

Mobile app | As Adventure x Natuurpunt hiking app

in a App Store nikiel

Inhoudsopgave

1. INLEIDING	4
2. ANALYSE	5
2.1. De applicatie	5
2.1.1. Product Breakdown Structure	5
2.1.2. Use Cases	6
2.2. Gebruikte tools	7
2.2.1. PostgreSQL	7
2.2.2. React Native	7
2.2.3. Java Spring Boot	8
2.2.4. Visual Studio Code	9
2.2.5. IntelliJ IDEA	9
2.2.6. Bitbucket	9
2.3. Data	9
2.3.1. Datamodel	9
2.3.2. Data flow	10
3. WIE?	12
3.1. Yonderland/As Adventure	12
3.2. Natuurpunt	12
4. WAAROM?	12
4.1. Toename in verkoop	12
4.2. Natuurpunt	12
5. WAAR?	13
6. WANNEER?	13
6.1. Work Breakdown Structure	13
7. WAT?	15
7.1. Beschrijving	15
7.2. Naam	15
7.3. Logo	15
7.4. Schermen	15
8. HOE?	21
8.1. SQL-scripts	21
8.2. Modificatie	21
8.3. Repository – Service – Controller	21
8.4. Entiteiten en DTO's	21
8.5. MapStruct	22
8.6. Componenten	22
8.7. API requests	23
8.8. App layout	24

8.9. Vertaling statische data	25
8.10. Achtergrondafbeeldingen + kaarten	27
8.11. Navigatie naar Google Maps en Waze	27
8.12. Kleurpallet	28
9. BESLUIT	29
LITERATUURLIJST	30

1. Inleiding

In dit document wordt er uitgelegd hoe TrailFit (de naam van de ontwikkelde applicatie) in elkaar zit en hoe ik te werk ben gegaan tijdens de looptijd van dit project. Bij het hoofdstuk 'Analyse' is te vinden wat er in de voorbereidende fase van het project onderzocht en beslist is. Dingen als de functionaliteiten van de applicatie, de keuze van de tools en de structuur van de data, zijn hier in detail uitgelegd met argumentatie waarom de gekozen optie beter is dan een andere.

In het hoofdstuk 'Wie?' is meer informatie over de klant/werkgever te vinden. Dit schept al een beetje een beeld over waarom de applicatie mogelijk gemaakt wordt. Dit wordt ook nog verder uitgelegd in het hoofdstuk 'Waarom?'. Dit hoofdstuk bevat informatie over waarom de applicatie gemaakt wordt, wat de voordelen voor de betrokkenen partijen zijn, enz.

'Waar?' en 'Wanneer?' zijn twee kortere hoofdstukken die een beeld scheppen over het bereik van de applicatie en hoe de periode van het project verdeeld was over de verschillende taken.

Het 'Wat?' hoofdstuk is vrij vanzelfsprekend, hier wordt er getoond en uitgelegd wat er precies opgeleverd is tijdens de stageperiode. Er wordt uitgelegd waarom er voor een bepaalde naam en logo is gekozen en alle schermen worden getoond en ook uitgelegd. Dit laatste deel kan ook in detail gelezen worden in de handleiding.

Hoe ben ik te werk gegaan? Welke structuren, best practices en tips heb ik toegepast? Dit wordt allemaal uitgelegd in het 'Hoe?' hoofdstuk. Hier laat ik zien hoe de applicatie achter de schermen in elkaar zit. Dingen als structuur van de api of best practices in het front-end worden hier besproken en uitgelegd.

In het besluit valt nog te lezen wat er eventueel nog beter kan en waarom ik dit niet heb kunnen implementeren tijdens mijn stageperiode. Ook wordt er uitgelegd wat er uiteindelijk is opgeleverd.

