



UNIVERSITAT_{DE}
BARCELONA

PROYECTO INTEGRADO DE SOFTWARE

MEMORIA

Práctica 2: Modelo de dominio + Demo

COINNOTE

AUTORES:

MARÍA ROMÁN MARTÍN
MIGUEL HUAYLLAS CHOQUE
ELENA DEGRÀCIA JARQUE
ALICIA CARRASCO GUARDIOLA

NIUBs:

20222252
17510710
20206863
20150513

ABRIL DE 2021

Indice

1	Introducción	2
2	Cuenta de prueba para el profesor	2
3	Modelo de dominio	2
4	Desarrollo del proyecto	4
4.1	Familiarización con Android Studio	4
4.2	Layouts	4
4.3	Flujo de la aplicación	5
4.4	Firebase	5
4.5	Modelo View ViewModel	6
4.6	Librerías usadas	6
5	Tutorial	7
5.1	Página inicial, registro e iniciar sesión	8
5.2	Página principal y crear nota	10
5.3	Visualizar nota y crear gasto	12
5.4	Perfil	13
6	Futuras implementaciones	14
7	Distribución del trabajo	15
8	Conclusiones	15

1 Introducción

Tras la última entrega, en la que definimos las funcionalidades, llegó el momento de empezar a dar forma a CoinNote, presentando su correspondiente modelo de dominio, que muestra la interacción de clases, acompañado de una demo de la aplicación. Para dicha demo se nos han ido proporcionando ideas a lo largo de éstas últimas semanas, para que pudiéramos implementarlas y así enriquecer nuestro proyecto. Por otro lado, hemos usado ideas obtenidas de sitios externos para adaptar CoinNote a la que consideramos su mejor versión. La demo que presentamos consiste en una aplicación totalmente funcional en el apartado visual, además de incluir algunas funcionalidades de Firebase como la authentication, lectura y escritura en la Firestore.

2 Cuenta de prueba para el profesor

Para facilitar la interacción del profesor con la aplicación, adjuntamos el correo y contraseña, con algunas notas y gastos. Así le será más fácil realizar los testeos.

Email:

"pis2021@gmail.com"

Password:

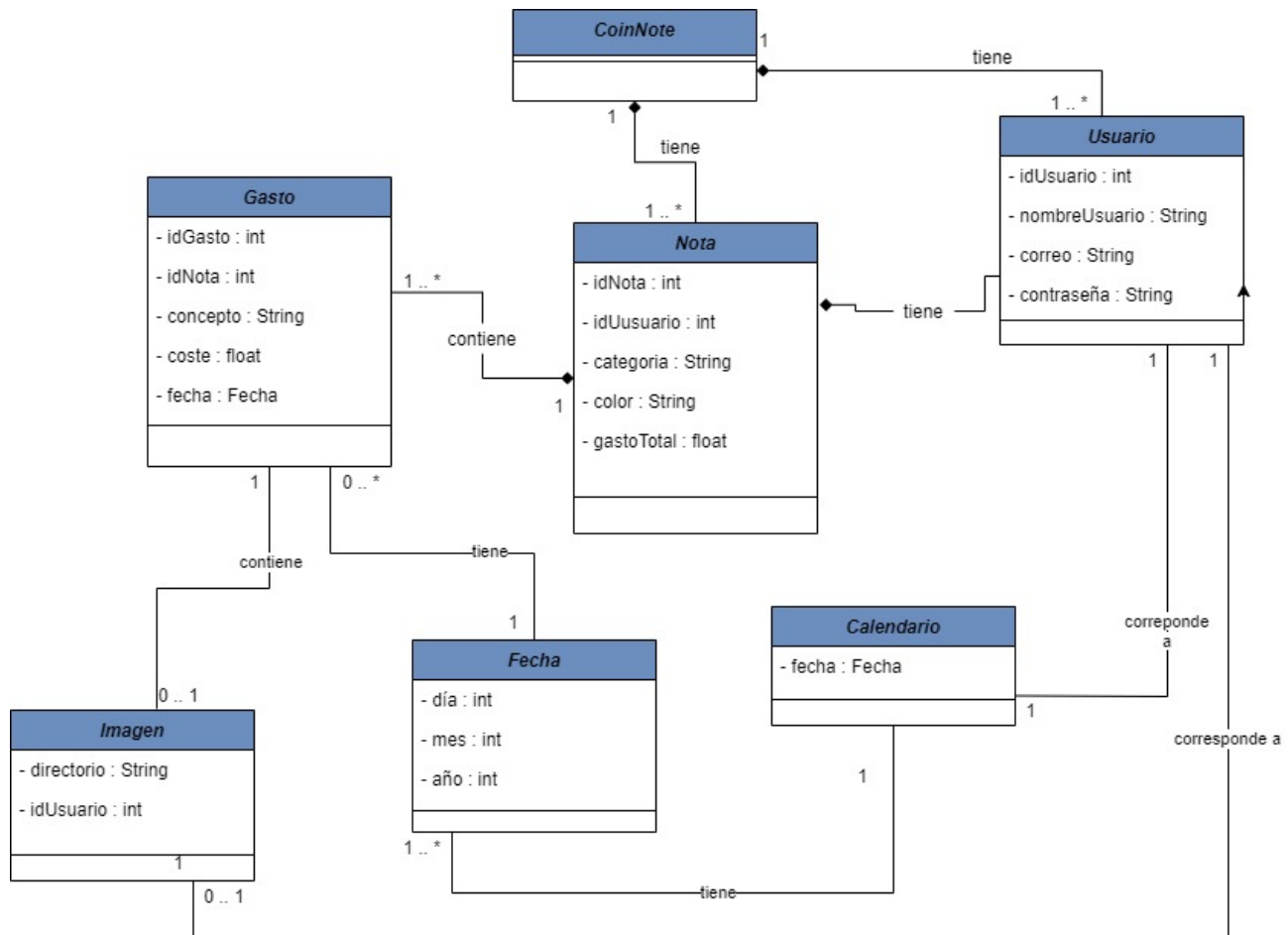
"PISpis2021"

