: el 71% de la gente ya usa banca digital, y se espera que llegue al 85% en cinco años, tal como se refleja en la Ilustración 6, pero hay un problema: muy poca gente usa apps específicas para gestionar sus finanzas personales.

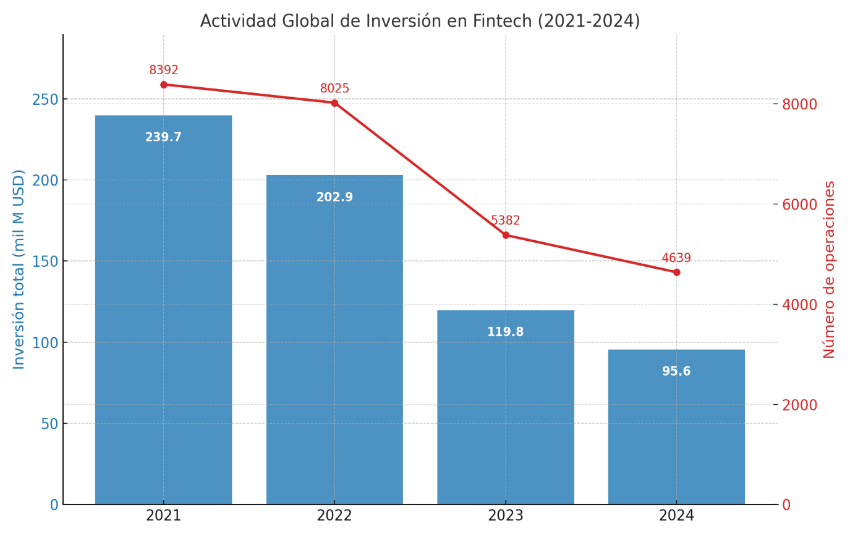


Ilustración Inversiones en fintech

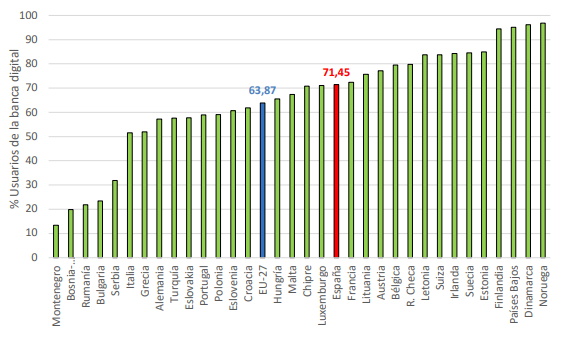


Ilustración Uso de la banca digital en Europa.

4.2.2. Estructura del mercado

Desarrolladores: Startups fintech y empresas tecnológicas que crean aplicaciones de gestión financiera.

Proveedores de servicios financieros: Bancos y neobancos que integran funcionalidades de gestión financiera en sus plataformas.

Intermediarios: Plataformas de agregación de datos financieros que permiten la conexión entre cuentas bancarias y aplicaciones.

Prescriptores: Influencers y medios especializados que recomiendan aplicaciones financieras.

Usuarios: Consumidores interesados en controlar y optimizar sus finanzas personales.

4.2.3. Nuestros clientes

* **Quién**: adultos jóvenes (18-40 años), especialmente millennials y generación Z.
* **Qué buscan**: herramientas para controlar gastos, ahorrar y planificar presupuestos.
* **Cuándo:** uso frecuente, con interacciones diarias.
* **Dónde**: en áreas urbanas con alta conectividad digital.
* **Cuánto compra**: entre 0 y 5 euros mensuales, dependiendo de la suscripción.

4.2.4. La competencia

Aplicaciones de bancos tradicionales. Compañías como BBVA, Santander o CaixaBank ofrecen funciones básicas como categorías de gasto o alertas. Son útiles para consultar movimientos, pero limitadas para una gestión real y solo funcionan con sus propios productos.

Neobancos[[4]](#footnote-4). Plataformas como Revolut, N26 o imagin integran herramientas de control financiero con una buena experiencia de usuario. Sin embargo, no permiten ver cuentas externas, lo que limita su utilidad si el usuario trabaja con varios bancos.

Aplicaciones especializadas. Fintonic, YNAB o Goin ofrecen funciones más completas, como presupuestos, ahorro automático o informes. Aun así, muchas son complejas, de pago o poco accesibles para el usuario medio.

4.2.5 DAFO

Este análisis resume los puntos fuertes y débiles de Robank, así como los factores externos que pueden influir en su evolución.

**Debilidades:**

* Dependencia de servicios externos como Firebase y agregadores bancarios
* Marca nueva, sin trayectoria ni base de usuarios consolidada
* Capacidad limitada para marketing o expansión frente a grandes competidores

**Amenazas**:

* Alta competencia: bancos tradicionales, neobancos y apps especializadas
* Cambios en normativas financieras o requisitos de seguridad
* Posible desinterés o abandono del usuario si no se ofrece valor constante

**Fortalezas**:

* Aplicación clara, intuitiva y fácil de usar desde el primer momento
* Independiente de entidades bancarias, permite agrupar cuentas de varios bancos
* Diseñada como multiplataforma desde el inicio (web y móvil), sin perder funcionalidad

**Oportunidades**:

* Creciente uso de soluciones fintech, especialmente entre jóvenes
* Falta de herramientas accesibles para quienes no tienen conocimientos financieros
* Posibilidad de escalar con nuevas funciones (inversión, notificaciones, IA, etc.).

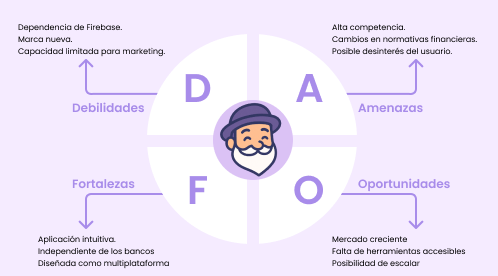


Ilustración Matriz DAFO

5. Arquitectura del Sistema

5.1. Visión General de Componentes

Hemos diseñado el sistema Robank como tres proyectos interconectados:

* **Robank Backend**[[5]](#footnote-5)funciona como el núcleo del sistema, gestionando todos los datos y la lógica de negocio[[6]](#footnote-6) a través de una API REST.
* **Robank Android** es nuestra aplicación móvil nativa que proporciona acceso completo a las funcionalidades del sistema desde dispositivos móviles.
* **Robank Frontend**[[7]](#footnote-7) es nuestra aplicación web que permite acceder a todas las funcionalidades desde cualquier navegador, complementando la experiencia móvil.

5.2. Flujo de Datos entre Plataformas

Las aplicaciones cliente obtienen automáticamente el token de autenticación de Firebase y envían solicitudes HTTP al backend. El servidor verifica la identidad del usuario con Firebase, procesa la solicitud en PostgreSQL y devuelve una respuesta que actualiza la interfaz del cliente.

Este enfoque centralizado permite acceder a los datos desde cualquier dispositivo manteniendo la información sincronizada.

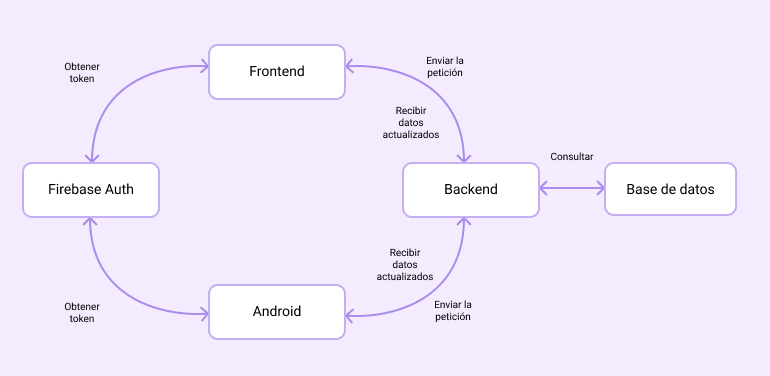


Ilustración Diagrama simplificado del flujo de datos en Robank

5.3. Seguridad

5.3.1. Integración con Firebase Authentication

Decidimos usar Firebase Authentication para gestionar las identidades de forma centralizada. Así, los usuarios pueden registrarse e iniciar sesión con su correo y contraseña, o directamente con su cuenta de Google, tanto en la app de Android como en la versión web. Además, este enfoque mejora la seguridad, ya que se apoya en sistemas mucho más sólidos y evita que el propietario del servicio tenga acceso a las credenciales privadas de los usuarios.

5.3.2. Protección de Endpoints

Implementamos un sistema de verificación en el backend donde cada solicitud a nuestra API debe incluir el token Firebase. El servidor verifica este token y, si es válido, permite el acceso a los recursos del usuario autenticado. En caso contrario, rechaza la solicitud.

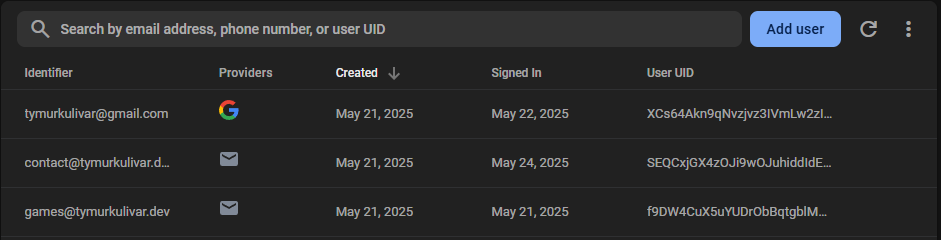


Ilustración Consola de Firebase Authentication

Solo se permiten solicitudes anónimas a ciertos endpoints públicos definidos explícitamente, el resto requiere autenticación. Garantizamos que cada usuario solo pueda acceder a su información personal mediante el aislamiento de datos en todos los endpoints[[8]](#footnote-8) de la API.

