**CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL**

*Organismo Público Descentralizado Federal*

**

**Aplicación móvil para finanzas personales y ahorro**

**INTEGRANTES**

Heredia Pérez Héctor Alonso

Monraz Chávez Luis Alejandro

**ASESOR**

Molina Martínez Carlos

**GUADALAJARA, JAL.**

# DEDICATORIAS Y AGRADECIMIENTOS

Héctor: No me gustaría comenzar una dedicatoria sin agradecer antes a mi familia, quienes, con tanta paciencia y afecto me han ayudado a labrar un camino muy tosco, y, que desde muy chico me han forjado para ser la mejor versión de mí que sé que puedo ser. Ha sido un duro camino, en más de cuatro años aprendí que la mayor gloria no es no caer, sino siempre levantarse, y es por eso que dedico este trabajo a mi madre, siempre atenta y preocupada por mi bienestar, por encima de cualquier deber, pues ella me enseñó algo que no entendía del todo, y era cuidar la salud por encima de cualquier calificación u obstáculo. Cometo errores, más a menudo de lo que quisiera, pero ninguno de ellos es una falla, porque he aprendido de cada uno de esos tropiezos todo este tiempo, cosa que no sería así si no tuviera tanto apoyo de mi padre, a quien le dedico mi proyecto por igual. A mis profesores les agradezco todas las enseñanzas y paciencia que me tuvieron, particularmente al profe Coco y a Brambila, quienes siempre estuvieron ahí para resolver mis dudas de novato. Agradezco profundamente a esta institución llamada CETI, pues me hizo aprender por las buenas que para ser bueno en algo tienes que tener la suficiente voluntad para ser malo en ello por un buen tiempo. A los amigos que conocí, a los que mantengo y a los que perdí, porque de cada persona he aprendido, y cada aprendizaje representa un peldaño, un peldaño que coloco cada día en mi camino al éxito. ¡Gracias a todos!

Alex: No me dan las palabras para agradecerle lo suficiente a mis padres quienes desde pequeño me han ayudado e impulsado en todas las decisiones que he tomado sean para mi beneficio o aprendizaje, a mis amigos y compañeros por sus palabras de aliento, por su compañía y aún más importante por poder contar con ellos. Agradezco con toda mi sinceridad al CETI y a sus profesores por haber sido parte de mi formación académica y aún más importante por todas lecciones de vida que pude experimentar y todo el conocimiento, paciencia y retroalimentación de mis profesores.

Y a Lalan

# RESUMEN

En México la educación y sanidad financiera es un problema que está muy presente en la sociedad y lleva siendo así por un largo tiempo. El manejo de las finanzas personales es una de las muchas habilidades que desgraciadamente no son enseñadas en la escuela y muchas veces tampoco en el hogar; llevando a tener una vida económica desfavorable y muchas veces insuficiente para los mexicanos.

Durante la carrera de desarrollo de software se lleva a cabo la programación móvil, la cual es coordinada por la academia de programación, en la cual desarrollamos competencias y conocimientos para desarrollar aplicaciones, gracias a ellas llegamos a la conclusión de que una aplicación móvil es el mejor método para brindar una herramienta financiera.

En el presente documento se mostrarán todos los procesos realizados para la elaboración del proyecto “Aplicación móvil para finanzas personales y ahorro”. El proyecto será un prototipo funcional de aplicación móvil para uso de cualquier persona con cualquier estado financiero y conocimiento básico del manejo de un celular. La aplicación cuenta con múltiples módulos interactivos los cuales buscan brindar al usuario una experiencia agradable e intuitiva. El objetivo es ofrecer la herramienta más completa y útil sin ser necesariamente sofisticada para toda la población, adaptándose a sus necesidades.

# Abstract

In Mexico, financial education and health is a problem that is very present in society and has been this way for a long time. Managing personal finances is one of the many skills that unfortunately are not taught at school and often not at home, leading to an unfavorable and often insufficient economic life for Mexicans.

During the software development career, mobile programming is carried out, which is coordinated by the programming academy, in which we develop skills and knowledge to develop applications, thanks to them we come to the conclusion that a mobile application is the best method to provide a financial tool.

This document will show all the processes carried out to develop the project “Mobile application for personal finances and savings”. The project will be a functional prototype of a mobile application for use by anyone with any financial status and basic knowledge of using a cell phone. The application has multiple interactive modules which seek to provide the user with a pleasant and intuitive experience. The objective is to offer the most complete and useful tool without necessarily being sophisticated for the entire population, adapting to their needs.

AGO-DIC 2023

**Tabla de contenido**

Contenido

[DEDICATORIAS Y AGRADECIMIENTOS 2](#_Toc155347790)

[RESUMEN 3](#_Toc155347791)

[Abstract 4](#_Toc155347792)

[Índices Particulares 6](#_Toc155347793)

[INTRODUCCIÓN 6](#_Toc155347794)

[Justificación 7](#_Toc155347795)

[NOMENCLATURAS 8](#_Toc155347796)

[CAPÍTULO I. EL AHORRO EN MÉXICO 9](#_Toc155347797)

[1.1 LA ECONOMÍA DEL MEXICANO 9](#_Toc155347798)

[1.2 EL MANEJO DEL DINERO 9](#_Toc155347799)

[1.3 FONDOS DE EMERGENCIA 10](#_Toc155347800)

[CAPÍTULO II. LA INFLUENCIA DE LA TECNOLOGÍA COMO HERRAMIENTA 11](#_Toc155347801)

[2.1 ORÍGENES DE LA TECNOLOGÍA PORTÁTIL 11](#_Toc155347802)

[2.2 ACCESIBILIDAD Y USO EN MÉXICO 11](#_Toc155347803)

[2.3 USO Y VICIO 11](#_Toc155347804)

[CAPÍTULO III. HERRAMIENTAS FINANCIERAS 12](#_Toc155347805)

[3.1 TRANSPARENCIA ENTRE BANCOS 12](#_Toc155347806)

[3.2 ¿DESINFORMACIÓN O DESIDIA? 12](#_Toc155347807)

[3.3 IMPACTO POST COVID-19 12](#_Toc155347808)

[3.4 ¿EL PASADO SIGUE SIENDO EFECTIVO EN LA ACTUALIDAD? 13](#_Toc155347809)

[CAPÍTULO IV. E-RRAMIENTAS FINANCIERAS 14](#_Toc155347810)

[4.1 PRACTICIDAD Y EFECTIVIDAD 14](#_Toc155347811)

[4.2 TIPOS DE HERRAMIENTAS 14](#_Toc155347812)

[4.3 HERRAMIENTAS DE AHORRO Y PRESUPUESTOS 15](#_Toc155347813)

[CAPÍTULO V. ENTORNO DE DESARROLLO DE SAVETRACK 16](#_Toc155347814)

[5.1 DISPONIBILIDAD 16](#_Toc155347815)

[5.1.1 SERVIDORES COMO HOST 16](#_Toc155347816)

[5.1.2 API’S 16](#_Toc155347817)

[5.2 JAVA VS KOTLIN 17](#_Toc155347818)

[5.3 VERSATILIDAD EN KOTLIN 18](#_Toc155347819)

[CAPÍTULO VI. LIMITACIONES 19](#_Toc155347820)

[6.1 DESIDIA Y MAL USO 19](#_Toc155347821)

[6.2 CREDENCIALES Y PERMISOS 19](#_Toc155347822)

[6.3 AMPLITUD DEL SISTEMA 19](#_Toc155347823)

[CAPÍTULO VII. PLANIFICACIÓN Y VISIÓN DE SAVETRACK 21](#_Toc155347824)

[7.1 ANÁLISIS DE REQUERIMIENTOS 21](#_Toc155347825)

[7.1.1 CARACTERÍSTICAS GENERALES 21](#_Toc155347826)

[7.1.2 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES 21](#_Toc155347827)

[Requerimientos Funcionales 21](#_Toc155347828)

[Requerimientos no Funcionales 25](#_Toc155347829)

[7.2 MODELADO GRÁFICO 32](#_Toc155347830)

[7.3 FASES DEL PROYECTO 33](#_Toc155347831)

[CAPÍTULO VIII. IMPLEMENTACIÓN/DESARROLLO DEL PROYECTO 38](#_Toc155347832)

[1 Gradle 38](#_Toc155347833)

[2 Login 40](#_Toc155347834)

[2.1 login binding al fragment 40](#_Toc155347835)

[2.2 caché de usuarios en login 41](#_Toc155347836)

[2.2.1 Obtener el caché de usuarios 42](#_Toc155347837)

[2.3 animaciones de navegación 43](#_Toc155347838)

[2.4 Ingreso de datos y concatenación de query 43](#_Toc155347839)

[2.5 Comprobación de credenciales y redirect 44](#_Toc155347840)

[2.6 comprobación de la base de datos nativa 45](#_Toc155347841)

[2.7 Crear backup 46](#_Toc155347842)

[2.8 Upload de backup a la base de datos 47](#_Toc155347843)

[2.9 Verificación de id y restore de la bd 54](#_Toc155347844)

[3 Recuperar cuenta 61](#_Toc155347845)

[3.1 Envío de correo 61](#_Toc155347846)

[3.2 Obtener Email para recuperación 63](#_Toc155347847)

[3.3 Reestablecer contraseña 65](#_Toc155347848)

[4 Registrar cuenta 67](#_Toc155347849)

[4.1 Register 67](#_Toc155347850)

[5 Index 71](#_Toc155347851)

[5.1 Inflator y generación de vistas 71](#_Toc155347852)

[5.2 Navegación entre activitys y fragments 72](#_Toc155347853)

[5.3 Incialización de index 73](#_Toc155347854)

[5.4 Iniciación de apariencia 74](#_Toc155347855)

[5.5 labels 76](#_Toc155347856)

[5.6 Setup PieChart gastos 77](#_Toc155347857)

[5.7 setup PieChart ingresos 80](#_Toc155347858)

[5.8 Redirects Index 83](#_Toc155347859)

[5.9 Interacción de PieChart y medidor de ahorro 85](#_Toc155347860)

[5.10 Calendario y actualización de PieChart 87](#_Toc155347861)

[5.11 Selección de etiquetas en PieChart 91](#_Toc155347862)

[5.12 Suspend Montos 93](#_Toc155347863)

[5.13 Setup de alarmas 96](#_Toc155347864)

[5.14 Montos por pagar 97](#_Toc155347865)

[5.15 Setup IndexAdd 102](#_Toc155347866)

[5.16 Index Add Confirm 106](#_Toc155347867)

[5.17 Montos especiales 111](#_Toc155347868)

[5.18 Tabulación de montos 115](#_Toc155347869)

[5.19 Tabulación ingresos 119](#_Toc155347870)

[5.20 Preparación de mandados 121](#_Toc155347871)

[5.21 Mandados 125](#_Toc155347872)

[5.22 Update Monto 127](#_Toc155347873)

[5.23 Confirmar y Cancelar Update 131](#_Toc155347874)

[5.24 Por pagar 136](#_Toc155347875)

[6 Perfil 138](#_Toc155347876)

[6.1 Bajar foto 138](#_Toc155347877)

[6.2 Mostrar datos del usuario 139](#_Toc155347878)

[6.3 Display Adapter Grupos 142](#_Toc155347879)

[6.4 Grupos del usuario 142](#_Toc155347880)

[6.5 Cambiar la foto sin internet 145](#_Toc155347881)

[6.6 Preparar upload foto 145](#_Toc155347882)

[6.7 Upload foto 147](#_Toc155347883)

[6.8 Guardar cambios 149](#_Toc155347884)

[6.9 Get grupos 152](#_Toc155347885)

[6.10 Agregar grupos y usuarios 153](#_Toc155347886)

[6.11 Lista de grupos 156](#_Toc155347887)

[6.12 Vista de grupo 158](#_Toc155347888)

[6.13 Compartir grupo redes sociales 160](#_Toc155347889)

[6.14 Salir del grupo 162](#_Toc155347890)

[6.15 Agregar monto 164](#_Toc155347891)

[6.16 Lista de miembros 169](#_Toc155347892)

[6.17 Editar grupo 172](#_Toc155347893)

[6.18 Eliminar grupo 175](#_Toc155347894)

[7 Historial 176](#_Toc155347895)

[7.1 Fragment historial 176](#_Toc155347896)

[7.2 Etiquetas 178](#_Toc155347897)

[7.3 Historial 179](#_Toc155347898)

[7.4 Modificar etiquetas 182](#_Toc155347899)

[7.6 Historial montos 185](#_Toc155347900)

[7.7 Papelera 187](#_Toc155347901)

[7.8 Favoritos 190](#_Toc155347902)

[8 Planes de Ahorro 193](#_Toc155347903)

[8.1 Diseño planes de ahorro 193](#_Toc155347904)

[8.2 Inicialización de montos 195](#_Toc155347905)

[9 Finanzas 202](#_Toc155347906)

[9.1 Diseño finanzas 202](#_Toc155347907)

[9.2 Producto recomendado 205](#_Toc155347908)

[9.3 Calculadora de afinidad 208](#_Toc155347909)

[9.4 Obtener consejos 209](#_Toc155347910)

[9.5 Consejos y Sugerencias 210](#_Toc155347911)

[9.6 Eventos 213](#_Toc155347912)

[9.7 Agregar Evento 215](#_Toc155347913)

[9.8 Update eventos 217](#_Toc155347914)

[9.9 Estadísticas de ahorro 220](#_Toc155347915)

[9.10 Valor del dólar 221](#_Toc155347916)

[9.11Valor de divisas 222](#_Toc155347917)

[9.12 Gráfica de divisas 224](#_Toc155347918)

[10 Containers 231](#_Toc155347919)

[10.1 Container Index 231](#_Toc155347920)

[10.2 Container Perfil 232](#_Toc155347921)

[10.3 Container Historial 233](#_Toc155347922)

[10.4 Container Planes de Ahorro 235](#_Toc155347923)

[10.5 Container Finanzas 236](#_Toc155347924)

[11 Notificaciones 237](#_Toc155347925)

[11.1 Aumento del dólar 237](#_Toc155347926)

[12 Configuración 238](#_Toc155347927)

[12.1 Configuraciones globales 238](#_Toc155347928)

[12.2 Cambio de tema 240](#_Toc155347929)

[12.3 Notificaciones y Manual de Usuario 241](#_Toc155347930)

[12.4 Cerrar sesión 242](#_Toc155347931)

[13 Decoder 256](#_Toc155347932)

[13.1 Frecuencia 256](#_Toc155347933)

[13.2 Label 257](#_Toc155347934)

[13.3 Date 257](#_Toc155347935)

[13.4 hayNet 261](#_Toc155347936)

[14 Widgets 262](#_Toc155347937)

[14.1 Redirect 262](#_Toc155347938)

[14.2 Reddirect provider 263](#_Toc155347939)

[14.3 Widget provider gasto 264](#_Toc155347940)

[14.4 Add gasto 266](#_Toc155347941)

[14.5 WidgetServiceGasto 268](#_Toc155347942)

[15 DAO’s 272](#_Toc155347943)

[15.1 UsuarioDao 272](#_Toc155347944)

[15.2 UserCacheDao 274](#_Toc155347945)

[15.3 MontoGrupoDao 274](#_Toc155347946)

[15.4 MontoDao 276](#_Toc155347947)

[15.5 LabelsDao 288](#_Toc155347948)

[15.6 IngresosGastosDao 289](#_Toc155347949)

[15.7 GruposDao 290](#_Toc155347950)

[15.8 EventosDao 292](#_Toc155347951)

[15.9 ConySugDao 293](#_Toc155347952)

[15.10 AssetsDao 294](#_Toc155347953)

[CAPÍTULO IX. RESULTADOS 295](#_Toc155347954)

[9.1 ANÁLISIS DE LOS RESULTADOS 295](#_Toc155347955)

[9.2 RESULTADO DLAS PRUEBAS O VALIDACIONES 296](#_Toc155347956)

[9.3 CONCLUSIONES 296](#_Toc155347957)

[9.4 REFERENCIAS 297](#_Toc155347958)

[9.5 ANEXO 299](#_Toc155347959)

[Manual de usuario 299](#_Toc155347960)

[XML 323](#_Toc155347961)

[Login 323](#_Toc155347962)

[Index 327](#_Toc155347963)

[Perfil 335](#_Toc155347964)

[Historial 341](#_Toc155347965)

[Planes de ahorro 345](#_Toc155347966)

[Finanzas 348](#_Toc155347967)

[Colores 354](#_Toc155347968)

# Índices Particulares

Imagen 1, Diagrama de Módulos.............................................................42

Imagen 2, Maquetado..............................................................................................43

Imagen 3, Login.......................................................................................................53

Imagen 4, Reestablecer Contraseña.......................................................................70

Imagen 5, Restablecimiento de Contraseña............................................................74

Imagen 6, Registrar Cuenta.....................................................................................77

Imagen 7, Índice......................................................................................................84

Imagen 8, Calendario..............................................................................................97

Imagen 9, Montos por Pagar.................................................................................107

Imagen 10, Añadir Monto......................................................................................116

Imagen 11, Favoritos.............................................................................................121

Imagen 12, Gastos................................................................................................124

Imagen 13, Ingresos..............................................................................................129

Imagen 14, Mandados...........................................................................................134

Imagen 15, Datos Usuario.....................................................................................148

Imagen 16, Grupos................................................................................................152

Imagen 17, Visualizar grupos................................................................................166

Imagen 18, Vista de grupo.....................................................................................169

Imagen 19, QR para compartir grupo....................................................................171

Imagen 20, Añadir monto de grupo.......................................................................174

Imagen 21, Miembros de grupo.............................................................................180

Imagen 22, Historial...............................................................................................186

Imagen 23, Etiquetas.............................................................................................188

Imagen 24, Historial con etiquetas........................................................................190

Imagen 25, Lista de historial..................................................................................198

Imagen 26, Papelera.............................................................................................198

Imagen 27, Favoritos.............................................................................................201

Imagen 28, Planes de ahorro................................................................................205

Imagen 29, Deudas...............................................................................................207

Imagen 30, Finanzas.............................................................................................213

Imagen 31, Producto Recomendado.....................................................................215

Imagen 32, Consejos.............................................................................................220

Imagen 33, Fechas especiales..............................................................................224

Imagen 34, Añadir Evento.....................................................................................226

Imagen 35, Gráficas..............................................................................................231

Imagen 36, Gráfico divisas....................................................................................235

Imagen 37, Notificaciones.....................................................................................248

Imagen 38, Configuración.....................................................................................249

Imagen 39, Widgets..............................................................................................278

Imagen 40, Login y Registro.................................................................................311

# INTRODUCCIÓN

El proyecto consiste en “Aplicación móvil para finanzas personales y ahorro”, la cual es un prototipo funcional de aplicación móvil diseñada para cualquier persona con conocimientos básicos de manejo de un celular y que tenga finanzas activas. La aplicación ofrece una experiencia amigable e intuitiva para que los usuarios puedan llevar cuenta de sus finanzas y tener un control más detallado de las mismas. También incluye una sección de gráficas para los usuarios más avanzados donde podrán tener proyecciones de ahorro e inversiones junto con información graficada de sus finanzas en general. El proyecto tiene como objetivo brindar una herramienta para las finanzas personales y ayudar a concientizar acerca del ahorro.

# Justificación

La cultura del ahorro y en general la educación financiera es un privilegio hoy en día en México ya que son pocas las personas que son educadas en esa área y aun menos son las que realmente lo llevan a cabo de manera satisfactoria. No obstante, prácticamente todos tenemos acceso a la tecnología móvil como lo son los celulares, por lo cual se llegó a la solución de desarrollar una herramienta financiera que pudiese ser usada por cualquier tipo de usuario.

Ahorrar es una habilidad que es complicada de desarrollar debido a la poca cultura financiera que hay en el país junto con las diversas situaciones económicas que tienen los mexicanos, en especial por la falta de presupuestos o gastos inesperados, sin mencionar los salarios los cuales no suelen ser suficientes para cubrir las necesidades y gastos de la población.

La aplicación tiene como objetivo principal el llevar cuenta de los gastos e ingresos del usuario mediante un seguimiento de días y fechas, apoyado con etiquetas y distintos recursos visuales con la finalidad de ser lo más efectiva posible, y mientras que el usuario aprende a usar la aplicación de manera óptima se le van recomendando distintas acciones que puede efectuar para mejorar su ahorro, concientizar la diversificación económica, entre otros.

# NOMENCLATURAS

**Kotlin: Lenguaje de programación móvil que se compila en Java.**

**Volley: Biblioteca de red para Android que simplifica las solicitudes HTTP.**

**Coil: Biblioteca de carga de imágenes para Android.**

**Módulo: Conjunto de funciones o componentes relacionados que forman una unidad.**

**Query: Instrucción para recuperar datos de una base de datos.**

**JSON: Formato ligero de intercambio de datos basado en texto.**

**Autómata Finito: Modelo matemático utilizado en teoría de la computación para representar sistemas de estados.**

**Room Framework: Biblioteca de persistencia para Android que simplifica el manejo de bases de datos SQLite.**

**API: Interfaz de Programación de Aplicaciones; conjunto de reglas que permite la comunicación entre software.**

**DAO: Objeto de Acceso a Datos; patrón de diseño que proporciona una interfaz para acceder a una fuente de datos.**

**XML: Lenguaje de marcado utilizado para almacenar y transportar datos de manera legible.**

**MVVM: Modelo-Vista-ViewModel; patrón arquitectónico que separa la lógica de presentación de la lógica de negocio en una aplicación.**

# CAPÍTULO I. EL AHORRO EN MÉXICO

## 1.1 LA ECONOMÍA DEL MEXICANO

En 2023, el mexicano gana $6223 MXN mensualmente como salario mínimo. Partiendo de este punto se puede empezar a desarrollar la situación actual, ya que el costo de vida mensualmente es de $21,231.66 MXN según el INEGI en base a la encuesta de ingresos y gastos de los hogares 2022, por lo que se puede empezar a denotar un poco la complejidad del asunto junto con la creciente inflación y encarecimiento de los servicios básicos.

Además, cabe denotar que, en México, un salario que pague “alto” o sea considerado bien remunerado es de $20,000 MXN o más mensualmente sin mencionar que dicho salario solo lo perciben 2 de cada 10 mexicanos, un “punto medio” en esta brecha salarial serían los $10, 402 MXN mensuales que percibe el mexicano en promedio, pero aun así no es ni la mitad del costo de vida mensual, por lo que ni teniendo una pareja o compañero que apoye económicamente se cubriría dicho coste.

## 1.2 EL MANEJO DEL DINERO

En México se hizo una encuesta dirigida por la CONDUSEF, la cual dio resultados un tanto alarmantes. 6 de cada 10 mexicanos (casi 42 millones) no llevan un registro de sus gastos mensuales, y los que si lo llevan el 64% lo hace de manera mental, por lo que muy probablemente terminen con gastos que no recordasen y a final de mes tengan que pedir prestado o recurrir a sus ahorros, en caso de que tengan ahorro. También se preguntó acerca del ahorro, lo cual se estimó que el 44% de los mexicanos que ahorran, lo hacen a través de mecanismos no formales siendo estos: guardar efectivo en casa 65%, participar en tandas 32% sin tener en cuenta lo riesgoso que puede llegar a ser.

## 1.3 FONDOS DE EMERGENCIA

De acuerdo con el estudio Estrés financiero: crisis silenciosa del bienestar de empleados y empresarios, realizado por Max Capital, el 47% de los mexicanos sufre de algún tipo de estrés por deudas. En la cultura actual de las tarjetas de crédito, es fácil minimizar la importancia de construir un fondo de emergencia. Si no tienes el efectivo, puedes tomarlo prestado, lo que lleva a plantear, ¿qué es un fondo de emergencia?

Es un ahorro en efectivo guardado de forma segura, preferiblemente en una cuenta que te de intereses. El dinero en esta cuenta puede ayudarte a enfrentar sucesos sorpresivos como la pérdida de empleo o una enfermedad grave. Pero también está disponible para “emergencias cotidianas”. En lugar de buscar una tarjeta de crédito cuando tu boiler se descompone o necesitas un nuevo juego de llantas, usa ese dinero para cubrir los gastos.

Se tiende a pensar que se debe de ahorrar una cantidad grande para que dicho fondo se pueda notar cuando realmente el principio es la constancia, no importa tanto que tanto dinero puedas apartar si no que día con día, semana con semana o mes con mes se haga el ahorro.

# CAPÍTULO II. LA INFLUENCIA DE LA TECNOLOGÍA COMO HERRAMIENTA

## 2.1 ORÍGENES DE LA TECNOLOGÍA PORTÁTIL

Los dispositivos móviles, comenzaron con la idea de poder llevar la capacidad de procesamiento de una computadora, la conexión a internet, la memoria y otros, pero con la portabilidad de tenerlo al alcance de la mano. Desde los comienzos del siglo XXI hasta los fines de la década de los 2000 hubo un avance exponencial en el desarrollo de las tablets, celulares, laptops, entre otros.

Una de las principales características es lo amigable del dispositivo por su pantalla táctil, y las demás funciones que ofrece como los acelerómetros, giroscopios, oxímetros, y otras aplicaciones que permiten tener diversas aplicaciones y herramientas dentro de un mismo dispositivo.

## 2.2 ACCESIBILIDAD Y USO EN MÉXICO

Un estudio realizado por compañías de marketing digital en 2023 obtuvo como resultado que la cantidad de conexiones celulares en México ascienden a los 123,500,000; dando como conclusión que el 96.5% de la población mexicana tiene un teléfono celular.

## 2.3 USO Y VICIO

El mexicano promedio utiliza el celular durante 4 horas con 56 minutos diarios, siendo usado principalmente como ocio, por lo que se designa México como “territorio Zuckerberg” ya que las 5 aplicaciones más descargadas y usadas en el país son pertenecientes a Meta, la empresa norteamericana.

# CAPÍTULO III. HERRAMIENTAS FINANCIERAS

## 3.1 TRANSPARENCIA ENTRE BANCOS

Como instituciones públicas, los bancos tienen la obligación moral de proveer con toda la información requerida por sus usuarios y parte de esta responsabilidad va con las herramientas financieras ya que muchos bancos suelen publicitarse como una entidad además de segura, que te apoyará para mejorar tu economía y ahorro, ¿Pero realmente es cierto que se obtienen dichas mejoras?

Actualmente se utilizan alrededor de 10 herramientas para la gestión financiera, de las cuales los bancos tradicionales en México cuentan con 1.9 como promedio, y los neobancos e instituciones financieras con 2.6 aproximadamente.

## 3.2 ¿DESINFORMACIÓN O DESIDIA?

Una encuesta realizada por la revista financiera Expansión, el 63% de los mexicanos no tiene idea de en que gasta su dinero, mientras que solo el 15% cuenta con un presupuesto o algo similar. Con estas alarmantes cifras se puede deducir que un sector muy grande de la población no es consciente de la importancia de tener un control sobre sus finanzas, pero la cuestión real aquí es si existe una brecha entre la desinformación o la desidia, ya que teniendo en cuenta todas las entrevistas, encuestas e información recolectada durante estos años, y sin tener en cuenta que prácticamente toda la población mexicana tiene acceso a la tecnología y por ende al internet, ¿Realmente existe tal desinformación, o somos victimas de nuestra propia desidia?

## 3.3 IMPACTO POST COVID-19

La revista Forbes informó que aproximadamente el 22% de los mexicanos tuvo una caída en sus ingresos durante la pandemia, el 63% tuvo que reducir sus gastos, siendo esta porción de la población la que principalmente requiere de asesoramiento financiero y como se pudo observar, existe una carencia muy grande.

Cabe resaltar el hecho que entre los horarios de pandemia y la poca accesibilidad, gran parte de los bancos e instituciones financieras se vieron en la necesidad de transferir prácticamente toda su atención junto con herramientas a las aplicaciones que ofrecen para hacer tus transacciones y demás, haciendo que cada vez sea más complicado el separar las herramientas de administración financiera de las herramientas de trasferencias, estados de cuenta y demás.

## 3.4 ¿EL PASADO SIGUE SIENDO EFECTIVO EN LA ACTUALIDAD?

Hace no tantos años, aún se acostumbraba a tener únicamente cajeros “de carne y hueso”, que las transacciones solo pudiesen ser hechas en sucursales y demás. Pero con los avances tecnológicos que se han tenido durante estas últimas 2 décadas todo ha cambiado, lo que nos lleva a la conclusión de ¿las herramientas financieras siguen siendo vigentes hoy en día?

Se puede dar por hecho que sí, si se tienen los ajustes necesarios, pero una de las principales contradictorias hoy en día sería la accesibilidad que dicta en todos nuestros entornos, ya que dichas herramientas no son necesariamente sencillas de comprender o utilizar para toda la población. De ahí se puede concluir que a pesar de que siguen funcionando, no son del todo adecuadas para la población actualmente.

# CAPÍTULO IV. E-RRAMIENTAS FINANCIERAS

## 4.1 PRACTICIDAD Y EFECTIVIDAD

Una de las principales comodidades que tiene el humano hoy en día es la accesibilidad y disponibilidad que le dan los smartphones y el internet, en este caso la facilidad que se tiene para consultar estados de cuenta, transferir y cobrar dinero, entre otras operaciones financieras.

En el caso de las herramientas financieras también se han visto compresas y simplificadas algunas de ellas por parte de bancos e instituciones financieras, todo con fin de tener un control efectivo de la economía de sus usuarios.

## 4.2 TIPOS DE HERRAMIENTAS

En la actualidad se pueden dividir las herramientas financieras en 4, siendo: “peer 2 peer” y pago de servicios, consulta de estado de cuenta, ahorro y búsqueda de ofertas y presupuestos.

Afortunadamente los bancos suelen dar acceso a varias de estas herramientas en una sola aplicación, siendo el principal uso las transferencias con el incremento del pago con tarjetas, así como el poder automatizar pagos de servicios y tener distintas ofertas por el uso de sus tarjetas.

Otra herramienta interesante son las herramientas de ahorro y búsqueda de ofertas, ya que mientras que las de ahorro buscan incentivar el ahorro y cumplir metas económicas, las herramientas de ofertas suelen ser efectivas a la hora de querer buscar el mejor precio en servicios como vuelos, reservas, o incluso en bienes como electrodomésticos.

En el caso de las herramientas de presupuestos se tiene una variedad de enfoques distintos y suelen ser muy accesibles, aun que muchas veces cobran su uso debido a la complejidad del sistema.

## 4.3 HERRAMIENTAS DE AHORRO Y PRESUPUESTOS

Las herramientas de ahorro y presupuestos pueden ser vistas de la mano, ya que parte de un buen presupuesto incluye el tener ahorro para distintas cosas, pero debido a que implican aspectos diferentes a los presupuestos suelen ser tomados de manera muy general o evitados. En la actualidad se utilizan 3 principales aplicaciones para tener un control y seguimiento de las finanzas personales, siendo estas: Monefy, Buddy y Mint. Las 3 tienen un enfoque muy diferente a pesar de que tienen la misma intención, mientras que una es muy interactiva y amigable, otra es excelente para compartir gastos y otra ideal si se tienen inversiones, en base a todas estas variaciones y aplicaciones se llega a la conclusión que sería ideal poder juntar la mayor cantidad de herramientas posibles en una sola aplicación.

# CAPÍTULO V. ENTORNO DE DESARROLLO DE SAVETRACK

## 5.1 DISPONIBILIDAD

Como ya se trató con anterioridad, hoy en día una de las principales características de muchos servicios es poder contar con ellos 24/7, por lo que una de las prioridades a la hora de planificar SaveTrack fue poder contar con acceso a la aplicación en todo momento y además a través de distintos dispositivos.

## 5.1.1 SERVIDORES COMO HOST

Como solución para poder tener toda la información disponible en todo momento fue el uso de un servidor como Host, de manera que la información de los usuarios esta respaldada, además que permite tener acceso a la información desde distintos dispositivos y facilita la conexión con los servicios conectados para el desarrollo de la aplicación.

Una de las características que más se busca en un servidor fuera de la capacidad de almacenamiento y la disponibilidad es la capacidad de transferencia de información, ya que se manejan una gran cantidad de datos los cuales entran y salen del sistema y el servidor, por lo que un flujo eficiente representa un funcionamiento adecuado de la aplicación.

## 5.1.2 API’S

La aplicación cuenta con conexión a múltiples API’s para poder hacer consultas a internet y distintos servicios para obtener información relevante para el sistema, las API’s utilizadas son: rest API, esta se utiliza para poder permitir una comunicación exitosa entre el servidor y la aplicación, por lo que es vital que la API tenga bien definidos sus parámetros, verificaciones y demás. Mercado Libre es otra de las API’s que se utilizan para el desarrollo de la aplicación, esta permite el tener productos recomendados los cuales se basan en un parámetro de afinidad con los gastos del usuario de manera que siempre se tendrá un producto que sea relevante para el usuario. Currency API es la encargada de tener en todo momento el precio de las divisas seleccionadas respecto al peso, por lo que el usuario puede tener en mente inversiones en divisas extranjeras o ahorro en distintas divisas.

## 5.2 JAVA VS KOTLIN

Desarrollar cualquier proyecto es un proceso complejo y un aspecto muy importante para tener en cuenta es el lenguaje para utilizarse para desarrollarlo, ya que como todo tiene sus pros y contras, pero en el caso del desarrollo móvil se busca que tenga el alcance necesario para poder cumplir con los requerimientos de la aplicación.

Basándose en estos puntos, se puede asumir que es necesario elegir un lenguaje que permita tener una gran transferencia de datos, accesibilidad y permita tener una escalabilidad en código, por lo que se tienen como opción Java y Kotlin como principales contendientes.

Java se caracteriza por ser un monstruo en la industria por su desarrollo durante todos estos años, contando con un gran soporte, una gran cantidad de desarrolladores y comunidad enorme, sin tener en cuenta que tiene una gran variedad de versiones y cuenta con su propio sistema de compilado, motor gráfico y gran accesibilidad a escalabilidad gracias a los applets.

Por otro lado, Kotlin es un lenguaje el cual durante la última década se ha visto muy fuerte en su desarrollo y crecimiento, además que nace para buscar saciar las necesidades que Java en su momento no conseguía.

Basándose en las necesidades del sistema, cuenta con una simplicidad de código muy práctica, está integrada con Android Studio lo que reduce el inconveniente de versiones en gran medida y uno de los principales factores son las corrutinas, ya que la aplicación al tener una gran entrada y salida de datos necesita poder administrar correctamente toda la información que recibe y envía.

Teniendo en cuenta todos estos aspectos y otros menos relevantes como lo son la curva de aprendizaje y otros detalles, poniendo frente a frente a Java contra Kotlin, se decidió que usar Kotlin sería lo más efectivo, así como eficiente hablando en código e implementación.

## 5.3 VERSATILIDAD EN KOTLIN

Tocando más a fondo las cualidades de Kotlin, el hecho que tenga una versatilidad tan grande es un aspecto muy relevante para tener en cuenta, ya que puede ser usado en sistemas móviles, web, servidores, incrustados, etc. Por lo que la amplitud del lenguaje es una característica muy buena acerca del lenguaje. Otro aspecto muy relevante es la capacidad de desarrollo multiplataforma, lo que permitirá que en un futuro la aplicación pueda ser utilizada en otros sistemas operativos, además que no está peleada con Java permitiendo interoperabilidad del lenguaje nativo, así como uso de otros frameworks como Angular, JQuery, React, etc.

# CAPÍTULO VI. LIMITACIONES

## 6.1 DESIDIA Y MAL USO

Una de las problemáticas que se pueden presentar durante el uso de la aplicación son todos los detalles que el desarrollador no tiene control como lo es el usuario, ya que a pesar de que se busque evitar cualquier confusión o inconveniente por parte del sistema, el sistema requiere del usuario para poder trabajar por lo que en caso de una aplicación de finanzas, si el usuario cae en desidia y no introduce todos los valores requeridos, exagera en montos, y hace mal uso en general sobre la aplicación, va a resultar en un mal funcionamiento de esta, y por ende una mala experiencia ya que no va a cumplir con su objetivo la aplicación.

## 6.2 CREDENCIALES Y PERMISOS

Una desventaja muy grande de la aplicación frente a otras aplicaciones ya establecidas en el mercado, es la conectividad con bancos y entidades financieras, ya que no se cuentan con las credenciales requeridas para dicho acceso, además que los permisos por parte del gobierno también son otro inconveniente para poder solicitar dicho acceso, aun que se busque que a futuro la aplicación tenga desarrollo en dicho ámbito, es un hecho que actualmente no se cuenta con dichos accesos por lo que el usuario va a tener que consultar sus estados de cuenta de manera independiente, y en general algunos valores no se pueden automatizar como se desearía.

## 6.3 AMPLITUD DEL SISTEMA

Anteriormente se trató el tema de la amplitud que suelen abarcar las aplicaciones de este tipo, ya que mientras mayor sea el enfoque del sistema, es mucho más sencillo tener un funcionamiento lo más eficiente posible, en el caso de SaveTrack se busca abarcar una gran parte de las finanzas personales del mexicano por lo que aun que se busque tener el mejor rendimiento posible, existe la posibilidad de que el usuario no ingrese la información adecuadamente, o que la aplicación al depender de varios aspectos internos, si llegase a tener un dato comprometedor llegue a generar un problema mayor en cuanto a la información a procesar y por ende el resultado no será el adecuado.

# CAPÍTULO VII. PLANIFICACIÓN Y VISIÓN DE SAVETRACK

## 7.1 ANÁLISIS DE REQUERIMIENTOS

A continuación, se detallarán todas las características de la aplicación junto con los requerimientos funcionales y no funcionales de la misma.

## 7.1.1 CARACTERÍSTICAS GENERALES

El prototipo es una aplicación para dispositivos móviles Android, los cuales deben de contar con acceso a internet para el correcto funcionamiento de la aplicación.

La aplicación cuenta con 3 apartados principales los cuales son: grupos, gastos e ingresos y la sección de finanzas e inversiones. También cabe resaltar que la aplicación busca ser utilizable por cualquier persona con conocimiento básico del uso de un celular junto además que tenga un flujo económico mínimo de manera que se pueda tener contabilidad de este.

## 7.1.2 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

### Requerimientos Funcionales

|  |  |
| --- | --- |
| 1 | La app cuenta con un login al ingresar a la pantalla principal |
| 2 | El campo login solo se requiere al registrar al usuario o iniciar sesión |
| 3 | Si ya existe un usuario activo la autenticación será una contraseña |
| 4 | La pantalla principal tiene dos opciones para hacer login |
| 5 | La pantalla principal cuenta con la opción de: "registrar usuario nuevo" |
| 6 | La pantalla principal cuenta con la opción de: "recuperar contraseña" |
| 7 | La app tiene 5 vistas principales, cada una con su propio ícono |
| 8 | La primera vista es el perfil, muestra la información del usuario |
| 9 | La segunda vista es el historial, muestra un calendario con gastos |
| 10 | La tercera vista es el index, muestra una vista general de la situación económica |
| 11 | La cuarta vista son los planes de ahorro, muestra opciones de ahorro para el usuario |
| 12 | La quinta vista son finanzas, muestra datos que podrían interesarle al usuario |
| 13 | Habrá una barra inferior con los cinco íconos para un desplazamiento eficaz |
| 14 | Todas las vistas tendrán subvistas a las que se accede con botones |
| 15 | Todas las subvistas tendrán encabezados |
| 16 | Todas las subvistas tendrán un botón para volver a la pantalla previa |
| 17 | Todas las vistas y subvistas tendrán un botón de configuración |
| 18 | La barra inferior se mostrará en todas las vistas y subvistas |
| 19 | Al ingresar a la app se iniciará en la vista del índex |
| 20 | El index cuenta con el gráfico de pastel el cual muestra los gastos categorizados con colores |
| 21 | El index muestra en un ícono de calendario el día actual |
| 22 | El index muestra en una "sidebar" las fechas de pago más proximas en un orden de prioridad |
| 23 | El index muestra un ícono de tubo de ensayo el cual será reactivo y muestra la "puntuación" de ahorro del usuario |
| 24 | El index muestra un ícono con un más, que agrega un ingreso |
| 25 | El index muestra un ícono con un menos, que agrega un gasto |
| 26 | En el centro del gráfico de pastel en el index se muestra una caja con los ingresos brutos |
| 27 | En el centro del gráfico de pastel en el index se muestra una caja con los gastos totales |
| 28 | En el centro del gráfico de pastel en el index se muestra una caja con los ingresos netos |
| 29 | Habrá una subvista que mostrará los ingresos general |
| 30 | Se podrá acceder a la vista de ingresos presionando en ingresos brutos |
| 31 | Se podrá acceder a la vista de ingresos presionando en ingresos netos |
| 32 | En la vista de ingresos se mostrará una caja con el nombre del ingreso |
| 33 | En la vista de ingresos se mostrará la fecha de pago en la caja con el nombre del ingreso |
| 34 | En la vista de ingresos se mostrará un botón para hacer trigger a un ingreso de tipo venta |
| 35 | En la vista de ingresos se mostrará el monto del ingreso a la derecha de cada caja |
| 36 | En la vista de ingresos habrá un botón para agregar un ingreso, igual que si fuera desde el index |
| 37 | En la vista de ingresos se mostrará el total |
| 38 | Habrá una subvista de gastos general |
| 39 | En la vista de gastos se mostrará un número con la etiqueta a la que pertenece el gasto |
| 40 | En la vista de gastos aparecerá el monto del gasto |
| 41 | Los gastos mostrados que aparecerán serán solamente los realizados en la semana |
| 42 | Existirá la subvista de agregar monto |
| 43 | En la subvista de agregar monto existirá un recuadro para agregar la etiqueta |
| 44 | En la subvista de agregar monto existirá un recuadro para agregar el nombre |
| 45 | En la subvista de agregar monto existirá un recuadro para agregar la cantidad del monto |
| 46 | En la subvista de agregar monto existirá un toggle switch para especificar si es un gasto o un ingreso |
| 47 | En la subvista de agregar monto existirá un recuadro para agregar la frecuencia del monto (si es único o rutinario) |
| 48 | Existirá una subvista con cosas por pagar a corto plazo |
| 49 | En la vista de por pagar, aparecerán los pagos como checkboxes |
| 50 | Las checkboxes de por pagar redirigirán a agregar monto (para especificar el coste) |
| 51 | En la subvista de por pagar habrá otra subvista de mandados |
| 52 | En la subvista de mandados aparecerán las rutinas (supermercado, por ejemplo) |
| 53 | En todas las pantallas de agregar y eliminar existirá un botón de confirmar |
| 54 | El perfil cuenta con el campo de foto de perfil el cual se ubica en la parte superior izquierda |
| 55 | El perfil muestra en la parte izquierda la información personal del usuario |
| 56 | El perfil muestra los grupos del usuario |
| 57 | El perfil muestra el balance del usuario |
| 58 | El perfil muestra los días que lleva ahorrando el usuario |
| 59 | El perfil tiene un icono de "engranaje" para acceder a la configuración del perfil |
| 60 | La configuración del perfil cuenta con un icono de "lápiz" al lado de cada campo para modificar el valor |
| 61 | La configuración de perfil cuenta con un icono de foto de perfil para moder modificar la misma |
| 62 | La configuración de perfil cuenta con un campo para "username", "edad" y "sitiuación laboral" |
| 63 | La configuración de perfil cuenta con un icono de "checkmark" para confirmar los cambios hechos en el perfil |
| 64 | En la sección de grupos se podrá administrar gastos en conjunto con otras personas |
| 65 | La sección de grupos tendrá utilidades de adición de estos |
| 66 | La sección de grupos tendrá opción de edición de estos |
| 67 | La sección de grupos tendrá opción de eliminación de estos |
| 68 | La sección de grupos tendrá opciones de agregar y eliminar personas |
| 69 | La sección de grupos tendrá opción de unirse a través de un QR |
| 70 | La sección de grupos tendrá opciones de limitar si un grupo es temporal o fijo |
| 71 | La sección de grupos tendrá opción para modificar su nombre |
| 72 | El historial cuenta con una etiqueta desplegable en la sección superior izquierda para seleccionar el mes a visualizar |
| 73 | El historial cuenta con un calendario el cual admitirá etiquetas para denotar las fechas de pago |
| 74 | El historial cuenta con una sección de etiquetas para poder agregar y eliminarlas |
| 75 | El historial cuenta con una sección de etiquetas personalizadas las que podrá ingresar el usuario manualmente |
| 76 | La sección de personalización se detalla en fecha, tipo y se podrá automatizar para gastos recursivos |
| 77 | La seccion de personalización contará con una "checkmark" para guardar las etiquetas nuevas |
| 78 | La sección de línea de tiempo contará con una línea cronologica con años para una vista a mayor escala |
| 79 | La sección de línea de tiempo contará con una vista de los eliminados y guardados |
| 80 | La vista de eliminados será igual al calendario, pero muestra solamente datos eliminados |
| 81 | La vista de guardados será igual al calendario, pero mostrando solamente los guardados y su fecha exacta |
| 82 | En la configuración se podrán editar los colores de la etiquetas |
| 83 | En la configuración se podrán editar la personalización de los detalles de las etiquetas |
| 84 | La vista de planes de ahorro contendrá un módulo para consultar el valor actual de la moneda |
| 85 | La vista de planes de ahorro contendrá un módulo para verificar la cantidad de días que lleva ahorrando |
| 86 | La vista de planes de ahorro contendrá un módulo para consultar cuánto dinero tiene ahorrado en total |
| 87 | La vista de planes de ahorro contendrá un módulo para consultar sus deudas grandes o productos en periodo de pago |
| 88 | El módulo de deudas tendrá la fecha límite de pago y el monto a pagar para dicha fecha |
| 89 | El módulo de deudas redirigirá a una subvista de proceso de pago |
| 90 | La subvista de en proceso de pago contendrá una caja para cada pago |
| 91 | En el recuadro de cada pago se mostrará la etiqueta del monto |
| 92 | En el recuadro de cada pago se mostrará el nombre del monto |
| 93 | En el recuadro de cada pago se mostrará la categoría del monto |
| 94 | En el recuadro de cada pago se mostrará la cantidad final del monto |
| 95 | En el recuadro de cada pago se mostrará un botón de eliminar |
| 96 | En el recuadro de cada pago se mostrará un botón de marcar como pagado |
| 97 | Al presionar en los recuadros se puede acceder a una edición de cada monto |
| 98 | La subvista de en proceso de pago se mostrará un botón para confirmar los cambios |
| 99 | La vista de finanzas contendrá un módulo para consejos de uso de la aplicación y sugerencias financieras |
| 100 | La vista de finanzas contendrá un módulo para estadísticas graficadas del usuario |
| 101 | La vista de finanzas contendrá un módulo de producto recomendado, el cual se utilizará como un anuncio |
| 102 | La vista de finanzas contendrá un módulo de fechas relevantes para la economía del usuario (no historial, fechas a futuro) |
| 103 | La subvista de consejos tendrá una caja para cada sugerencia |
| 104 | Las cajas de sugerencias en la subvista de consejos tendrán un botón de aceptar |
| 105 | Las cajas de sugerencias en la subvista de consejos tendrán un botón de rechazar, y no se volverá a sugerir |
| 106 | Si se acepta un consejo, se redirigirá a la pantalla en donde se encuentre la configuración para hacer el ajuste |
| 107 | La subvista de estadísticas contendrá una gráfica que contendrá las categorías en las que se tiene ahorrado en porcentaje |
| 108 | Al presionar en un área de la gráfica, redirigirá a una gráfica más específica que muestre la fluctuación monetaria |
| 109 | Existirá un área de ahorro estático en la subvista de estadísticas |
| 110 | Existirá un área de ahorro de inversión en la subvista de estadísticas |
| 111 | Existirá un área de ahorro en otra moneda en la subvista de estadísticas |
| 112 | Para el ahorro estático se tendrá en cuenta la fluctuación de la moneda con relación al dólar |
| 113 | Para el ahorro de inversión se tendrán en cuenta estadísticas ingresables |
| 114 | En el caso de ingreso de datos de inversión, se solicitará el porcentaje de crecimiento para graficar el aumento |
| 115 | Para el ahorro de moneda se graficará la fluctuación entre el peso, el dolar y otra moneda a elegir |
| 116 | En el caso de ahorro de divisa extranjera, se utilizará una API de finanzas de google |
| 117 | Las gráficas contarán con un porcentaje que indique el crecimiento obtenido en un plazo personalizable |
| 118 | Se podrá consultar el porcentaje de cada fecha a plazo personalizable |
| 119 | En el caso del ahorro de moneda, habrá una tabla |
| 120 | La tabla del ahorro de moneda contendrá el símbolo de la divisa |
| 121 | La tabla del ahorro de moneda contendrá el valor actual en MXN de la divisa |
| 122 | La tabla del ahorro de moneda contendrá el crecimiento o decrecimiento de la divisa según el plazo personalizable |
| 123 | El módulo de producto recomendado tendrá un botón de ver más |
| 124 | El módulo de producto recomendado tendrá un botón de ver menos |
| 125 | El módulo de producto recomendado tendrá un botón de no mostrar este producto |
| 126 | El módulo de producto recomendado tendrá dos porcentajes de afinidad |
| 127 | El primero de los porcentajes de afinidad utilizará los productos recientemente comprados y se sugerirá a partir de estos |
| 128 | El segundo de los porcentajes de afinidad se basará en el balance económico actual del usuario |
| 129 | Si se presiona en el botón "ver más", se redirigirá a una página externa |
| 130 | Si se presiona en el módulo de fechas relevantes redirigirá a un calendario |
| 131 | El calendario será un CRUD |
| 132 | Cada fecha podrá programarse con eventos repetibles, igual que el historial |
| 133 | Se podrán personalizar fechas a modo de cumpleaños o aniversarios |
| 134 | Fechas como navidad o año nuevo vendrán por defecto para que el usuario tenga la opción de hacer planes de regalos |
| 135 | El usuario recibirá una notificación cuando se detecte una fluctuación fuerte en la moneda |
| 136 | El usuario recibirá una notificación una semana antes de las fechas relevantes |
| 137 | El usuario recibirá una notificación los dias de fecha relevante |
| 138 | Existirán widgets basados en la vista de index |
| 139 | El usuario tendrá la opción de crear un widget en el que se presione y agregue al instante un monto de ingreso |
| 140 | El usuario tendrá la opción de crear un widget en el que se presione y agregue al instante un monto de resta |
| 141 | El usuario tendrá la opción de crear un widget en el que se presione y abra la subvista de agregar monto personalizado |
| 142 | La vista de configuración es accesible desde todas las pantallas |
| 143 | La vista de configuración tendrá una opción para cambiar el modo de oscuro a claro |
| 144 | La vista de configuración tendrá opciones para solicitar ayuda |
| 145 | Si se solicita ayuda, se redirigirá a un manual |
| 146 | En la vista de configuración aparecerá el contacto de los programadores |
| 147 | En la vista de configuración tendrá un switch para habilitar o deshabilitar las notificaciones |

