ExpenseTrack

27/02/2025

Lucía Mengual Ramírez
I.E.S. Rafael Alberti

11012, Cádiz

Introducción

ExpenseTrack es una aplicación para Android pensada para hacerte la vida más fácil a la hora de llevar el control de tus finanzas personales. Permitiendo registrar ingresos y gastos de manera sencilla, categorizar las transacciones con categorías personalizadas, crear y gestionar tus cuentas personales, realizar transferencias entre cuentas, visualizar el gasto mensual de las categorías deseadas, visualizar el historial de transacciones y transferencias con distintos gráficos y filtros, además de poder exportar los datos de dichos historiales a archivo excel. Por último, es posible visualizar tu balance total convertido a tu divisa favorita en caso de tener cuentas con múltiples divisas.

Desarrollo de la idea inicial

La idea inicial de ExpenseTrack es desarrollar una herramienta intuitiva y eficiente para la gestión de finanzas personales al alcance de tu mano. Su propósito principal es ayudar a los usuarios a registrar, categorizar y analizar sus ingresos y gastos, permitiéndoles tener un mejor control sobre sus finanzas, ya que en la actualidad, muchos usuarios encuentran dificultades para ello.

Tecnologías utilizadas

- **Ionic.** Es un conjunto de herramientas de interfaz de usuario (UI Toolkit) de código abierto que utiliza tecnologías web como HTML, CSS y JavaScript/TypeScript para crear aplicaciones multiplataforma e híbridas web, móviles y de escritorio con integración para los framework más populares como Angular, Vue o bibliotecas como React.

- **React.** Es un conjunto de bibliotecas de Javascript/TypeScript que permiten construir interfaces de usuario web o nativas a partir de piezas individuales llamadas componentes. Los componentes son funciones de JavaScript/Typescript, lo que hacen que sean más fáciles de crear, mantener y eliminar.

- **Firebase.** Es una plataforma en la nube diseñada para facilitar el desarrollo de aplicaciones web y móviles. En esta aplicación se usarán los productos Firebase Cloud Firestore para almacenar los datos en una base de datos NoSQL y Firebase Authentication para la autenticación de los usuarios.

- **Typescript.** Es un lenguaje de programación superconjunto de JavaScript, que permite añadir tipos de datos estáticos y objetos basados en clases. Este será el lenguaje de programación principal de la aplicación.

- **ExchangeRate-API.** Es una API que ofrece en tiempo real la tasa de cambio de todas las divisas mediante la llamada a dicha API.

Aplicaciones similares

ExpenseTrack ha sido desarrollada basándose en las aplicaciones más populares y que reúnen más descargas en la Google Play Store. Aquí se recopilan alguna de estas aplicaciones:

- **Gestor de gastos - finanzas**

  (https://play.google.com/store/apps/details?id=ru.innim.my_finance&hl=es_419)

- **Wallet: Finanzas Personales**

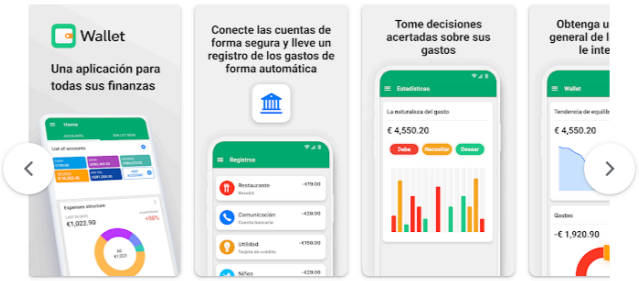
  (https://play.google.com/store/apps/details?id=com.droid4you.application.wallet&hl=es_419)

Wallet: Finanzas Personales
BudgetBakers
Compras directas desde la app

Tus finanzas en un solo lugar. Gestor de gastos y planificador de presupuestos.

4.6★ 339 k opiniones | 10 M+ Descargas | PEGI 3

Instalar



- **Spendee Budget & Money Tracker**

(https://play.google.com/store/apps/details?id=com.cleevio.spendee&hl=es_419)

Spendee Budget & Money Tracker
SPENDEE a.s.
Compras directas desde la app

Controla mejor tus finanzas con Spendee budgeting app para gastos e ingresos.

4.6★ 53.8 k opiniones | 1 M+ Descargas | PEGI 3

Instalar

- **Presupuesto Rápido - Gastos**

(https://play.google.com/store/search?q=Presupuesto%20R%C3%A1pido%20-%20Gastos&c=apps&hl=es_419)

Presupuesto Rápido - Gastos
AppFer SRL
Contiene anuncios · Compras directas desde la app

Administre sus finanzas personales y controle fácilmente su dinero y sus gastos

4.8★ 123 k opiniones | 5 M+ Descargas | PEGI 3

Instalar

Descripción

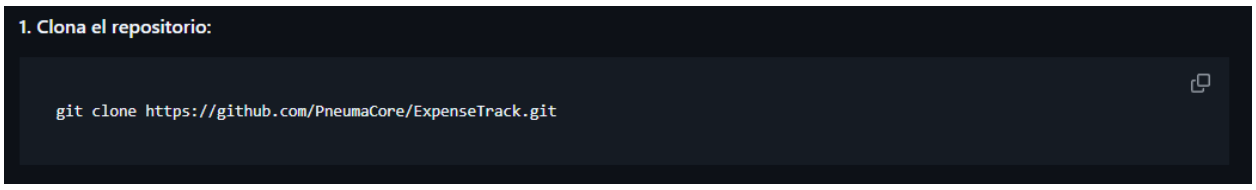
Funcionalidades

- Registrar transacciones de ingresos y gastos.
- Crear y gestionar cuentas personales.
- Transferencias entre cuentas y visualización del historial de transferencias.
- Clasificar transacciones en categorías de tipo gasto o ingreso (por defecto o personalizadas).
- Visualización de gasto mensual de una categoría deseada (presupuestos).
- Distintos tipos de filtro (por cuenta, fecha, nombre, descripción).
- Visualización del saldo total de todas las cuentas en la divisa preferida del usuario.
- Estadísticas detalladas y gráficos sobre finanzas personales.
- Exportación de historial de transacciones y transferencias a archivo excel.

