Firebase

De backend van het project is Firebase, hiervoor is gekozen, omdat Firebase ons als team alles bied wat wij nodig hadden. En omdat het een bestaand product is, hoeven wij ons niet druk te maken over veel zaken. Zo kunnen wij ons meer richting op de functionaliteit van de app. Het zelf maken en hosten van een back-end is ook prima mogelijk en op de lange duur zelfs beter, omdat je dan volledige controle hebt, was het bij dit project niet haalbaar.

Van Firebase maken wij gebruik van de volgende sub onderdelen:

- Authentication
- Firestore database
- Storage

Authentication

De authentication zorgt ervoor dat mensen een account kunnen maken via een email adres en wachtwoord. Het voordeel is dat wij hier als team weinig hoeven te doen om een account aan te maken. Nadat het account is aangemaakt, krijgt de gebruiker een email, om zo het adres van de gebruiker te verifiëren. Als een gebruikt zijn account niet heeft geverifieerd, kan hij ook niet gebruik maken van de app. Vervolgens kan de gebruiker gewoon met deze gegeven in de app inloggen. Door Firebase wordt de rest afgehandeld, of een gebruiker geldig is, hij nog ingelogd etc.

Firestore

Nu de gebruiker een account heeft, wordt er meteen onder zijn user ID een document aangemaakt waar de gegevens van de gebruiker worden opgeslagen. Hieronder is de opzet van dit document te zien. Op het moment dat de gebruiker een account aanmaakt zullen veel van deze velden `null` zijn. Zo om zo weinig gegeven op te slaan van een gebruiker.

```
{
    "accountType"       : Integer = 0,
    "emailAddress"      : String,
    "files" : [
        {
            "extension" : String,
            "favourite" : Boolean,
            "hidden"    : Boolean,
            "name"      : String,
            "thumbnail" : Boolean,
        },
        ...
    ]
    "firstName"         : String = null,
    "lastName"          : String = null,
    "pin"               : String = null,
    "profilePicture"    : String = null,
    "verified"          : Boolean = false
}
```

Verder word er voor de gebruiker een lijst (`files`) bijgehouden waarin staat welke bestanden er van deze gebruiker zijn. Deze lijst bestaat weer uit array's. Deze gegevens zijn er alleen om zo het bestand binnen de app te kunnen classificeren. Dit houdt in, het bijhouden of het bestanden verborgen is, en of het een thumbnail heeft, en andere zaken. Voorbeeld:

```
{
    "extension" : ".jpg"
    "favourite" : true
    "hidden"    : false
    "name"      : "sls_functionalparts"
    "thumbnail" : true
}
```

`accountType` Kan aangeven of een gebruiker betaald voor de dienst. Dit is verder niet uitwerkt, maar op basis van de waarde hier kunnen regels worden op gelegd. Zo is nu type 0, een gratis gebruiker.

`pin` Dit is de pin die door gebruiker is ingesteld om bij de verborgen map te komen. Deze pin is max 4 cijfers lang. Voorbeeld: `"1234"`

`profilePicture` Hier komt een `StorageReferenceUri` te staan nadat een gebruiker een profiel foto heeft geüpload. Voorbeeld: `gs://atos-msafe.appspot.com/cI90JMciV8WqfS2rnPE9QLjCGo93/profilePicture/_profilePicture.jpg`.

`verified` Dit is de Boolean die pas `True` wordt nadat de gebruiker de link in de verificatie email heeft geopend. Dit veld moet waar zijn, wil de gebruiker gebruik kunnen maken van de dienst.