2. Analyse

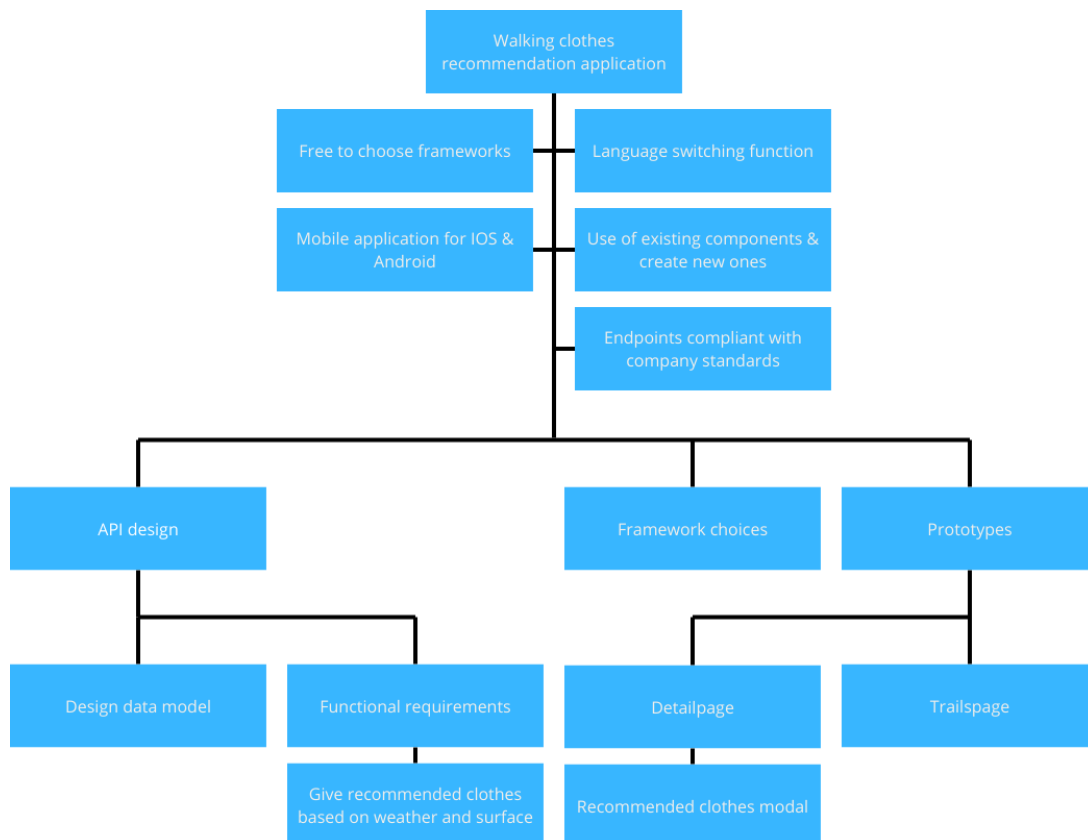
Tijdens de eerste weken van de stage heb ik vooral gewerkt aan het analyseren en onderzoeken van de opdracht en haar onderdelen. Zo deed ik onderzoek naar welke taal ik ging gebruiken voor de backend, hoe de data doorheen de applicatie ging en welke taal ik ging gebruiken om het front-end gedeelte te ontwikkelen. Ook ging ik kijken welke functionaliteiten de app Als bevatten en ging ik een planning opstellen voor het verloop van het project. Toen ik eenmaal toegang had tot wat data, kon ik ook gaan kijken hoe mijn datamodel eruit zou zien en welk platform ik zou gebruiken voor mijn database.

2.1. De applicatie

Om een duidelijk beeld te scheppen van de applicatie, leek het me een goed idee om alle functionaliteiten van de applicatie in een overzichtelijk schema weer te geven. Dingen als een Product Breakdown Structure en een Use Case Diagram lijken me zeer geschikt om dit te schetsen.

2.1.1. Product Breakdown Structure

Het leek me gemakkelijk om een PBS op te stellen, zodat ik hieruit de belangrijkste taken binnen mijn project kon halen. Het geeft ook een mooi overzicht van welke taken voorgaan op andere, waardoor ik een correctere planning kon opstellen.