Este enfoque multicapa nos permite proteger la información financiera personal de nuestros usuarios independientemente de la plataforma que utilicen para acceder al sistema.

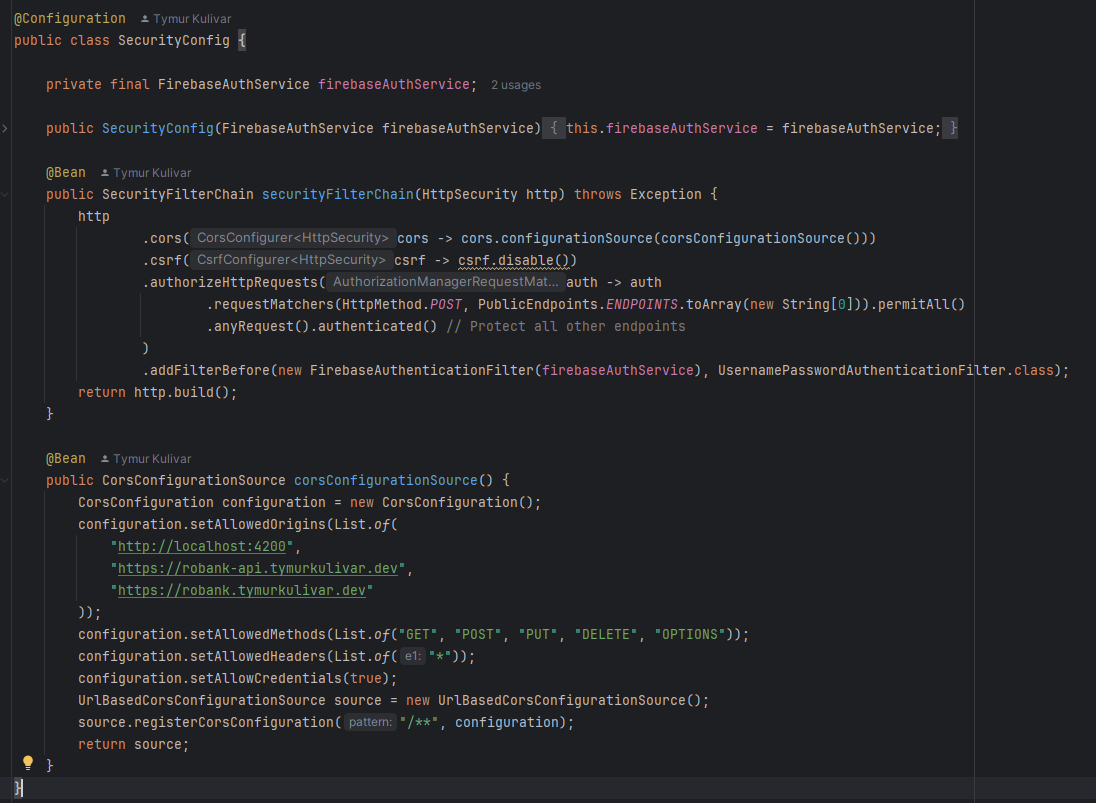


Ilustración Configuración de Spring Security

6. Backend

En este apartado hablaremos sobre la aplicación núcleo del proyecto, el programa que permite interconectar y procesar todo el flujo de información, el backend.

6.1. Conceptos importantes.

**Java**: es un lenguaje de programación poderoso y fácil de aprender, orientado a objetos, que permite desarrollar aplicaciones que pueden correr en distintas plataformas sin necesidad de modificar el código gracias a la *Java Virtual Machine* (JVM). Algunas de sus características más destacadas son su **portabilidad, seguridad, rendimiento** y su enfoque en la **programación orientada a objetos**, que lo convierten en una opción ideal para aplicaciones complejas.

**Spring**: Es un marco de trabajo (*framework*) de desarrollo para aplicaciones Java que facilita la creación de aplicaciones empresariales, proporcionando herramientas para la gestión de dependencias, la configuración y la integración con otras tecnologías.

**Spring Boot**: Es una extensión de Spring que simplifica la configuración y el desarrollo de aplicaciones Java al proporcionar una estructura predeterminada y herramientas que permiten crear aplicaciones independientes y listas para producción con mínima configuración.

**Gradle**: Es una herramienta de automatización de construcción de proyectos que permite gestionar proyectos de software, optimizando la compilación, las dependencias y las tareas de despliegue, y es especialmente popular en proyectos Java, Kotlin y Android.

**RESTful API**: Es una arquitectura para diseñar servicios web que siguen principios REST (*Representational State Transfer*), utilizando métodos HTTP estándar como GET, POST, PUT y DELETE, y permitiendo la interacción entre sistemas de forma simple y escalable.

**PostgreSQL**: Es un SGBD (sistema de gestión de bases de datos) relacional, conocido por su robustez y fiabilidad. Ofrece soporte para transacciones complejas, extensibilidad y una amplia variedad de tipos de datos.

**Lombok**: Es una biblioteca para Java que simplifica la escritura de código al generar automáticamente métodos comunes como *getters, setters, constructors*, y otros, mediante anotaciones. Esto reduce la cantidad de código repetitivo y mejora la legibilidad y el mantenimiento del código.

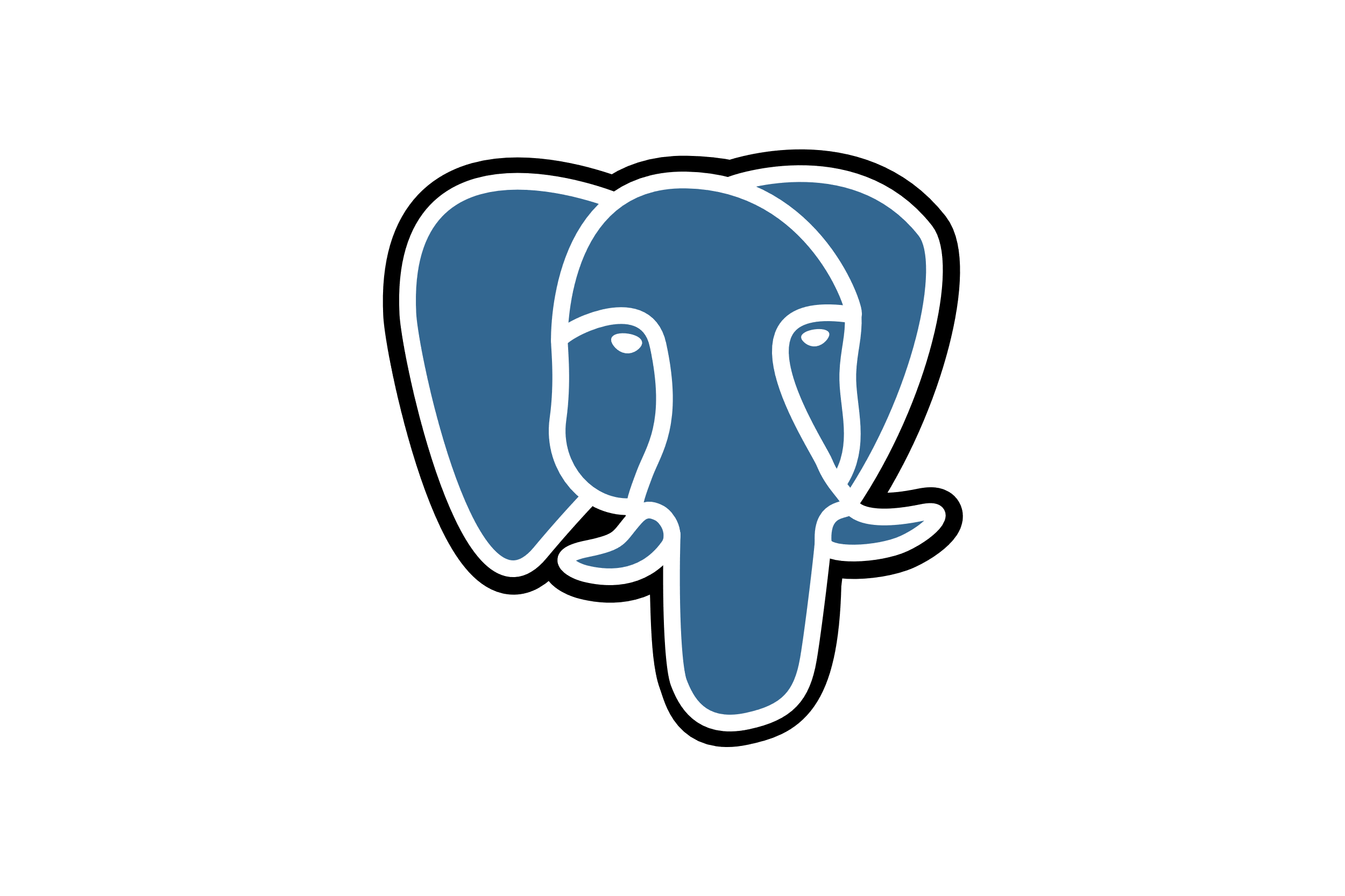
 

Ilustración Logotipo PostgreSql

Ilustración Logotipo Spring

Ilustración Logotipo Java

6.2. Configuración del proyecto

Al configurar una aplicación Spring Boot con Gradle, es fundamental comprender el propósito de los siguientes archivos:

**build.gradle:** define la configuración del proyecto, incluyendo repositorios[[9]](#footnote-9) y dependencias[[10]](#footnote-10).



