Sebastiaan Maes

[E-mailadres]

Abstract

[Trek de aandacht van uw lezer met een interessante samenvatting. Dit is meestal een kort overzicht van het document.   
Wanneer u uw inhoud wilt toevoegen, klikt u hier en begint u te typen.]

Project mobiele apps

Savingstracker app

Inhoud

[Opdrachtomschrijving 2](#_Toc154435472)

[Doel van de App 2](#_Toc154435473)

[Programmeertools 2](#_Toc154435474)

[Android Studio 2](#_Toc154435475)

[Kotlin 2](#_Toc154435476)

[Opbouw van de app 3](#_Toc154435477)

[Naamconventie bestanden 3](#_Toc154435478)

[Login scherm 3](#_Toc154435479)

[MVVM structuur 4](#_Toc154435480)

[Functies direct aanroepen vanuit XML 4](#_Toc154435481)

[Zichtrekeningen-tab 5](#_Toc154435482)

[Opbouw 5](#_Toc154435483)

[Verschillende layouts in recyclerview 8](#_Toc154435484)

[Spaarrekeningen-tab 10](#_Toc154435485)

[Roomdatabase 10](#_Toc154435486)

[Relaties tussen tabellen 11](#_Toc154435487)

[Investeringen-tab 12](#_Toc154435488)

[Updaten van de assets 12](#_Toc154435489)

[Berekening van de data voor de grafieken 13](#_Toc154435490)

[Extralegale voordelen-tab 13](#_Toc154435491)

[Doorgeven van Data 14](#_Toc154435492)

[Transacties toevoegen 15](#_Toc154435493)

[Conclusie 15](#_Toc154435494)

# Opdrachtomschrijving

Vereisten app project:

* Maak een Android app in Kotlin m.b.v. Android Studio die gebruik maakt van MVVM.
* De user interface mag opgebouwd worden met views of met composables. Maak minstens gebruik van een database (sqlite, Firebase, Azure, ...) of een api.
* Probeer verder zoveel mogelijk van Android te ontdekken. Je zou minstens 42u moeten besteden aan de ontwikkeling van je app, de lessen inbegrepen (en best maximaal 50u om op het vlak van tijdsbesteding geen problemen te krijgen met andere projecten) .
* Stuur je opdrachtomschrijving door per mail ten laatse op 27/10.

# Doel van de App

De dag van vandaag zijn jongeren minder en minder bewust van de waarde van geld en de verleiding om het uit te geven is groot, veel jongeren hebben dan ook geen overzicht van hun financiële situatie. Veel jongeren beginnen tegenwoordig ook met investeren of houden zich bezig met het aankopen van crypto. Het doel van de app is dan ook om een volledig overzicht te geven van je huidige financiën, de zichtrekening, spaarrekening, aandelenportefeuille, cryptomunten, ETF’s, maar ook je eco- en maaltijdcheques.

# Programmeertools

In de module Mobiele apps hebben we gebruik gemaakt van Kotlin en de programmeeromgeving Android Studio. Android Studio en Kotlin vormen een krachtige combinatie voor het ontwikkelen van Android-applicaties. Android Studio is de officiële geïntegreerde ontwikkelomgeving (IDE) voor Android, en Kotlin is een moderne programmeertaal die speciaal is ontworpen om naadloos te integreren met Android. Samen bieden ze ontwikkelaars een efficiënte manier om hoogwaardige Android-apps te bouwen. We bespreken deze nog kort apart.

## Android Studio

Android Studio is de officiële geïntegreerde ontwikkelomgeving (IDE) die wordt gebruikt voor het bouwen van Android-applicaties. Het is ontwikkeld door Google en biedt een uitgebreid scala aan functies om het gehele ontwikkelproces te ondersteunen zoals nieuwste Android SDK-functionaliteit, mogelijkheid om libraries te implementeren en te gebruiken, IntelliSense, emulator, en mogelijkheid om de app rechtstreeks naar de telefoon te downloaden inclusief mogelijkheid tot debuggen.