Username:

"PIS2021"

3 Modelo de dominio

En relación con nuestro modelo de dominio, hemos ido teniendo varias ideas a medida que íbamos realizando e implementando nuestra aplicación. A día de hoy, seguimos teniendo alguna que otra duda en relación a como combinarlo con Firebase, ya que en modelos anteriores de otras asignaturas, no disponíamos de una base de datos como la que estamos utilizando ahora.



Actualmente, nuestro modelo consta de siete clases distintas. Primero tenemos la clase `CoinNote`, que es la propia aplicación. Seguidamente tenemos el usuario. Esta consta de un `idUsuario`, para poder reconocer a que usuario nos referimos. Este `idUsuario` se crea automáticamente cuando un cliente se registra en la aplicación. También disponemos en la misma clase de un nombre de usuario (username), un mail y contraseña. Usuario y `CoinNote` tienen una relación de composición, al igual que Usuario y `Nota`, es decir, `CoinNote` no existiría si no existieran usuarios y una nota no existiría si no existe un usuario. Estos dos tienen una relación en la que `CoinNote` puede tener entre uno e infinito usuarios y usuarios solo puede tener un `CoinNote`. Lo mismo sucede con la clase `Nota`. Esta tiene la misma relación con `CoinNote`, ya que si no hay notas, no existe la aplicación. Para esta clase tendremos tres atributos. El primero es `idNotas`. Este nos ayuda a poder reconocer x nota, ya que puede haber dos notas que se llamen iguales. También tendremos un atributo `idUsuario`, ya que cada usuario tiene sus propias notas. De esta manera podemos saber nosotros que notas mostrar dependiendo del usuario que se ha entrado. Por último tendremos una categoría, un color y un gasto total para cada nota.

`Gasto` es la cuarta clase que encontramos. Esta tiene un `idGasto`, un concepto y un coste, un objeto `Fecha` (explicado posteriormente) y un `idNota`. Este `idNota` nos ayudará a reconocer a que nota pertenece un gasto. La relación que tienen `Gasto` y `Nota` es de composición, como hemos visto anteriormente. No puede existir una nota si no existe un gasto.

La clase `Fecha` la hemos implementado para poder crear fechas reales. Esta está constituida por tres atributos: año, mes y día. Como se ha mencionado anteriormente, la clase `Gasto` tendrá un atributo tipo `Fecha`. Esto significa que a cada `Gasto` le corresponde una única fecha, mientras que una fecha puede aparecer en distintos gastos.

Seguidamente tenemos la clase calendario. Esta tiene como atributo un objeto Fecha. Como se puede observar en el diagrama, la relación que hay entre Fecha y Calendario es que un calendario contiene varias fechas mientras que una fecha solo corresponde a un calendario. También se ve una relación entre usuario y calendario, ya que a cada usuario le corresponde solo un calendario, donde podrá observar sus gastos.

Por último tenemos la clase Imagen. Para esta clase seguimos teniendo alguna duda, ya que las imágenes que tenemos en nuestra aplicación se guardan en la base de datos Firebase. Por lo tanto, hemos decidido que esta clase Imagen tenga como atributo un directorio para poder obtener la imagen pertinente de Firebase. También necesitamos saber de qué usuario queremos obtener la imagen deseada, ya que al guardar una imagen en la aplicación, Firebase crea una carpeta con el idUsuario. Para ello, hemos pensado que esta clase debería tenerlo. Esta clase está relacionada tanto con gasto como con usuario, ya que cada gasto puede tener una imagen y cada usuario puede tener una foto de perfil.

4 Desarrollo del proyecto

Para el desarrollo técnico de la práctica se aconsejó por parte del profesorado comenzar por la parte visual, convirtiendo los mockups a realidad mediante Android Studio. Posterior a eso unir todos los mockups con Intents para así tener al menos una aplicación visual que, aunque sin lógica, se viera y sintiera como una app real.

El siguiente paso claro era pensar en la lógica y conexión con Firebase. Pero ya, debido a nuestro enfoque y falta de experiencia, tardamos mas en crear la parte visual. Respecto a la parte lógica, solo se pudieron implementar ideas en el proyecto, aunque sin la arquitectura MVVM.

A continuación se describe un resumen de todo el trabajo hecho, dividido en etapas, además de incluir las funcionalidades que se implementaron en cada una de ellas.