### Requerimientos no Funcionales

|  |  |
| --- | --- |
| 1 | La app será desarrollada en Android Studio |
| 2 | La app será desarrollada con los lenguajes de Kotlin para las interfaces y Python para las API's |
| 3 | Todos los datos que maneje la app serán almacenados en una base de datos relacional la cual usará SQLite |
| 4 | La base de datos será hosteada mediante un servidor personal (encontrar cual app usar) |
| 5 | Se usarán colores pálidos de tonos azules y fuentes sólidas para el login |
| 6 | Se usará la fuente Verdana para todos los textos y se usará en negritas para títulos |
| 7 | Se mostrará el logo en la parte superior central al hacer login |
| 8 | Para números, se utilizará la fuente Century Gothic |
| 9 | Los campos de texto para ingresar la información de login será transparente |
| 10 | Las opciones de registrar usuario nuevo y recuperar contraseña serán de color azul |
| 11 | Las opciones de registrar usuario nuevo y recuperar contraseña redirigirán a sus respectivas vistas tras ser presionados |
| 12 | La app comprobará las credenciales de login en la BD |
| 13 | La app mostrará los íconos en un tamaño de 24dp por 24dp |
| 14 | La app mostrará los íconos con colores azules |
| 15 | El ícono de la vista en la que se esté, se marcará en verde en vez de azul |
| 16 | La caja que contendrá los iconos será de fondo gris |
| 17 | La caja que contendrá los iconos será de borde gris oscuro |
| 18 | Los iconos que se muestren serán "minimalistas" para hacerlos lo mas intuitivos posible |
| 19 | El ícono del perfil se asemeja a una persona |
| 20 | El ícono del historial se asemeja a un reloj |
| 21 | El ícono del índex se asemeja a una casa |
| 22 | El ícono de planes de ahorro se asemeja a una gema |
| 23 | El ícono de finanzas se asemeja a un símbolo de dinero |
| 24 | Las vistas serán accesibles mediante íconos intuitivos |
| 25 | Los iconos serán mostrados en la parte inferior central |
| 26 | Los iconos serán contenidos dentro de una caja |
| 27 | Tras presionar el ícono correspondiente, se redirigirá al usuario a la vista correspondiente |
| 28 | Cuando se desplace entre pantallas, se tendrá un efecto de desplazamiento suave |
| 29 | Los textos de los encabezados tendrán fuente de 48dp |
| 30 | Los textos de los encabezados tendrán el color común de la vista en la que se encuentren |
| 31 | El botón de volver a la pantalla previa será una flecha del color dominante de la pantalla |
| 32 | El botón de configuración tendrá forma de engranaje |
| 33 | Cuando se esté en una vista, el ícono de la vista se agrandará |
| 34 | Cuando se esté en una vista, el ícono de la vista brillará |
| 35 | Cada vista tendrá un color dominante que compartirá con sus subvistas |
| 36 | Cada vista tendrá tres colores basados en el dominante, normal, pálido y oscuro |
| 37 | Los fondos de todas las pantallas serán de color pálido |
| 38 | Los textos serán de color oscuro para poder discernirse en el fondo pálido |
| 39 | Las condiciones de colores pálidos y oscuros se invertirán en toda la aplicación si se activa el modo oscuro |
| 40 | El index es de colores morados |
| 41 | El gráfico de pastel mostrará cada categoría en distintos colores, haciendo un conjunto de rojo, naranja, amarillo, verde y azul |
| 42 | La app obtendrá del sistema operativo la fecha actual para mostrarlo en el calendario |
| 43 | La app mostrará con colores cada vez más intensos a manera progresiva las fechas de pago en el "sidebar" |
| 44 | El ícono del calendario será de colores violetas |
| 45 | El ícono del tubo de ensayo tendrá bordes violetas |
| 46 | El ícono del tubo de ensayo tendrá fondo transparente |
| 47 | El ícono del tubo de ensayo tendrá relleno lila |
| 48 | El ícono de más tendrá bordes azul cobalto |
| 49 | El ícono de más tendrá fondo azul claro |
| 50 | El ícono de más tendrá bordes rojizos |
| 51 | El ícono de más tendrá fondo rojizo claro |
| 52 | El ícono del calendario será cuadrado |
| 53 | Los íconos de agregar serán circulares |
| 54 | La sidebar será de colores de prioridades |
| 55 | La sidebar tendrá borde color violeta |
| 56 | La sidebar será rectangular |
| 57 | La sidebar tendrá checkboxes |
| 58 | En el centro del gráfico de pastel en el index, los ingresos brutos se marcarán en azul claro |
| 59 | En el centro del gráfico de pastel en el index, los gastos totales se marcarán en rojo |
| 60 | En el centro del gráfico de pastel en el index, los ingresos netos se marcarán en amarillo |
| 61 | Las cajas donde se indica el nombre de ingreso y fecha son cuadradas |
| 62 | Las cajas donde se indica el nombre de ingreso y fecha son de bordes violetas |
| 63 | Las cajas donde se indica el nombre de ingreso y fecha son de fondo lila |
| 64 | Los botones de tipo venta contendrán el nombre del producto y su precio dentro de sí |
| 65 | Los botones de tipo venta serán circulares |
| 66 | El botón de agregar ingreso será de bordes azul cobalto |
| 67 | El botón de agregar ingreso será de fondo azul claro |
| 68 | La caja con los ingresos brutos tendrá forma rectangular |
| 69 | La caja con los ingresos brutos tendrá bordes suaves |
| 70 | El borde de la caja con los ingresos brutos tendrá color violeta oscuro |
| 71 | El fondo de la caja con los ingresos brutos será color lila |
| 72 | Las cajas contenedoras de las etiquetas de los gastos tendrán forma de triángulo invertido |
| 73 | Las cajas contenedoras de las etiquetas de los gastos tendrán borde violeta oscuro |
| 74 | Las cajas contenedoras de las etiquetas de los gastos tendrá fondo violeta claro |
| 75 | Se accederá a la subvista de por pagar desde la sidebar |
| 76 | Se accederá a la subvista de mandados desde la subvista de por pagar |
| 77 | En la vista de por pagar, las checkboxes serán circulares |
| 78 | En la vista de por pagar, el borde de las checkboxes serán del color de la etiqueta |
| 79 | En la vista de por pagar, el fondo de las checkboxes serán del color de la etiqueta, pero más tenue |
| 80 | En sidebar en el index si una checkbox es de una deuda, aparecerá como primera y de color negro y rojo |
| 81 | En la vista de por pagar, si una checkbox es de una deuda, aparecerá como primera y de color negro y rojo |
| 82 | Las checkboxes de por pagar redirigirán a agregar monto (para especificar el coste) |
| 83 | En la subvista de agregar monto, habrá un ícono de una etiqueta a lado del campo etiqueta |
| 84 | En la subvista de agregar monto, habrá un ícono de un lapiz a lado del campo nombre |
| 85 | En la subvista de agregar monto, habrá un toggle switch que cambie de rojo a verde a lado del campo monto |
| 86 | En la subvista de agregar monto, habrá un menú colgante con los días a determinar de repetición de gasto |
| 87 | Los bordes de los íconos serán color violeta |
| 88 | Los fondos de los íconos serán color lila |
| 89 | En la subvista de por pagar habrá otra subvista de mandados |
| 90 | En la vista de mandados, las checkboxes serán circulares |
| 91 | En la vista de mandados, el borde de las checkboxes serán del color base |
| 92 | En la vista de mandados, el fondo de las checkboxes será del color de la etiqueta, pero más tenue |
| 93 | En la subvista de mandados, las rutinas siguen un ciclo propio de gasto |
| 94 | En la subvista de mandados, las rutinas se pueden editar y eliminar |
| 95 | En la subvista de mandados, las rutinas tendrán monto predeterminado por el usuario |
| 96 | En la subvista de mandados se agregará un botón de gastos hormiga a determinar siempre |
| 97 | En todas las pantallas de agregar y eliminar, el botón de confirmar tendrá color verde |
| 98 | En todas las pantallas de agregar y eliminar, el botón de confirmar tendrá forma circular |
| 99 | En todas las pantallas de agregar y eliminar, el botón de confirmar tendrá fondo verde claro |
| 100 | En todas las pantallas de agregar y eliminar, el botón de confirmar tendrá una palomita |
| 101 | El perfil usa colores de tonalidades azules |
| 102 | La foto de perfil podrá ser tomada desde la cámara o elegirse desde la galería del dispositivo |
| 103 | Si se hace click en el ícono del balance se redirigirá a la vista de planes de ahorro |
| 104 | Si se hace click en el ícono de los días que se lleva ahorrando se redirigirá a la vista de historial |
| 105 | El borde de la foto de perfil será azul oscuro de grosor medio |
| 106 | La caja de información del usuario tendrá fondo azul claro |
| 107 | La caja de información del usuario tendrá borde azul oscuro |
| 108 | La caja de grupos tendrá fondo azul claro |
| 109 | La caja de grupos tendrá borde azul oscuro |
| 110 | El fondo del ícono de días ahorrados será de un color amarillento |
| 111 | El borde del ícono de días ahorrados será de un color anaranjado |
| 112 | El ícono de balance será de colores rojizos |
| 113 | En la sección de edición, las cajas serán transparentes |
| 114 | En la sección de edición, los bordes de las cajas serán azul claro |
| 115 | Los íconos de lápices serán de color azul cobalto |
| 116 | Los íconos de menú colgante serán azul cobalto |
| 117 | Los íconos de persona serán de color azul cobalto |
| 118 | En la sección de grupos, la caja de adición y edición de grupos tendrá bordes color azul oscuro |
| 119 | En la sección de grupos, la caja de adición y edición de grupos tendrá fondo color azul claro |
| 120 | En la sección de grupos, el botón de agregar persona tendrá borde azul oscuro |
| 121 | En la sección de grupos, el botón de agregar persona tendrá fondo azul claro |
| 122 | En la sección de grupos, el botón de aceptar tendrá bordes color azul cobalto |
| 123 | En la sección de grupos, el botón de aceptar tendrá fondo azul claro |
| 124 | En la sección de grupos, el botón de cancelar tendrá bordes color rojizo |
| 125 | En la sección de grupos, el botón de cancelar tendrá fondo rojizo claro |
| 126 | El historial usa colores anaranjados y amarillos |
| 127 | El botón para entrar al apartado de etiquetas tendrá un borde anaranjado |
| 128 | El botón para entrar al apartado de etiquetas tendrá un fondo amarillento |
| 129 | En el apartado de etiquetas, la caja contenedora tendrá bordes grisáceos |
| 130 | En el apartado de etiquetas, la caja contenedora tendrá fondo del color que sea la etiqueta personalizable |
| 131 | El calendario tendrá bordes anaranjados |
| 132 | El calendario tendrá fondo amarillo claro |
| 133 | Las fechas marcadas en el calendario tendrán fondo amarillo |
| 134 | Las fechas importantes en el calendario tendrán fondo amarillo oscuro |
| 135 | Las fechas importantes en el calendario tendrán fondo color rojo oscuro |
| 136 | Cuando se seleccione una fecha, la caja del dato tendrá bordes anaranjados |
| 137 | Cuando se seleccione una fecha, la caja del dato tendrá fondo amarillo |
| 138 | Cuando se seleccione una fecha, el botón cancelar tendrá bordes rojizos |
| 139 | Cuando se seleccione una fecha, el botón cancelar tendrá fondo rojizo claro |
| 140 | Cuando se seleccione una fecha, la etiqueta será del color seleccionado en el apartado de etiquetas |
| 141 | La línea de tiempo tendrá bordes anaranjado oscuro |
| 142 | La línea de tiempo tendrá fondo amarillo claro |
| 143 | Las marcas en la línea del tiempo serán color amarillo claro |
| 144 | El basurero que redirige al apartado de eliminados será de color gris oscuro |
| 145 | La vista de eliminados, siendo igual que la de calendario, tendrá bordes grises |
| 146 | La vista de eliminados, siendo igual que la de calendario, tendrá fondos grises claro |
| 147 | La vista de eliminados, siendo igual que la de calendario, sus fechas marcadas como borradas tendrán fondos |
| 148 | Las cajas de edición serán transparentes |
| 149 | Las cajas de edición tendrán bordes suaves de color gris claro |
| 150 | En la confirmación de edición, el botón de aceptar tendrá bordes color azul cobalto |
| 151 | En la confirmación de edición, el botón de aceptar tendrá fondo azul claro |
| 152 | En la confirmación de edición, el botón de cancelar tendrá bordes color rojizo |
| 153 | En la confirmación de edición, el botón de cancelar tendrá fondo rojizo claro |
| 154 | La vista de planes de ahorro será de colores verdes |
| 155 | El módulo para consultar el valor actual de la moneda tendrá forma de burbujas |
| 156 | El módulo para consultar el valor actual de la moneda será de bordes verdes oscuro |
| 157 | El módulo para consultar el valor actual de la moneda será de fondo verde claro |
| 158 | El módulo para consultar la cantidad de días ahorrando tendrá forma rectangular |
| 159 | El módulo para consultar la cantidad de días ahorrando tendrá forma rectangular |
| 160 | El módulo para consultar la cantidad de días ahorrando será de bordes verde oscuro |
| 161 | El módulo para consultar la cantidad de días ahorrando será de fondo verde claro |
| 162 | El módulo para consultar la cantidad de dinero ahorrado tendrá forma hexagonal |
| 163 | El módulo para consultar la cantidad de dinero ahorrado tendrá doble borde, ambos verde oscuro |
| 164 | El módulo para consultar la cantidad de dinero ahorrado tendrá fondo verde claro |
| 165 | El módulo para consultar las deudas y pagos en proceso, tendrá forma rectangular a modo de barra horizontal |
| 166 | El módulo para consultar las deudas y pagos en proceso, dará forma cuadrada a cada pago |
| 167 | Los recuadros dentro del módulo de pagos en proceso, contendrán un campo para el nombre del pago |
| 168 | Los recuadros dentro del módulo de pagos en proceso, contendrán un campo para la fecha límite |
| 169 | Los recuadros dentro del módulo de pagos en proceso, contendrán un campo para el monto |
| 170 | Los recuadros de cada pago en la subvista tendrán forma rectangular |
| 171 | Los recuadros de cada pago en la subvista tendrán borde verde oscuro |
| 172 | Los recuadros de cada pago en la subvista tendrán borde verde claro |
| 173 | La etiqueta en el recuadro del monto se mostrará del color y número de la respectiva etiqueta |
| 174 | El botón eliminar tendrá forma circular |
| 175 | El botón eliminar tendrá borde rojo |
| 176 | El botón eliminar tendrá fondo rojo claro |
| 177 | El botón eliminar tendrá una equis roja |
| 178 | El botón de marcar como pagado tendrá forma circular |
| 179 | El botón de marcar como pagado tendrá borde verde |
| 180 | El botón de marcar como pagado tendrá fondo verde claro |
| 181 | El botón de marcar como pagado tendrá una palomita |
| 182 | El botón confirmar tendrá las mismas características que el botón de marcar como pagado |
| 183 | La vista de finanzas será de colores blanquecinos y negros |
| 184 | El módulo de consejos y sugerencias tendrá fondo gris claro |
| 185 | El módulo de consejos y sugerencias tendrá borde gris oscuro |
| 186 | El módulo de consejos y sugerencias tendrá forma cuadrada, de 92dp por lado |
| 187 | Las cajas de sugerencias tendrán fondo gris claro |
| 188 | Los bordes de las cajas serán gris oscuro |
| 189 | Las cajas de sugerencias serán de forma rectangular, de 250dp por 30dp (adaptable según el largo del texto) |
| 190 | El botón de rechazar en la caja de sugerencias tendrá fondo rojizo |
| 191 | El botón de rechazar en la caja de sugerencias tendrá bordes rojos oscuros |
| 192 | El botón de rechazar en la caja de sugerencias contendrá una equis |
| 193 | El botón de aceptar en la caja de sugerencias tendrá fondo verde claro |
| 194 | El botón de aceptar en la caja de sugerencias tendrá bordes verdes oscuros |
| 195 | El botón de aceptar en la caja de sugerencias contendrá una flecha volteando hacia la derecha |
| 196 | El módulo de estadísticas graficadas del usuario tendrá fondo gris claro |
| 197 | El módulo de estadísticas graficadas del usuario tendrá borde gris oscuro |
| 198 | El módulo de estadísticas graficadas del usuario tendrá forma cuadrada, de 62dp por lado |
| 199 | En la subvista de estadísticas graficadas, la gráfica de categorías tendrá forma triangular |
| 200 | Se podrá personalizar la forma de la gráfica |
| # | RNF |
| 201 | En las subvistas de las gráficas de progreso económico, las gráficas diferiraán entre colores vivos para discernir entre una y otra |
| 202 | El porcentaje que indique el crecimiento o decrecimiento estará contenido dentro de una burbuja |
| 203 | La burbuja será de bordes blanquecinos |
| 204 | La tabla en el apartado de moneda tendrá fondos de celda color gris claro |
| 205 | La tabla en el apartado de moneda tendrá bordes gris oscuro |
| 206 | El módulo de producto recomendado tendrá fondo gris claro |
| 207 | El módulo de producto recomendado tendrá borde gris oscuro |
| 208 | El módulo de producto recomendado tendrá forma cuadrada, de 72dp por lado |
| 209 | El botón de ver más será un símbolo "+" |
| 210 | El botón de ver menos será un símbolo "-" |
| 211 | El botón de no mostrar producto será un símbolo "x" |
| 212 | Los porcentajes se mostrarán al exterior del módulo, del lado izquierdo |
| 213 | El módulo de fechas relevantes tendrá fondo gris claro |
| 214 | El módulo de fechas relevantes tendrá borde gris oscuro |
| 215 | El módulo de fechas relevantes tendrá forma cuadrada, de 98dp por lado |
| 216 | Las celdas del calendario serán de color gris claro |
| 217 | Los bordes del calendario serán gris oscuro |
| 218 | El botón de confirmar tendrá fondo verde claro |
| 219 | El botón de confirmar tendrá borde verde claro |
| 220 | Las flechas para alternar el mes y año tendrán bordes gris claro |
| 221 | El menú colgante para la repetición será color gris claro |
| 222 | Los widgets tendrán el diseño y colores de su contraparte en la aplicación |

## 7.2 MODELADO GRÁFICO

El primer diagrama de módulos de la aplicación, el cual detalla la entrada y salida de información, así como el destino y remitente

Imagen 1, diagrama de módulos

**Tabla

Descripción generada automáticamente**

Maquetado principal, en el cual se basó el desarrollo de la aplicación y se definió el flujo del usuario al utilizar la aplicación

Imagen 2, maquetado

Imagen que contiene Diagrama

Descripción generada automáticamente

## 7.3 FASES DEL PROYECTO

A continuación, se presenta la carta de Gantt, la cual se detallan las actividades y fechas de trabajo en las fases del proyecto.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | RF | RN | Responsable | Semana | Horas |
|  |  |  |  | Día |  |
|  |  |  |  | Fecha |  |
| **Fase 1: Base de datos y Login** | 1 - 6 | 1 - 12 |  |  | Total: 45 - 60 horas |
| Mapeo de Base de Datos | 1 |  | Alez |  | 2 - 3 horas |
| Creación de base de datos | 2 |  | Alez |  | 2 - 3 horas |
| Creación de tablas | 3 |  | Alez |  | 2 - 3 horas |
| Relación de las bases de datos | 1 |  | Héctor |  | 4 - 5 horas |
| Crear interfaz del login | 2 |  | Héctor |  | 4 - 5 horas |
| Crear interfaz del registro | 2 |  | Héctor |  | 3 - 4 horas |
| Crear interfaz de recuperación de contraseña | 6 |  | Héctor |  | 5 - 6 horas |
| Estructura interna del MVC | 1 |  | Alez |  | 3 - 5 horas |
| Relaciones del MVC | 1 |  | Alez |  | 4 - 6 horas |
| Acomodo frontend del MVC | 4 |  | Héctor |  | 2 - 3 horas |
| Paleta de colores frontend del MVC | 1 |  | Héctor |  | 1 - 3 horas |
| Detalles de cajas en el frontend del MVC | 1 |  | Héctor |  | 1 - 3 horas |
| UI del login | 5 |  | Héctor |  | 2 - 3 horas |
| Conexión del login a BD | 2 |  | Alez |  | 1 - 3 horas |
| Conexión del login al MVC | 2 |  | Alez |  | 1 - 3 horas |
| PRUEBAS: Login, Registro | - |  | Ambos |  | 3 horas |
| PRUEBAS: BD y cambio de contraseña | - |  | Ambos |  | 3 horas |
| PRUEBAS: correcta colocación | - |  | Ambos |  | 2 horas |
| **Fase 2: Vistas básicas y estructura general** | 7 - 18 | 13 - 39 |  |  | Total: 17 - 29 horas |
| Creación de la estructura básicas de las vistas | 7 |  | Héctor |  | 3 - 4 horas |
| Creación de botones y redirecciones | 14 |  | Héctor |  | 1 - 2 horas |
| Relación desde BD con las demás funciones de la app | 14 |  | Alez |  | 1 - 2 horas |
| Relación desde BD con las redirecciones | 14 |  | Alez |  | 1 - 2 horas |
| Relación desde BD con vistas | 7 |  | Alez |  | 1 - 2 horas |
| Creación de la barra de desplazamiento | 13 |  | Alez |  | 2 - 4 horas |
| Creación de vínculos en la barra de desplazamiento | 13 |  | Alez |  | 1 - 2 horas |
| Creación de íconos | 7 |  | Héctor |  | 1 - 2 horas |
| Creación de formatos de diseño | 7 |  | Héctor |  | 2 - 3 horas |
| Ubicación de paletas de colores | 7 |  | Héctor |  | 1 - 2 horas |
| Ubicación de formatos | 7 |  | Héctor |  | 1 - 2 horas |
| PRUEBAS: redirección y desplazamiento correcto entre todas las vistas | - |  | Ambos |  | 2 horas |
| **Fase 3: Índex** | 19 - 53 | 40 - 100 |  |  | Total: 38 - 58 horas |
| Estructura de los módulos y botones, diseño e íconos | 19 |  | Héctor |  | 2 - 3 horas |
| Backend del Gráfico de Pastel | 20 |  | Héctor |  | 3 - 4 horas |
| Ecuaciones del Gráfico de Pastel | 20 |  | Héctor |  | 2 - 3 horas |
| Frontend y redirecciones del Gráfico de Pastel (UI) | 20 |  | Héctor |  | 1 - 2 horas |
| Diseño de "To do List" | 22 |  | Héctor |  | 4 - 5 horas |
| Creación de Plantilla de CRUD para "Nuevo Monto" | 24 |  | Héctor |  | 4 - 6 horas |
| Creación de Plantilla de CRUD para "Por pagar" | 25 |  | Alez |  | 4 - 6 horas |
| Subvista de Ingresos | 29 |  | Alez |  | 2 - 4 horas |
| Enlaces de la subvista de ingresos | 30 |  | Alez |  | 1 - 2 horas |
| Subvista de Gastos | 38 |  | Alez |  | 4 - 6 horas |
| Enlaces a bd en la subvista de gastos | 34 |  | Alez |  | 2 - 4 horas |
| Vínculos desde otras vistas a subvista de gastos | 38 |  | Alez |  | 1 - 2 horas |
| Subvista de Cosas por pagar | 48 |  | Alez |  | 4 - 6 horas |
| Vínculos a bd de subvista de cosas por pagar | 50 |  | Alez |  | 1 - 2 horas |
| PRUEBAS: Inserción de montos positivos y negativos en BD | - |  | Ambos |  | 2 horas |
| PRUEBAS: Redirecciones | - |  | Ambos |  | 1 hora |
| **Fase 4: Perfil** | 54 - 71 | 101 - 125 |  |  | Total: 45 - 66 horas |
| Información básica del usuario | 55 |  | Alez |  | 3 - 4 horas |
| Foto de perfil | 54 |  | Alez |  | 2 - 3 horas |
| Íconos varios | 59 |  | Héctor |  | 1 - 3 horas |
| Acomodo de los módulos y redirecciones | 59 |  | Héctor |  | 1 - 2 horas |
| Ubicación de redirecciones de configuraciones | 60 |  | Héctor |  | 1 - 2 horas |
| Tipo de trabajo | 62 |  | Héctor |  | 2 - 3 horas |
| Generación dinámica de acuerdo al tipo de trabajo | 62 |  | Héctor |  | 4 - 6 horas |
| CRUD de grupos | 64 |  | Alez |  | 4 - 6 horas |
| Lista de grupos | 64 |  | Alez |  | 2 - 4 horas |
| Grupos: creación de las tablas en bd | 65 |  | Héctor |  | 1 - 2 horas |
| Grupos: unión con otros usuarios | 69 |  | Héctor |  | 3 - 4 horas |
| Grupos: adición a una "cuenta de gastos compartida" | 64 |  | Héctor |  | 4 - 5 horas |
| Grupos: permisos | 66 |  | Héctor |  | 3 - 4 horas |
| Grupos: CRUD de miembros | 64 |  | Alez |  | 4 - 6 horas |
| Grupos: interfaces con subvistas de pagos y gastos | 65 |  | Alez |  | 3 - 4 horas |
| Grupos: personalización en los grupos | 68 |  | Héctor |  | 1 -2 horas |
| PRUEBAS: Generación dinámica | - |  | Ambos |  | 3 horas |
| PRUEBAS: Grupos | - |  | Ambos |  | 3 horas |
| **Fase 4: Historial** | 72 - 83 | 126 - 153 |  |  | Total: 30 - 38 horas |
| Estructura del calendario | 72 |  | Héctor |  | 3 - 4 horas |
| Vínculos a datos del dispositivo y BD para fechas | 72 |  | Alez |  | 1 - 2 horas |
| Características de módulos para posición de fecha | 73 |  | Alez |  | 1 - 2 horas |
| Ubicación de fecha actual | 76 |  | Alez |  | 1 - 2 horas |
| Etiquetas | 74 |  | Alez |  | 3 - 4 horas |
| Filtro de etiquetas | 75 |  | Alez |  | 2 - 4 horas |
| Personalización de etiquetas | 82 |  | Alez |  | 1 - 3 horas |
| Línea de tiempo | 78 |  | Héctor |  | 4 - 5 horas |
| (UI de posición de fecha) | 79 |  | Héctor |  | 1 - 2 horas |
| Guardados y eliminados | 80 |  | Héctor |  | 1 - 3 horas |
| Papelera | 81 |  | Héctor |  | 2 - 3 horas |
| Configuración de fechas | 83 |  | Héctor |  | 2 - 4 horas |
| PRUEBAS: Guardado de datos | - |  | Ambos |  | 3 horas |
| PRUEBAS: Papelera | - |  | Ambos |  | 2 horas |
| PRUEBAS: Etiquetas y filtros | - |  | Ambos |  | 3 horas |
| **Fase 5: Vista de planes de ahorro** | 84 - 98 | 154 - 182 |  |  | Total: 23 - 36 horas |
| Estructura de la vista | 84 |  | Héctor |  | 2 - 3 horas |
| Módulos y vínculos con otras vistas | 84 |  | Héctor |  | 1 - 2 horas |
| Implemento de un CRUD para deuda | 90 |  | Héctor |  | 3 - 4 horas |
| Enlace de CRUD para deuda, símil al CRUD de gasto | 90 |  | Héctor |  | 1 - 3 horas |
| Fechas de límite | 88 |  | Héctor |  | 1 - 2 horas |
| Vinculación con otras subvistas de las fechas límite | 88 |  | Héctor |  | 1 - 3 horas |
| Vinculación con historial | 85 |  | Héctor |  | 2 - 3 horas |
| Subvista de pagos | 89 |  | Alez |  | 3 - 4 horas |
| Lista con todos los datos del pago | 86 |  | Alez |  | 2 - 3 horas |
| Autodecrementos de acuerdo a las fechas en las deudas | 94 |  | Alez |  | 1 hora |
| Autoincrementos de acuerdo a las fechas en las deudas | 94 |  | Alez |  | 1 hora |
| Ediciones de gastos | 93 |  | Alez |  | 1 - 2 horas |
| Confirmaciones de gastos | 98 |  | Alez |  | 1 - 2 horas |
| PRUEBAS: Confirmar la información brindada | - |  | Ambos |  | 2 horas |
| PRUEBAS: Efectos de las alertas de deuda | - |  | Ambos |  | 1 hora |
| **Fase 6: Vista de finanzas** | 99 - 137 | 183 - 221 |  |  | Total: 35 - 54 horas |
| Estructura y módulos | 99 |  | Héctor |  | 2 - 3 horas |
| Consejos y Recomendaciones de uso (automatizadas) | 103 |  | Héctor |  | 4 - 6 horas |
| Fechas relevantes (vinculadas a otras vistas) | 102 |  | Héctor |  | 3 - 4 horas |
| Aceptación y rechazo de sugerencias | 99 |  | Héctor |  | 2 - 3 horas |
| Algoritmo de detección de ahorro | 100 |  | Héctor |  | 4 - 6 horas |
| Plantilla de gráfica | 107 |  | Héctor |  | 3 - 5 horas |
| Graficación del ahorro | 109 |  | Héctor |  | 2 - 4 horas |
| Tipos de ahorro | 107 |  | Héctor |  | 3 - 4 horas |
| Divisas | 111 |  | Alez |  | 2 - 4 horas |
| Módulo de Producto Recomendado | 123 |  | Alez |  | 2 - 3 horas |
| Fechas Importantes | 130 |  | Alez |  | 1 - 3 horas |
| Notificaciones | 137 |  | Alez |  | 1 - 3 horas |
| PRUEBAS: Sugerencias correctas y con efectos adecuados | - |  | Ambos |  | 4 horas |
| PRUEBAS: Gráficas correctas | - |  | Ambos |  | 2 horas |
| **Fase 7: Widgets, configuraciones y demás** | 138 - 147 | 222 |  |  | Total: 29 - 42 horas |
| Widget de ingreso | 139 |  | Héctor |  | 3 - 6 horas |
| Widget de gasto | 140 |  | Héctor |  | 2 - 4 horas |
| Widget de apertura a monto personalizado | 141 |  | Héctor |  | 2 - 3 horas |
| Configuración y opción de estilo claro u oscuro | 143 |  | Alez |  | 2 - 3 horas |
| Cambio de paletas de colores | 143 |  | Alez |  | 2 - 3 horas |
| Anexo de información sobre la aplicación | 142 |  | Alez |  | 4 - 5 horas |
| Manual de uso para el usuario | 145 |  | Ambos |  | 4 - 6 horas |
| Manual de programador | 144 |  | Ambos |  | 4 - 6 horas |
| PRUEBAS: finalización, comprobación de todos los aspectos de la aplicación | - |  | Ambos |  | 6 horas |

# CAPÍTULO VIII. IMPLEMENTACIÓN/DESARROLLO DEL PROYECTO

## 1 Gradle

plugins {  
 id 'com.android.application'  
 id 'org.jetbrains.kotlin.android'  
 id 'kotlin-kapt'  
 id 'kotlin-parcelize'  
}  
  
android {  
 namespace 'com.example.st5'  
 compileSdk 33  
  
 defaultConfig {  
 applicationId "com.example.st5"  
 minSdk 28  
 targetSdk 33  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 }  
 viewBinding {  
 enabled = true  
 }  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 }  
 }  
 compileOptions {  
 coreLibraryDesugaringEnabled true  
  
 sourceCompatibility JavaVersion.VERSION\_1\_8  
 targetCompatibility JavaVersion.VERSION\_1\_8  
 }  
 kotlinOptions {  
 jvmTarget = '1.8'  
 }  
 buildFeatures {  
 viewBinding true  
 dataBinding true  
 }  
 packagingOptions {  
 exclude 'license/\*'  
 }  
}  
  
dependencies {  
 implementation fileTree(dir: "libs", include: ["\*.jar"])  
 implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin\_version"  
 implementation 'androidx.core:core-ktx:1.10.0'  
 implementation 'androidx.appcompat:appcompat:1.6.1'  
 implementation 'com.google.android.material:material:1.8.0'  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
 implementation 'com.google.android.gms:play-services-wearable:17.1.0'  
 coreLibraryDesugaring 'com.android.tools:desugar\_jdk\_libs:2.0.3'  
  
 // Volley  
 implementation 'com.android.volley:volley:1.2.1'  
  
 // Navigation  
 implementation 'androidx.navigation:navigation-fragment-ktx:2.5.3'  
 implementation 'androidx.navigation:navigation-ui-ktx:2.5.3'  
  
 // Room  
 implementation "androidx.room:room-runtime:2.5.1"  
 implementation "androidx.room:room-ktx:2.5.1"  
 implementation 'androidx.test:core-ktx:1.5.0'  
 implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'  
 implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'  
 implementation "androidx.work:work-runtime-ktx:2.8.0"  
 kapt "androidx.room:room-compiler:2.5.1"  
  
 implementation 'com.google.code.gson:gson:2.9.0'  
  
 testImplementation 'junit:junit:4.13.2'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
  
 // Email  
 implementation 'com.sun.mail:android-mail:1.6.4'  
 implementation 'com.sun.mail:android-activation:1.6.4'  
  
 // Visuals  
 implementation "io.coil-kt:coil:1.1.1"  
 implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'  
 implementation 'com.polyak:icon-switch:1.0.0'  
 implementation 'com.android.support:support-annotations:28.0.0'  
 implementation 'com.kizitonwose.calendar:view:2.0.0'  
 implementation 'pl.droidsonroids.gif:android-gif-drawable:1.2.19'  
 implementation 'ca.antonious:materialdaypicker:0.7.3'  
 implementation 'com.github.yukuku:ambilwarna:2.0.1'  
 implementation 'com.github.kenglxn.QRGen:javase:3.0.1'  
 implementation "com.google.assistant.appactions:widgets:0.0.1"  
}

## 2 Login

### 2.1 login binding al fragment

La función ‘FragmentLoginBinding’ es una función la cual se utiliza para unir el fragmento con las vistas en las cuales se va a mostrar, por lo que este tipo de binding se verá dentro de todo el proyecto siendo renombrado mas cumpliendo con exactamente la misma función.

class Login : Fragment(), AdapterView.OnItemSelectedListener {  
 private lateinit var binding: FragmentLoginBinding  
  
 private var mutableNombres: MutableList<String> = mutableListOf()  
 private var mutableIds: MutableList<Long> = mutableListOf()  
  
 private var selectedLabel = 0  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 requireActivity().onBackPressedDispatcher.addCallback(  
 this,  
 object : OnBackPressedCallback(true) {  
 override fun handleOnBackPressed() {  
 val actual = Login()  
 parentFragmentManager.beginTransaction().replace(R.id.FragContainer, actual)  
 .addToBackStack(null).commit()  
 }  
 })  
 }

En resumen, la función permite navegar dentro de los bindings dentro de la app.

### 2.2 caché de usuarios en login

La clase ‘FragmentLoginBinding’ al ser ejecutada utiliza una corrutina llamada ‘lifecycleScope.launch’ la cual al ejecutarse busca un contexto o en este caso un caché, el cual está almacenado en el teléfono con la finalidad de buscar a los usuarios que ya hayan iniciado sesión previamente, en caso de que existiesen se muestra en un spinner para solo ingresar la contraseña y poder acceder, caso contrario no se muestra ningún usuario.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?  
): View {  
 binding = FragmentLoginBinding.inflate(inflater, container, false)  
  
 lifecycleScope.launch {  
 getUsersCache()  
  
 val arrayUsers = mutableNombres  
 val adapterG = ArrayAdapter(  
 requireContext(),  
 android.R.layout.simple\_spinner\_item,  
 arrayUsers  
 )  
 adapterG.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
  
 binding.SpinnerUsers.adapter = adapterG  
 binding.SpinnerUsers.onItemSelectedListener = this@Login  
 }  
  
 return binding.root  
}

En resumen, la clase se encarga de mostrar las cuentas que se tengan guardadas en el caché dentro de un spinner para que sea más interactivo el re-ingreso a la cuenta.

### 2.2.1 Obtener el caché de usuarios

La función ‘getUsersCache’ es usada para acceder a la base de datos local del dispositivo, donde se encuentra con 2 campos: ‘mutableIds’ y ‘mutableNombres’.

Dichos campos son usados para verificar que su identificador y nombre de usuario sea el correcto. Dentro de la función se utiliza el método ‘getMaxId’ para obtener la cantidad máxima de

private suspend fun getUsersCache() {  
 withContext(Dispatchers.IO) {  
 val usersCacheDao = Stlite.getInstance(requireContext()).getUserCacheDao()  
  
 val max = usersCacheDao.getMaxId()  
  
  
 mutableIds.add(0)  
 mutableNombres.add("Seleccionar")  
 for (i in 1..max) {  
 if (usersCacheDao.getNombreUserCache(i.toLong()) != null){  
 mutableIds.add(usersCacheDao.getIdUserCache(i.toLong()))  
 mutableNombres.add(usersCacheDao.getNombreUserCache(i.toLong()))  
 }  
 }  
 Log.v("idu", "$mutableIds")  
 Log.v("nombres", "$mutableNombres")  
 }  
}

usuarios almacenados en caché y a partir de ese punto se comienza a iterar y verificar los mutables.

En resumen la función ‘getUsersCache’ se encarga de obtener el caché del dispositivo para obtener los usuarios que hayan hecho login previamente y verificar la integridad de los campos el caché.

### 2.3 animaciones de navegación

La función ‘parentFragmentManager.beginTransaction’ se usa para tener animaciones al navegar entre fragmentos dentro de las vistas, estas animaciones se pueden realizar de izquierda a derecha, arriba abajo y viceversa.

binding.buttonSinCuenta.setOnClickListener {  
 val fragmentregister = Register()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.FragContainer, fragmentregister).addToBackStack(null).commit()  
}

En resumen, la función ‘parenFragmentManager.beginTransaction’ se puede ver dentro de todo el código ya que se usa complementando la navegación.

### 2.4 Ingreso de datos y concatenación de query

El método se utiliza para solicitar al usuario que ingrese el usuario y contraseña en los campos del login, mediante volley se inicia una solicitud con ‘queue’, verifica que haya conexión a internet mediante ‘Decoder’, si cumple con las condiciones concatena la URL del sitio con el usuario y contraseña mediante ‘checkUserURL’.

Imagen 3, login

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

binding.buttonIniSes.setOnClickListener {  
 val queue = Volley.newRequestQueue(requireContext())  
 val url = "http://savetrack.com.mx/usrlogin.php"  
  
 val username = binding.editTextTextPersonName.text.toString()  
 val password = binding.editTextTextPassword.text.toString()  
  
 binding.perame.alpha = 1f  
 if (username != "" && password != "" && Decoder(requireContext()).hayNet()) {  
 val checkUserUrl = "$url?username=$username&password=$password"  
 Log.d("checkUserUrl", checkUserUrl)  
 /\*  
 CHECK USER REQUEST  
 \*/

En resumen, el método se usa para comenzar el proceso de login concatenando los inputs del usuario.

### 2.5 Comprobación de credenciales y redirect

El método es usado para pedir una solicitud GET a través de la clase ‘StringRequest’ con la finalidad de comprobar las credenciales de login proporcionadas por el usuario, en caso de ser incorrectas se arrojará un toast con la leyenda “Usuario y/o contraseña incorrectos”, caso contrario se redirigirá a la actividad de perfil.

val checkUserReq = StringRequest(Request.Method.GET, checkUserUrl, { response ->  
 val strResp = response.toString()  
 Log.d("API checkuser", strResp)  
 if (response != "exist") {  
 Toast.makeText(  
 requireContext(),  
 "Usuario y/o contraseña incorrectos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 } else {  
 // Usuario y contraseña correctos, redirigir a la actividad de perfil  
 Log.v("username", username)  
 Log.v("password", password)

En resumen, el método permite comprobar el acceso y redirigir al usuario para finalizar el login de la aplicación.

### 2.6 comprobación de la base de datos nativa

El método consta de 2 fases las cuales son la comprobación de si existe o no un “backup” el cual se construye bajo el nombre ‘checkbackupurl’ y verifica si existe o no un respaldo con el nombre almacenado en la variable ‘$username’, en caso de que reciba un NULL realiza una segunda consulta con el nombre ‘idurl’ la cual asocia el id del usuario con el backup a crear.

var checkbackupurl = "http://savetrack.com.mx/backuget.php"  
checkbackupurl += "?username=$username"  
/\*  
CHECK BACKUP REQUEST  
\*/  
val backReq: StringRequest = object : StringRequest(Method.GET,  
 checkbackupurl,  
 Response.Listener { response ->  
 val backReq2 = response.toString()  
 Log.d("API backupreq", backReq2)  
 var idurl = "http://savetrack.com.mx/idget.php"  
 idurl += "?username=$username"  
 /\*  
 CHECK ID REQUEST  
 \*/  
 if (response.toString() == "null") {  
 // CHECK ID AND CREATE BACKUP  
 val idReq: StringRequest = object : StringRequest(Method.GET,  
 idurl,  
 Response.Listener { response ->  
 val id: Long = response.toLong()  
 val idReq2 = response.toString()  
 Log.d("API id bnull", idReq2)

En resumen, este fragmento de código se encarga de comprobar la existencia de un backup en la base de datos y en caso que no exista registra al usuario para poder crear el backup.

### 2.7 Crear backup

La función ‘insertarNuevoUsuario’ es un proceso que se ejecuta de manera asíncrona mediante corrutinas las cuales toman los valores de ‘id’ y ‘username’ para identificar los campos a insertar dentro de varias tablas dentro de la bd, todo esto mediante “DAO’S” que son usados con la clase ‘Stlite’. Una vez se tiene el acceso se crean instancias de objetos con valores predeterminados, los cuales son posteriormente insertados dentro de las tablas mediante métodos como ‘insertUsuario’, ‘insertIngresosGastos’ y demás, correspondiendo a cada uno de los objetos y tablas, finalmente se hace una consulta a la base de datos y al ser exitosa se imprime un mensaje del registro seleccionado.

// CREATING BACKUP  
suspend fun insertarNuevoUsuario(  
 id: Long, username: String  
) {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresosGastosDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val montoGrupoDao = Stlite.getInstance(requireContext()).getMontoGrupoDao()  
 val gruposDao = Stlite.getInstance(requireContext()).getGruposDao()  
 val assetsDao = Stlite.getInstance(requireContext()).getAssetsDao()  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
  
 val nuevoUsuario = Usuario(  
 iduser = id,  
 nombre = username,  
 edad = 0,  
 chamba = 0,  
 foto = "",  
 diasaho = 0,  
 balance = 0.0  
 )  
 val nuevosIG = IngresosGastos(  
 iduser = id,  
 summaryingresos = 0.0,  
 summarygastos = 0.0  
 )  
 val nuevoMontoGrupo = MontoGrupo(  
 idmonto = 0,  
 idgrupo = 0,  
 iduser = id,  
 )  
 val nuevoGrupo = Grupos(  
 Id = 0,  
 name = "",  
 description = "",  
 admin = id,  
 nmembers = 1,  
 enlace = ""  
 )  
 val defaultAssets = Assets(  
 theme = 0,  
 lastprocess = ""  
 )  
  
 usuarioDao.clean()  
 ingresosGastosDao.clean()  
 montoDao.clean()  
 montoGrupoDao.clean()  
 gruposDao.clean()  
 assetsDao.clean()  
 labelsDao.clean()  
  
 usuarioDao.insertUsuario(nuevoUsuario)  
 ingresosGastosDao.insertIngresosGastos(nuevosIG)  
 montoGrupoDao.insertMontoG(nuevoMontoGrupo)  
 gruposDao.insertGrupo(nuevoGrupo)  
 assetsDao.insertAsset(defaultAssets)  
  
 val selected = usuarioDao.getUserData()  
 Log.v("SELECTED USERS", selected.toString())

En resumen, la función se encarga de limpiar registros en caso de existir dentro de las tablas, insertar al usuario dentro de las mismas y generar los datos default para el correcto funcionamiento de la base de datos.

### 2.8 Upload de backup a la base de datos

La carga del respaldo hacia el servidor es realizada mediante distintas solicitudes HTTP PUT las cuales son apuntadas a distintos endopoints en el servidor y se respaldan entidades como ‘Usuario’, ‘IngresosGastos’, ‘Monto’, etc. Todo esto se realiza mediante la creación de ‘JSONobject’ tabla a tabla agregando los datos correspondientes para después construir las URL’s de los distintos endpoints. Como parámetros dentro de la solicitud PUT se usan el ‘username’ y ‘JSONobject’ dentro de la URL, todo esto através de la función ‘StringRequest’ tabla a tabla, dentro del cuerpo de la solicitud ‘getBody()’ el JSONobject es convertido a un array de bytes y todas estas solicitudes son encoladas mediante solicitudes de Volley usando ‘queue’.

Finalmente, mientras que se hacen las solicitudes con Volley, se realiza una solicitud adicional ‘inipic.php’ la cual se encarga de cargar una imagen asociada al usuario la cual es usada como foto de perfil.

UPLOADING BACKUP  
 \*/  
  
 // Tabla Usuario  
 val jsonObjectUsuario = JSONObject()  
 jsonObjectUsuario.put("iduser", nuevoUsuario.iduser)  
 jsonObjectUsuario.put("edad", nuevoUsuario.edad)  
 jsonObjectUsuario.put("nombre", nuevoUsuario.nombre)  
 jsonObjectUsuario.put("chamba", nuevoUsuario.chamba)  
 jsonObjectUsuario.put("diasaho", nuevoUsuario.diasaho)  
 jsonObjectUsuario.put("balance", nuevoUsuario.balance)  
  
 // Tabla IngresosGastos  
 val jsonObjectIngresosGastos = JSONObject()  
 jsonObjectIngresosGastos.put("iduser", nuevosIG.iduser)  
 jsonObjectIngresosGastos.put("summaryingresos", nuevosIG.summaryingresos)  
 jsonObjectIngresosGastos.put("summarygastos", nuevosIG.summarygastos)  
  
 // Tabla Monto  
 val jsonObjectMonto = JSONObject()  
  
 // Tabla MontoGrupo  
 val jsonObjectMontoGrupo = JSONObject()  
 jsonObjectMontoGrupo.put("idmonto", nuevoMontoGrupo.idmonto)  
 jsonObjectMontoGrupo.put("idgrupo", nuevoMontoGrupo.idgrupo)  
 jsonObjectMontoGrupo.put("iduser", nuevoMontoGrupo.iduser)  
  
 // Tabla Grupos  
 val jsonObjectGrupos = JSONObject()  
 jsonObjectGrupos.put("Id", nuevoGrupo.Id)  
 jsonObjectGrupos.put("name", nuevoGrupo.name)  
 jsonObjectGrupos.put("description", nuevoGrupo.description)  
 jsonObjectGrupos.put("admin", nuevoGrupo.admin)  
 jsonObjectGrupos.put("nmembers", nuevoGrupo.nmembers)  
 jsonObjectGrupos.put("enlace", nuevoGrupo.enlace)  
  
 // Tabla Labels  
 val jsonObjectLabels = JSONObject()  
  
 Log.v("jsonObjectUsuario", jsonObjectUsuario.toString())  
  
 val uploadurl =  
 "http://savetrack.com.mx/backupput.php?username=$username&backup=$jsonObjectUsuario"  
 val uploadReq: StringRequest =  
 object : StringRequest(Method.PUT,  
 uploadurl,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", uploadReq.toString()  
 )  
 queue.add(uploadReq)  
  
 val upload2url =  
 "http://savetrack.com.mx/backupput2.php?username=$username&backup=$jsonObjectIngresosGastos"  
 val upload2Req: StringRequest =  
 object : StringRequest(Method.PUT,  
 upload2url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload2Req.toString()  
 )  
 queue.add(upload2Req)  
  
 val upload3url =  
 "http://savetrack.com.mx/backupput3.php?username=$username&backup=$jsonObjectMonto"  
 val upload3Req: StringRequest =  
 object : StringRequest(Method.PUT,  
 upload3url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload3Req.toString()  
 )  
 queue.add(upload3Req)  
  
 val upload4url =  
 "http://savetrack.com.mx/backupput4.php?username=$username&backup=$jsonObjectMontoGrupo"  
 val upload4Req: StringRequest =  
 object : StringRequest(Method.PUT,  
 upload4url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload4Req.toString()  
 )  
 queue.add(upload4Req)  
  
 val upload5url =  
 "http://savetrack.com.mx/backupput5.php?username=$username&backup=$jsonObjectGrupos"  
 val upload5Req: StringRequest =  
 object : StringRequest(Method.PUT,  
 upload5url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload5Req.toString()  
 )  
 queue.add(upload5Req)  
  
 val upload6url =  
 "http://savetrack.com.mx/backupput6.php?username=$username&backup=$jsonObjectLabels"  
 val upload6Req: StringRequest =  
 object : StringRequest(Method.PUT,  
 upload6url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload6Req.toString()  
 )  
 queue.add(upload6Req)  
  
 val url7 =  
 "http://savetrack.com.mx/images/inipic.php?username=$username"  
 val stringRequest = object : StringRequest(  
 Method.POST, url7,  
 Response.Listener { response ->  
 try {  
 Log.d("UPLOAD SUCCESS", response)  
 } catch (e: JSONException) {  
 e.printStackTrace()  
 }  
 },  
 Response.ErrorListener { error ->  
 Log.e("UPLOAD API ERROR", error.toString())  
 Toast.makeText(requireContext(), "No se ha podido establecer conexión a Internet", Toast.LENGTH\_LONG).show()  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
  
 val socketTimeout = 5000  
 val policy: RetryPolicy = DefaultRetryPolicy(  
 socketTimeout,  
 DefaultRetryPolicy.DEFAULT\_MAX\_RETRIES,  
 DefaultRetryPolicy.DEFAULT\_BACKOFF\_MULT  
 )  
 stringRequest.retryPolicy = policy  
 val requestQueue = Volley.newRequestQueue(requireContext())  
 requestQueue.add(stringRequest)  
  
 }  
 }  
 lifecycleScope.launch {  
 insertarNuevoUsuario(id, username)  
 }  
 },  
 Response.ErrorListener { error ->  
 Log.e("API error", "error => $error")  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(Charset.defaultCharset())  
 }  
}  
Log.d("idReq", idReq.toString())  
queue.add(idReq)

En resumen, este fragmento del código se encarga de asegurar que el respaldo creado sea correctamente insertado mediante PUT dentro de la base de datos, así como la carga adecuada de la foto de perfil del usuario, todo esto siendo controlado mediante ‘queue’ de Volley.

### 2.9 Verificación de id y restore de la bd

Tras subir el respaldo a la bd se realiza una solicitud HTTP llamada ‘checkUserReq’ la cual se encarga de verificar que el usuario exista dentro de la bd, tras verificarlo realiza otra solicitud HTTP ‘backReq’ con la finalidad de verificar la copia de seguridad, acto seguido se ejecuta la función ‘extraerDatosBackup’ en un thread para extraer los datos del ususario como lo son sus datos personales, sus gastos, ingresos, grupos y demás. Mediante los DAO’s se insertan todos estos datos dentro de las tablas correspondientes y tras verificar correctamente el backup, se arroja un toast de bienvenida con la leyenda “Bienvenido, $username” concatenando el usarname y se inicia la activity ‘Index::class.java’.

// CHECK ID AND RESTORE  
 } else {  
  
 val idReq: StringRequest = object : StringRequest(Method.GET,  
 idurl,  
 Response.Listener { response ->  
 val idReq2 = response.toString()  
 Log.d("API id bnotnull", idReq2)  
  
 // Restore  
 suspend fun extraerDatosBackup(username: String) {  
 withContext(Dispatchers.IO) {  
  
 val jsonObject1 =  
 JSONObject(URL("http://savetrack.com.mx/backupget.php?username=$username").readText())  
 val idu: Long = jsonObject1.getLong("iduser")  
 val nombre: String = jsonObject1.getString("nombre")  
 val edad: Long = jsonObject1.optLong("edad")  
 val chamba: Long = jsonObject1.optLong("chamba")  
 val diasaho: Long = jsonObject1.optLong("diasaho")  
 val balance: Double = jsonObject1.optDouble("balance")  
 val usuarioDao = Stlite.getInstance(  
 requireContext()  
 ).getUsuarioDao()  
  
 val jsonObject2 =  
 JSONObject(URL("http://savetrack.com.mx/backupget2.php?username=$username").readText())  
 val idus: Long = jsonObject2.getLong("iduser")  
 val summaryingresos: Double = jsonObject2.optDouble("summaryingresos")  
 val summarygastos: Double = jsonObject2.optDouble("summarygastos")  
 val ingresosGastosDao = Stlite.getInstance(  
 requireContext()  
 ).getIngresosGastosDao()  
  
 val jsonArray3 =  
 JSONArray(URL("http://savetrack.com.mx/backupget3.php?username=$username").readText())  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 montoDao.clean()  
 Log.v("jsonArray3", jsonArray3.toString())  
 for (i in 0 until jsonArray3.length())  
 {  
 val jsonObject3 = jsonArray3.getJSONObject(i)  
 if (jsonObject3.getLong("idmonto") != null) {  
 val idmonto: Long =  
 jsonObject3.getLong("idmonto")  
 val iduse: Long =  
 jsonObject3.getLong("iduser")  
 val concepto: String =  
 jsonObject3.optString("concepto")  
 val valor: Double =  
 jsonObject3.optDouble("valor")  
 val fecha: String =  
 jsonObject3.optString("fecha")  
 val frecuencia: Long =  
 jsonObject3.optLong("frecuencia")  
 val etiqueta: Long =  
 jsonObject3.optLong("etiqueta")  
 val interes: Double =  
 jsonObject3.optDouble("interes")  
 val veces: Long =  
 jsonObject3.optLong("veces")  
  
 val nuevoMonto = Monto(  
 idmonto = idmonto,  
 iduser = iduse,  
 concepto = concepto,  
 valor = valor,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 veces = veces  
 )  
 Log.v("Current monto $i", nuevoMonto.toString())  
  
 montoDao.insertMonto(nuevoMonto)  
 } else {  
 Log.v("Current monto $i", "VACÍO")  
 }  
 }  
  
 val jsonObject4 =  
 JSONObject(URL("http://savetrack.com.mx/backupget4.php?username=$username").readText())  
 val idmontog: Long = jsonObject4.getLong("idmonto")  
 val idg: Long = jsonObject4.getLong("idgrupo")  
 val idusemg: Long = jsonObject4.getLong("iduser")  
 val montoGrupoDao = Stlite.getInstance(  
 requireContext()  
 ).getMontoGrupoDao()  
  
 val jsonObject5 =  
 JSONObject(URL("http://savetrack.com.mx/backupget5.php?username=$username").readText())  
 val idgru: Long = jsonObject5.getLong("Id")  
 val nameg: String = jsonObject5.getString("name")  
 val description: String = jsonObject5.optString("description")  
 val admin: Long = jsonObject5.getLong("admin")  
 val nmembers: Long = jsonObject5.optLong("nmembers")  
 val enlace: String = jsonObject5.getString("enlace")  
 val gruposDao = Stlite.getInstance(  
 requireContext()  
 ).getGruposDao()  
  
 val jsonArray6 =  
 JSONArray(URL("http://savetrack.com.mx/backupget6.php?username=$username").readText())  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
 labelsDao.clean()  
 Log.v("jsonArray6", jsonArray6.toString())  
 for (i in 0 until jsonArray6.length())  
 {  
 val jsonObject6 = jsonArray6.getJSONObject(i)  
 if (jsonObject6.getLong("idlabel") != null) {  
 val idlabel: Long =  
 jsonObject6.getLong("idlabel")  
 val plabel: String =  
 jsonObject6.getString("plabel")  
 val color: String =  
 jsonObject6.optString("color")  
  
 val nuevasLabels = Labels(  
 idlabel = idlabel,  
 plabel = plabel,  
 color = color,  
 )  
 Log.v("Current monto $i", nuevasLabels.toString())  
  
 labelsDao.insertLabel(nuevasLabels)  
 } else {  
 Log.v("Current monto $i", "VACÍO")  
 }  
 }  
  
 val assetsDao = Stlite.getInstance(requireContext()).getAssetsDao()  
  
 val nuevoUsuario = Usuario(  
 iduser = idu,  
 nombre = nombre,  
 edad = edad,  
 chamba = chamba,  
 foto = "",  
 diasaho = diasaho,  
 balance = balance  
 )  
  
 val nuevosIG = IngresosGastos(  
 iduser = idus,  
 summaryingresos = summaryingresos,  
 summarygastos = summarygastos  
 )  
  
 val nuevoMontoGrupo = MontoGrupo(  
 idmonto = idmontog,  
 idgrupo = idg,  
 iduser = idusemg,  
 )  
 val nuevoGrupo = Grupos(  
 Id = idgru,  
 name = nameg,  
 description = description,  
 admin = admin,  
 nmembers = nmembers,  
 enlace = enlace  
 )  
 val defaultAssets = Assets(  
 theme = 0,  
 lastprocess = ""  
 )  
  
 usuarioDao.clean()  
 ingresosGastosDao.clean()  
 montoGrupoDao.clean()  
 gruposDao.clean()  
 assetsDao.clean()  
  
 usuarioDao.insertUsuario(nuevoUsuario)  
 ingresosGastosDao.insertIngresosGastos(nuevosIG)  
 montoGrupoDao.insertMontoG(nuevoMontoGrupo)  
 gruposDao.insertGrupo(nuevoGrupo)  
 assetsDao.insertAsset(defaultAssets)  
  
 val selected = usuarioDao.getUserData()  
 Log.v("SELECTED USERS", selected.toString())  
  
 }  
 }  
 lifecycleScope.launch {  
 extraerDatosBackup(username)  
 }  
 },  
 Response.ErrorListener { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar a la red", Toast.LENGTH\_SHORT  
 ).show()  
 Log.e("API id", "error => $error")  
 }) {  
 override fun getBody(): ByteArray {  
 return idurl.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.d("idReq", idReq.toString())  
 queue.add(idReq)  
 }  
 },  
 Response.ErrorListener { error ->  
 Log.e("API backupreq", "error => $error")  
 }) {  
 override fun getBody(): ByteArray {  
 return checkbackupurl.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.d("backReq", backReq.toString())  
 queue.add(backReq)  
  
 Toast.makeText(  
 requireContext(), "Bienvenido, $username", Toast.LENGTH\_SHORT  
 ).show()  
  
 val intent = Intent(activity, Index::class.java)  
 startActivity(intent)  
 }  
 }, { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar a la red", Toast.LENGTH\_SHORT  
 ).show()  
 Log.e("API", "error => $error")  
 })  
 Log.d("checkUserReq", checkUserReq.toString())  
 queue.add(checkUserReq)  
 } else {  
 Toast.makeText(  
 requireContext(), "Los campos no pueden estar vacíos", Toast.LENGTH\_SHORT  
 ).show()  
 }  
 }  
 }  
}

En resumen, este fragmento del login es la última fase del mismo, dentro del cual se carga cualquier backup que exista en la bd bajo el username ingresado al hacer login, se cargan los datos y se redirige al index junto con un mensaje de bienvenida.

## 3 Recuperar cuenta

### 3.1 Envío de correo

El código se encarga de interactuar con la vista de ‘Login’ para redirigir a la vista, dentro se define una función ‘correooo’ en la que envía un correo electrónico para la confirmación de cambio de contraseña, después se ejecuta la clase ‘OlvidoAux’ para tener una sesión de correo y escribir el cuerpo completo del correo y si el envío es exitoso se muestra un toast y se hace una transición a ‘Olvido2’.

