Stappenteller, Eindrapport, Groep E

Arthur Saelens – Linde Roggeman – Maxim Ponomariov – Staf Vanhauwaert

Begeleiders: Vincent Bracke, Maarten de Mildt

23/12/2022

**Inhoudsopgave**

[**1.** **ACCELEROMETER** 5](#_Toc122477413)

[**1.1.** **Versnelling** 5](#_Toc122477414)

[**1.2.** **Verschillende snelheden** 5](#_Toc122477415)

[**1.3.** **De assen** 5](#_Toc122477416)

[**1.4.** **De zin van de assen** 6](#_Toc122477418)

[**2.** **VERSCHILLENDE DETECTORS** 6](#_Toc122477420)

[**2.1.** **1D signaal** 6](#_Toc122477421)

[**2.2.** **Ruis filteren** 6](#_Toc122477424)

[**2.3.** **Detectoren** 7](#_Toc122477427)

[**2.4.** **Algoritme** 7](#_Toc122477428)

[**3.** **DETECTOR IN JAVA PROGRAMMEREN** 8](#_Toc122477431)

[**3.1.** **Offline analysis** 8](#_Toc122477432)

[**3.2.** **Definitieve filter** 9](#_Toc122477433)

[**3.3.** **Live data** 9](#_Toc122477436)

[**3.4.** **Complexe drempelwaarde** 9](#_Toc122477437)

[**4.** **INTENTS** 10](#_Toc122477438)

[**5.** **APPLICATIE** 10](#_Toc122477439)

[**5.1.** **Mock-up** 10](#_Toc122477440)

[**5.2.** **Material design** 11](#_Toc122477442)

[**5.3.** **Material Design toegepast** 11](#_Toc122477443)

[**5.4.** **Interactie met de gebruiker** 11](#_Toc122477445)

**Lijst met illustraties**

[**Figuur 1: De assen van de accelerometer ten opzicht van de smartphone 4**](#_Toc122477683)

[**Figuur 2: De zin van de assen met de zin 5**](#_Toc122477684)

[**Grafiek 1: ongefilterd 1D signaal 5**](#_Toc122477685)

[**Grafiek 2: Metingen van het experiment 6**](#_Toc122477686)

[**Flowchart 1: Algoritme om stap te registreren 7**](#_Toc122477687)

[**Figuur 3: Mogelijke situatie aangeduid op grafiek met drempelwaarde 7**](#_Toc122477688)

[**Grafiek 3: Grafiek van de versnelling na het toepassen van de definitieve filter 8**](#_Toc122477689)

[**Figuur 2: lay-out app 9**](#_Toc122477690)

**Verklarende woordenlijst**

**Accelerometer** = een sensor die de dynamische trillingen van een fysiek object omzet naar een spanning.

**Mock-up =** een mock-up is een tijdens de [ontwerp](https://nl.wikipedia.org/wiki/Ontwerp)- of [productiefase](https://nl.wikipedia.org/wiki/Productieproces) op [schaal](https://nl.wikipedia.org/wiki/Schaal_(verhouding)) of op ware grootte gemaakt [model](https://nl.wikipedia.org/wiki/Schaalmodel) van een ontwerp of product.

**Android Studio =** de officiële integrated development environment (IDE) om applicaties te ontwikkelen voor Android (besturingssysteem).

**User Interface** = ook wel gebruiksomgeving genoemd, is de visuele vorm waarin de gebruiker het product of de dienst te zien krijgt, en de tools waarmee hij er gebruik van maakt.