4.1 Familiarización con Android Studio

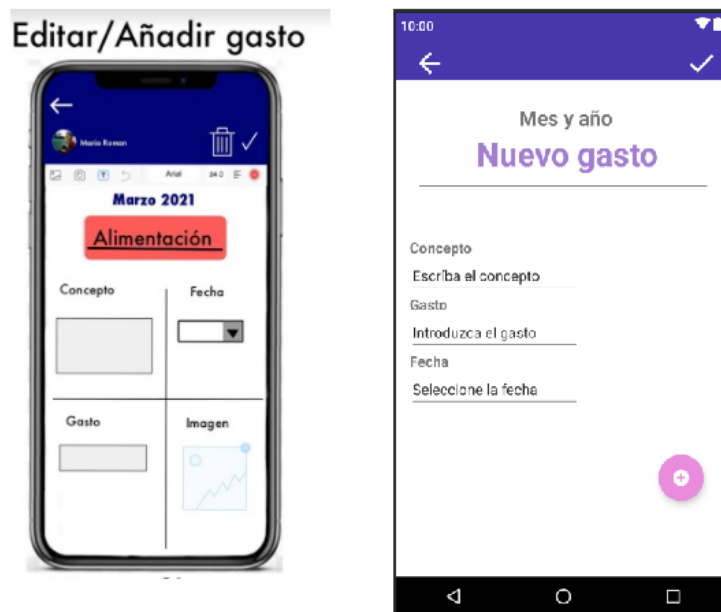
El primer paso que se decidió fue el de tomar contacto con Android Studio. Para ello se repasó la documentación ya proporcionada y se implementaron proyectos independientes para consolidar dichos conocimientos.

En esta etapa, además, se dedicó un periodo de tiempo para decidir los estándares y reglas con los que se trabajaría para evitar diseños totalmente dispares entre sí.

Se eligió una paleta de colores para diseñar los distintos elementos y componentes de la aplicación como los botones o el fondo.

4.2 Layouts

En esta etapa nos acostumbramos mas android studio pues hasta ahora íbamos temerosos pues era algo que nunca habíamos usado. Entre lo mas destacable de esta etapa era darnos cuenta de las limitaciones que teníamos a la hora de pasar un mockup a una actividad y también las limitaciones que podíamos hacer con Android.



4.3 Flujo de la aplicación

Una vez finalizados los layouts, se añadieron Intents para que CoinNote fuera funcional y se pudiera navegar a través de ella. Para ello, se usaron diferentes recursos, tales como una toolbar, botones flotantes, elementos clickables, etc.

Finalmente, se resolvieron los distintos bugs que aparecían al usar la app, además de los ocasionales problemas de compilación.

4.4 Firebase

A continuación, se inició la que consideramos la etapa más complicada, la cual constaba de agregar elementos en la base de datos: notas, gastos y usuarios. Para las notas y gastos, se cargarían al iniciar la aplicación y, por lo relacionado con los usuarios, se dispondría de la base de datos para su autenticación.

A modo de síntesis, las funcionalidades de Firebase que se han usado son las siguientes:

- Agregar una base de datos a la app para mostrar las notas (Firestore)
- Agregar autenticación para se pueda hacer inicio de sesión y registro (Authentication)
- Guardar archivos como imágenes en la nube (Storage)

A la par de usar estos servicios ofrecidos por Firebase se necesitaban ciertos elementos y líneas de código para que funcionaran en la app. Éstos son los siguientes:

- RecyclerView + Grid para cargar las notas con la base de datos y mostrarlas.
- Card como elemento que sería la representación de una nota en el RecyclerView.
- Diálogos para meter información de las notas, filtrarlas o compartirlas.
- Diálogos propios de Google para usar un login con este.

- Envío de correo electrónico, en caso de haber olvidado la contraseña.
- Librerías externas para agregar funciones, como escoger un color o borrar un gasto deslizando un dedo.
- View Pager, para implementar las dos pantallas en la pagina principal (Mis notas y Calendario).
- Calendario para mostrar en qué fecha se agregaron las notas y gastos.

Entre las implementaciones explicadas anteriormente y otras líneas de código en las actividades correspondientes se logró tener una app que se sintiera real.

4.5 Modelo View ViewModel

El motivo principal por el cual no se pudo implementar correctamente el MVVM es debido a la priorización de los apartados anteriores. Se quería tener el diseño de layouts definitivo, los cuales podrían interactuar con ellos de forma correcta, además de implementar la base de datos y las funcionalidades más básicas de nuestra aplicación.

Un problema que impidió la rápida implementación del patrón MVVM fue un mal entendimiento, dado que creamos casi todos los layouts con activitys, en vez de usar fragments, error ante el cual se pueden aplicar diferentes soluciones:

- Cambiar todo a Fragments, aunque parece ser el menos óptimo, dado el tiempo que requiere.
- Hacer el ModelView usando los patrones Singleton y Factory para que, aunque se instancie en diferentes actividades, los datos sean los mismos.
- Serializar y deserializar el ModelView con Parcelable y enviarlos con Intents entre actividades.

4.6 Librerías usadas

A continuación se muestran, junto a un pequeño resumen, las librerías implementadas más relevantes:

"yukuku ambilwarna"

Librería que sirve para escoger el color de manera dinámica. En este caso servirá para que el usuario pueda escoger el color de la nota a través de un diálogo donde se muestra una paleta de colores extensa.

"firebaseui:firebase-ui-firestore"

Librería que provee de un Adapter personalizado para usar Firebase.

"com.squareup.picasso:picasso:2.71828"











Librería para realizar acciones deslizando el dedo en componentes, usada para eliminar los gastos.

Además de usar otras librerías complementarias para la aplicación como las librerías de Google para usar los servicios de Firebase.