Imagen 4, reestablecer contraseña

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

@SuppressLint("SetTextI18n")  
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
  
 binding.buttonConCuenta.setOnClickListener {  
 val fragmentlogin = Login()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.FragContainer, fragmentlogin)  
 .addToBackStack(null)  
 .commit()  
 }  
  
 suspend fun correoo(strEmail: String, strUsername: String){  
 withContext(Dispatchers.IO){  
 val session = OlvidoAux().session()  
  
 try {  
 val message = MimeMessage(session)  
 message.setFrom(InternetAddress("pingenerator@savetrack.com.mx"))  
 message.setRecipients(  
 Message.RecipientType.TO,  
 InternetAddress.parse(strEmail)  
 )  
 message.subject = "Confirmación de cambio de contraseña"  
 message.setText("Tu pin de confirmación es $truepin")  
  
 Transport.send(message)  
  
 (context as Activity).runOnUiThread {  
 Toast.makeText(  
 requireContext(),  
 "Se ha enviado un pin de confirmación, revise su Correo",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
  
 val fragmentrecupe = Olvido2()  
 val bundle = Bundle()  
 bundle.putString("Username", strUsername)  
 fragmentrecupe.arguments = bundle  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.FragContainer, fragmentrecupe)  
 .addToBackStack(null)  
 .commit()  
  
 } catch (e: MessagingException) {  
 throw RuntimeException(e)  
 }  
 }  
 }

En resumen, esta función permite enviar el correo electrónico para la recuperación de contraseña y redirecciona para cambiar la misma.

### 3.2 Obtener Email para recuperación

El siguiente código se ejecuta tras presionar el botón ‘buttonOlviContra’ haciendo una request de Volley con HTTP para obtener el campo de texto y construye una URL para comprobar la existencia del usuario, si existe se hace una ‘StringRequest’ y se obtiene el correo electrónico del usuario para enviar el correo de recuperación, caso contrario se arroja un toast.

binding.buttonOlviContra.setOnClickListener {  
 lifecycleScope.launch {  
 val queue = Volley.newRequestQueue(requireContext())  
  
 val username = binding.Usuario.text.toString()  
 Log.d("Username", username)  
 if (username != "") {  
 val checkUserUrl = "http://savetrack.com.mx/usrget.php?username=$username"  
 val checkUserReq = StringRequest(  
 Request.Method.GET, checkUserUrl,  
 { response ->  
 val strResp = response.toString()  
 Log.d("API", strResp)  
 if (response == "exist") {  
 val checkEmailUrl = "http://savetrack.com.mx/emailget.php?username=$username"  
 val checkEmailReq = StringRequest(  
 Request.Method.GET, checkEmailUrl,  
 { response2 ->  
 Log.d("Pin", truepin.toString())  
 val strEmail = response2.toString()  
 Log.d("Email", strEmail)  
  
 // Enviar correo electrónico al usuario con su pin de confirmación  
 lifecycleScope.launch {  
 correoo(strEmail, username)  
 }  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar a la red", Toast.LENGTH\_SHORT  
 ).show()  
 Log.e("API error", "error => $error")  
 }  
 )  
 Log.d("checkEmailReq", checkEmailReq.toString())  
 queue.add(checkEmailReq)  
 } else {  
 // Usuario no existe  
 Toast.makeText(  
 requireContext(),  
 "El usuario ingresado no existe",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar a la red", Toast.LENGTH\_SHORT  
 ).show()  
 Log.e("API error", "error => $error")  
 }  
 )  
 Log.d("checkUserReq", checkUserReq.toString())  
 queue.add(checkUserReq)  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "El campo no puede estar vacío",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 }  
 }  
 }  
}

En resumen, este código se encarga de obtener el correo asociado con el usuario para permitir la recuperación de contraseña.

### 3.3 Reestablecer contraseña

El código se encarga de crear una cola ed Volley para iniciar una URL y actualizar la contraseña, obtiene el ‘PIN’ y la ‘password’ y ‘password2’ desde la interfaz del usuario, si el PIN concuerda se hace la actualización de la contraseña y se redirecciona al inicio de sesión ‘Login’, caso contario se manda un toast indicando que el PIN es equivocado.

Imagen 5, reestablecer contraseña 2

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

binding.buttonReestablecer.setOnClickListener {  
 val queue = Volley.newRequestQueue(requireContext())  
 var url = "http://savetrack.com.mx/passupdate.php?"  
  
 val truepinval = truepin.toString()  
 val pin = binding.textPin.text.toString()  
 val password = binding.RegPass.text.toString()  
 val password2 = binding.RegConPass.text.toString()  
  
 if (password == password2 && password != "" && pin == truepinval) {  
 if (password.length in 4..18) {  
 Log.d("Pin", truepin.toString())  
 Log.d("password", password)  
 val username = arguments?.getString("Username")  
 val requestBody = "username=$username&newpassword=$password"  
 url += requestBody  
  
 val stringReq: StringRequest =  
 object : StringRequest(Method.PUT, url,  
 Response.Listener { response ->  
 // response  
 val strResp2 = response.toString()  
 Log.d("API", strResp2)  
  
 },  
 Response.ErrorListener { error ->  
 Log.d("API", "error => $error")  
 }  
 ) {  
 override fun getBody(): ByteArray {  
 return requestBody.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.e("stringReq", stringReq.toString())  
 queue.add(stringReq)  
 Toast.makeText(  
 requireContext(),  
 "Contraseña actualizada correctamente",  
 Toast.LENGTH\_SHORT  
 ).show()  
 val fragmentlogin = Login()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.FragContainer, fragmentlogin)  
 .addToBackStack(null)  
 .commit()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "Contraseña: entre 4 y 18 caracteres",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 } else {  
 if (binding.textPin.text.isEmpty() || binding.RegPass.text.isEmpty() || binding.RegConPass.text.isEmpty()) {  
 Toast.makeText(  
 requireContext(),  
 "No puede haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "Las contraseñas no coinciden o el pin es incorrecto",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 }  
 }  
 }  
}

En resumen, el código permite que se efectúe el cambio de contraseña mediante HTTP y un PIN para actualizar la contraseña.

## 4 Registrar cuenta

### 4.1 Register

La siguiente clase maneja un evento mediante el botón ‘buttonRegister’ el cual verifica varios campos como lo son el usuario, contraseña, correo, entre otros los cuales obtiene de la interfaz de usuario. Verifica que la longitud de los campos sea la correcta, que las contraseñas sean idénticas y que no existan campos vacíos. Si se pasan las verificaciones se ejecuta un GET para comprobar que el usuario exista o no, si no existe se hace un POST y se muestra un toast de registro satisfactorio y se redirecciona a ‘Login’. Se contemplan errores como errores de conexión y que el usuario ya exista, todo esto se comunica al usuario mediante toasts.

Imagen 6, registrar cuenta

Interfaz de usuario gráfica

Descripción generada automáticamente

binding.buttonRegister.setOnClickListener {  
 val queue = Volley.newRequestQueue(requireContext())  
 var url = "http://savetrack.com.mx/usrpost.php?"  
  
 val username = binding.RegCorreo.text.toString()  
 val password = binding.RegPass.text.toString()  
 val email = binding.textEmail.text.toString()  
 val password2 = binding.RegConPass.text.toString()  
  
 if (password == password2 && password != "" && username != "" && email != "") {  
 if (username.length in 6..32 && password.length in 4..18) {  
 val checkUserUrl = "http://savetrack.com.mx/usrget.php?username=$username"  
  
 val checkUserReq = StringRequest(  
 Request.Method.GET, checkUserUrl,  
 { response ->  
 val strResp = response.toString()  
 Log.d("API", strResp)  
 if (response == "exist") {  
 Toast.makeText(  
 requireContext(),  
 "El usuario ya existe",  
 Toast.LENGTH\_SHORT  
 ).show()  
 } else {  
 // Usuario no existe, registrar nuevo usuario  
 Log.v("username", username)  
 Log.v("password", password)  
 Log.v("email", email)  
 val requestBody = "username=$username&password=$password&email=$email"  
 url += requestBody  
  
 val stringReq: StringRequest =  
 object : StringRequest(Method.POST, url,  
 Response.Listener { response ->  
 // response  
 val strResp2 = response.toString()  
 Log.d("API", strResp2)  
  
 },  
 Response.ErrorListener { error ->  
 Log.d("API", "error => $error")  
 }  
 ) {  
 override fun getBody(): ByteArray {  
 return requestBody.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.e("stringReq", stringReq.toString())  
 queue.add(stringReq)  
 Toast.makeText(  
 requireContext(),  
 "Usuario creado correctamente",  
 Toast.LENGTH\_SHORT  
 ).show()  
 val fragmentlogin = Login()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.FragContainer, fragmentlogin)  
 .addToBackStack(null)  
 .commit()  
 }  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar a la red", Toast.LENGTH\_SHORT  
 ).show()  
 Log.d("API", "error => $error")  
 }  
 )  
 Log.e("checkUserReq", checkUserReq.toString())  
 queue.add(checkUserReq)  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "Nombre: entre 6 y 32 caracteres, Contraseña: entre 4 y 18 caracteres",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 } else {  
 if (binding.RegCorreo.text.isEmpty() || binding.RegPass.text.isEmpty() || binding.RegConPass.text.isEmpty()) {  
 Toast.makeText(  
 requireContext(),  
 "No puede haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "Las contraseñas no coinciden",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 }  
 }  
 }  
}

En resumen, el código se encarga de registrar apropiadamente al usuario en la bd y en caso ed haber errores indicárselo al usuario.

## 5 Index

### 5.1 Inflator y generación de vistas

El fragmento anterior es el encargado de inicializar el la actividad del index, la cual es ejecutada mediante el método ‘onCreate’ y se infla mediante el ‘ActivityIndexBinding’ por lo que toma la referencia XML para aplicar el diseño, colores, constraints y demás elementos que se apliquen en la vista. Dentro del código está una función booleana llamada ‘isDarkMode’ la cual se encarga de verificar si el modo oscuro está activado por lo que se obtienen los tonos/colores ‘colorPrimaryVariant’ y ‘colorOnPrimary’ para aplicarse en el XML.

@SuppressLint("ResourceType")  
override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityIndexBinding.inflate(layoutInflater)  
 setContentView(binding.root)  
  
 val isDarkMode = intent.getBooleanExtra("isDarkMode", false)  
 if (isDarkMode) {  
 setTheme(R.style.Theme\_AppTheme)  
 binding.bottomBar.setBackgroundResource(R.drawable.n1bar)  
 } else {  
 setTheme(R.style.Theme\_AppTheme\_Light)  
 binding.bottomBar.setBackgroundResource(R.drawable.n5bar)  
 }  
  
 val themeAttrs = intArrayOf(com.google.android.material.R.attr.colorOnPrimary, com.google.android.material.R.attr.colorPrimaryVariant)  
  
 val typedArray = obtainStyledAttributes(themeAttrs)  
 val colorOnPrimary = typedArray.getColor(0, 0)  
 val colorPrimaryVariant = typedArray.getColor(1, 0)  
  
 binding.bottomBar.itemIconTintList = ColorStateList.valueOf(colorOnPrimary)  
 binding.bottomBar.itemTextColor = ColorStateList.valueOf(colorPrimaryVariant)  
 binding.bottomBar.outlineAmbientShadowColor = colorOnPrimary  
  
 typedArray.recycle()

En resumen, el fragmento anterior es el encargado de darle formato, diseño y color a las actividades tras ser inicializadas utilizando Kotlin y XML, cabe destacar que dicho fragmento es reutilizado dentro de todas las actividades por lo que únicamente se mencionará una única vez para evitar redundancia en el documento.

### 5.2 Navegación entre activitys y fragments

El código anterior consta de 2 partes, siendo la primera el selector de fragments dentro de la activity, este selector se encarga de corroborar que el fragment en el que se encuentra sea el adecuado y se muestre en el activity correspondiente esto mediante ‘currentView’ para corroborar la vista, ‘extraContainer’ para tener atrapado cualquier intent entre activitys y el ‘Log.i’ para corroboro el tema (oscuro/claro) a la hora de hacer las transiciones. La segunda parte es la navegación entre activitys, esta se realiza mediante una función llamada ‘offscreenPageLimit’ la cual permite tener x cantidad de fragmentos en la memoria, en este caso se tiene 5 los cuales son las distintas vistas de la aplicación, todo esto es corroborado por el ‘ViewPager’ por lo que al cambiar entre vistas el mismo se encarga de hacer la transición.

Log.i("MODO", isDarkMode.toString())  
  
 val extraContainer = intent.getIntExtra("currentView", 2)  
 currentView = extraContainer  
 val fragToGo = intent.getIntExtra("fragToGo", 0)  
 currentFrag = fragToGo  
  
 val sectionsPagerAdapter = SectionsPagerAdapter(this, supportFragmentManager, isDarkMode, currentView, currentFrag)  
 val viewPager: ViewPager = binding.ViewContainer  
 viewPager.adapter = sectionsPagerAdapter  
 viewPager.offscreenPageLimit = 5  
  
 val bottomNavigationView: BottomNavigationView = binding.bottomBar  
 bottomNavigationView.setOnNavigationItemSelectedListener { menuItem ->  
 when (menuItem.itemId) {  
 R.id.menu\_item\_perfil -> viewPager.currentItem = 0  
 R.id.menu\_item\_historial -> viewPager.currentItem = 1  
 R.id.menu\_item\_index -> viewPager.currentItem = 2  
 R.id.menu\_item\_pda -> viewPager.currentItem = 3  
 R.id.menu\_item\_finanzas -> viewPager.currentItem = 4  
 }  
 true  
 }  
 viewPager.addOnPageChangeListener(object : ViewPager.OnPageChangeListener {  
 override fun onPageScrolled(position: Int, positionOffset: Float, positionOffsetPixels: Int) {}  
 override fun onPageSelected(position: Int) {  
 bottomNavigationView.menu.getItem(position).isChecked = true  
 }  
 override fun onPageScrollStateChanged(state: Int) {}  
 })  
  
 viewPager.currentItem = currentView  
 }  
}

En resumen, este código es el encargado de mantener la navegación dentro e toda la aplicación así como la navegación entre los fragmentos dentro de una vista por lo que este código se recicla dentro de toda la aplicación y solo será referenciado una vez para evitar redundancia en el documento.

### 5.3 Incialización de index

Este Código define a la clase ‘indexmain’ el cual es un fragmento, este fragmento implementa la interfaz ‘OnChartValueSelectedListener’ para inicializar varias propiedades como lo son ‘medidaT’, ‘rango’, ‘switchval’ entre otros. Dentro de las inicializaciones se comprueban las notificaciones y se inicia el fragmento ‘indexGastosList()’.

Una vez se inicializan los elementos necesarios, se comprueba la conexión a internet mediante el objeto ‘Decoder’ el cual si existe una conexión lanza una corrutina ‘lifecycleScope.launch’ para configurar todos los colores con ‘setupcolors()’ y se inician procesos como las alarmas y notificaciones.

class indexmain : Fragment(), OnChartValueSelectedListener {  
 private lateinit var binding: FragmentIndexmainBinding  
 private val colorsI: MutableList<Int> = mutableListOf()  
  
 private val textI: MutableList<String> = mutableListOf()  
 private val numI: MutableList<Float?> = mutableListOf()  
 private val numG: MutableList<Float?> = mutableListOf()  
  
 private var medidaT: Double = 0.0  
 private var rango: Long = 15L  
 private var switchVal = false  
 private var lista: Fragment = indexGastosList()  
  
 private lateinit var fastable: List<Monto>  
 private var mutableEtiquetas: MutableList<String> = mutableListOf()  
 private var mutableIds: MutableList<Long> = mutableListOf()  
 private var mutableColores: MutableList<Int> = mutableListOf()  
  
 private lateinit var notificationHelper: notificationManager  
 private var notifActive = false  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 lifecycleScope.launch {  
 setupColors()  
 delay(10000)  
 setupAlarm()  
 }  
 }  
 notificationHelper = notificationManager(requireContext())  
  
 requireActivity().onBackPressedDispatcher.addCallback(  
 this,  
 object : OnBackPressedCallback(true) {  
 override fun handleOnBackPressed() {  
 }  
 })  
 }

En resumen, este fragmento se dedica a inicializar procesos y rutinas así como propiedades dentro del index.

### 5.4 Iniciación de apariencia

Este fragmento es encargado de 3 funciones de apariencia dentro del index las cuales son ‘isDarkModeEnabled’ para comprobar si el modo oscuro está activado y en caso de serlo poner el tema adecuado, ‘areNotiEnabled’ para comprobar si las notificaciones están activadas y en caso de serlo comprobar con la bd room las notificaciones a lanzar, accediendo mediante los DAO’s y por último ‘setupColors’ para ejecutar un ciclo suspendido que compruebe y le de colores a las etiquetas así como sus nombres.

Imagen 7, index

Diagrama

Descripción generada automáticamente

private suspend fun isDarkModeEnabled(context: Context): Boolean {  
 var komodo: Boolean  
 withContext(Dispatchers.IO) {  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 val mode = assetsDao.getTheme()  
 komodo = mode != 0  
 }  
 return komodo  
}  
  
private suspend fun areNotifEnabled(context: Context): Boolean {  
 var modo: Boolean  
 withContext(Dispatchers.IO){  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 val mode = assetsDao.getNotif()  
 modo = mode != 0  
 Log.v("NOTIF", "$modo")  
 }  
 return modo  
}

private fun setupColors() {  
 lifecycleScope.launch {  
 getLabels()  
  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V9))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V10))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V11))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V12))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V13))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V14))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V7))  
 colorsI.add(ContextCompat.getColor(requireContext(), R.color.V8))  
  
 textI.add("Salarios")  
 textI.add("Ventas")  
 textI.add("Becas")  
 textI.add("Pensiones")  
 textI.add("Manutención")  
 textI.add("Ingresos Pasivos")  
 textI.add("Regalos")  
 textI.add("Préstamos")  
 }  
}

En resumen, este fragmento se encarga de darle color a todo el activity así como obtener información para el funcionamiento de las etiquetas y notificaciones.

### 5.5 labels

Este fragmento trabaja con la función ‘getLabels’ la cual usa distintos hilos para la entrada y salida de datos en la bd room mediante los DAO’s, utilizando la clase ‘Stlite’ se llama a los métodos de las etiquetas para setear todos los colores y obtener los valores, esto con el fin de iterar desde 1 hasta el valor máximo de cada etiqueta para obtener el tamaño. La otra función utilizada es ‘seuqenceGet’ la cual utiliza 3 parámetros: ‘s’, ‘t’ y ‘f’, se realizan operaciones matemáticas con estas 3 variables para obtener los nombres de las etiquetas ya que están numeradas dentro de la base de datos, pero mediante esta función se traducen.

private suspend fun getLabels() {  
 withContext(Dispatchers.IO) {  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
  
 val ids = labelsDao.getIds()  
 val max = labelsDao.getMaxLabel()  
  
 for (i in 1..max) {  
 if (labelsDao.getPlabel(i) != ""){  
 mutableIds.add(labelsDao.getIdLabel(i))  
 mutableEtiquetas.add(labelsDao.getPlabel(i))  
 mutableColores.add(labelsDao.getColor(i))  
 }  
 }  
 Log.v("idl", "$mutableIds")  
 Log.v("plabel", "$mutableEtiquetas")  
 Log.v("color", "$mutableColores")  
 }  
}  
  
private fun sequenceGet(s: String, t: Int, f: Int?): Int {  
 val numeros = s.split(".").map { it.toIntOrNull() }  
 val frec = maxOf(f ?: 1, 1)  
 val last = maxOf(t/frec, 1)  
 val ultimosXNumeros = numeros.filterNotNull().takeLast(last)  
 return ultimosXNumeros.sum()  
}

En resumen, el código se encarga de ejecutar las etiquetas y tener una comunicación con la bd así como traducir las etiquetas para una mejor comprensión del usuario.

### 5.6 Setup PieChart gastos

El fragmento es el encargado de iniciar todos los valores y parámetros necesarios para el funcionamiento de la PieChart, el cual es un elemento visual utilizado el cual nos permite tener una especie de círculo el cual está segmentado con las etiquetas previamente inicializadas. Se incializa con la función ‘setupPieChartG’ la cual se comunica a la bd mediante ‘Dispatchers.IO’, después se ejecuta un ciclo que recolecta los gastos realizados durante x periodo de tiempo para almacenarlos en la lista ‘expenses’. Una vez se tienen todos los gastos se suman y acomodan dentro de cada etiqueta y se segmentan dentro de la PieChart para después asignase sus colores, texto y demás.

Finalmente se suman todos los gastos y se insertan en medio de la PieChart.

private suspend fun setupPieChartG(range: Long) {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val ingresosGastosDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
 val labelsDao = Stlite.getInstance((requireContext())).getLabelsDao()  
  
 val maxLabels = labelsDao.getMaxLabel()  
 val expenses = mutableListOf<List<Monto>>()  
  
 val fechaActual = LocalDate.now().toString()  
 val then = LocalDate.now().minusDays(range)  
 val formatoFecha = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(fechaActual)  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 for (i in 0 until range) {  
 val currentDate = then.plusDays(i)  
 val todayInt = currentDate.format(DateTimeFormatter.BASIC\_ISO\_DATE).toInt()  
 val dom = currentDate.dayOfMonth  
 val w = currentDate.dayOfWeek.value  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 Log.i("FECHAAA", "$todayInt")  
 for (j in 0..maxLabels) {  
 if (montoDao.getGR(todayInt, dom, dow, 100, j, todayInt).toString() != "[]") {  
 expenses.add(montoDao.getGR(todayInt, dom, dow, 100, j, todayInt))  
 }  
 }  
 }  
  
 for (j in 1..maxLabels) { //Para que no salte colores  
 expenses.add(listOf(Monto(idmonto=0, iduser=0, concepto="", valor=0.0, valorfinal=0.0, fecha=0, frecuencia=0, etiqueta=j, interes=0.0, veces=0L, estado=0, adddate=0, enddate = 0, cooldown = 0, sequence = "0.")))  
 }  
  
 Log.v("EXPENSES", expenses.toString())  
 val entries = mutableListOf<PieEntry>()  
  
 val totalIngresos = ingresosGastosDao.checkSummaryI()  
 Log.v("INGRESOS", totalIngresos.toString())  
 val totalGastos = ingresosGastosDao.checkSummaryG()  
 Log.v("GASTOS", totalGastos.toString())  
 val totalisimo = totalIngresos - totalGastos  
 Log.v("GRAN TOTAL", totalisimo.toString())  
  
 val decimalFormat = DecimalFormat("#.##")  
 val etiquetaSumMap = mutableMapOf<Long, Double>()  
 for (i in 0 until expenses.size) {  
 for (monto in expenses[i]) {  
 val etiqueta = monto.etiqueta.toLong()  
 val times = sequenceGet(monto.sequence, range.toInt(), monto.frecuencia)  
 val current = monto.valor \* times  
  
 if (etiquetaSumMap.containsKey(etiqueta)) {  
 val currentSum = etiquetaSumMap[etiqueta] ?: 0.0  
 etiquetaSumMap[etiqueta] = currentSum + current  
 } else {  
 etiquetaSumMap[etiqueta] = current  
 }  
 }  
 }  
  
 for ((\_, sum) in etiquetaSumMap) {  
 val percentI = if (totalGastos != 0.0) {  
 decimalFormat.format((sum.toFloat() \* 100 / totalGastos.toFloat())).toFloat()  
 } else {  
 0F  
 }  
 numG.add(percentI)  
 entries.add(PieEntry(sum.toFloat()))  
 }  
  
 val dataSet = PieDataSet(entries, "Gastos")  
 dataSet.colors = mutableColores  
  
 val data = PieData(dataSet)  
 binding.PieChart.data = data  
  
 binding.PieChart.centerText = "$${truncateDouble(totalIngresos)} - $${truncateDouble(totalGastos)} = $${truncateDouble(totalisimo)}"  
 binding.PieChart.setCenterTextSize(18f)  
 binding.PieChart.setCenterTextColor(R.color.white)  
 binding.PieChart.description.isEnabled = false  
 binding.PieChart.legend.isEnabled = false  
 }  
}

En resumen, esta función es la encargada de obtener los gastos y segmentarlos dentro de la PieChart.

### 5.7 setup PieChart ingresos

El fragmento actual es muy similar al fragmento de código anterior ya que realizan prácticamente la misma función, en este caso se utiliza la función ‘setupPieChartI’ la cual accede a la bd mediante los DAO’s para obtener los ingresos que se tuvo dentro de x rango de tiempo, después de obtenerlos se calculan los ingresos en cada sección y se calcula el porcentaje del círculo para la PieChart, una vez se ingresan a la PieChart se calcula el monto total de dinero restando el ingreso contra el gasto y se inserta en medio de la PieChart.

private suspend fun setupPieChartI(range: Long) {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val ingresosGastosDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
 val incomes = mutableListOf<List<Monto>>()  
  
 val fechaActual = LocalDate.now().toString()  
 val then = LocalDate.now().minusDays(range)  
 val formatoFecha = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(fechaActual)  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 for (i in 0 until range) {  
 val currentDate = then.plusDays(i)  
 val todayInt = currentDate.format(DateTimeFormatter.BASIC\_ISO\_DATE).toInt()  
 val dom = currentDate.dayOfMonth  
 val w = currentDate.dayOfWeek.value  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 Log.i("FECHAAA", "$todayInt")  
 for (j in 10001..10007) {  
 if (montoDao.getGR(todayInt, dom, dow, 100, j, todayInt).toString() != "[]") {  
 incomes.add(montoDao.getGR(todayInt, dom, dow, 100, j, todayInt))  
 }  
 }  
 }  
  
 for (j in 10001..10007) { //Para que no salte colores  
 incomes.add(listOf(Monto(idmonto=0, iduser=0, concepto="", valor=0.0, valorfinal=0.0, fecha=0, frecuencia=0, etiqueta=j, interes=0.0, veces=0L, estado=0, adddate=0, enddate = 0, cooldown = 0, sequence = "0.")))  
 }  
  
 Log.v("incomes", incomes.toString())  
 val entries = mutableListOf<PieEntry>()  
  
 val totalIngresos = ingresosGastosDao.checkSummaryI()  
 Log.v("INGRESOS", totalIngresos.toString())  
 val totalGastos = ingresosGastosDao.checkSummaryG()  
 Log.v("GASTOS", totalGastos.toString())  
 val totalisimo = totalIngresos - totalGastos  
 Log.v("GRAN TOTAL", totalisimo.toString())  
  
 val decimalFormat = DecimalFormat("#.##")  
 val etiquetaSumMap = mutableMapOf<Long, Double>()  
 for (i in 0 until incomes.size) {  
 for (monto in incomes[i]) {  
 val etiqueta = monto.etiqueta.toLong()  
 val times = sequenceGet(monto.sequence, range.toInt(), monto.frecuencia)  
 val current = monto.valor \* times  
  
 if (etiquetaSumMap.containsKey(etiqueta)) {  
 val currentSum = etiquetaSumMap[etiqueta] ?: 0.0  
 etiquetaSumMap[etiqueta] = currentSum + current  
 } else {  
 etiquetaSumMap[etiqueta] = current  
 }  
 }  
 }  
  
 for ((\_, sum) in etiquetaSumMap) {  
 val percentI = if (totalIngresos != 0.0) {  
 decimalFormat.format((sum.toFloat() \* 100 / totalIngresos.toFloat())).toFloat()  
 } else {  
 0F  
 }  
 numI.add(percentI)  
 entries.add(PieEntry(sum.toFloat()))  
 }  
  
 val dataSet = PieDataSet(entries, "Gastos")  
 dataSet.colors = colorsI  
  
 val data = PieData(dataSet)  
 binding.PieChart.data = data  
  
 binding.PieChart.centerText = "$${truncateDouble(totalIngresos)} - $${truncateDouble(totalGastos)} = $${truncateDouble(totalisimo)}"  
 binding.PieChart.setCenterTextSize(18f)  
 binding.PieChart.setCenterTextColor(R.color.white)  
 binding.PieChart.description.isEnabled = false  
 binding.PieChart.legend.isEnabled = false  
 }  
}

Resumiendo, la función se encarga de obtener, calcular e insertar todos los ingresos que se tuvo durante x tiempo dentro de la PieChart, además de dar un monto neto que se posee calculando los gastos e ingresos.

### 5.8 Redirects Index

El código es el encargado de permitir la navegación al interactuar con distintos elementos dentro del index como lo son los botones para agregar gasto, ingreso, el medidor de ahorro, entre otros.

Al presionar el reddirect corespondiente se ejecuta el método ‘onViewCreated’ y la corrutina ‘lifecycleScope’ para hacer la redirección, así como obtener una lista de datos ‘fastable’ mediante el ‘Decoder’ en caso de contar con conexión a internet.

También se setea un retraso de 500 milisegundos a la PieChart y gráficos en index para al momento de volver a la vista tener un mejor rendimiento.

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 val addWithSwitchOn = indexadd.newInstance(true)  
 val addWithSwitchOff = indexadd.newInstance(false)  
  
 lifecycleScope.launch {  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 procesarMontos()  
 }  
  
 fastable = fastget()  
 binding.displayCheck.adapter = MontoAdapter(fastable)  
 binding.displayCheck.animate()  
 .alpha(1f)  
 .translationX(0f)  
 .translationZ(0f)  
 .setDuration(300)  
 .start()  
 val mT = "$medidaT%"  
 binding.Medidor.text = mT  
 }  
 lifecycleScope.launch {  
 gi(switchVal, rango)  
 delay(500)  
 binding.GraficoPastel.alpha = 0f  
 }  
  
 binding.MedidorDeAhorroButton.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, lista).addToBackStack(null).commit()  
 }  
  
 binding.redirect.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexIngresosList()).addToBackStack(null).commit()  
 }  
  
 binding.Options.setOnClickListener {  
 binding.drawerLayout.openDrawer(binding.barrita)  
 }  
  
 binding.barrita.setNavigationItemSelectedListener { menuItem ->  
 when (menuItem.itemId) {  
 R.id.Mandados -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexmandados()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.porPagar -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexPorPagar()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.addMonto -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexadd()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.gastosList -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexGastosList()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.ingresosList -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, indexIngresosList()).addToBackStack(null).commit()  
  
 true  
 }  
  
 else -> false  
 }  
 }

En resumen, este código se encarga de darle navegación a la vista, así como buen rendimiento al navegar y procesar las animaciones.

### 5.9 Interacción de PieChart y medidor de ahorro

El Código se encarga de tener un listener en la PieChart, de manera que al ser presionado x fragmento dentro de la misma se haga una animación de selección dentro del gráfico. También se tiene un ‘IconSwitch’ el cual depende su estado (izquierda o derecha) muestra los gastos o ingresos en el gráfico, todo esto mediante la variable ‘switchVal’, además el medidor de ahorro funciona a la par de la gráfica ya que al presionar cierto fragmento del gráfico se inicializa el ‘SeekBar’ en el medidor para usar el ‘rango’ y mostrar el nivel de ahorro.

binding.PieChart.setOnChartValueSelectedListener(pieChartOnChartValueSelectedListener())  
  
 binding.SultanOfSwing.setCheckedChangeListener {  
 switchVal = when (binding.SultanOfSwing.checked) {  
 IconSwitch.Checked.LEFT -> {  
 true  
 }  
 IconSwitch.Checked.RIGHT -> {  
 false  
 }  
 else -> {  
 false  
 }  
 }  
 lifecycleScope.launch {  
 gi(switchVal, rango)  
 }  
 }  
  
 binding.RangoSeekbar.min = 1  
 binding.RangoSeekbar.max = 12  
 binding.RangoSeekbar.progress = 8  
 binding.RangoSeekbar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {  
 override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {  
 val range = when (progress) {  
 1, 2, 3, 4, 5, 6, 7 -> progress  
 8 -> 15  
 9 -> 30  
 10 -> 60  
 11 -> 180  
 12 -> 365  
 else -> rango  
 }  
 binding.Rango.setText(range.toString())  
 }  
  
 override fun onStartTrackingTouch(seekBar: SeekBar?) {}  
  
 override fun onStopTrackingTouch(seekBar: SeekBar?) {}  
 })  
  
 binding.Rango.addTextChangedListener(object : TextWatcher {  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}  
  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}  
  
 override fun afterTextChanged(s: Editable?) {  
 val text = s.toString()  
 if (text.isNotEmpty()) {  
 rango = text.toLong()  
 lifecycleScope.launch {  
 gi(switchVal, rango)  
 }  
 }  
 }  
 })  
  
 binding.AgregarIngresoButton.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, addWithSwitchOn).addToBackStack(null).commit()  
 }  
 binding.AgregarGastoButton.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.index\_container, addWithSwitchOff).addToBackStack(null).commit()  
 }  
 binding.MedidorDeAhorroButton.rotation = -90f  
}

En resumen, este fragmento da interacción al PieChart y animaciones para que el usuario tenga una interacción más entendible con el gráfico.

### 5.10 Calendario y actualización de PieChart

El fragmento de código trabaja con 2 funciones, la primera es ‘fastget()’ la cual mediante rutinas realiza operaciones síncronas y asíncronas mediante DAO’s sobre el ‘monto’. Después se obtiene el día actual para asociarlo mediante ‘binding.Calendario’ y en caso de ser ejecutada una consulta a la bd se almacenan los datos relacionados a la fecha en la variable ‘fastable’.

La segunda función ‘gi(switchVal: Boolean, range: Long)’ recibe 2 parámetros y dependiendo el parámetro son las operaciones que realiza. En ambas se actualizan el color, textos y etiquetas a graficar en la PieChart, la diferencia son las funciones para ejecutar (‘setupPieChartG’ y ‘setupPieChartI’) dependiendo cada parámetro.

Imagen 8, calendario

Diagrama

Descripción generada automáticamente

@SuppressLint("SetTextI18n")  
private suspend fun fastget(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
  
 val fechaActual = LocalDate.now().toString()  
 val today: Int = fechaActual.replace("-", "").toInt()  
  
 val formatoFecha = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(fechaActual)  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 val dom = calendar.get(Calendar.DAY\_OF\_MONTH)  
 val w = calendar.get(Calendar.DAY\_OF\_WEEK)  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 val yyy = calendar.get(Calendar.YEAR)  
 val mesesito = when (calendar.get(Calendar.MONTH)) {  
 0 -> "Enero"  
 1 -> "Febrero"  
 2 -> "Marzo"  
 3 -> "Abril"  
 4 -> "Mayo"  
 5 -> "Junio"  
 6 -> "Julio"  
 7 -> "Agosto"  
 8 -> "Septiembre"  
 9 -> "Octubre"  
 10 -> "Noviembre"  
 11 -> "Diciembre"  
 else -> "cualquier mes"  
 }  
 val semanita = when (dow) {  
 47 -> "Domingo"  
 41 -> "Lunes"  
 42 -> "Martes"  
 43 -> "Miércoles"  
 44 -> "Jueves"  
 45 -> "Viernes"  
 46 -> "Sábado"  
 else -> ""  
 }  
  
 binding.Calendario.text = "$semanita $dom de $mesesito"  
 val addd: Int = today  
  
 Log.i("DOM", dom.toString())  
 Log.i("DOW", dow.toString())  
  
 Log.i("todayyyy", today.toString())  
  
 fastable = montoDao.getGXFecha(today, dom, dow, 100, addd)  
 Log.i("ALL TODOLIST", fastable.toString())  
 }  
 return fastable  
}  
  
private fun gi(switchVal: Boolean, range: Long) {  
 Log.v("masomenos", switchVal.toString())  
 if (!switchVal) {  
 lifecycleScope.launch {  
 setupPieChartG(range)  
 lista = indexGastosList()  
 binding.PieChart.performClick()  
 binding.PieChart.animateY(1200, Easing.EaseInOutQuad)  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(resources.getColor(R.color.R1))  
 binding.MedidorDeAhorroButton.progressBackgroundTintList = ColorStateList.valueOf(resources.getColor(R.color.R1))  
 binding.searchforlabel.text = null  
 binding.searchforlabel.hint = "Gastos"  
 }  
 } else {  
 lifecycleScope.launch {  
 setupPieChartI(range)  
 lista = indexIngresosList()  
 binding.PieChart.performClick()  
 binding.PieChart.animateY(1200, Easing.EaseInOutQuad)  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(resources.getColor(R.color.G2))  
 binding.MedidorDeAhorroButton.progressBackgroundTintList = ColorStateList.valueOf(resources.getColor(R.color.G2))  
 binding.searchforlabel.text = null  
 binding.searchforlabel.hint = "Ingresos"  
 }  
 }  
}

En resumen, el código trabaja con la PieChart para actualizarla siempre que se interactue con el swich y dependiendo de la fecha a trabajar se obtienen datos de la bd con consultas.

### 5.11 Selección de etiquetas en PieChart

El fragment de Código trabaja con la función ‘truncateDouble’ para obtener el valor de un monto y redondearlo a 2 decimales para una mejor presentación en la gráfica, después se implementa la interfaz ‘OnChartValueSelectedListener’ para gestionar los eventos de la gráfica como lo son la selección de un valor/etiqueta o el cambio de gráfica (gastos e ingresos).

Al seleccionar una etiqueta se implementan los métodos ‘onValueSelected’ para resaltar dicho segmento de la gráfica o al desseleccionarlo o en su defecto no haber seleccionado ningún fragmento ‘onNothingSelected’.

private fun truncateDouble(value: Double): Double {  
 val decimalFormat = DecimalFormat("#.##")  
 return decimalFormat.format(value).toDouble()  
}  
  
inner class pieChartOnChartValueSelectedListener : OnChartValueSelectedListener {  
  
 @SuppressLint("SetTextI18n")  
 override fun onValueSelected(e: Entry, h: Highlight) {  
 Log.i("VAL SELECTED", "Value: " + e.y + ", index: " + h.x.toInt())  
  
 val chart = binding.PieChart  
  
 chart.highlightValue(h)  
  
 lista = if (!switchVal) {  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(mutableColores[h.x.toInt()])  
 binding.MedidorDeAhorroButton.progressTintList = ColorStateList.valueOf(mutableColores[h.x.toInt()])  
 binding.MedidorDeAhorroButton.progress = numG[h.x.toInt()]?.toInt() ?: 0  
 binding.Medidor.text = numG[h.x.toInt()].toString() + "%"  
 binding.searchforlabel.text = mutableEtiquetas[h.x.toInt()]  
 val iglinstance = indexGastosList.labelSearch(h.x.toInt())  
 iglinstance  
 } else {  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(colorsI[h.x.toInt()])  
 binding.MedidorDeAhorroButton.progressTintList = ColorStateList.valueOf(colorsI[h.x.toInt()])  
 binding.MedidorDeAhorroButton.progress = numI[h.x.toInt()]?.toInt() ?: 0  
 binding.Medidor.text = numI[h.x.toInt()].toString() + "%"  
 binding.searchforlabel.text = textI[h.x.toInt()]  
 val iilinstance = indexIngresosList.labelSearch(h.x.toInt())  
 iilinstance  
 }  
 }  
  
 override fun onNothingSelected() {  
 Log.i("PieChart", "nothing selected")  
 lista = if (!switchVal) {  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(resources.getColor(R.color.R1))  
 binding.MedidorDeAhorroButton.progressBackgroundTintList = ColorStateList.valueOf(resources.getColor(R.color.R1))  
 binding.searchforlabel.text = null  
 binding.searchforlabel.hint = "Gastos"  
 indexGastosList()  
 } else {  
 binding.MedidorDeAhorroButton.indeterminateTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.progressTintMode = PorterDuff.Mode.SRC\_IN  
 binding.MedidorDeAhorroButton.indeterminateTintList = ColorStateList.valueOf(resources.getColor(R.color.G2))  
 binding.MedidorDeAhorroButton.progressBackgroundTintList = ColorStateList.valueOf(resources.getColor(R.color.G2))  
 binding.searchforlabel.text = null  
 binding.searchforlabel.hint = "Ingresos"  
 indexIngresosList()  
 }  
 }  
}  
  
override fun onValueSelected(e: Entry, h: Highlight) {  
 Log.i("VAL SELECTED", "Value: " + e.y + ", index: " + h.x)  
}  
  
override fun onNothingSelected() {  
 Log.i("PieChart", "nothing selected")  
}

En resumen, este código se encarga de dar un correcto funcionamiento a la selección y desselección de las etiquetas o fragmentos de la PieChart mediante eventos de selección.

### 5.12 Suspend Montos

Es una función suspendida llamada ‘procesarMontos’ la cual se encarga de procesar todos los montos relacionados con gastos e ingresos los cuales se ejecutan mediante ‘Dispachers.IO’ usando los DAO’s para acceder a la base ‘Stlite’. Esta función también realiza cálculos de fecha par obtener montos de distintas fechas, así como frecuencia de los montos.

private suspend fun procesarMontos() {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val ingresoGastoDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
 val assetsDao = Stlite.getInstance(requireContext()).getAssetsDao()  
  
 val fechaActual = LocalDate.now().toString()  
 Log.d("HOY", fechaActual)  
 val today: Int = fechaActual.replace("-", "").toInt()  
 val prev = assetsDao.getLastProcess()  
  
 val formatoFecha = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(fechaActual)  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 val dom = calendar.get(Calendar.DAY\_OF\_MONTH)  
 val w = calendar.get(Calendar.DAY\_OF\_WEEK)  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 val addd: Int = today  
  
 Log.i("DOM", dom.toString())  
 Log.i("DOW", dow.toString())  
  
 Log.i("todayyyy", today.toString())  
 Log.i("prevvvvv", prev.toString())  
  
 val montos = montoDao.getMontoXFecha(today, dom, dow, 100, addd)  
  
 if (prev < today) {  
 for (monto in montos) {  
 if (monto.cooldown == 0) {  
 val totalIngresos = ingresoGastoDao.checkSummaryI()  
  
 Log.i("MONTO PROCESADO", monto.toString())  
 val weekMonto = monto.fecha  
 Log.v("wek", weekMonto.toString())  
  
 if (monto.etiqueta > 10000) {  
 ingresoGastoDao.updateSummaryI(monto.iduser.toInt(), totalIngresos + monto.valor)  
 monto.veces = monto.veces?.plus(1)  
 monto.sequence = monto.sequence + "1."  
 montoDao.updateMonto(monto)  
 } else {  
 var status = 0  
 var cooldown = 0  
 val sequence = monto.sequence + "0."  
 when (monto.estado) {  
 1 -> status = 0  
 4 -> status = 3  
 6 -> status = 5  
 9 -> status = 8  
 }  
 when (monto.frecuencia) {  
 14 -> cooldown = 1  
 61 -> cooldown = 1  
 91 -> cooldown = 2  
 122 -> cooldown = 3  
 183 -> cooldown = 5  
 365 -> cooldown = 11  
 }  
 val toCheckMonto = Monto(  
 idmonto = monto.idmonto,  
 iduser = monto.iduser,  
 concepto = monto.concepto,  
 valor = monto.valor,  
 valorfinal = monto.valorfinal,  
 fecha = monto.fecha,  
 frecuencia = monto.frecuencia,  
 etiqueta = monto.etiqueta,  
 interes = monto.interes,  
 tipointeres = monto.tipointeres,  
 veces = monto.veces,  
 estado = status,  
 adddate = monto.adddate,  
 enddate = monto.enddate,  
 cooldown = cooldown,  
 delay = monto.delay,  
 sequence = sequence  
 )  
 montoDao.updateMonto(toCheckMonto)  
 }  
 } else {  
 val newcool = monto.cooldown + 1  
  
 val toMeltMonto = Monto(  
 idmonto = monto.idmonto,  
 iduser = monto.iduser,  
 concepto = monto.concepto,  
 valor = monto.valor,  
 valorfinal = monto.valorfinal,  
 fecha = monto.fecha,  
 frecuencia = monto.frecuencia,  
 etiqueta = monto.etiqueta,  
 interes = monto.interes,  
 tipointeres = monto.tipointeres,  
 veces = monto.veces,  
 estado = monto.estado,  
 adddate = monto.adddate,  
 enddate = monto.enddate,  
 cooldown = newcool,  
 delay = monto.delay,  
 sequence = monto.sequence  
 )  
 montoDao.updateMonto(toMeltMonto)  
 }  
 }  
 }  
 assetsDao.updateLastprocess(today)  
 }  
}

En resumen, la función es la encargada de todos los movimientos financieros acerca de montos en comunicación con la base de datos.

### 5.13 Setup de alarmas

La función ‘setupAlarm’ es la encargada de configurar alarmas las cuales se repiten a la hora que se configure mediante el servicio de ‘AlarmManager’ de Android. La alarma que está programada en este fragmento la programa a las 3 AM, se realizan 3 intents ‘alarma’, ‘recordatorios’, ‘consejosFactory’ como punteros para la alarma.

private fun setupAlarm() {  
 val calendar = Calendar.getInstance()  
 calendar.timeInMillis = System.currentTimeMillis()  
 calendar.set(Calendar.HOUR\_OF\_DAY, 3)  
 calendar.set(Calendar.MINUTE, 0)  
 calendar.set(Calendar.SECOND, 0)  
  
 val alarmManager = requireContext().getSystemService(ALARM\_SERVICE) as AlarmManager  
  
 val alarma = Intent(requireContext(), Alarma::class.java)  
 val pendingAlarma = PendingIntent.getBroadcast(requireContext(), 0, alarma, FLAG\_IMMUTABLE)  
 val recordatorios = Intent(requireContext(), Recordatorios::class.java)  
 val pendingRecordatorios = PendingIntent.getBroadcast(requireContext(), 0, recordatorios, FLAG\_IMMUTABLE)  
 val consejosFactory = Intent(requireContext(), consejosFactory::class.java)  
 val pendingCF = PendingIntent.getBroadcast(requireContext(), 0, consejosFactory, FLAG\_IMMUTABLE)  
  
 alarmManager.setRepeating(  
 AlarmManager.RTC\_WAKEUP,  
 calendar.timeInMillis,  
 AlarmManager.INTERVAL\_DAY,  
 pendingAlarma  
 )  
 alarmManager.setRepeating(  
 AlarmManager.RTC\_WAKEUP,  
 calendar.timeInMillis,  
 AlarmManager.INTERVAL\_DAY,  
 pendingRecordatorios  
 )  
 alarmManager.setRepeating(  
 AlarmManager.RTC\_WAKEUP,  
 calendar.timeInMillis,  
 AlarmManager.INTERVAL\_DAY,  
 pendingCF  
 )  
}

En resumen, esta clase es la encargada de realizar una alarma a las 3 am para 3 intents dentro del index las cuales son la alarma, los recordatorios y los consejos.

### 5.14 Montos por pagar

La siguiente clase ‘MontoAdapter’ es la encargada de obtener, mostrar y administrar los montos por pagar, dichos montos son mostrados dentro de un ‘RecyclerView’ dentro del cual se encuentran ‘TextViews’ y ‘Buttons’. Dentro del RecyclerView se mostrarán los montos por pagar junto con los botones ‘Marcar como pagado’, ‘Posponer’ y ‘Omitir’, dichos montos son mostrados con distintos colores los cuales son tomados de la lista ‘mutableColores’ dependiendo de un rango de importancia (entre más cerca la fecha por pagar más crítico).

Imagen 9, montos por pagar

Patrón de fondo

Descripción generada automáticamente

private inner class MontoAdapter(private val montos: List<Monto>) :  
 RecyclerView.Adapter<MontoAdapter.MontoViewHolder>() {  
 inner class MontoViewHolder(  
 itemView: View,  
 val conceptoTextView: TextView,  
 val valorTextView: TextView,  
 val fechaTextView: TextView,  
 val updateM: Button,  
 val checkM: Button,  
 val delayM: Button,  
 val skipM: Button  
 ) : RecyclerView.ViewHolder(itemView)  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MontoViewHolder {  
 val itemView =  
 LayoutInflater.from(parent.context).inflate(R.layout.item\_porpagar, parent, false)  
 val conceptoTextView = itemView.findViewById<TextView>(R.id.pNombre)  
 val valorTextView = itemView.findViewById<TextView>(R.id.pValor)  
 val fechaTextView = itemView.findViewById<TextView>(R.id.pFecha)  
 val updateM = itemView.findViewById<Button>(R.id.editP)  
 val checkM = itemView.findViewById<Button>(R.id.checkP)  
 val delayM = itemView.findViewById<Button>(R.id.delayP)  
 val skipM = itemView.findViewById<Button>(R.id.skipP)  
 return MontoViewHolder(  
 itemView,  
 conceptoTextView,  
 valorTextView,  
 fechaTextView,  
 updateM,  
 checkM,  
 delayM,  
 skipM  
 )  
 }  
  
  
 override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = truncateDouble(monto.valor).toString()  
 holder.fechaTextView.text = monto.fecha.toString()  
 val upup = indexmontoupdate.sendMonto(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
 holder.updateM.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, upup).addToBackStack(null).commit()  
 }  
 holder.checkM.setOnClickListener {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres marcar el gasto ${monto.concepto} como pagado?")  
 .setPositiveButton("Confirmar") { dialog, \_ ->  
 lifecycleScope.launch {  
 montoCheck(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmain()).addToBackStack(null)  
 .commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 }  
 holder.delayM.setOnClickListener {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres posponer el gasto ${monto.concepto}?")  
 .setPositiveButton("Posponer") { dialog, \_ ->  
 lifecycleScope.launch {  
 delay(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmain()).addToBackStack(null)  
 .commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 }  
 holder.skipM.setOnClickListener {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres omitir el gasto ${monto.concepto} por hoy?")  
 .setPositiveButton("Confirmar") { dialog, \_ ->  
 lifecycleScope.launch {  
 skip(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmain()).addToBackStack(null)  
 .commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 }  
 if (position == 0){  
 holder.itemView.setBackgroundResource(R.drawable.p1topitem)  
 }  
 if (position == montos.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.p1bottomitem)  
 }  
 if (position == 0 && montos.size - 1 == 0){  
 holder.itemView.setBackgroundResource(R.drawable.p1onlyitem)  
 }  
 var colorin = 0x000000  
 for (i in mutableColores.indices) {  
 if (mutableIds[i] == monto.etiqueta.toLong()) colorin = mutableColores[i]  
 }  
 holder.itemView.setBackgroundColor(colorin)  
 if ((monto.estado ?: return) >= 5 && (monto.estado ?: return) <= 9) {  
 holder.itemView.setBackgroundColor(resources.getColor(R.color.R0))  
 }  
 }  
  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }

En resumen, esta función es la que permite tener los montos por pagar dentro del index, además de administrar su estado dentro de la base de datos.

### 5.15 Setup IndexAdd

El siguiente código se encarga de redirigirse desde ‘indexmain’ y después toma un valor booleano de ‘switchValue’ para verificar si iniciar la corrutina de agregar monto o ingreso, dependiendo de la bandera se llaman las funciones ‘masmenos’ y ‘getLabels’. Se crean 3 spinners con recursos predefinidos como lo son días, meses, años, montos, etiquetas, entre otros.

@SuppressLint("SetTextI18n")  
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 val back = indexmain()  
  
 val switchValue = arguments?.getBoolean(switchval) ?: false  
 lifecycleScope.launch {  
 masmenos(switchValue)  
 getLabels()  
 }  
  
 val adapterF = ArrayAdapter.createFromResource(  
 requireContext(),  
 R.array.frecuenciaoptions,  
 android.R.layout.simple\_spinner\_item  
 )  
 val adapterI = ArrayAdapter.createFromResource(  
 requireContext(),  
 R.array.tipooptions,  
 android.R.layout.simple\_spinner\_item  
 )  
 val arrayEtiquetas = mutableEtiquetas  
 val adapterG = ArrayAdapter(  
 requireContext(),  
 android.R.layout.simple\_spinner\_item,  
 arrayEtiquetas  
 )  
 adapterF.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 adapterG.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 adapterI.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
  
 binding.FrecuenciaField.adapter = adapterF  
 binding.FrecuenciaField.alpha = 0f  
 binding.FreqText.alpha = 0f  
  
 binding.goback.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 }  
  
 binding.updownSwitch.setCheckedChangeListener {  
 when (binding.updownSwitch.checked) {  
 IconSwitch.Checked.LEFT -> {  
 binding.ValorField.hint = "Ingreso"  
 binding.LabelField.adapter = adapterI  
 binding.yocreoquesi.text = "Préstamo"  
 esGasto = false  
 }  
 IconSwitch.Checked.RIGHT -> {  
 binding.ValorField.hint = "Gasto"  
 binding.LabelField.adapter = adapterG  
 binding.yocreoquesi.text = "Deuda"  
 esGasto = true  
 }  
 else -> {}  
 }  
 hideAll()  
 }  
  
 binding.LabelField.onItemSelectedListener = this  
  
 binding.FrecuenciaField.onItemSelectedListener = this  
  
 val max = 10000  
 val min = 0  
 binding.InteresSeekbar.max = max  
 binding.InteresSeekbar.min = min  
  
  
 binding.InteresSeekbar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {  
 override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {  
 val value = progress.toFloat() / 100  
 binding.InteresField.setText(value.toString())  
 }  
  
 override fun onStartTrackingTouch(seekBar: SeekBar?) {}  
  
 override fun onStopTrackingTouch(seekBar: SeekBar?) {}  
 })  
  
 binding.InteresField.addTextChangedListener(object : TextWatcher {  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}  
  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}  
  
 override fun afterTextChanged(s: Editable?) {  
 val text = s.toString()  
 if (text.isNotEmpty()) {  
 val value = text.toFloat() \* 100  
 binding.InteresSeekbar.progress = value.toInt()  
 }  
 }  
 })  
  
 binding.yocreoquesi.setOnClickListener {  
 if (binding.yocreoquesi.isChecked) {  
 displayFrecField()  
 displayInteresField()  
 displayFechaFinalField()  
 displayIC()  
 } else {  
 hideFrecField()  
 hideInteresField()  
 hideFechaFinalField()  
 hideIC()  
 }  
 }  
  
 binding.WeekField.selectionMode = SingleSelectionMode.create()  
 binding.WeekField.locale = Locale.getDefault()  
 binding.WeekField.setDaySelectionChangedListener { selectedDays ->  
 if (selectedDays.isNotEmpty()) {  
 val aux = selectedDays[0].name.lowercase()  
 selectedDay = when (aux) {  
 "monday" -> 41  
 "tuesday" -> 42  
 "wednesday" -> 43  
 "thursday" -> 44  
 "friday" -> 45  
 "saturday" -> 46  
 "sunday" -> 47  
  
 else -> {  
 40  
 }  
 }  
 Log.i("Día seleccionado", selectedDay.toString())  
 } else {  
 Log.i("Día seleccionado", "NONE")  
 }  
 }

En resumen, este fragmento es el encargado de dar formato a la selección de los datos necesarios para ingresar un gasto o ingreso.