**Samenvatting**



Doorheen dit rapport zullen de verschillende stappen en experimenten beschreven worden die uitgevoerd moesten worden om een werkende stappenteller te bekomen. Eerst moest er een manier gevonden worden om stappen waar te nemen met de smartphone. Dit kon worden gerealiseerd door gebruik te maken van de accelerometer die zich in de telefoon bevindt. Aan de hand van enkele simpele experimenten werd duidelijk hoe deze sensor werkte. Vervolgens werd er een filter ontwikkeld waardoor de ruis geëlimineerd werd en er dus zo uit de data van de accelerometer stappen konden worden waargenomen. Hierna kwam het ontwikkelen van de effectieve app. Hier moesten twee zaken gerealiseerd worden. De ontwikkelde filter moest geïmplementeerd worden maar er moest ook een werkende app gevormd worden. Deze twee zaken konden worden gedaan met het programma Android Studio. Met dit programma werden drie schermen ontwikkeld binnen de app. Het eerste scherm is het ‘home’ scherm waar onder andere de stappen worden weergegeven, op het ‘profile’ scherm kunnen de gegevens van de gebruiker worden ingegeven zoals geslacht en leeftijd. Tot slot is er dan ook het ‘settings’ scherm waar de gebruiker bijvoorbeeld zijn persoonlijk doel kan wijzigen.

# **ACCELEROMETER**

Om stappen waar te nemen met de smartphone werd gebruik gemaakt van een instrument dat bewegingen kan detecteren en meten. Dit instrument zat al in de smartphone ingebouwd en wordt een accelerometer genoemd. Maar voor er met deze accelerometer gewerkt kon worden, moest eerst begrepen worden hoe deze werkt en wat de opties en mogelijkheden hiermee zijn.

## **Versnelling**

De accelerometer is een sensor die aan de hand van trillingen de versnelling van de telefoon meet. Er zal echter rekening moeten gehouden worden dat het gaat om de versnelling ten opzicht van een vallend object. Dit komt doordat de accelerometer krachten detecteert en aangezien de zwaartekracht op aarde altijd actief is, zal de telefoon altijd een extra versnelling waarnemen die gelijk is aan de valversnelling.

## Getting Raw Accelerometer Events | Apple Developer Documentation**Verschillende snelheden**



De eerste mogelijkheid van de accelerometer waarmee geëxperimenteerd werd, was de snelheid van de metingen. Er zijn namelijk vier verschillende opties die kunnen worden gebruikt om te bepalen hoe vaak er een meting gedaan moet worden. Deze vier opties zijn ‘normal’, ‘UI’, ‘game’ en ‘fastest’. Om te weten te komen hoe snel de metingen van elke optie uitgevoerd worden, werd een simpel experiment uitgevoerd. Er werd met elke optie 10 seconden gemeten en daarna werd gekeken hoeveel metingen er waren genomen. Hieruit kon worden afgeleid dat ‘normal’ 300 metingen per minuut uitvoert, ‘UI’ 1.000, ‘game’ 3.000’ en ‘fastest’ 6.000. Het is belangrijk om genoeg data te hebben om stappen te meten maar de batterij mag hier ook niet onder lijden. Daarom werd gekozen om vanaf nu met de optie ‘game’ verder te werken want de optie ‘fastest’ zou de batterij te veel belasten.