Ilustración build.gradle

**application.properties**: contiene propiedades clave de la aplicación, como credenciales de base de datos, contantes personales, entre otras constantes de Spring.

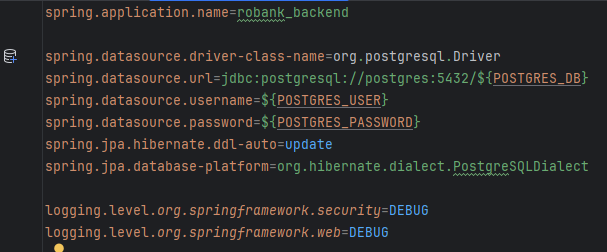


Ilustración application.properties

**Dockerfile** y **compose.yaml**: son archivos esenciales para el despliegue. El primero define cómo se construye la imagen de la aplicación, mientras que el segundo orquesta servicios como la aplicación y la base de datos.

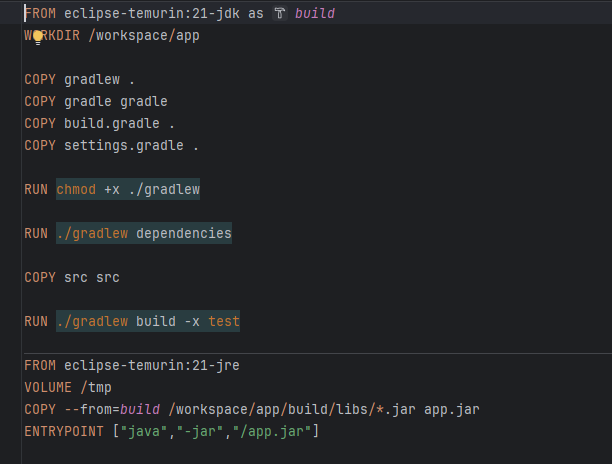


Ilustración Dockerfile

6.3. Modelo de Datos

Robank cuenta con la siguiente estructura de base de datos:

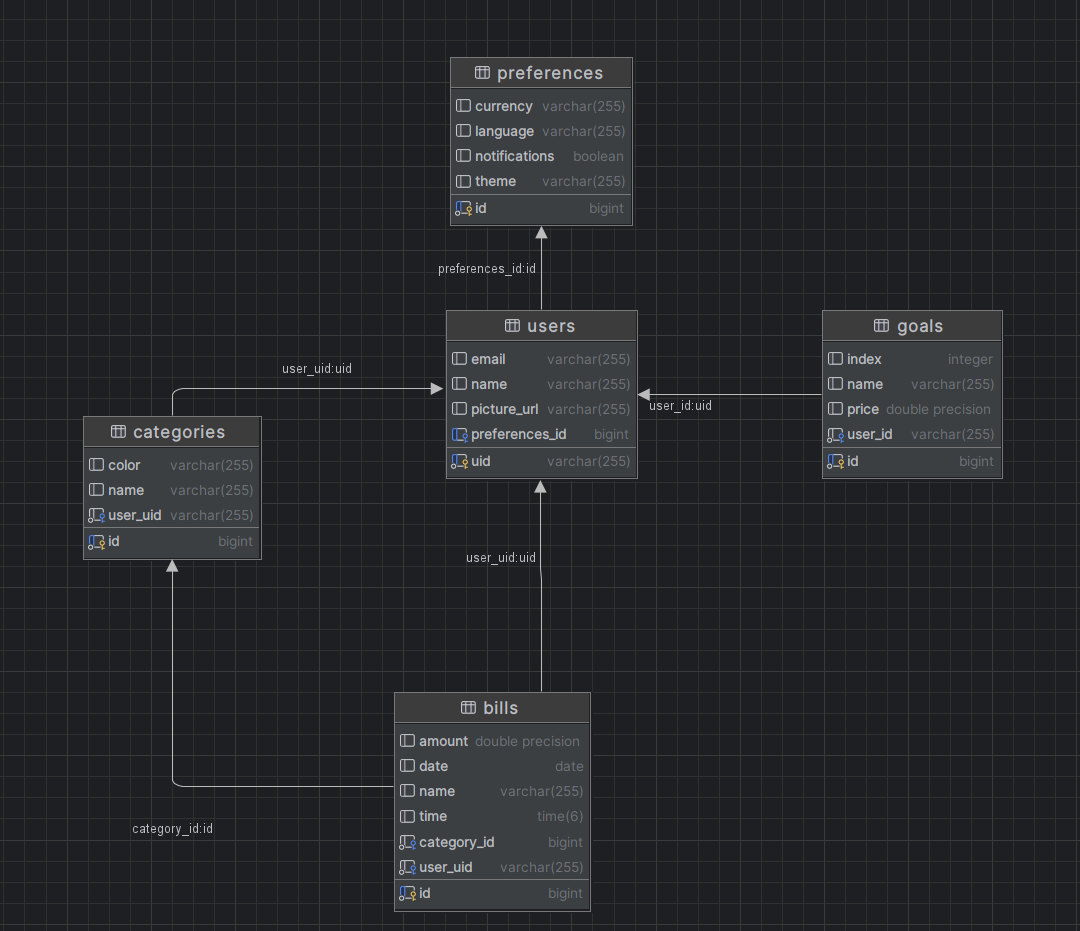
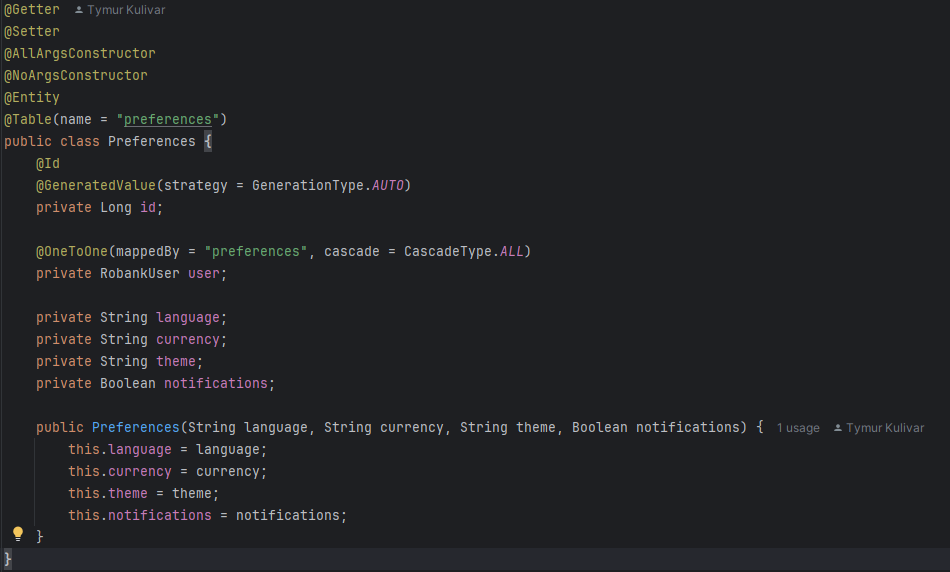


Ilustración Diagrama relacional

Podemos ver que la tabla principal es *users*, la cual se relaciona con el resto de las tablas: *goals, preferences, categories y bills*. Esta configuración representa la solución más adecuada que encontramos para cubrir las necesidades de la aplicación, buscando un equilibrio entre eficiencia en las consultas y complejidad en la implementación.

En el código, estas tablas están representadas en el directorio *entities*, donde se definen las clases Java marcadas con la anotación @Entity. Esto permite a Spring reconocerlas como entidades de base de datos. Además, se utilizan anotaciones como @OneToOne y @OneToMany para definir las relaciones entre tablas de forma clara y precisa.



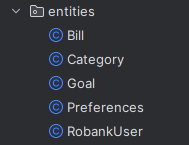
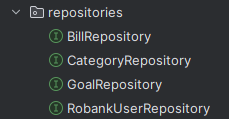


Ilustración Directorio entities

Ilustración Entidad preferences

6.4. Repositorios

Los repositorios en Spring son interfaces que gestionan el acceso a la base de datos. Al extender **JpaRepository**, permiten realizar operaciones como guardar, buscar, actualizar y eliminar entidades sin necesidad de escribir SQL. Además, se pueden definir métodos personalizados siguiendo una convención de nombres, y Spring se encarga de generar su implementación automáticamente. Esto simplifica considerablemente el trabajo con la base de datos y mejora la legibilidad del código.

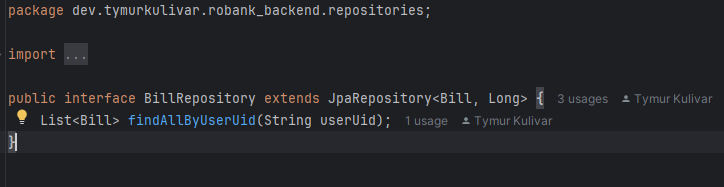


Ilustración Repositorio BillRepository

Ilustración Directorio repositories

6.5. Servicios

Los servicios en Spring encapsulan la lógica de negocio de la aplicación. Son clases que se comunican con los repositorios para obtener o modificar datos, y exponen métodos que luego utilizan los controladores. Esta separación permite mantener el código organizado, facilita las pruebas unitarias y promueve la reutilización de la lógica. En Spring Boot los servicios tienen que tener una anotación @Service. En Robank, los servicios gestionan operaciones como la creación de usuarios, asignación de categorías o procesamiento de objetivos financieros.