### 5.16 Index Add Confirm

El siguiente Código refiere a 2 botones dentro de la vista, el primero es ‘confirm’ el cual al ser presionado realiza una serie de validaciones como lo son la verificación de ingresar los valores correctamente, tener frecuencia y parámetros válidos, entre otros. Acto seguido se pide una confirmación que tras ser presionada se realizan cálculos para el caso de las deudas, se calculan las tasas de interés y demás. El segundo botón es el botón de ‘Cancel’, este botón descarta los cambios y regresa al usuario al fragmento anterior ‘Index’.

Imagen 10, añadir monto

Texto

Descripción generada automáticamente con confianza baja

binding.Confirm.setOnClickListener {  
 val concepto = binding.ConceptoField.text.toString()  
 val valorstr = binding.ValorField.text.toString()  
 var veces = 0L  
  
 var interes = 0.0  
  
 if (label != 0 && concepto != "" && valorstr != "" && valorstr != "." && selectedDay != 39) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 var valor = valorstr.toDouble()  
 valor = truncateDouble(valor)  
 var valorfinal = valor  
  
 fecha = when (frecuencia) {  
 0 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaField.year  
 val datedate = "$year$fMonth$fDay"  
 val fsi: Int = datedate.replace("-", "").toInt()  
  
 fsi  
 } // Único  
 1 -> {  
 veces = 0  
 100  
 } // Diario  
 7, 14 -> {  
 selectedDay  
 } // Semanales  
 30, 61, 91, 122, 183, 365 -> {  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
  
 intday  
 } // Mensuales  
 else -> {  
 val intyear = binding.FechaField.year - 1900  
 Log.w("YEAR", intyear.toString())  
 val intmonth = binding.FechaField.month  
 Log.w("MONTH", intmonth.toString())  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
 val datedate = "$intyear$intmonth$intday"  
 Log.w("DATE", datedate)  
  
 datedate.toInt()  
 } // Catch que agarra la fecha actual  
 }  
  
 val adddateStr: String = LocalDate.now().toString()  
 val adddate = adddateStr.replace("-", "").toInt()  
  
 if (binding.yocreoquesi.isChecked) {  
 estado = 5  
 interes = binding.InteresField.text.toString().toDouble()  
 val day = binding.FechaFinalField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaFinalField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaFinalField.year  
 val datedate = "$year$fMonth$fDay"  
 enddate = datedate.replace("-", "").toInt()  
  
 val addY = adddate.toString().substring(0, 4).toInt()  
 val addM = adddate.toString().substring(4, 6).toInt()  
 val addD = adddate.toString().substring(6, 8).toInt()  
 val endY = enddate.toString().substring(0, 4).toInt()  
 val endM = enddate.toString().substring(4, 6).toInt()  
 val endD = enddate.toString().substring(6, 8).toInt()  
 val addedAt = LocalDate.of(addY, addM, addD)  
 val toEnd = LocalDate.of(endY, endM, endD)  
 Log.v("FECHASASAS", "$addedAt, $toEnd, ${ChronoUnit.DAYS.between(addedAt, toEnd)}")  
 val duracion: Int = when (frecuencia) {  
 0 -> { 1 }  
 1 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt() }  
 7 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/7 }  
 14 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/14 }  
 30 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt() }  
 61 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/2 }  
 91 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/3 }  
 122 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/4 }  
 183 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/6 }  
 365 -> { ChronoUnit.YEARS.between(addedAt, toEnd).toInt() }  
 else -> { 1 }  
 }  
  
 val tasa: Double = interes / duracion  
 val aux = valorfinal  
 tipointeres = if (binding.interesCompuesto.isChecked) {  
 valor = (aux / duracion) + aux \* (tasa / 100)  
 for (i in 0 until duracion) {  
 valorfinal += valorfinal \* (tasa / 100)  
 }  
  
 2  
 } else {  
 valorfinal += (valorfinal \* interes) / 100  
 valor = valorfinal / duracion  
  
 1  
 }  
 }  
  
 Log.v("Concepto", concepto)  
 Log.v("Valor", valor.toString())  
 Log.v("Fecha", fecha.toString())  
 Log.v("Frecuencia", frecuencia.toString())  
 Log.v("Etiqueta", label.toString())  
 Log.v("Interes", interes.toString())  
 Log.v("Veces", veces.toString())  
 Log.v("Addate", adddate.toString())  
 lifecycleScope.launch {  
 montoadd(  
 concepto,  
 valor,  
 valorfinal,  
 fecha,  
 frecuencia,  
 label,  
 interes,  
 tipointeres,  
 veces,  
 estado,  
 adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "No pueden haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 }  
  
 binding.Cancel.setOnClickListener {  
 val cancelDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres descartar cambios?")  
 .setPositiveButton("Descartar") { dialog, \_ ->  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 cancelDialog.show()  
 }  
  
 hideAll()  
}

En resumen, el fragmento de código se encarga de realizar los cálculos e insertar los cambios necesarios para el monto ingresado, así como cancelar la operación, todo esto mediante 2 botones.

### 5.17 Montos especiales

Las siguientes funciones son 2 funciones las cuales modifican al monto, en este caso ‘montoPapelera’ y ‘montoFavorito’ son funciones que modifican la bd room tomando los parámetros del monto como lo son el concepto, valor, fecha y demás. Ambas funciones utilizan ‘withContext(Dispatchers.IO)’ para poder hacer operaciones de entrada/salida mediante threads para comunicarse mediante el DAO. Se realiza una conversión del estado del monto creando atributos adicionales al monto como lo es ‘viejomonto’ con los valores nuevos y finalmente se obtienen estos montos mediante ‘montoDao.getMonto()’ para imprimir los montos en la tabla dependiendo del estatus seleccionado ‘ALL MONTOS’, ‘favoritos’ y ‘papelera’.

Imagen 11, favoritos

Forma

Descripción generada automáticamente

private suspend fun montoPapelera(  
 idmonto: Long,  
 concepto: String,  
 valor: Double,  
 fecha: Int?,  
 frecuencia: Int?,  
 etiqueta: Int,  
 interes: Double?,  
 veces: Long?,  
 estado: Int?,  
 adddate: Int  
) {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 var status = 2  
 when (estado) {  
 1 -> status = 2  
 6 -> status = 7  
 }  
  
 val enddate = montoDao.getEnded(idmonto.toInt())  
 val valorfinal = montoDao.getValorFinal(idmonto.toInt())  
 val tipointeres = montoDao.getTipoInteres(idmonto.toInt())  
 val delay = montoDao.getDelay(idmonto.toInt())  
 val sequence = montoDao.getSequence(idmonto.toInt())  
 val cooldown = montoDao.getCooldown(idmonto.toInt())  
 val iduser = usuarioDao.checkId().toLong()  
 val viejoMonto = Monto(  
 idmonto = idmonto,  
 iduser = iduser,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = veces,  
 estado = status,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = sequence  
 )  
  
 montoDao.updateMonto(viejoMonto)  
 val montos = montoDao.getMonto()  
 Log.i("ALL MONTOS", montos.toString())  
  
 }  
}  
  
private suspend fun montoFavorito(  
 idmonto: Long,  
 concepto: String,  
 valor: Double,  
 fecha: Int?,  
 frecuencia: Int?,  
 etiqueta: Int,  
 interes: Double?,  
 veces: Long?,  
 estado: Int?,  
 adddate: Int  
) {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val status = when (estado) {  
 0 -> 3  
 1 -> 4  
 5 -> 8  
 6 -> 9  
  
 3 -> 0  
 4 -> 1  
 8 -> 5  
 9 -> 6  
  
 else -> 3  
 }  
 val enddate = montoDao.getEnded(idmonto.toInt())  
 val valorfinal = montoDao.getValorFinal(idmonto.toInt())  
 val tipointeres = montoDao.getTipoInteres(idmonto.toInt())  
 val delay = montoDao.getDelay(idmonto.toInt())  
 val sequence = montoDao.getSequence(idmonto.toInt())  
 val cooldown = montoDao.getCooldown(idmonto.toInt())  
 val iduser = usuarioDao.checkId().toLong()  
 val viejoMonto = Monto(  
 idmonto = idmonto,  
 iduser = iduser,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = veces,  
 estado = status,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = sequence  
 )  
  
 montoDao.updateMonto(viejoMonto)  
 val montos = montoDao.getMonto()  
 Log.i("ALL MONTOS", montos.toString())  
  
 }  
}

En resumen, estas funciones se encargan de recopilar los montos en una tabla, así como modificar los montos para tener montos “especiales”.

### 5.18 Tabulación de montos

El siguiente código se encarga de tomar los valores de los ‘monto’ e insertarlos en un ‘ViewHolder’, usa la clase ‘Decoder’ para darle formato a valores como la moneda y fecha. Cuenta con botones para interactuar con los montos para marcar y desmarcar como favorito, eliminar, editar, entre otros.

Finalmente se establece un fondo a la tabla y usando la función ‘getItemCount’ se define el tamaño total para la lista y se ejecuta.

Imagen 12, gastos

Texto

Descripción generada automáticamente con confianza media

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 var tempstat = 5  
 val decoder = Decoder(requireContext())  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.fechaTextView.text = monto.fecha?.let { decoder.date(it) }  
 lifecycleScope.launch {  
 holder.etiquetaTextView.text = decoder.label(monto.etiqueta)  
 }  
  
 if (monto.estado == 0 || monto.estado == 1 || monto.estado == 5 || monto.estado == 6) {  
 tempstat = 5  
 }  
 if (monto.estado == 3 || monto.estado == 4 || monto.estado == 8 || monto.estado == 9) {  
 tempstat = 8  
 }  
  
 holder.optionsM.setOnClickListener {  
 val popupMenu = PopupMenu(requireContext(), holder.optionsM, Gravity.END)  
 val inflater = popupMenu.menuInflater  
 inflater.inflate(R.menu.options\_item\_tabla, popupMenu.menu)  
 popupMenu.setOnMenuItemClickListener { item: MenuItem ->  
 when (item.itemId) {  
 R.id.action\_favMonto -> {  
 if (tempstat == 5) {  
 tempstat = 8  
 }  
 if (tempstat == 8) {  
 tempstat = 5  
 }  
 lifecycleScope.launch {  
 montoFavorito(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
 }  
  
 true  
 }  
 R.id.action\_verMonto -> {  
 val ver = verMonto.sendMonto(monto.idmonto)  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, ver).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_editMonto -> {  
 val upup = indexmontoupdate.sendMonto(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, upup).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_deleteMonto -> {  
 if (monto.estado == 3 || monto.estado == 4 || monto.estado == 8 || monto.estado == 9 || tempstat == 8) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("El monto ${monto.concepto} no se puede eliminar porque está marcado como favorito")  
 .setPositiveButton("Aceptar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres enviar el monto ${monto.concepto} a la papelera?")  
 .setPositiveButton("Eliminar") { dialog, \_ ->  
  
 Log.v("Id del monto actualizado", monto.idmonto.toString())  
 Log.v("Concepto", monto.concepto)  
 Log.v("Valor", monto.valor.toString())  
 Log.v("Fecha", monto.fecha.toString())  
 Log.v("Frecuencia", monto.frecuencia.toString())  
 Log.v("Etiqueta", monto.etiqueta.toString())  
 Log.v("Interes", monto.interes.toString())  
 Log.v("Veces", monto.veces.toString())  
 lifecycleScope.launch {  
 montoPapelera(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, indexmain())  
 .addToBackStack(null)  
 .commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 }  
  
 true  
 }  
 else -> false  
 }  
 }  
 popupMenu.show()  
 }  
 if (position == gastos.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.p1bottomcell)  
 }  
 }  
  
  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }  
 }  
}

En resumen, el código se encarga de insertar los montos en una tabla para que el usuario pueda visualizarlos.

### 5.19 Tabulación ingresos

El siguiente código Kotlin es el encargado de tomar los valores de los ingresos de la base de datos, tomar los ‘Monto’, ‘Usuario’ e ‘IngresosGastos’ para graficarlos dentro de la tabla junto con los botones como se vio en el punto 5.18 pero en este caso se tabulan los ingresos.

Imagen 13, ingresos

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

suspend fun fup(  
 id: Long,  
 concepto: String,  
 valor: Double,  
 fecha: Int?,  
 frecuencia: Int?,  
 etiqueta: Int,  
 interes: Double?,  
 veces: Long?,  
 adddate: Int  
 ) {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresoGastoDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
  
 var nv: Long? = 1  
 if (veces != null)  
 nv = veces + 1  
  
 val sequence = montoDao.getSequence(id.toInt())  
 val values = sequence.trim('.').split('.').map { it.toInt() }.toMutableList()  
 if (values.isNotEmpty()) {  
 val lastIndex = values.size - 1  
 values[lastIndex] += 1  
 }  
 val updatedString = values.joinToString(".")  
 val result = "$updatedString."  
  
 val enddate = montoDao.getEnded(id.toInt())  
 val valorfinal = montoDao.getValorFinal(id.toInt())  
 val tipointeres = montoDao.getTipoInteres(id.toInt())  
 val delay = montoDao.getDelay(id.toInt())  
 val cooldown = montoDao.getCooldown(id.toInt())  
 val iduser = usuarioDao.checkId().toLong()  
 val montoPresionado = Monto(  
 idmonto = id,  
 iduser = iduser,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = nv,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = result  
 )  
  
 val totalIngresos = ingresoGastoDao.checkSummaryI()  
  
 ingresoGastoDao.updateSummaryI(  
 montoPresionado.iduser.toInt(),  
 totalIngresos + montoPresionado.valor  
 )  
 montoDao.updateMonto(montoPresionado)  
 val montos = montoDao.getMonto()  
 Log.i("ALL MONTOS", montos.toString())  
 }  
 }  
  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }  
 }  
}

En resumen, este fragmento es muy similar al punto anterior, la única diferencia es que en lugar de trabajar con los gastos trabaja con los ingresos.

### 5.20 Preparación de mandados

El siguiente código trabaja con 2 funciones, la primera ‘fastget’ es la encargada de usar corrutinas para manejar los threads y mantener una comunicación efectiva con la bd mediante DAO’s, después se calcula la fecha actual y se convierte en un número entero para almacenarlo en ‘today’ esto con la finalidad de consultar a la bd y almacenar el resultado en ‘fastable’. La segunda función ‘MontoAdapter’ es la encargada de generar un ‘RecyclerView’ y ‘ViewHolder’ para los elementos de la vista, en este caso los mandados.

private suspend fun fastget(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
  
 val fechaActual = LocalDate.now().toString()  
 val today: Int = fechaActual.replace("-", "").toInt()  
 val addd: Int = today  
  
 Log.i("todayyyy", today.toString())  
  
 fastable = montoDao.getGXFecha(100, 100, 100, 100, addd)  
 Log.i("ALL TODOLIST", fastable.toString())  
 }  
 return fastable  
}  
  
private inner class MontoAdapter(private val montos: List<Monto>) :  
 RecyclerView.Adapter<MontoAdapter.MontoViewHolder>() {  
 inner class MontoViewHolder(  
 itemView: View,  
 val vecesTextView: TextView,  
 val conceptoTextView: TextView,  
 val valorTextView: TextView,  
 val skipButton: ImageButton,  
 val delayButton: ImageButton  
 ) : RecyclerView.ViewHolder(itemView)  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MontoViewHolder {  
 val itemView =  
 LayoutInflater.from(parent.context).inflate(R.layout.item\_fastadd, parent, false)  
 val vecesTextView = itemView.findViewById<TextView>(R.id.fastVeces)  
 val conceptoTextView = itemView.findViewById<TextView>(R.id.fastNombre)  
 val valorTextView = itemView.findViewById<TextView>(R.id.fastValor)  
 val skipButton = itemView.findViewById<ImageButton>(R.id.skipMonto)  
 val delayButton = itemView.findViewById<ImageButton>(R.id.delayMonto)  
 return MontoViewHolder(itemView, vecesTextView, conceptoTextView, valorTextView, skipButton, delayButton)  
 }  
  
  
 override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 val decoder = Decoder(requireContext())  
  
 if (monto.delay >= 1) holder.itemView.setBackgroundResource(R.drawable.fastshapedelayed)  
 if (monto.delay >= 3) holder.itemView.setBackgroundResource(R.drawable.fastshapeurgent)  
 holder.itemView.outlineAmbientShadowColor = mutableColores[monto.delay]  
 holder.itemView.outlineSpotShadowColor = mutableColores[monto.delay]  
  
 holder.itemView.setOnClickListener {  
 lifecycleScope.launch {  
 fup(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmandados()).addToBackStack(null)  
 .commit()  
 }  
 }  
 holder.vecesTextView.text = monto.veces.toString()  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.skipButton.setOnClickListener {  
 lifecycleScope.launch {  
 skip(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmandados()).addToBackStack(null)  
 .commit()  
 }  
 }  
 holder.delayButton.setOnClickListener {  
 lifecycleScope.launch {  
 delay(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexmandados()).addToBackStack(null)  
 .commit()  
 }  
 }  
 }

En resumen, estas funciones funcionan en conjunto para poder generar un campo de trabajo para los mandados, tener una lista y poder accionar en ella.

### 5.21 Mandados

El siguiente código trabaja con una función suspendida ‘fup’ la cual manipula la bd para tomar varios parámetros, esto con la finalidad de obtener un ‘id’ de los montos a mostrar como mandados, esto incluyendo los parámetros de los montos como lo son ‘valor’, ‘fecha’, ‘etiqueta’ entre otros.

Con estos parámetros se hace una serie de cálculos con la finalidad de dar la lista de prioridad en los mandados y plasmarlos dentro del ‘ViewHolder’.

Imagen 14, mandados

Patrón de fondo

Descripción generada automáticamente

suspend fun fup(  
 id: Long,  
 concepto: String,  
 valor: Double,  
 fecha: Int?,  
 frecuencia: Int?,  
 etiqueta: Int,  
 interes: Double?,  
 veces: Long?,  
 estado: Int?,  
 adddate: Int  
) {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresoGastoDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
  
 var valorfinal = montoDao.getValorFinal(id.toInt())  
 if (estado == 5 || estado == 8){  
 valorfinal -= valor  
 }  
  
 var nv: Long? = 1  
 if (veces != null)  
 nv = veces + 1  
  
 var status = estado  
 var delay = montoDao.getDelay(id.toInt())  
 if (delay != 0) {  
 delay -= 1  
 delay = maxOf(delay, 0)  
 } else {  
 when (estado) {  
 0 -> status = 1  
 3 -> status = 4  
 5 -> status = 6  
 8 -> status = 9  
 }  
 }  
 var cooldown = 0  
 when (frecuencia) {  
 14 -> cooldown = 1  
 61 -> cooldown = 1  
 91 -> cooldown = 2  
 122 -> cooldown = 3  
 183 -> cooldown = 5  
 365 -> cooldown = 11  
 }  
  
 val sequence = montoDao.getSequence(id.toInt())  
 val values = sequence.trim('.').split('.').map { it.toInt() }.toMutableList()  
 if (values.isNotEmpty()) {  
 val lastIndex = values.size - 1  
 values[lastIndex] += 1  
 }  
 val updatedString = values.joinToString(".")  
 val result = "$updatedString."  
 val tipointeres = montoDao.getTipoInteres(id.toInt())  
 val enddate = montoDao.getEnded(id.toInt())  
 val iduser = usuarioDao.checkId().toLong()  
 val montoPresionado = Monto(  
 idmonto = id,  
 iduser = iduser,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = nv,  
 estado = status,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = result  
 )  
  
 val totalGastos = ingresoGastoDao.checkSummaryG()  
  
 ingresoGastoDao.updateSummaryG(  
 montoPresionado.iduser.toInt(),  
 totalGastos + montoPresionado.valor  
 )  
 montoDao.updateMonto(montoPresionado)  
 val montos = montoDao.getMonto()  
 Log.i("ALL MONTOS", montos.toString())  
 }  
}

### 5.22 Update Monto

El código utiliza las mismas funciones y métodos que se utilizan al momento de crear un monto nuevo, en este caso se utilizan para modificar los valores del monto previamente establecido, esto mediante los spinners de frecuencia y tipos, se utiliza ‘lifecycleScope.launch’ para modificar los valores de los spinners y finalmente se usa ‘WeekField’ para modificar la fecha del monto.

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 val back = indexmain()  
  
 binding.ConceptoField.setText(arguments?.getString(concep))  
 binding.InteresField.setText(arguments?.getDouble(interes.toString()).toString())  
  
 val aux = arguments?.getDouble(valu)  
 var switchValue = false  
 if (aux != null) {  
 if (aux >= 0){switchValue = true}  
 }  
  
 lifecycleScope.launch {  
 masmenos(switchValue)  
 getLabels()  
 }  
  
 val adapterF = ArrayAdapter.createFromResource(  
 requireContext(),  
 R.array.frecuenciaoptions,  
 android.R.layout.simple\_spinner\_item  
 )  
 val adapterI = ArrayAdapter.createFromResource(  
 requireContext(),  
 R.array.tipooptions,  
 android.R.layout.simple\_spinner\_item  
 )  
 val arrayEtiquetas = mutableEtiquetas  
 val adapterG = ArrayAdapter(  
 requireContext(),  
 android.R.layout.simple\_spinner\_item,  
 arrayEtiquetas  
 )  
 adapterF.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 adapterG.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 adapterI.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
  
 binding.FrecuenciaField.adapter = adapterF  
 binding.FrecuenciaField.alpha = 0f  
 binding.FreqText.alpha = 0f  
  
 binding.goback.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, planesdeahorromain()).addToBackStack(null).commit()  
 }  
  
 binding.updownSwitch.setCheckedChangeListener {  
 when (binding.updownSwitch.checked) {  
 IconSwitch.Checked.LEFT -> {  
 binding.ValorField.hint = "Ingreso"  
 binding.LabelField.adapter = adapterI  
 binding.yocreoquesi.text = "Préstamo"  
 }  
 IconSwitch.Checked.RIGHT -> {  
 binding.ValorField.hint = "Gasto"  
 binding.LabelField.adapter = adapterG  
 binding.yocreoquesi.text = "Deuda"  
 }  
 else -> {}  
 }  
 hideAll()  
 }  
  
 binding.LabelField.onItemSelectedListener = this  
  
 binding.FrecuenciaField.onItemSelectedListener = this  
  
 val max = 10000  
 val min = 0  
 binding.InteresSeekbar.max = max  
 binding.InteresSeekbar.min = min  
  
  
 binding.InteresSeekbar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {  
 override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {  
 val value = progress.toFloat() / 100  
 binding.InteresField.setText(value.toString())  
 }  
  
 override fun onStartTrackingTouch(seekBar: SeekBar?) {}  
  
 override fun onStopTrackingTouch(seekBar: SeekBar?) {}  
 })  
  
 binding.InteresField.addTextChangedListener(object : TextWatcher {  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}  
  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}  
  
 override fun afterTextChanged(s: Editable?) {  
 val text = s.toString()  
 if (text.isNotEmpty()) {  
 val value = text.toFloat() \* 100  
 binding.InteresSeekbar.progress = value.toInt()  
 }  
 }  
 })  
  
 binding.yocreoquesi.setOnClickListener {  
 if (binding.yocreoquesi.isChecked) {  
 displayFrecField()  
 displayInteresField()  
 displayFechaFinalField()  
 displayIC()  
 } else {  
 hideFrecField()  
 hideInteresField()  
 hideFechaFinalField()  
 hideIC()  
 }  
 }  
  
 binding.WeekField.selectionMode = SingleSelectionMode.create()  
 binding.WeekField.locale = Locale.getDefault()  
 binding.WeekField.setDaySelectionChangedListener { selectedDays ->  
 if (selectedDays.isNotEmpty()) {  
 var aux = selectedDays[0].name.toLowerCase()  
 selectedDay = when (aux) {  
 "monday" -> 41  
 "tuesday" -> 42  
 "wednesday" -> 43  
 "thursday" -> 44  
 "friday" -> 45  
 "saturday" -> 46  
 "sunday" -> 47  
  
 else -> {40}  
 }  
 Log.i("Día seleccionado", selectedDay.toString())  
 } else {  
 Log.i("Día seleccionado", "NONE")  
 }  
 }

En resumen, este fragmento se encarga de permitir la manipulación de los valores previamente establecidos de un monto.

### 5.23 Confirmar y Cancelar Update

Al igual que al momento de ingresar un monto se cuenta con 2 botones, siendo estos confirmar y cancelar como se vio previamente, al momento de presionar confirmar se pide una confirmación ‘confirmDialog’ el cual tras ser efectuado se corrobora que no existan campos vacíos y efectivamente se haya cambiado algún campo para después guardar dichos cambios dentro de la bd mediante la función ‘montoupdate’ la cual está dentro de una corrutina. Después de hacer el cambio se navega de regreso a la vista de las tablas de montos mediante el ‘FragmentManager’. Mismo caso, el botón de ’cancel’ pide una confirmación la cual al ser presionada descarta dichos cambios y se navega a la visa de tabla de montos.

binding.Confirm.setOnClickListener {  
 val concepto = binding.ConceptoField.text.toString()  
 val valorstr = binding.ValorField.text.toString()  
 var veces = 0L  
  
 var interes = 0.0  
  
 if (label != 0 && concepto != "" && valorstr != "" && valorstr != "." && selectedDay != 39) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 var valor = valorstr.toDouble()  
 valor = truncateDouble(valor)  
 var valorfinal = valor  
  
 fecha = when (frecuencia) {  
 0 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaField.year  
 val datedate = "$year$fMonth$fDay"  
 val fsi: Int = datedate.replace("-", "").toInt()  
  
 fsi  
 } // Único  
 1 -> {  
 veces = 0  
 100  
 } // Diario  
 7, 14 -> {  
 selectedDay  
 } // Semanales  
 30, 61, 91, 122, 183, 365 -> {  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
  
 intday  
 } // Mensuales  
 else -> {  
 val intyear = binding.FechaField.year - 1900  
 Log.w("YEAR", intyear.toString())  
 val intmonth = binding.FechaField.month  
 Log.w("MONTH", intmonth.toString())  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
 val datedate = "$intyear$intmonth$intday"  
 Log.w("DATE", datedate)  
  
 datedate.toInt()  
 } // Catch que agarra la fecha actual  
 }  
  
 val adddateStr: String = LocalDate.now().toString()  
 val adddate = adddateStr.replace("-", "").toInt()  
  
 if (binding.yocreoquesi.isChecked) {  
 estado = 5  
 interes = binding.InteresField.text.toString().toDouble()  
 val day = binding.FechaFinalField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaFinalField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaFinalField.year  
 val datedate = "$year$fMonth$fDay"  
 enddate = datedate.replace("-", "").toInt()  
  
 val addY = adddate.toString().substring(0, 4).toInt()  
 val addM = adddate.toString().substring(4, 6).toInt()  
 val addD = adddate.toString().substring(6, 8).toInt()  
 val endY = enddate.toString().substring(0, 4).toInt()  
 val endM = enddate.toString().substring(4, 6).toInt()  
 val endD = enddate.toString().substring(6, 8).toInt()  
 val addedAt = LocalDate.of(addY, addM, addD)  
 val toEnd = LocalDate.of(endY, endM, endD)  
 Log.v("FECHASASAS", "$addedAt, $toEnd, ${ChronoUnit.DAYS.between(addedAt, toEnd)}")  
 val duracion: Int = when (frecuencia) {  
 0 -> { 1 }  
 1 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt() }  
 7 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/7 }  
 14 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/14 }  
 30 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt() }  
 61 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/2 }  
 91 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/3 }  
 122 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/4 }  
 183 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/6 }  
 365 -> { ChronoUnit.YEARS.between(addedAt, toEnd).toInt() }  
 else -> { 1 }  
 }  
  
 val tasa: Double = interes / duracion  
 val aux = valorfinal  
 tipointeres = if (binding.interesCompuesto.isChecked) {  
 valor = (aux / duracion) + aux \* (tasa / 100)  
 for (i in 0 until duracion) {  
 valorfinal += valorfinal \* (tasa / 100)  
 }  
  
 2  
 } else {  
 valorfinal += (valorfinal \* interes) / 100  
 valor = valorfinal / duracion  
  
 1  
 }  
 }  
  
 Log.v("Id", id.toString())  
 Log.v("Concepto", concepto)  
 Log.v("Valor", valor.toString())  
 Log.v("Fecha", fecha.toString())  
 Log.v("Frecuencia", frecuencia.toString())  
 Log.v("Etiqueta", label.toString())  
 Log.v("Interes", interes.toString())  
 Log.v("Veces", veces.toString())  
 Log.v("Addate", adddate.toString())  
 lifecycleScope.launch {  
 montoupdate(  
 monto.idmonto,  
 concepto,  
 valor,  
 valorfinal,  
 fecha,  
 frecuencia,  
 label,  
 interes,  
 tipointeres,  
 veces,  
 estado,  
 adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, planesdeahorromain()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "No pueden haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
  
 }  
  
 binding.Cancel.setOnClickListener {  
 val cancelDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres descartar cambios?")  
 .setPositiveButton("Descartar") { dialog, \_ ->  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, back).addToBackStack(null).commit()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, planesdeahorromain()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 cancelDialog.show()  
 }  
  
 hideAll()  
}

En resumen, estas 2 clases funcionan igual que al momento de agregar los montos solo que en este caso se utiliza para modificar el monto.

### 5.24 Por pagar

El siguiente código es el encargado de permitir la visualización y manipulación de los montos por pagar, estos se muestran de manera exactamente igual a los ‘mandados’, solo que en este caso son catalogados como ‘monto’ por pagar dentro de la bd room. Siguen siendo mostrados mediante ‘onBindViewHolder’ para mostrar la lista y cuenta con el color ajustable según la prioridad del monto.

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 val decoder = Decoder(requireContext())  
 if (monto.delay >= 1) holder.itemView.setBackgroundResource(R.drawable.fastshapedelayed)  
 if (monto.delay >= 3) holder.itemView.setBackgroundResource(R.drawable.fastshapeurgent)  
 holder.itemView.outlineAmbientShadowColor = mutableColores[monto.delay]  
 holder.itemView.outlineSpotShadowColor = mutableColores[monto.delay]  
  
 holder.itemView.setOnClickListener {  
 lifecycleScope.launch {  
 fup(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexPorPagar()).addToBackStack(null)  
 .commit()  
 }  
 }  
 holder.vecesTextView.text = monto.veces.toString()  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.skipButton.setOnClickListener {  
 lifecycleScope.launch {  
 skip(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexPorPagar()).addToBackStack(null)  
 .commit()  
 }  
 }  
 holder.delayButton.setOnClickListener {  
 lifecycleScope.launch {  
 delay(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.index\_container, indexPorPagar()).addToBackStack(null)  
 .commit()  
 }  
 }  
}

En resumen, este fragmento funciona exactamente igual que los ‘mandado’, la única diferencia es al momento de mostrarlos ya que son mostrados en distintos fragments.

## 6 Perfil

### 6.1 Bajar foto

La función ‘bajarfoto’ utiliza ‘withContext(Dispatchers.IO)’ para realizar carga de la imagen de perfil mediante un thread, esta función utiliza la biblioteca de carga de imágenes ‘Coil’ la cual carga dicha imagen mediante una URL ‘link’ ubicada en la vista de perfil ‘binding.ProfilePicture’. Cabe destacar que para tener el mejor rendimiento posible de la aplicación se delimitan el tiempo de descarga ‘placeholder(R.drawable.ic\_Person)’ y se escala la imagen ‘Scale.FILL’ para obtener una imagen que cubra todo el ícono.

private suspend fun bajarfoto(link: String) {  
 withContext(Dispatchers.IO) {  
 binding.ProfilePicture.load(link) {  
 crossfade(true)  
 placeholder(R.drawable.ic\_person)  
 transformations(CircleCropTransformation())  
 scale(Scale.FILL)  
 }  
 }  
}

En resumen, la función es la encargada de bajar del servidor la imagen de perfil del usuario utilizando la librería Coil.

### 6.2 Mostrar datos del usuario

La función ‘mostrardatos’ se encarga de tomar todos los datos del usuario que toma de la bd mediante los DAO’s usando threads de manera que toma todos los datos desde su información personal, su balance total, días ahorrando, grupos y demás. Mediante el uso de ‘binding’ se asegura que la información mostrada sea la correcta en caso de que se actualice y mediante la función ‘bajarfoto’ se obtiene y muestra la imagen de perfil.

Imagen 15, datos usuario

Diagrama

Descripción generada automáticamente con confianza baja

private suspend fun mostrarDatos() {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresosGastosDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
  
 val totalIngresos = ingresosGastosDao.checkSummaryI()  
 val totalGastos = ingresosGastosDao.checkSummaryG()  
 val totalisimo = totalIngresos - totalGastos  
 val decimalFormat = DecimalFormat("#.##")  
 val balance = "${decimalFormat.format(totalisimo)}$"  
 val nombre = usuarioDao.checkName()  
 val edad = usuarioDao.checkAge()  
 val lachamba = usuarioDao.checkChamba()  
 val foto = usuarioDao.checkFoto()  
 val diasaho = usuarioDao.checkDiasaho()  
 usuarioDao.updateBalance(usuarioDao.checkId(), totalisimo)  
  
 var chamba = ""  
 val c = String.format("%06d", lachamba).toCharArray()  
 if (c[0] == '1') {  
 chamba += "asalariado, "  
 }  
 if (c[1] == '2') {  
 chamba += "vendedor, "  
 }  
 if (c[2] == '3') {  
 chamba += "pensionado, "  
 }  
 if (c[3] == '4') {  
 chamba += "becado, "  
 }  
 if (c[4] == '5') {  
 chamba += "mantenido, "  
 }  
 if (c[5] == '6') {  
 chamba += "inversionista, "  
 }  
  
 if (chamba.isNotEmpty()) {  
 chamba = chamba.dropLast(2)  
 chamba = chamba.replaceFirstChar { it.uppercaseChar() }  
 }  
  
 Log.v("Name", nombre)  
 Log.v("Age", edad.toString())  
 Log.v("Código de Chamba", lachamba.toString())  
 Log.v("Descripción de Chamba", chamba)  
 Log.v("Foto ", foto)  
 Log.v("Diasaho", diasaho.toString())  
 Log.v("Balance", balance)  
  
 binding.UsernameTV.text = nombre  
 if (edad != 0) {  
 binding.AgeTV.text = buildString {  
 append(edad.toString())  
 append(" años")  
 }  
 } else {  
 binding.AgeTV.text = buildString {  
 append("Edad sin registrar")  
 }  
 }  
  
 binding.OcupationTV.text = buildString {  
 append(chamba)  
 }  
 binding.DaysSavingButton.text = buildString {  
 append("¡")  
 append(diasaho.toString())  
 append(" días ahorrando!")  
 }  
 binding.BalanceTV.text = buildString {  
 append("Balance: ")  
 append(balance)  
 }  
 binding.BalanceTV.outlineAmbientShadowColor = resources.getColor(R.color.X0)  
 binding.BalanceTV.outlineSpotShadowColor = resources.getColor(R.color.X0)  
 if (diasaho > 0) {  
 binding.ahorrando.setColorFilter(resources.getColor(R.color.O2), PorterDuff.Mode.SRC\_IN)  
 }  
 if (diasaho > 14) {  
 binding.ahorrando.setColorFilter(resources.getColor(R.color.B2), PorterDuff.Mode.SRC\_IN)  
 }  
 if (diasaho > 60) {  
 binding.ahorrando.setColorFilter(resources.getColor(R.color.P2), PorterDuff.Mode.SRC\_IN)  
 }  
  
 val linkfoto = "http://savetrack.com.mx/images/$nombre.jpg"  
 Log.v("Link foto", linkfoto)  
 lifecycleScope.launch {  
 bajarfoto(linkfoto)  
 }  
 }  
}

En resumen, esta función se encarga de mostrar todos los datos del usuario dentro del perfil.

### 6.3 Display Adapter Grupos

Dentro del perfil se muestran los grupos mediante el uso de ‘GruposDisplayAdapter’, clase interna que se utiliza para proporcionar los datos a un ‘RecylcerView’.

private inner class GruposDisplayAdapter (private val grupos: List<Grupos>) :  
 RecyclerView.Adapter<GruposDisplayAdapter.GrupoViewHolder>() {  
 inner class GrupoViewHolder(  
 itemView: View,  
 val nombreTextView: TextView,  
 val tipoImage: ImageView  
 ) : RecyclerView.ViewHolder(itemView)  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GrupoViewHolder {  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.item\_displaygrupos, parent, false)  
 val nombreTextView = itemView.findViewById<TextView>(R.id.GNombre)  
 val tipoImage = itemView.findViewById<ImageView>(R.id.GTipo)  
 return GrupoViewHolder(  
 itemView,  
 nombreTextView,  
 tipoImage  
 )  
 }

En resumen, este fragmento de código permite mostrar y agrupar los distintos grupos a los que pertenezca el usuario.

### 6.4 Grupos del usuario

El código utiliza una función suspendida ‘procesarGrupos’ el cual toma los grupos nativos en la bd room mediante los DAO’s y se ejecuta una solicitud HTTP para obtener la información actualizada del grupo desde la URL del grupo almacenado dentro del servidor, se obtiene un JSON como respuesta y tras analizar las condiciones de los campos se actualiza al grupo local con los nuevos datos, caso contrario se notifica que el grupo está vacío. Finalmente se actualiza la lista de los grupos.

Imagen 16, grupos

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

private suspend fun procesarGrupos(context: Context) {  
 withContext(Dispatchers.IO) {  
 withContext(Dispatchers.IO) {  
 val gruposDao = Stlite.getInstance(context).getGruposDao()  
 val grupos = gruposDao.getAllGrupos()  
  
 for ((Id, nameg, description, type, admin, idori, color, enlace) in grupos) {  
 if (Id != null && nameg != null) {  
 val viejoGrupo = Grupos(  
 Id = Id,  
 nameg = nameg,  
 description = description,  
 type = type,  
 admin = admin,  
 idori = idori,  
 color = color,  
 enlace = enlace  
 )  
  
 val grupoJson = withContext(Dispatchers.IO) { JSONObject(URL("http://savetrack.com.mx/grupoGet.php?localid=${viejoGrupo.idori}&admin=${viejoGrupo.admin}").readText()) }  
  
 if (grupoJson.getLong("idgrupoglobal") != null && grupoJson.getInt("tipo") != 2) {  
 val idoriA: Long = grupoJson.getLong("idgrupolocal")  
 val adminA: Long = grupoJson.getLong("idadmin")  
 val namegA: String = grupoJson.optString("nombre")  
 val descA: String = grupoJson.optString("descripcion")  
 val tipoA: Int = grupoJson.optInt("tipo")  
 val colorA: Int = grupoJson.optInt("color")  
  
 val grupoActualizado = Grupos(  
 Id = Id,  
 nameg = namegA,  
 description = descA,  
 type = tipoA,  
 admin = adminA,  
 idori = idoriA,  
 color = colorA,  
 enlace = enlace  
 )  
  
 gruposDao.updateGrupo(grupoActualizado)  
 } else {  
 Log.v("Current grupo", "VACÍO")  
 }  
 val grup = gruposDao.getAllGrupos()  
 Log.i("ALL GRUPOS ALARMA", grup.toString())  
 }  
 }  
 }  
 }  
 }  
}

En resumen, este fragmento se encarga de mantener actualizados los grupos, así como mostrarlos adecuadamente en el perfil.

### 6.5 Cambiar la foto sin internet

El siguiente fragmento de código funciona tras presionar la foto de perfil dentro de la vista editar perfil, tras presionarlo se instancia el ‘Decoder’ para comprobar la conexión a internet mediante el método ‘hayNet()’ de manera que si hay conectividad se invoca la función ‘showFileChooser()’ para permitir que el usuario seleccione una foto del sistema, pero en caso que no exista conexión a internet se arroja un toast acerca de la falta.

binding.agregarfotobtn.setOnClickListener {  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 showFileChooser()  
 } else {  
 Toast.makeText(requireContext(), "No hay acceso a internet", Toast.LENGTH\_SHORT).show()  
 }  
}

En resumen, este fragmento permite subir una foto de perfil desde la galería del teléfono siempre y cuando se cuente con acceso a internet.

### 6.6 Preparar upload foto

El código trabaja con una actividad de selección ‘onActivityResult’ la cual primero verifica si el código de solicitud para la selección de imagen fue exitoso, que los datos no sean nulos y se tenga la dirección de la imagen, una vez se obtiene todo esto se carga la imagen como bitmap y se comprime usando la función ‘compressImage’, después se convierte en una cadena codificada en base64 mediante la función ‘getStringImage’ para posteriormente enviar la imagen mediante ‘sendImage’ y una vez se suba la imagen se actualiza la imagen en el perfil del usuario mediante la función ‘cambiarFoto’, cabe destacar que todo este código es realizado mediante corrutinas de manera que permita al usuario continuar utilizando la aplicación mientras que se sube la imagen.

@Deprecated("Deprecated in Java")  
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 super.onActivityResult(requestCode, resultCode, data)  
 if (requestCode == pickImageRequest && resultCode == RESULT\_OK && data != null && data.data != null) {  
 val filePath: Uri? = data.data  
 try {  
 val bitmap = MediaStore.Images.Media.getBitmap(requireContext().contentResolver, filePath)  
 val fo = filePath.toString()  
  
 val compressedBitmap = compressImage(bitmap)  
  
 val image: String = getStringImage(compressedBitmap)  
 Log.d("bytes", compressedBitmap.byteCount.toString())  
 Log.d("actual image size", image.length.toString())  
  
 lifecycleScope.launch{  
 sendImage(image, username)  
 }  
 lifecycleScope.launch{  
 cambiarFoto(fo)  
 }  
 } catch (e: IOException) {  
 e.printStackTrace()  
 }  
 }  
}  
private fun getStringImage(bmp: Bitmap): String {  
 val baos = ByteArrayOutputStream()  
 bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos)  
 val imageBytes: ByteArray = baos.toByteArray()  
 return Base64.encodeToString(imageBytes, Base64.DEFAULT)  
}  
private fun compressImage(image: Bitmap): Bitmap {  
 val maxFileSize = 75000  
 val tempo: String = getStringImage(image)  
 val outputStream = ByteArrayOutputStream()  
 image.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)  
 var compressedSize = tempo.length  
  
 var quality = 90  
 while (compressedSize > maxFileSize && quality > 10) {  
 outputStream.reset()  
 image.compress(Bitmap.CompressFormat.JPEG, quality, outputStream)  
 compressedSize = outputStream.toByteArray().toString().length  
 Log.v("reducing size to", compressedSize.toString())  
 quality -= 10  
 }  
  
 val compressedBitmap = BitmapFactory.decodeByteArray(outputStream.toByteArray(), 0, outputStream.size())  
 outputStream.close()  
  
 return compressedBitmap  
}  
private suspend fun cambiarFoto(nfo: String) {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
  
 val id = usuarioDao.checkId()  
 usuarioDao.updatePhoto(id, nfo)  
  
 lifecycleScope.launch{  
 bajarfoto(nfo)  
 }  
 }  
}

En resumen, este fragmento es el encargado de asegurar que la foto de perfil sea actualizada de manera adecuada sin sobresaturar los hilos para que la aplicación siga funcionando adecuadamente.

### 6.7 Upload foto

La función suspendida ‘sendImage’ es la encargada de enviar la imagen previamente preparada al servidor mediante trozos. Utilizando ‘Dispatchers.IO’ se prepara un ‘looper’ para manipular los mensajes, se establece la URL base y el tamaño de los trozos de la imagen, se registran en los logs de manera que se verifique que la cantidad de trozos sea menor o igual a 200. Una vez este dividida la imagen se comienza a iterar la URL con los distintos trozos y se comienzan a mandar bajo los parámetros del usuario, el índice del trozo y su cantidad, todo esto mediante PUT. Entre cada solicitud hay un retraso de 10 segundos para evitar saturar el servidor. En caso que la imagen no pueda ser dividida en 200 o menos fragmentos se notifica mediante un toast al usuario que la imagen es demasiado grande.

private suspend fun sendImage(image: String, username: String) {  
 withContext(Dispatchers.IO) {  
 Looper.prepare()  
 val baseUrl = "http://savetrack.com.mx/images/chunkpic.php"  
 val chunkSize = 2000  
  
 val chunks = image.chunked(chunkSize)  
 Log.w("TAMAÑO", (chunks.size \* chunkSize).toString())  
 Log.w("CHUNKS", chunks.size.toString())  
  
 if (chunks.size <= 200) {  
 for ((index, i) in chunks.indices.withIndex()) {  
 val chunk = chunks[i]  
 val params = hashMapOf<String, String>()  
 params["username"] = username  
 params["chunkIndex"] = index.toString()  
 params["totalChunks"] = chunks.size.toString()  
 params["imageChunk"] = chunk  
 val url =  
 "$baseUrl?username=$username&chunkIndex=$i&totalChunks=${chunks.size}&imageChunk=$chunk"  
 Log.v("CurrURL", url)  
 val stringRequest = object : StringRequest(  
 Method.PUT, url,  
 Response.Listener { response ->  
 Log.d("UPLOAD SUCCESS", i.toString())  
 Log.i("response", response)  
 },  
 Response.ErrorListener { error ->  
 Log.e("UPLOAD API ERROR", error.toString())  
 }) {  
 @Throws(AuthFailureError::class)  
 override fun getParams(): Map<String, String> {  
 val paramo: MutableMap<String, String> = Hashtable()  
 paramo["image"] = image  
 return paramo  
 }  
 }  
  
 val socketTimeout = 45000  
 val policy: RetryPolicy = DefaultRetryPolicy(  
 socketTimeout,  
 DefaultRetryPolicy.DEFAULT\_MAX\_RETRIES,  
 DefaultRetryPolicy.DEFAULT\_BACKOFF\_MULT  
 )  
 stringRequest.retryPolicy = policy  
 val requestQueue = Volley.newRequestQueue(requireContext())  
 requestQueue.add(stringRequest)  
  
 delay(10000)  
 }  
 } else {  
 binding.agregarfotobtn.clear()  
 Toast.makeText(requireContext(), "La imagen es demasiado grande, intente con una más pequeña", Toast.LENGTH\_SHORT).show()  
 }  
  
 }  
}

En resumen, este código es el encargado de asegurar que la imagen sea subida ed manera correcta al servidor y en caso contario se le notifique al usuario.

### 6.8 Guardar cambios

La función suspendida ‘guardarCambios’ utiliza múltiples corrutinas con la finalidad de: actualizar los datos del usuario que hayan sido modificados, establecer un array el cual funciona como ‘autómata finito’ de manera que al establecer valores en la situación financiera del usuario se cree una cadena que muestre su situación, un ejemplo sería que si el usuario es del tipo 1,3 sea un usuario asalariado y que cuenta con beca. Al momento de querer actualizar y guardar los datos se cuenta con 2 validaciones, la primera es que la edad el usuario debe de ser igual o mayor a 5 años y la meta de ahorro debe de ser mayor a 0, una vez se ejecutan todos los cambios se arroja un toast que indica que los cambios fueron efectuados y se cancelan las corrutinas para liberar los hilos.

private suspend fun guardarCambios() {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
  
 val idt = usuarioDao.checkId()  
 val nuevaEdad = binding.AgeeditperfTV.text.toString()  
 val nuevaMeta = binding.GoaleditperfTV.text.toString()  
  
 // ARBOL DE DECISIONES PARA CADA CASO DE CHAMBA  
 // AUTÓMATA FINITO o algo así  
 /\*  
 CHAMBAS  
 1. Salario  
 2. Vendedor  
 3. Pensionado  
 4. Becado  
 5. Mantenido  
 6. Inversionista  
 \*/  
  
 val c = arrayOf("0", "0", "0", "0", "0", "0")  
  
 if (binding.chamba1.isChecked) {  
 c[0] = "1"  
 }  
 if (binding.chamba2.isChecked) {  
 c[1] = "2"  
 }  
 if (binding.chamba3.isChecked) {  
 c[2] = "3"  
 }  
 if (binding.chamba4.isChecked) {  
 c[3] = "4"  
 }  
 if (binding.chamba5.isChecked) {  
 c[4] = "5"  
 }  
 if (binding.chamba6.isChecked) {  
 c[5] = "6"  
 }  
  
 var cFinal = ""  
 for (v in c) {  
 cFinal += v  
 }  
  
 if (cFinal == "000000"){  
 chambachanged = false  
 }  
  
 val nuevaChamba = cFinal.toLong()  
  
 val edadlong = if (nuevaEdad.isNotEmpty()) {  
 nuevaEdad.toLong()  
 } else {  
 0  
 }  
 if (edadlong <= 5 || edadlong >= 121 || nuevaEdad == "") {  
 edadchanged = false  
 withContext(Dispatchers.Default) {  
 Looper.prepare()  
 Toast.makeText(requireContext(), "La edad ingresada no es válida", Toast.LENGTH\_SHORT).show()  
 }  
 ok = 0  
 Dispatchers.Default.cancel()  
 } else {  
 ok = 1  
 }  
 val metadouble = if (nuevaMeta.isNotEmpty()) {  
 nuevaMeta.toDouble()  
 } else {  
 0.0  
 }  
 if (metadouble <= 0.0 || nuevaMeta == "") {  
 metachanged = false  
 ok = 0  
 Dispatchers.Default.cancel()  
 } else {  
 ok = 1  
 }  
  
 if (edadchanged){  
 usuarioDao.updateAge(idt, edadlong)  
 }  
 if (chambachanged){  
 usuarioDao.updateChamba(idt, nuevaChamba)  
 }  
 if (metachanged){  
 usuarioDao.updateMeta(idt, metadouble)  
 }  
  
 withContext(Dispatchers.Main) {  
 if (edadchanged || chambachanged){  
 Toast.makeText(  
 requireContext(),  
 "Cambios guardados correctamente",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
 Dispatchers.Main.cancel()  
 }  
 }  
 }  
}

En resumen, el código trabaja en distintos hilos para realizar múltiples tareas simultáneamente de manera que garantiza que las verificaciones necesarias sean ejecutadas, así como la interpretación del autómata sea correcta.

### 6.9 Get grupos

La actividad ‘getGrupo’ se encarga de obtener información sobre los grupos mediante URL utilizando un intent, usando el método ‘onCreate’ se recupera la información recabada y se extraen los parámetros. Después se lanza una corrutina que utiliza las bibliotecas ‘JSONObject’ y ‘JSONArray’ para poder realizar requests a 2 URLs y obtener información sobre el grupo y sus miembros. Dentro de estos detalles vienen el nombre, tipo de grupo, color y miembros.

class getGrupo : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_grupos)  
  
 val intentData: Uri? = intent.data  
 if (intentData != null) {  
 val localid = intentData.getQueryParameter("zxcd125s5d765e7wqa87sdftgh")  
 val admin = intentData.getQueryParameter("mnhjkmnbg1yhb3vdrtgvc98swe")  
  
 lifecycleScope.launch {  
 try {  
 val grupoJson = withContext(Dispatchers.IO) { JSONObject(URL("http://savetrack.com.mx/grupoGet.php?localid=$localid&admin=$admin").readText()) }  
 val miembrosJSON = withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/gruposMiembrosGet.php?localid=$localid&admin=$admin").readText()) }  
 val miembrosG = Array(miembrosJSON.length()) { miembrosJSON.getInt(it) }  
  
 if (grupoJson.getLong("idgrupoglobal") != null && grupoJson.getInt("tipo") != 2) {  
 val idori: Long = grupoJson.getLong("idgrupolocal")  
 val idadmin: Long = grupoJson.getLong("idadmin")  
 val nombre: String = grupoJson.optString("nombre")  
 val desc: String = grupoJson.optString("descripcion")  
 val tipo: Int = grupoJson.optInt("tipo")  
 val color: Int = grupoJson.optInt("color")  
 val enlace: Long = grupoJson.optLong("enlace")  
  
 grupoAdd(nombre, desc, tipo, color, idadmin, idori, enlace, miembrosG)  
 } else {  
 Log.v("Current grupo", "VACÍO")  
 }  
 } catch (e: Exception) {  
 Log.e("NetworkError", "Error during network call", e)  
 }  
 }  
 }

En resumen, el fragment de código se encarga de obtener toda la información acerca de los grupos utilizando JSON y URL.

### 6.10 Agregar grupos y usuarios

La función suspendida ‘grupoAdd’ realiza las operaciones relacionadas a la adición de un grupo en la aplicación, estas operaciones se ejecutan dentro de un thread para permitir que el usuario pueda seguir navegando dentro de la aplicación. Mediante ‘Stlite’ se obtienen las etiquetas y usuarios relacionados con los grupos y tras agregar el grupo se realiza una consulta HTTP a través de Volley para agregar a los miembros al grupo con un PUT lleno con los datos del usuario, todo eso siempre y cuando el usuario no esté en el grupo, en caso de que ya esté se notificará que ya pertenece al grupo.

private suspend fun grupoAdd(  
 nombre: String,  
 descripcion: String,  
 type: Int,  
 color: Int,  
 admin: Long,  
 idori: Long,  
 enlace: Long,  
 miembros: Array<Int>  
 ) {  
 withContext(Dispatchers.IO) {  
 val gruposDao = Stlite.getInstance(this@getGrupo).getGruposDao()  
 val usuarioDao = Stlite.getInstance(this@getGrupo).getUsuarioDao()  
 val labelsDao = Stlite.getInstance(this@getGrupo).getLabelsDao()  
  
 val iduser = usuarioDao.checkId()  
 if (!miembros.contains(iduser)) {  
 val nuevosMiembros: MutableList<Int> = mutableListOf()  
 for (element in miembros) {  
 nuevosMiembros.add(element)  
 }  
  
 val nuevoGrupo = Grupos(  
 nameg = nombre,  
 description = descripcion,  
 type = type,  
 admin = admin,  
 idori = idori,  
 color = color,  
 enlace = enlace  
 )  
 gruposDao.insertGrupo(nuevoGrupo)  
  
 val nuevaLabel = Labels(  
 plabel = nombre,  
 color = color  
 )  
 labelsDao.insertLabel(nuevaLabel)  
  
 val idlabel = labelsDao.getMaxLabel().toLong()  
 val gId = gruposDao.getMaxGrupo().toLong()  
 val grupoUp = Grupos(  
 Id = gId,  
 nameg = nombre,  
 description = descripcion,  
 type = type,  
 admin = admin,  
 idori = idori,  
 color = color,  
 enlace = idlabel  
 )  
 gruposDao.updateGrupo(grupoUp)  
  
 val queue = Volley.newRequestQueue(this@getGrupo)  
 var url = "http://savetrack.com.mx/gruposMiembrosPut.php?"  
 nuevosMiembros.add(iduser)  
 val uniqueMiembros = nuevosMiembros.toSet().toList()  
 val jsonMiembros = JSONArray(uniqueMiembros)  
  
 Log.v("JSONMIEMBROS", "$jsonMiembros")  
 Log.v("MUTABLELISTMIEMBROS", "$nuevosMiembros")  
 val requestBody = "localid=$idori&admin=$admin&miembros=$jsonMiembros"  
 url += requestBody  
 val stringReq: StringRequest =  
 object : StringRequest(  
 Method.PUT, url,  
 Response.Listener { response ->  
 val strResp2 = response.toString()  
 Log.d("API", strResp2)  
  
 },  
 Response.ErrorListener { error ->  
 Log.d("API", "error => $error")  
 }  
 ) {  
 override fun getBody(): ByteArray {  
 return requestBody.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.e("stringReq", stringReq.toString())  
 queue.add(stringReq)  
  
 val grupos = gruposDao.getAllGrupos()  
 val labels = labelsDao.getAllLabels()  
 Log.i("ALL GRUPOS", grupos.toString())  
 Log.i("ALL LABELS", labels.toString())  
 } else {  
 Toast.makeText(this@getGrupo, "Ya formas parte del grupo", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 }  
}

En resumen, esta actividad es la encargada de permitir el acceso a los usuarios dentro de los grupos agregándolos en la bd.