Instalación

En este apartado se recopilan todos los pasos necesarios para el funcionamiento del proyecto y su instalación para su compilación con capturas directamente del repositorio de GitHub.

(https://github.com/PneumaCore/ExpenseTrack/blob/master/README.md)



2. Instala Ionic:

Antes de ejecutar la aplicación, deberás tener instalado Node.js en tu equipo. Una vez hecho esto, instala el cliente de Ionic con el siguiente comando:

```
npm install -g @ionic/cli
```

3. Instala las dependencias:

Navega al directorio de la aplicación y ejecuta el siguiente comando para instalar todas las dependencias:

```
cd ExpenseTrack
npm install
```

4. Configurar Firebase:

Para conectar la aplicación con Firebase, necesitarás las claves API de tu proyecto de Firebase. Sigue estos pasos para obtenerlas:

- Regístrate o inicia sesión en Firebase.

- Crea un nuevo proyecto llamado ExpenseTrack en Firebase Console.

- En la pantalla principal del proyecto, en el apartado para agregar Firebase a tu app, pulsa en 'Web', con esto, generarás las claves API.

- Ahora, crea un archivo '.env' en el directorio raíz de la aplicación y agrega las claves API generadas de la siguiente forma:

```
VITE_APP_API_KEY=tu_clave_api
VITE_APP_AUTH_DOMAIN=tu_clave_auth_domain
VITE_APP_PROJECT_ID=tu_clave_project_id
VITE_APP_STORAGE_BUCKET=tu_clave_storage_bucket
VITE_APP_MESSAGING_SENDER_ID=tu_clave_messaging_sender
VITE_APP_APP_ID=tu_clave_app_id
```

5. Crear colecciones en Cloud Firestore:

Además de las claves, necesitarás las colecciones de la base de datos que se manejan en la aplicación, para ello, vuelve nuevamente a la pantalla principal del proyecto de Firebase. Pulsa en 'Todos los productos' y busca en la lista 'Cloud Firestore' y pulsa en 'Crear base de datos'. Selecciona un servidor europeo e inicia la base de datos en modo de producción. A continuación, deberás crear las siugientes colecciones:

- Colección 'users':

Almacena los datos del usuario.

```
{
  user_id: Cadena,
  profile_photo: Cadena,
  name: Cadena,
  surname_1: Cadena,
  surname_2: Cadena,
  currency: Cadena,
  isAccountSetup: Boleano
}
```

- Colección 'accounts':

Almacena las cuentas personales del usuario.

```
{
  account_id: Cadena,
  user_id: Cadena,
  name: Cadena,
  currency: Cadena,
  balance: Número,
  icon: Cadena,
  color: Cadena
}
```

- Colección 'categories':

Almacena las categorías por defecto y las personalizadas por el usuario.

```
{
  category_id: Cadena,
  user_id: Cadena,
  name: Cadena,
  mensualBudget: Número,
  type: Cadena,
  icon: Cadena,
  color: Cadena
}
```

- Colección 'transactions':

Almacena las transacciones realizadas por el usuario.

```
{
  transaction_id: Cadena,
  user_id: Cadena,
  type: Cadena,
  category_id: Cadena,
  account_id: Cadena,
  amount: Número,
  currency: Cadena,
  date: Marca de tiempo,
  note: Cadena,
  image: Array
}
```

- Colección 'recurringTransactios':

Almacena las transacciones recurrentes programadas por del usuario.

```
{
  recurring_transaction_id: Cadena,
  user_id: Cadena,
  type: Cadena,
  name: Cadena,
  category_id: Cadena,
  account_id: Cadena,
  amount: Número,
  currency: Cadena,
  date: Marca de tiempo,
  frequency: Cadena,
  next_execution: Marca de tiempo
}
```

- Colección 'transfers':

Almacena las transferencias entre cuentas realizadas por del usuario.

```
{
  transfer_id: Cadena,
  user_id: Cadena,
  source_account_id: Cadena,
  destination_account_id: Cadena,
  amount: Número,
  converted_amount: Número,
  source_currency: Cadena,
  destination_currency: Cadena,
  date: Marca de tiempo,
  note: Cadena
}
```

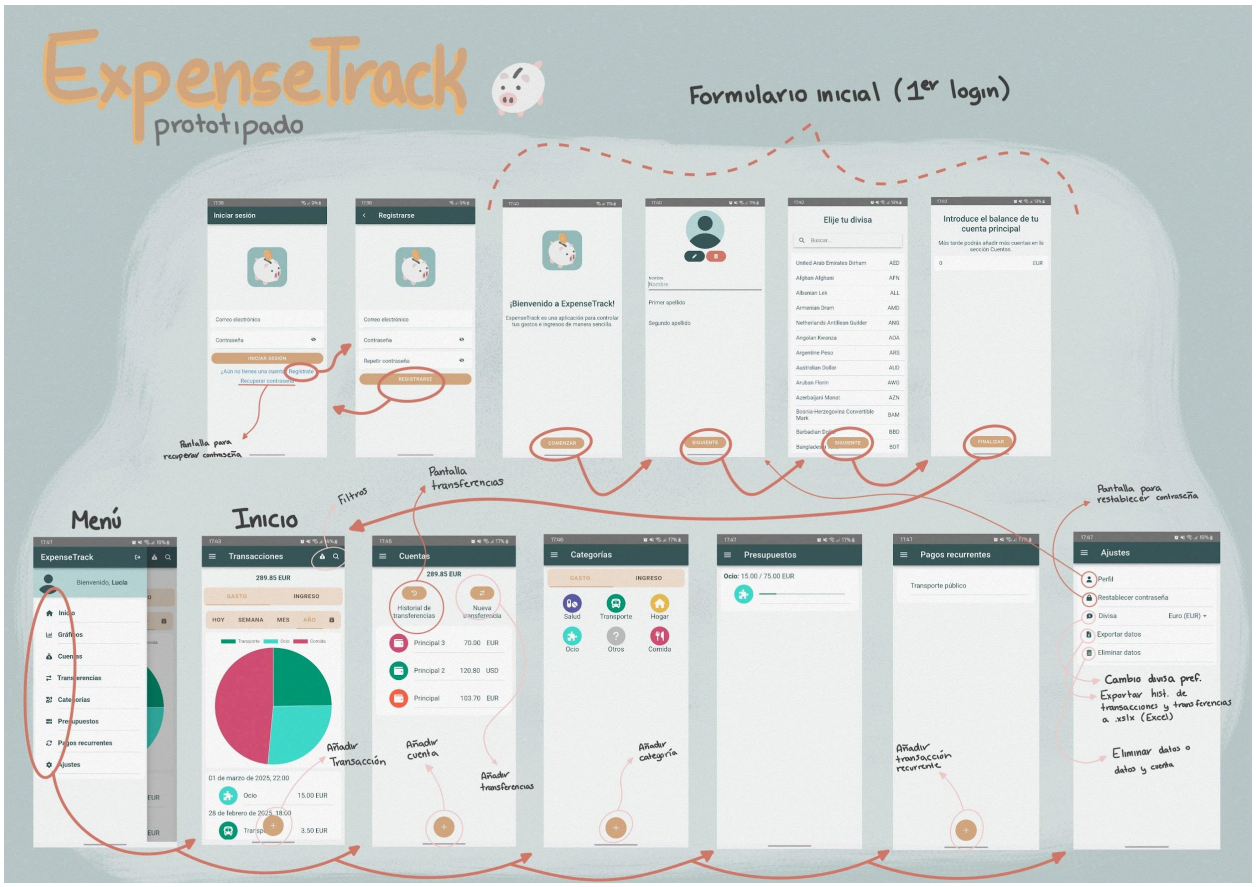
6. Ejecuta la aplicación:

```
ionic serve
```