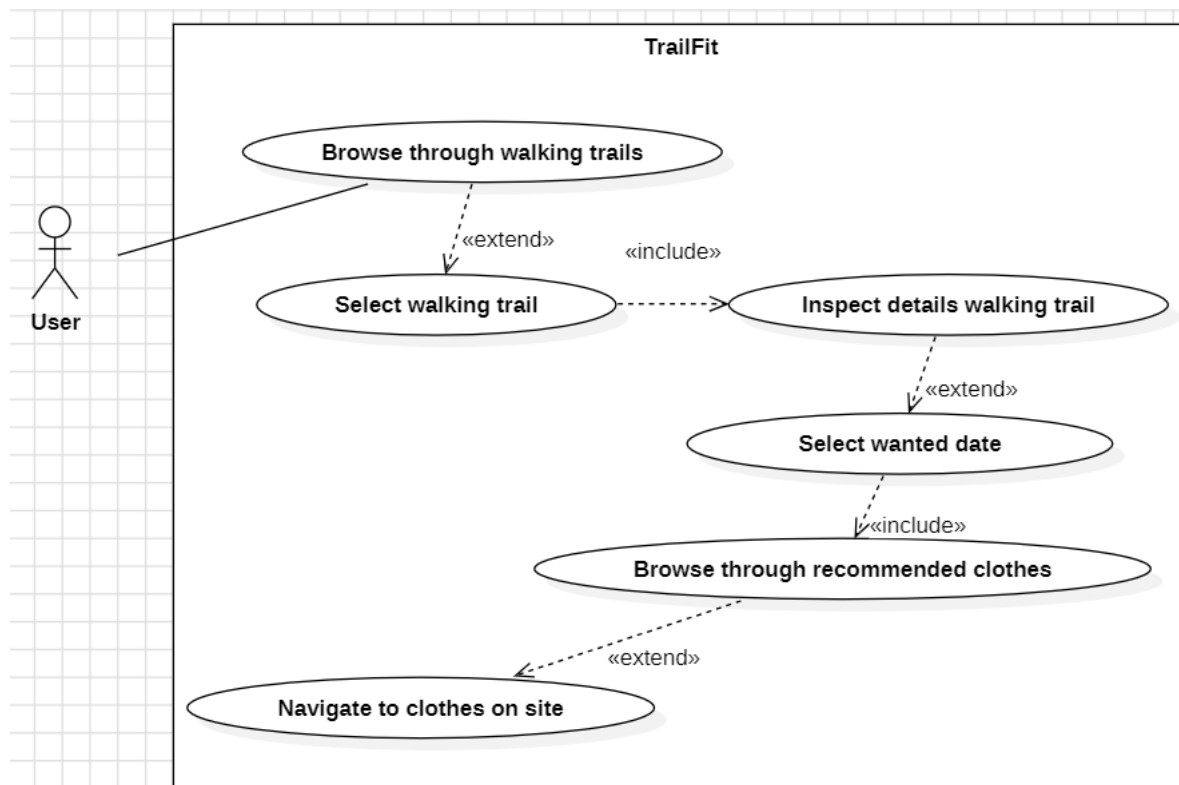


Bovenaan is de “naam” van het project/product te zien, dit geeft mooi de core-functionaliteit van de applicatie weer. Een app die wandelkleding moet kunnen aanraden. De volgende vakken zijn de vereisten voor deze applicatie. We mogen de frameworks vrij kiezen, er moet een taalfunctie zijn, de endpoints moeten in lijn liggen met die van het bedrijf en de applicatie moet beschikbaar zijn voor iOS en Android.

Het niveau onder de vereisten zijn dan de voornaamste taken te vinden. Het ontwerpen van de API, het kiezen van frameworks en het ontwerpen van prototypes. Het designen van de API houdt in: het ontwerpen van het datamodel en het maken van de API zelf rekening houdend met de functionele vereisten. Het maken van prototypes wordt verder onderverdeeld naar de belangrijkste pagina's.