5 Tutorial

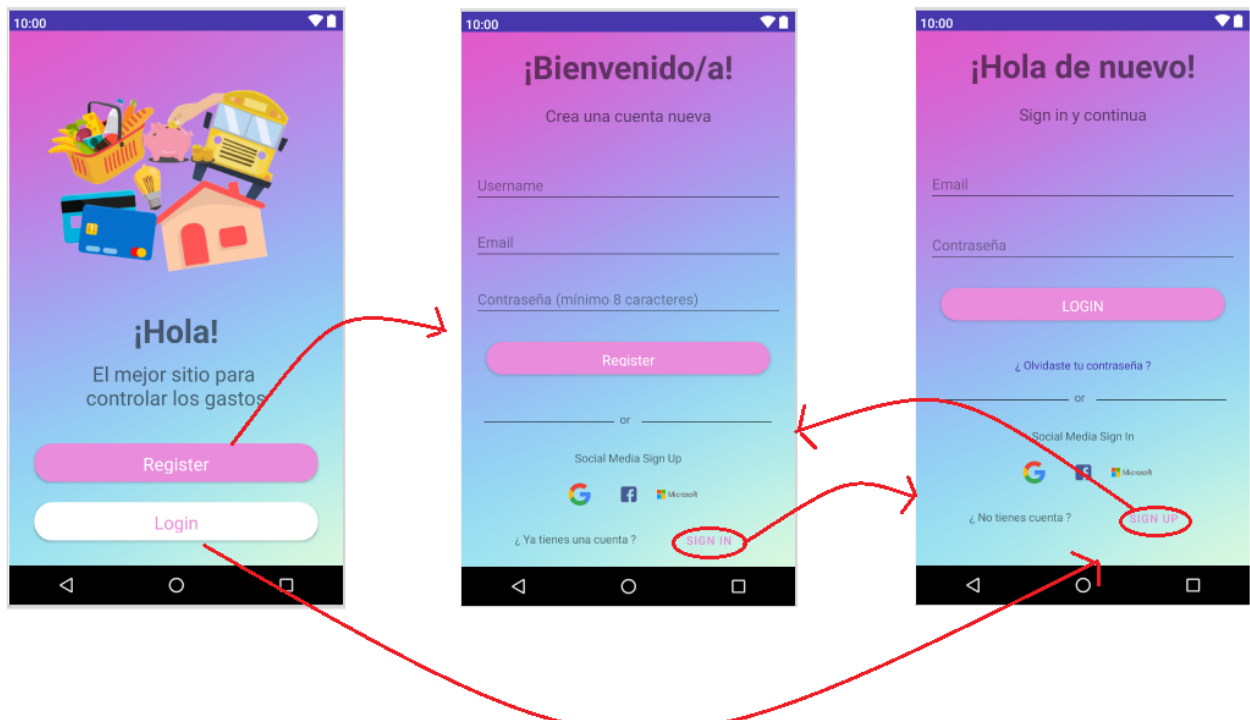
A continuación se explicaran los layouts y la navegación entre estos. Se ha intentado en todo momento seguir y respetar el diseño de los mockups originales que se entregaron en la práctica anterior. No obstante, se han hecho algunos pequeños cambios para facilitar y mejorar la experiencia del usuario con nuestra aplicación.

Como se ha mencionado previamente, antes de empezar a crear los layouts se priorizó crear un diseño y una paleta de colores para todos los elementos de la aplicación, como ahora los botones. La paleta será la siguiente:

	<code><color name="Primary">#4737ad</color> <!--</code>
	<code><color name="Text">#a37dd1</color> <!-- Te</code>
	<code><color name="Tertiary">#E15AC9</color></code>
	<code><color name="Highlight">#B7EDE8</color></code>
	<code><color name="Background">#FFFFFF</color></code>
	<code><color name="Paragraph">#abd1c6</color></code>
	<code><color name="HeadLine">#A549D6</color></code>
	<code><color name="Button">#E88CDB</color> <!--</code>
	<code><color name="ButtonText">#FFFFFF</color></code>
	<code><color name="Stroke">#001e1d</color></code>

Después de varias pruebas con distintas combinaciones, se optó para una paleta utilizando distintas cromáticas del color lila y rosa.

5.1 Página inicial, registro e iniciar sesión



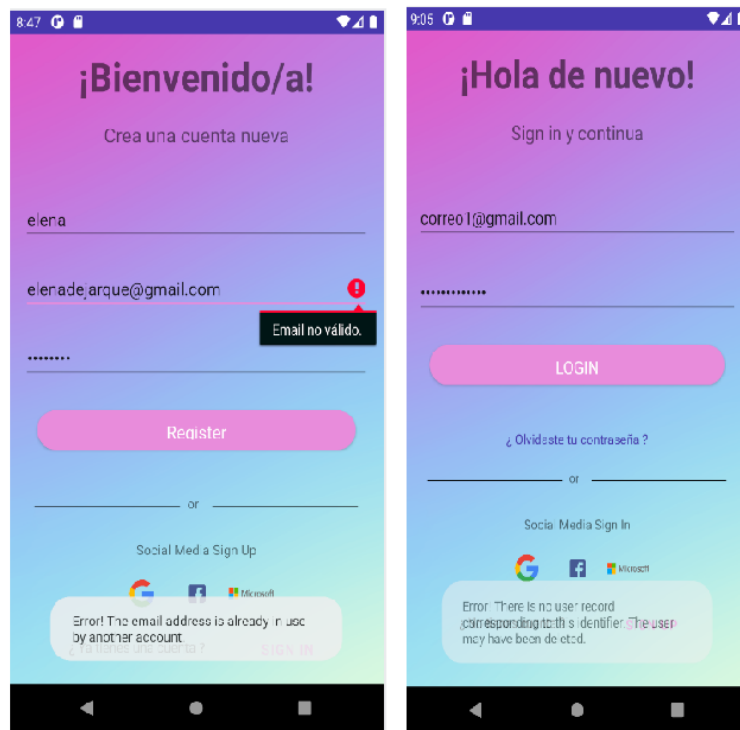
Cuando un usuario abra por primera vez la aplicación o no haya abierto la sesión anteriormente se le mostrará la página de bienvenida.