Cloud storage

De bestanden zelf worden geüpload naar de Firebase Cloud storage. Dit is een "bucket" waar de ontwikkelaar zelf structure in moet aanbrengen. De Bucket mag worden behandeld als elke andere bestanden structure, dus er kunnen gemakkelijk mappen worden aan gemaakt.
Op het moment dat een gebruikt een bestand uploaden, wordt er er eest een map aangemaakt onder zijn user ID:
(*hier is de user ID: cI90JMciV8WqfS2rnPE9QLjCGo93*)
`gs://atos-msafe.appspot.com/cI90JMciV8WqfS2rnPE9QLjCGo93`
De volgende stap is dat er een map wordt aangemaakt onder de naam van het bestand zelf:
`gs://atos-msafe.appspot.com/cI90JMciV8WqfS2rnPE9QLjCGo93/sls_functionalparts`
De naam van deze map komt overeen met `"name"` in de Firestore document. In deze map wordt vervolgens het bestand zelf geüpload `filename + extension`. mocht het hier gaan om een `.png`, `jpg` of `jpeg` dan zal er in deze map een thumbnail map worden gemaakt:
`gs://atos-msafe.appspot.com/cI90JMciV8WqfS2rnPE9QLjCGo93/sls_functionalparts/THUMB`.
Hier zal een gecomprimeerde versie (15% +- van het origineel) van het bestand komen te staan, deze heeft verder ook een andere naam (`filename + "_THUMB" + extension`).
`gs://atos-msafe.appspot.com/cI90JMciV8WqfS2rnPE9QLjCGo93/sls_functionalparts/THUMB/sls_functionalparts_THUMB.jpg`

Dit alles is gekoppeld aan het document van de Firestore, in de array van `files`.

Koppelingen met andere app

ontvangen van een bestand

Het is mogelijk om bestanden te "delen" naar de mSafe app. Dit gebeurt met de intents. In de `AndroidManifest.xml` zijn er `<intent-filter>`'s aanwezig. Zo kan de app alleen de "juiste" bestanden ontvangen. De volgende bestanden worden geaccepteerd:

```
<action android:name="android.intent.action.SEND" />
<action android:name="android.intent.action.OPEN_DOCUMENT" />
<category android:name="android.intent.category.DEFAULT" />

<data android:mimeType="image/*" />
<data android:mimeType="audio/*" />
<data android:mimeType="video/*" /> =
<data android:mimeType="application/pdf" />             <!-- PDF -->
<data android:mimeType="application/zip" />             <!-- .zip -->
<data android:mimeType="application/msword" />          <!-- Word -->
<data android:mimeType="application/vnd.ms-excel" />    <!-- Excel -->
<data android:mimeType="application/mspowerpoint" />    <!-- Powerpoint
-->
```

Op het moment dat de app opstart ontvangt `SetupActivit.kt` de intent, en wordt er gekeken of er iets moet worden gedaan.

```
when (intent?.action) {
```

```
    Intent.ACTION_SEND -> {
        getUriToFile(intent)
    }
}
```

Moet er wat gebeuren dan, zal de `Uri` van het bestanden worden opgeslagen, `getUriToFile`. Als de gebruiker dan ook ingelogd is, zal de starten via de `Intent` wordt dan de `Uri meegegeven.

```
private fun launchCoreActivity() {
    intent = Intent(this@SetupActivity, CoreActivity::class.java)
    if (uriShareSheet != null) {
        intent.putExtra("SHARE_SHEET", uriShareSheet) // Uri from the
other app given via the ShareSheet
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
    }
    intent.putExtra("USER", user)
    this@SetupActivity.startActivity(intent)
    finish()
}
```

Nu kan er in de fragment wat met het bestand worden gedaan. Zo wordt er eerst een kopie gemaakt van het bestand naar de cache. Daarna wordt er gevraagd aan de gebruiker wat er met het bestand moet gebeuren (verborgen of niet). Op basis van dit antwoord wordt het bestand geüpload naar de cloud en het document (firestore) van de gebruiker geüpdate.

```
// Here we get the intent from the SetupActivity
if (activity?.intent?.extras?.get("SHARE_SHEET") != null) {
    receivedUri = activity?.intent?.extras?.get("SHARE_SHEET") as Uri
    makeCachedFileFromUri()
    // Then prompt the user where they want to save the file
    showDialog(
        R.drawable.ic_baseline_upload,
        R.string.txt_dialog_upload_file,
        getString(R.string.txt_dialog_upload_file_explain),
        "Hidden folder",
        "normal folder"
    )
}
```