## Kotlin

Kotlin, aan de andere kant, is een moderne programmeertaal die concisiteit, leesbaarheid en interoperabiliteit met bestaande Java-code bevordert. Het biedt functies zoals null safety en uitbreidingsfuncties, waardoor de ontwikkeling efficiënter en minder foutgevoelig wordt. Kotlin is de voorkeurstaal geworden voor Android-ontwikkeling en heeft de traditionele Java-taal in veel projecten vervangen.

# Opbouw van de app

De app is opgebouwd startend vanuit een login scherm, hier zal de gebruiker eerst een pincode moeten invoeren vooraleer men de app en data kan zien, we willen immers niet iedereen toegang geven tot het financieel overzicht van deze persoon.

Daarna komen we in de applicatie terecht waar we onderaan centraal een navigatiebar hebben. Deze bestaat uit een tab voor de zichtrekeningen, spaarrekeningen, investeringen en de extralegale voordelen zoals eco- en maaltijdcheques. We zullen van deze structuur gebruik maken om de app stap per stap uit te leggen en enkele roadblocks die ik ben tegen gekomen uit te leggen.

## Naamconventie bestanden

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Door de grootte van de app en het ontbreken van de mogelijkheid om snel en efficiënt mappen aan te maken in Kotlin heb ik gekozen voor een andere benadering. Eerder is er al besproken dat de app uit vier subtabs bestaat, verder hebben we nog de bestanden van de database en bestanden die voor iedere tab gelden. Door voor elke file de naam te schrijven waartoe het bestand behoort krijgen we een gestructureerd en overzichtelijk project

Er moet er wel vermeld worden dat het wel mogelijk is, dit kunnen we doen door in de gradle file van het project alle submappen te definiëren. Het probleem is dat dit niet mogelijk is bij layout files en ook in sommige situaties leidt tot bestanden die niet herkend worden en dus aanleiding geven tot compilatie-errors.

# Login scherm

Afbeelding met tekst, schermopname, cirkel, Kleurrijkheid

Automatisch gegenereerde beschrijving

Het login scherm is het eerste scherm dat de gebruiker te zien krijgt bij het openen van de app. De gebruiker toetst de code in, elke keer als er een cijfer wordt ingevoerd zal er een nieuw bolletje oplichten. Bij het drukken op “OK” zal de code gecontroleerd worden aan de hand van de code die in de database is opgeslagen. Is de code fout dan zal er een melding “incorrect code, please try again” verschijnen.

Wanneer we de opbouw gaan bekijken zien we meteen bepaalde elementen terugkomen, een user interface met buttons, functies die worden uitgevoerd bij het drukken op toetsen, ophalen van data uit de database. Om dit overzichtelijk te houden maken we gebruik van MVVM die we in de volgende paragraaf zullen toelichten.

## MVVM structuur

MVVM staat voor Model-View-ViewModel en is een architectuurpatroon dat vaak wordt gebruikt bij het ontwikkelen van Android-applicaties. Het doel van MVVM is om de code overzichtelijk te organiseren, verschillende verantwoordelijkheden van bepaalde stukken code te scheiden en de testbaarheid van de code te vergroten. Hier een schematisch overzicht van MVVM in Android:

Afbeelding met tekst, Lettertype, logo, Graphics

Automatisch gegenereerde beschrijving

* Model: bevat de gegevens en bedrijfslogica van de applicatie. Het bevat de code die toegang verstrekt tot databases of om netwerkservices aan te spreken.
* View: Bevat de code voor de UI (gebruikersinterface) van de applicatie. Het observeert wijzigingen in het ViewModel en werkt de gebruikersinterface dienovereenkomstig bij.
* ViewModel: Fungeert als tussenliggende laag tussen het Model en de View. Het bevat de presentatielogica en bewaart de toestand van de UI. Het observeert verder gegevenswijzigingen in het Model en zal zorgen dat de View wordt geactualiseerd indien nodig en andersom.