En ella se muestra el logo de la app, un mensaje de bienvenida y dos botones, uno para registrarse en el caso de no tener una cuenta y otro para hacer log in en el caso que sí se tenga cuenta.

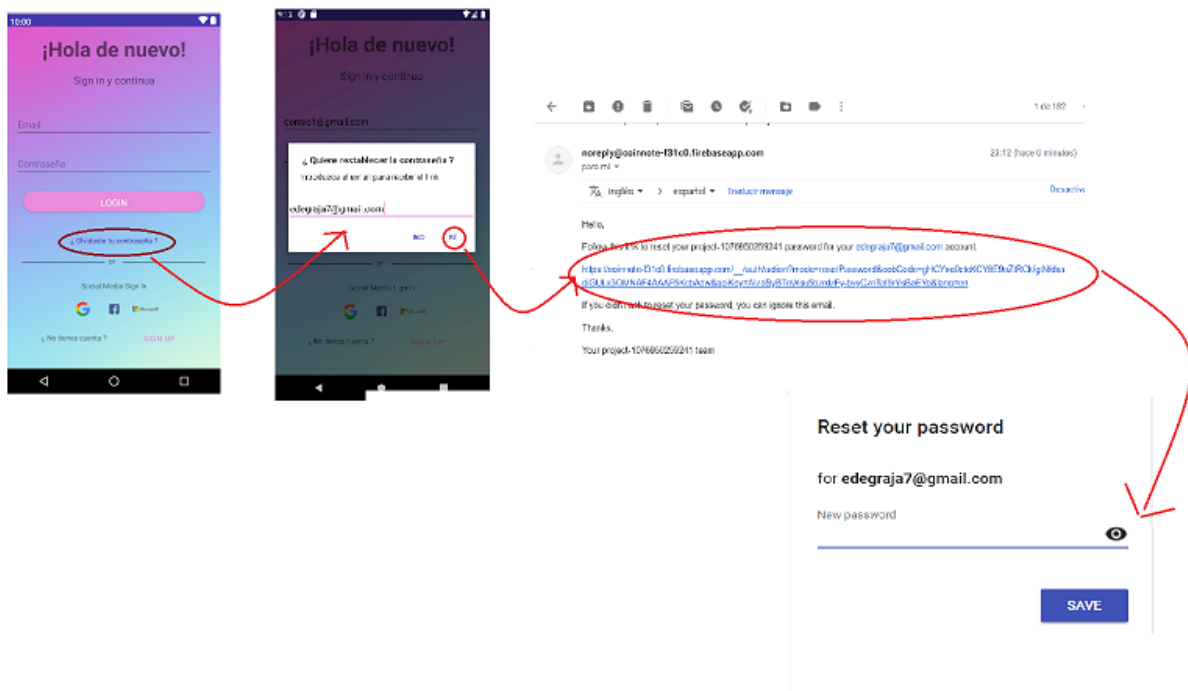
El usuario podrá escoger lo que quiere hacer, si registrarse o iniciar sesión. Para registrarse será necesario introducir un nombre de usuario, un correo electrónico y una contraseña que deberá estar formada por mínimo 8 caracteres, 1 número, 1 mayúscula y una minúscula.

También podrá registrarse a través de Google. En el caso que uno de los parámetros introducidos sea incorrecto no se permitirá hacer el registro y se mostrará un mensaje de error.

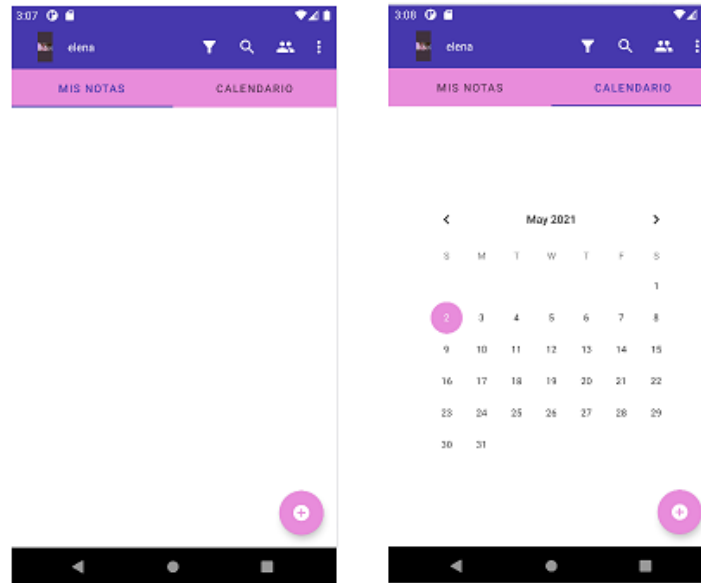
Se dará la opción al cliente de ir a la página de log in a partir de la página de registro y viceversa a través del texto clickable en la parte inferior derecha de la pantalla. Para iniciar sesión sólo habrá que introducir el correo y la contraseña, en el caso que uno de los parámetros sea incorrecto (por ejemplo, que no haya ninguna cuenta asociada al correo) se mostrará un mensaje de error y no se le mostrará la página principal.