### 6.11 Lista de grupos

La clase ‘GrupoAdapter’ es una clase con constructor que recibe una lista de objetos de tipo ‘Grupos’ el cual utiliza ‘GrupoViewHolder’ para representar de manera individual pero dentro de la lista cada uno de los grupos a los cuales pertenece el usuario.

Imagen 17, visualizargrupos

Imagen que contiene Aplicación

Descripción generada automáticamente

private inner class GrupoAdapter(private val grupos: List<Grupos>) :  
 RecyclerView.Adapter<GrupoAdapter.GrupoViewHolder>() {  
 inner class GrupoViewHolder(  
 itemView: View,  
 val grupo: TextView,  
 val tipo: TextView,  
 val color: ImageView,  
 val verGrupo: Button  
 ) : RecyclerView.ViewHolder(itemView)  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GrupoViewHolder {  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.item\_grupo, parent, false)  
 val grupo = itemView.findViewById<TextView>(R.id.GNombre)  
 val tipo = itemView.findViewById<TextView>(R.id.GTipo)  
 val color = itemView.findViewById<ImageView>(R.id.GColor)  
 val verGrupo = itemView.findViewById<Button>(R.id.viewGrupo)  
 return GrupoViewHolder(  
 itemView,  
 grupo,  
 tipo,  
 color,  
 verGrupo  
 )  
 }  
  
  
 override fun onBindViewHolder(holder: GrupoViewHolder, position: Int) {  
 val grupo = grupos[position]  
 holder.grupo.text = grupo.nameg  
 holder.tipo.text = when (grupo.type) {  
 0 -> "Fijo"  
 1 -> "Temporal"  
 else -> "Eliminado"  
 }  
 holder.color.setBackgroundColor(grupo.color)  
 holder.verGrupo.setOnClickListener {  
 val verGrupo = montosGrupo.grupoSearch(grupo.Id)  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.GruposContainer, verGrupo).addToBackStack(null).commit()  
 }  
 }  
  
  
 override fun getItemCount(): Int {  
 Log.v("size de Grupossss", grupos.size.toString())  
 return grupos.size  
 }  
 }  
}

En resumen, la actividad se encarga de recibir los grupos a los que pertenece el usuario y graficarlos en una lista.

### 6.12 Vista de grupo

El fragmento de código tiene el fragmento ‘FragmentViewgrupoBinding’ a través del cual infla la interfaz, en la que se llama a la función ‘colorPicker()’ para establecer el color del grupo. Mediante una corrutina se solicitan mediante HTTP información del grupo como lo es el nombre, color, administrador, miembros, entre otros. Además se verifica que el usuario siga siendo parte del grupo, de manera que si el mismo ya no forma parte del mismo es redirigido y se le muestra un toast indicando que ya no forma parte del grupo.

Imagen 18, vista grupo

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

@SuppressLint("SetTextI18n")  
override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View {  
 binding = FragmentViewgrupoBinding.inflate(inflater, container, false)  
 colorPicker()  
 lifecycleScope.launch {  
 val isDarkMode = isDarkModeEnabled(requireContext())  
  
 if (isDarkMode) {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil2)  
 } else {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil)  
 }  
  
 Log.i("MODO", isDarkMode.toString())  
  
 getGrupo()  
 try {  
 val jsonObjectG = withContext(Dispatchers.IO) { JSONObject(URL("http://savetrack.com.mx/grupoGet.php?localid=$idori&admin=$admin").readText()) }  
 val miembrosJSON = withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/gruposMiembrosGet.php?localid=$idori&admin=$admin").readText()) }  
 val miembrosG = Array(miembrosJSON.length()) { miembrosJSON.getInt(it) }  
 //val idgglobal: Long = jsonObjectG.getLong("idgrupoglobal")  
 //val idglocal: String = jsonObjectG.getString("idgrupolocal")  
 //val idadmin: Long = jsonObjectG.optLong("idadmin")  
  
 val nombre: String = jsonObjectG.optString("nombre")  
 val tipo: Int = jsonObjectG.optInt("tipo")  
 val desc: String = jsonObjectG.optString("descripcion")  
 color = jsonObjectG.optInt("color")  
 val created: String = jsonObjectG.optString("created")  
 val createdString = "Creado el $created"  
 val adminName = withContext(Dispatchers.IO) { URL("http://savetrack.com.mx/usernameget.php?id=$admin").readText() }  
  
 if (!miembrosG.contains(iduser.toInt())) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.GruposContainer, gruposList()).addToBackStack(null).commit()  
 Toast.makeText(  
 requireContext(),  
 "Ya no formas parte del grupo",  
 Toast.LENGTH\_SHORT  
 ).show()  
  
 deleteGrupo()  
 } else {  
 binding.bar.text = nombre  
 binding.NombreField.text = nombre  
 binding.ColorField.setBackgroundColor(color)  
 binding.DescripcionField.text = desc  
 binding.TypeField.text = when (tipo) {  
 0 -> "Fijo"  
 1 -> "Temporal"  
 else -> "Eliminado"  
 }  
 binding.AdminField.text = adminName  
 binding.CreatedField.text = createdString  
 }  
 } catch (e: Exception) {  
 Log.e("NetworkError", "Error during network call", e)  
 }  
 }  
 return binding.root  
}  
  
private fun colorPicker() {  
 binding.ColorField.setBackgroundColor(color)  
}

En resumen, la función se encarga de administrar la información que se obtiene del servidor acerca del grupo, mostrarla al usuario y corroborar que tenga acceso a la misma por el formar parte del grupo.

### 6.13 Compartir grupo redes sociales

El código forma un botón que permite compartir el grupo, de manera que mas usuarios puedan formar parte de él, se tienen 2 maneras de compartir el grupo, la primera es a través de Whatsapp y el segundo a través de Gmail. Para compartir la invitación se crea un enlace personalizado con la función ‘linkToShare’, después se genera un código QR a partir del enlace para tener un acceso más sencillo a terceros. En caso de que no se encuentre ningún medio posible para compartir la invitación se arroja un toast indicando el error.

Imagen 19, QR compartir grupo

Código QR

Descripción generada automáticamente

binding.Share.setOnClickListener {  
 val linkToShare = "http://savetrack.com.mx/joingroup.php?zxcd125s5d765e7wqa87sdftgh=$idori&mnhjkmnbg1yhb3vdrtgvc98swe=$admin"  
  
 val bitmap: Bitmap = generateQRCode(linkToShare)  
  
 val whatsappIntent = Intent(Intent.ACTION\_SEND)  
 whatsappIntent.type = "image/jpeg"  
 whatsappIntent.setPackage("com.whatsapp")  
 whatsappIntent.putExtra(Intent.EXTRA\_TEXT, "¡Únete a mi grupo de SaveTrack: $linkToShare!")  
 whatsappIntent.putExtra(Intent.EXTRA\_STREAM, getImageUri(requireContext(), bitmap))  
  
 val gmailIntent = Intent(Intent.ACTION\_SEND)  
 gmailIntent.type = "image/jpeg"  
 gmailIntent.setPackage("com.google.android.gm")  
 gmailIntent.putExtra(Intent.EXTRA\_SUBJECT, "¡Únete a mi grupo de SaveTrack!")  
 gmailIntent.putExtra(Intent.EXTRA\_TEXT, "¡Haz click en el enlace para unirte: $linkToShare!")  
 gmailIntent.putExtra(Intent.EXTRA\_STREAM, getImageUri(requireContext(), bitmap))  
  
 val chooserIntent = Intent.createChooser(gmailIntent, "Compartir a través de:")  
 chooserIntent.putExtra(Intent.EXTRA\_INITIAL\_INTENTS, arrayOf(whatsappIntent))  
  
 try {  
 startActivity(chooserIntent)  
 } catch (e: ActivityNotFoundException) {  
 Toast.makeText(  
 requireContext(),  
 "No se encontraron aplicaciones para compartir",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
}

En resumen, el Código permite compartir el grupo ya sea mediante un QR o un correo electrónico.

### 6.14 Salir del grupo

El botón descrito ejecuta un evento ‘SalirDeGrupo’ el cual pide una confirmación que al ser aceptada se ejecuta una corrutina asíncrona, primero se llama la función ‘SalirDeGrupo’ y después la función ‘DeleteGrupo’ para acto seguido redirigir al fragmento anterior. Terminada la corrutina se utiliza Volley para hacer una solicitud URL PUT para actualizar la lista de usuarios del grupo en la cual se elimina al usuario que salió y se construye un JSON con los nuevos usuarios el cual va dentro de la solicitud.

binding.SalirDeGrupo.setOnClickListener {  
 val builder = AlertDialog.Builder(context)  
 builder.setTitle("Confirmación")  
 builder.setMessage("¿Estás seguro de que deseas salir del grupo $ngrupo?")  
 builder.setPositiveButton("Sí") { dialog, \_ ->  
 lifecycleScope.launch {  
 salirDeGrupo(idori, admin, iduser)  
 deleteGrupo()  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.GruposContainer, back).addToBackStack(null).commit()  
 }  
 builder.setNegativeButton("No") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 builder.show()  
 }  
}  
  
private suspend fun salirDeGrupo(idori: Long, admin: Long, iduser: Long) {  
 val queue = Volley.newRequestQueue(requireContext())  
 var url = "http://savetrack.com.mx/gruposMiembrosPut.php?"  
  
 val miembrosJSON = withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/gruposMiembrosGet.php?localid=$idori&admin=$admin").readText()) }  
 val miembrosG = Array(miembrosJSON.length()) { miembrosJSON.getInt(it) }  
 val nuevosMiembros: MutableList<Int> = mutableListOf()  
 for (element in miembrosG) {  
 if (element != iduser.toInt()) {  
 nuevosMiembros.add(element)  
 }  
 }  
 val uniqueMiembros = nuevosMiembros.toSet().toList()  
 val jsonMiembros = JSONArray(uniqueMiembros)  
  
 Log.v("JSONMIEMBROS", "$jsonMiembros")  
 Log.v("MUTABLELISTMIEMBROS", "$nuevosMiembros")  
 val requestBody = "localid=$idori&admin=$admin&miembros=$jsonMiembros"  
 url += requestBody  
 val stringReq: StringRequest =  
 object : StringRequest(  
 Method.PUT, url,  
 Response.Listener { response ->  
 val strResp2 = response.toString()  
 Log.d("API", strResp2)  
  
 },  
 Response.ErrorListener { error ->  
 Log.d("API", "error => $error")  
 }  
 ) {  
 override fun getBody(): ByteArray {  
 return requestBody.toByteArray(Charset.defaultCharset())  
 }  
 }  
 Log.e("stringReq", stringReq.toString())  
 queue.add(stringReq)  
}

En resumen, el botón se encarga de permitirnos salir de grupos, así como mantener la información de los usuarios dentro del grupo en el servidor al corriente.

### 6.15 Agregar monto

El código es un gestionador de eventos el cual tras presionar el botón de agregar un monto nos redirige a la vista de ‘monto’ en la cual contamos con todos los parámetros posibles para agregar los montos, teniendo entre ellos el nombre, la etiqueta, monto, si es una deuda sus parámetros, fecha, entre otros. Se cuenta con un botón de confirmación el cual tras ser presionado pide una segunda confirmación, si es presionado ‘si’ se realiza una serie de verificaciones como no tener campos vacíos, que se tengan valores adecuados y demás, caso contrario se arroja un toast.

Imagen 20, añadir monto

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

binding.yocreoquesi.setOnClickListener {  
 if (binding.yocreoquesi.isChecked) {  
 displayInteresField()  
 displayFechaFinalField()  
 displayIC()  
 } else {  
 hideInteresField()  
 hideFechaFinalField()  
 hideIC()  
 }  
}  
  
  
binding.WeekField.selectionMode = SingleSelectionMode.create()  
binding.WeekField.locale = Locale.getDefault()  
binding.WeekField.setDaySelectionChangedListener { selectedDays ->  
 if (selectedDays.isNotEmpty()) {  
 val aux = selectedDays[0].name.lowercase()  
 selectedDay = when (aux) {  
 "monday" -> 41  
 "tuesday" -> 42  
 "wednesday" -> 43  
 "thursday" -> 44  
 "friday" -> 45  
 "saturday" -> 46  
 "sunday" -> 47  
  
 else -> {  
 40  
 }  
 }  
 Log.i("Día seleccionado", selectedDay.toString())  
 } else {  
 Log.i("Día seleccionado", "NONE")  
 }  
}  
  
binding.Confirm.setOnClickListener {  
 val concepto = binding.ConceptoField.text.toString()  
 val valorstr = binding.ValorField.text.toString()  
 var veces = 0L  
  
 var interes = 0.0  
  
 if (concepto != "" && valorstr != "" && valorstr != "." && selectedDay != 39) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 var valor = valorstr.toDouble()  
 valor = truncateDouble(valor)  
 var valorfinal = valor  
  
 fecha = when (frecuencia) {  
 0 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaField.year  
 val datedate = "$year$fMonth$fDay"  
 val fsi: Int = datedate.replace("-", "").toInt()  
  
 fsi  
 } // Único  
 1 -> {  
 veces = 0  
 100  
 } // Diario  
 7, 14 -> {  
 selectedDay  
 } // Semanales  
 30, 61, 91, 122, 183, 365 -> {  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
  
 intday  
 } // Mensuales  
 else -> {  
 val intyear = binding.FechaField.year - 1900  
 Log.w("YEAR", intyear.toString())  
 val intmonth = binding.FechaField.month  
 Log.w("MONTH", intmonth.toString())  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
 val datedate = "$intyear$intmonth$intday"  
 Log.w("DATE", datedate)  
  
 datedate.toInt()  
 } // Catch que agarra la fecha actual  
 }  
  
 val adddateStr: String = LocalDate.now().toString()  
 val adddate = adddateStr.replace("-", "").toInt()  
  
 if (binding.yocreoquesi.isChecked) {  
 estado = 5  
 interes = binding.InteresField.text.toString().toDouble()  
 val day = binding.FechaFinalField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaFinalField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaFinalField.year  
 val datedate = "$year$fMonth$fDay"  
 enddate = datedate.replace("-", "").toInt()  
  
 val addY = adddate.toString().substring(0, 4).toInt()  
 val addM = adddate.toString().substring(4, 6).toInt()  
 val addD = adddate.toString().substring(6, 8).toInt()  
 val endY = enddate.toString().substring(0, 4).toInt()  
 val endM = enddate.toString().substring(4, 6).toInt()  
 val endD = enddate.toString().substring(6, 8).toInt()  
 val addedAt = LocalDate.of(addY, addM, addD)  
 val toEnd = LocalDate.of(endY, endM, endD)  
 Log.v("FECHASASAS", "$addedAt, $toEnd, ${ChronoUnit.DAYS.between(addedAt, toEnd)}")  
 val duracion: Int = when (frecuencia) {  
 0 -> { 1 }  
 1 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt() }  
 7 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/7 }  
 14 -> { ChronoUnit.DAYS.between(addedAt, toEnd).toInt()/14 }  
 30 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt() }  
 61 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/2 }  
 91 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/3 }  
 122 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/4 }  
 183 -> { ChronoUnit.MONTHS.between(addedAt, toEnd).toInt()/6 }  
 365 -> { ChronoUnit.YEARS.between(addedAt, toEnd).toInt() }  
 else -> { 1 }  
 }  
  
 val tasa: Double = interes / duracion  
 val aux = valorfinal  
 tipointeres = if (binding.interesCompuesto.isChecked) {  
 valor = (aux / duracion) + aux \* (tasa / 100)  
 for (i in 0 until duracion) {  
 valorfinal += valorfinal \* (tasa / 100)  
 }  
  
 2  
 } else {  
 valorfinal += (valorfinal \* interes) / 100  
 valor = valorfinal / duracion  
  
 1  
 }  
 }  
  
 Log.v("Concepto", concepto)  
 Log.v("Valor", valor.toString())  
 Log.v("Fecha", fecha.toString())  
 Log.v("Frecuencia", frecuencia.toString())  
 Log.v("Etiqueta", label.toString())  
 Log.v("Interes", interes.toString())  
 Log.v("Veces", veces.toString())  
 Log.v("Addate", adddate.toString())  
 lifecycleScope.launch {  
 montoadd(  
 concepto,  
 valor,  
 valorfinal,  
 fecha,  
 frecuencia,  
 label,  
 interes,  
 tipointeres,  
 veces,  
 estado,  
 adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.GruposContainer, back).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(requireContext(), "No pueden haber campos vacíos", Toast.LENGTH\_SHORT).show()  
 }  
  
}

En resumen, el código nos permite insertar los montos para que los usuarios del grupo puedan ver el gasto.

### 6.16 Lista de miembros

La interfaz de usuario ‘FragmentGruposmiembroslistBinding’ se encarga de dar espacio para mostrar la lista de todos los usuarios los cuales forman parte del grupo, esta lista es alimentada por una solicitud HTTP la cual devuelve una lista con los nombres de los miembros del grupo los cuales van dentro de un objeto ‘usuario’, además se obtienen los ‘monto’ que se han insertado por los usuarios gracias a otra solicitud HTTP la cual ingresa los montos en un objeto ‘Monto’. Todo esto es realizado mediante threads.

Imagen 21, miembros grupos

Forma

Descripción generada automáticamente con confianza media

@SuppressLint("SetTextI18n")  
override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View {  
 binding = FragmentGruposmiembroslistBinding.inflate(inflater, container, false)  
 val decoder = Decoder(requireContext())  
 lifecycleScope.launch {  
 val isDarkMode = isDarkModeEnabled(requireContext())  
  
 if (isDarkMode) {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil2)  
 } else {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil)  
 }  
  
 Log.i("MODO", isDarkMode.toString())  
  
 montos = montosget()  
 try {  
 val miembrosJSON = withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/gruposMiembrosGet.php?localid=${group.idori}&admin=${group.admin}").readText()) }  
 val miembrosG = Array(miembrosJSON.length()) { miembrosJSON.getInt(it) }  
  
 if (!miembrosG.contains(iduser.toInt())) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.GruposContainer, gruposList()).addToBackStack(null).commit()  
 Toast.makeText(  
 requireContext(),  
 "Ya no formas parte del grupo",  
 Toast.LENGTH\_SHORT  
 ).show()  
  
 deleteGrupo()  
 } else {  
 val barText = "Miembros de ${group.nameg}"  
 binding.bar.text = barText  
  
 for (id in miembrosG) {  
 val nombre = withContext(Dispatchers.IO) { URL("http://savetrack.com.mx/usernameget.php?id=$id").readText().trim() }  
 Log.v("Memberrr", nombre)  
 val nuevoMiembro = Usuario(  
 iduser = id.toLong(),  
 nombre = nombre,  
 edad = 0,  
 chamba = 0,  
 foto = "",  
 diasaho = 0,  
 balance = 0.0,  
 meta = 0.0,  
 )  
 miembros.add(nuevoMiembro)  
 }  
  
 Log.v("idori", group.idori.toString())  
 Log.v("admin", group.admin.toString())  
 val jsonArr = withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/montosgrupoMultiget?localid=${group.idori}&admin=${group.admin}").readText()) }  
 Log.v("jsonArr", jsonArr.toString())  
 for (i in 0 until jsonArr.length()) {  
 val jsonObj = jsonArr.getJSONObject(i)  
 if (jsonObj.getLong("idmglocal") != null) {  
 val idmonto: Long = jsonObj.getLong("idmglocal")  
 val iduse: Long = jsonObj.getLong("idusuario")  
 val concepto: String = jsonObj.optString("concepto")  
 val valor: Double = jsonObj.optDouble("valor")  
 val valorfinal: Double = jsonObj.optDouble("valorfinal")  
 val fecha: Int = jsonObj.optInt("fecha")  
 val frecuencia: Int = jsonObj.optInt("frecuencia")  
 val etiqueta: Int = jsonObj.optInt("etiqueta")  
 val interes: Double = jsonObj.optDouble("interes")  
 val tipointeres: Int = jsonObj.optInt("tipointeres")  
 val veces: Long = jsonObj.optLong("veces")  
 val estado: Int = jsonObj.optInt("estado")  
 val adddate: Int = jsonObj.optInt("adddate")  
 val enddate: Int = jsonObj.optInt("enddate")  
 val cooldown: Int = jsonObj.optInt("cooldown")  
 val delay: Int = jsonObj.optInt("delay")  
 val sequence: String = jsonObj.optString("sequence")  
  
 val nuevoMonto = Monto(  
 idmonto = idmonto,  
 iduser = iduse,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = 8000 + group.Id.toInt(),  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = veces,  
 estado = estado,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = sequence  
 )  
 Log.v("Current monto $i", nuevoMonto.toString())  
  
 montos.add(nuevoMonto)  
 }  
 }  
 }  
 } catch (e: Exception) {  
 Log.e("NetworkError", "Error during network call", e)  
 }  
 binding.displayMiembros.adapter = MiembroAdapter(miembros)  
 }  
 return binding.root  
}

En resumen, la vista permite visualizar la lista de usuarios junto con los movimientos realizados por los mismos, todo eso mediante solicitudes HTTP.

### 6.17 Editar grupo

El fragmento ‘FragmentEditGrupo’ es el encargado de permitir la modificación de los valores del grupo, mediante solicitudes HTTP se obtienen los datos del grupo y sus miembros, los datos modificables son: nombre, color, descripción y tipo Una vez se efectúen los cambios y se manejen las posibles excepciones se manda un PUT para actualizar los datos del grupo en el servidor.

@SuppressLint("SetTextI18n")  
override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View {  
 binding = FragmentEditgrupoBinding.inflate(inflater, container, false)  
 colorPicker()  
 lifecycleScope.launch {  
 val isDarkMode = isDarkModeEnabled(requireContext())  
  
 if (isDarkMode) {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil2)  
 } else {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_perfil)  
 }  
  
 Log.i("MODO", isDarkMode.toString())  
  
 getGrupo()  
 try {  
 val jsonObjectG =  
 withContext(Dispatchers.IO) { JSONObject(URL("http://savetrack.com.mx/grupoGet.php?localid=$idori&admin=$admin").readText()) }  
 val miembrosJSON =  
 withContext(Dispatchers.IO) { JSONArray(URL("http://savetrack.com.mx/gruposMiembrosGet.php?localid=$idori&admin=$admin").readText()) }  
 val miembrosG = Array(miembrosJSON.length()) { miembrosJSON.getInt(it) }  
 //val idgglobal: Long = jsonObjectG.getLong("idgrupoglobal")  
 //val idglocal: String = jsonObjectG.getString("idgrupolocal")  
 //val idadmin: Long = jsonObjectG.optLong("idadmin")  
  
 val nombre: String = jsonObjectG.optString("nombre")  
 val tipo: Int = jsonObjectG.optInt("tipo")  
 val desc: String = jsonObjectG.optString("descripcion")  
 color = jsonObjectG.optInt("color")  
 val created: String = jsonObjectG.optString("created")  
 val createdString = "Creado el $created"  
 val adminName =  
 withContext(Dispatchers.IO) { URL("http://savetrack.com.mx/usernameget.php?id=$admin").readText() }  
  
 if (!miembrosG.contains(iduser.toInt())) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.GruposContainer, gruposList()).addToBackStack(null).commit()  
 Toast.makeText(  
 requireContext(),  
 "Ya no formas parte del grupo",  
 Toast.LENGTH\_SHORT  
 ).show()  
  
 deleteGrupo()  
 } else {  
 binding.bar.text = nombre  
 binding.NombreField.setText(nombre)  
 binding.ColorField.setBackgroundColor(color)  
 binding.DescripcionField.setText(desc)  
 binding.TypeField.text = when (tipo) {  
 0 -> "Fijo"  
 1 -> "Temporal"  
 else -> "Eliminado"  
 }  
 binding.AdminField.text = adminName  
 binding.CreatedField.text = createdString  
 }  
 } catch (e: Exception) {  
 Log.e("NetworkError", "Error during network call", e)  
 }  
 }  
 return binding.root  
}

En resumen, el fragmento permite modificar ciertos valores del grupo.

### 6.18 Eliminar grupo

La función suspendida ‘deleteGrupo’ utiliza corrutinas para extraer el identificador del grupo a eliminar, después se actualiza su estado a tipo ‘2’ o ‘muerto’ para posteriormente registrar las etiquetas del grupo en los logs y se transiciona de regreso a la lista de grupos.

private suspend fun deleteGrupo() {  
 withContext(Dispatchers.IO) {  
 val gruposDao = Stlite.getInstance(requireContext()).getGruposDao()  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
  
 val idg = arguments?.getLong(idv)  
 if (idg != null) {  
 val muertoGrupo = Grupos(  
 Id = idg,  
 nameg = gruposDao.getNameG(idg.toInt()),  
 description = gruposDao.getDescription(idg.toInt()),  
 type = 2,  
 admin = gruposDao.getAdmin(idg.toInt()),  
 idori = gruposDao.getIdori(idg.toInt()),  
 color = gruposDao.getColor(idg.toInt()),  
 enlace = gruposDao.getEnlace(idg.toInt())  
 )  
 gruposDao.updateGrupo(muertoGrupo)  
  
 val plabel = labelsDao.getPlabel(idg.toInt())  
 val muertaLabel = Labels(  
 idlabel = enlace,  
 plabel = plabel,  
 color = color,  
 estado = 1  
 )  
  
 labelsDao.updateLabel(muertaLabel)  
 val labelss = labelsDao.getAllLabels()  
 Log.i("ALL LABELS", labelss.toString())  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.GruposContainer, gruposList()).addToBackStack(null)  
 .commit()  
 }  
 }  
}

En resumen, la función elimina el grupo seleccionado y actualiza su estado en el servidor además de guardar las etiquetas en los logs para seguir teniendo dicha información.

## 7 Historial

### 7.1 Fragment historial

El fragmento del historial se compone de básicamente los mismos elementos que los fragmentos, en el caso del historial se cuenta con un ‘Spinner’ para las etiquetas las cuales se muestran en un array.

Este fragmento se encarga de dar interfaz para el calendario el cual configura un selector de meses y muestra la cuadrícula correspondiente.

Imagen 22, historial

Calendario

Descripción generada automáticamente

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View {  
 binding = FragmentHistorialmainBinding.inflate(inflater, container, false)  
 lifecycleScope.launch {  
 isDarkMode = isDarkModeEnabled(requireContext())  
 getLabels()  
  
 if (isDarkMode) {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_historial2)  
 } else {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_historial)  
 }  
  
 Log.i("MODO", isDarkMode.toString())  
  
 val hoy = LocalDate.now()  
 val day = hoy.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = hoy.monthValue - 1  
 val fMonth = String.format("%02d", month)  
 val year = hoy.year  
 val fecha = "$year$fMonth$fDay"  
 val today: Int = fecha.replace("-", "").toInt()  
 Log.v("today", "$today")  
  
 val startYear = 1998  
  
 val monthsPagerAdapter = HistorialPagerAdapter(requireContext(), isDarkMode, day)  
 val viewPager: ViewPager = binding.calendarView  
 viewPager.adapter = monthsPagerAdapter  
 viewPager.offscreenPageLimit = 1  
  
 viewPager.currentItem = month + (year - startYear) \* 12  
  
 val arrayEtiquetas = mutableEtiquetas  
 val adapterG = ArrayAdapter(  
 requireContext(),  
 android.R.layout.simple\_spinner\_item,  
 arrayEtiquetas  
 )  
 adapterG.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
  
 binding.SpinnerFilter.adapter = adapterG  
 binding.SpinnerFilter.onItemSelectedListener = this@historialmain  
 }  
 return binding.root  
}

En resumen, el fragmento es la vista principal del historial por lo que en este se visualiza el calendario y las etiquetas.

### 7.2 Etiquetas

La función suspendida ‘getLabels’ se utiliza para obtener las etiquetas de la bd Stlite y se llama al método ‘getMaxLabel’ para obtener el valor máximo de las etiquetas, después se obtiene el color y se hace una iteración para mover las etiquetas hasta llegar al punto en que la lista no se quede vacía, después los almacena dentro de 3 listas multitablas.

Imagen 23, etiquetas

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

private suspend fun getLabels() {  
 withContext(Dispatchers.IO) {  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
  
 val max = labelsDao.getMaxLabel()  
 val themeAttrs = intArrayOf(com.google.android.material.R.attr.colorPrimaryVariant)  
  
 val typedArray = requireContext().obtainStyledAttributes(themeAttrs)  
 val colorPrimaryVariant = typedArray.getColor(0, 0)  
  
 mutableIds.add(0)  
 mutableEtiquetas.add("Seleccionar")  
 mutableColores.add(colorPrimaryVariant)  
 for (i in 1..max) {  
 if (labelsDao.getPlabel(i) != ""){  
 mutableIds.add(labelsDao.getIdLabel(i))  
 mutableEtiquetas.add(labelsDao.getPlabel(i))  
 mutableColores.add(labelsDao.getColor(i))  
 }  
 }  
 Log.v("idl", "$mutableIds")  
 Log.v("plabel", "$mutableEtiquetas")  
 Log.v("color", "$mutableColores")  
 }  
}

En resumen, la función descarga los datos de las etiquetas y los almacena en multitablas.

### 7.3 Historial

El fragmento tiene una interfaz de calendario la cual es referida como ‘Historial’, esta interfaz puede ser interactuada por el usuario mediante varios métodos, uno es el botón ‘verMontos’ el cual al seleccionar una fecha en el calendario con el ‘DatePicker’ se realiza una búsqueda de montos de dicho día. El botón ‘options’ abre un menú lateral para navegar distintas secciones del historial como las etiquetas, papelera y demás.

Imagen 24, historialetiquetas

Calendario

Descripción generada automáticamente

@SuppressLint("ClickableViewAccessibility")  
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 var fechaSeleccionada: String  
  
 binding.verMontos.setOnClickListener {  
 lifecycleScope.launch {  
 val day = binding.spinnerPick.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.spinnerPick.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.spinnerPick.year  
 fechaSeleccionada = "$year$fMonth$fDay"  
 val fsi: Int = fechaSeleccionada.replace("-", "").toInt()  
 Toast.makeText(requireContext(), "$fsi", Toast.LENGTH\_SHORT).show()  
 val montosF: historialMontosList = if (selectedLabel != 0) {  
 historialMontosList.fechaSearch(fsi, selectedLabel)  
 } else {  
 historialMontosList.fechaSearch(fsi)  
 }  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, montosF).addToBackStack(null).commit()  
 }  
 }  
  
 val hoy = LocalDate.now()  
 val day = hoy.dayOfMonth  
 val month = hoy.monthValue - 1  
 val year = hoy.year  
 binding.spinnerPick.init(year, month, day)  
 { \_, nYear, nMonth, \_ ->  
 //val monthsPagerAdapter = HistorialPagerAdapter(requireContext(), isDarkMode, nDay)  
 //binding.calendarView.adapter = monthsPagerAdapter  
 binding.calendarView.currentItem = nMonth + (nYear - 1998) \* 12  
 }  
  
 binding.calendarView.addOnPageChangeListener(object : ViewPager.OnPageChangeListener {  
 override fun onPageScrolled(position: Int, positionOffset: Float, positionOffsetPixels: Int) {}  
 override fun onPageScrollStateChanged(state: Int) {}  
 override fun onPageSelected(position: Int) {}  
 })  
  
 binding.Options.setOnClickListener {  
 binding.drawerLayout.openDrawer(binding.barrita)  
 }  
  
 binding.barrita.setNavigationItemSelectedListener { menuItem ->  
 when (menuItem.itemId) {  
 R.id.etiquetas -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.historial\_container, historialEtiquetas()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.favoritos -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.historial\_container, historialFavoritos()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.papelera -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.historial\_container, historialPapelera()).addToBackStack(null).commit()  
  
 true  
 }  
  
 else -> false  
 }  
 }  
}

En resumen, la activity implementa las funciones básicas de un calendario permitiendo visualizar fechas además de poder buscar dentro de x fechas los montos realizados.

### 7.4 Modificar etiquetas

Mediante el método ‘LabelsViewHolder’ registra la información referente a las etiquetas por lo que establece las propiedades de las etiquetas como lo son el color y el nombre, además permite editar y eliminar los valores de la etiqueta, ambas con un cuadro de confirmación, al eliminar se registra en los logs mediante ‘labeldelete’.

Imágen 25, editaretiqueta

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

override fun onBindViewHolder(holder: LabelsViewHolder, position: Int) {  
 val labels = labelss[position]  
 Log.i("INFO", labels.idlabel.toString() + ", " + labels.plabel + ", " + labels.color)  
 holder.nombreTextView.text = labels.plabel  
 holder.colorImageView.setBackgroundColor(labels.color)  
 val upup = historialUpdate.sendLabel(labels.idlabel, labels.plabel, labels.color)  
  
 holder.optionsM.setOnClickListener {  
 val popupMenu = PopupMenu(requireContext(), holder.optionsM, Gravity.END)  
 val inflater = popupMenu.menuInflater  
 inflater.inflate(R.menu.options\_item\_label, popupMenu.menu)  
  
 popupMenu.setOnMenuItemClickListener { item: MenuItem ->  
 when (item.itemId) {  
 R.id.action\_editLabel -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, upup).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_deleteLabel -> {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres eliminar la etiqueta ${labels.plabel}? Esta acción no se puede deshacer")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
  
 Log.v("Id de la etiqueta actualizada", labels.idlabel.toString())  
 Log.v("Plabel", labels.plabel)  
 Log.v("Color", labels.color.toString())  
 lifecycleScope.launch {  
 labeldelete(labels.idlabel, labels.plabel, labels.color)  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, historialmain()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
  
 true  
 }  
 else -> false  
 }  
 }  
  
 popupMenu.show()  
 }  
 if (position == labelsp.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.y1bottomcell)  
 }  
}

En resumen, el código maneja las etiquetas junto con su edición y eliminación.

7.5 Agregar etiquetas

El código se compone de 2 eventos, el primero ejecuta una función ‘colorPicker’ el cual abre un selector de color para la etiqueta, el segundo ocurre al presionar el botón de confirmar el cual registra la nueva etiqueta en la bd room mediante la función ‘labeladd’, en caso de que un campo esté vacío se remarcará con un toast.

binding.ColorField.setOnClickListener {  
 colorPicker(false)  
}  
  
binding.Confirm.setOnClickListener {  
 val plabel = binding.PlabelField.text.toString()  
  
 if (plabel != "" && color != 0xffffff) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 Log.v("Plabel", plabel)  
 Log.v("Color", color.toString())  
  
 lifecycleScope.launch {  
 labeladd(plabel, color)  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, back).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "No pueden haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
  
}

En resumen, la función permite agregar etiquetas personalizadas con color personalizado.

### 7.6 Historial montos

La función suspendida ‘montoget’ utiliza corrutinas para obtener los montos por medio de DAO’s utilizando threads por el cual se ejecuta una serie de operaciones para obtener la fecha, el día del mes, la semana, y el nombre del mes para obtener el ‘monto’. El sistema devuelve una lista de ‘montosf’ la cual es filtrada mediante la fecha buscada por el usuario.

@SuppressLint("SetTextI18n")  
private suspend fun montosget(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
  
 val formatoFecha = SimpleDateFormat("yyyyMMdd", Locale.getDefault())  
 val datedate: Int? = arguments?.getInt(fecha)  
  
 val fs = "$datedate"  
  
 val truefecha = formatoFecha.parse(fs)  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 val dom = calendar.get(Calendar.DAY\_OF\_MONTH)  
 val w = calendar.get(Calendar.DAY\_OF\_WEEK)  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 val yyy = calendar.get(Calendar.YEAR)  
 val mesesito = when (calendar.get(Calendar.MONTH)) {  
 0 -> "Enero"  
 1 -> "Febrero"  
 2 -> "Marzo"  
 3 -> "Abril"  
 4 -> "Mayo"  
 5 -> "Junio"  
 6 -> "Julio"  
 7 -> "Agosto"  
 8 -> "Septiembre"  
 9 -> "Octubre"  
 10 -> "Noviembre"  
 11 -> "Diciembre"  
 else -> "cualquier mes"  
 }  
 val semanita = when (dow) {  
 47 -> "Domingo"  
 41 -> "Lunes"  
 42 -> "Martes"  
 43 -> "Miércoles"  
 44 -> "Jueves"  
 45 -> "Viernes"  
 46 -> "Sábado"  
 else -> ""  
 }  
  
 Log.i("datedate", fs)  
 Log.i("DOM", dom.toString())  
 Log.i("DOW", dow.toString())  
  
 binding.bar.text = "Montos del $semanita $dom de $mesesito del $yyy"  
  
 Log.v("labelll", "$label")  
 montosf = if (datedate != null) {  
 if (label != 0) {  
 Log.v("con etiquetona", "")  
 montoDao.getMontoXFecha(datedate, dom, dow, 100, datedate, label)  
 } else {  
 Log.v("sin etiquetona", "")  
 montoDao.getMontoXFecha(datedate, dom, dow, 100, datedate)  
 }  
 } else {  
 montoDao.getMontoXFecha()  
 }  
 Log.i("ALL MONTOS", montosf.toString())  
 }  
 return montosf  
}

En resumen, la función se encarga de buscar y filtrar los montos que fueron efectuados en el día que el usuario ingrese.

### 7.7 Papelera

El código implementa un Recycler View el cual muestra los montos que han sido eliminados, de manera que no se muestran en el calendario, pero de igual manera están guardados, mediante corrutinas se pueden regresar a la tabla ‘monto’ en caso que el usuario desee recuperar el monto, caso contrario puede eliminarlo definitivamente.

Imagen 26, papelera

Patrón de fondo

Descripción generada automáticamente

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 val decoder = Decoder(requireContext())  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.vecesTextView.text = monto.veces.toString()  
 lifecycleScope.launch {  
 holder.etiquetaTextView.text = decoder.label(monto.etiqueta)  
 }  
  
 holder.optionsM.setOnClickListener {  
 val popupMenu = PopupMenu(requireContext(), holder.optionsM, Gravity.END)  
 val inflater = popupMenu.menuInflater  
 inflater.inflate(R.menu.options\_item\_papelera, popupMenu.menu)  
  
 popupMenu.setOnMenuItemClickListener { item: MenuItem ->  
 when (item.itemId) {  
 R.id.action\_verMonto -> {  
 val ver = verMonto.sendMonto(monto.idmonto)  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, ver).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_getBackMonto -> {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres recuperar el monto ${monto.concepto}?")  
 .setPositiveButton("Recuperar") { dialog, \_ ->  
  
 Log.v("Id del monto actualizado", monto.idmonto.toString())  
 Log.v("Concepto", monto.concepto)  
 Log.v("Valor", monto.valor.toString())  
 Log.v("Fecha", monto.fecha.toString())  
 Log.v("Frecuencia", monto.frecuencia.toString())  
 Log.v("Etiqueta", monto.etiqueta.toString())  
 Log.v("Interes", monto.interes.toString())  
 Log.v("Veces", monto.veces.toString())  
 lifecycleScope.launch {  
 montoback(monto.idmonto, monto.concepto, monto.valor, monto.fecha, monto.frecuencia, monto.etiqueta, monto.interes, monto.veces, monto.estado, monto.adddate)  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, indexmain()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
  
 true  
 }  
 R.id.action\_killMonto -> {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres eliminar el monto ${monto.concepto}? Esta acción no se puede deshacer")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
  
 Log.v("Id del monto actualizado", monto.idmonto.toString())  
 Log.v("Concepto", monto.concepto)  
 Log.v("Valor", monto.valor.toString())  
 Log.v("Fecha", monto.fecha.toString())  
 Log.v("Frecuencia", monto.frecuencia.toString())  
 Log.v("Etiqueta", monto.etiqueta.toString())  
 Log.v("Interes", monto.interes.toString())  
 Log.v("Veces", monto.veces.toString())  
 lifecycleScope.launch {  
 montodelete(monto.idmonto, monto.concepto, monto.valor, monto.fecha, monto.frecuencia, monto.etiqueta, monto.interes, monto.veces, monto.adddate)  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.index\_container, indexmain()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
  
 true  
 }  
 else -> false  
 }  
 }  
  
 popupMenu.show()  
 }  
 if (position == montosp.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.y1bottomcell)  
 }

En resumen, la papelera permite tener un mayor control en caso de que el usuario se equivoque al eliminar un monto.

### 7.8 Favoritos

El fragmento utiliza el método ‘onBindViewHolder’ para almacenar un objeto ‘monto’ y se obtienen los montos que estén en favoritos los cuales se verán resaltados a la hora de filtrar las etiquetas.

Imagen 27, favoritos

Patrón de fondo

Descripción generada automáticamente

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 var tempstat = 5  
 val decoder = Decoder(requireContext())  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.fechaTextView.text = decoder.date(monto.adddate)  
 lifecycleScope.launch {  
 holder.etiquetaTextView.text = decoder.label(monto.etiqueta)  
 }  
  
 if (monto.estado == 0 || monto.estado == 1 || monto.estado == 5 || monto.estado == 6){  
 tempstat = 5  
 }  
 if (monto.estado == 3 || monto.estado == 4 || monto.estado == 8 || monto.estado == 9){  
 tempstat = 8  
 }  
  
 holder.optionsM.setOnClickListener {  
 val popupMenu = PopupMenu(requireContext(), holder.optionsM, Gravity.END)  
 val inflater = popupMenu.menuInflater  
 inflater.inflate(R.menu.options\_item\_fav, popupMenu.menu)  
  
 popupMenu.setOnMenuItemClickListener { item: MenuItem ->  
 when (item.itemId) {  
 R.id.action\_favMonto -> {  
 if (tempstat == 5){  
 tempstat = 8  
 }  
 if (tempstat == 8){  
 tempstat = 5  
 }  
 lifecycleScope.launch {  
 montoFavorito(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate,  
 monto.enddate,  
 monto.cooldown  
 )  
 }  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.historial\_container, historialFavoritos()).addToBackStack(null)  
 .commit()  
  
 true  
 }  
 R.id.action\_verMonto -> {  
 val ver = verMonto.sendMonto(monto.idmonto)  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, ver).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_editMonto -> {  
 val upup = indexmontoupdate.sendMonto(monto.idmonto, monto.concepto, monto.valor, monto.fecha, monto.frecuencia, monto.etiqueta, monto.interes, monto.veces, monto.adddate)  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.historial\_container, upup).addToBackStack(null).commit()  
  
 true  
 }  
 else -> false  
 }  
 }  
 popupMenu.show()  
 }  
 if (position == montosp.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.y1bottomcell)  
 }  
 }  
  
  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }  
 }  
}

En resumen, el código se encarga de filtrar las etiquetas que estén marcados como favoritos.

## 8 Planes de Ahorro

### 8.1 Diseño planes de ahorro

La función ‘mostrarDatos’ utiliza DAO’s para controlar los threads y obtener información de la bd relacionados a los gastos e ingresos de los usuarios mediante la clase ‘Stlite’, después se consultan los días ahorrando ‘diasaho’, se calcula el balance total, se formatea a 2 decimales y se inserta en los campos correspondientes del XML. Después se construye una URL GET para obtener el valor del dólar del día actual y de igual manera se inserta.

private suspend fun mostrarDatos() {  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresosGastosDao = Stlite.getInstance(requireContext()).getIngresosGastosDao()  
  
 val diasaho = usuarioDao.checkDiasaho()  
 val totalIngresos = ingresosGastosDao.checkSummaryI()  
 val totalGastos = ingresosGastosDao.checkSummaryG()  
 val totalisimo = totalIngresos - totalGastos  
 val decimalFormat = DecimalFormat("#.##")  
 val balance = "${decimalFormat.format(totalisimo)}$"  
 usuarioDao.updateBalance(usuarioDao.checkId(), totalisimo)  
  
 val durl = "http://savetrack.com.mx/dlrval.php"  
 val queue: RequestQueue = Volley.newRequestQueue(requireContext())  
 val checkDollar = StringRequest(  
 Request.Method.GET, durl,  
 { response ->  
 dollarValue = response.toString()  
 Log.d("DÓLAR HOY", dollarValue)  
 binding.PACurrencyButton.text = buildString {  
 append("Dolar HOY: ")  
 append("$$dollarValue")  
 }  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(), "No se ha podido conectar al valor del dólar hoy", Toast.LENGTH\_SHORT  
 ).show()  
 binding.PACurrencyButton.text = buildString {  
 append("Sin conexión")  
 }  
 Log.d("error => $error", "SIE API ERROR")  
 }  
 )  
 queue.add(checkDollar)  
  
 Log.v("Diasaho", diasaho.toString())  
 Log.v("Balance", balance)  
  
 binding.PADiasAhorrandoButton.text = buildString {  
 append("¡")  
 append(diasaho.toString())  
 append(" días ahorrando!")  
 }  
 binding.PASaldoActual.text = buildString {  
 append("Balance: ")  
 append(balance)  
 }  
 binding.PASaldoActual.outlineAmbientShadowColor = resources.getColor(R.color.X0)  
 binding.PASaldoActual.outlineSpotShadowColor = resources.getColor(R.color.X0)  
 }  
}

En resumen, la clase se encarga de obtener todos los datos a mostrar en la sección de planes de ahorro y mostrarlos.

### 8.2 Inicialización de montos

El código utiliza el contexto ‘suspend’ para realizar múltiples consultas a la bd mediante room, entre las funciones están ‘montosget’, ‘montosgetAlfabetica’, ‘montosgetValuados’, ‘montosgetVeces’ y demás. Todos estos se encargan de tener los datos sobre monto y las deudas.

Imagen 28, planes de ahorro

Imagen que contiene Icono

Descripción generada automáticamente

@SuppressLint("SetTextI18n")  
private suspend fun montosget(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 binding.bar.text = "Deudas"  
 deudas = montoDao.getDeudasList()  
 Log.i("ALL MONTOS", deudas.toString())  
 }  
 return deudas  
}  
  
private suspend fun montosgetAlfabetica(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 deudas = montoDao.getDeudasAlfabetica()  
 Log.i("ALL MONTOS", deudas.toString())  
 }  
 return deudas  
}  
  
private suspend fun montosgetValuados(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 deudas = montoDao.getDeudasValuadas()  
 Log.i("ALL MONTOS", deudas.toString())  
 }  
 return deudas  
}  
  
private suspend fun montosgetVeces(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 deudas = montoDao.getDeudasVeces()  
 Log.i("ALL MONTOS", deudas.toString())  
 }  
 return deudas  
}  
  
private suspend fun montosgetEtiquetados(): List<Monto> {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 deudas = montoDao.getDeudasEtiquetadas()  
 Log.i("ALL MONTOS", deudas.toString())  
 }  
 return deudas  
}

En resumen, el código inicializa las variables que se necesitan para procesar las deudas del usuario.

8.3 Ver deudas

El fragmento que se redirecciona al presionar la deuda en planes de ahorro es un fragmento compuesto de un grid el cual se llena con las deudas registradas por el usuario, para las deudas se extraen de la bd mediante room datos como concepto, interés, valor, fecha, entre otros. Entre las opciones de interacción el usuario cuenta con botones para liquidar, eliminar o editar la deuda. Para mostrar todos estos datos se usa una Recycler View y la función ‘getItemCount’ la cual devuelve la cantidad de elementos de la lista.

Imagen 29, deudas

Interfaz de usuario gráfica

Descripción generada automáticamente

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 var tempstat = 5  
 val decoder = Decoder(requireContext())  
  
 holder.itemView.setOnClickListener {  
 val ver = verMonto.sendMonto(monto.idmonto)  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, ver).addToBackStack(null).commit()  
 }  
 holder.conceptoTextView.text = monto.concepto  
 holder.interesTextView.text = monto.interes.toString()  
 holder.valorTextView.text = truncateDouble(monto.valor).toString()  
 holder.valorFinalTextView.text = monto.valorfinal?.let { truncateDouble(it).toString() }  
 holder.fechaTextView.text = monto.fecha?.let { decoder.date(it) }  
 holder.fechaFinalTextView.text = monto.enddate?.let { decoder.date(it) }  
 lifecycleScope.launch {  
 holder.etiquetaTextView.text = decoder.label(monto.etiqueta)  
 }  
  
 if (monto.estado == 0 || monto.estado == 1 || monto.estado == 5 || monto.estado == 6) {  
 tempstat = 5  
 }  
 if (monto.estado == 3 || monto.estado == 4 || monto.estado == 8 || monto.estado == 9) {  
 tempstat = 8  
 }  
  
 holder.optionsM.setOnClickListener {  
 val popupMenu = PopupMenu(requireContext(), holder.optionsM, Gravity.END)  
 val inflater = popupMenu.menuInflater  
 inflater.inflate(R.menu.options\_item\_deuda, popupMenu.menu)  
  
 popupMenu.setOnMenuItemClickListener { item: MenuItem ->  
 when (item.itemId) {  
 R.id.action\_favMonto -> {  
 if (tempstat == 5) {  
 tempstat = 8  
 }  
 if (tempstat == 8) {  
 tempstat = 5  
 }  
 lifecycleScope.launch {  
 montoFavorito(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.estado,  
 monto.adddate  
 )  
 }  
  
 true  
 }  
 R.id.action\_verMonto -> {  
 val ver = verMonto.sendMonto(monto.idmonto)  
  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, ver).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.action\_liquidateMonto -> {  
 lifecycleScope.launch {  
 liq(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.pda\_container, pdaDeudasList()).addToBackStack(null)  
 .commit()  
 }  
  
 true  
 }  
 R.id.action\_editMonto -> {  
 val upup = monto.valorfinal?.let { it1 ->  
 indexmontoupdate.sendMonto(  
 monto.idmonto,  
 monto.concepto,  
 it1,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
 }  
  
 if (upup != null) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, upup).addToBackStack(null).commit()  
 }  
  
 true  
 }  
 R.id.action\_deleteMonto -> {  
 if (monto.estado == 3 || monto.estado == 4 || monto.estado == 8 || monto.estado == 9 || tempstat == 8) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("El monto ${monto.concepto} no se puede eliminar porque está marcado como favorito")  
 .setPositiveButton("Aceptar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres enviar el monto ${monto.concepto} a la papelera?")  
 .setPositiveButton("Eliminar") { dialog, \_ ->  
  
 Log.v("Id del monto actualizado", monto.idmonto.toString())  
 Log.v("Concepto", monto.concepto)  
 Log.v("Valor", monto.valor.toString())  
 Log.v("Fecha", monto.fecha.toString())  
 Log.v("Frecuencia", monto.frecuencia.toString())  
 Log.v("Etiqueta", monto.etiqueta.toString())  
 Log.v("Interes", monto.interes.toString())  
 Log.v("Veces", monto.veces.toString())  
 lifecycleScope.launch {  
 montoPapelera(  
 monto.idmonto,  
 monto.concepto,  
 monto.valor,  
 monto.fecha,  
 monto.frecuencia,  
 monto.etiqueta,  
 monto.interes,  
 monto.veces,  
 monto.adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.pda\_container, pdaDeudasList()).addToBackStack(null)  
 .commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 }  
  
 true  
 }  
 else -> false  
 }  
 }  
 popupMenu.show()  
 }  
 }  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }  
 }  
}

En resumen, el fragmento permite que el usuario pueda ver sus deudas de manera detallada gracias a la bd que almacena toda la información necesaria y además puede modificar las deudas.

## 9 Finanzas

### 9.1 Diseño finanzas

El método ‘onViewCreated’ es el encargado de inflar el fragmento de finanzas, entre los elementos de finanzas se encuentran la sección de consejos y sugerencias, producto recomendado, gráficas y fechas.

Imagen 30, finanzas

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 binding.Options.setOnClickListener {  
 binding.drawerLayout.openDrawer(binding.barrita)  
 }  
  
 binding.barrita.setNavigationItemSelectedListener { menuItem ->  
 when (menuItem.itemId) {  
 R.id.conysug-> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasConySug()).addToBackStack(null).commit()  
  
 true  
 }  
 R.id.economia -> {  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasstatsahorro()).addToBackStack(null).commit()  
 } else {  
 Toast.makeText(requireContext(), "No hay acceso a internet", Toast.LENGTH\_SHORT).show()  
 }  
  
 true  
 }  
 R.id.eventos -> {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasEventos()).addToBackStack(null).commit()  
  
 true  
 }  
  
 else -> false  
 }  
 }  
  
 binding.FinanzasConsejosySugerencias.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasConySug()).addToBackStack(null).commit()  
 }  
  
 binding.Economia.setOnClickListener {  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasstatsahorro()).addToBackStack(null).commit()  
 } else {  
 Toast.makeText(requireContext(), "No hay acceso a internet", Toast.LENGTH\_SHORT).show()  
 }  
 }  
  
 binding.FinanzasCalendar.setOnClickListener {  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromleft, R.anim.toright)  
 .replace(R.id.finanzas\_container, finanzasEventos()).addToBackStack(null).commit()  
 }  
}

En resumen, el fragmento de finanzas contiene elementos visuales para el funcionamiento de la aplicación junto con los redirects necesarios para navegar dentro de la app.

### 9.2 Producto recomendado

La función suspendida ‘mostrarProuctos’ se encarga de obtener y mostrar información de los productos utilizando la API de Mercado Libre, primero se accede a la bd room para obtener los conceptos de la querie y después se hace una quiere a la API para obtener los productos asociados a los conceptos utilizados, se procesa el JSON y se obtiene ID, Título, Precio, Enlace Permanente y la miniatura del producto. Todos los productos que cumplan con los requisitos de mostrarProductos son introducidos a una lista de ‘Item’ los cuales se les calcula la afinidad mediante la similitud del nombre entre ambos productos, se calcula la afinidad y se muestra en el fragment. Se cuenta con 3 botones para modificar la lista ‘Item’, los cuales pueden mostrar más productos, recomendar otro producto y dejar de recomendar ese producto. Si se presiona el producto se redirige a la página de Mercado Libre.