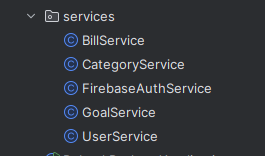


Ilustración Servicio BillService

Ilustración Directorio services

6.6. Controladores

Los controladores en Spring Boot manejan las solicitudes HTTP que llegan a la aplicación. Son clases anotadas con @RestController, lo que indica que sus métodos devolverán directamente datos (normalmente en formato **JSON**[[11]](#footnote-11)) en lugar de vistas HTML.

Dentro de los controladores se definen los endpoints utilizando anotaciones como @GetMapping, @PostMapping, @PutMapping, etc., que indican el tipo de petición y la ruta asociada. Los métodos pueden recibir parámetros desde la URL con @PathVariable, datos de consulta con @RequestParam o cuerpos completos en formato JSON con @RequestBody. Esta flexibilidad permite adaptar fácilmente cada *endpoint* a las necesidades de la aplicación.

En Robank, los controladores actúan como punto de entrada a las funciones principales de la aplicación, como la gestión de usuarios, categorías o facturas, canalizando las peticiones del cliente hacia los servicios que contienen la lógica.

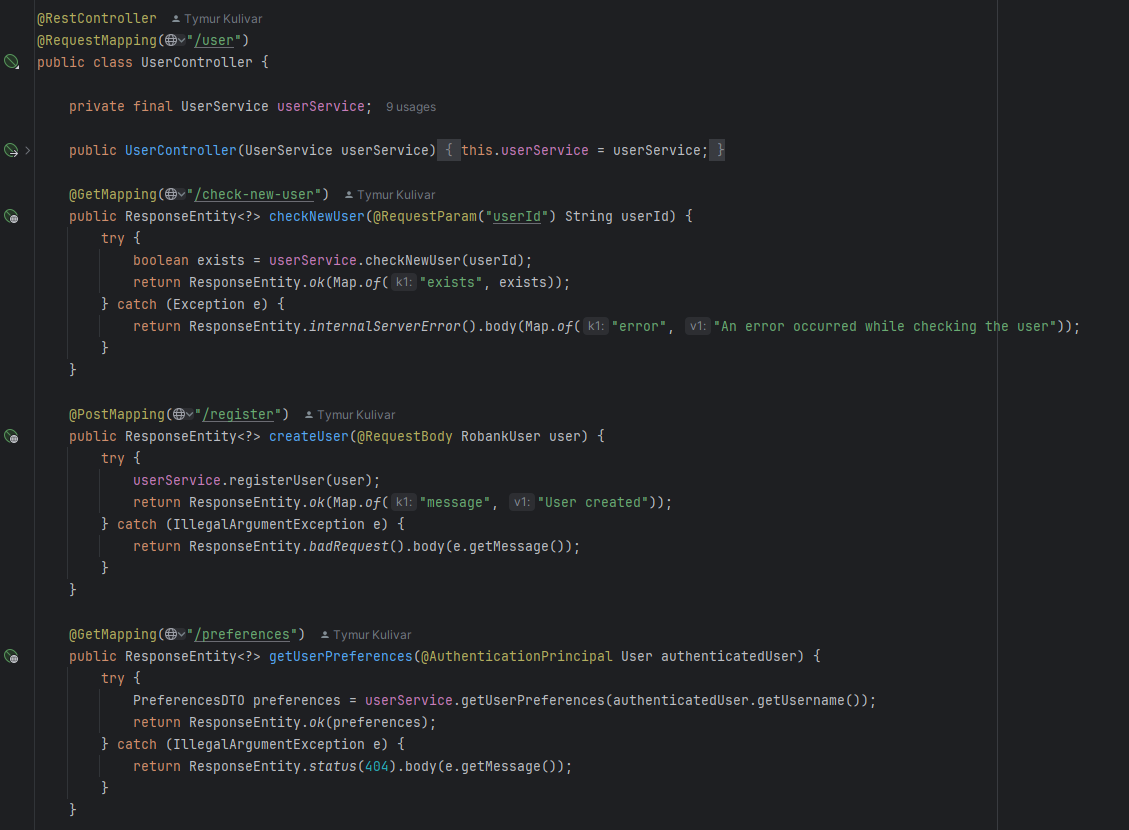


Ilustración Controlador UserController

6.7. Flujo

**El cliente** envía una solicitud HTTP a un endpoint expuesto por un **controlador**. Antes de llegar, **Spring Security** intercepta la petición y valida el token de autenticación generado por **Firebase**. Si el token es válido, el controlador procesa la solicitud y delega la lógica al **servicio** correspondiente.

El servicio se encarga de aplicar las reglas de negocio y se comunica con los **repositorios** para acceder a la **base de datos** mediante JPA. Finalmente, la respuesta (en formato **JSON**) vuelve al cliente, completando el ciclo.

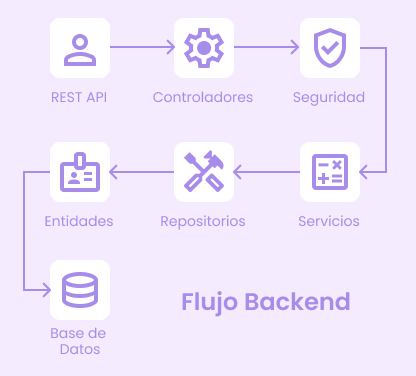


Ilustración Flujo de datos en el backend

7. Android

En este apartado hablaremos sobre la aplicación móvil nativa de Android del proyecto Robank, desarrollada para ofrecer a los usuarios una experiencia fluida y moderna en sus dispositivos móviles.

7.1. Conceptos importantes

**Kotlin**. Es un lenguaje de programación moderno y conciso, completamente compatible con Java. Se ha convertido en el lenguaje oficial para el desarrollo Android, ofreciendo características como la seguridad de nulos[[12]](#footnote-12), que permiten escribir código más limpio, seguro y eficiente.

**Jetpack Compose**. Es una herramienta moderna de Android para crear interfaces de usuario. Utiliza un enfoque declarativo[[13]](#footnote-13), lo que permite describir la interfaz directamente desde el código, eliminando la necesidad de crear y gestionar las vistas XML.

**Credential Manager**. Es la solución actual de Android para la gestión de credenciales de usuario. Soporta autenticación de diferentes tipos, proporcionando una interfaz unificada y segura para la autenticación en aplicaciones Android.

**Retrofit**. Es una biblioteca que simplifica la comunicación con API REST. Permite definir interfaces en Kotlin que representan las llamadas HTTP, y convierte automáticamente las respuestas JSON en objetos de datos mediante conversores como **Gson**. Facilita el consumo de servicios web de manera clara y mantenible.

**Compose Charts**. Es una librería de gráficos para Jetpack Compose que permite crear gráficos de barras, usadas en la vista de facturas y de categorías. Compose Charts permiten personalizar y animar gráficos de forma intuitiva, sin necesidad de utilizar vistas tradicionales o bibliotecas externas complejas.



Ilustración Logotipo Jetpack Compose

Ilustración Logotipo Kotlin

7.2. Configuración del proyecto

La aplicación Android utiliza Gradle como sistema de construcción, con la siguiente estructura de configuración:

**build.gradle.kts**: Define las dependencias del proyecto, incluyendo Jetpack Compose, Firebase, Retrofit y otras bibliotecas necesarias para el funcionamiento de la aplicación.



Ilustración build.gradle.kts

**local.propereties**: Archivo de configuración local del proyecto. En el caso de Robank contiene las claves y configuraciones necesarias para la integración con los servicios de Firebase.

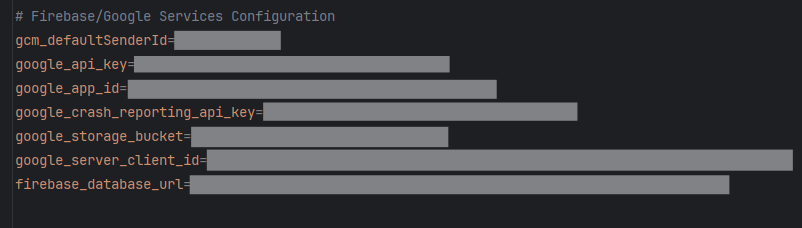


Ilustración local.properties

**AndroidManifest.xml**: Define los permisos de la aplicación y componentes principales, incluyendo permisos de internet necesarios para la comunicación con el backend.

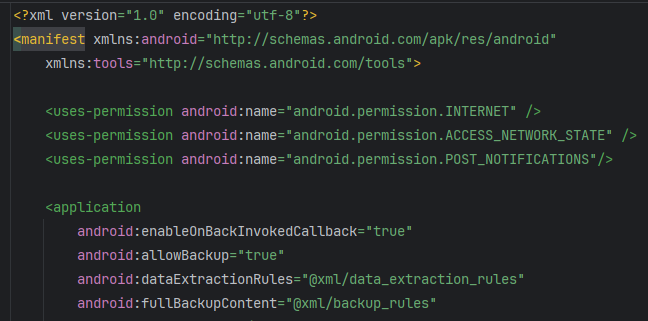


Ilustración AndroidMatifest.xml

7.3. Gestión de la autenticación

La clase **AccountManager** centraliza toda la lógica de autenticación de usuarios, integrando Firebase Authentication con el Credential Manager de Android:

* Registro de usuarios y inicio de sesión: Soporta múltiples métodos de autenticación incluyendo credenciales guardadas, Google Sign-In y autenticación tradicional por email/contraseña.
* Gestión de perfiles: Permite la carga y actualización de imágenes de perfil desde archivos locales o URLs de proveedores externos como Google.