Prototipado

Prototipo wireframe de ExpenseTrack. Para mayor calidad de visualización, el prototipo se encuentra en el repositorio de GitHub.

(https://github.com/PneumaCore/ExpenseTrack/blob/master/README.md)



Diseño funcional

ExpenseTrack es una aplicación diseñada para gestionar las finanzas personales de forma sencilla e intuitiva. Los usuarios pueden registrar ingresos y gastos, gestionar múltiples cuentas, realizar transferencias y visualizar gráficos sobre sus finanzas personales. La aplicación ofrece opciones de personalización, como categorías personalizadas y conversión de saldo a la divisa preferida del usuario.

A nivel funcional, ExpenseTrack se estructura en los siguientes módulos principales:

- **Gestión de usuarios:** autenticación mediante Firebase.
- **Gestión de cuentas:** creación, edición y eliminación de cuentas.
- **Registro de transacciones:** ingresos y gastos asociados a una cuenta y categoría.
- **Transferencias entre cuentas:** movimiento de saldo entre cuentas del usuario.
- **Visualización de estadísticas:** gráficos y reportes sobre los gastos e ingresos.
- **Exportación de datos:** generación de archivos Excel con el historial de transacciones.
- **Conversión de divisas:** cálculo del saldo total en la divisa preferida utilizando ExchangeRate-API.

Diagrama entidad relación (ER)

Dado que ExpenseTrack utiliza Firebase Cloud Firestore como base de datos, la estructura de almacenamiento se basa en colecciones y documentos no relacionales. Sin embargo, los documentos han sido relacionados de manera similar a las bases de datos tradicionales.

- **Usuarios (users):** Almacena los datos del usuario.
- **Cuentas (accounts):** Almacena las cuentas personales del usuario.
- **Categorías (categories):** Almacena las categorías por defecto y las personalizadas por el usuario.
- **Transacciones (transactions):** Almacena las transacciones realizadas por el usuario.

- **Transacciones recurrentes (recurringTransactions):** Almacena las transacciones recurrentes programadas por el usuario.
- **Transferencias (transfers):** Almacena las transferencias entre cuentas realizadas por el usuario.