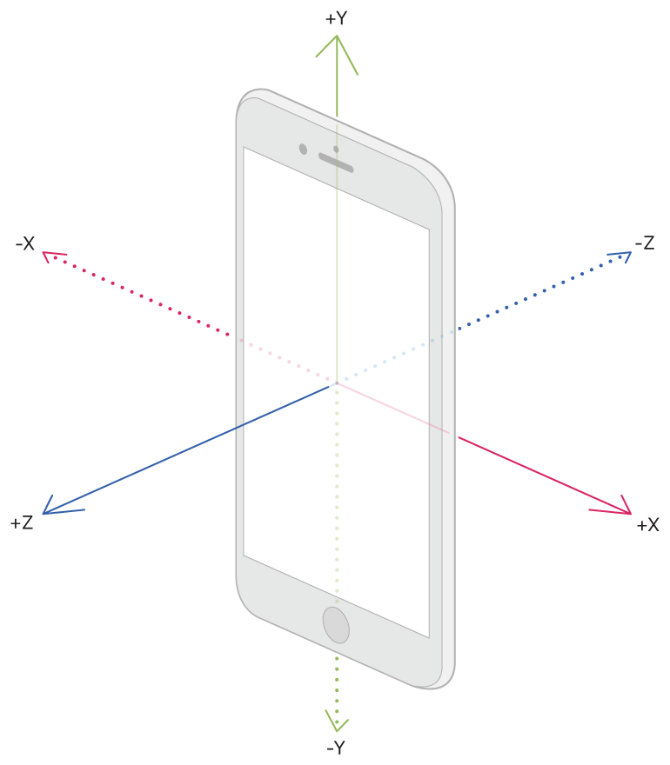


## **De assen**

Het eerste wat opviel toen gekeken werd naar de meetresultaten was dat er drie afzonderlijke kolommen met meetresultaten waren. Dit komt omdat de accelerometer de versnelling op drie assen afzonderlijk meet namelijk de x-,y- en z-as. Nu moest onderzocht worden hoe deze assen georiënteerd liggen in de smartphone om beter te begrijpen hoe de accelerometer werkt. Het eerste experiment dat uitgevoerd was om de telefoon met de achterkant naar onder op tafel te leggen en even te meten. Wanneer deze metingen bekeken werden, werd gezien dat er enkel een grote versnelling met de grote 9,8 m/s2 werd waargenomen op de z-as. Hieruit kan worden afgeleid dat de z-as loodrecht op het scherm van de smartphone staat. Dit experiment werd vervolgens herhaald maar met de smartphone rechtopstaand en op de zijkant. Hieruit kon worden afgeleid dat de assen loodrecht op elkaar staan en zoals op volgende afbeelding georiënteerd zijn.

### Figuur 1: De assen van de accelerometer ten opzicht van de smartphone

## **De zin van de assen**

Aangezien geweten is dat er een versnelling wordt waargenomen ten opzichte van een vallend object kan worden besloten dat de smartphone een versnelling naar boven moet waarnemen wanneer deze stil ligt. Hieruit kan de zin van de assen afgeleid worden. En deze is zoals u op volgende afbeelding ziet.

### Figuur 2: De zin van de assen met de zin

# **VERSCHILLENDE DETECTORS**

Voor het ontwikkelen van een werkende stappenteller moest er geëxperimenteerd worden met verschillende soorten detectors om de stappen zo optimaal mogelijk te kunnen detecteren en tellen.

## **1D signaal**

Voordat de detectors mochten geïmplementeerd worden, moest de data eerst gefilterd worden. De eerste stap was om de gevonden data die in 3D wordt gemeten in een 1D signaal om te zetten. Doordat de x-, y- en z-assen loodrecht op elkaar staan, kan de stelling van Pythagoras toegepast worden om een 1D signaal te ontwikkelen aan de hand van volgende formule waarbij a de totale versnelling van de smartphone is.

Dit signaal is nog niet perfect want de accelerometer meet namelijk de versnelling ten opzicht van een vallend object. Dit betekent dat de valversnelling op elk moment bij het signaal wordt toegevoegd. Dit werd ook experimenteel ondervonden want wanneer de filter toegepast werd op de telefoon die stil lag, werd een versnelling van ongeveer 9.81 m/s2 gemeten. Om ervoor te zorgen dat de effectieve versnelling van de telefoon wordt weergegeven moet er dus nog 9.81 van het 1D signaal worden afgetrokken. Daardoor wordt de volgende formule gevonden voor het definitieve 1D signaal.

## **Ruis filteren**

Hieronder ziet u de grafiek van dit 1D signaal waar duidelijk nog veel ruis op te zien is.

### Grafiek 1: ongefilterd 1D signaal

Om deze ruis weg te filteren werd met verschillende soorten filters geëxperimenteerd. Eerst werd er geëxperimenteerd met lopende gemiddelden, dit zijn gemiddelden van een bepaald aantal metingen. Als je bijvoorbeeld het lopend gemiddelde van vijf waarden wilt dan neem je de som van de waarden van metingen bijvoorbeeld van metingen één tot en met vijf en deel je dit door vijf. Daardoor wordt dus het gemiddelde van die vijf metingen genomen. Hierdoor wordt de grafiek minder beïnvloed door afwijkende meetresultaten. De tweede manier waarmee geëxperimenteerd werd, is die van het gewogen gemiddelde. Dit neemt niet zomaar het gemiddelde van de waarden maar laat de metingen ervoor en erna minder of meer meebepalen. Je zou bijvoorbeeld de metingen ervoor en erna voor 30 procent kunnen laten meetellen en de meting op het tijdstip zelf dan voor 40 procent. Dit zou je dan doen door de som te nemen van 0.3 keer de metingen erna en ervoor plus 0.4 keer de meting op het tijdstip zelf. Na heel wat experimenteren werd uiteindelijk ondervonden dat als er een gewogen gemiddelde van een gewogen gemiddelde wordt genomen dat dit een mooie grafiek oplevert. De grafiek hieronder is daar een voorbeeld van.