Versturen van bestanden naar een andere app

Als een bestand eenmaal in de app en dus de cloud staat kan het bestand ook weer worden gedeeld naar een andere app. Hiervoor wordt het bestand eerst gedownload naar een tijdelijk bestand. (`createTempFile()` & `temFile.storageReference.getFile(fileTemp)`)

```
suspend fun shareFileToOther(itemFile: ItemFile, view: View?, context:
```

```
Context?) {
    val snackBar = view?.let {
showSnackBar(context!!.getString(R.string.txt_toast_file_downloading),
it, context) }
    try {
        withContext(Dispatchers.IO) {
            val fileTemp: java.io.File? = runCatching {
                createTempFile(itemFile.name, itemFile.fileType)
            }.getOrNull()
            withTimeout(7_500) {
                if (fileTemp != null) {

itemFile.storageReference.getFile(fileTemp).addOnCompleteListener {
                        if (it.isSuccessful) {
                            if (context != null) {
                                snackBar?.dismiss()
                            }
                        }
                    }
                }
            }
        }
    } catch (e: Exception) {
        throw FirebaseExceptionError("Unable to download file to cash:
${e.localizedMessage}")
    }
}
```

Nu het bestand op de telefoon zelf staat kan deze weer via een `Intent` doorgestuurd worden naar een andere app.
(`Intent.ACTION_SEND`)
In dit voorbeeld wordt de ShareSheet weer geopend en kan de gebruiker aangeven naar welke app hij / zij het bestand wil sturen.

```
val content = FileProvider.getUriForFile(
Objects.requireNonNull(context),
    BuildConfig.APPLICATION_ID + ".provider", fileTemp)
val intent = Intent()
intent.action = Intent.ACTION_SEND
intent.type = "application/octet-stream"
intent.putExtra(Intent.EXTRA_STREAM, content)
intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
startActivity(context, createChooser(intent, "Share your file"), null)
```

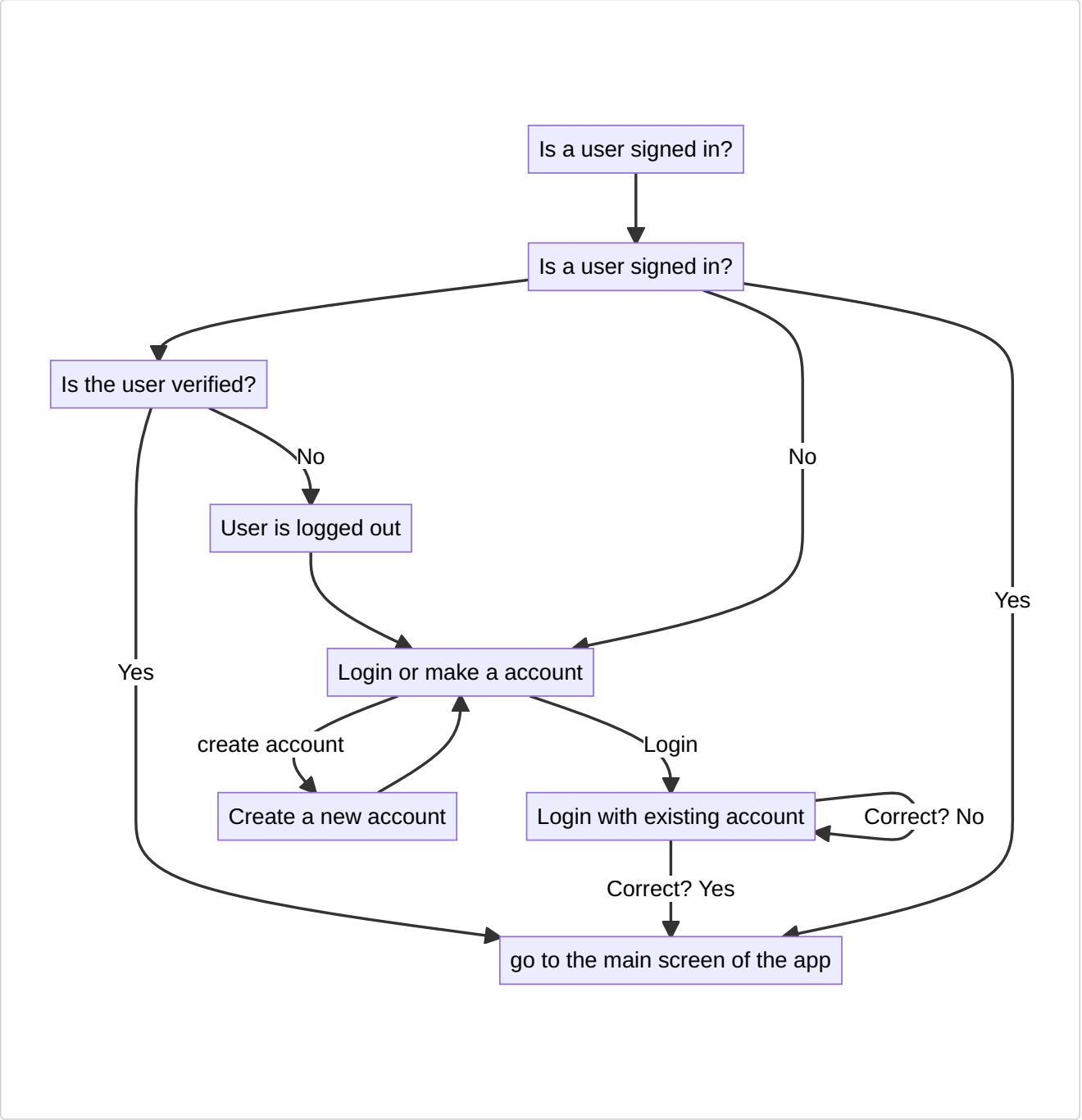
De `Intent()` word verder nog meer gebruikt om het bestand gewoon te kunnen openen:
(`Intent.ACTION_VIEW`)

```
val intent = Intent()
```

```
intent.action = Intent.ACTION_VIEW
intent.setDataAndType(content, mime)
intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
startActivity(context, intent, null)
```