## Functies direct aanroepen vanuit XML

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Daarnet kreeg u een korte introductie tot MVVM, om aan een knop een bepaalde functionaliteit te hangen zullen we normaal in het ViewModel een “setOnClickListener” definiëren. Deze functie zal uitgevoerd worden wanneer er op de knop wordt geduwd. Echter hebben we hier 10 knoppen (0 t.e.m. 9) die allemaal naar dezelfde functie verwijzen met een verschillend argument. Dit zou een enorme overhead creëren in het viewModel, we zouden dan immers 10 verwijzingen naar dezelfde functie krijgen met een verschillend argument. Daardoor hebben gekozen om deze rechtstreeks functies in de XML op te roepen.

# Zichtrekeningen-tab

Afbeelding met tekst, schermopname, multimedia, Mobiele telefoon

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, multimedia, Mobiele telefoon

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Mobiele telefoon, Mobiel apparaat

Automatisch gegenereerde beschrijving

In dit tabblad krijgen we een overzicht van de zichtrekeningen en kredietkaarten. Wanneer we op de drie rondjes klikken wordt het menu uitgeklapt en krijgen we de transacties te zien. Verder heben we de mogelijkheid om naar rechts te swipen en dan krijgen we een andere kaart te zien, tot slot kunnen we eventueel een nieuwe kaart en transacties toevoegen.

De volledige app maakt opnieuw gebruik van het MVVM model dat we in het vorige punt hebben uitgelegd. Om niet opnieuw in herhaling te vallen is er daarom gekozen om telkens bepaalde onderdelen uit te diepen die dit tabblad uniek maken. We behandelen hier de opbouw, maar anderen zaken zoals de transacties komen later nog aan bod.

## Opbouw

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

Vaak zien apps er vrij eenvoudig uit qua design maar zit er veel meer achter dan op het eerste zicht lijkt. Door de beperkte grootte van het scherm moet de app compact en overzichtelijk zijn maar bevat toch vaak enorm veel functionaliteit.

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijvingAfbeelding met tekst, Mobiele telefoon, schermopname, Mobiel apparaat

Automatisch gegenereerde beschrijving

De opbouw begint allemaal vanuit een fragment dat uit de navigatie wordt opgeroepen. Dit fragment bevat de layout die u hier ziet, maar erachter zit nog het model en een viewmodel.

De UI bevat basis elementen zoals de titel en de velden om het totaal te tonen. Verder ziet u de drie bolletjes, dit is een afbeelding waaraan een functie gelinkt is. Deze functie zal bij het klikken de illusie wekken dat het witte menu naar boven schuift (rechtste foto). Deze functionaliteit hebben we in het viewmodel gestoken.

Verder kan u nog de grafiek met het verloop zien, dit is een library die we hebben gebruikt en met data vullen vanuit het model. Tot slot zit er nog een viewpager element in, dit gaan we nu verder bespreken.

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Een viewpager is een element dat ervoor zorgt dat we naar rechts en links kunnen vegen met als gevolg dat er dan andere elementen getoond worden (zie bovenste foto’s). We kunnen de viewpager zien als het omslaan van een fotoalbum naar een andere bladzijde.

De viewpager is een element die in de layout zit zoals hierboven besproken, de data die we meegeven (vanuit het model) zal verwerkt worden in de Viewpager adapter. In het getoonde voorbeeld is er dus een array van twee zichtrekeningen mee gegeven en zullen er bijgevolg ook twee kaarten gecreëerd worden;

Afbeelding met tekst, schermopname, software, Besturingssysteem

Automatisch gegenereerde beschrijving

In de viewpager adapter een meegegeven array van data worden doorlopen en per element bepaalde layouts worden gecreërd. De layout die zal worden gecreëerd is hier op de foto weergeven. De layout bevat elementen die de kaartanimatie tonen, de “transaction” titel, en de recyclerview als element.