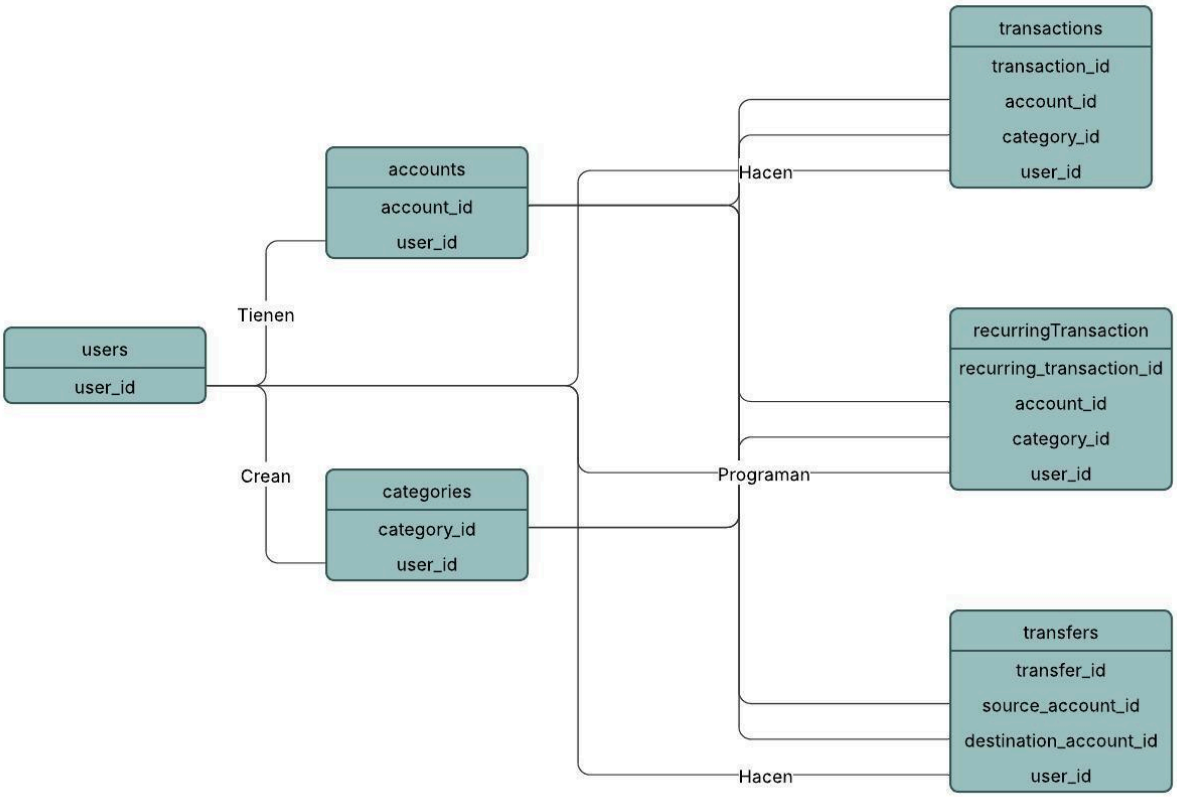
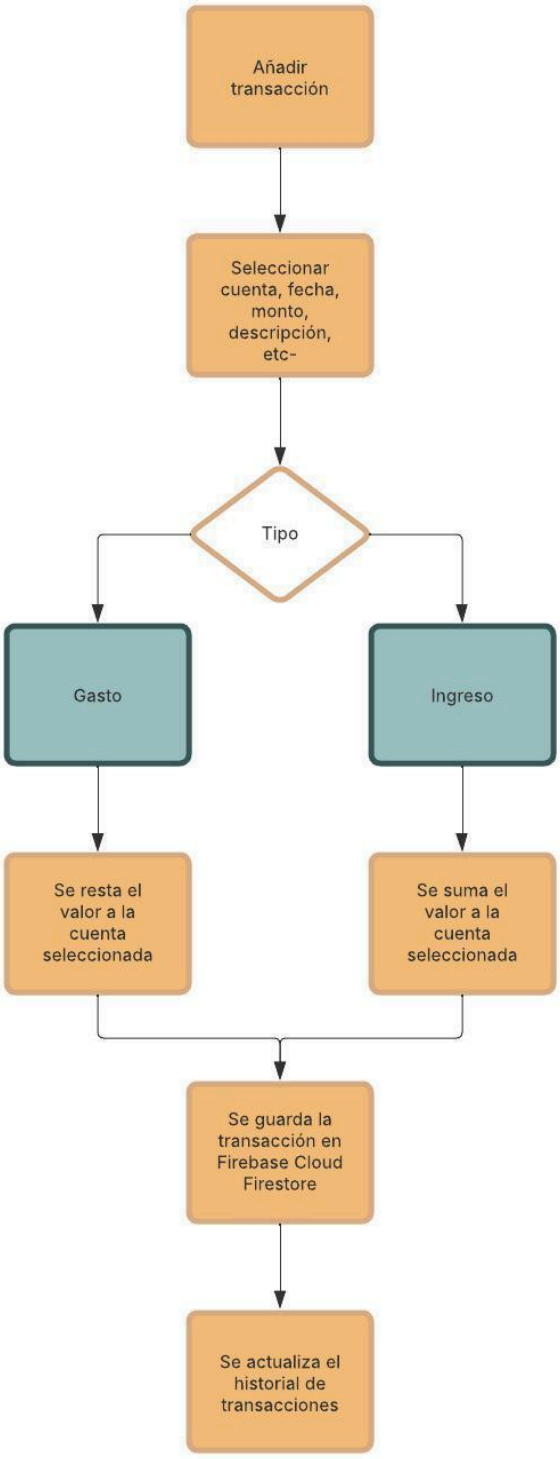
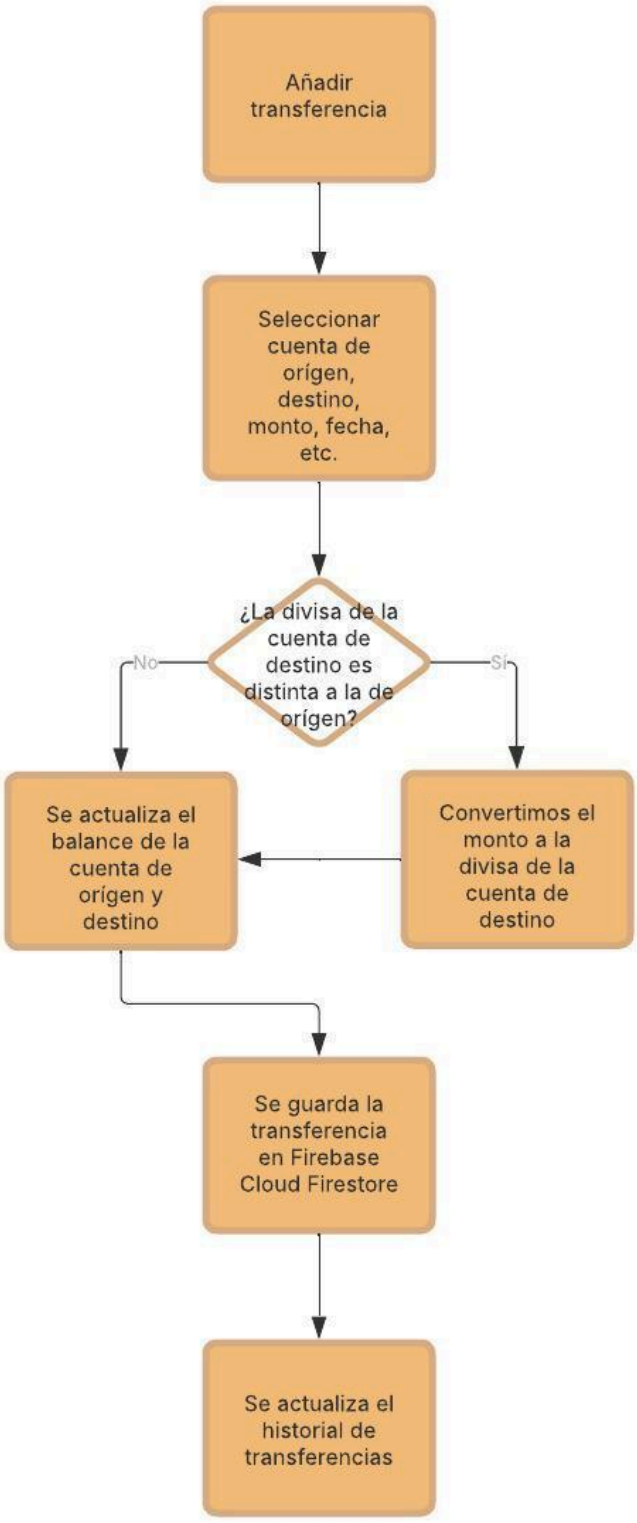