Ilustración AccountManager.kt

7.4. Comunicación con la API

La configuración de red se gestiona a través de RetrofitClient, que implementa:

* **Interceptor** de autenticación, que añade automáticamente el token de Firebase a todas las peticiones HTTP, excepto para el endpoint de registro.
* **RobankApiService**, que define todos los endpoints disponibles mediante anotaciones de Retrofit, incluyendo operaciones CRUD para usuarios, objetivos, categorías y facturas.

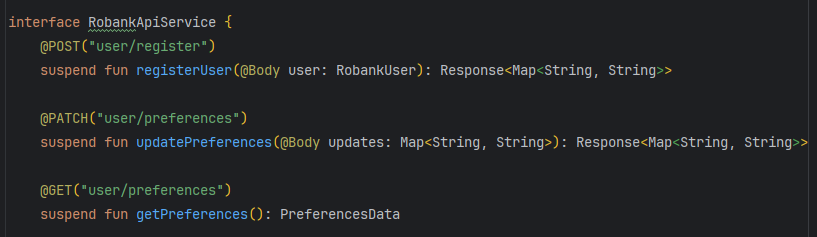


Ilustración RobankApiService.kt

7.5. Gestión del estado

La aplicación utiliza múltiples managers como **BillManager**, **CategoryManager** y **GoalManager** para gestionar el estado de los datos. Utilizan **StateFlow** para proporcionar actualizaciones reactivas a la interfaz de usuario. Implementan operaciones de carga y creación de facturas con manejo de errores robusto. Estos managers hacen peticiones asíncronas al backend/firebase y mantienen el estado actualizado automáticamente.

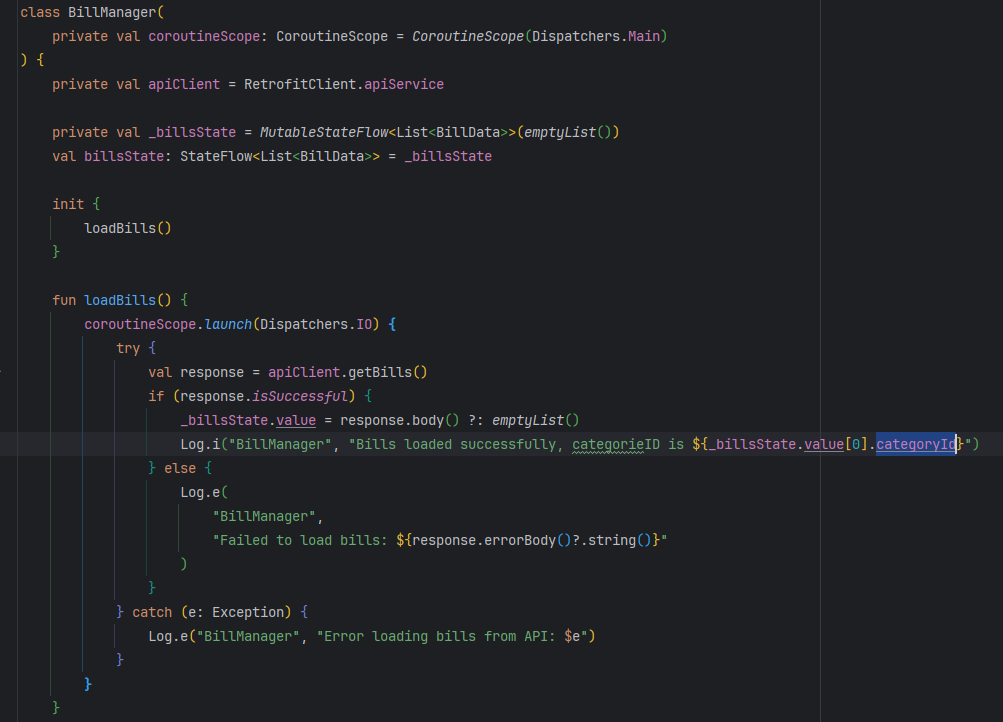


Ilustración BillManager.kt

7.6. Interfaz de usuario

La interfaz está construida completamente con Jetpack Compose, siguiendo los principios de Material Design 3[[14]](#footnote-14):

**Componentes reutilizables** como **RobankTextField** que proporciona un estilo consistente para todos los campos de entrada.

**Navegación**: utiliza Navigation Compose para gestionar la navegación entre pantallas.

**Temas**: soporta temas claro y oscuro definidos en los archivos Themes.kt y Colors.kt.

**Visualización de datos**: la pantalla de facturas incluye gráficos interactivos que muestran los gastos semanales y un listado detallado de todas las transacciones.



Ilustración Colors.kt

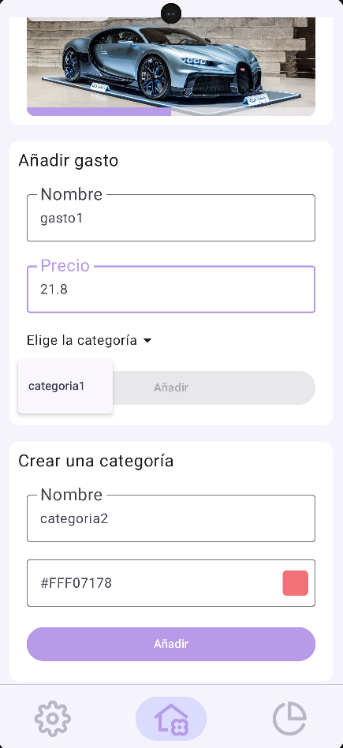


Ilustración Pantalla principal

Ilustración Pantalla de login

7.7. Localización

La aplicación soporta múltiples idiomas (español e inglés) con recursos organizados en archivos correspondientes. Esta estructura facilita añadir nuevos idiomas en el futuro sin complicar la organización del código.

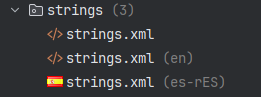


Ilustración Directorio de localización

8. Frontend

En este apartado hablaremos sobre la aplicación frontend de Robank, desarrollada para proporcionar una interfaz de usuario intuitiva y moderna que permita visualizar las finanzas personales en cualquier dispositivo con la conexión a internet.

8.1. Conceptos importantes.

**Angular**: es un framework de desarrollo web creado por Google que permite construir aplicaciones modernas de una sola página (SPA).[[15]](#footnote-15) Utiliza una arquitectura basada en componentes y servicios con inyección de dependencias[[16]](#footnote-16), proporcionando una estructura robusta para proyectos complejos. Incluye herramientas integradas como enrutamiento, formularios reactivos y cliente HTTP.

**TypeScript**: es una extensión de JavaScript añadiendo el tipado estático al lenguaje. Esto ayuda a detectar errores desde etapas tempranas y facilita el mantenimiento. Al compilarse a JavaScript estándar, asegura la compatibilidad con todos los navegadores.

**HTML**: es el lenguaje básico que da forma y contenido a las páginas web. A través de sus etiquetas, permite organizar la información de forma clara y comprensible, tanto para los usuarios como para los navegadores.

**CSS**: es el lenguaje estándar utilizado para definir el estilo visual en páginas web. Permite separar completamente el contenido de su apariencia, personalizando los colores, tipografías y layouts[[17]](#footnote-17). Las características modernas como **Flexbox** y **CSS Grid** facilitan la creación de interfaces responsivas[[18]](#footnote-18).

**Chart.js**: es una biblioteca JavaScript ligera para crear gráficos interactivos y visualmente atractivos. Soporta diversos tipos de gráficos como líneas, barras y dona, con capacidades de animación y personalización.

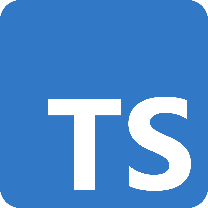


Ilustración Logotipo Angular

Ilustración Logotipo TypeScript

Ilustración Logotipo HTML

8.2. Estructura del proyecto

Nuestro frontend está organizado siguiendo una estructura modular que favorece la mantenibilidad y escalabilidad del proyecto. Dividimos la aplicación en cuatro áreas principales: **public, core, features y shared.**

El directorio *public* contiene recursos estáticos como imágenes. La carpeta *core* incluye servicios globales, interceptores, entornos y guardias de ruta[[19]](#footnote-19) que afectan al funcionamiento general de la aplicación. El directorio *features* agrupa todas las pestañas de la aplicación. Finalmente, *shared* contiene los componentes reutilizables en múltiples partes de la aplicación.

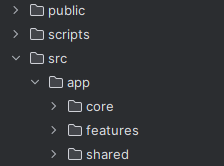


Ilustración Estructura simplificada del frontend

Esta organización permite mantener un código limpio y estructurado, reduciendo el tiempo de desarrollo y mejorando la consistencia visual.

8.3. Autenticación y seguridad

Al igual que en la aplicación para Android, utilizamos Firebase Authentication. Sin embargo, la implementación varía debido a las particularidades del lenguaje. En este caso, desarrollamos un interceptor para las peticiones HTTP, junto con un servicio específico para cada controlador del backend. A diferencia de Android, no empleamos *local.properties*, sino que seguimos el enfoque recomendado por Angular mediante archivos de entorno, donde definimos la dirección del backend y las credenciales necesarias para su acceso.

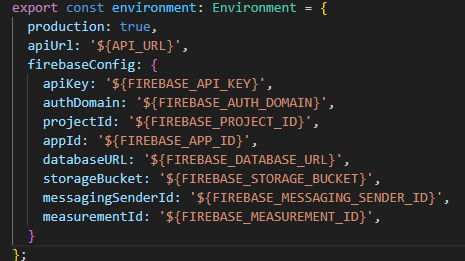


Ilustración plantilla de entorno de Angular

Para proteger las rutas que requieren autenticación, implementamos guardias de ruta que verifican el estado de autenticación antes de permitir el acceso. Esto garantiza que sólo usuarios autenticados puedan acceder a áreas sensibles financiero o la configuración de la cuenta.