Afbeelding met tekst, schermopname, software, diagram

Automatisch gegenereerde beschrijving

De recyclerview is een element die wordt gebruikt voor het weergeven van een grote set gegevens die kunnen veranderen. Vaak worden deze gebruikt om efficiënt lange lijsten met gegevens weer te geven, een transactielijst is hier één van. De recyclerview zit ook in de xml layout verwerkt.

Afbeelding met schermopname, tekst, software, ontwerp

Automatisch gegenereerde beschrijving

Voor de Recyclerview hebben we ook weer een adapter nodig waaraan we de data meegeven. In de recyclerview zal de array van data worden doorlopen en per element zal een transaction item worden geïntroduceerd in de view. We hebben de mogelijkheid om verschillende items te introduceren en dit gaan we in de volgende paragraaf bespreken.

## Verschillende layouts in recyclerview

Afbeelding met schermopname, tekst, software, ontwerp

Automatisch gegenereerde beschrijving

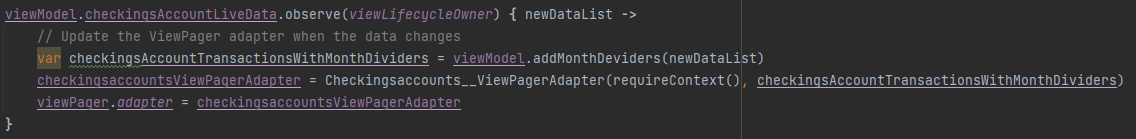
Afbeelding met tekst, schermopname, Lettertype, Rechthoek

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

In deze layout zien we dat er een recyclerview is gebruikt waarin de transacties worden getoond. Echter zien we dat hier twee types van layout inzitten, namelijk één voor de maanden van de transacties te tonen en de andere zijn de effectieve transacties.



Het begint allemaal al vanbuiten de lijst. We hebben immers een lijst met transacties opgehaald uit de database maar deze zijn niet gesorteerd en er zijn geen dividers beschikbaar die de maanden van de transacties weergeven. Het is niet mogelijk om deze toe te voegen in de adapter zelf (hier komen we nog op terug waarom), en daarom roepen we eerst een functie op vanuit het viewmodel om deze toe te voegen



Het volgende is dat we de adapter klasse gaan uitvoeren en de transacties als lijst binnenkrijgen. In een normale situatie wordt de lijst dan doorlopen en per item in de lijst wordt een layout voor aangemaakt die dan in d recyclerview wordt weergeven. We werken echter met dividers waardoor de werken iets anders wordt.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Als eerste zal de het item doorlopen worden en zal er bepaald worden van welk type het item is, dit kan een transactie zijn of een divider, dit type geven we terug.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Aan de hand van het type zal dan een andere layout worden geïntroduceerd.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Daarna zal de overeenkomstige ViewHolder worden uitgevoerd om de elementen in de layout op te halende en tot slot wordt de data van het element in de array aan de UI gekoppeld.

# Spaarrekeningen-tab

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, multimedia, ontwerp

Automatisch gegenereerde beschrijving

In dit tabblad krijgen we een grafisch verloop van het spaarsaldo, verder kunnen we het saldo zien per spaarrekening. Wanneer men op de spaarrekening klikt zal men een overzicht krijgen van de transacties die er op deze rekening verricht zijn.

## Roomdatabase

Het hoofddoel van de app is om alle transacties en data op te slaan en hier een visuele representatie van te maken. Om deze data op te slaan maken we gebruik van een lokale database op de gsm.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

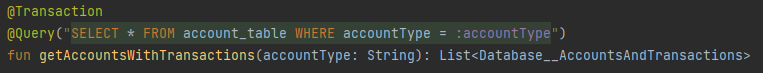
Bij het opstarten van de app zullen we de database klasse als eerst uitvoeren. We gaan hierin kijken of er reeds een instantie op het toestel aanwezig is, indien dit niet het geval is zal een instantie worden aangemaakt. Na het aanmaken van de database zal de database gevuld worden met “mockdata” om een overzicht te geven van de mogelijkheden van de app.