Si el usuario no recuerda la contraseña podrá restablecerla a través del texto clickable que se encuentra justo debajo del botón de LOGIN. El usuario tendrá que introducir su correo y el sistema le enviará un correo con un link para introducir su nueva contraseña.

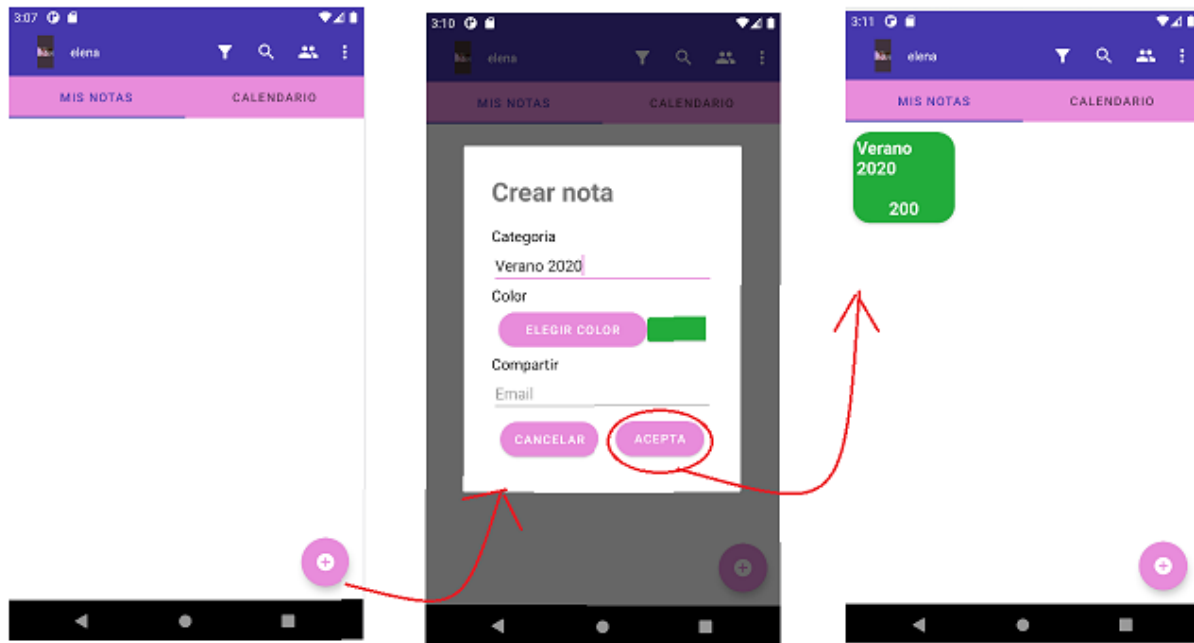


5.2 Página principal y crear nota

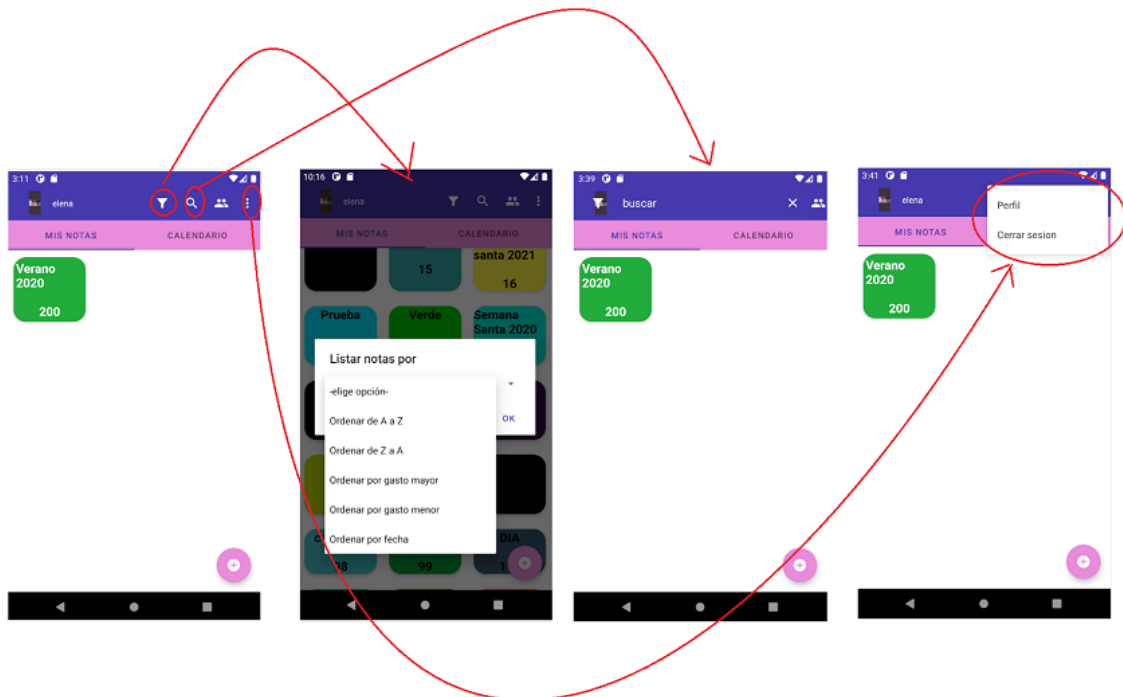


Una vez se haya podido iniciar sesión se le mostrará la página principal. En ella se ha modificado el diseño plantado inicialmente en los mockups. Se ha creado un `TabLayout` y un `View Pager`, en una pestaña se mostrarán las notas clickables y en otra el calendario donde más adelante se podrán ver los gastos asociados a las distintas fechas.