Imagen 31, producto recomendado

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

@SuppressLint("SetTextI18n")  
private suspend fun mostrarProductos() {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val conceptos = montoDao.getTopConceptos()  
 val precios = montoDao.getTopValor()  
 Log.v("CONCEPTOS", conceptos.toString())  
 if (conceptos.size >= 3) {  
 for (i in 0 until 3) {  
  
 val durl = "https://api.mercadolibre.com/sites/MLM/search?q=${conceptos[i]}"  
 val queue: RequestQueue = Volley.newRequestQueue(requireContext())  
 val elproducto = JsonObjectRequest(  
 Request.Method.GET, durl, null,  
 { response ->  
 try {  
 val results = response.getJSONArray("results")  
 if (results.length() > 0) {  
 val firstResult = results.getJSONObject(0)  
 val id: String = firstResult.getString("id")  
 val title: String = firstResult.getString("title")  
 val price: Double = firstResult.getDouble("price")  
 val permalink: String = firstResult.getString("permalink")  
 val thumbnail: String = firstResult.getString("thumbnail")  
  
 Log.v(  
 "PRODUCT DATA",  
 "$id, $title, $price, $permalink, $thumbnail"  
 )  
 Item.ItemsRepository.add(  
 Item(  
 id,  
 title,  
 price,  
 permalink,  
 thumbnail  
 )  
 )  
 } else {  
 binding.FinanzasItemImage.setBackgroundResource(R.drawable.ic\_notfound)  
 }  
 } catch (e: JSONException) {  
 binding.FinanzasItemImage.setBackgroundResource(R.drawable.ic\_notnet)  
 }  
 },  
 { error ->  
 binding.FinanzasItemImage.setBackgroundResource(R.drawable.ic\_notnet)  
 Log.e("error => $error", "SIE API ERROR")  
 }  
 )  
 queue.add(elproducto)  
 }  
 } else {  
 lifecycleScope.launch {  
 binding.FinanzasItemImage.setBackgroundResource(R.drawable.ic\_notfound)  
 }  
 }  
  
 val query = Item.ItemsRepository.getAll()  
 val ids: List<String> = query.map { it.id }  
 val titles: List<String> = query.map { it.title }  
 val prices: List<Double> = query.map { it.price }  
 val links: List<String> = query.map { it.permalink }  
 val images: List<String> = query.map { it.thumb }  
  
 var i = 1  
 if (i <= images.size) {  
 val afinity = afinityCalculator(conceptos[i], titles[i])  
 val saving = saveCalculator(precios[i], prices[i])  
 val fAfinity = decimalFormat.format(afinity)  
 val fSaving = decimalFormat.format(saving)  
 val pic = images[i]  
 Log.v("PIC", adaptLink(pic))  
 lifecycleScope.launch {  
 bajarfoto(adaptLink(pic))  
 binding.prodhint.text = titles[i]  
 binding.afinity.text = "Afinidad: $fAfinity%"  
 binding.saving.text = "Ahorro: $fSaving%"  
 }  
 Log.v(  
 "PRODUCT DATA",  
 "${ids[i]}, ${titles[i]}, ${prices[i]}, ${links[i]}, ${images[i]}"  
 )  
  
 binding.VerMasMERCALIBRE.setOnClickListener {  
 val url = links[i]  
 val intent = Intent(Intent.ACTION\_VIEW)  
 intent.data = Uri.parse(url)  
 startActivity(intent)  
 }  
  
 binding.VerMenosMERCALIBRE.setOnClickListener {  
 lifecycleScope.launch {  
 Item.ItemsRepository.remove(ids[i])  
 delay(500)  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.finanzas\_container, finanzasmain()).addToBackStack(null)  
 .commit()  
 }  
 }  
  
 binding.NoVerMERCALIBRE.setOnClickListener {  
 lifecycleScope.launch {  
 Item.ItemsRepository.clean()  
 delay(500)  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.finanzas\_container, finanzasmain()).addToBackStack(null)  
 .commit()  
 }  
 }  
 }  
 }  
}

En resumen, la función ‘mostrarProductos’ es la encargada de manejar la recomendación de productos por medio de la API de MercadoLibre

### 9.3 Calculadora de afinidad

El código se encarga de tomar 2 cadenas: ‘search’ y ‘result’ las cuales son ingresadas a una matriz y comparadas mediante diferencia de Levenshtein con la función ‘afinityCalculator’ la cual compara ambas cadenas y devuelve el porcentaje de similitud entre ambas dando así el producto con mayor afinidad al producto que se tomó como parámetro para mandar la quiere.

private fun afinityCalculator(search: String, result: String): Double {  
 val len1 = search.length  
 val len2 = result.length  
 val matrix = Array(len1 + 1) { IntArray(len2 + 1) }  
  
 for (i in 0..len1) {  
 matrix[i][0] = i  
 }  
  
 for (j in 0..len2) {  
 matrix[0][j] = j  
 }  
  
 for (i in 1..len1) {  
 for (j in 1..len2) {  
 val cost = if (search[i - 1] == result[j - 1]) 0 else 1  
 matrix[i][j] = minOf(  
 matrix[i - 1][j] + 1,  
 matrix[i][j - 1] + 1,  
 matrix[i - 1][j - 1] + cost  
 )  
 }  
 }  
  
 val maxLen = maxOf(len1, len2)  
 val similarity = 1.0 - matrix[len1][len2].toDouble() / maxLen.toDouble()  
 return similarity \* 100  
}

En resumen, la distancia de Levenshtein es la mejor herramienta para calcular afinidades mediante cadenas por lo que el proceso es extremadamente efectivo.

### 9.4 Obtener consejos

La función suspendida ‘getConsejos’ utiliza corrutinas para obtener con threads mediante DAO’s los consejos y sugerencias a través de el método ‘getAllActiveCon’ se obtienen todos los consejos activos y se almacenan en ‘consejos’, también se llama el método ‘getAllCon’ el cual devuelve todos los consejos y sugerencias y finalmente devuelve la lista de los consejos activos.

private suspend fun getConsejos(): List<ConySug> {  
 withContext(Dispatchers.IO) {  
 val conySugDao = Stlite.getInstance(requireContext()).getConySugDao()  
  
 consejos = conySugDao.getAllActiveCon()  
 val allCon = conySugDao.getAllCon()  
 Log.v("ALLCON", "$allCon")  
 Log.v("ACTIVECON", "$consejos")  
 }  
 return consejos  
}

En resumen, la función carga los consejos activos y depura los consejos que no están activos.

### 9.5 Consejos y Sugerencias

El objeto ‘Consejo’ tiene una posición y vista asignada los cuales se sitúan dentro del ‘ViewHolder’, dentro de estos también se establece el color ‘CardView’, el estilo ‘Style’ y 2 botones, los cuales permiten declinar el consejo y eliminarlo o aceptarlo lo cual redireccionaría a la sección del consejo.

Imagen 32, consejos

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

override fun onBindViewHolder(holder: ConsejoViewHolder, position: Int) {  
 val consejo = consejos[position]  
 holder.cardView.setCardBackgroundColor(when (consejo.style) {  
 1 -> resources.getColor(R.color.Y1)  
 2 -> resources.getColor(R.color.O2)  
 3 -> resources.getColor(R.color.R2)  
 else -> resources.getColor(R.color.G4)  
 })  
 holder.consejo.text = consejo.nombre  
 holder.descripcion.text = consejo.contenido  
  
 holder.accept.setOnClickListener {  
 lifecycleScope.launch {  
 acceptedOrRejected(consejo.idcon, 2)  
 val intent = Intent(activity, Index::class.java)  
 var goTo = when (consejo.idcon) {  
 in 0..99 -> 0  
 in 100..199 -> 4  
 in 200..299 -> 4  
 in 300..399 -> 0  
 in 400..499 -> 2  
 in 500..599 -> 1  
 in 600..699 -> 2  
 in 700..799 -> 3  
 in 800..899 -> 1  
 else -> 2  
 }  
 val fragmento = when (consejo.idcon) {  
 500L, 501L, 700L, in 800L..802L -> 1  
 0L, 1L, 200L, 204L, 205L, 502L -> 2  
 203L -> {  
 goTo = 0  
 2  
 }  
 201L, 202L -> 3  
 400L, in 600L..604L -> 4  
 401L -> 5  
 100L, 101L -> 6  
 else -> 0  
 }  
 Log.v("Intent data", "${consejo.idcon}, $goTo, $fragmento")  
 intent.putExtra("currentView", goTo)  
 intent.putExtra("fragToGo", fragmento)  
 startActivity(intent)  
 }  
 }  
 holder.ignore.setOnClickListener {  
 lifecycleScope.launch {  
 acceptedOrRejected(consejo.idcon, 3)  
 goodbye(holder.itemView)  
 }  
 }  
 }  
  
  
 override fun getItemCount(): Int {  
 Log.v("size de Consejossss", consejos.size.toString())  
 return consejos.size  
 }  
  
 suspend fun acceptedOrRejected(  
 id: Long,  
 estado: Int,  
 ) {  
 withContext(Dispatchers.IO) {  
 val conySugDao = Stlite.getInstance(requireContext()).getConySugDao()  
  
 val nombre = conySugDao.getNombre(id.toInt())  
 val contenido = conySugDao.getContenido(id.toInt())  
 //val estado = conySugDao.getEstado(id.toInt())  
 val flag = conySugDao.getFlag(id.toInt())  
 val type = conySugDao.getType(id.toInt())  
 val style = conySugDao.getStyle(id.toInt())  
 val pressedSug = ConySug(  
 idcon = id,  
 nombre = nombre,  
 contenido = contenido,  
 estado = estado,  
 flag = flag,  
 type = type,  
 style = style  
 )  
  
 conySugDao.updateCon(pressedSug)  
 val montos = conySugDao.getAllAcceptedCon()  
 Log.i("ALL MONTOS", montos.toString())  
 }  
 }  
  
 fun goodbye (viewToAnimate: View) {  
 val bounceLeft = ObjectAnimator.ofFloat(viewToAnimate, "translationX", 0f, -100f)  
 bounceLeft.duration = 100  
 bounceLeft.interpolator = AccelerateDecelerateInterpolator()  
  
 val bounceRight = ObjectAnimator.ofFloat(viewToAnimate, "translationX", -100f, 800f)  
 bounceRight.duration = 500  
  
 val animatorSet = AnimatorSet()  
 animatorSet.playSequentially(bounceLeft, bounceRight)  
 animatorSet.start()  
  
 animatorSet.addListener(object : Animator.AnimatorListener {  
 override fun onAnimationStart(p0: Animator) {}  
  
 override fun onAnimationEnd(p0: Animator) {  
 parentFragmentManager.beginTransaction()  
 .replace(R.id.finanzas\_container, finanzasConySug()).addToBackStack(null)  
 .commit()  
 }  
  
 override fun onAnimationCancel(p0: Animator) {}  
  
 override fun onAnimationRepeat(p0: Animator) {}  
 })  
  
 animatorSet.start()  
 }  
 }  
}

En resumen, la función se encarga de obtener y mostrar los consejos desde la bd y mostrarlos en el ViewHolder.

### 9.6 Eventos

La clase interna ‘EventosAdapter’ se encarga de adaptar los datos de eventos a una vista de lista ‘RecyclerView’ el cual sirve como contenedor para los elementos de la lista Eventos. Después se obtienen los valores correspondientes de cada evento como lo son el nombre, fecha, etiqueta, entre otros.

Imagen 33, fechas especiales

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

private inner class EventosAdapter(private val eventos: List<Eventos>) :  
 RecyclerView.Adapter<EventosAdapter.EventosViewHolder>() {  
 inner class EventosViewHolder(  
 itemView: View,  
 val nombreTextView: TextView,  
 val fechaTextView: TextView,  
 val etiquetaTextView: TextView,  
 val optionsM: Button  
 ) : RecyclerView.ViewHolder(itemView)  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): EventosViewHolder {  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.item\_evento, parent, false)  
 val nombreTextView = itemView.findViewById<TextView>(R.id.ENombre)  
 val fechaTextView = itemView.findViewById<TextView>(R.id.EFecha)  
 val etiquetaTextView = itemView.findViewById<TextView>(R.id.EEtiqueta)  
 val optionsM = itemView.findViewById<Button>(R.id.itemOptions)  
 return EventosViewHolder(  
 itemView,  
 nombreTextView,  
 fechaTextView,  
 etiquetaTextView,  
 optionsM  
 )  
 }

En resumen, el código realiza la estructura básica para mostrar la lista de eventos con todos los datos relevantes de los mismos.

### 9.7 Agregar Evento

El código se encarga de aceptar los datos que ingrese el usuario respecto al evento, en este caso se verifica que cumpla con criterios para asegurar un funcionamiento adecuado. Al guardar el evento se guarda en el registro la frecuencia del evento, su concepto, la fecha, etiqueta y demás. Para agregar dichos eventos se utiliza la función ‘eventoadd’.

Imagen 34, añadir evento

Tabla

Descripción generada automáticamente

if (label != 0 && concepto != "" && selectedDay != 39) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 fecha = when (frecuencia) {  
 0 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaField.year  
 val datedate = "$year$fMonth$fDay"  
 val fsi: Int = datedate.replace("-", "").toInt()  
  
 fsi  
 } // Único  
 30 -> {  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
  
 intday  
 }  
 61, 91, 122, 183, 365 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val datedate = "5$fMonth$fDay"  
 val fsi: Int = datedate.toInt()  
  
 fsi  
 } // Mensuales  
 else -> {  
 val intyear = binding.FechaField.year - 1900  
 Log.w("YEAR", intyear.toString())  
 val intmonth = binding.FechaField.month  
 Log.w("MONTH", intmonth.toString())  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
 val datedate = "$intyear$intmonth$intday"  
 Log.w("DATE", datedate)  
  
 datedate.toInt()  
 } // Catch que agarra la fecha actual  
 }  
  
 val adddateStr: String = LocalDate.now().toString()  
 val adddate = adddateStr.replace("-", "").toInt()  
  
 Log.v("Concepto", concepto)  
 Log.v("Fecha", fecha.toString())  
 Log.v("Frecuencia", frecuencia.toString())  
 Log.v("Etiqueta", label.toString())  
 Log.v("Addate", adddate.toString())  
 lifecycleScope.launch {  
 eventoadd(  
 concepto,  
 fecha,  
 frecuencia,  
 label,  
 estado,  
 adddate  
 )  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.finanzas\_container, finanzasEventos()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "No pueden haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
  
}

En resumen, el código se encarga de confirmar los datos ingresados por el usuario, así como procesar los datos antes de ingresar el evento.

### 9.8 Update eventos

El fragmento es el encargado de actualizar los eventos, para que el cambio se pueda efectuar se deben de cumplir 3 condiciones: que ‘label’ no sea igual a cero, ‘concepto’ no este vacío y ‘selectedDay’ no sea igual a 39, si todas estas se cumplen se puede actualizar el evento llamando a la función ‘eventoupdate’, una vez se efectúe el cambio se arroja un toast confirmando el cambio.

if (label != 0 && concepto != "" && selectedDay != 39) {  
 val confirmDialog = AlertDialog.Builder(requireContext())  
 .setTitle("¿Seguro que quieres guardar cambios?")  
 .setPositiveButton("Guardar") { dialog, \_ ->  
 fecha = when (frecuencia) {  
 0 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val year = binding.FechaField.year  
 val datedate = "$year$fMonth$fDay"  
 val fsi: Int = datedate.replace("-", "").toInt()  
  
 fsi  
 } // Único  
 30 -> {  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
  
 intday  
 }  
 61, 91, 122, 183, 365 -> {  
 val day = binding.FechaField.dayOfMonth  
 val fDay = String.format("%02d", day)  
 val month = binding.FechaField.month + 1  
 val fMonth = String.format("%02d", month)  
 val datedate = "5$fMonth$fDay"  
 val fsi: Int = datedate.toInt()  
  
 fsi  
 } // Mensuales  
 else -> {  
 val intyear = binding.FechaField.year - 1900  
 Log.w("YEAR", intyear.toString())  
 val intmonth = binding.FechaField.month  
 Log.w("MONTH", intmonth.toString())  
 val intday = binding.FechaField.dayOfMonth  
 Log.w("DAY", intday.toString())  
 val datedate = "$intyear$intmonth$intday"  
 Log.w("DATE", datedate)  
  
 datedate.toInt()  
 } // Catch que agarra la fecha actual  
 }  
  
 Log.v("Concepto", concepto)  
 Log.v("Fecha", fecha.toString())  
 Log.v("Frecuencia", frecuencia.toString())  
 Log.v("Etiqueta", label.toString())  
 lifecycleScope.launch {  
 if (ide != null) {  
 eventoupdate(  
 ide.toLong(),  
 concepto,  
 fecha,  
 frecuencia,  
 label,  
 estado,  
 )  
 }  
 }  
 dialog.dismiss()  
 parentFragmentManager.beginTransaction()  
 .setCustomAnimations(R.anim.fromright, R.anim.toleft)  
 .replace(R.id.finanzas\_container, finanzasEventos()).addToBackStack(null).commit()  
 }  
 .setNegativeButton("Cancelar") { dialog, \_ ->  
 dialog.dismiss()  
 }  
 .create()  
  
 confirmDialog.show()  
 } else {  
 Toast.makeText(  
 requireContext(),  
 "No pueden haber campos vacíos",  
 Toast.LENGTH\_SHORT  
 ).show()  
 }  
  
}

En resumen, el código se encarga de corroborar que las condiciones de cambio sean las adecuadas, así como que el cambio se efectúe.

### 9.9 Estadísticas de ahorro

El código cuenta con 2 funciones las cuales calculan el ahorro del usuario, la primera ‘getAhorros’ calcula los ahorros diarios de las últimas 5 semanas mediante corrutinas e ingresa la información obtenida dentro de un mapa ‘ahorrosMap’. La función ‘getAhorrosXDia’ calcula el ahorro según el día que se de y se ubique en ‘ahorrosMap’.

Imagen 35, gráficas

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

private fun getAhorros() {  
 val today = LocalDate.now()  
 val weeksAgo = today.minusWeeks(5)  
  
 val daysInRange = ChronoUnit.DAYS.between(weeksAgo, today)  
  
 for (i in 0 until daysInRange) {  
 lifecycleScope.launch {  
 val currentDate = weeksAgo.plusDays(i)  
 val dayAhorro = getAhorrosXDia(currentDate)  
 ahorrosMap[currentDate] = dayAhorro  
 }  
 }  
}  
private suspend fun getAhorrosXDia(date: LocalDate): Float {  
 var ingresos = 0F  
 var gastos = 0F  
 withContext(Dispatchers.IO) {  
 val todayInt = date.format(DateTimeFormatter.BASIC\_ISO\_DATE).toInt()  
 val dom = date.dayOfMonth  
 val w = date.dayOfWeek.value  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
 Log.v("FECHA", "$todayInt / $dom / $dow")  
  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 ingresos = montoDao.getStatI(todayInt, dom, dow, 100, todayInt).sumOf { it.valor }.toFloat()  
 gastos = montoDao.getStatG(todayInt, dom, dow, 100, todayInt).sumOf { it.valor }.toFloat()  
 }  
 Log.v("INGRESOS & GASTOS", "$ingresos - $gastos = ${ingresos - gastos}")  
 return ingresos - gastos  
}

### 9.10 Valor del dólar

El código utiliza 2 funciones, la primera ‘getDollar’ realiza 2 quieres para obtener la taza de compra y venta del dólar, mediante volley se accede a ‘getAhorros’ lo que devuelve los ahorros del usuario y se retorna el valor de sus ahorros en dólares.

private suspend fun getDollar() {  
  
 val durl = "http://savetrack.com.mx/dlrvalCompra.php"  
 val durl2 = "http://savetrack.com.mx/dlrvalVenta.php"  
 val queue: RequestQueue = Volley.newRequestQueue(requireContext())  
 val checkDollar = StringRequest(  
 Request.Method.GET, durl,  
 { response ->  
 dollarC = response.toString().toFloat()  
 Log.d("DÓLAR COMPRA", dollarC.toString())  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(),  
 "No se ha podido conectar al valor del dólar hoy",  
 Toast.LENGTH\_SHORT  
 ).show()  
 Log.d("error => $error", "SIE API ERROR")  
 }  
 )  
 queue.add(checkDollar)  
 val checkDollar2 = StringRequest(  
 Request.Method.GET, durl2,  
 { response ->  
 dollarV = response.toString().toFloat()  
 Log.d("DÓLAR VENTA", dollarV.toString())  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(),  
 "No se ha podido conectar al valor del dólar hoy",  
 Toast.LENGTH\_SHORT  
 ).show()  
 Log.d("error => $error", "SIE API ERROR")  
 }  
 )  
 queue.add(checkDollar2)  
 delay(400)  
}

En resumen, el código se dedica a obtener el valor del dólar en compra y venta y comparar su valor con los ahorros del usuario.

### 9.11Valor de divisas

La función ‘getDivisas’ utiliza volley para realizar distintas quieres HTTP a la API de divisas mediante una URL base se obtienen las tarifas de cambio respecto al peso, se calcula un rango de fechas de la última semana y se especifican las tarifas de cambio dentro de ‘currencies’, se recibe un JSON y dentro de ‘currencyData’ se comienza a almacenar la información.

private fun getDivisas() {  
 val baseUrl = "http://savetrack.com.mx/divisas.php?basecurrency=MXN"  
  
 val today = LocalDate.now()  
 val ago = today.minusDays(7)  
 val daysInRange = ChronoUnit.DAYS.between(ago, today)  
 val dateFormat = DateTimeFormatter.ofPattern("yyyy-MM-dd")  
 val queue: RequestQueue = Volley.newRequestQueue(requireContext())  
 for (i in 0 .. daysInRange) {  
 val date = ago.plusDays(i)  
 val formattedDate = date.format(dateFormat)  
 val apiUrl = "$baseUrl&date=$formattedDate&currencies=${currencies.joinToString(",")}"  
  
 val checkDollar = StringRequest(  
 Request.Method.GET, apiUrl,  
 { response ->  
 Log.v("RES", response)  
 try {  
 val responseData = JSONObject(response)  
 Log.v("Try", "$responseData")  
 for (currency in currencies) {  
 val value = 1/responseData.optDouble(currency)  
 Log.v("VALUES", "$value")  
 if (!currencyData.containsKey(currency)) {  
 currencyData[currency] = mutableListOf()  
 }  
 currencyData[currency]?.add(value.toFloat())  
 }  
 } catch (e: JSONException) {  
 e.printStackTrace()  
 }  
 },  
 { error ->  
 Toast.makeText(  
 requireContext(),  
 "No se ha podido conectar al valor del dólar hoy",  
 Toast.LENGTH\_SHORT  
 ).show()  
 Log.d("error => $error", "SIE API ERROR")  
 }  
 )  
 queue.add(checkDollar)  
 }  
}

En resumen, la función obtiene el valor de las divisas en los últimos 7 días y las almacena.

### 9.12 Gráfica de divisas

La función ‘setData’ configura y actualiza el gráfico de líneas ‘LineChart’ el cual dependiendo de la variable ‘position’ realiza acciones para configurar el gráfico. Se crean 3 listas de datos las cuales son graficadas en la ‘LineChart’ teniendo como referencia cada una una divisa distinta. Se configuran los colores, estilos y tamaños en conjuntos.

Imagen 36, gráfico divisas

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

@SuppressLint("UseCompatLoadingForDrawables")  
private fun setData(count: Int, chart: LineChart, position: Int) {  
 val values = Stack<Entry>()  
 val values2 = Stack<Entry>()  
 val values3 = Stack<Entry>()  
 val range = 0F  
  
 Log.v("DIVISAS", currencyData.toString())  
 val ahorros = ahorrosMap.toSortedMap().values.toList()  
 val size = ahorros.size - 1  
 when (position) {  
 0 -> {  
 var j = 0F  
 for (i in size - count .. size) {  
 j++  
 values.add(Entry(j, ahorros[i], R.drawable.ic\_bubbles))  
 values2.add(Entry(j, ahorros[i], R.drawable.ic\_bubbles))  
 }  
  
 val setA: LineDataSet  
 val setB: LineDataSet  
 if (chart.data != null && chart.data.dataSetCount > 0) {  
 setA = chart.data.getDataSetByIndex(0) as LineDataSet  
 setA.values = values  
 setA.notifyDataSetChanged()  
 setB = chart.data.getDataSetByIndex(0) as LineDataSet  
 setB.values = values2  
 setB.notifyDataSetChanged()  
  
 chart.data.notifyDataChanged()  
 chart.notifyDataSetChanged()  
 } else {  
 setA = LineDataSet(values, "Estático en MXN")  
 setA.setDrawIcons(false)  
 setA.enableDashedLine(16f, 0f, 0f)  
 setA.color = R.color.P1  
 setA.setCircleColor(R.color.G0)  
 setA.lineWidth = 10f  
 setA.circleRadius = 3f  
 setA.setDrawCircleHole(true)  
 setA.formLineWidth = 1f  
 setA.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setA.formSize = 15f  
 setA.valueTextSize = 9f  
 setA.enableDashedHighlightLine(10f, 5f, 0f)  
 setA.setDrawFilled(false)  
 setA.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setA.fillColor = R.color.B1  
  
 setB = LineDataSet(values2, "Estático en USD")  
 setB.setDrawIcons(false)  
 setB.enableDashedLine(16f, 0f, 0f)  
 setB.color = R.color.P1  
 setB.setCircleColor(R.color.G0)  
 setB.lineWidth = 10f  
 setB.circleRadius = 3f  
 setB.setDrawCircleHole(true)  
 setB.formLineWidth = 1f  
 setB.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setB.formSize = 15f  
 setB.valueTextSize = 9f  
 setB.enableDashedHighlightLine(10f, 5f, 0f)  
 setB.setDrawFilled(false)  
 setB.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setB.fillColor = R.color.R0  
  
 val dataSets: ArrayList<ILineDataSet> = ArrayList()  
 dataSets.add(setA)  
 dataSets.add(setB)  
 val data = LineData(dataSets)  
  
 chart.data = data  
 }  
 }  
 1 -> {  
 val dolaresList = currencyData["USD"]  
 var j = 0F  
 if (dolaresList != null) {  
 for (i in dolaresList.size - count - 1 until dolaresList.size) {  
 j++  
 val current = dolaresList[i]  
 Log.v("CURRENT COMPRA", dollarC.toString())  
 values.add(Entry(j, current, R.drawable.ic\_bubbles))  
 }  
 }  
  
 val setA: LineDataSet  
 val setB: LineDataSet  
 if (chart.data != null && chart.data.dataSetCount > 0) {  
 setA = chart.data.getDataSetByIndex(0) as LineDataSet  
 setA.values = values  
 setA.notifyDataSetChanged()  
 setB = chart.data.getDataSetByIndex(0) as LineDataSet  
 setB.values = values  
 setB.notifyDataSetChanged()  
  
 chart.data.notifyDataChanged()  
 chart.notifyDataSetChanged()  
 } else {  
 setA = LineDataSet(values, "Compra")  
 setA.setDrawIcons(false)  
 setA.enableDashedLine(16f, 0f, 0f)  
 setA.color = R.color.P1  
 setA.setCircleColor(R.color.G0)  
 setA.lineWidth = 10f  
 setA.circleRadius = 3f  
 setA.setDrawCircleHole(true)  
 setA.formLineWidth = 1f  
 setA.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setA.formSize = 15f  
 setA.valueTextSize = 9f  
 setA.enableDashedHighlightLine(10f, 5f, 0f)  
 setA.setDrawFilled(false)  
 setA.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setA.fillColor = R.color.B1  
  
 setB = LineDataSet(values, "Venta")  
 setB.setDrawIcons(false)  
 setB.enableDashedLine(16f, 0f, 0f)  
 setB.color = R.color.P1  
 setB.setCircleColor(R.color.G0)  
 setB.lineWidth = 10f  
 setB.circleRadius = 3f  
 setB.setDrawCircleHole(true)  
 setB.formLineWidth = 1f  
 setB.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setB.formSize = 15f  
 setB.valueTextSize = 9f  
 setB.enableDashedHighlightLine(10f, 5f, 0f)  
 setB.setDrawFilled(false)  
 setB.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setB.fillColor = R.color.R0  
  
 val dataSets: ArrayList<ILineDataSet> = ArrayList()  
 dataSets.add(setA)  
 dataSets.add(setB)  
 val data = LineData(dataSets)  
  
 chart.data = data  
 }  
 }  
 2 -> {  
 val usd = "USD"  
 val eur = "EUR"  
 val cad = "CAD"  
 val dolaresList = currencyData[usd]  
 val eurosList = currencyData[eur]  
 val canadaList = currencyData[cad]  
  
 var j = 0F  
 if (dolaresList != null) {  
 for (i in dolaresList.size - count - 1 until dolaresList.size) {  
 j++  
 val current1 = dolaresList[i]  
 values.add(Entry(j, current1, R.drawable.ic\_dollar))  
 val current2 = eurosList?.get(i)  
 values2.add(current2?.let {  
 Entry(j, it, R.drawable.ic\_euro)  
 })  
 val current3 = canadaList?.get(i)  
 values3.add(current3?.let {  
 Entry(j, it, R.drawable.ic\_dollar)  
 })  
 }  
 }  
  
 val setA: LineDataSet  
 val setB: LineDataSet  
 val setC: LineDataSet  
 if (chart.data != null && chart.data.dataSetCount > 0) {  
 setA = chart.data.getDataSetByIndex(0) as LineDataSet  
 setA.values = values  
 setA.notifyDataSetChanged()  
 setB = chart.data.getDataSetByIndex(0) as LineDataSet  
 setB.values = values2  
 setB.notifyDataSetChanged()  
 setC = chart.data.getDataSetByIndex(0) as LineDataSet  
 setC.values = values3  
 setC.notifyDataSetChanged()  
  
 chart.data.notifyDataChanged()  
 chart.notifyDataSetChanged()  
 } else {  
 setA = LineDataSet(values, usd)  
 setA.setDrawIcons(false)  
 setA.enableDashedLine(16f, 0f, 0f)  
 setA.color = R.color.P1  
 setA.setCircleColor(R.color.G0)  
 setA.lineWidth = 10f  
 setA.circleRadius = 3f  
 setA.setDrawCircleHole(true)  
 setA.formLineWidth = 1f  
 setA.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setA.formSize = 15f  
 setA.valueTextSize = 9f  
 setA.enableDashedHighlightLine(10f, 5f, 0f)  
 setA.setDrawFilled(false)  
 setA.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setA.fillColor = R.color.B1  
  
 setB = LineDataSet(values2, eur)  
 setB.setDrawIcons(false)  
 setB.enableDashedLine(16f, 0f, 0f)  
 setB.color = R.color.P1  
 setB.setCircleColor(R.color.G0)  
 setB.lineWidth = 10f  
 setB.circleRadius = 3f  
 setB.setDrawCircleHole(true)  
 setB.formLineWidth = 1f  
 setB.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setB.formSize = 15f  
 setB.valueTextSize = 9f  
 setB.enableDashedHighlightLine(10f, 5f, 0f)  
 setB.setDrawFilled(false)  
 setB.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setB.fillColor = R.color.R0  
  
 setC = LineDataSet(values3, cad)  
 setC.setDrawIcons(false)  
 setC.enableDashedLine(16f, 0f, 0f)  
 setC.color = R.color.P1  
 setC.setCircleColor(R.color.G0)  
 setC.lineWidth = 10f  
 setC.circleRadius = 3f  
 setC.setDrawCircleHole(true)  
 setC.formLineWidth = 1f  
 setC.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 setC.formSize = 15f  
 setC.valueTextSize = 9f  
 setC.enableDashedHighlightLine(10f, 5f, 0f)  
 setC.setDrawFilled(false)  
 setC.fillFormatter = IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 setC.fillColor = R.color.G2  
  
 val dataSets: ArrayList<ILineDataSet> = ArrayList()  
 dataSets.add(setA)  
 dataSets.add(setB)  
 dataSets.add(setC)  
 val data = LineData(dataSets)  
  
 chart.data = data  
 }  
 }  
 else -> {  
 for (i in 0 until count) {  
 values.add(Entry(i.toFloat(), range, R.drawable.ic\_bubbles))  
 }  
  
 val set1: LineDataSet  
 if (chart.data != null && chart.data.dataSetCount > 0) {  
 set1 = chart.data.getDataSetByIndex(0) as LineDataSet  
 set1.values = values  
 set1.notifyDataSetChanged()  
 chart.data.notifyDataChanged()  
 chart.notifyDataSetChanged()  
 } else {  
 set1 = LineDataSet(values, "DataSet 1")  
 set1.setDrawIcons(false)  
 set1.enableDashedLine(16f, 0f, 0f)  
 set1.color = R.color.P1  
 set1.setCircleColor(R.color.R0)  
 set1.lineWidth = 10f  
 set1.circleRadius = 3f  
 set1.setDrawCircleHole(true)  
 set1.formLineWidth = 1f  
 set1.formLineDashEffect = DashPathEffect(floatArrayOf(10f, 5f), 0f)  
 set1.formSize = 15f  
 set1.valueTextSize = 9f  
 set1.enableDashedHighlightLine(10f, 5f, 0f)  
 set1.setDrawFilled(false)  
 set1.fillFormatter =  
 IFillFormatter { \_, \_ -> chart.axisLeft.axisMinimum }  
 set1.fillColor = R.color.G2  
  
 val dataSets: ArrayList<ILineDataSet> = ArrayList()  
 dataSets.add(set1)  
 val data = LineData(dataSets)  
  
 chart.data = data  
 }  
 }  
 }  
}

En resumen, la función ‘setData’ es la encargada de manejar la configuración de los gráficos.

## 10 Containers

### 10.1 Container Index

El código utiliza el método ‘onCreateView’ utilizando la clase ‘ContainerIndexBinding’ para inflar el XML mientras que permite una navegación mediante el fragment manager.

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {  
 binding = ContainerIndexBinding.inflate(inflater, container, false)  
 parentFragmentManager.commit {  
 val fragToGo = arguments?.getInt(fragTo) ?: 0  
 Log.v("fragToGo", "$fragToGo")  
 when (fragToGo) {  
 0 -> replace<indexmain>(R.id.index\_container, "indexmain")  
 1 -> replace<indexmandados>(R.id.index\_container, "indexmandados")  
 2 -> replace<indexPorPagar>(R.id.index\_container, "indexPorPagar")  
 3 -> replace<indexmontoupdate>(R.id.index\_container, "indexmontoupdate")  
 4 -> replace<indexadd>(R.id.index\_container, "indexadd")  
 5 -> replace<indexIngresosList>(R.id.index\_container, "indexIngresosList")  
 6 -> replace<indexGastosList>(R.id.index\_container, "indexGastosList")  
 else -> replace<indexmain>(R.id.index\_container, "indexmain")  
 }  
 setReorderingAllowed(true)  
 addToBackStack(null)  
 }  
 return binding.root  
 }  
 companion object {  
 const val ARG\_SECTION\_NUMBER = "section\_number"  
 private const val fragTo = "frag\_to\_go"  
 @JvmStatic  
 fun newInstance(sectionNumber: Int, fragToGo: Int): containerIndex {  
 return containerIndex().apply {  
 arguments = Bundle().apply {  
 putInt(ARG\_SECTION\_NUMBER, sectionNumber)  
 putInt(fragTo, fragToGo)  
 }  
 }  
 }  
 }  
}

En resumen, el código permite mostrar distintos fragments dentro de un mismo activity permitiendo tener una navegación apropiada.

### 10.2 Container Perfil

El código utiliza el método ‘onCreateView’ utilizando la clase ‘ContainerIndexBinding’ para inflar el XML mientras que permite una navegación mediante el fragment manager.

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {  
 binding = ContainerPerfilBinding.inflate(inflater, container, false)  
 parentFragmentManager.commit {  
 val fragToGo = arguments?.getInt(fragTo) ?: 0  
 Log.v("fragToGo", "$fragToGo")  
 when (fragToGo) {  
 0 -> replace<perfilmain>(R.id.perfil\_container, "perfilmain")  
 1 -> replace<perfileditar>(R.id.perfil\_container, "perfileditar")  
 2 -> replace<Configuracion>(R.id.perfil\_container, "Configuracion")  
 else -> replace<perfilmain>(R.id.perfil\_container, "perfilmain")  
 }  
 setReorderingAllowed(true)  
 addToBackStack(null)  
 }  
 return binding.root  
 }  
 companion object {  
 const val ARG\_SECTION\_NUMBER = "section\_number"  
 private const val fragTo = "frag\_to\_go"  
 @JvmStatic  
 fun newInstance(sectionNumber: Int, fragToGo: Int): containerPerfil {  
 return containerPerfil().apply {  
 arguments = Bundle().apply {  
 putInt(ARG\_SECTION\_NUMBER, sectionNumber)  
 putInt(fragTo, fragToGo)  
 }  
 }  
 }  
 }  
}

En resumen, el código permite mostrar distintos fragments dentro de un mismo activity permitiendo tener una navegación apropiada.

### 10.3 Container Historial

El código utiliza el método ‘onCreateView’ utilizando la clase ‘ContainerIndexBinding’ para inflar el XML mientras que permite una navegación mediante el fragment manager.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 binding = ContainerHistorialBinding.inflate(inflater, container, false)  
 parentFragmentManager.commit {  
 val fragToGo = arguments?.getInt(fragTo) ?: 0  
 Log.v("fragToGo", "$fragToGo")  
 when (fragToGo) {  
 0 -> replace<historialmain>(R.id.historial\_container, "historialmain")  
 1 -> replace<historialAdd>(R.id.historial\_container, "historialAdd")  
 2 -> replace<historialEtiquetas>(R.id.historial\_container, "historialEtiquetas")  
 3 -> replace<historialUpdate>(R.id.historial\_container, "historialUpdate")  
 4 -> replace<historialFavoritos>(R.id.historial\_container, "historialFavoritos")  
 5 -> replace<historialMontosList>(R.id.historial\_container, "historialMontosList")  
 else -> replace<historialmain>(R.id.historial\_container, "historialmain")  
 }  
 setReorderingAllowed(true)  
 addToBackStack(null)  
 }  
 return binding.root  
  
 }  
 companion object {  
 const val ARG\_SECTION\_NUMBER = "section\_number"  
 private const val fragTo = "frag\_to\_go"  
 @JvmStatic  
 fun newInstance(sectionNumber: Int, fragToGo: Int): containerHistorial {  
 return containerHistorial().apply {  
 arguments = Bundle().apply {  
 putInt(ARG\_SECTION\_NUMBER, sectionNumber)  
 putInt(fragTo, fragToGo)  
 }  
 }  
 }  
 }  
}

En resumen, el código permite mostrar distintos fragments dentro de un mismo activity permitiendo tener una navegación apropiada.

### 10.4 Container Planes de Ahorro

El código utiliza el método ‘onCreateView’ utilizando la clase ‘ContainerIndexBinding’ para inflar el XML mientras que permite una navegación mediante el fragment manager.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 binding = ContainerPdaBinding.inflate(inflater, container, false)  
 parentFragmentManager.commit {  
 val fragToGo = arguments?.getInt(fragTo) ?: 0  
 Log.v("fragToGo", "$fragToGo")  
 when (fragToGo) {  
 0 -> replace<planesdeahorromain>(R.id.pda\_container, "planesdeahorromain")  
 1 -> replace<pdaDeudasList>(R.id.pda\_container, "pdaDeudasList")  
 else -> replace<planesdeahorromain>(R.id.pda\_container, "planesdeahorromain")  
 }  
 setReorderingAllowed(true)  
 addToBackStack(null)  
 }  
 return binding.root  
  
 }  
 companion object {  
 const val ARG\_SECTION\_NUMBER = "section\_number"  
 private const val fragTo = "frag\_to\_go"  
 @JvmStatic  
 fun newInstance(sectionNumber: Int, fragToGo: Int): containerPlanesDeAhorro {  
 return containerPlanesDeAhorro().apply {  
 arguments = Bundle().apply {  
 putInt(ARG\_SECTION\_NUMBER, sectionNumber)  
 putInt(fragTo, fragToGo)  
 }  
 }  
 }  
 }  
}

En resumen, el código permite mostrar distintos fragments dentro de un mismo activity permitiendo tener una navegación apropiada.

### 10.5 Container Finanzas

El código utiliza el método ‘onCreateView’ utilizando la clase ‘ContainerIndexBinding’ para inflar el XML mientras que permite una navegación mediante el fragment manager.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 binding = ContainerFinanzasBinding.inflate(inflater, container, false)  
 parentFragmentManager.commit {  
 val fragToGo = arguments?.getInt(fragTo) ?: 0  
 Log.v("fragToGo", "$fragToGo")  
 when (fragToGo) {  
 0 -> replace<finanzasmain>(R.id.finanzas\_container, "finanzasmain")  
 1 -> replace<finanzasEventos>(R.id.finanzas\_container, "finanzasEventos")  
 2 -> replace<finanzasEventosAdd>(R.id.finanzas\_container, "finanzasEventosAdd")  
 3 -> replace<finanzasEventosList>(R.id.finanzas\_container, "finanzasEventosList")  
 4 -> replace<finanzasEventosUpdate>(R.id.finanzas\_container, "finanzasEventosUpdate")  
 5 -> replace<finanzasstatsahorro>(R.id.finanzas\_container, "finanzasstatsahorro")  
 6 -> replace<finanzasConySug>(R.id.finanzas\_container, "finanzasConySug")  
 else -> replace<finanzasmain>(R.id.finanzas\_container, "finanzasmain")  
 }  
 setReorderingAllowed(true)  
 addToBackStack(null)  
 }  
 return binding.root  
  
 }  
 companion object {  
 const val ARG\_SECTION\_NUMBER = "section\_number"  
 private const val fragTo = "frag\_to\_go"  
  
 @JvmStatic  
 fun newInstance(sectionNumber: Int, fragToGo: Int): containerFinanzas {  
 return containerFinanzas().apply {  
 arguments = Bundle().apply {  
 putInt(ARG\_SECTION\_NUMBER, sectionNumber)  
 putInt(fragTo, fragToGo)  
 }  
 }  
 }  
 }  
}

En resumen, el código permite mostrar distintos fragments dentro de un mismo activity permitiendo tener una navegación apropiada.

## 11 Notificaciones

### 11.1 Aumento del dólar

La función ‘callDollarNotif’ recibe 2 parámetros, el primero ‘dolarA’ y el segundo ‘dolarH’ son los valores del dólar ayer y el dólar hoy por lo que mediante volley se hace una quierie HTTP a la API de divisas para obtener el valor y comparar ambos, en caso que ‘notifActive’ no sea 0 (que si tenga las notificaciones activadas) se manda una notificación indicando el crecimiento del dólar respecto al día anterior.

Imagen 37, notificaciones

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

private fun callDollarNotif(dolarA: Float, dolarH: Float) {  
 Log.v("DOLARESHOY", "AYER: $dolarA, HOY: $dolarH")  
  
 val dif = dolarA - dolarH  
 val mindif = 0.05  
 val percent = (dif / dolarA) \* 100  
 if (dif >= mindif && notifActive != 0) {  
 notificationHelper.sendNotification(  
 "General",  
 R.drawable.logo1,  
 "El dólar subió",  
 "Subió $$dif ($percent%)",  
 0,  
 0,  
 0  
 )  
 }  
 if (dif <= mindif && notifActive != 0) {  
 notificationHelper.sendNotification(  
 "General",  
 R.drawable.logo1,  
 "El dólar bajó",  
 "Bajó $$dif ($percent%)",  
 0,  
 0,  
 0  
 )  
 }  
}

En resumen, la función devuelve el incremento que se tuvo respecto al día anterior en el valor del dólar siempre y cuando se tengan las notificaciones activadas.

## 12 Configuración

### 12.1 Configuraciones globales

El código tiene la configuración de varios aspectos los cuales actúan de manera global en la aplicación, ‘isDarkModeEnabled’ cambia el tema al modo oscuro, ‘areNotifEnabled’ permite mandar notificaciones al teléfono, ‘updateTheme’ y ‘updateNotif’ permiten que el tema y las notificaciones se actualicen conforme el evento se presente, todos estos procesos se ejecutan mediante DAO’s.

Imagen 38, configuración

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

private suspend fun isDarkModeEnabled(context: Context): Boolean {  
 var komodo: Boolean  
 withContext(Dispatchers.IO){  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 val mode = assetsDao.getTheme()  
 komodo = mode != 0  
 }  
 return komodo  
}  
  
private suspend fun areNotifEnabled(context: Context): Boolean {  
 var modo: Boolean  
 withContext(Dispatchers.IO){  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 val mode = assetsDao.getNotif()  
 modo = mode != 0  
 }  
 return modo  
}  
  
private suspend fun updateTheme(context: Context, komodo: Long){  
 withContext(Dispatchers.IO){  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 assetsDao.updateTheme(komodo)  
 }  
}  
  
private suspend fun updateNotif(context: Context, modo: Long){  
 withContext(Dispatchers.IO){  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
 assetsDao.updateNotif(modo)  
 }  
}

En resumen, este bloque de código permite que las notificaciones y el tema de la aplicación funcionen y sean actualizados a necesidad.

### 12.2 Cambio de tema

Mediante un ‘IconSwitch’ se cambia entre el modo claro y oscuro, los cuales cambian la apariencia de los elementos de la aplicación, al ejecutarse el comando se inicia una corrutina que llama la función ‘updateTheme’ la cual cambiará el tema por el inactivo y viceversa.

binding.claroscuro.setCheckedChangeListener {  
 when (binding.claroscuro.checked) {  
 IconSwitch.Checked.LEFT -> {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_finanzas)  
 binding.bar.setTextColor(resources.getColor(R.color.X0))  
 binding.Modo.setTextColor(resources.getColor(R.color.X0))  
 binding.Notificaciones.setTextColor(resources.getColor(R.color.X0))  
 binding.faq.setTextColor(resources.getColor(R.color.X0))  
 binding.logout.setTextColor(resources.getColor(R.color.X0))  
 binding.by.setTextColor(resources.getColor(R.color.X0))  
 binding.email.setTextColor(resources.getColor(R.color.X0))  
 binding.goback.setBackgroundResource(R.drawable.ic\_back\_dark)  
 lifecycleScope.launch{  
 updateTheme(requireContext(), 0)  
 }  
 isDarkMode = true  
 }  
 IconSwitch.Checked.RIGHT -> {  
 binding.background.setBackgroundResource(R.drawable.gradient\_background\_finanzas2)  
 binding.bar.setTextColor(resources.getColor(R.color.X4))  
 binding.Modo.setTextColor(resources.getColor(R.color.X4))  
 binding.Notificaciones.setTextColor(resources.getColor(R.color.X4))  
 binding.faq.setTextColor(resources.getColor(R.color.X4))  
 binding.logout.setTextColor(resources.getColor(R.color.X4))  
 binding.by.setTextColor(resources.getColor(R.color.X4))  
 binding.email.setTextColor(resources.getColor(R.color.X4))  
 binding.goback.setBackgroundResource(R.drawable.ic\_back\_light)  
 lifecycleScope.launch{  
 updateTheme(requireContext(), 1)  
 }  
 isDarkMode = false  
 }  
 else -> {}  
 }  
}

En resumen, un switch es el encargado de cambiar el tema dentro de la aplicación por lo que dependiendo que lado esté activado es el tema de la aplicación.

### 12.3 Notificaciones y Manual de Usuario

El switch que activa o desactiva las notificaciones funciona de manera similar al pasado, la diferencia principal radica en el efecto, en este caso permitiendo recibir notificaciones por parte de la aplicación en la función denominada ‘notificame’. El manual de usuario descargado mediante el presionar de un botón el cual ejecuta el ‘Decoder’ con la finalidad de corroborar la conexión a internet y en caso de contar con ella, se descarga el manual.

binding.notificame.setCheckedChangeListener {  
 when (binding.notificame.checked) {  
 IconSwitch.Checked.LEFT -> {  
 lifecycleScope.launch{  
 updateNotif(requireContext(), 1)  
 }  
 notifActive = true  
 }  
 IconSwitch.Checked.RIGHT -> {  
 lifecycleScope.launch{  
 updateNotif(requireContext(), 0)  
 }  
 notifActive = false  
 }  
 else -> {}  
 }  
}  
  
binding.manu.setOnClickListener {  
 val decoder = Decoder(requireContext())  
 if (decoder.hayNet()) {  
 manu()  
 } else {  
 Toast.makeText(requireContext(), "No hay acceso a internet", Toast.LENGTH\_SHORT).show()  
 }  
}

En resumen, el switch de notificaciones funciona prácticamente igual al switch de tema, mientras que el manual de usuario es descargado al presionar un botón siempre y cuando se tenga conexión a internet.