Diagrama de flujo

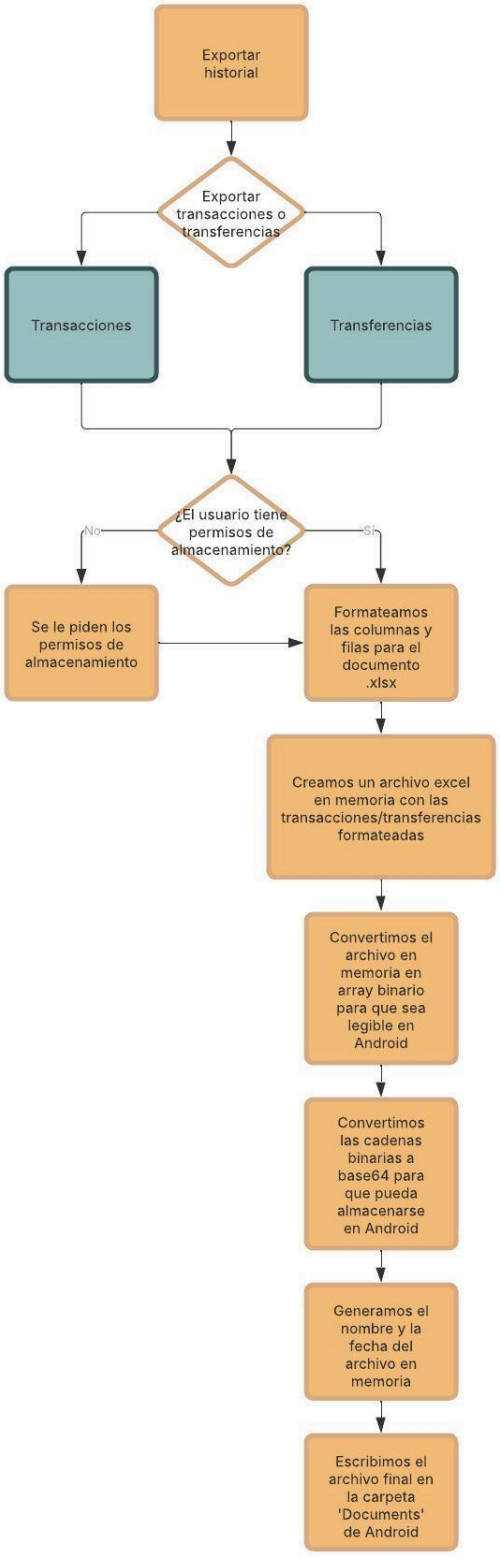
Registrar transacciones



Realizar una transferencia entre cuentas



Exportar historial de transacciones o transferencias a excel



Desarrollo

Secuencia de desarrollo

Planificación inicial

Inicialmente, la aplicación iba a ser desarrollada con React Native, ya que estuvo presente la posibilidad de aprenderlo en las prácticas. Una vez allí, me comentaron que cambiaron a Ionic ya que les resultaba más cómodo y que la mayoría de las aplicaciones que tenían desarrolladas en la empresa eran con Ionic. Finalmente, me decanté por Ionic, ya que era lo que iban a enseñar en la empresa y quería poner en práctica los conocimientos adquiridos. La temática de la aplicación fue decidida unos meses antes ya que me resultaba interesante desarrollar una aplicación como esta. Se decidió por usar Firebase Cloud Firestore como base de datos ya que me interesaba que los datos de mi aplicación pasaran de un dispositivo a otro.

Interfaz

En el desarrollo de la interfaz de la aplicación, nunca se realizó un prototipo como tal de cómo se vería la aplicación. La idea de la interfaz siempre estuvo en el aire, basándome en las interfaces de las aplicaciones más descargadas de la Google Play Store similares a ExpenseTrack. Aunque a principios del desarrollo, esto me causó un poco de problemas, ya que no tenía un diseño fijo y tuve que cambiarlo más adelante del desarrollo porque no me gustaba del todo.

Implementación de Firebase

Lo primero que hice después de crear el proyecto, fue implementar Firebase, una vez que aseguré que funcionaba correctamente con mi aplicación, hice las pantallas para inicio de sesión y registro, implementando así la autenticación de la aplicación.

Funcionalidades básicas

Después de las pantallas de inicio de sesión, hice las pantallas de funcionalidades básicas y sus funcionalidades, como listar transacciones, editarlas o eliminar, lo mismo con categorías, cuentas, transferencias, etc.

Implementación de ExchangeRate-API

Cuando tenía las funcionalidades básicas asentadas, decidí implementar la API de conversión de divisas, ya que si el usuario tenía varias cuentas con más de una divisa siempre pudiera visualizar su total en su divisa preferida. También, la API era necesaria para las transferencias entre cuentas con distintas divisas, para que el monto siempre se convirtiera en la divisa de la cuenta de destino.

Exportación de datos

Implementé que el usuario pudiera exportar todo su historial de transacciones y transferencias a archivo excel por si deseaba conservar el archivo en algún dispositivo o manipular sus datos en alguna aplicación de contabilidad.

Eliminación de datos y cuenta

Para que el usuario pudiera empezar de cero a usar la aplicación si en algún momento lo necesitaba, implementé una opción para que pudiera hacerlo eliminando de Firebase Cloud Firestore todos los documentos realizados por el. Además de implementar la misma funcionalidad, pero eliminando su cuenta de la base de datos, por si simplemente desea darse de baja de la aplicación y no usarla más.

Dificultades encontradas

- **Firebase.** Al principio del desarrollo, me costó implementar y hacer funcionar Firebase, ya que desconocía que para agregar las

claves API al proyecto, tenían que tener un nombre en específico en el fichero .env, esto se me hizo muy frustrante.