En la parte inferior derecha de la pantalla hay un botón flotante para añadir una nueva nota a la página principal. El usuario tendrá que introducir una categoría y un color asociado a esta. Si el usuario no rellena la casilla de categoría no se podrá crear una nota.

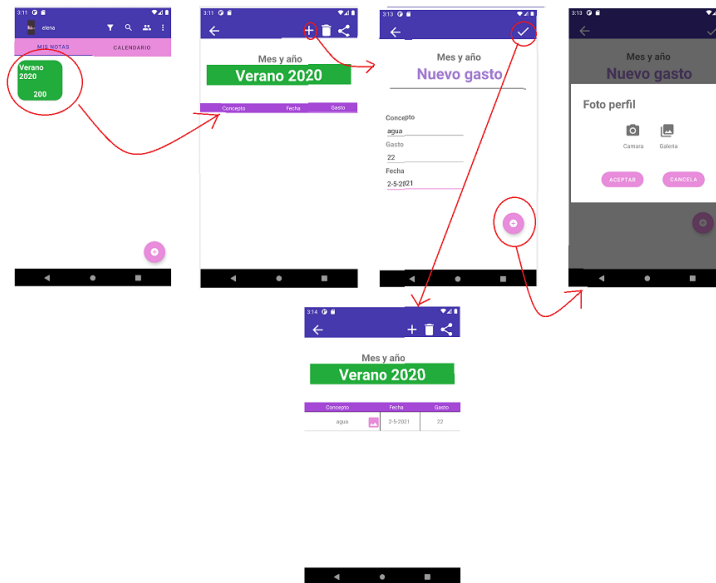


Por lo que hace la toolbar, se podrá ver la foto y el nombre de usuario, la opción de listar las notas, buscar una nota en concreto, visualizar las notas compartidas, visualizar el perfil y cerrar la sesión.



5.3 Visualizar nota y crear gasto

Si el usuario quiere visualizar todos los gastos relacionados con las notas solo tendrá que hacer click en la nota y se le mostrará la siguiente ventana:

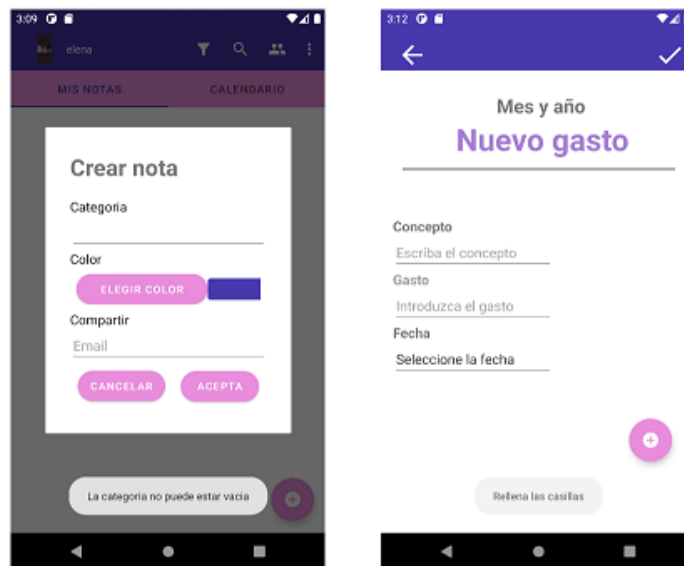


Se ha decidido simplificar la ventana del mockup reduciendo el número de botones en ella. En la toolbar se puede añadir un nuevo gasto, eliminar la nota en completo y compartir la nota.

Se le podrá asociar una foto en cada gasto que se podrá visualizar clickando en el botón al lado del concepto. Para eliminar un gasto se podrá hacer swipe a la derecha o a la izquierda. El gasto se eliminará de la base de datos también.