2.1.2. Use Cases

Om duidelijkheid te scheppen over wat de belangrijkste functionaliteiten van de applicatie zijn, is het opstellen van een use case diagram zeer handig.



De applicatie Als initieel niet super veel functionaliteiten hebben, dus lijkt het use case diagram vrij simpel. Dit zijn de belangrijkste functionaliteiten met een korte beschrijving:

- Door de wandelroutes scrollen
Een gebruiker moet alle wandelroutes kunnen zien en hierdoor kunnen scrollen.
- Een route selecteren
Als de gebruiker geïnteresseerd is in een route, moet hij/zij deze kunnen selecteren.
- Details van wandelroute zien
Wanneer de gebruiker dan heeft geklikt op een route, moet de applicatie navigeren naar de detail pagina waar de gebruiker meer details van de wandelroute kan zien.
- Selecteer de gewenste datum
Is de gebruiker geïnteresseerd om deze wandeling te doen, dan kan hij/zij een datum selecteren waarop de gebruiker het liefst deze route doet.
- Scrol door suggesties
Eenmaal een datum is geselecteerd en om suggesties is gevraagd, moet de gebruiker door de aangeraden kleding kunnen scrollen.
- Navigeer naar webshop
Als de gebruiker dan geïnteresseerd is in kleding, moet deze via een knop naar de webshop van AS Adventure kunnen navigeren.

2.2. Gebruikte tools

De programmeertalen voor front -en backend heb ik tegen elkaar afgewogen met behulp van de Weighted Ranking Methode. Deze heb ik ook toegepast om te beslissen welke database ik ging gebruiken, natuurlijk heb ik de uiteindelijke beslissing nog overlopen met mijn stagebegeleider. De code editors heb ik beslist op basis van de uiteindelijk besliste programmeertalen, maar deze zijn weinig relevant tot het project.

2.2.1. PostgreSQL

De meest geschikte backend ben ik, zoals eerder vermeld, gaan beslissen op basis van een Weighted Decision Matrix. Ik heb de 3 (naar mijn mening) beste opties genomen en ben deze dan tegen elkaar gaan opmeten met behulp van verschillende factoren.

	Ratings		
Criteria	MySql	PostgreSQL	MongoDB
Performance	2	1	3
Complex query's	2	3	1
Voorkeur bedrijf	2	3	1
Relational data	2	3	1
Community	2	3	1
Total	10	13	7

Ik heb gebruik gemaakt van factoren als prestatie, of het complexere query's aankan, wat de voorkeur van het bedrijf is, of het relationele data is en of er een grote community rond is. Hier is een korte uitleg van wat elke categorie inhoudt:

- Prestatie
Hoe snel de database query's kan verwerken en data kan teruggeven.
- Complexe query's
Of de database in staat is om complexere query's te verwerken.
- Voorkeur bedrijf
Welke voorkeur het bedrijf heeft, ik heb dit overlopen met mijn stagebegeleider (na later onderzoek kwam hij er zelfs achter dat alle nieuwe databases PostgreSQL moesten zijn).
- Relationele data
Mijn data waren relationeel, dus de database moet ook in staat zijn om relationele data aan te kunnen.
- Community
Hoe actief de community is rond de database-vorm en hoeveel documentatie er beschikbaar is.

Na wat onderzoek en scores is duidelijk te zien dat PostgreSQL het meest geschikte platform is voor de database voor de TrailFit applicatie. Na dit besproken te hebben met mijn stagebegeleider, bleek zoals in de "Voorkeur bedrijf" categorie te zien is, dat alle nieuwe databases PostgreSQL databases moesten zijn.

2.2.2. React Native

Zoals ook bij de databasekeuze, heb ik hier ook de Weighted Decision Method toegepast. De (volgens mij) 3 beste opties zijn met elkaar vergeleken op basis van 4 factoren.