- **Notificaciones recurrentes.** En un punto del desarrollo, decidí implementar una pantalla para añadir notificaciones o recordatorios nativos de Android. Al final, tuve que descartar esa idea, ya que mis dispositivos reales, mi teléfono móvil y mi tablet, tenían versiones demasiado recientes de Android. En dichas versiones, Android gestiona según su criterio la relevancia de las notificaciones según la aplicación. Lo que provocaba, que las notificaciones llegaran, a veces con retraso o directamente nunca. Actualmente, la pantalla de notificaciones no existe en el código, pero su código puede ser revisado en el control de versiones de GitHub.
- **Conversión de divisas.** Me costó plantear en qué apartados usar la conversión entre divisas, ya que fue una funcionalidad que añadí cuando la aplicación era prácticamente funcional.

Decisiones afrontadas

- **Transacciones recurrentes.** Como mencione en el punto anterior, tuve dificultades implementando los recordatorios en mi aplicación, así que en su lugar, decidí implementar que el usuario pudiera añadir transacciones recurrentes, en vez de que la aplicación le recordara al usuario que tenía que hacer alguna transacción, la transacción se realiza automáticamente al abrir la aplicación en la fecha, hora y frecuencias seleccionadas por el usuario, lo cual es una mejora respecto a la idea anterior.
- **Conversión de divisas.** Decidí que la conversión de divisas se tuviera en cuenta al visualizar el balance total de todas las cuentas y en las transferencias entre una cuenta a otra.

Herramientas de control de versiones y revisión de código

Para el control de versiones, utilicé Git y GitHub. Trabajé siempre en la misma rama y me cercioraba de que el código funcionara correctamente antes de hacer 'push' a los cambios. Como hacía 'commits' de manera constante, si me equivocaba siempre volvía a un punto anterior para arreglar las excepciones.

Pruebas

Pruebas funcionales

Se verificó cada funcionalidad clave de la aplicación como el registro de transacciones, cuentas, categorías, etc, funcionaran como se esperaba. Teniendo en cuenta funcionalidades más críticas, como la conversión de divisas o la de exportar a archivo excel, ya que eran funcionalidades graves en caso de no funcionar.

Pruebas de interfaz

La aplicación se probó en varios tipos de pantallas, tanto virtuales como pantallas de dispositivos físicos. Pude probarla en mi dispositivo móvil y en mi tablet. Por suerte, no hubo demasiados problemas, gracias a que Ionic y React enriquecen mucho el apartado 'responsive' de la aplicación.

Pruebas de integración

Realicé pruebas de integración con la API de conversión de divisas, comprobando que realmente se actualizaban cada día las tasas de conversión en tiempo real. También hubo pruebas de integración con Firebase Cloud Firestore, comprobando que obtenía información de la base de datos o que se guardaban los datos correctamente en ella.

Todas las pruebas anteriormente mencionadas, fueron realizadas de forma manual y no automatizada debido a que era la primera vez que

desarrollaba un proyecto tan extenso con este lenguaje de programación y desconocía las librerías necesarias para ello. A futuro, desearía aprender más sobre librerías de JavaScript/TypeScript como Jest o React Testing para mejorar la calidad del código.

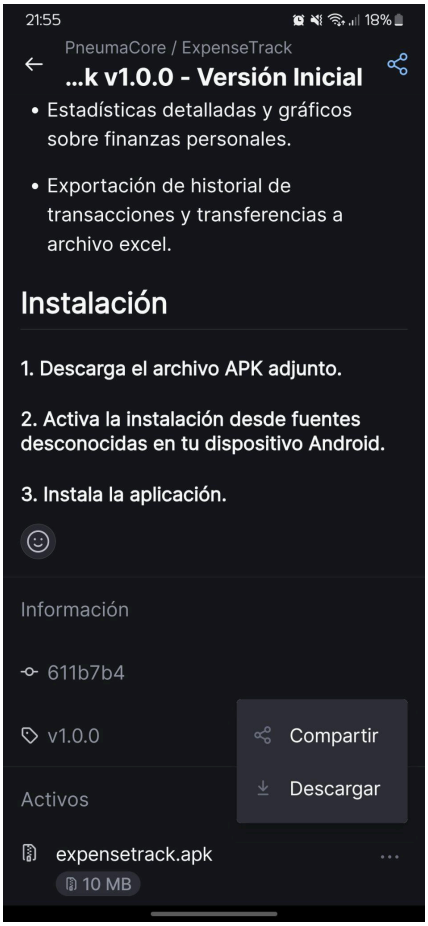
Distribución

ExpenseTrack es distribuida a través de su repositorio de GitHub mediante un archivo APK en su sección de 'releases'. En el siguiente apartado se explicará su instalación.