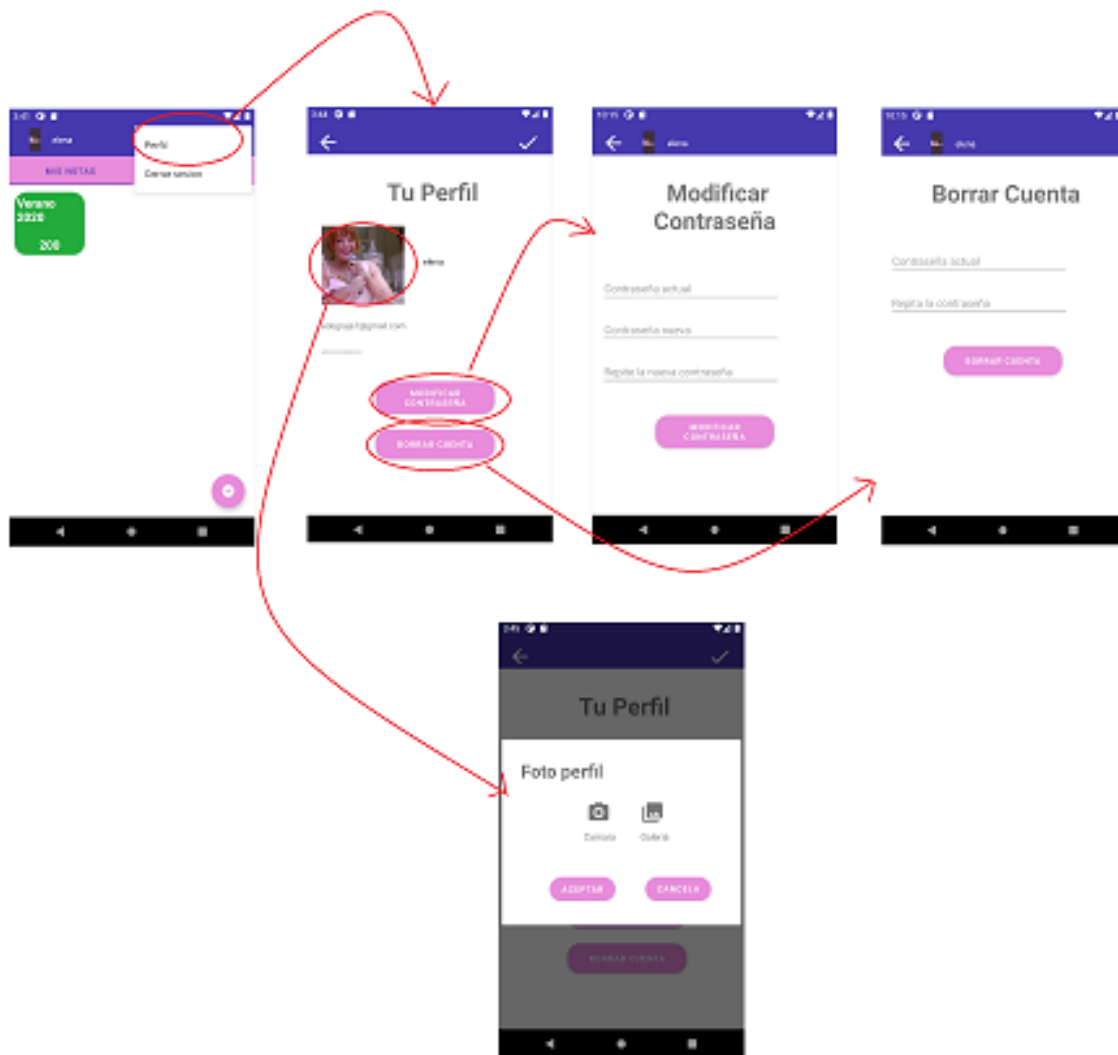
Para editar un gasto ya existente solo habrá que hacer click encima de la nota.

Para crear un gasto se pedirá que se introduzca el concepto del gasto, la fecha y la cantidad del gasto. También se da la opción de añadir una foto del recibo de la compra por ejemplo. La podrá añadir una foto ya existente de la galería o bien hacer una foto con la cámara.



5.4 Perfil

El usuario podrá modificar sus datos a través del perfil. Podrá cambiar su foto de perfil y se espera que en un futuro pueda modificar su contraseña y eliminar la cuenta. Para añadir o cambiar la foto de perfil se podrá acceder a la galería o a la cámara.



6 Futuras implementaciones

A continuación se enumeran las funcionalidades que se prevee que estén implementadas para el proyecto final:

- Solucionar bugs, como iniciar sesión en google.
- Agregar gastos al calendario.
- Funcionalidades de eliminar nota, cambiar contraseña y borrar cuenta.
- Posibilidad de compartir notas con otros usuarios.
- Asociar la funcionalidad de listar notas al botón desplegable y poder buscarlas.
- Cálculo del coste total de cada nota.
- Inicio de sesión con Facebook y Microsoft.
- Implementar modelo MVVM

7 Distribución del trabajo

Respecto a la distribución de las faenas, se han seguido las mismas reglas que para la anterior entrega, dado que éstas tuvieron un resultado exitoso. Se han hecho una o dos reuniones semanales en las que se exponía el trabajo realizado, se comentaba algún problema que pudiera haber surgido o se hacían observaciones y se asignaba una nueva tarea.

Se inició haciendo entre todos el modelo de dominio, debatiendo tanto las clases, como la relación entre ellas, o sus atributos..., hasta tener una versión principal sobre la que trabajar. A continuación se comenzó la parte de diseño, pasar los mockups a Android Studio mediante activitys, fragments, etc. Cada uno tenía uno o dos, en función de la dificultad, y siguieron distribuyendo hasta que no quedaba ninguno.

Seguidamente se implementó una lógica para que las activitys se pudieran relacionar mediante los botones, además de aprovechar para pulir algún aspecto a mejorar. Ya se tenía, pues, el flujo de la aplicación hecho.

En siguiente lugar se conectó nuestro proyecto a Firebase, para poder implementar el guardado de usuarios, notas y gastos, además de añadir intents a otras aplicaciones (galería, cámara, google, etc).

Finalmente, se distribuyeron las funcionalidades a realizar entre todos los participantes del grupo y se fueron implementando las máximas hasta la fecha de entrega.

8 Conclusiones

A modo de conclusión, estamos muy satisfechos, dado que creemos que hemos tocado muchos aspectos distintos de los que nos proporciona Android Studio. Si bien teníamos problemas en algún momento, contactábamos con los demás integrantes para ayudarnos y aprender unos de otros. El resultado final consideramos que se asemeja bastante al esperado para el proyecto final, aunque falten algunas funcionalidades.