### 12.4 Cerrar sesión

El botón ‘cerrarsesion’ ejecuta una gran cantidad de procesos, el primero es comprobar la conectividad a internet para poder hacer y subir el respaldo de la cuenta, en caso de no tener la conexión se arroja un toast rechazando el cierre de sesión. En caso de contar con internet se lanza la corrutina ‘cerrarSesion’ la cual respalda todas las tablas de la bd room y mediante volley hace quieres HTTP para subir al servidor el respaldo del usuario, esto en formato JSON. Finalmente se cierra la sesión actual y se redirige al ‘login’.

binding.cerrarsesion.setOnClickListener {  
 val decoder = Decoder(requireContext())  
 binding.perame.alpha = 1f  
 if (decoder.hayNet()) {  
 suspend fun cerrarSesion() {  
 val queue = Volley.newRequestQueue(requireContext())  
 withContext(Dispatchers.IO) {  
 val usuarioDao = Stlite.getInstance(requireContext()).getUsuarioDao()  
 val ingresosGastosDao =  
 Stlite.getInstance(requireContext()).getIngresosGastosDao()  
 val montoDao = Stlite.getInstance(requireContext()).getMontoDao()  
 val montoGrupoDao = Stlite.getInstance(requireContext()).getMontoGrupoDao()  
 val gruposDao = Stlite.getInstance(requireContext()).getGruposDao()  
 val labelsDao = Stlite.getInstance(requireContext()).getLabelsDao()  
 val assetsDao = Stlite.getInstance(requireContext()).getAssetsDao()  
 val eventosDao = Stlite.getInstance(requireContext()).getEventosDao()  
 val conySugDao = Stlite.getInstance(requireContext()).getConySugDao()  
 val widgetsDao = Stlite.getInstance(requireContext()).getWidgetsDao()  
  
 val perocuantosmontos = montoDao.getMaxMonto()  
 val perocuantosmontosg = montoGrupoDao.getMaxMontoGrupo()  
 val perocuantosgrupos = gruposDao.getMaxGrupo()  
 val perocuantaslabels = labelsDao.getMaxLabel()  
 val perocuantoseventos = eventosDao.getMaxEvento()  
 val perocuantosconsejos = conySugDao.getMaxConsejo()  
  
 val iduser = usuarioDao.checkId().toLong()  
 val username = usuarioDao.checkName()  
 val edad = usuarioDao.checkAge().toLong()  
 val lachamba = usuarioDao.checkChamba().toLong()  
 val diasaho = usuarioDao.checkDiasaho().toLong()  
 val balance = usuarioDao.checkBalance()  
 val foto = usuarioDao.checkFoto()  
 val meta = usuarioDao.checkMeta()  
 val summaryingresos = ingresosGastosDao.checkSummaryI()  
 val summarygastos = ingresosGastosDao.checkSummaryG()  
 val tema = assetsDao.getTheme().toLong()  
 val lastprocess = assetsDao.getLastProcess()  
  
 val viejoUsuario = Usuario(  
 iduser = iduser,  
 nombre = username,  
 edad = edad,  
 chamba = lachamba,  
 foto = foto,  
 diasaho = diasaho,  
 balance = balance,  
 meta = meta  
 )  
 val viejosIG = IngresosGastos(  
 iduser = iduser,  
 summaryingresos = summaryingresos,  
 summarygastos = summarygastos  
 )  
 val viejosAssets = Assets(  
 idtheme = 0,  
 theme = tema,  
 lastprocess = lastprocess  
 )  
  
  
 val selectedbefore = usuarioDao.getUserData()  
 Log.v("PRE SELECTED USERS", selectedbefore.toString())  
  
 /\*  
 UPLOADING BACKUP  
 \*/  
  
 // Tabla Usuario  
 val jsonObjectUsuario = JSONObject()  
 jsonObjectUsuario.put("iduser", viejoUsuario.iduser)  
 jsonObjectUsuario.put("edad", viejoUsuario.edad)  
 jsonObjectUsuario.put("nombre", viejoUsuario.nombre)  
 jsonObjectUsuario.put("chamba", viejoUsuario.chamba)  
 jsonObjectUsuario.put("foto", viejoUsuario.foto)  
 jsonObjectUsuario.put("diasaho", viejoUsuario.diasaho)  
 jsonObjectUsuario.put("balance", viejoUsuario.balance)  
 jsonObjectUsuario.put("meta", viejoUsuario.meta)  
 jsonObjectUsuario.put("theme", viejosAssets.theme)  
 jsonObjectUsuario.put("lastprocess", viejosAssets.lastprocess)  
  
 // Tabla IngresosGastos  
 val jsonObjectIngresosGastos = JSONObject()  
 jsonObjectIngresosGastos.put("iduser", viejosIG.iduser)  
 jsonObjectIngresosGastos.put("summaryingresos", viejosIG.summaryingresos)  
 jsonObjectIngresosGastos.put("summarygastos", viejosIG.summarygastos)  
  
 // Tabla Monto  
 for (idmonto in 0..perocuantosmontos) {  
 if (montoDao.getConcepto(idmonto) != null) {  
 Log.v("Current idmonto", idmonto.toString())  
 val viejoMonto = Monto(  
 idmonto = montoDao.getIdmonto(idmonto),  
 iduser = montoDao.getIduser(idmonto),  
 concepto = montoDao.getConcepto(idmonto),  
 valor = montoDao.getValor(idmonto),  
 valorfinal = montoDao.getValorFinal(idmonto),  
 fecha = montoDao.getFecha(idmonto),  
 frecuencia = montoDao.getFrecuencia(idmonto),  
 etiqueta = montoDao.getEtiqueta(idmonto),  
 interes = montoDao.getInteres(idmonto),  
 tipointeres = montoDao.getTipoInteres(idmonto),  
 veces = montoDao.getVeces(idmonto),  
 estado = montoDao.getEstado(idmonto),  
 adddate = montoDao.getAdded(idmonto),  
 enddate = montoDao.getEnded(idmonto),  
 cooldown = montoDao.getCooldown(idmonto),  
 delay = montoDao.getDelay(idmonto),  
 sequence = montoDao.getSequence(idmonto)  
 )  
 Log.v("Current monto $idmonto", viejoMonto.toString())  
 val jsonObjectMonto = JSONObject()  
 jsonObjectMonto.put("idmonto", viejoMonto.idmonto)  
 jsonObjectMonto.put("iduser", viejoMonto.iduser)  
 jsonObjectMonto.put("concepto", viejoMonto.concepto)  
 jsonObjectMonto.put("valor", viejoMonto.valor)  
 jsonObjectMonto.put("valorfinal", viejoMonto.valorfinal)  
 jsonObjectMonto.put("fecha", viejoMonto.fecha)  
 jsonObjectMonto.put("frecuencia", viejoMonto.frecuencia)  
 jsonObjectMonto.put("etiqueta", viejoMonto.etiqueta)  
 jsonObjectMonto.put("interes", viejoMonto.interes)  
 jsonObjectMonto.put("tipointeres", viejoMonto.tipointeres)  
 jsonObjectMonto.put("veces", viejoMonto.veces)  
 jsonObjectMonto.put("estado", viejoMonto.estado)  
 jsonObjectMonto.put("adddate", viejoMonto.adddate)  
 jsonObjectMonto.put("enddate", viejoMonto.enddate)  
 jsonObjectMonto.put("cooldown", viejoMonto.cooldown)  
 jsonObjectMonto.put("delay", viejoMonto.delay)  
 jsonObjectMonto.put("sequence", viejoMonto.sequence)  
  
 jsonArrayMonto.put(jsonObjectMonto)  
  
 Log.v("Current object", jsonObjectMonto.toString())  
 Log.v("Current array", jsonArrayMonto.toString())  
 } else {  
 Log.v("Current monto $idmonto", "VACÍO")  
 }  
 }  
  
 // Tabla MontoGrupo  
 for (idmonto in 0..perocuantosmontosg) {  
 if (montoGrupoDao.getIdMonto(idmonto) != null && montoGrupoDao.getIdGrupo(  
 idmonto  
 ) != null  
 ) {  
 Log.v("Current idmonto", idmonto.toString())  
 val viejoMontoGrupo = MontoGrupo(  
 idmonto = montoGrupoDao.getIdMonto(idmonto),  
 idgrupo = montoGrupoDao.getIdGrupo(idmonto),  
 iduser = montoGrupoDao.getIdUser(idmonto),  
 )  
 Log.v("Current label $idmonto", viejoMontoGrupo.toString())  
 val jsonObjectMontoGrupo = JSONObject()  
 jsonObjectMontoGrupo.put("idmonto", viejoMontoGrupo.idmonto)  
 jsonObjectMontoGrupo.put("idgrupo", viejoMontoGrupo.idgrupo)  
 jsonObjectMontoGrupo.put("iduser", viejoMontoGrupo.iduser)  
  
 jsonArrayMontoGrupo.put(jsonObjectMontoGrupo)  
  
 Log.v("Current object", jsonObjectMontoGrupo.toString())  
 Log.v("Current array", jsonArrayMontoGrupo.toString())  
 } else {  
 Log.v("Current montogrupo $idmonto", "VACÍO")  
 }  
 }  
  
 // Tabla Grupos  
 for (Id in 0..perocuantosgrupos) {  
 if (gruposDao.getIdGrupo(Id) != null && gruposDao.getNameG(Id) != null) {  
 Log.v("Current idlabel", Id.toString())  
 val viejoGrupo = Grupos(  
 Id = gruposDao.getIdGrupo(Id),  
 nameg = gruposDao.getNameG(Id),  
 description = gruposDao.getDescription(Id),  
 type = gruposDao.getType(Id),  
 admin = gruposDao.getAdmin(Id),  
 idori = gruposDao.getIdori(Id),  
 color = gruposDao.getColor(Id),  
 enlace = gruposDao.getEnlace(Id)  
 )  
 Log.v("Current label $Id", viejoGrupo.toString())  
 val jsonObjectGrupos = JSONObject()  
 jsonObjectGrupos.put("Id", viejoGrupo.Id)  
 jsonObjectGrupos.put("name", viejoGrupo.nameg)  
 jsonObjectGrupos.put("description", viejoGrupo.description)  
 jsonObjectGrupos.put("type", viejoGrupo.type)  
 jsonObjectGrupos.put("admin", viejoGrupo.admin)  
 jsonObjectGrupos.put("idori", viejoGrupo.idori)  
 jsonObjectGrupos.put("color", viejoGrupo.color)  
 jsonObjectGrupos.put("enlace", viejoGrupo.enlace)  
  
 jsonArrayGrupos.put(jsonObjectGrupos)  
  
 Log.v("Current object", jsonObjectGrupos.toString())  
 Log.v("Current array", jsonArrayGrupos.toString())  
 } else {  
 Log.v("Current grupo $Id", "VACÍO")  
 }  
 }  
  
 // Tabla Labels  
 for (idlabel in 1..perocuantaslabels) {  
 if (labelsDao.getIdLabel(idlabel) != null && labelsDao.getPlabel(idlabel) != null) {  
 Log.v("Current idlabel", idlabel.toString())  
 val viejaLabel = Labels(  
 idlabel = labelsDao.getIdLabel(idlabel),  
 plabel = labelsDao.getPlabel(idlabel),  
 color = labelsDao.getColor(idlabel),  
 estado = labelsDao.getEstado(idlabel)  
 )  
 Log.v("Current label $idlabel", viejaLabel.toString())  
 val jsonObjectLabels = JSONObject()  
 jsonObjectLabels.put("idlabel", viejaLabel.idlabel)  
 jsonObjectLabels.put("plabel", viejaLabel.plabel)  
 jsonObjectLabels.put("color", viejaLabel.color)  
 jsonObjectLabels.put("estado", viejaLabel.estado)  
  
 jsonArrayLabels.put(jsonObjectLabels)  
  
 Log.v("Current object", jsonObjectLabels.toString())  
 Log.v("Current array", jsonArrayLabels.toString())  
 } else {  
 Log.v("Current label $idlabel", "VACÍO")  
 }  
 }  
  
 for (idevento in 1..perocuantoseventos) {  
 if (eventosDao.getIdevento(idevento) != null && eventosDao.getNombre(  
 idevento  
 ) != null  
 ) {  
 Log.v("Current evento", idevento.toString())  
 val viejoEvento = Eventos(  
 idevento = eventosDao.getIdevento(idevento),  
 nombre = eventosDao.getNombre(idevento),  
 fecha = eventosDao.getFecha(idevento),  
 frecuencia = eventosDao.getFrecuencia(idevento),  
 etiqueta = eventosDao.getEtiqueta(idevento),  
 estado = eventosDao.getEstado(idevento),  
 adddate = eventosDao.getAddDate(idevento)  
 )  
 Log.v("Current evento $idevento", viejoEvento.toString())  
 val jsonObjectEventos = JSONObject()  
 jsonObjectEventos.put("idevento", viejoEvento.idevento)  
 jsonObjectEventos.put("nombre", viejoEvento.nombre)  
 jsonObjectEventos.put("fecha", viejoEvento.fecha)  
 jsonObjectEventos.put("frecuencia", viejoEvento.frecuencia)  
 jsonObjectEventos.put("etiqueta", viejoEvento.etiqueta)  
 jsonObjectEventos.put("estado", viejoEvento.estado)  
 jsonObjectEventos.put("adddate", viejoEvento.adddate)  
  
 jsonArrayEventos.put(jsonObjectEventos)  
  
 Log.v("Current object", jsonObjectEventos.toString())  
 Log.v("Current array", jsonArrayEventos.toString())  
 } else {  
 Log.v("Current evento $idevento", "VACÍO")  
 }  
 }  
  
 for (idcon in 1..perocuantosconsejos) {  
 if (conySugDao.getIdcon(idcon) != null && conySugDao.getNombre(idcon) != null) {  
 Log.v("Current idcon", idcon.toString())  
 val viejoConsejo = ConySug(  
 idcon = conySugDao.getIdcon(idcon),  
 nombre = conySugDao.getNombre(idcon),  
 contenido = conySugDao.getContenido(idcon),  
 estado = conySugDao.getEstado(idcon),  
 flag = conySugDao.getFlag(idcon),  
 type = conySugDao.getType(idcon),  
 style = conySugDao.getStyle(idcon)  
 )  
 Log.v("Current consejo $idcon", viejoConsejo.toString())  
 val jsonObjectCon = JSONObject()  
 jsonObjectCon.put("idcon", viejoConsejo.idcon)  
 jsonObjectCon.put("nombre", viejoConsejo.nombre)  
 jsonObjectCon.put("contenido", viejoConsejo.contenido)  
 jsonObjectCon.put("estado", viejoConsejo.estado)  
 jsonObjectCon.put("flag", viejoConsejo.flag)  
 jsonObjectCon.put("type", viejoConsejo.type)  
 jsonObjectCon.put("style", viejoConsejo.style)  
  
 jsonArrayConySug.put(jsonObjectCon)  
  
 Log.v("Current object", jsonObjectCon.toString())  
 Log.v("Current array", jsonArrayConySug.toString())  
 } else {  
 Log.v("Current consejo $idcon", "VACÍO")  
 }  
 }  
  
 Log.v("jsonObjectUsuario", jsonObjectUsuario.toString())  
  
 val uploadurl =  
 "http://savetrack.com.mx/backupput.php?username=$username&backup=$jsonObjectUsuario"  
 val uploadReq: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 uploadurl,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return uploadurl.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", uploadReq.toString()  
 )  
 queue.add(uploadReq)  
  
 val upload2url =  
 "http://savetrack.com.mx/backupput2.php?username=$username&backup=$jsonObjectIngresosGastos"  
 val upload2Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload2url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload2url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload2Req.toString()  
 )  
 queue.add(upload2Req)  
  
 val upload3url =  
 "http://savetrack.com.mx/backupput3.php?username=$username&backup=$jsonArrayMonto"  
 val upload3Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload3url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload3url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload3Req.toString()  
 )  
 queue.add(upload3Req)  
  
 val upload4url =  
 "http://savetrack.com.mx/backupput4.php?username=$username&backup=$jsonArrayMontoGrupo"  
 val upload4Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload4url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload4url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload4Req.toString()  
 )  
 queue.add(upload4Req)  
  
 val upload5url =  
 "http://savetrack.com.mx/backupput5.php?username=$username&backup=$jsonArrayGrupos"  
 val upload5Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload5url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload5url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload5Req.toString()  
 )  
 queue.add(upload5Req)  
  
 val upload6url =  
 "http://savetrack.com.mx/backupput6.php?username=$username&backup=$jsonArrayLabels"  
 val upload6Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload6url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error", "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload6url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload6Req.toString()  
 )  
 queue.add(upload6Req)  
  
 val upload7url =  
 "http://savetrack.com.mx/backupput7.php?username=$username&backup=$jsonArrayEventos"  
 val upload7Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload7url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error",  
 "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload7url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload7Req.toString()  
 )  
 queue.add(upload7Req)  
  
 val upload8url =  
 "http://savetrack.com.mx/backupput8.php?username=$username&backup=$jsonArrayConySug"  
 val upload8Req: StringRequest =  
 object : StringRequest(  
 Method.PUT,  
 upload8url,  
 Response.Listener { response ->  
 Log.d(  
 "response", response  
 )  
 },  
 Response.ErrorListener { error ->  
 Log.e(  
 "API error",  
 "error => $error"  
 )  
 }) {  
 override fun getBody(): ByteArray {  
 return upload8url.toByteArray(  
 Charset.defaultCharset()  
 )  
 }  
 }  
 Log.d(  
 "uploadReq", upload8Req.toString()  
 )  
 queue.add(upload8Req)  
  
 usuarioDao.clean()  
 ingresosGastosDao.clean()  
 montoDao.clean()  
 montoGrupoDao.clean()  
 gruposDao.clean()  
 labelsDao.clean()  
 widgetsDao.clean()  
  
 val selectedafter = usuarioDao.getUserData()  
 Log.v(  
 "POST SELECTED USERS", selectedafter.toString()  
 )  
 }  
 }  
  
 lifecycleScope.launch {  
 cerrarSesion()  
 }  
 val intent = Intent(activity, MainActivity::class.java)  
 startActivity(intent)  
 } else {  
 Toast.makeText(requireContext(), "No hay acceso a internet", Toast.LENGTH\_SHORT).show()  
 binding.perame.alpha = 0f  
 }  
 }  
}

Como resumen, al presionar el botón de cerrar sesión se lleva a cabo el proceso de hacer un respaldo y subirlo al servidor, siempre y cuando se cuente con conexión a internet.

## 13 Decoder

### 13.1 Frecuencia

Se tiene una variable de tipo entero la cual devuelve una cadena con la frecuencia de manera proporcional al número, en este caso al número de días por lo que si 1 es diario, 365 es anual.

fun freq (freq: Int): String {  
 val freqStr = when (freq) {  
 0 -> "Única vez"  
 1 -> "Diario"  
 7 -> "Semanal"  
 14 -> "Quincenal"  
 30 -> "Mensual"  
 61 -> "Bimestral"  
 91 -> "Trimestral"  
 122 -> "Cuatrimestral"  
 183 -> "Semestral"  
 365 -> "Anual"  
 else -> "error"  
 }  
 return freqStr  
}

En resumen, la relación de frecuencia es directamente proporcional a su cantidad, pero en días.

### 13.2 Label

Se tiene una variable de tipo entero que devuelve una cadena con el tipo de ingreso que se tiene.

suspend fun label (label: Int): String {  
 var labelStr: String  
 withContext(Dispatchers.IO) {  
 val labelsDao = Stlite.getInstance(cntxt).getLabelsDao()  
 labelStr = when (label) {  
 10001 -> "Salario"  
 10002 -> "Venta"  
 10003 -> "Beca"  
 10004 -> "Pensión"  
 10005 -> "Manutención"  
 10006 -> "Ingreso pasivo"  
 10007 -> "Regalo"  
 else -> labelsDao.getPlabel(label)  
 }  
 }  
 return labelStr  
}

En resumen, dependiendo el valor de la etiqueta será el tipo de ingreso.

### 13.3 Date

Date acepta un parámetro de tipo entero y devuelve una cadena de texto la cual representa la fecha del valor que se tiene, utiliza un formato yyyyMMdd, por lo que se tienen rangos de 0 a 9999, 0 a 12 y 0 a 31.

fun date (date: Int): String {  
 val datedate = when (date) {  
 in 1..31 -> {  
 val aux = "300012$date"  
 val calendar = Calendar.getInstance()  
 val formatoFecha = SimpleDateFormat("yyyyMMdd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(aux)  
 calendar.time = truefecha  
  
 val now = LocalDate.now()  
 val calendarToday = Calendar.getInstance()  
 val today = formatoFecha.parse(now.toString())  
 calendarToday.time = today  
  
 var month = calendar.get(Calendar.MONTH)  
 if (calendar.get(Calendar.DAY\_OF\_MONTH) > calendarToday.get(Calendar.DAY\_OF\_MONTH)) month++  
 val mes = when (month) {  
 1 -> "Enero"  
 2 -> "Febrero"  
 3 -> "Marzo"  
 4 -> "Abril"  
 5 -> "Mayo"  
 6 -> "Junio"  
 7 -> "Julio"  
 8 -> "Agosto"  
 9 -> "Septiembre"  
 10 -> "Octubre"  
 11 -> "Noviembre"  
 12 -> "Diciembre"  
 else -> "cualquier mes"  
 }  
  
 "$date de $mes"  
 }  
  
 41 -> "Lunes"  
 42 -> "Martes"  
 43 -> "Miércoles"  
 44 -> "Jueves"  
 45 -> "Viernes"  
 46 -> "Sábado"  
 47 -> "Domingo"  
 100 -> "Diario"  
  
 in 50000 .. 59999 -> {  
 val aux = "300$date"  
 val calendar = Calendar.getInstance()  
 val formatoFecha = SimpleDateFormat("yyyyMMdd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(aux)  
 calendar.time = truefecha  
  
 val now = LocalDate.now()  
 val calendarToday = Calendar.getInstance()  
 val today = formatoFecha.parse(now.toString())  
 calendarToday.time = today  
  
 val day = calendar.get(Calendar.DAY\_OF\_MONTH)  
 var month = calendar.get(Calendar.MONTH)  
 month++  
 val mes = when (month) {  
 1 -> "Enero"  
 2 -> "Febrero"  
 3 -> "Marzo"  
 4 -> "Abril"  
 5 -> "Mayo"  
 6 -> "Junio"  
 7 -> "Julio"  
 8 -> "Agosto"  
 9 -> "Septiembre"  
 10 -> "Octubre"  
 11 -> "Noviembre"  
 12 -> "Diciembre"  
 else -> "cualquier mes"  
 }  
  
 "$day de $mes"  
 }  
  
 in 10000000..30000001 -> {  
 val formatoFecha = SimpleDateFormat("yyyyMMdd", Locale.getDefault())  
 val truefecha = formatoFecha.parse(date.toString())  
 val calendar = Calendar.getInstance()  
 calendar.time = truefecha  
  
 val dom = calendar.get(Calendar.DAY\_OF\_MONTH)  
 val w = calendar.get(Calendar.DAY\_OF\_WEEK)  
 var dow = 100  
 when (w) {  
 1 -> dow = 47  
 2 -> dow = 41  
 3 -> dow = 42  
 4 -> dow = 43  
 5 -> dow = 44  
 6 -> dow = 45  
 7 -> dow = 46  
 }  
  
 val yyyy = calendar.get(Calendar.YEAR)  
 val mesesito = when (calendar.get(Calendar.MONTH)) {  
 0 -> "Enero"  
 1 -> "Febrero"  
 2 -> "Marzo"  
 3 -> "Abril"  
 4 -> "Mayo"  
 5 -> "Junio"  
 6 -> "Julio"  
 7 -> "Agosto"  
 8 -> "Septiembre"  
 9 -> "Octubre"  
 10 -> "Noviembre"  
 11 -> "Diciembre"  
 else -> "cualquier mes"  
 }  
 val semanita = when (dow) {  
 47 -> "Domingo"  
 41 -> "Lunes"  
 42 -> "Martes"  
 43 -> "Miércoles"  
 44 -> "Jueves"  
 45 -> "Viernes"  
 46 -> "Sábado"  
 else -> ""  
 }  
  
 "$semanita $dom de $mesesito del $yyyy"  
 }  
 else -> LocalDate.now().toString()  
 }  
  
 return datedate  
}

En resumen, la función utiliza la clase ‘Calendar’ para manipular y decodificar las fechas que se ingresan como número y se obtiene un texto.

### 13.4 hayNet

La función ‘hayNet()’ verifica que exista una conexión a internet mediante ‘ConnectivityManager’ para obtener información sobre la conectividad de red, primero comprueba si está utilizando Wi-Fi o datos móviles, en caso de no contar con conexión a internet de ninguna manera arroja un booleando negativo, en caso de tener la conexión se arroja uno positivo.

fun hayNet(): Boolean {  
 val connectivityManager = cntxt.getSystemService(CONNECTIVITY\_SERVICE) as ConnectivityManager  
  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 val networkCapabilities = connectivityManager.activeNetwork ?: return false  
 val activeNetwork = connectivityManager.getNetworkCapabilities(networkCapabilities) ?: return false  
  
 return when {  
 activeNetwork.hasTransport(NetworkCapabilities.TRANSPORT\_WIFI) -> true  
 activeNetwork.hasTransport(NetworkCapabilities.TRANSPORT\_CELLULAR) -> true  
  
 else -> false  
 }  
 } else {  
 val networkInfo: NetworkInfo? = connectivityManager.activeNetworkInfo  
 return networkInfo?.isConnected == true  
 }  
 }  
}

En resumen, la conectividad se comprueba mediante un servicio y por medio de un booleano se limitan o no múltiples funciones dentro de la aplicación.

## 14 Widgets

### 14.1 Redirect

La actividad ‘activityWidgetRedirect’ utiliza el intent del ‘widgetProviderRedirect’ para obtener la lógica del proovedor, con el intent al hacer clic sobre el widget se creará una instancia de ‘RemoteView’ y se redireccionará al proovedor.

class activityWidgetRedirect : AppCompatActivity() {  
 lateinit var binding: WidgetContainerBinding  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 val intent = Intent(this, widgetProviderRedirect::class.java)  
 intent.action = "android.appwidget.action.APPWIDGET\_UPDATE"  
  
 val pendingIntent = PendingIntent.getBroadcast(this, 30000, intent, PendingIntent.FLAG\_IMMUTABLE or PendingIntent.FLAG\_UPDATE\_CURRENT)  
  
 val views = RemoteViews("com.example.st5", R.layout.item\_widgetredirect)  
 views.setOnClickPendingIntent(R.id.fastAddM, pendingIntent)  
  
 val appWidgetManager = AppWidgetManager.getInstance(this)  
 val appWidgetIds = appWidgetManager.getAppWidgetIds(ComponentName(this, widgetProviderRedirect::class.java))  
 appWidgetManager.updateAppWidget(appWidgetIds, views)  
  
 val resultValue = Intent().putExtra(AppWidgetManager.EXTRA\_APPWIDGET\_ID, appWidgetIds)  
 setResult(RESULT\_OK, resultValue)  
  
 Log.v("Widget creado", "PARA AÑADIR MONTO")  
  
 finish()  
 }  
}

En resumen, el redirect aunque esté bien definido igual depende del provider para poder hacer acciones como redirect o crear widgets.

### 14.2 Reddirect provider

Al tener una función ‘onRecieve’ se invoca tras un intent y se ejecuta la acción del widget la cual es hacer un redirect al fragment de añadir monto, esto se realiza mediante courrutinas y remote views.

override fun onReceive(context: Context, intent: Intent) {  
 super.onReceive(context, intent)  
  
 Log.v("onReceive", "onReceive")  
 if (intent.action == "android.appwidget.action.APPWIDGET\_UPDATE") {  
 val appWidgetManager = AppWidgetManager.getInstance(context)  
 val appWidgetIds = appWidgetManager.getAppWidgetIds(  
 ComponentName(  
 context,  
 widgetProviderRedirect::class.java  
 )  
 )  
 onUpdate(context, appWidgetManager, appWidgetIds)  
 }  
 }  
  
 override fun onUpdate(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetIds: IntArray  
 ) {  
 for (appWidgetId in appWidgetIds) {  
 updateAppWidget(context, appWidgetManager, appWidgetId)  
 Log.v("onUpdate", "onUpdate")  
 }  
 }  
  
 companion object {  
 fun updateAppWidget(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetId: Int  
 ) {  
 var komodo = false  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val assetsDao = Stlite.getInstance(context).getAssetsDao()  
  
 val mode = assetsDao.getTheme()  
 komodo = mode != 0  
 }  
 }  
 Thread.sleep(300)  
 val intent = Intent(context, Index::class.java)  
 intent.putExtra("isDarkMode", komodo)  
 intent.putExtra("currentView", 2)  
 intent.putExtra("fragToGo", 4)  
 val pendingIntent = PendingIntent.getActivity(context, 30000, intent, PendingIntent.FLAG\_IMMUTABLE or PendingIntent.FLAG\_UPDATE\_CURRENT)  
  
 val views = RemoteViews("com.example.st5", R.layout.item\_widgetredirect)  
 views.setOnClickPendingIntent(R.id.fastAddM, pendingIntent)  
  
 Log.v("Añadir monto", "Intent")  
 appWidgetManager.updateAppWidget(appWidgetId, views)  
 }  
 }  
}

En resumen, tras tener un intent, el provider puede ejecutar su tarea designada.

### 14.3 Widget provider gasto

La clase ‘AppWidgetProvider’ es un componente que permite crear widgets así como recibir una acción de difusión la cual en este caso extrae datos de ‘montos’ y actualiza la bd mediante DAO’s para insertar un gasto.

override fun onReceive(context: Context, intent: Intent) {  
 super.onReceive(context, intent)  
  
 if (intent.action == "android.appwidget.action.APPWIDGET\_UPDATE") {  
 montoIdToShow = intent.getLongExtra("IDM", -1)  
 widgetId = intent.getIntExtra("IDW", -1)  
 if (montoIdToShow != -1L && widgetId != -1) {  
 val appWidgetManager = AppWidgetManager.getInstance(context)  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val widgetsDao = Stlite.getInstance(context).getWidgetsDao()  
 val nuevoWidget = Widgets(  
 idwidget = widgetId,  
 idmonto = montoIdToShow  
 )  
  
 if (!widgetsDao.getAllWidgetIds().contains(nuevoWidget.idwidget)) {  
 widgetsDao.insertWidget(nuevoWidget)  
 }  
  
 val allWidgets = widgetsDao.getAllWidgets()  
 Log.v("widget ALL", "$allWidgets")  
  
 val appWidgetIdArray: IntArray = intArrayOf(widgetId)  
 onUpdate(context, appWidgetManager, appWidgetIdArray)  
 }  
 }  
 }  
 }  
 }  
  
 override fun onUpdate(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetIds: IntArray  
 ) {  
 for (appWidgetId in appWidgetIds) {  
 Log.v("widget id at update", "$appWidgetId")  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val widgetsDao = Stlite.getInstance(context).getWidgetsDao()  
 val idMonto = widgetsDao.getIdMontoDeWidget(appWidgetId)  
  
 updateAppWidget(context, appWidgetManager, appWidgetId, idMonto)  
 }  
 }  
 }  
 }  
  
 companion object {  
 fun updateAppWidget(  
 context: Context,  
 appWidgetManager: AppWidgetManager,  
 appWidgetId: Int,  
 montoId: Long  
 ) {  
 val views = RemoteViews(context.packageName, R.layout.item\_widgetfast)  
 Log.v("widget Id", "$appWidgetId")  
 val intent = widgetServiceGasto.newIncrementIntent(context, appWidgetId, montoId)  
 views.setOnClickPendingIntent(R.id.fastAddW, intent)  
  
 widgetServiceGasto.incrementCount(context, appWidgetId)  
  
 val veces = widgetServiceGasto.getVeces(context, appWidgetId, montoId)  
 val concepto = widgetServiceGasto.getConcepto(context, appWidgetId, montoId)  
 val valor = widgetServiceGasto.getValor(context, appWidgetId, montoId)  
 views.setTextViewText(R.id.fastVeces, veces.toString())  
 views.setTextViewText(R.id.fastConcepto, concepto)  
 views.setTextViewText(R.id.fastValor, valor)  
  
 appWidgetManager.updateAppWidget(appWidgetId, views)  
 }  
 }  
}

En resumen, mediante el ‘AppWidgetProvider’ se obtiene la funcionalidad del widget, cabe resaltar que debido a su extrema similitud se explicará únicamente el provider de Gasto ya que el provider de Ingreso funciona prácticamente igual.

### 14.4 Add gasto

Mediante los métodos ‘onBindViewHolder’ y ‘getItemCount’ dentro del recyler view se establecen los datos de la vista a modificar, en este caso ‘monto’ y ‘getItemCount’, tras presionar el widget se ejecutan ambas sumando uno al item count y el monto al gasto neto.

override fun onBindViewHolder(holder: MontoViewHolder, position: Int) {  
 val monto = montos[position]  
 val decoder = Decoder(requireContext())  
 holder.conceptoTextView.text = monto.concepto  
 holder.valorTextView.text = decoder.format(monto.valor).toString()  
 holder.vecesTextView.text = monto.veces.toString()  
 lifecycleScope.launch {  
 holder.etiquetaTextView.text = decoder.label(monto.etiqueta)  
 }  
 holder.addM.setOnClickListener {  
 val appWidgetManager = AppWidgetManager.getInstance(requireContext())  
 val appWidgetIds = appWidgetManager.getAppWidgetIds(ComponentName(requireContext(), widgetProviderGasto::class.java))  
 val idw = appWidgetIds.last()  
 Log.v("IDW", "$idw")  
 val intent = Intent(context, widgetProviderGasto::class.java)  
 intent.action = "android.appwidget.action.APPWIDGET\_UPDATE"  
 intent.putExtra("IDM", monto.idmonto)  
 intent.putExtra("IDW", idw)  
  
 val wId = 10000 + monto.idmonto.toInt()  
 val pendingIntent = PendingIntent.getBroadcast(requireContext(), wId, intent, PendingIntent.FLAG\_IMMUTABLE or PendingIntent.FLAG\_UPDATE\_CURRENT)  
  
 val views = RemoteViews("com.example.st5", R.layout.item\_widgetfast)  
 views.setOnClickPendingIntent(R.id.fastConcepto, pendingIntent)  
  
 appWidgetManager.updateAppWidget(appWidgetIds, views)  
  
 val resultValue = Intent().putExtra(AppWidgetManager.EXTRA\_APPWIDGET\_ID, appWidgetIds)  
 requireActivity().setResult(RESULT\_OK, resultValue)  
  
 Log.v("Widget creado", monto.concepto)  
  
 requireActivity().finish()  
 }  
 if (position == gastos.size - 1){  
 holder.itemView.setBackgroundResource(R.drawable.p1bottomcell)  
 }  
 }  
  
  
 override fun getItemCount(): Int {  
 Log.v("size de montossss", montos.size.toString())  
 return montos.size  
 }  
 }  
}

En resumen, mediante Recycler View se puede interactuar con la bd y el widget de manera simultánea para que trabaje adecuadamente el mismo. Debido a la similitud únicamente se explicará el addGasto a que addIngreso es prácticamente igual.

### 14.5 WidgetServiceGasto

Este código se relaciona con el servicio ‘onStartCommand’ el cual maneja la lógica de incrementar el contador asociado al widget mediante un intent ‘ACTION\_INCREMENT’, al recibirlo extrae el identificador de ‘montoId’ y del ‘widgetId’ para efectuar los cambios dentro de la bd.

Imagen 39, widgets

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {  
 if (intent?.action == ACTION\_INCREMENT) {  
 val montoId = getMontoId(intent)  
 val widgetId = getWidgetId(intent)  
 Log.v("widget IDM", "$montoId")  
 val appWidgetManager = AppWidgetManager.getInstance(this)  
  
 incrementCount(this, widgetId)  
 Log.v("widget service increment", "$widgetId")  
 widgetProviderGasto.updateAppWidget(this, appWidgetManager, widgetId, montoId)  
 }  
  
 return super.onStartCommand(intent, flags, startId)  
 }  
  
 companion object {  
 private const val ACTION\_INCREMENT = "com.example.st5.INCREMENT"  
  
 fun newIncrementIntent(context: Context, widgetId: Int, montoId: Long): PendingIntent {  
 val intent = Intent(context, widgetServiceGasto::class.java)  
 intent.action = ACTION\_INCREMENT  
 intent.putExtra("MONTO\_ID", montoId)  
 intent.putExtra("WIDGET\_ID", widgetId)  
 Log.v("widget IDM", "$montoId")  
 return PendingIntent.getService(  
 context,  
 1000,  
 intent,  
 PendingIntent.FLAG\_IMMUTABLE or PendingIntent.FLAG\_UPDATE\_CURRENT  
 )  
 }  
  
 fun getVeces(context: Context, appWidgetId: Int, montoId: Long): Int {  
 var veces = -1L  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(context).getMontoDao()  
 veces = montoDao.getVeces(montoId.toInt()) + 1  
 }  
 }  
 while (veces == -1L) sleep(10)  
 return veces.toInt()  
 }  
 fun getConcepto(context: Context, appWidgetId: Int, montoId: Long): String {  
 var concepto = ""  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(context).getMontoDao()  
 concepto = montoDao.getConcepto(montoId.toInt())  
 }  
 }  
 while (concepto == "") sleep(10)  
 return concepto  
 }  
 fun getValor(context: Context, appWidgetId: Int, montoId: Long): String {  
 var valor = -1.111  
 val decoder = Decoder(context)  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val montoDao = Stlite.getInstance(context).getMontoDao()  
 valor = montoDao.getValor(montoId.toInt())  
 }  
 }  
 while (valor == -1.111) sleep(10)  
 return "$" + decoder.format(valor)  
 }  
  
 fun incrementCount(context: Context, appWidgetId: Int) {  
 CoroutineScope(Dispatchers.IO).launch {  
 withContext(Dispatchers.IO) {  
 val widgetsDao = Stlite.getInstance(context).getWidgetsDao()  
 val montoId = widgetsDao.getIdMontoDeWidget(appWidgetId)  
 Log.v("widget montoId", "$montoId")  
  
 if (montoId != null && montoId != -0L && montoId != -1L) {  
 val montoDao = Stlite.getInstance(context).getMontoDao()  
 val usuarioDao = Stlite.getInstance(context).getUsuarioDao()  
 val ingresoGastoDao = Stlite.getInstance(context).getIngresosGastosDao()  
  
 val veces = montoDao.getVeces(montoId.toInt())  
 val concepto = montoDao.getConcepto(montoId.toInt())  
 val valor = montoDao.getValor(montoId.toInt())  
 val fecha = montoDao.getFecha(montoId.toInt())  
 val frecuencia = montoDao.getFecha(montoId.toInt())  
 val etiqueta = montoDao.getEtiqueta(montoId.toInt())  
 val interes = montoDao.getInteres(montoId.toInt())  
 val adddate = montoDao.getAdded(montoId.toInt())  
 var nv: Long? = 1  
 if (veces != null)  
 nv = veces + 1  
  
 val sequence = montoDao.getSequence(montoId.toInt())  
 val values =  
 sequence.trim('.').split('.').map { it.toInt() }.toMutableList()  
 if (values.isNotEmpty()) {  
 val lastIndex = values.size - 1  
 values[lastIndex] += 1  
 }  
 val updatedString = values.joinToString(".")  
 val result = "$updatedString."  
  
 val enddate = montoDao.getEnded(montoId.toInt())  
 val valorfinal = montoDao.getValorFinal(montoId.toInt())  
 val tipointeres = montoDao.getTipoInteres(montoId.toInt())  
 val delay = montoDao.getDelay(montoId.toInt())  
 val cooldown = montoDao.getCooldown(montoId.toInt())  
 val iduser = usuarioDao.checkId().toLong()  
 val montoPresionado = Monto(  
 idmonto = montoId,  
 iduser = iduser,  
 concepto = concepto,  
 valor = valor,  
 valorfinal = valorfinal,  
 fecha = fecha,  
 frecuencia = frecuencia,  
 etiqueta = etiqueta,  
 interes = interes,  
 tipointeres = tipointeres,  
 veces = nv,  
 adddate = adddate,  
 enddate = enddate,  
 cooldown = cooldown,  
 delay = delay,  
 sequence = result  
 )  
  
 val totalGastos = ingresoGastoDao.checkSummaryG()  
  
 ingresoGastoDao.updateSummaryG(  
 montoPresionado.iduser.toInt(),  
 totalGastos + montoPresionado.valor  
 )  
 montoDao.updateMonto(montoPresionado)  
 val montos = montoDao.getMonto()  
 Log.i("ALL MONTOS", montos.toString())  
 }  
 }  
 }  
 }  
  
 private fun getMontoId(intent: Intent): Long {  
 Log.v("widget IDM", "${intent.getLongExtra("MONTO\_ID", -1)}")  
 return intent.getLongExtra("MONTO\_ID", -1)  
 }  
 private fun getWidgetId(intent: Intent): Int {  
 Log.v("widget IDW", "${intent.getLongExtra("WIDGET\_ID", -1)}")  
 return intent.getIntExtra("WIDGET\_ID", -1)  
 }  
 }  
}

En resumen, el Código funciona mediante lógica para manejar la interacción del usuario con el widget. Debido a la similitud del código solo se explicará el Service de Gasto ya que el Service de Ingreso es prácticamente igual.

## 15 DAO’s

### 15.1 UsuarioDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface UsuarioDao {  
 @RewriteQueriesToDropUnusedColumns  
 @Query("SELECT \* FROM usuario")  
 suspend fun getUserData(): List<Usuario>  
  
  
 @Query("SELECT iduser FROM usuario")  
 suspend fun checkId(): Int  
 @Query("SELECT nombre FROM usuario")  
 suspend fun checkName(): String  
 @Query("SELECT edad FROM usuario")  
 suspend fun checkAge(): Int  
 @Query("SELECT chamba FROM usuario")  
 suspend fun checkChamba(): Int  
 @Query("SELECT foto FROM usuario")  
 suspend fun checkFoto(): String  
 @Query("SELECT diasaho FROM usuario")  
 suspend fun checkDiasaho(): Int  
 @Query("SELECT balance FROM usuario")  
 suspend fun checkBalance(): Double  
 @Query("SELECT meta FROM usuario")  
 suspend fun checkMeta(): Double  
  
  
 @Query("DELETE FROM usuario")  
 suspend fun clean()  
  
  
 @Insert  
 suspend fun insertUsuario(usuario: Usuario)  
 @Update  
 suspend fun updateUsuario(usuario: Usuario)  
 @Delete  
 suspend fun deleteUsuario(usuario: Usuario)  
  
  
 @Query("UPDATE usuario SET edad = :age WHERE iduser = :id")  
 suspend fun updateAge(id: Int, age: Long?)  
 @Query("UPDATE usuario SET chamba = :field WHERE iduser = :id")  
 suspend fun updateChamba(id: Int, field: Long?)  
 @Query("UPDATE usuario SET meta = :meta WHERE iduser = :id")  
 suspend fun updateMeta(id: Int, meta: Double)  
 @Query("UPDATE usuario SET foto = :photo WHERE iduser = :id")  
 suspend fun updatePhoto(id: Int, photo: String?)  
 @Query("UPDATE usuario SET diasaho = :days WHERE iduser = :id")  
 suspend fun updateDiasaho(id: Int, days: Long)  
 @Query("UPDATE usuario SET balance = :balance WHERE iduser = :id")  
 suspend fun updateBalance(id: Int, balance: Double)  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.2 UserCacheDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface UserCacheDao {  
 @Insert  
 fun insertUserCache(UserCache: UserCache)  
  
 @Query("DELETE FROM usercache")  
 fun clean()  
  
 @Query("SELECT MAX(iduser) FROM usercache")  
 fun getMaxId(): Int  
 @Query("SELECT nombre FROM usercache")  
 fun getAllUserCacheNombres(): List<String>  
 @Query("SELECT nombre FROM usercache WHERE iduser = :id")  
 fun getNombreUserCache(id: Long): String  
 @Query("SELECT iduser FROM usercache WHERE iduser = :id")  
 fun getIdUserCache(id: Long): Long  
 @Query("SELECT \* FROM usercache")  
 fun getAllUserCache(): List<UserCache>  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.3 MontoGrupoDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface MontoGrupoDao {  
 @Query("SELECT \* FROM montogrupo")  
 fun getMontoG(): List<MontoGrupo>  
 @Query("DELETE FROM montogrupo")  
 suspend fun clean()  
  
 @Insert  
 fun insertMontoG(montoGrupo: MontoGrupo)  
  
 @Update  
 fun updateMontoG(montoGrupo: MontoGrupo)  
  
 @Delete  
 fun deleteMontoG(montoGrupo: MontoGrupo)  
  
 @Query("SELECT MAX(idmonto) FROM montogrupo")  
 fun getMaxMontoGrupo(): Int  
  
 @Query("SELECT idmonto FROM montogrupo WHERE idmonto = :id")  
 fun getIdMonto(id: Int): Long  
 @Query("SELECT idgrupo FROM montogrupo WHERE idmonto = :id")  
 fun getIdGrupo(id: Int): Long  
 @Query("SELECT iduser FROM montogrupo WHERE idmonto = :id")  
 fun getIdUser(id: Int): Long  
  
 @Query("SELECT \* FROM montogrupo WHERE idgrupo = :idg")  
 fun getAllMontosdeGrupo(idg: Int): List<MontoGrupo>  
  
 @Query("SELECT \* FROM montogrupo")  
 fun getAllMG(): List<MontoGrupo>  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.4 MontoDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface MontoDao {  
  
 // region BASIC QUERIES  
 @Insert  
 fun insertMonto(monto: Monto)  
  
 @Update  
 fun updateMonto(monto: Monto)  
  
 @Delete  
 fun deleteMonto(monto: Monto)  
  
 @Query("SELECT \* FROM monto")  
 fun getMonto(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE idmonto = :id")  
 fun getM(id: Int): Monto  
  
 @Query("SELECT MAX(idmonto) FROM monto")  
 fun getMaxMonto(): Int  
  
 @Query("DELETE FROM monto")  
 suspend fun clean()  
  
 // endregion  
  
 // region  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND etiqueta = :etiqueta AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)")  
 fun getGR(fecha: Int, dom: Int, dow: Int, dai: Int, etiqueta: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND etiqueta <= 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)")  
 fun getStatG(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)")  
 fun getStatI(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
 // endregion  
  
 // region GET GASTOS/INGRESOS  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000")  
 fun getIngresos(): List<Monto>  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY :filter ASC")  
 fun getIngresosOrdered(filter: String): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000")  
 fun getGastos(): List<Monto>  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY :filter ASC")  
 fun getGastosOrdered(filter: String): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND etiqueta = :e")  
 fun getIngresos(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND etiqueta = :e")  
 fun getGastos(e: Int): List<Monto>  
  
 //endregion  
  
 // region GET POR FILTROS para listas de ingresos  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY concepto ASC")  
 fun getIngresosAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY fecha ASC")  
 fun getIngresosFechados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY valor ASC")  
 fun getIngresosValuados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY frecuencia ASC")  
 fun getIngresosFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY etiqueta ASC")  
 fun getIngresosEtiquetados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY interes ASC")  
 fun getIngresosInteres(): List<Monto>  
  
 // endregion  
  
 // region GET POR FILTROS para listas de gastos  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY concepto ASC")  
 fun getGastosAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY fecha ASC")  
 fun getGastosFechados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY valor ASC")  
 fun getGastosValuados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY frecuencia ASC")  
 fun getGastosFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY etiqueta ASC")  
 fun getGastosEtiquetados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 ORDER BY interes ASC")  
 fun getGastosInteres(): List<Monto>  
  
 // endregion  
  
 // region GET POR FILTROS para listas de ingresos (con etiqueta)  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY concepto ASC")  
 fun getIngresosAlfabetica(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY fecha ASC")  
 fun getIngresosFechados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY valor ASC")  
 fun getIngresosValuados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY frecuencia ASC")  
 fun getIngresosFrecuentes(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY etiqueta ASC")  
 fun getIngresosEtiquetados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY interes ASC")  
 fun getIngresosInteres(e: Int): List<Monto>  
  
 // endregion  
  
 // region GET POR FILTROS para listas de gastos (con etiqueta)  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY concepto ASC")  
 fun getGastosAlfabetica(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY fecha ASC")  
 fun getGastosFechados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY valor ASC")  
 fun getGastosValuados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY frecuencia ASC")  
 fun getGastosFrecuentes(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY etiqueta ASC")  
 fun getGastosEtiquetados(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta = :e ORDER BY interes ASC")  
 fun getGastosInteres(e: Int): List<Monto>  
  
 // endregion  
  
 // region GET ATRIBUTOS  
 @Query("SELECT idmonto FROM monto WHERE idmonto = :id")  
 fun getIdmonto(id: Int): Long  
  
 @Query("SELECT iduser FROM monto WHERE idmonto = :id")  
 fun getIduser(id: Int): Long  
  
 @Query("SELECT concepto FROM monto WHERE idmonto = :id")  
 fun getConcepto(id: Int): String  
  
 @Query("SELECT valor FROM monto WHERE idmonto = :id")  
 fun getValor(id: Int): Double  
  
 @Query("SELECT valorfinal FROM monto WHERE idmonto = :id")  
 fun getValorFinal(id: Int): Double  
  
 @Query("SELECT fecha FROM monto WHERE idmonto = :id")  
 fun getFecha(id: Int): Int  
  
 @Query("SELECT frecuencia FROM monto WHERE idmonto = :id")  
 fun getFrecuencia(id: Int): Int  
  
 @Query("SELECT etiqueta FROM monto WHERE idmonto = :id")  
 fun getEtiqueta(id: Int): Int  
  
 @Query("SELECT interes FROM monto WHERE idmonto = :id")  
 fun getInteres(id: Int): Double  
  
 @Query("SELECT tipointeres FROM monto WHERE idmonto = :id")  
 fun getTipoInteres(id: Int): Int  
  
 @Query("SELECT veces FROM monto WHERE idmonto = :id")  
 fun getVeces(id: Int): Long  
  
 @Query("SELECT estado FROM monto WHERE idmonto = :id")  
 fun getEstado(id: Int): Int  
  
 @Query("SELECT adddate FROM monto WHERE idmonto = :id")  
 fun getAdded(id: Int): Int  
  
 @Query("SELECT enddate FROM monto WHERE idmonto = :id")  
 fun getEnded(id: Int): Int?  
  
 @Query("SELECT cooldown FROM monto WHERE idmonto = :id")  
 fun getCooldown(id: Int): Int  
  
 @Query("SELECT delay FROM monto WHERE idmonto = :id")  
 fun getDelay(id: Int): Int  
  
 @Query("SELECT sequence FROM monto WHERE idmonto = :id")  
 fun getSequence(id: Int): String  
  
 // endregion  
  
 // region GET POR FECHA ESPECÍFICA  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)")  
 fun getMontoXFecha(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY concepto ASC")  
 fun getMontoXFechaAlfabetica(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY valor ASC")  
 fun getMontoXFechaValuados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY frecuencia ASC")  
 fun getMontoXFechaFrecuentes(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY etiqueta ASC")  
 fun getMontoXFechaEtiquetados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY interes ASC")  
 fun getMontoXFechaInteres(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) ORDER BY veces ASC")  
 fun getMontoXFechaVeces(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label")  
 fun getMontoXFecha(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY concepto ASC")  
 fun getMontoXFechaAlfabetica(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY valor ASC")  
 fun getMontoXFechaValuados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY frecuencia ASC")  
 fun getMontoXFechaFrecuentes(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY etiqueta ASC")  
 fun getMontoXFechaEtiquetados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY interes ASC")  
 fun getMontoXFechaInteres(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE adddate <= :din AND enddate >= :din AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND etiqueta = :label ORDER BY veces ASC")  
 fun getMontoXFechaVeces(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int, label: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto")  
 fun getMontoXFecha(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY concepto ASC")  
 fun getMontoXFechaAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY valor ASC")  
 fun getMontoXFechaValuados(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY frecuencia ASC")  
 fun getMontoXFechaFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY etiqueta ASC")  
 fun getMontoXFechaEtiquetados(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY interes ASC")  
 fun getMontoXFechaInteres(): List<Monto>  
  
 @Query("SELECT \* FROM monto ORDER BY veces ASC")  
 fun getMontoXFechaVeces(): List<Monto>  
  
 // endregion  
  
 // region GET INGRESOS POR FECHA ESPECÍFICA  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND adddate <= :din")  
 fun getIXFecha(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY concepto ASC")  
 fun getIXFechaAlfabetica(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY valor ASC")  
 fun getIXFechaValuados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY frecuencia ASC")  
 fun getIXFechaFrecuentes(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY etiqueta ASC")  
 fun getIXFechaEtiquetados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY interes ASC")  
 fun getIXFechaInteres(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai) AND enddate >= :din AND adddate <= :din ORDER BY veces ASC")  
 fun getIXFechaVeces(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000")  
 fun getIXFecha(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY concepto ASC")  
 fun getIXFechaAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY valor ASC")  
 fun getIXFechaValuados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY frecuencia ASC")  
 fun getIXFechaFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY etiqueta ASC")  
 fun getIXFechaEtiquetados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY interes ASC")  
 fun getIXFechaInteres(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 ORDER BY veces ASC")  
 fun getIXFechaVeces(): List<Monto>  
  
 // endregion  
  
  
 // region GET GASTOS POR FECHA ESPECÍFICA  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY estado DESC")  
 fun getGXFecha(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY concepto ASC")  
 fun getGXFechaAlfabetica(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND adddate <= :din ORDER BY valor ASC")  
 fun getGXFechaValuados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY frecuencia ASC")  
 fun getGXFechaFrecuentes(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY etiqueta ASC")  
 fun getGXFechaEtiquetados(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY interes ASC")  
 fun getGXFechaInteres(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (((estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) AND (fecha = :fecha OR fecha = :dom OR fecha = :dow OR fecha = :dai)) OR delay >= 1) AND enddate >= :din AND adddate <= :din ORDER BY veces ASC")  
 fun getGXFechaVeces(fecha: Int, dom: Int, dow: Int, dai: Int, din: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8)")  
 fun getGXFecha(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) ORDER BY concepto ASC")  
 fun getGXFechaAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) ORDER BY valor ASC")  
 fun getGXFechaValuados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 83) ORDER BY frecuencia ASC")  
 fun getGXFechaFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) ORDER BY etiqueta ASC")  
 fun getGXFechaEtiquetados(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) ORDER BY interes ASC")  
 fun getGXFechaInteres(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) ORDER BY veces ASC")  
 fun getGXFechaVeces(): List<Monto>  
  
 // endregion  
  
 // region GETFAST GASTOS/INGRESOS  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND (etiqueta = 10002 OR etiqueta = 10005)")  
 fun getIFast(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (etiqueta = 10002 OR etiqueta = 10005)")  
 fun getGFast(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta > 10000 AND etiqueta = :e AND (etiqueta = 10002 OR etiqueta = 10005)")  
 fun getIFast(e: Int): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND etiqueta = :e AND (etiqueta = 10002 OR etiqueta = 10005)")  
 fun getGFast(e: Int): List<Monto>  
  
 //endregion  
  
 @Query("SELECT \* FROM monto WHERE (estado = 2 OR estado = 7)")  
 fun getPapelera(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE (estado = 3 OR estado = 4 OR estado = 8 OR estado = 9)")  
 fun getFavoritos(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 8) OR ((estado = 6 OR estado = 9) AND delay >= 1)")  
 fun getPDADeudas(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8)")  
 fun getDeudasList(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY concepto ASC")  
 fun getDeudasAlfabetica(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY valor ASC")  
 fun getDeudasValuadas(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY frecuencia ASC")  
 fun getDeudasFrecuentes(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY etiqueta ASC")  
 fun getDeudasEtiquetadas(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY interes ASC")  
 fun getDeudasInteres(): List<Monto>  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 5 OR estado = 6 OR estado = 8) ORDER BY veces ASC")  
 fun getDeudasVeces(): List<Monto>  
  
 // region  
 @Query("SELECT concepto FROM monto WHERE etiqueta < 10000 ORDER BY valor DESC")  
 fun getTopConceptos(): Array<String>  
 @Query("SELECT valor FROM monto WHERE etiqueta < 10000 ORDER BY valor DESC")  
 fun getTopValor(): Array<Double>  
 //endregion  
  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND (estado = 0 OR estado = 3 OR estado = 5 OR estado = 8) OR delay >= 1 AND enddate >= :din AND adddate <= :din")  
 fun getDelayed(din: Int): List<Monto>  
 @Query("SELECT \* FROM monto WHERE estado != 10")  
 fun getAllMontos(): List<Monto>  
 @Query("SELECT \* FROM monto WHERE etiqueta < 10000 AND estado != 10")  
 fun getAllGastos(): List<Monto>  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.5 LabelsDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface LabelsDao {  
  
 @Insert  
 fun insertLabel(labels: Labels)  
 @Update  
 fun updateLabel(labels: Labels)  
 @Delete  
 fun deleteLabel(labels: Labels)  
  
 @Query("DELETE FROM labels")  
 suspend fun clean()  
  
 @Query("SELECT \* FROM labels WHERE idlabel != 0")  
 fun getAllLabels(): List<Labels>  
  
 @Query("SELECT \* FROM labels WHERE estado != 1")  
 fun getAllLabelsZero(): List<Labels>  
  
 @Query("SELECT idlabel FROM labels")  
 fun getIds(): MutableList<Long>  
 @Query("SELECT plabel FROM labels")  
 fun getPlabels(): MutableList<String>  
 @Query("SELECT color FROM labels")  
 fun getColors(): MutableList<Int>  
  
  
 @Query("SELECT plabel FROM labels")  
 fun getPlabelsArray(): Array<String>  
 @Query("SELECT color FROM labels")  
 fun getColorsArray(): Array<Int>  
  
 @Query("SELECT MAX(idlabel) FROM labels")  
 fun getMaxLabel(): Int  
  
 @Query("SELECT idlabel FROM labels WHERE idlabel = :id")  
 fun getIdLabel(id: Int): Long  
 @Query("SELECT plabel FROM labels WHERE idlabel = :id")  
 fun getPlabel(id: Int): String  
 @Query("SELECT color FROM labels WHERE idlabel = :id")  
 fun getColor(id: Int): Int  
 @Query("SELECT estado FROM labels WHERE idlabel = :id")  
 fun getEstado(id: Int): Int  
  
 @Query("SELECT idlabel FROM labels WHERE plabel = :p")  
 fun getIdL(p: String): Long  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.6 IngresosGastosDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface IngresosGastosDao {  
 @Query("SELECT \* FROM ingresosgastos")  
 fun getIngresosGastos(): List<IngresosGastos>  
  
 @Query("SELECT summaryingresos FROM ingresosgastos")  
 suspend fun checkSummaryI(): Double  
 @Query("SELECT summarygastos FROM ingresosgastos")  
 suspend fun checkSummaryG(): Double  
  
 @Query("DELETE FROM ingresosgastos")  
 suspend fun clean()  
  
 @Insert  
 fun insertIngresosGastos(ingresoGasto: IngresosGastos)  
 @Update  
 fun updateIngresosGastos(ingresoGasto: IngresosGastos)  
 @Delete  
 fun deleteIngresosGastos(ingresoGasto: IngresosGastos)  
  
 @Query("UPDATE ingresosgastos SET summaryingresos = :field WHERE iduser = :id")  
 suspend fun updateSummaryI(id: Int, field: Double?)  
 @Query("UPDATE ingresosgastos SET summarygastos = :field WHERE iduser = :id")  
 suspend fun updateSummaryG(id: Int, field: Double?)  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.7 GruposDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface GruposDao {  
 @Query("SELECT \* FROM grupos")  
 fun getGrupo(): List<Grupos>  
  
 @Query("DELETE FROM grupos")  
 suspend fun clean()  
  
 @Insert  
 fun insertGrupo(grupo: Grupos)  
  
 @Update  
 fun updateGrupo(grupo: Grupos)  
  
 @Delete  
 fun deleteGrupo(grupo: Grupos)  
  
 @Query("SELECT MAX(Id) FROM grupos")  
 fun getMaxGrupo(): Int  
  
 @Query("SELECT Id FROM grupos WHERE Id = :id")  
 fun getIdGrupo(id: Int): Long  
 @Query("SELECT nameg FROM grupos WHERE Id = :id")  
 fun getNameG(id: Int): String  
 @Query("SELECT description FROM grupos WHERE Id = :id")  
 fun getDescription(id: Int): String  
 @Query("SELECT type FROM grupos WHERE Id = :id")  
 fun getType(id: Int): Int  
 @Query("SELECT admin FROM grupos WHERE Id = :id")  
 fun getAdmin(id: Int): Long  
 @Query("SELECT idori FROM grupos WHERE Id = :id")  
 fun getIdori(id: Int): Long  
 @Query("SELECT color FROM grupos WHERE Id = :id")  
 fun getColor(id: Int): Int  
 @Query("SELECT enlace FROM grupos WHERE Id = :id")  
 fun getEnlace(id: Int): Long  
  
 @Query("SELECT \* FROM grupos WHERE type != 2")  
 fun getAllGrupos(): List<Grupos>  
  
 @Query("SELECT \* FROM grupos WHERE Id = :idg")  
 fun getG(idg: Long): Grupos  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.8 EventosDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface EventosDao {  
 @Insert  
 fun insertEvento(eventos: Eventos)  
  
 @Update  
 fun updateEvento(eventos: Eventos)  
  
 @Query("DELETE FROM eventos")  
 fun clean()  
  
 @Query("SELECT MAX(idevento) FROM eventos")  
 fun getMaxEvento(): Int  
 @Query("SELECT \* FROM eventos WHERE estado != 4")  
 fun getAllEventos(): List<Eventos>  
 @Query("SELECT \* FROM eventos WHERE estado == 1 OR estado == 3")  
 fun getAllUnabledEventos(): List<Eventos>  
 @Query("SELECT MAX(adddate) FROM eventos")  
 fun getMaxAddDate(): Int  
  
 @Query("SELECT idevento FROM eventos WHERE idevento = :id")  
 fun getIdevento(id: Int): Long  
 @Query("SELECT nombre FROM eventos WHERE idevento = :id")  
 fun getNombre(id: Int): String  
 @Query("SELECT fecha FROM eventos WHERE idevento = :id")  
 fun getFecha(id: Int): Int  
 @Query("SELECT frecuencia FROM eventos WHERE idevento = :id")  
 fun getFrecuencia(id: Int): Int  
 @Query("SELECT etiqueta FROM eventos WHERE idevento = :id")  
 fun getEtiqueta(id: Int): Int  
 @Query("SELECT estado FROM eventos WHERE idevento = :id")  
 fun getEstado(id: Int): Int  
 @Query("SELECT adddate FROM eventos WHERE idevento = :id")  
 fun getAddDate(id: Int): Int  
  
 @Query("SELECT \* FROM eventos WHERE adddate <= :fecha AND (fecha = :fecha OR fecha = :dom OR fecha = :dommon)")  
 fun getEventosXFecha(fecha: Int, dom: Int, dommon: Int): List<Eventos>  
 @Query("SELECT \* FROM eventos WHERE adddate <= :fecha AND (fecha = :fecha OR fecha = :dom OR fecha = :dommon OR fecha = :futuro OR fecha = :domFuturo OR fecha = :dommonFuturo)")  
 fun getEventosX2Fechas(fecha: Int, dom: Int, dommon: Int, futuro: Int, domFuturo: Int, dommonFuturo: Int): List<Eventos>  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.9 ConySugDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface ConySugDao {  
 @Insert(onConflict = OnConflictStrategy.REPLACE)  
 fun insertCon(conySug: ConySug)  
  
 @Update  
 fun updateCon(conySug: ConySug)  
  
 @Query("DELETE FROM conysug")  
 fun clean()  
  
 @Query("SELECT MAX(idcon) FROM conysug")  
 fun getMaxConsejo(): Int  
  
 @Query("SELECT idcon FROM conysug WHERE idcon = :id")  
 fun getIdcon(id: Int): Long  
 @Query("SELECT nombre FROM conysug WHERE idcon = :id")  
 fun getNombre(id: Int): String  
 @Query("SELECT contenido FROM conysug WHERE idcon = :id")  
 fun getContenido(id: Int): String  
 @Query("SELECT estado FROM conysug WHERE idcon = :id")  
 fun getEstado(id: Int): Int  
 @Query("SELECT flag FROM conysug WHERE idcon = :id")  
 fun getFlag(id: Int): Int  
 @Query("SELECT type FROM conysug WHERE idcon = :id")  
 fun getType(id: Int): Int  
 @Query("SELECT style FROM conysug WHERE idcon = :id")  
 fun getStyle(id: Int): Int  
  
 @Query("SELECT \* FROM conysug")  
 fun getAllCon(): List<ConySug>  
 @Query("SELECT \* FROM conysug WHERE estado = 1")  
 fun getAllActiveCon(): List<ConySug>  
 @Query("SELECT \* FROM conysug WHERE estado = 2")  
 fun getAllAcceptedCon(): List<ConySug>  
 @Query("SELECT \* FROM conysug WHERE estado = 3")  
 fun getAllRejectedCon(): List<ConySug>  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

### 15.10 AssetsDao

Los dao utilizan la anotación ‘@Dao’ para indicar el acceso a una entidad en la bd room, además de contar con métodos de consulta ‘@Query’ junto con las operaciones correspondientes ‘@Insert’, ‘@Update’ y ‘@Delete’

@Dao  
interface AssetsDao {  
 @Insert  
 fun insertAsset(asset: Assets)  
  
 @Query("DELETE FROM assets")  
 fun clean()  
  
 @Query("SELECT theme FROM assets")  
 fun getTheme(): Int  
 @Query("UPDATE assets SET theme = :mode")  
 fun updateTheme(mode: Long)  
  
 @Query("UPDATE assets SET lastprocess = :today")  
 fun updateLastprocess(today: Int)  
  
 @Query("SELECT lastprocess FROM assets")  
 fun getLastProcess(): Int  
  
 @Query("SELECT notificaciones FROM assets")  
 fun getNotif(): Int  
 @Query("UPDATE assets SET notificaciones = :mode")  
 fun updateNotif(mode: Long)  
}

En resumen, los dao permiten comunicar una base de datos room con una aplicación permitiendo efectuar operaciones CRUD

**16 Base de Datos**

# CAPÍTULO IX. RESULTADOS

## 9.1 ANÁLISIS DE LOS RESULTADOS

Concluido el desarrollo del proyecto se encontraron diversas conclusiones y puntos a seguir respecto al seguimiento. Un aspecto que generó mucho interés fue los cambios que se hicieron para adaptar la aplicación a las tecnologías con las cuales se trabajaron, así como la interacción de los usuarios con la misma, además que se descubrieron múltiples aspectos a mejorar y desarrollar a futuro para que la aplicación tenga un alcance mucho mayor.