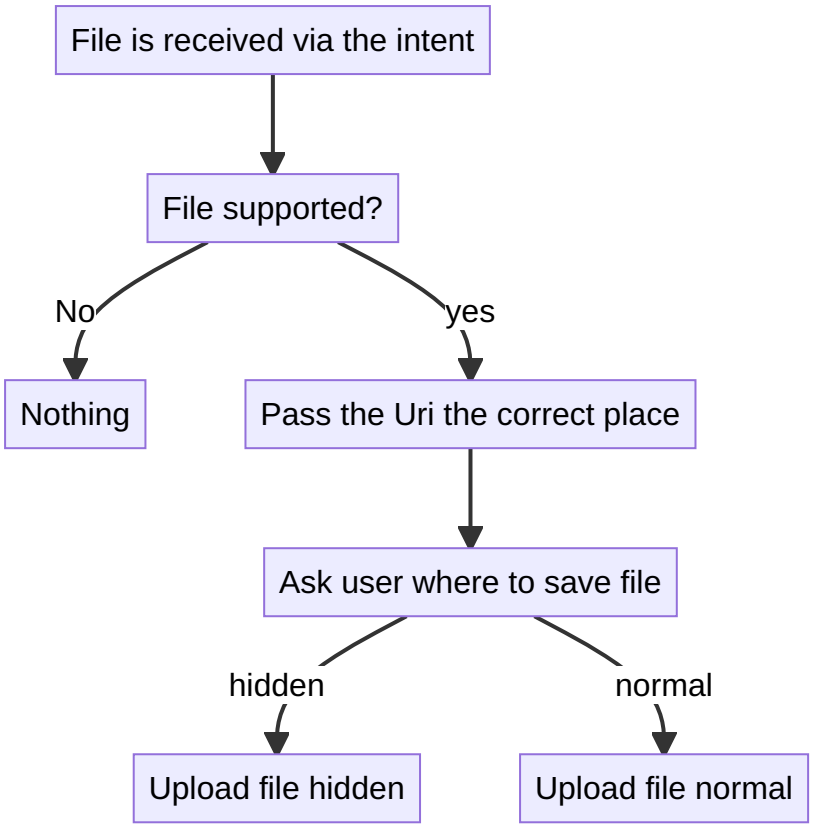
🔃 Flow binnen de applicatie

Openen app



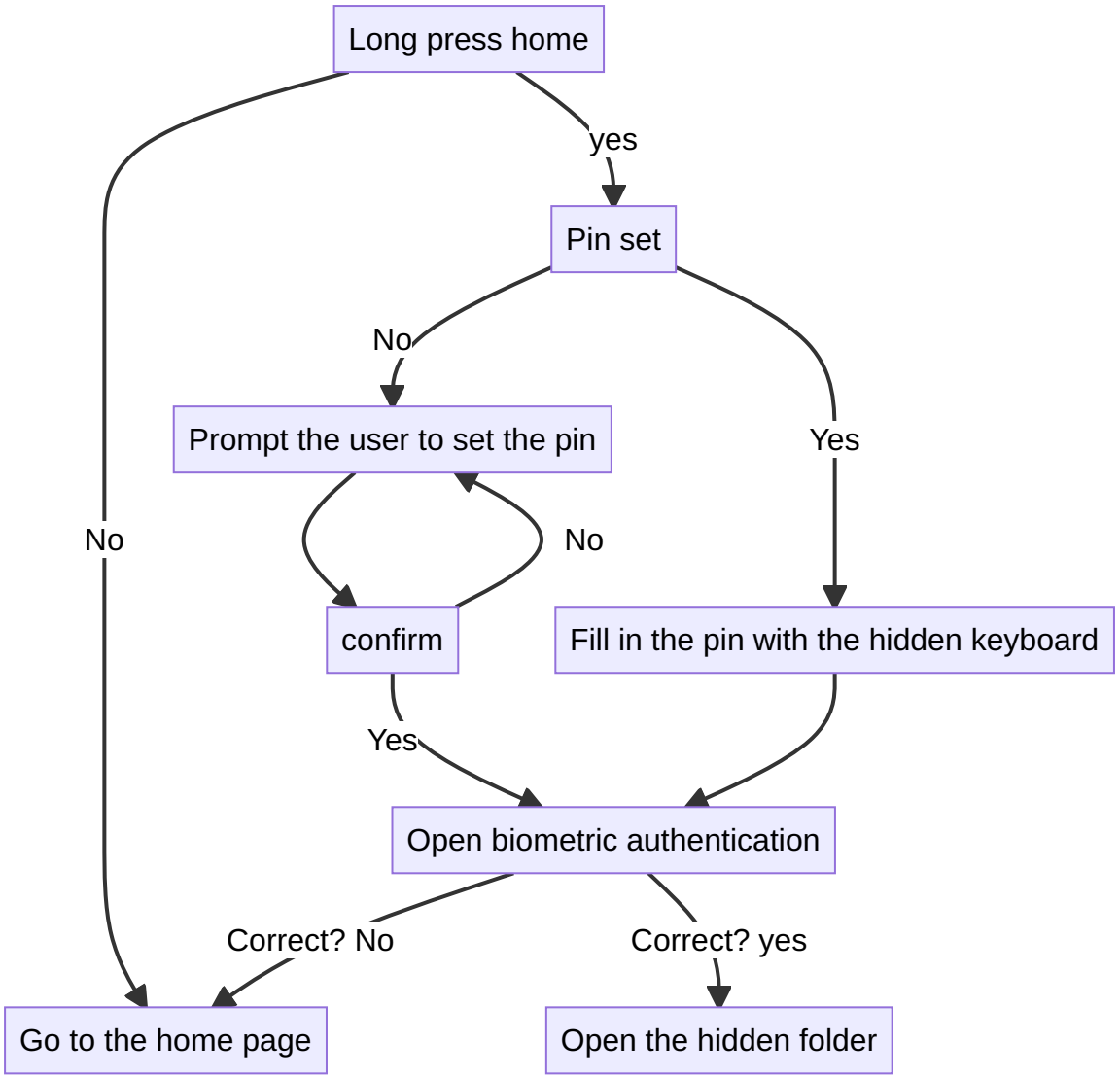
ontvangen bestand

Zie hier de flow hoe de app de `Intent` van een andere app af handelt.



bij verborgen map komen

Zie hier de flow om bij de verborgen map te komen.