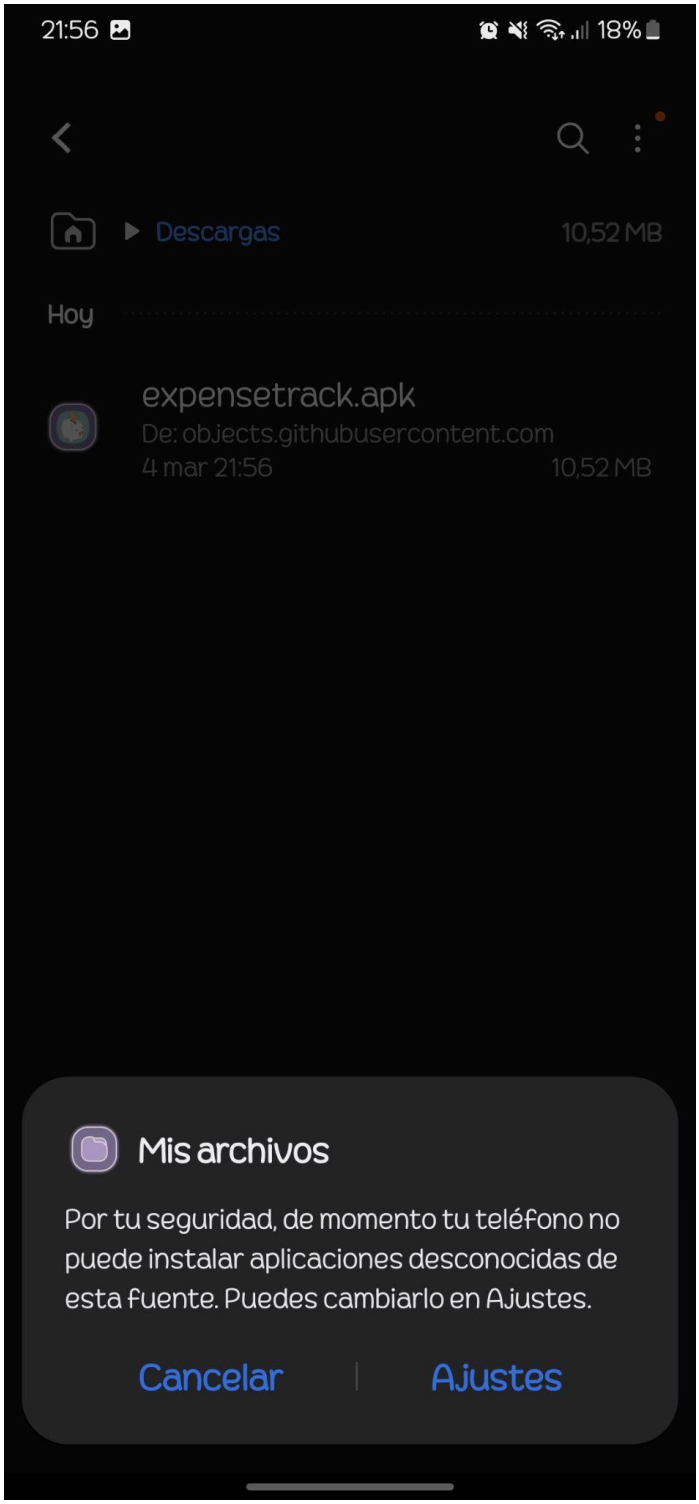
(https://github.com/PneumaCore/ExpenseTrack/releases/tag/v1.0.0)

Manual

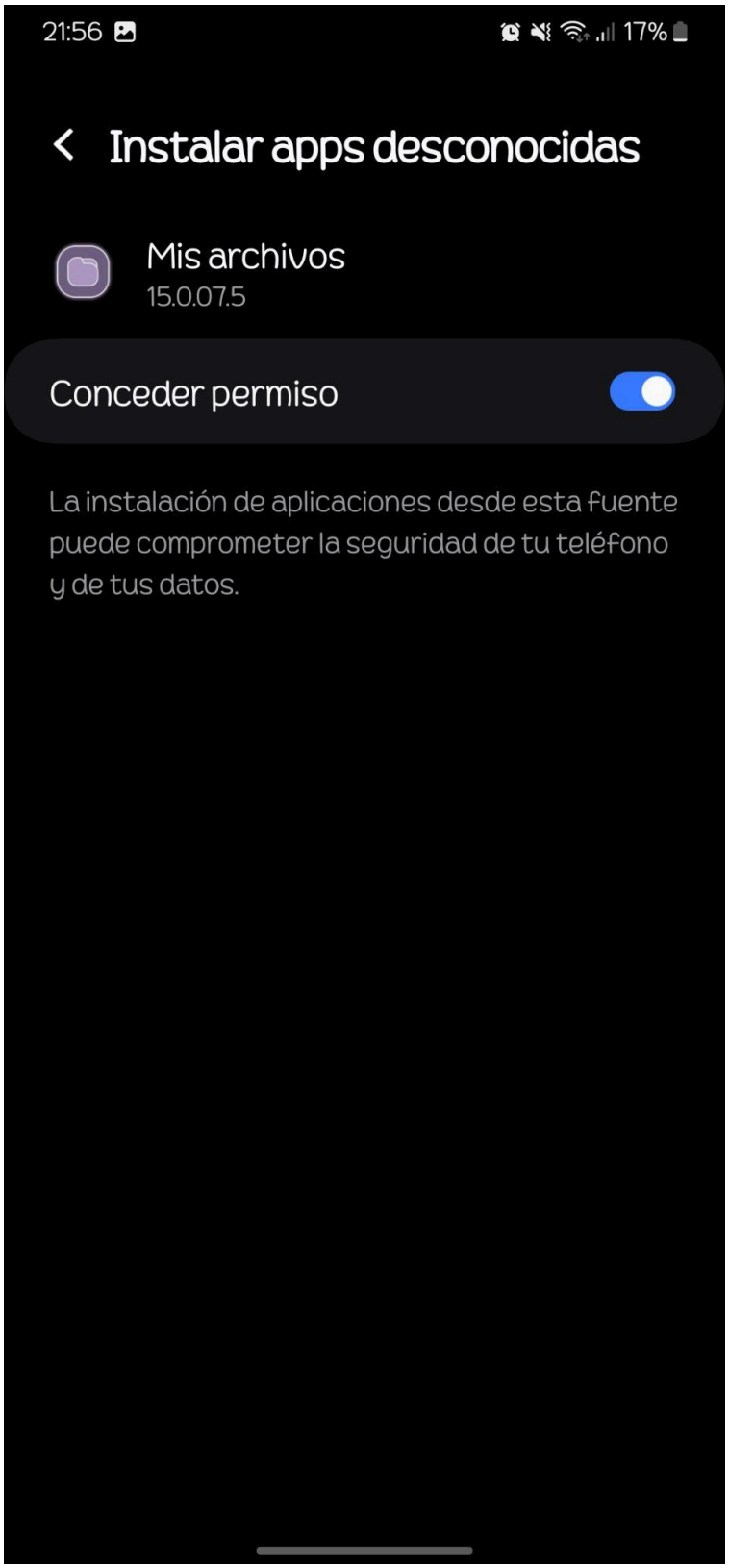
1. Descargamos el archivo APK directamente desde el repositorio.



2. Nos dirigimos a la carpeta ‘Descargas’ de nuestro dispositivo. Al intentar instalar la aplicación nos pedirá permisos de orígenes desconocidos.