Ilustración Guardia de rutas

Además, incorporamos un sistema de verificación por correo electrónico para confirmar la identidad de los usuarios, lo que mejora la seguridad de las cuentas y ayuda a prevenir suplantaciones de identidad.

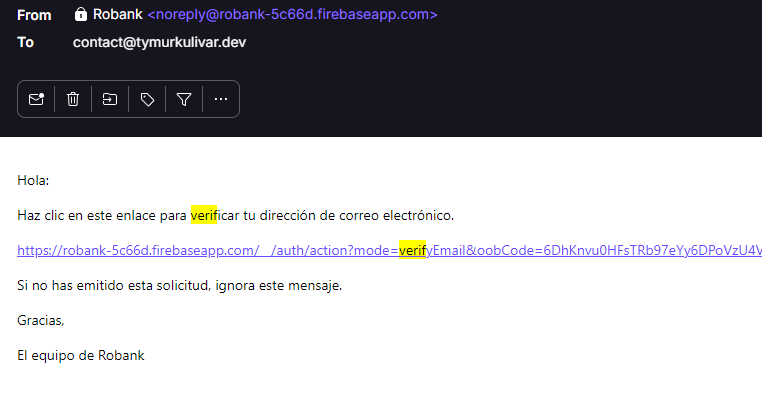


Ilustración Email de verificación de Robank

8.4. Gestión de preferencias y estado

Para la gestión de preferencias de usuario, utilizamos una estrategia híbrida. Almacenamos configuraciones como el tema visual (claro/oscuro) y preferencias de idioma en *localStorage*[[20]](#footnote-20) para acceso rápido y funcionamiento offline, pero también las sincronizamos con nuestra base de datos cuando el usuario está conectado.

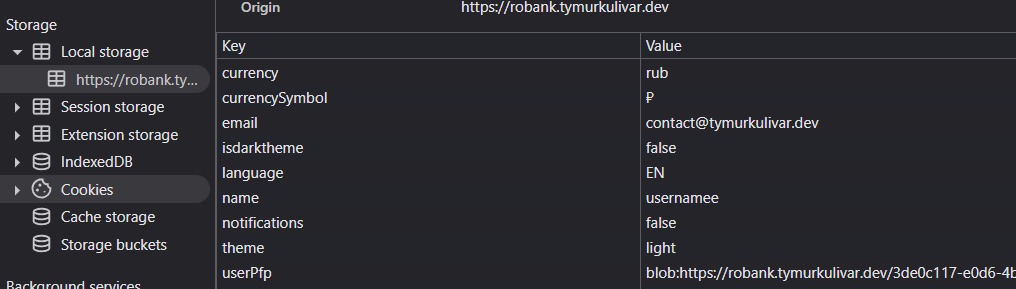


Ilustración Local storage

El sistema detecta automáticamente las preferencias del navegador al primer uso, adaptándose al tema del sistema operativo del usuario y ofreciendo una experiencia personalizada desde el inicio. Esta detección se complementa con controles manuales que permiten al usuario cambiar estas configuraciones según sus preferencias.

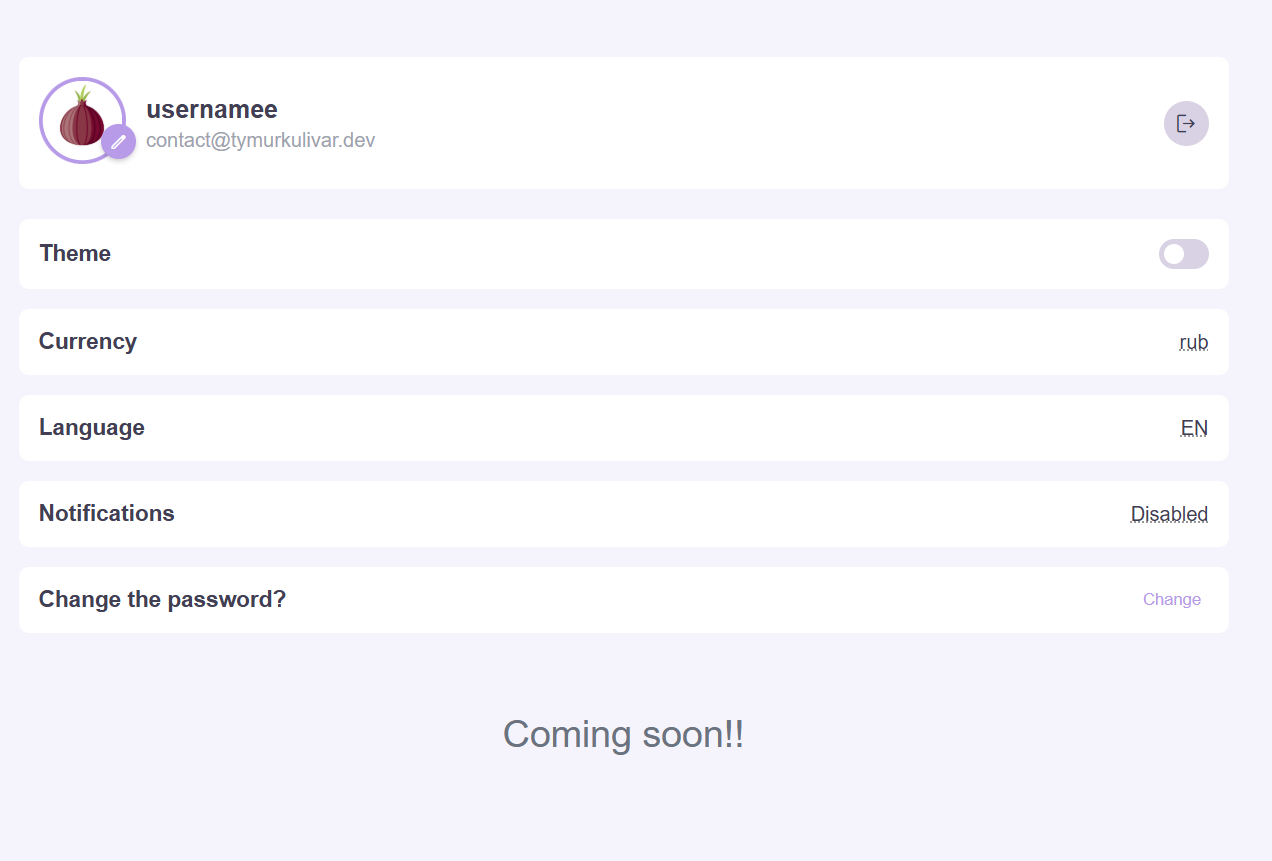


Ilustración Página se ajustes

8.5. Componentes

Nuestra aplicación web mantiene una estructura parecida a la de Android, pero adaptamos algunos componentes para que funcionen mejor en navegadores web.

8.5.1. Página de inicio

Creamos una página de bienvenida para atraer nuevos usuarios, donde explicamos qué ofrece Robank y facilitamos el registro. Como es lo primero que ven las personas, pusimos especial atención en que el diseño sea claro y atractivo.