📦 File structure

Het project is in tweeën gedeeld, zo is er de "Setup", en de "Core".

`Setup` De setup is verantwoordelijk voor het controleren van de gegevens van het account, op het moment dat de app start, gaat de setup kijken of een gebruiker is ingelogd, zo niet dan blijft de app op de `SetupActivity`. Zo kan een gebruiker hier ook inloggen en een account aanmaken.

`Core` als de `SetupActivity` heeft goedgekeurd om verde te gaan, zal deze een `Intent` starten om naar de `CoreActivity` te gaan. In de `CoreActivity` gebeurt alle het werk van app.

```
private fun launchCoreActivity() {
    intent = Intent(this@SetupActivity, CoreActivity::class.java)
    ...
    this@SetupActivity.startActivity(intent)
    finish()
}
```

overview

```
📦atos
 ┗ 📂msafe
 ┃ ┣ 📂core                          # All code for the "core" of the app
 ┃ ┃ ┣ 📂adapter                     # adapter for the recyclerViewer
 ┃ ┃ ┃ ┗ 📜FileAdapter.kt
 ┃ ┃ ┣ 📂ui                          # All fragments within the "core"
 ┃ ┃ ┃ ┣ 📜AccountFragment.kt
 ┃ ┃ ┃ ┣ 📜HomeFragment.kt
 ┃ ┃ ┃ ┗ 📜SavedFragment.kt
 ┃ ┃ ┣ 📂viewmodel                   # Viewmodel to go with the fragments
of the "Core"
 ┃ ┃ ┃ ┣ 📜AccountViewModel.kt
 ┃ ┃ ┃ ┗ 📜CoreViewModel.kt
 ┃ ┃ ┣ 📜CoreActivity.kt             # Activity for the "Core"
 ┃ ┃ ┗ 📜ManagePermission.kt         # Class to handle the permissions
needed for the app
 ┃ ┣ 📂model                         # Data classes
 ┃ ┃ ┣ 📜ItemFile.kt
 ┃ ┃ ┣ 📜NewUser.kt
 ┃ ┃ ┗ 📜User.kt
 ┃ ┣ 📂repository
 ┃ ┃ ┗ 📜FirebaseRepository.kt       # Firebase repository to make API
calls to it
 ┃ ┣ 📂setup                         # All the code for the "Setup" of the
app
 ┃ ┃ ┣ 📂ui                          # All fragments within the "Setup"
 ┃ ┃ ┃ ┣ 📜BootFragment.kt
 ┃ ┃ ┃ ┣ 📜LoginFragment.kt
 ┃ ┃ ┃ ┣ 📜SetupFragment.kt
 ┃ ┃ ┃ ┗ 📜SignupFragment.kt
 ┃ ┃ ┣ 📂viewmodel                   # Viewmodel to go with the fragments
of the "Setup"
 ┃ ┃ ┃ ┣ 📜LoginViewModel.kt
 ┃ ┃ ┃ ┗ 📜SignupViewModel.kt
 ┃ ┃ ┗ 📜SetupActivity.kt            # Activity for the "Setup"
 ┃ ┗ 📂shared
 ┃ ┃ ┗ 📜Extensions.kt               # File that contains all the
extensions
```

Bekende problemen

- Het downloaden van een bestand kan soms blijven hangen.
- Geen dark mode support

Vervolg stappen

- Het hernoemen van bestanden
  Een bestand hernoemen gaat niet helemaal zoals verwacht. Het bestand moet namelijk eerst

worden gedownload, dan moet de naam van dit bestand worden aangepast. En dan weer worden geüpload. Dit zijn nogal wat stappen die fout kunnen gaan, hier moet een robuuste structure voor worden opgezet.

- Toevoegen van meer bevelling
  De app bied verder geen encryptie of iets dergelijks tussen de front en back-end.
- Betaalde opties
  Een optie toevoegen om gebruikers te laten betalen voor meer opslag.