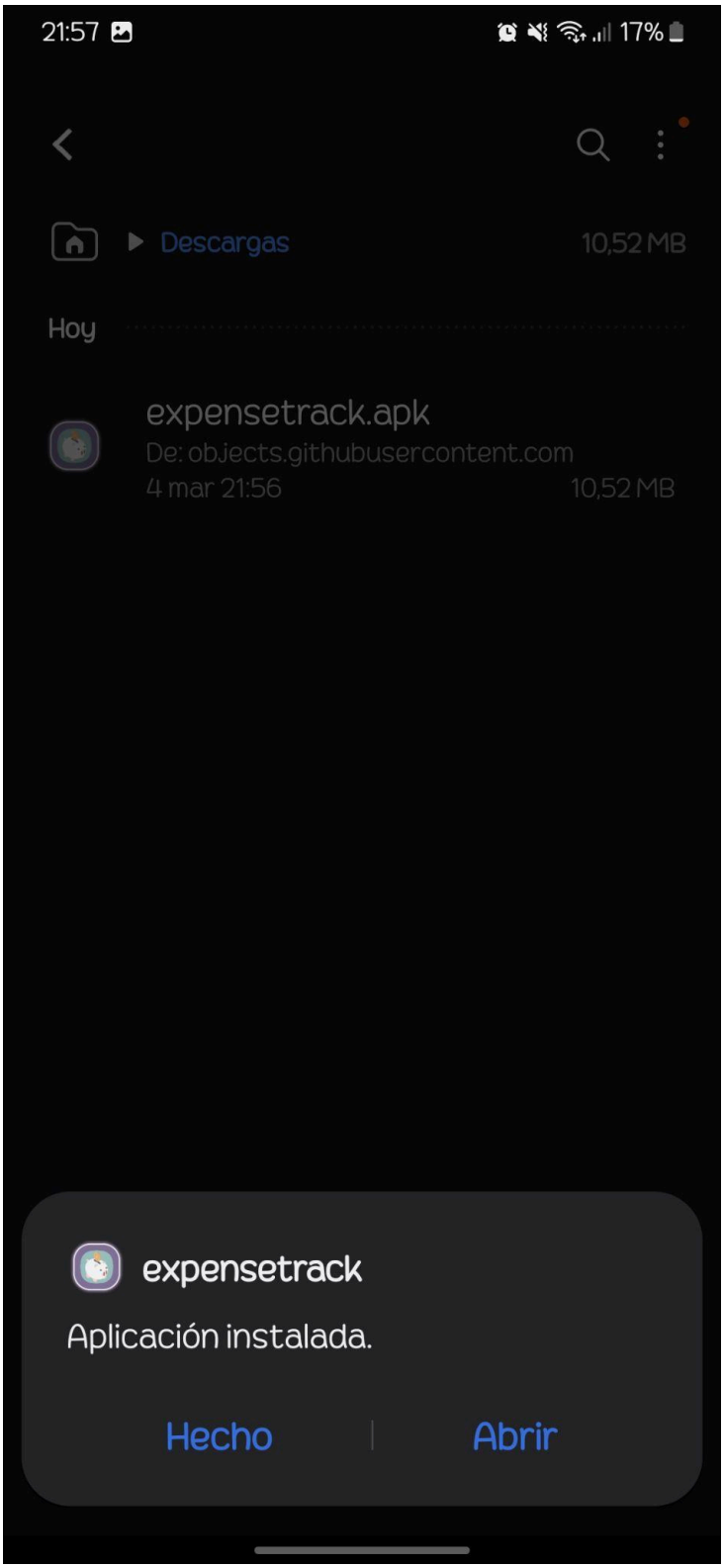


3. Activamos los permisos de orígenes desconocidos.



4. Ahora sí, instalamos la aplicación.

5. Esperamos a que la aplicación se instale por completo y ¡Listo!



Conclusiones

Comparación del resultado con la idea inicial

En conclusión, estoy muy orgullosa del resultado final obtenido con ExpenseTrack. Me siento satisfecha porque se podría decir que he obtenido los resultados esperados comparándolos con la idea inicial. Incluso con el problema que tuve con el apartado de recordatorios recurrentes, logré implementar las transacciones recurrentes automáticas, que podría decirse que es una mejora frente a los recordatorios, aunque me gustaría intentarlo nuevamente en el futuro. Disfruté mucho el desarrollo de la aplicación gracias a Ionic y React, para ser la primera aplicación que desarrollo con estas tecnologías, se me hizo muy interesante y ameno el desarrollo del proyecto, incluso divertido. Firebase me dió problemas de implementación al principio, también fue una tecnología muy intuitiva y fácil de usar. Me siento muy gratificada con el resultado de la aplicación final, aun así, pienso que tiene mucho potencial para futuras mejoras que voy a nombrar en el siguiente apartado.

Futuras mejoras

- **Implementación de recordatorios o notificaciones 'push'.**
- **Traducción a varios idiomas.** La mayoría del código está en inglés porque en un principio la aplicación estaba pensada para desarrollarse en inglés. Además, algunos componentes nativos de Ionic o React aparecen en inglés por el momento (a la hora de seleccionar o hacer una foto desde la galería el diálogo de texto aparece en inglés).
- **Mejorar la autenticación con huella dactilar o biometría.**
- **Modo 'offline' con sincronización al recuperar la conexión.**
- **Inicio de sesión automático con Google o Facebook.**
- **Crear una versión para dispositivos iOS o web.**

- **Implementación de alguna tecnología para escanear los tickets o recibos.**
- **Posibilidad de visualizar las imágenes de las transacciones dentro de la aplicación.**

Bibliografía

Referencias

- ExpenseTrack

  (https://github.com/PneumaCore/ExpenseTrack)

- Ionic

  (https://ionicframework.com/docs)

- React

  (https://es.react.dev/reference/react)

- Firebase

  (https://firebase.google.com/docs?hl=es-419)

- Firebase Cloud Firestore

  (https://firebase.google.com/docs/firestore?hl=es-419)

- Firebase Authentication

  (https://firebase.google.com/docs/firestore?hl=es-419)

- TypeScript

  (https://www.typescriptlang.org/docs/)

- ExchangeRate-API

  (https://www.exchangerate-api.com/docs/overview)