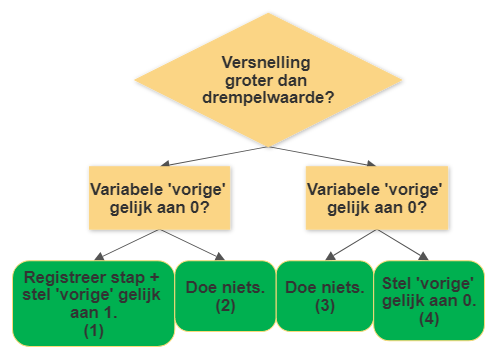
### Grafiek 2: Metingen van het experiment

## **Detectoren**

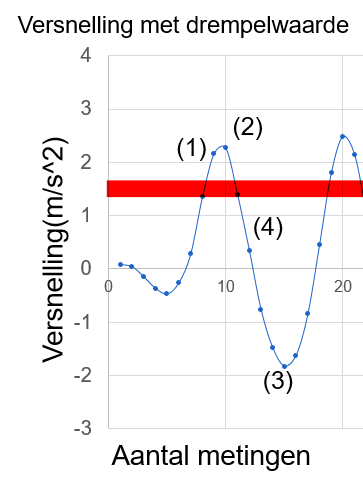
In de vierde labsessie werd de eerste detector in Java geïmplementeerd de zogenaamde ‘Dummydetector‘ en wat deze detector doet is heel simpel. Deze detector detecteert namelijk niks anders dan het aantal metingen die er gedaan wordt, dus hier worden nog geen stappen geteld. Daarna werd de functie addAccelerationData() aangemaakt om de stappen te tellen en daarin werd dus ook de gevonden filter geïmplementeerd. Hierna kon begonnen worden met het zoeken van de perfecte detector. Hiervoor werd weer met twee opties geëxperimenteerd. De eerste detector heet de ‘*SimpleTresholdDetector’*. Hierbij wordt een drempelwaarde gekozen en telkens wanneer de grafiek van de versnelling van de smartphone boven deze waarde gaat, zal het algoritme een stap tellen. De tweede optie was een ‘*DynamicTresholdDetector’* deze detector werkt op dezelfde manier als de vorige, het enige verschil is dat de drempelwaarde hier kan variëren. Zo kan bijvoorbeeld gekozen worden om de drempelwaarde in te stellen op 50 procent van de 100 metingen rond dat tijdstip. Als je de drempel waarde dan zou willen weten op meting 50 zal het algoritme de metingen nul tot en met honderd nemen en hiervan het maximum zoeken en vervolgens de drempelwaarde instellen op 50 procent van dit maximum. Deze tweede detector houdt dus rekening met de manier waarop bewogen wordt. Zo zal de drempelwaarde bij lopen hoger liggen dan bij stappen omdat er een grotere versnelling is wanneer er gelopen wordt met de smartphone. Na het vergelijken van de twee detectoren werd dan ook besloten dat de ‘*DynamicTresholdDetector’* iets nauwkeuriger was.