## Relaties tussen tabellen

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Zoals boven beschreven hebben we verschillende spaarrekeningen. Een spaarrekening bevat verschillende gegevens zoals de naam, het saldo, rekeningnummer (zie afbeelding net onder de hoofdtitel). Maar ook bevat deze alle transacties bijhorend bij de rekening, dit is echter een andere tabel in de database. Wanneer we in een grafiek het verloop willen laten zien hebben we zowel de huidige balans als de transacties nodig (komt verder in de tekst nog aan bod). Om deze data te verkrijgen zouden we twee queries moeten uitvoeren. Maar dit kan ook op een gemakkelijker manier, namelijk door relaties te leggen.



Wanneer we met relaties gaan werken schrijven we eerst een querry om één variabele in te vullen, zoals hier bijvoorbeeld de variabele “account”. Daarna definiëren we in de data klasse een relatie, dit is een veldnaam die in beide tabellen voorkomt. Vervolgens weer Room automatisch dat er nog een tabel moet worden opgevraagd van het opgegeven type met als relatie de opgegeven veldnaam, de room library zal deze dan automatisch ophalen en invullen in de variable.

# Investeringen-tab

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

In dit tabblad kan de gebruiker zijn portfolio opvolgen, met portfolio bedoelen we aandelen, cryptomunten, ETF’s maar ook obligaties. Eerst zal er globaal een overzicht worden getoond van de evolutie, daarna krijgt men dit overzicht per categorie, en als laatste krijgt men een overzicht per asset individueel. Om deze evolutie weer te geven hebben we de huidige actuele prijs nodig en de prijs van de afgelopen dagen, deze vragen we op aan de hand van een API.

## Updaten van de assets

Een bank of zichtrekening veranderd afhankelijk van het transacties, deze moeten keer op keer worden ingegeven. Wanneer we echter aandelen of crypto in bezit hebben veranderd de waarde hiervan dagelijks. Hebben we een normaal tot uitgebreid portfolio kunnen hier al rap 25 aandelen in zitten die we om de zoveel tijd moeten gaan updaten qua waarde, daarom hebben we dan ook gebruik gemaakt van een API die deze data ophaalt.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Hierboven is een functie te zien waar we voor elke asset het GET request vormen daarna we naar de API toesturen. We krijgen dan een resultaat terug in JSON formaat dat we zullen parsen in een data klasse.

## Berekening van de data voor de grafieken

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Om snel een overzicht te krijgen over je huidige situatie is het gemakkelijk om meteen te huidige waarde, maar ook een grafiek te zien wat er de afgelopen 30 dagen is gebeurd. Op die manier krijgt de gebruiker bijvoorbeeld een overzicht wat de beurs in de afgelopen dagen heeft gedaan.

Bij de transacties is dit iets simpeler dan bij de investeringen. Bij de transacties houden we immers de laatste waarde bij en we tellen hierbij de transacties van afgelopen 30 dagen op of af, bij de investeringen gaan we iets anders te werk.

Uit de database halen we alle transacties die we dan met een functie zullen filteren tot de transacties van de afgelopen 30 dagen, samen met en het huidige aantal van de asset dat we bezitten. Op die manier kunnen we over de afgelopen 30 dagen berekenen wat het aantal was per dag. Van de API halen we verder de prijs per dag die wen tenslotte zullen vermenigvuldigen met het aantal stuks per dag. Op deze manier kunnen we grafisch de evolutie van het aandeel laten zien. Willen we een globaal overzicht laten zien dan zullen we dit voor alle assets doen zullen we nog een extra functie toevoegen. We zullen hier een lijst doorlopen met de evolutie per asset en hier vervolgens een totaalsom per dag van maken, deze lijst zullen we dan terug geven en tonen.

# Extralegale voordelen-tab

Afbeelding met tekst, schermopname, Lettertype, Mobiele telefoon

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, multimedia, gadget

Automatisch gegenereerde beschrijving

Dit tab is quasi hetzelfde als het zichtrekeningen tabblad. We maken hier namelijk gebruik van dezelfde structuur maar in plaats van de zichtrekeningen tonen we hier de kaarten voor maaltijd- en ecocheques. We hebben ook hier weer de mogelijkheid om transacties of kaarten toe te voegen.