	Ratings		
Criteria	React Native	Angular	Kotlin
Cross-platform	3	2	1
Experience	3	2	1
Usage in company	3	2	1
Documentation	3	1	2
Total	12	7	5

Om de front-end programmeertaal te beslissen heb ik gebruik gemaakt van factoren als cross-platform beschikbaarheid, ervaring, gebruik in het bedrijf en documentatie. Hier nog een korte uitleg van alle factoren:

- Cross-platform beschikbaar
Is het mogelijk om makkelijk de Android en iOS packages van de applicatie te exporteren?
- Ervaring
Hoeveel ervaring heb ik met de programmeertaal? Met welke taal kan ik het beste uit de voeten?
- Gebruik in bedrijf
Welke taal wordt er momenteel veel gebruikt in het bedrijf? Kan ik goede ondersteuning krijgen voor deze taal?
- Documentatie
Is er veel documentatie beschikbaar en heeft het een grote/duidelijke community?

Na het afwegen van de scores is duidelijk te zien dat React Native het meest geschikt is voor deze applicatie. Ook gebruiken ze in het bedrijf React (wat niet compleet hetzelfde is), waar ik dan toch wat mogelijke ondersteuning voor kan krijgen.

2.2.3. Java Spring Boot

Ook de beste optie qua programmeertaal voor de backend heb ik beslist op basis van een Weighted Decision Matrix. Ik heb ook hier weer 3 opties genomen die mij geschikt leken voor deze applicatie en deze tegen elkaar opgewogen op basis van enkele factoren.

	Ratings		
Criteria	Spring Boot	.NET	Node.js
Usage in company	3	1	2
Experience	3	2	1
Documentation	2	1	3
Integration with frontend	2	1	3
Total	10	5	9

Hier heb ik gebruik gemaakt van factoren zoals het gebruik in het bedrijf, de ervaring die ik heb, de beschikbare documentatie en of deze makkelijk te integreren valt met de frontend. Hier nog even een korte uitleg over wat deze factoren inhouden:

- Gebruik in het bedrijf
Welke taal wordt het meeste gebruikt in het bedrijf? Waar kan ik het meeste/de beste ondersteuning bij krijgen Als ik in de problemen geraken.
- Ervaring
Hoeveel ervaring heb ik met deze programmeertaal? Met welke taal kan ik het beste overweg?

- Documentatie
Van welke taal is er het meeste/de duidelijkste documentatie en waar is de community zeer actief?
- Integratie met de frontend
Welke taal is het makkelijkst te integreren met React Native?

Na al deze factoren en scores in kaart te brengen, was er maar een klein verschil tussen Java Spring Boot en Node.js. Ik heb dan samen met mijn stagebegeleider beslist om te gaan voor Java Spring Boot, omdat ik hier meer ervaring mee heb en dit is ook de taal die in het bedrijf gebruikt wordt.

2.2.4. Visual Studio Code

Om het front-end te ontwikkelen is er natuurlijk een code editor nodig. Op school hebben we al lessen in React Native gehad en hiervoor hebben we telkens VS Code gebruikt. Omdat dit toen goed werkte en dit in het algemeen een code editor is met veel handige extensies, heb ik beslist deze te gebruiken doorheen het project.

2.2.5. IntelliJ IDEA

Ook voor backend development had ik natuurlijk een code editor nodig. De keuze hier was nog makkelijker dan voor het front-end, want er wordt doorheen het bedrijf namelijk enkel gebruik gemaakt van IntelliJ IDEA. Ook gebruikten wij deze editor doorheen de lessen Java Advanced, dus ik was er al bekend mee, wat zorgt voor een snellere werking. IntelliJ heeft ook een handige optie waarbij je je database tabellen makkelijk kunt zien.

2.2.6. Bitbucket

Het bedrijf maakt gebruik van Bitbucket als repository-tool, dus was het een logische keuze om dit ook te gebruiken. Het bedrijf heeft zo ook makkelijker toegang tot de code wanneer mijn stage klaar is.