Hablando acerca del futuro del desarrollo del proyecto, se apunta a mejorar todo lo que sea posible la experiencia del usuario, de manera que el producto sea una interacción satisfactoria y sencilla, además de seguir mejorando los procesos de ahorro para ofrecer cada vez mejores consejos así como lanzar la aplicación al mercado y en base a resultados que se obtengan, decidir si seguir mejorando como aplicación local o apuntar a una aplicación con un enfoque más a las finanzas generales y no tanto las finanzas personales.

## 9.2 RESULTADO DLAS PRUEBAS O VALIDACIONES

El proyecto visto en contra de los requerimientos funcionales y no funcionales tuvo un resultado satisfactorio mas no perfecto, debido a que hubo múltiples cambios haciendo que varios requerimientos fuesen transformados de por medio, sin contar que algunos de ellos no pudieron ser llevados a cabo, todos estos serán detallados a continuación.

-Si ya existe un usuario activo la autenticación será una contraseña: No se tuvo como tal una contraseña cada que se quisiera ingresar mientras se tuviese un usuario activo.

- La pantalla principal tiene dos opciones para hacer login: Realmente solo existe una opción para hacer login ya que no se ofrecen opciones como tipos de usuario o login por terceros como Google, Facebook, etc.

-Dentro de los requerimientos no funcionales se tuvieron muchos requerimientos en rojo debido a arreglos que se efectuaron para tener una mejor interfaz de usuario tanto en colores, tamaños de letras, íconos, formas, fuentes y tonalidades.

-Gran parte de la problemática con los colores se tuvo debido a que se consideró que los colores no tenían un balance o armonía por lo que se optó por usar tonalidades de blanco y negro para prácticamente todo el proyecto por lo que los requerimientos no funcionales que definían colores y sombreados no pudieron ser realizados en casi su totalidad.

## 9.3 CONCLUSIONES

Héctor: Hubo procesos complicados, fue un proyecto lleno de aprendizaje, y se nota en la comparación de la etapa de la planeación con el producto final, porque es muy diferente, los procesos para alcanzar el resultado deseado no fueron los que se habían previsto, por eso pienso que la fase de desarrollo estuvo llena de gratificaciones, pues tuvimos mucho por aprender, y lo logramos. Fue complejo, incluso ahora hay algunos detalles que se pueden mejorar, quisiera poder hacerlo una vez entre a la universidad, y me entusiasma el poder gozar de un privilegio como lo es poder realizar este tipo de proyecto, hecho de sangre, sudor y lágrimas, y que al final se pudo ensamblar una idea relativamente improvisada, se pudo hacer bien y sin muchos errores. No fue tan intuitiva como me hubiera gustado, perdimos la noción de diseño entre tantas funciones requeridas para el sistema, y aunque lo arreglamos y maquillamos bastante bien, no pudo ser tan intuitiva como era la meta, quizá es de las primeras cosas que mejoraría en cuanto le diera su primera actualización.

Alex: En lo personal puedo concluir que el desarrollo del proyecto es una experiencia que aun que en un principio tiene muchas dudas e inquietudes, parece que muchas veces siguen mientras este se desarrolla, en definitiva creo que no es una materia para ser cursada dentro de 2 semestres debido a la complejidad del desafío como tal, sin contar la cantidad de procesos que lleva de por medio, haciendo que sea una materia realmente estresante, pero fuera de ello es una experiencia muy enriquecedora en muchos aspectos, empezando por el manejo de actividades y tiempos, planificación y en el caso del desarrollo de software el planificar los algoritmos a utilizar ya que muchas veces se tenía una idea en la propuesta o “al aire” la cual a la hora de ser ejecutada resultó ser muy diferente afectando de por medio otros aspectos del proyecto por lo que aun que el resultado es gratificante y se cumplió el objetivo, hubo mucha confusión de por medio. Hablando respecto al proyecto y sus resultados, se puede observar que la gente realmente le entusiasma la idea de poder llevar un control más detallado de su economía así como el poder visualizarla de manera más dinámica y no tanto como un estado de cuenta ya que esto lo hace un poco más entendible para muchas personas; además que como desarrollador es importante tener este tipo de interacciones con los usuarios para tener un feedback efectivo acerca del uso de la aplicación para en casos futuros evitar cualquier malentendido y en general siempre apuntar hacia arriba y seguir mejorando.

## 9.4 REFERENCIAS

-CONDUSEF. (2023). 31 de octubre Día Mundial del Ahorro: cómo ahorramos las y los mexicanos?. Gobierno de México. <https://www.condusef.gob.mx/?p=contenido&idc=1568&idcat=1>

- Redacción. (2016). ¿Cuántas horas se conectan los mexicanos a sus smartphones?. El Financiero. <https://www.elfinanciero.com.mx/empresas/cuantas-horas-se-conectan-los-mexicanos-a-sus-smartphones/>

- Wikipedia. (2023). Dispositivo móvil. Wikipedia. <https://es.wikipedia.org/wiki/Dispositivo_móvil>

- INEGI. (2023). EL INEGI PRESENTA LOS RESULTADOS DE LA ENCUESTA NACIONAL DE INGRESOS Y GASTOS DE LOS HOGARES (ENIGH) 2022. INEGI. <https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2023/ENIGH2022.pdf>

- Kim Porter. (2022). Herramientas digitales financieras. AARP. <https://www.aarp.org/espanol/dinero/presupuesto-y-ahorro/info-2022/herramientas-digitales-financieras.html>

- Yanina Muradas. (2021). Kotlin vs Java. Open Webinars. <https://openwebinars.net/blog/kotlin-vs-java/>

- Pamela López. (2022). Las herramientas de ahorro digital pueden fortalecer la salud financiera. BBVA. <https://www.bbva.com/es/pe/salud-financiera/las-herramientas-de-ahorro-digital-pueden-fortalecer-la-salud-financiera/>

- Communications. (2023). Estas son las nueve 'apps' de finanzas que ayudan a ahorrar. BBVA. <https://www.bbva.com/es/salud-financiera/estas-son-las-nueve-apps-de-finanzas-que-ayudan-a-ahorrar/>

- Finerio Connect. (2023). México necesita mejores herramientas de finanzas personales. Finerio Connect. <https://blog.finerioconnect.com/mexico-necesita-mejores-herramientas-de-finanzas-personales/>

- JetBrains. (2020). Kotlin. JetBrains. [https://www.jetbrains.com/es-es/lp/devecosystem-2020/kotlin/#:~:text=Kotlin%20es%20tremendamente%20versátil.,que%20funcionan%20en%20varias%20plataformas](https://www.jetbrains.com/es-es/lp/devecosystem-2020/kotlin/" \l ":~:text=Kotlin%20es%20tremendamente%20versátil.,que%20funcionan%20en%20varias%20plataformas).

- Octavio Islas. (2023). Resultados del estudio Digital 2023 México. El Universal. <https://www.eluniversal.com.mx/opinion/octavio-islas/resultados-del-estudio-digital-2023-mexico-1/#:~:text=El%20total%20de%20conexiones%20de,creció%203.7%25%20(4%2C400%2C000)>.

- Saber más Ser más. (2022). Siete aplicaciones para tener en orden sus finanzas. Saber más Ser más. <https://www.sabermassermas.com/siete-aplicaciones-para-tener-en-orden-sus-finanzas/>

- KeepCoding. (2023). Ventajas y desventajas de Kotlin. KeepCoding. <https://keepcoding.io/blog/ventajas-y-desventajas-de-kotlin/>

## 9.5 ANEXO

### Manual de usuario

1. Introducción

1.1. Sobre Savetrack Gran parte de la población no lleva una vida financiera estable debido a la poca o nula cultura económica, por lo que es usual el tener problemas económicos o excederse en gastos al momento de comprar incluso lo más esencial.

Misión: Satisfacer los conocimientos y necesidad de herramientas del usuario para la correcta organización de sus finanzas, personales o familiares, para así evitar deudas y problemas económicos, así como mejorar su calidad de vida en el menor tiempo posible, todo de manera intuitiva y amigable para que cualquiera, desde jóvenes hasta personas de la tercera edad, puedan utilizarla. Brindar la mejor educación financiera a nuestros usuarios y darles los mejores planes financieros para sus distintas necesidades.

Visión: Desarrollar una herramienta para solventar una necesidad en todas las poblaciones de todos los estratos sociales, intentando mejorar la calidad de vida de los usuarios. Ser apoyo educativo en escuelas y para bancos o entidades financieras al momento de educar a los clientes acerca de sus finanzas. Ser la aplicación #1 de finanzas en el país y eventualmente en el mundo.

2. Login y Registro

Imagen 40, login y registro

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente2.1. Cómo registrarse Nos localizamos en esta vista al iniciar la app por primera vez, y para crear una cuenta deberemos presionar “No tengo cuenta”.

Al presionar dicho botón nos redirigirá a esta vista, la cual deberemos llenar con nuestros datos para crear una cuenta.

2.2. Cómo iniciar sesión En la primera vista tendremos que introducir nuestros datos de usuario y contraseña en los campos correspondientes.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

2.3. Cómo reestablecer mi contraseña En la primera vista, tendremos que presionar “Olvidé mi contraseña”.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Después de presionar el botón nos encontraremos en esta vista, en la cual tendremos que introducir nuestro usuario, y nos llegará un correo con un pin.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Dicho pin deberemos introducirlo en esta vista, junto con una nueva contraseña y su confirmación. Si todo es correcto, podrás reestablecer tu contraseña con éxito.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

3. Inicio

3.1. Montos Cómo crear un monto En la vista de inicio pueden apreciarse dos botones que destacan por sus colores verde y rojo, el verde redirige a la creación de un ingreso, mientras que el rojo a la de un gasto. Mientras que los ingresos representan un valor positivo al balance del usuario, los gastos son su contraparte negativa. Ambos son tipos de monto, y, por lo tanto, comparten características y propiedades.

Imagen que contiene Aplicación

Descripción generada automáticamente

Propiedades de los montos

▪ Concepto: Es el nombre con el que se identifica el monto.

▪ Valor: Es la cantidad que representa el monto.

▪ Fecha: Es la fecha en la que se registra un ingreso o en la que se debe pagar determinado gasto. Puede ser una fecha única, así como un día de un mes o día de la semana. Nota: Si un monto está programado mensualmente para el 31 de cada mes, tomará el día más cercano posible, en el caso de, por ejemplo, febrero, será el 28, o 29 si es bisiesto)

Texto

Descripción generada automáticamente con confianza baja▪ Frecuencia: Es la repetición que se le da a un monto, por ejemplo, en caso de que un monto tenga como fecha un lunes, pero la frecuencia está configurada para quincenal, entonces será el lunes de cada dos semanas.

Las frecuencias posibles son:

• Única vez

• Diario

• Semanal

• Quincenal

• Mensual

• Bimestral

• Trimestral

• Cuatrimestral

• Semestral

• Anual

▪ Fecha de creación: Marca la fecha en la que el monto fue creado

▪ Etiqueta: En el caso de los ingresos, las etiquetas son fijas, y son las siguientes:

• Salario

• Venta

• Beca

• Pensión

• Manutención

• Ingreso Pasivo

• Regalo

Sin embargo, las etiquetas de los gastos son personalizables, las que vienen por defecto pueden modificarse en nombre y color:

• Alimentos

• Hogar

• Bienestar

• Otras Necesidades

• Gastos Hormiga

• Ocio y demás

• Obsequio

▪ Estado: El estado de un estado puede variar, sobre todo en el caso de los gastos, pues determina si se ha pagado o no, o si es deuda.

▪ Veces: Indica las veces en las que un monto se ha añadido.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

3.2. Partes de la vista

Diagrama

Descripción generada automáticamenteGráfica de pai Grafica de manera precisa los gastos o ingresos realizados en un periodo de tiempo personalizado, los datos se muestran por etiqueta y con porcentaje del total, el rango de días puede ajustarse utilizando una barra, y se puede cambiar entre gastos e ingresos por medio de un switch. Se puede seleccionar una categoría simplemente presionando un área de la gráfica en la que se represente alguna etiqueta.

▪ Partes de la gráfica: Mientras que a su alrededor se identifican por colores cada gasto o ingreso, en su interior tiene los ingresos netos como resultado de la resta entre los ingresos brutos y los egresos

▪ Switch ingresos/gastos

▪ Barra de rango

Indicador de porcentaje Indica el porcentaje que ocupa determinada etiqueta en el total de ingresos o gastos a lo largo del periodo personalizable. Si se presiona, redirigirá a una lista de los montos pertenecientes a la categoría seleccionada.

Diagrama

Descripción generada automáticamente

Fecha actual Existe un pequeño módulo que indica la fecha actual.

Diagrama

Descripción generada automáticamente

Barra lateral de montos por pagar En la barra lateral se muestra, además de las subvistas, una lista con los montos cuya fecha de pago se acerca.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Subvistas de montos

▪ Mandados: Muestra los montos a pagar el día de hoy. Cada monto puede marcarse como pagado si se presiona en el área grande de color gris oscuro. Puede posponerse si se presiona el botón amarillo y omitirse si se presiona el botón rojo.

Patrón de fondo

Descripción generada automáticamente con confianza media

▪ Por pagar: Muestra los montos a pagar a lo largo de la semana, mas no los diarios. Cada monto puede marcarse como pagado si se presiona en el área grande de color gris oscuro. Puede posponerse si se presiona el botón amarillo y omitirse si se presiona el botón rojo.

Texto, Aplicación

Descripción generada automáticamente con confianza media

▪ Lista de egresos: Muestra una lista generalizada de todos los gastos. Se puede crear un gasto directamente desde esta vista.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

▪ Lista de ingresos: Muestra una lista generalizada de todos los ingresos. También se puede crear un ingreso directamente desde esta vista. Los montos de tipo venta se muestran aquí, y se pueden agregar simplemente presionándolos.

▪ Opciones de los montos

• Favorito: Marca un monto como favorito.

• Ver detalles: Redirige a una vista que contiene toda la información de un monto de manera legible

• Editar: Redirige a una vista de actualizar monto

• Eliminar: Envía el monto a la papelera

• Liquidar (para deudas): En vez de hacer el pago de una deuda con su valor, utiliza valor final y elimina el monto.

Imagen que contiene Tabla

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente

Vista de ver detalles, se accede a esta cuando se presiona la opción de ver detalles.

4. Perfil

4.1. Partes de la vista Indicador de balance Muestra la cantidad de dinero que tiene el usuario.

Diagrama

Descripción generada automáticamente

Indicador de días ahorrando Muestra la cantidad de días que el usuario lleva ahorrando. Esto se mide a través de:

▪ El cumplimiento diario de su meta de ahorro

▪ Que no tenga gastos que excedan sus ingresos brutos

Interfaz de usuario gráfica, Diagrama, Aplicación

Descripción generada automáticamente

Datos del usuario

▪ Nombre ▪ Edad

▪ Tipos de ingreso, de entre los cuales los disponibles son:

• Salario • Venta

• Pensión

• Beca

• Manutención

• Ingreso Pasivo

Diagrama

Descripción generada automáticamente

Grupos Se muestra una vista previa de los grupos del usuario, muestra un máximo de 3.

Imagen que contiene Diagrama

Descripción generada automáticamente

4.2. Subvistas

Edición de perfil

▪ Nombre: Es fijo y no se puede editar.

▪ Edad: El usuario ingresa su edad.

▪ Meta de ahorro: El usuario decide si quiere tener una meta de ahorro y qué tan grande es. ▪ Tipo de ingreso: Se personaliza según los tipos de ingreso del usuario

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Grupos

▪ Creación de grupo

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente▪ Características de los grupos

• Administrador o Editar o Eliminar

• Tipo

• Color

▪ Añadir monto a grupo

▪ Salir de grupo

Respectivamente de izquierda a derecha: Muestra la creación de un monto de grupo, la vista básica del grupo con sus montos y de los miembros del grupo.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Configuración de la app

▪ Modo claro/oscuro: presionando el switch, se puede alternar entre modo claro y oscuro

▪ Activar/desactivar notificaciones: presionando el switch se pueden activar o desactivar las notificaciones.

▪ Acceso al manual: si se presiona el manual se descargará este mismo manual.

▪ Cierre de sesión: si se tiene acceso a internet, se respaldarán todos los datos y se cerrará sesión.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

5. Historial

5.1. Calendario Navegar por el calendario

Se puede deslizar por el calendario de izquierda a derecha o girar arriba y abajo la fecha escrita. Acceder a un día específico Se presiona en el botón azul con la flecha Filtro de etiquetas Se pueden filtrar las etiquetas para acceder a una lista con los montos de dicha etiqueta en un día específico presionando el botón con forma de filtro

Calendario

Descripción generada automáticamente

5.2. Subvistas Etiquetas Al momento de crear, consultar y editar etiquetas, se tienen en cuenta dos propiedades: nombre y color.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Favoritos

Se muestra una lista con los montos marcados como favoritos

Papelera

Se muestra una lista con los montos eliminados y la opción de eliminarlos definitivamente o de recuperarlos.

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente con confianza media

6. Planes de ahorro

6.1. Partes de la vista Indicador del valor del dólar Hay una pequeña caja que muestra el valor de venta del dólar del día actual.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Indicador de días ahorrando, Así como su equivalente en la vista de perfil, muestra la cantidad de días que el usuario lleva ahorrando.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Indicador de balance, Así como su equivalente en la vista de perfil, muestra la cantidad de dinero que tiene ahorrado el usuario

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Barra inferior de deudas Muestra las deudas que tiene el usuario, con las opciones de marcar como favorito, editar, posponer y omitir.

Imagen que contiene Icono

Descripción generada automáticamente

6.2. Subvista de deudas Muestra una lista con todas las deudas que tiene el usuario, muestra su concepto, valor, interés, valor y valor final, fecha próxima y fecha final, y la etiqueta.

Tabla

Descripción generada automáticamente con confianza media

7. Finanzas

7.1. Partes de la vista Consejos y sugerencias Hay consejos únicos, repetibles y múltiples, y se adaptan a las necesidades y preferencias del usuario. Si se rechaza, no se volverá a aconsejar, pero si se acepta redirigirá a la vista para realizar la acción recomendada.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Economía

Gráfica de ahorro estático: Toma los datos de ahorro de los últimos 6 días y los grafica de tal modo que se pueda observar el crecimiento del balance.

Gráfica de inversión: Toma los datos del valor de venta del dólar de los últimos días para observar de mejor manera cómo fluctúa.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Gráfica personalizada: La misma gráfica de inversión, pero con un valor modificable para observar mejor el crecimiento.

Gráfica de moneda: Compara el valor del dólar, del euro y del dólar canadiense.

Interfaz de usuario gráfica

Descripción generada automáticamente

Producto recomendado A partir de los montos recientes del usuario, se determinan cosas que podrían interesarle al usuario, y se calcula la afinidad (qué tanto podría interesarle) y el ahorro (qué tanto estaría ahorrando. Mientras más montos haya, mejores serán las recomendaciones.

Interfaz de usuario gráfica

Descripción generada automáticamente

Hay 3 botones, de izquierda a derecha:

• Ver más: redirige a la compra.

• Ver menos: cambia el producto.

• No ver: no vuelve a mostrar el producto.

Interfaz de usuario gráfica

Descripción generada automáticamente

Eventos

Calendario Igual al calendario del historial, pero con eventos, se desliza y muestra los eventos según su día

Interfaz de usuario gráfica

Descripción generada automáticamente

Propiedades de los eventos

o Fecha

o Frecuencia

o Etiqueta

o Estado

o Fecha de creación

Lista de eventos Muestra todos los eventos, con su fecha y su etiqueta, tienen las opciones de editar y eliminar.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

8. Otros

8.1. Notificaciones Si se tienen activadas las notificaciones, llegarán de tres tipos

• Montos: se avisará de montos cuya fecha de pago sea el día actual

• Eventos: se avisará de eventos cuya fecha sea el día actual

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteImagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media• El valor del dólar y su fluctuación.

Activar/desactivar

En la vista de perfil, al acceder a configuración

pueden activarse y desactivarse las notificaciones.

Botones En el caso de los eventos y montos, pueden posponerse u omitirse con sus respectivos botones.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

8.2. Widgets

Desde fuera de la aplicación, es posible crear widgets de la app. Es importante destacar que para inicializar un widget se tiene que presionar una vez sobre él para que muestre los datos.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Ingreso Se selecciona un ingreso de tipo venta, y al presionar el widget se agrega a los ingresos totales.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Gasto Se selecciona cualquier gasto, y al presionar el widget se agrega a los gastos totales.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Redirección Se crea un acceso directo a la subvista de crear monto.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## XML

### Login

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".Login"  
 android:background="@drawable/gradient\_background">  
  
 <ProgressBar  
 android:id="@+id/perame"  
 android:alpha="0"  
 android:layout\_width="240dp"  
 android:layout\_height="240dp"  
 android:translationZ="700dp"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:indeterminateTint="@color/B2" />  
  
 <TextView  
 android:paddingTop="10dp"  
 android:id="@+id/textview\_IniSes"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-black"  
 android:text="@string/TextLogin"  
 android:textColor="@color/X5"  
 android:textAlignment="center"  
 android:textSize="36sp"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="0dp"  
 tools:layout\_editor\_absoluteY="44dp" />  
  
 <Button  
 android:id="@+id/buttonIniSes"  
 android:layout\_width="130dp"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="sans-serif-black"  
 android:textColor="@color/X1"  
 android:text="@string/IniSes"  
 android:background="@drawable/p1backbox"  
 app:layout\_constraintBottom\_toTopOf="@+id/buttonSinCuenta"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editTextTextPassword"  
 app:layout\_constraintVertical\_bias="0.715" />  
  
 <EditText  
 android:id="@+id/editTextTextPassword"  
 android:background="@drawable/p1backbox"  
 android:padding="10sp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:autofillHints="@string/PASSWORD"  
 android:ems="10"  
 android:hint="@string/PASSWORD"  
 android:inputType="textPassword"  
 android:textColorHint="@color/X0"  
 android:textColor="@color/X0"  
 app:layout\_constraintBottom\_toTopOf="@+id/buttonSinCuenta"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.428"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textview\_IniSes"  
 tools:ignore="MissingConstraints" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:autofillHints="@string/EMAIL"  
 android:background="@drawable/p1backbox"  
 android:ems="10"  
 android:hint="@string/EMAIL"  
 android:inputType="textPersonName"  
 android:padding="10sp"  
 android:textColor="@color/X0"  
 android:textColorHint="@color/X0"  
 app:layout\_constraintBottom\_toTopOf="@+id/editTextTextPassword"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textview\_IniSes"  
 app:layout\_constraintVertical\_bias="0.806" />  
  
 <ImageButton  
 android:id="@+id/UserCache"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/transparent"  
 android:contentDescription="@string/usuarios"  
 app:layout\_constraintEnd\_toEndOf="@id/editTextTextPersonName"  
 app:layout\_constraintTop\_toTopOf="@id/editTextTextPersonName"  
 app:layout\_constraintBottom\_toBottomOf="@id/editTextTextPersonName"  
 app:srcCompat="@drawable/ic\_person"  
 app:tint="@color/X0" />  
  
 <Spinner  
 android:id="@+id/SpinnerUsers"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:contentDescription="@string/usuarios"  
 android:alpha="0"  
 app:layout\_constraintStart\_toStartOf="@id/UserCache"  
 app:layout\_constraintEnd\_toEndOf="@id/UserCache"  
 app:layout\_constraintTop\_toTopOf="@id/UserCache"  
 app:layout\_constraintBottom\_toBottomOf="@id/UserCache" />  
  
 <Button  
 android:id="@+id/buttonSinCuenta"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="@android:color/transparent"  
 android:fontFamily="sans-serif-black"  
 android:hint="@string/SinCuenta"  
 android:textColorHint="@color/X5"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.101"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/textview\_IniSes"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <Button  
 android:id="@+id/buttonOlviContra"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="@android:color/transparent"  
 android:fontFamily="sans-serif-black"  
 android:hint="@string/OlviContra"  
 android:textColorHint="@color/X5"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.93"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/textview\_IniSes"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="164dp"  
 android:layout\_height="159dp"  
 android:contentDescription="@string/TextLogin"  
 app:layout\_constraintBottom\_toTopOf="@+id/editTextTextPassword"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/textview\_IniSes"  
 app:srcCompat="@drawable/logo1" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### Index

<?xml version="1.0" encoding="utf-8"?>  
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:id="@+id/background"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:contentDescription="@string/wallpaper"  
 android:theme="@style/index\_background" />  
 <!--  
 style="?android:attr/progressBarStyle"  
 android:background="@drawable/ring"  
 android:indeterminateDrawable="@drawable/progresscolor"  
 -->  
 <ProgressBar  
 android:id="@+id/GraficoPastel"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:translationZ="10dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/searchforlabel"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_marginStart="70dp"  
 android:layout\_marginEnd="70dp"  
 android:layout\_marginBottom="15dp"  
 android:background="@drawable/p1table"  
 android:hint="@string/gastos"  
 android:paddingTop="7dp"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="34sp"  
 android:translationZ="200dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/PieChart"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <com.github.mikephil.charting.charts.PieChart  
 android:id="@+id/PieChart"  
 android:layout\_width="250dp"  
 android:layout\_height="250dp"  
 android:contentDescription=""  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/redirect"  
 android:layout\_width="120dp"  
 android:layout\_height="120dp"  
 android:translationZ="200dp"  
 app:layout\_constraintTop\_toTopOf="@id/PieChart"  
 app:layout\_constraintBottom\_toBottomOf="@id/PieChart"  
 app:layout\_constraintStart\_toStartOf="@id/PieChart"  
 app:layout\_constraintEnd\_toEndOf="@id/PieChart" />  
  
 <RelativeLayout  
 android:id="@+id/switchpordelante"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="15dp"  
 android:translationZ="100dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/PieChart">  
  
 <com.polyak.iconswitch.IconSwitch  
 android:id="@+id/SultanOfSwing"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:layout\_marginStart="145dp"  
 android:trackTint="@color/X2"  
 app:isw\_icon\_left="@drawable/ic\_add"  
 app:isw\_icon\_right="@drawable/ic\_remove"  
 app:isw\_inactive\_tint\_icon\_left="@color/G5"  
 app:isw\_inactive\_tint\_icon\_right="@color/R4"  
 app:isw\_thumb\_color\_left="@color/G2"  
 app:isw\_thumb\_color\_right="@color/R0" />  
  
 <EditText  
 android:id="@+id/Rango"  
 android:layout\_width="64dp"  
 android:layout\_height="32dp"  
 android:layout\_marginStart="150dp"  
 android:layout\_marginTop="60dp"  
 android:autofillHints=""  
 android:background="@drawable/p1table"  
 android:hint="@string/rango"  
 android:inputType="number"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="20sp"  
 android:translationZ="200dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/SultanOfSwing" />  
  
 <TextView  
 android:id="@+id/RangoTV"  
 android:layout\_width="64dp"  
 android:layout\_height="32dp"  
 android:layout\_marginStart="80dp"  
 android:layout\_marginTop="60dp"  
 android:autofillHints=""  
 android:hint="@string/rangotv"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="20sp"  
 app:layout\_constraintEnd\_toStartOf="@id/Rango"  
 app:layout\_constraintTop\_toBottomOf="@id/SultanOfSwing" />  
  
 <SeekBar  
 android:id="@+id/RangoSeekbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_marginStart="85dp"  
 android:layout\_marginTop="90dp"  
 android:layout\_marginEnd="85dp"  
 android:hint="@string/rango"  
 android:paddingTop="7dp"  
 android:progressTint="?attr/colorPrimaryVariant"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="20sp"  
 android:thumbTint="?attr/colorSecondary"  
 android:translationZ="200dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/Rango" />  
 </RelativeLayout>  
  
 <ImageButton  
 android:id="@+id/AgregarIngresoButton"  
 android:layout\_width="64dp"  
 android:layout\_height="64dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="96dp"  
 android:background="@drawable/confshape"  
 android:contentDescription="@string/nuevo\_ingreso"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:srcCompat="@drawable/ic\_add" />  
  
 <ImageButton  
 android:id="@+id/AgregarGastoButton"  
 android:layout\_width="64dp"  
 android:layout\_height="64dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="16dp"  
 android:background="@drawable/cancshape"  
 android:contentDescription="@string/nuevo\_gasto"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:srcCompat="@drawable/ic\_remove" />  
 <!--  
 <Switch  
 android:id="@+id/SwitchGrupos"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="182dp"  
 android:layout\_marginTop="482dp"  
 android:layout\_marginEnd="183dp"  
 android:layout\_marginBottom="222dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
 -->  
  
 <TextView  
 android:id="@+id/Calendario"  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:background="@drawable/ic\_calendar"  
 android:contentDescription="@string/dias\_ahorrando"  
 android:paddingLeft="12dp"  
 android:paddingTop="26dp"  
 android:paddingRight="12dp"  
 android:paddingBottom="12dp"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="12sp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
  
 <ProgressBar  
 android:id="@+id/MedidorDeAhorroButton"  
 style="?android:attr/progressBarStyleHorizontal"  
 android:layout\_width="128dp"  
 android:layout\_height="128dp"  
 android:layout\_centerInParent="true"  
 android:contentDescription="@string/medidor\_de\_ahorro"  
 android:progressBackgroundTint="@color/R0"  
 android:max="100"  
 android:progress="0"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 app:layout\_constraintBottom\_toTopOf="@id/AgregarIngresoButton"  
 app:layout\_constraintEnd\_toEndOf="@id/AgregarIngresoButton"  
 app:layout\_constraintStart\_toStartOf="@id/AgregarIngresoButton"  
 app:layout\_constraintTop\_toBottomOf="@id/Options" />  
  
 <TextView  
 android:id="@+id/Medidor"  
 android:layout\_width="45dp"  
 android:layout\_height="45dp"  
 android:background="@color/transparent"  
 android:contentDescription="@string/medidor\_de\_ahorro"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:translationZ="100dp"  
 app:layout\_constraintBottom\_toBottomOf="@id/MedidorDeAhorroButton"  
 app:layout\_constraintEnd\_toEndOf="@id/MedidorDeAhorroButton"  
 app:layout\_constraintStart\_toStartOf="@id/MedidorDeAhorroButton"  
 app:layout\_constraintTop\_toTopOf="@id/MedidorDeAhorroButton" />  
  
 <ImageButton  
 android:id="@+id/Options"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:backgroundTint="@color/transparent"  
 android:contentDescription="@string/opciones"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_list" />  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/barrita"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 app:headerLayout="@layout/header\_index"  
 app:menu="@menu/menu\_index">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toTopOf="@id/LayoutPendientes"  
 android:textSize="18sp"  
 android:textColor="?attr/colorPrimary"  
 android:text="@string/porpagar"/>  
 <ScrollView  
 android:id="@+id/LayoutPendientes"  
 android:layout\_width="match\_parent"  
 android:layout\_height="180dp"  
 android:orientation="vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent">  
  
 <LinearLayout  
 android:id="@+id/ProductoPorPagar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/displayCheck"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager" />  
  
 </LinearLayout>  
 </ScrollView>  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </com.google.android.material.navigation.NavigationView>  
  
</androidx.drawerlayout.widget.DrawerLayout>

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

### Perfil

<?xml version="1.0" encoding="utf-8"?>  
<androidx.drawerlayout.widget.DrawerLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:id="@+id/background"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_margin="0dp"  
 android:scaleType="centerCrop"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:theme="@style/perfil\_background"/>  
  
 <androidx.drawerlayout.widget.DrawerLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <ImageView  
 android:id="@+id/ProfilePicture"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginTop="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="@dimen/layout\_margin\_default"  
 android:translationZ="100dp"  
 app:layout\_constraintDimensionRatio="1:1"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@id/UserDataLayout"  
 android:background="@drawable/picshape"  
 android:contentDescription="@string/FotoPerfil"  
 app:srcCompat="@drawable/ic\_person" />  
  
 <ImageButton  
 android:id="@+id/Options"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="@dimen/layout\_margin\_default"  
 android:layout\_marginEnd="@dimen/layout\_margin\_default"  
 android:backgroundTint="@color/transparent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_list"  
 android:contentDescription="@string/opciones"/>  
  
  
 <Button  
 android:id="@+id/BalanceTV"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:backgroundTint="?attr/colorOnPrimary"  
 android:background="@drawable/ic\_circle"  
 android:translationZ="100dp"  
 app:layout\_constraintDimensionRatio="1:1"  
 android:textColor="?attr/colorPrimary"  
 android:contentDescription="@string/dias\_ahorrando"  
 app:layout\_constraintTop\_toBottomOf="@id/UserDataLayout"  
 app:layout\_constraintBottom\_toTopOf="@id/grupos"  
 app:layout\_constraintStart\_toEndOf="@id/grupos"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="-64dp"  
 android:layout\_marginEnd="@dimen/layout\_margin\_default"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="-64dp"  
 android:outlineAmbientShadowColor="@color/X2"  
 android:outlineSpotShadowColor="@color/X2" />  
  
 <LinearLayout  
 android:id="@+id/UserDataLayout"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginTop="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="@dimen/layout\_margin\_default"  
 app:layout\_constraintBottom\_toTopOf="@+id/grupos"  
 app:layout\_constraintEnd\_toStartOf="@id/DaysSavingButton"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/ProfilePicture"  
 app:layout\_constraintWidth\_percent="0.55"  
 android:elevation="15dp">  
  
 <TextView  
 android:id="@+id/UsernameTV"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:autofillHints="Username"  
 android:background="@drawable/b1topcell"  
 android:hint="@string/username"  
 android:paddingStart="10dp"  
 android:paddingTop="12dp"  
 android:paddingEnd="6dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/AgeTV"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:autofillHints="Age"  
 android:background="@drawable/b1midcell"  
 android:hint="@string/Edad"  
 android:paddingStart="10dp"  
 android:paddingTop="12dp"  
 android:paddingEnd="6dp"  
 android:layout\_marginTop="-2dp"  
 android:layout\_marginBottom="-2dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/OcupationTV"  
 android:layout\_width="match\_parent"  
 android:layout\_height="100dp"  
 android:background="@drawable/b1bottomcell"  
 android:hint="@string/TipoChamba"  
 android:paddingStart="10dp"  
 android:paddingTop="12dp"  
 android:paddingEnd="6dp"  
 android:autofillHints="Ocupation"  
 android:textSize="16sp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary" />  
 </LinearLayout>  
  
 <TableLayout  
 android:id="@+id/grupos"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:orientation="vertical"  
 app:layout\_constraintDimensionRatio="1:1"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@id/UserDataLayout"  
 app:layout\_constraintTop\_toBottomOf="@id/UserDataLayout"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="@dimen/layout\_margin\_default"  
 app:layout\_constraintWidth\_percent="0.55"  
 android:stretchColumns="0,1,2">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/displayGrupos"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
 app:layout\_constraintStart\_toStartOf="@+id/grupos"  
 app:layout\_constraintTop\_toBottomOf="@+id/grupos" />  
 </TableLayout>  
  
 <Button  
 android:id="@+id/DaysSavingButton"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:background="@drawable/b1table"  
 android:contentDescription="@string/balance"  
 android:textSize="12sp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textColorHint="?attr/colorSecondary"  
 android:textAlignment="textEnd"  
 android:paddingEnd="8dp"  
 android:paddingStart="40dp"  
 app:layout\_constraintTop\_toBottomOf="@id/ProfilePicture"  
 app:layout\_constraintBottom\_toTopOf="@id/UserDataLayout"  
 app:layout\_constraintStart\_toEndOf="@id/UserDataLayout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="-32dp"  
 android:layout\_marginEnd="@dimen/layout\_margin\_default"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="-32dp"/>  
  
 <ImageButton  
 android:id="@+id/ahorrando"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/transparent"  
 android:layout\_marginStart="-12dp"  
 app:layout\_constraintStart\_toStartOf="@id/DaysSavingButton"  
 app:layout\_constraintTop\_toTopOf="@id/DaysSavingButton"  
 app:layout\_constraintBottom\_toBottomOf="@id/DaysSavingButton"  
 app:srcCompat="@drawable/ic\_noahorrando"  
 android:contentDescription="@string/configuracion"/>  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/barrita"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 app:headerLayout="@layout/header\_perfil"  
 app:menu="@menu/menu\_perfil" />  
  
</androidx.drawerlayout.widget.DrawerLayout>

Interfaz de usuario gráfica

Descripción generada automáticamente

### Historial

<?xml version="1.0" encoding="utf-8"?>  
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:id="@+id/background"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:contentDescription="@string/wallpaper"  
 android:theme="@style/historial\_background" />  
  
 <TextView  
 android:id="@+id/bar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:text="@string/historial"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="32sp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/Options"  
 app:layout\_constraintBottom\_toBottomOf="@id/Options" />  
  
 <DatePicker  
 android:id="@+id/spinnerPick"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="160dp"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:theme="@style/historial"  
 android:datePickerMode="spinner"  
 android:calendarViewShown="false"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/bar" />  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:id="@+id/calendarContainer"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:layout\_marginTop="220dp"  
 android:layout\_marginBottom="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/spinnerPick">  
  
 <androidx.viewpager.widget.ViewPager  
 android:id="@+id/calendarView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <ImageButton  
 android:id="@+id/Options"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:backgroundTint="@color/transparent"  
 android:contentDescription="@string/opciones"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_list" />  
  
 <ImageButton  
 android:id="@+id/Filter"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="@dimen/layout\_margin\_default"  
 android:backgroundTint="@color/transparent"  
 android:contentDescription="@string/etiquetas"  
 app:layout\_constraintStart\_toStartOf="@id/Options"  
 app:layout\_constraintEnd\_toEndOf="@id/Options"  
 app:layout\_constraintTop\_toBottomOf="@id/Options"  
 app:srcCompat="@drawable/ic\_filter" />  
  
 <Spinner  
 android:id="@+id/SpinnerFilter"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:contentDescription="@string/etiquetas"  
 android:alpha="0"  
 app:layout\_constraintStart\_toStartOf="@id/Filter"  
 app:layout\_constraintEnd\_toEndOf="@id/Filter"  
 app:layout\_constraintTop\_toTopOf="@id/Filter"  
 app:layout\_constraintBottom\_toBottomOf="@id/Filter" />  
  
 <ImageButton  
 android:id="@+id/verMontos"  
 android:layout\_width="64dp"  
 android:layout\_height="64dp"  
 android:layout\_margin="@dimen/layout\_margin\_default"  
 android:background="@drawable/blueshape"  
 android:contentDescription="@string/historial"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:srcCompat="@drawable/ic\_go" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/barrita"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 app:headerLayout="@layout/header\_historial"  
 app:menu="@menu/menu\_historial" />  
  
</androidx.drawerlayout.widget.DrawerLayout>

Un conjunto de letras negras en un fondo blanco

Descripción generada automáticamente

### Planes de ahorro

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:theme="@style/Theme.AppCompat.Light">  
  
 <ImageView  
 android:id="@+id/background"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:theme="@style/pda\_background"  
 android:contentDescription="@string/wallpaper" />  
  
 <ScrollView  
 android:id="@+id/scrollView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <TableLayout  
 android:id="@+id/cabezamontos"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="80dp"  
 android:layout\_marginEnd="10dp"  
 android:stretchColumns="0,1,2,3,4,5,6"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/g1topcell"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/HConcepto"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/ConceptoInteres"  
 android:paddingStart="7dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="18sp"  
 android:layout\_weight="4" />  
  
 <TextView  
 android:id="@+id/HValor"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/ValorFinal"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="18sp"  
 android:layout\_weight="3" />  
  
 <TextView  
 android:id="@+id/HVeces"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Fecha"  
 android:textAlignment="center"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="18sp"  
 android:layout\_weight="4" />  
  
 <TextView  
 android:id="@+id/HEtiqueta"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Etiqueta"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="18sp"  
 android:layout\_weight="4" />  
 </TableRow>  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/displayMontos"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
 app:layout\_constraintStart\_toStartOf="@+id/cabezamontos"  
 app:layout\_constraintTop\_toBottomOf="@+id/cabezamontos" />  
 </TableLayout>  
 </androidx.constraintlayout.widget.ConstraintLayout>  
 </ScrollView>  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_anchor="@+id/scrollView2"  
 app:layout\_anchorGravity="center">  
  
 <TextView  
 android:id="@+id/bar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="36sp"  
 android:layout\_marginStart="77dp"  
 android:layout\_marginEnd="77dp"  
 android:layout\_marginTop="16dp"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <ImageButton  
 android:id="@+id/goback"  
 android:layout\_width="45dp"  
 android:layout\_height="45dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:background="@color/transparent"  
 android:contentDescription="@string/goback"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_back"  
 app:tint="?attr/colorOnPrimary" />  
 </androidx.constraintlayout.widget.ConstraintLayout>  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

Patrón de fondo

Descripción generada automáticamente con confianza baja

### Finanzas

<?xml version="1.0" encoding="utf-8"?>  
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:id="@+id/background"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:contentDescription="@string/wallpaper"  
 android:theme="@style/finanzas\_background"  
 tools:layout\_editor\_absoluteX="0dp"  
 tools:layout\_editor\_absoluteY="0dp" />  
  
 <TextView  
 android:id="@+id/bar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:text="@string/finanzas"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="32sp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@id/Options"  
 app:layout\_constraintBottom\_toBottomOf="@id/Options" />  
  
 <ImageButton  
 android:id="@+id/Options"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:backgroundTint="@color/transparent"  
 android:contentDescription="@string/opciones"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_list" />  
  
 <TextView  
 android:id="@+id/prodhint"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:maxWidth="164dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="16sp"  
 app:layout\_constraintBottom\_toTopOf="@id/FinanzasItemImage"  
 app:layout\_constraintEnd\_toEndOf="@id/FinanzasItemImage"  
 app:layout\_constraintStart\_toStartOf="@id/FinanzasItemImage" />  
  
 <TextView  
 android:id="@+id/afinity"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="16dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="16sp"  
 app:layout\_constraintStart\_toEndOf="@id/FinanzasItemImage"  
 app:layout\_constraintTop\_toTopOf="@id/FinanzasItemImage" />  
  
 <TextView  
 android:id="@+id/saving"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="16sp"  
 app:layout\_constraintStart\_toEndOf="@id/FinanzasItemImage"  
 app:layout\_constraintTop\_toBottomOf="@id/afinity" />  
  
 <ImageView  
 android:id="@+id/FinanzasItemImage"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="@dimen/layout\_margin\_default"  
 android:layout\_marginTop="@dimen/layout\_margin\_default"  
 android:layout\_marginEnd="@dimen/layout\_margin\_default"  
 android:layout\_marginBottom="64dp"  
 android:contentDescription="@string/vermas"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@id/NoVerMERCALIBRE"  
 app:layout\_constraintEnd\_toStartOf="@id/FinanzasCalendar"  
 app:layout\_constraintHeight\_percent="0.25"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/FinanzasConsejosySugerencias"  
 app:layout\_constraintWidth\_percent="0.40" />  
  
 <ImageButton  
 android:id="@+id/VerMasMERCALIBRE"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="8dp"  
 android:background="@drawable/confshape"  
 android:contentDescription="@string/vermas"  
 android:padding="16dp"  
 app:layout\_constraintStart\_toStartOf="@id/FinanzasItemImage"  
 app:layout\_constraintTop\_toBottomOf="@id/FinanzasItemImage"  
 app:srcCompat="@drawable/ic\_watch"  
 app:tint="@color/G1" />  
  
 <ImageButton  
 android:id="@+id/VerMenosMERCALIBRE"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:background="@drawable/skipshape"  
 android:contentDescription="@string/vermenos"  
 app:layout\_constraintStart\_toEndOf="@id/VerMasMERCALIBRE"  
 app:layout\_constraintTop\_toBottomOf="@id/FinanzasItemImage"  
 app:srcCompat="@drawable/ic\_next"  
 app:tint="@color/O1" />  
  
 <ImageButton  
 android:id="@+id/NoVerMERCALIBRE"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:background="@drawable/cancshape"  
 android:contentDescription="@string/nover"  
 app:layout\_constraintStart\_toEndOf="@id/VerMenosMERCALIBRE"  
 app:layout\_constraintTop\_toBottomOf="@id/FinanzasItemImage"  
 app:srcCompat="@drawable/ic\_cancel"  
 app:tint="@color/R1" />  
  
 <TextView  
 android:id="@+id/FinanzasConsejosySugerencias"  
 android:layout\_width="160dp"  
 android:layout\_height="160dp"  
 android:layout\_margin="@dimen/layout\_margin\_default"  
 android:background="@drawable/ic\_circle"  
 android:backgroundTint="?attr/colorOnPrimary"  
 android:gravity="center"  
 android:text="@string/consejos\_y\_sugerencias"  
 android:textColor="?attr/colorOnSecondary"  
 android:textSize="18sp"  
 app:layout\_constraintBottom\_toTopOf="@id/FinanzasItemImage"  
 app:layout\_constraintEnd\_toStartOf="@id/Economia"  
 app:layout\_constraintHeight\_percent="0.25"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/Options"  
 app:layout\_constraintWidth\_percent="0.40" />  
  
 <TextView  
 android:id="@+id/FinanzasCalendar"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_margin="@dimen/layout\_margin\_default"  
 android:background="@drawable/ic\_calendar"  
 android:backgroundTint="?attr/colorOnPrimary"  
 android:gravity="center"  
 android:text="@string/eventos"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="20sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHeight\_percent="0.15"  
 app:layout\_constraintStart\_toEndOf="@id/FinanzasItemImage"  
 app:layout\_constraintTop\_toBottomOf="@id/Economia"  
 app:layout\_constraintWidth\_percent="0.30" />  
  
 <ImageView  
 android:id="@+id/Economia"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_margin="@dimen/layout\_margin\_default"  
 android:background="@drawable/ic\_charts"  
 android:contentDescription="@string/economia"  
 android:translationZ="100dp"  
 app:layout\_constraintBottom\_toTopOf="@id/FinanzasCalendar"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHeight\_percent="0.25"  
 app:layout\_constraintStart\_toEndOf="@id/FinanzasConsejosySugerencias"  
 app:layout\_constraintTop\_toBottomOf="@id/Options"  
 app:layout\_constraintWidth\_percent="0.45" />  
  
 <TextView  
 android:id="@+id/EconomiaTV"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="-18dp"  
 android:gravity="center"  
 android:text="@string/economia"  
 android:textColor="?attr/colorOnPrimary"  
 android:textSize="20sp"  
 app:layout\_constraintEnd\_toEndOf="@id/Economia"  
 app:layout\_constraintStart\_toStartOf="@id/Economia"  
 app:layout\_constraintTop\_toBottomOf="@id/Economia" />  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/barrita"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 app:headerLayout="@layout/header\_finanzas"  
 app:menu="@menu/menu\_finanzas" />  
  
</androidx.drawerlayout.widget.DrawerLayout>

Gráfico, Gráfico en cascada

Descripción generada automáticamente

## Colores

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
  
 <!-- AlexTest -->  
 <color name="gris">#C9CCD6</color>  
 <color name="amarilloetiquetas">#E6D780</color>  
   
 <!-- Purple -->  
 <color name="P0">#FF400060</color>  
 <color name="P1">#FF5700A5</color>  
 <color name="P2">#8234C6</color>  
 <color name="P3">#FFD5ABFF</color>  
 <color name="P4">#FFE5CBFF</color>  
 <color name="P5">#FFEFDEFE</color>  
  
 <!-- Red -->  
 <color name="R0">#FFA80B1A</color>  
 <color name="R1">#FFDC3545</color>  
 <color name="R2">#FFEF476F</color>  
 <color name="R3">#FFFF6E6E</color>  
 <color name="R4">#FFFF9AA2</color>  
 <color name="R5">#FFFFC6C7</color>  
  
 <!-- Orange -->  
 <color name="O0">#FFA86502</color>  
 <color name="O1">#FFDC9225</color>  
 <color name="O2">#FFEF873F</color>  
 <color name="O3">#E89B57</color>  
 <color name="O4">#FFFFBD9A</color>  
 <color name="O5">#FFFFC6C7</color>  
  
 <!-- Yellow -->  
 <color name="Y1">#FFFFCA3A</color>  
 <color name="Y2">#FFFFBF5F</color>  
 <color name="Y3">#FFFDB813</color>  
 <color name="Y4">#FFFDCB6E</color>  
 <color name="Y5">#FFFFE7A6</color>  
  
 <!-- Green -->  
 <color name="G0">#073211</color>  
 <color name="G1">#135C24</color>  
 <color name="G2">#3AA824</color>  
 <color name="G3">#FFA3E635</color>  
 <color name="G4">#FFB4ED50</color>  
 <color name="G5">#FFC9F7B9</color>  
  
 <!-- Blue -->  
 <color name="B0">#124A88</color>  
 <color name="B1">#046DC8</color>  
 <color name="B2">#FF3DADF2</color>  
 <color name="B3">#FF85C1E9</color>  
 <color name="B4">#FFAFDCEC</color>  
 <color name="B5">#FFC7E7F3</color>  
 <color name="B6">#D6F2F8</color>  
  
 <!-- Neutral -->  
 <color name="black">#FF000000</color>  
 <color name="N0">#292929</color>  
 <color name="N1">#FF6C757D</color>  
 <color name="N2">#FF808080</color>  
 <color name="N3">#FFA9A9A9</color>  
 <color name="N4">#FFC8C8C8</color>  
 <color name="N5">#FFE0E0E0</color>  
 <color name="white">#FFFFFFFF</color>  
  
 <color name="grey">#FFC9CCD6</color>  
  
 <!-- X for b&w themes -->  
 <color name="X0">#0A0A0A</color>  
 <color name="X1">#222</color>  
 <color name="X2">#333</color>  
 <color name="X3">#EEE</color>  
 <color name="X4">#F0F0F0</color>  
 <color name="X5">#FFF</color>  
  
 <!-- Piechart -->  
 <color name="V1">#FFFDB813</color>  
 <color name="V2">#FF3DADF2</color>  
 <color name="V3">#FF3AA824</color>  
 <color name="V4">#FF124A88</color>  
 <color name="V5">#FFDC9225</color>  
 <color name="V6">#FFDC3545</color>  
 <color name="V7">#FF5700A5</color>  
 <color name="V8">#FFA80B1A</color>  
  
 <color name="V9">#FFB4ED50</color>  
 <color name="V10">#FFAFDCEC</color>  
 <color name="V11">#FFEF476F</color>  
 <color name="V12">#FFFFBD9A</color>  
 <color name="V13">#FFFFBF5F</color>  
 <color name="V14">#FF79529F</color>  
  
  
  
 <!-- Android -->  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="transparent">#00000000</color>  
</resources>