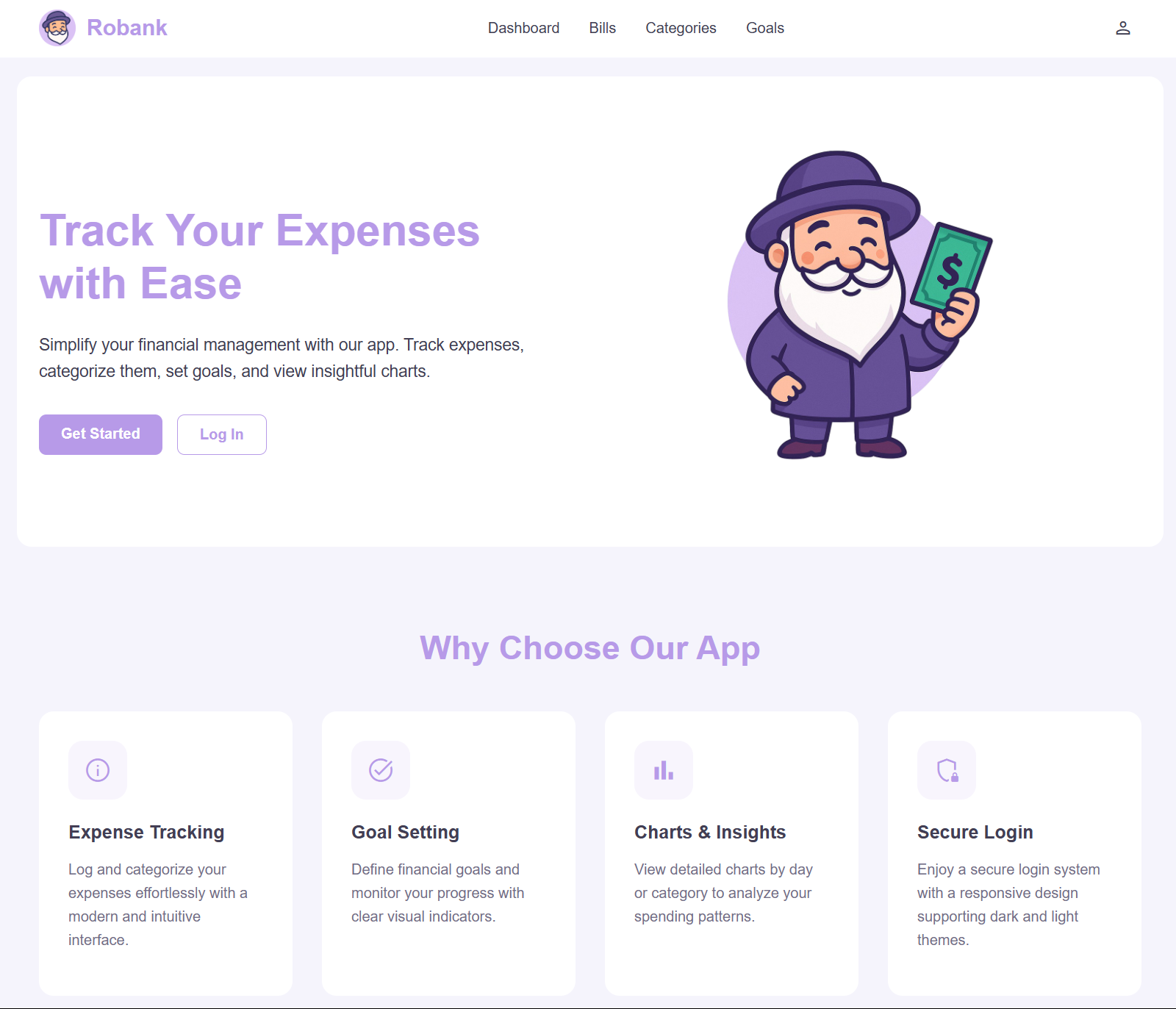


Ilustración Página de inicio

8.5.2. Página Error 404

Añadimos una página de error personalizada para cuando alguien llega a un enlace que no existe. En lugar de mostrar un mensaje técnico confuso, ofrecemos opciones para que el usuario pueda volver a las secciones principales de la aplicación sin perderse.



Ilustración Página de error 404

8.5.3. Barra de navegación y pie de página

Estos elementos aparecen en todas las páginas para que la navegación sea consistente. La barra superior da acceso rápido a las páginas más importantes, mientras que el pie de página incluye información legal y formas de contacto. Ambos se ajustan automáticamente al tamaño de la pantalla, ya sea en computadora o en móvil.



Ilustración Barra de navegación en computadora

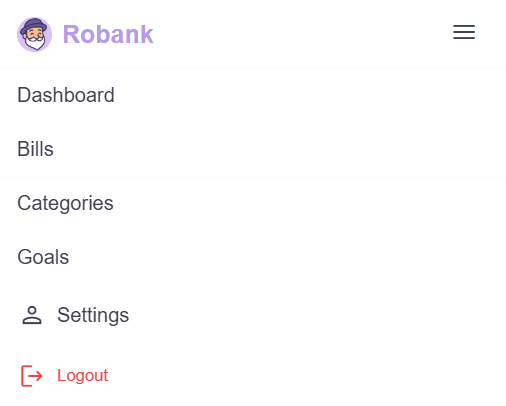


Ilustración Barra de navegación en móvil

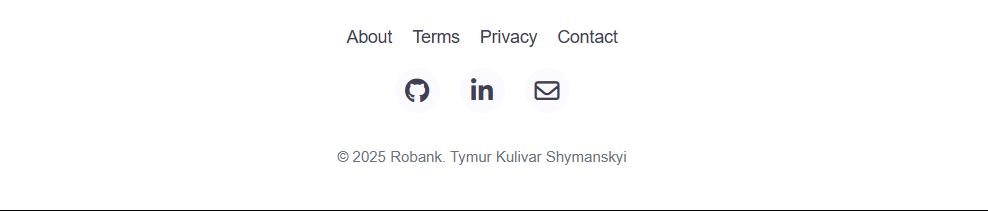


Ilustración Pie de página

9. Mejoras Futuras

Aunque el proyecto ya tiene una complejidad considerable, queremos seguir mejorando el producto. Entre las principales mejoras que planeamos implementar están:

**Completar funcionalidades del frontend.** Queremos actualizar la aplicación web para incluir todas las características disponibles en Android, como cambio de idioma, selección de monedas, y edición/eliminación de contenido desde el navegador.

**Optimización del rendimiento.** Planeamos mejorar los tiempos de carga optimizando las consultas a la base de datos e implementando un sistema de caché para hacer las operaciones más rápidas.

**Mejora del sistema de despliegue.** Actualmente usamos un equipo personal como servidor. Queremos investigar opciones de alojamiento en la nube que sean eficientes y económicamente viables para garantizar mayor estabilidad.

**Seguridad avanzada**. Nos gustaría reforzar las medidas de seguridad con autenticación de dos factores y mejor cifrado de datos, especialmente importante al manejar información financiera.

**Experiencia de usuario.** Planeamos realizar pruebas con usuarios reales para mejorar la interfaz y añadir funcionalidades como notificaciones y un sistema de ayuda más intuitivo.



Ilustración Mejoras futuras

10. Conclusión

Al empezar con Robank, queríamos crear una herramienta que realmente sirviera para gestionar las finanzas personales de forma sencilla. Creemos que hemos conseguido una buena base que funciona bien tanto en el móvil como en el ordenador, manteniendo la facilidad de uso sin perder funcionalidad.

Durante el desarrollo hemos aprendido mucho sobre cómo equilibrar la tecnología con las necesidades reales de los usuarios. Aunque sabemos que hay cosas por mejorar, especialmente en la versión web y algunos aspectos de seguridad, pensamos que hemos creado un sistema sólido y preparado para crecer.

En resumen, Robank representa un avance hacia una gestión financiera más sencilla y al alcance de todos, sin necesidad de tener conocimientos técnicos previos.

11. Bibliografía

Chad Darby. (mayo de 2025). *[NEW] Spring Boot 3, Spring 6 & Hibernate for Beginners*. En Udemy. <https://www.udemy.com/course/spring-hibernate-tutorial/>.

Spring. (s.f.). *Build systems*. En *Spring Boot Documentation*. <https://docs.spring.io/spring-boot/reference/using/build-systems.html#using.build-systems.gradle>.

Spring. (s.f.). *Security*. En *Spring Boot Documentation*. <https://docs.spring.io/spring-boot/how-to/security.html>.

Raghuvardhan Karanam. (21 de octubre de 2023). Route Guards in Angular. En Medium. <https://raghuvardhankaranam.medium.com/route-guards-in-angular-c2c01fe6167b>.

Angular. (s.f.). *Common routing tasks*. En Angular Developer Guide. <https://angular.dev/guide/routing/common-router-tasks>.

Andrés Guzmán. (abril de 2025). *Angular & Spring Boot: Creando Webapp Full Stack.* En Udemy. <https://www.udemy.com/course/angular-spring/>.

Android. (s.f.). *Android Basics with Compose*. En Android Developers. <https://developer.android.com/courses/android-basics-compose/course>.

Arístides Guimerá Orozco. (2 de marzo de 2023). *CURSO: ANDROID desde CERO con KOTLIN*. En YouTube. <https://youtu.be/vJapzH_46a8?si=xgMFylD_C9ro6dMx>.

Philipp Lackner. (12 de julio de 2021). *Jetpack Compose Navigation for Beginners.* En YouTube. <https://youtu.be/4gUeyNkGE3g?si=FrIY4fpZi0VPvLD7>.

Philipp Lackner*. (5 de abril de 2023). Firebase Google Sign-In With Jetpack Compose.* En YouTube. <https://youtu.be/zCIfBbm06QM?si=l2zUDUwwjaMHiLZF>.

KPMG. (febrero de 2025). *Pulse of Fintech H2 2024*. <https://kpmg.com/es/es/informes-publicaciones/2025/02/pulse-fintech-2024-segundo-semestre.html>.

Funcas. (septiembre de 2024). *La adopción de la nueva tecnología bancaria en España y Europa*. <https://www.funcas.es/wp-content/uploads/2024/09/ODF-Funcas-48-2024.pdf>