## **Algoritme**

Nu moest een algoritme ontwikkeld worden die uit de gefilterde 1D data stappen kon detecteren. Het idee was dus om een stap te registreren telkens de grafiek een piek maakte boven een bepaalde drempelwaarde. Het probleem was dat als je het aantal metingen boven de drempelwaarde zou tellen dat je een groter getal zal uitkomen dan het aantal stappen omdat binnen één piek meerdere punten zich boven de drempelwaarde kunnen bevinden. Het algoritme moest dus in staat zijn om te weten of een meting boven de drempelwaarde deel uitmaakt van een al getelde piek. Hiervoor werd een nieuwe variabele ingevoegd. Deze variabele werd ‘*vorige’* genoemd en werd standaard aan nul gelijk gesteld. Wat het algoritme doet, is het gelijkstellen van deze variabele aan één telkens het een stap registreert en dit is dus wanneer de versnelling van onder de drempelwaarde naar erboven verandert. Vervolgens wanneer de versnelling weer onder de drempelwaarde duikt, stelt het algoritme de variabele ‘*vorige’* weer terug gelijk aan nul. Hoe deze variabele in het algoritme gebruikt wordt, is als volgt. Het algoritme stelt altijd de volgende twee vragen: “Is de versnelling boven de drempelwaarde?” en “Is de variabele ‘*vorige’* gelijk aan 0?” Enkel als het antwoord op deze twee vragen ‘ja’ is, zal het algoritme een stap bijtellen en dus ook de variabele ‘*vorige’* gelijk stellen aan één. Als de versnelling weer onder de drempelwaarde duikt en de variabele weer gelijk gesteld moet worden aan 0 zal het antwoord op beide vragen dus ‘nee’ zijn. Hieronder vindt u een flowchart van het algoritme waarbij de rechter pijlen het antwoord ‘nee’ voorstellen en de linker pijlen het antwoord ‘ja’ voorstellen. U vindt ook een deel van een grafiek met de versnelling en de drempelwaarde aangeduid met een rode lijn. Hierop staan de verschillende situaties van de flowchart aangeduid op het juiste moment.



### Flowchart 1: Algoritme om stap te registreren



### Figuur 3: Mogelijke situatie aangeduid op grafiek met drempelwaarde

# **DETECTOR IN JAVA PROGRAMMEREN**

De smartphone geeft eigenlijk live metingen door die gelezen en omgezet moeten worden in bruikbare data. Deze bruikbare data moet dan ook nog eens correct worden geïnterpreteerd.

## **Offline analysis**

Om de detector te testen en om ermee te experimenteren kon beroep worden gedaan op de ‘*offline analysis*’. Hierbij wordt de detector toegepast op een eindig aantal metingen waarvan het aantal stappen al geweten is. Hierdoor kan geëxperimenteerd worden met veranderingen van variabele binnen de code. Doordat het aantal stappen dat zou moeten geregistreerd worden, gekend is, kan makkelijk vergelijken worden of de veranderingen een positieve of negatieve invloed hadden op de detector. Nadat de perfecte detector gevonden was, moest er enkel nog een manier gevonden worden om de live data bruikbaar te maken zodat de detector erop toegepast kon worden.

## **Definitieve filter**

Het omzetten van de data gebeurt in één eenvoudige formule. Deze formule hoe hij nu zal worden uitgelegd is natuurlijk vertaald in een hoop if-else-statements en een while-lus zodat Java deze begrijpt. Er wordt eerst een gewogen gemiddelde genomen met coëfficiënten 0.1, 0.2, 0.4, 0.2, 0.1 van alle assen apart, dus in de x-, y- en z-richting. Deze drie gewogen gemiddelden worden vervolgens omgezet in een 1D signaal met behulp van volgende formule die al gevonden was.

De valversnelling wordt nu niet meer afgetrokken van het resultaat omdat er rekening mee gehouden wordt bij de drempelwaarde. Het zou perfect ook anders kunnen. Van deze 1D data wordt vervolgens opnieuw een gewogen gemiddelde maar deze keer met de volgende coëfficiënten 0.1, 0.1, 0.3, 0.3, 0.2. Deze methode zorgt voor een zeer gladde grafiek zoals hieronder te zien, waar de pieken duidelijk te zien zijn en waar de drempelwaarde dus ook perfect op zal werken.

### Grafiek 3: Grafiek van de versnelling na het toepassen van de definitieve filter

Deze filter werkt voor alle beweegtoestanden zoals lopen en wandelen en op verschillende ondergronden. Natuurlijk zal de drempel moeten verschillen maar dit zal opgelost worden door gebruik te maken van een dynamische drempelwaarde.