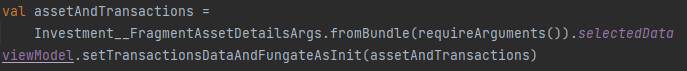
## Doorgeven van Data

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

In de app staat de data die we uit de database halen centraal, voor elk tabblad hebben we data nodig uit de database die we dan zullen verwerken, updaten, nieuwe data toevoegen, of deze grafisch weergeven. De data die we ophalen is aanzienlijk en wanneer we deze data elke keer gaan ophalen voor elk tab zou dit de app enorm vertragen. Daarom maken we gebruik van argumenten om data door te geven van het ene naar het andere fragment.

Bij het navigeren zien we dan ook dat we de data als argument meegeven. Om dit te kunnen doen is het noodzakelijk om in de navigation het type alsook een naam op te geven.



Tenslotte halen we deze data in het ViewModel van het nieuwe fragment ook weer op om deze door te geven aan het model, deze staat er immers voor in om deze te verwerken.

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Een belangrijke om te onthouden is het volgende: wanneer we in het viewmodel functies willen uitvoeren maken we gebruik van de functie INIT. Wanneer we echter argumenten doorgeven en de INIT functie wordt uitgevoerd is deze data uit het argument nog NIET beschikbaar. Daarom roepen we vauit het Viewmodel een functie uit van het model, en vanuit hier gaan we alle functies vervolgens aanroepen

## Transacties toevoegen

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Tenslotte willen we ook de mogelijkheid aanbieden om nieuwe transacties, kaarten en assets toe te voegen, hiervoor maken we gebruik van een nieuw fragment. Dit fragment zal de data view binding binnenhalen, deze input data contoleren op type, en indien deze niet voldoet een foutmelding genereren.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Is de data correct dan zal deze vervolgens de functie in het model uitvoeren om de data in de database te steken. We maken dan gebruik van het DAO model op de bijhorende “insert” functie uit te voeren. Wat we daarna doen is een booleaanse variabele hoog maken en deze observeren we in het viewmodel. Wanneer deze true is zullen we terug navigeren naar de vorige pagina en deze variabele resetten, dit doen we zodat de volgende keer dat we een transactie willen uitvoeren geen loop ontstaat die ons meteen terugverwijst naar de pagina waar we vandaan kwamen.

# Conclusie

Bij aanvang van deze module had ik er in eerste instantie niet zoveel zin in eerlijk gezegd. We hadden immers al een app gemaakt, ik had geen inspiratie voor een nieuwe app en wou niet dezelfde maken, maar vond Android studio ook een pak minder overzichtelijk.

Ik heb in eerste instantie voor een vrij eenvoudige app gekozen, maar dit uiteindelijk ook meer een hobbyproject geworden waar een ik een groot deel van mijn vrije tijd aan heb gespendeerd. Vorig jaar had ik reeds aangehaald dat ik de volgende app ook kleiner en meer in detail zou afwerken (daar heb ik mij dus duidelijk niet aan gehouden). Zoals beloofd wou ik de app ook goed afwerken en naarmate dat de scope van het project groeide voelde ik wel de noodzaak om deze zelf vooropgestelde doelstelling te halen.

Ik blijf er echter wel bij dat ik liever in .NET MAUI programmeer. Dit is een persoonlijke voorkeur omdat het voor mij een stuk overzichtelijker is, makkelijker om een structuur aan te brengen, ondersteuning van github copilot, maar ook omdat ik liever in c# programmeer dan in Kotlin. Ik heb wel gemerkt dat Android studio een stuk minder foutgevoelig is dan .NET MAUI en dit vertraagde ook wel het project in .NET MAUI. Beiden zijn echter omgevingen met hun voor- en nadelen waar fantastische apps in te bouwen zijn