12. Anexos

**Repositorios de Github:**

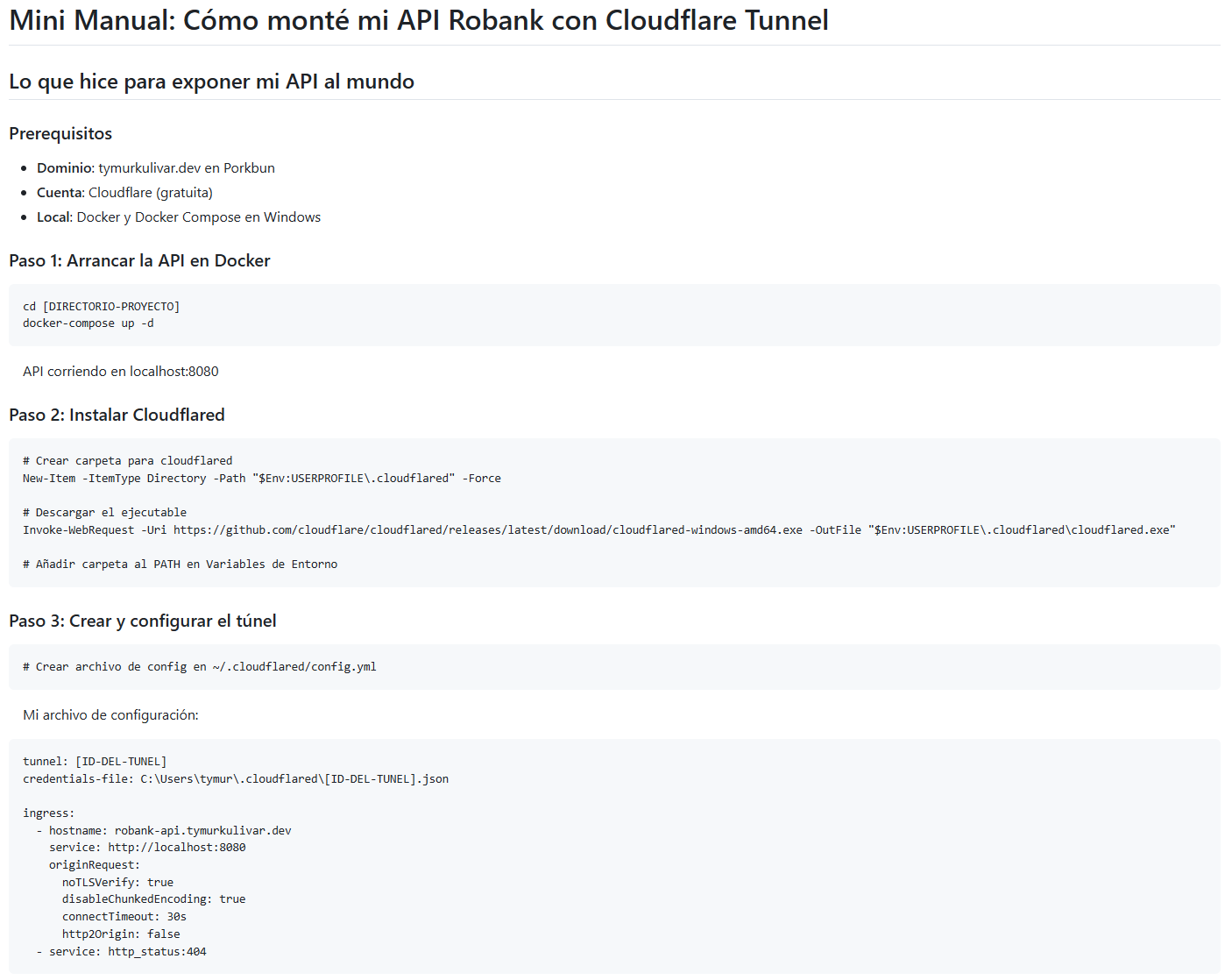
**General**: <https://github.com/Timasostima/Robank_TFG-DAM-2025_TymurKulivarShymanskyi>.

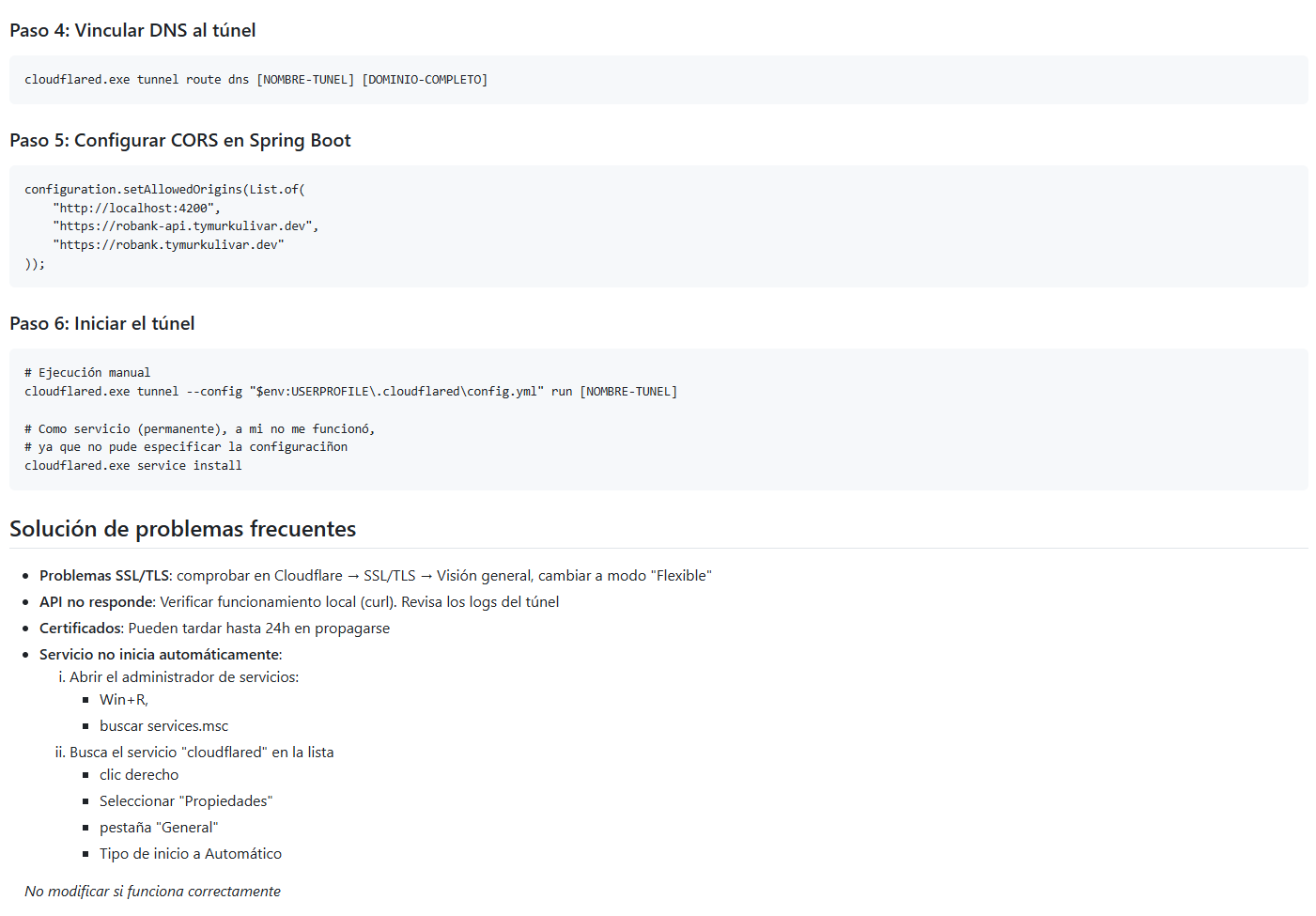
**Backend**: <https://github.com/Timasostima/robank_backend>.

**Frontend**: <https://github.com/Timasostima/robank_frontend>.

**Android**: <https://github.com/Timasostima/robank_android>.

**Perfil LinkedLn**: <https://www.linkedin.com/in/tymur-kulivar-shymanskyi/>.

Manual de despliegue de Backend:  




Muchas gracias a Javier Redondo Fernández, Víctor Sánchez Melero y Claudiu Iovanescu Alexandru Emanuel por ayudarme con la creación del proyecto.

1. Base de código: conjunto de archivos de código fuente que componen una aplicación o sistema software. [↑](#footnote-ref-1)
2. API REST: sistema basado en HTTP para gestionar recursos. Se explicará más adelante. [↑](#footnote-ref-2)
3. Fintech: sector que combina tecnología y servicios financieros para innovar en productos, procesos y modelos de negocio. [↑](#footnote-ref-3)
4. Neobancos: Bancos digitales que operan exclusivamente online, sin oficinas físicas y con gestión a través de apps. [↑](#footnote-ref-4)
5. Backend: Parte del sistema que procesa datos y gestiona la base de datos en el servidor. [↑](#footnote-ref-5)
6. Lógica de negocio: Reglas y procesos que definen cómo debe funcionar la aplicación. [↑](#footnote-ref-6)
7. Frontend: Interfaz visible con la que interactúan los usuarios. [↑](#footnote-ref-7)
8. Endpoint: es una URL específica dentro de una API que permite acceder a una funcionalidad o recurso determinado. [↑](#footnote-ref-8)
9. Repositorios: ubicaciones desde donde Gradle descarga las bibliotecas necesarias para el proyecto. [↑](#footnote-ref-9)
10. Dependencias: bibliotecas externas que el proyecto necesita para compilar y ejecutarse correctamente. [↑](#footnote-ref-10)
11. JSON (JavaScript Object Notation): formato ligero de intercambio de datos, fácil de leer para humanos y procesar por máquinas. [↑](#footnote-ref-11)
12. Seguridad de nulos: característica del lenguaje que previene errores causados por referencias nulas, obligando al desarrollador a manejar explícitamente los posibles valores. [↑](#footnote-ref-12)
13. Programación declarativa: estilo de programación en el que se describe qué se quiere lograr, en lugar de cómo hacerlo, delegando el control del flujo a la herramienta correspondiente. [↑](#footnote-ref-13)
14. Material Design 3: sistema de diseño de Google que permite crear interfaces modernas, adaptables y coherentes mediante componentes personalizables. [↑](#footnote-ref-14)
15. Aplicaciones de una sola página (SPA): Aplicación web que funciona en una sola página, actualizando contenido dinámicamente sin recargas completas. [↑](#footnote-ref-15)
16. Inyección de dependencias: patrón de diseño donde los objetos reciben sus dependencias desde el exterior, haciendo el código más modular y reutilizable. [↑](#footnote-ref-16)
17. Layout: estructura visual que organiza los elementos en una página web. [↑](#footnote-ref-17)
18. Interfaces responsivas: diseño web que se ajusta automáticamente a diferentes tamaños de pantalla y dispositivos. [↑](#footnote-ref-18)
19. Guardias de ruta: mecanismos de seguridad que controlan el acceso a diferentes secciones de la aplicación, verificando permisos y autenticación antes de permitir la navegación. [↑](#footnote-ref-19)
20. [↑](#footnote-ref-20)