## **Live data**

Iedere meting krijgt de gsm drie waarden van de accelerometer, één in de x-, y- en z-richting. Java kan de formule die we experimenteel gevonden hebben, daar niet op toepassen want de formule heeft waarden nodig op meerdere tijdstippen en momenteel onthoudt Java enkel de drie metingen die op dat tijdstip. De metingen gaan dus moeten opgeslagen worden. Ook willen we dat we oude data die we niet meer gebruiken verwijderen want na een paar uur zou er al heel veel onnodige data bestaan die gewoon opslag ruimte en batterij verbruikt. Om deze redenen werden er drie begrensde nul lijsten van lengte 200 aangemaakt. Deze worden ingevuld naargelang dat er data ontvangen wordt van de accelerometer. Op de plaats die bepaald wordt door index die we per meting één verhogen en waarvan de modulo 200 genomen van wordt, zodat we binnen de lijst blijven. Op deze lijsten kan de formule wel toegepast worden. Er wordt verwijzen naar de elementen van de lijst om de waarden te gebruiken. De uitkomst van deze live data formule wordt dat toegepast op onze drempel formule om zo de stappen te bepalen.

## **Complexe drempelwaarde**

Momenteel wordt er gewerkt met een vaste drempelwaarde. Dit is oké tot dat de gebruiker zijn bewegingstoestand veranderd en er plots geen stappen meer zou waargenomen worden. De drempelwaarde zou eigenlijk afhankelijk moeten zijn van de waarden die we binnenkrijgen. Er zal dus een extra lijst moeten zijn voor onze waarden die we krijgen uit de formule om later hiervan een maximum van te bepalen. We hebben experimenteel ondervonden dat als de drempelwaarde op 70% van de maximum waarde ligt die we in een interval hebben, we een correct beeld krijgen van de genomen stappen. Natuurlijk hebben we complicaties want als de code start heeft het enkel nul waarden, doordat we nullijsten aanmaakten. De max van een lijst die amper elementen bevat zal natuurlijk een fout beeld geven. Daarom dat we ook voor de eerste 200 metingen met een vaste drempelwaarde beginnen, namelijk 1.25\*9.81. Iedere keer dat een waarde wordt ingevuld op het laatste element van de lijst zal de drempelwaarde aangepast worden. Er wordt gewerkt met de meetsnelheid GAME en 200 metingen is gelijkwaardig met ongeveer 25 seconden. Er kan dus enige afwijking op zitten door de eerste metingen maar deze zal minimaal zijn.

# **INTENTS**

In één van onze practica hebben we kennis leren maken met intents en hoe deze functioneren. Er bestonden al een start, stop en reset intent maar deze zijn allemaal handmatig. Als een gebruiker een stappenteller wil gebruiken, zal hij toch niet de moeite er willen insteken om iedere dag handmatig zijn stappenteller te doen starten, stoppen en resetten. We hebben dus een intent proberen schrijven die het aantal stappen reset, door op nul te zetten, iedere dag om 12 uur ’s nachts. Dit kon aan de hand van een calender import. De tijd die Java opmeet hebben we ook moeten omzetten in ‘*real time’* zodat Java zou weten wanneer het 24 uur zou zijn. Dit is jammer genoeg niet volledig gelukt om uit te werken.

We hadden hetzelfde idee met de start intent. Er zou een if-statement kunnen aangemaakt worden dat als de gsm niet uitstaat en de detector nog geen waarden aan het verwerken is dat hij dat automatisch start. In plaats van eerst de app te moeten openen en dat de stappenteller dan pas van start gaat, hoe het nu is.

# **APPLICATIE**

Na het filteren van de gemeten resultaten, wordt het tijd om de app tot stand te brengen. Hiervoor wordt er eerst een mock-up\* gerealiseerd aan de hand van Android Studio\*.

## **Mock-up**

Om gestructureerd te werk te kunnen gaan, werd er eerst een design van de app gemaakt via Word.  
Het doel van de app is om het eenvoudig te houden, maar toch met voldoende info voor de gebruiker. Centraal in beeld staat het aantal stappen dat de gebruiker die dag gezet heeft. Onder het aantal stappen staat het dagelijks stappendoel. Dit kan de gebruiker zelf instellen. Zo ziet hij hoeveel stappen hij diezelfde dag nog moet zetten om zijn doel te bereiken. Ook het aantal calorieën die hij verbrand heeft, is te zien op de app, namelijk linksonder het stappendoel. Rechts ervan ziet men de afstand die hij die dag al afgelegd heeft, uitgedrukt in kilometers.  
Al deze informatie ziet men meteen bij het openen van de app. Helemaal onderaan het scherm staat echter een balk waarmee men kan navigeren naar 2 andere tabbladen, namelijk ‘*settings’* en ‘*profile’*. Het tabblad ‘*settings’* geeft de gebruiker de mogelijkheid om het stappendoel op een zeer eenvoudige manier aan te passen. Wanneer men rechtsonder op de knop ‘profile’ klikt, kan men zijn persoonlijke gegevens aanpassen, namelijk geslacht, leeftijd, gewicht en lengte. Deze gegevens worden gebruikt om het aantal verbrandde calorieën te berekenen.

De achtergrondkleur van de app is wit gebleven. Anders zou de lay-out te druk worden. Er zijn wel enkele accentkleuren in aangebracht, om de app iets aantrekkelijker te maken.



### Figuur 2: lay-out app

## **Material design**

Binnen Android Studio werd gebruik gemaakt van Material Design. Dit is een designtaal die ontwikkeld is door Google. Bij Material Design komt het erop neer dat ze een nieuwe visuele taal wouden maken die bestond uit een combinatie van technische en wetenschappelijke innovatie. Wat zeer opvallend is voor dit design is dat er alleen maar gebruik wordt gemaakt van een X- en een Y-as, het zogenaamde ‘platte’ design. Het is dus een zeer minimalistische designtaal. Ook animaties en kleuren spelen een belangrijke rol binnen dit design.

## **Material Design toegepast**

Na het ontwerpen van de lay-out via Word, kon het omgezet worden in een echte app aan de hand van Android Studio. Hiervoor werd gebruik gemaakt van een reeds bestaand document, dat de basis van de app vormde. Op dit bestand werd er dus verder gebouwd. De eerste stappen waren het toevoegen van een aantal basiselementen, namelijk een ‘EditText’, een ‘TextView’ en een ‘Button’. Deze drie elementen vormen later de basis van de app. Na het toevoegen van deze elementen aan de User Interface\*, werd er overgeschakeld naar het programmeren via de programmeertaal Java. Hier werden de eerder gevonden filters omgezet in deze programmeertaal. Deze filters werden dan later gekoppeld aan de elementen in de mock-up, zodat de juiste gegevens op de app verschenen.

## **Interactie met de gebruiker**

Het doel van de app is dat men ziet hoeveel stappen men dagelijks zet. Om de interactie met de gebruiker te verbeteren, kunnen hier nog enkele elementen aan toegevoegd worden. Een paar voorbeelden hiervan zijn de eerder genoemde ‘aantal calorieën’ en ‘afgelegde afstand’. Op het hoofdtabblad kan men dus alleen informatie aflezen. Op de twee andere tabbladen is er meer interactie met de gebruiker. Hier kan men zowel zijn gegevens als zijn stappendoel aanpassen. Op die manier wordt de app persoonlijker voor de gebruiker. Andere elementen om de interactie met de gebruiker te verbeteren, zijn bijvoorbeeld het toevoegen van een grafiek waarop de gebruiker zijn wekelijkse vooruitgang kan aflezen. Maar om het toch overzichtelijk te houden, is dit achterwegen gebleven in de app. Ook is het niet zo eenvoudig om zo een grafiek toe te voegen. Er komt namelijk heel wat bij kijken. Men moet zich afvragen bij welke dag je de week laat starten op de grafiek, of je telkens verdeelt per week of iedere maandag bijvoorbeeld een nieuwe grafiek start,…Dit is dus ook een van de redenen waardoor er geen grafiek toegevoegd is aan de app.

**Referenties**

*Material design for android*. (sd). Opgehaald van developers: https://developer.android.com/develop/ui/views/theming/look-and-feel

Tuk, Y. (2015, juni 12). *De invloed van Google: van flat naar material design*. Opgehaald van emerce.nl: https://www.emerce.nl/achtergrond/trends-invloed-van-google-flat-versus-material-design