2.3. Data

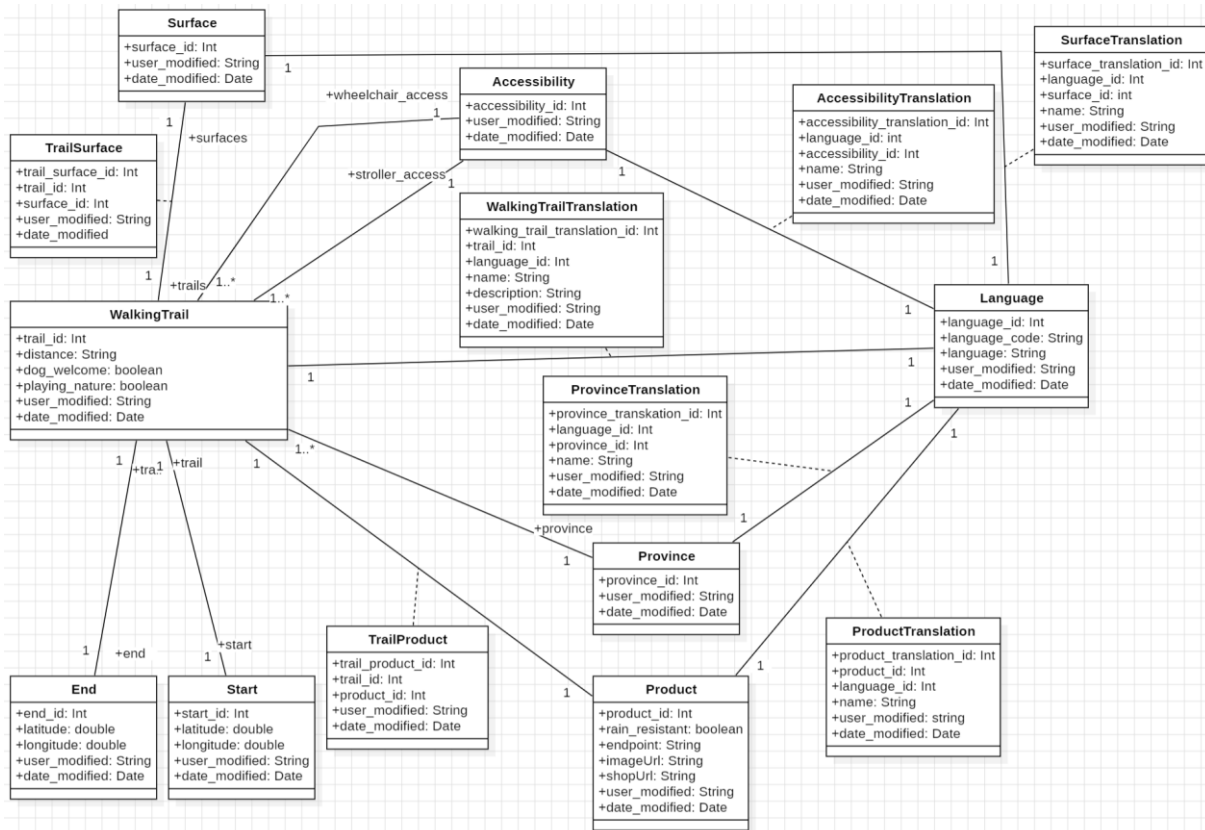
Nu de platformen beslist zijn, kon ik beginnen kijken naar de data. Hoe deze opgeslagen moet worden, hoe deze doorheen de applicatie verwerkt en gebruikt wordt. Dingen als een datamodel en data flow zijn essentiële onderdelen in deze fase.

2.3.1. Datamodel

Alle data die ik mocht/kon gebruiken heb ik gekregen via een link:

<https://www.asadventure.com/nl/expertise-tips/walking/Ontdek-de-A-S-Adventure-wandelroutes-van-Natuurpunt.html>

Dit is een statische pagina die niet verbonden is aan een database of API, wat zorgt voor veel onregelmatige data. Categorieën die anders geschreven worden, te veel verschillende toegankelijkheden en geen makkelijke manier om deze data op te halen waren allemaal moeilijkheden die ik in deze fase tegen kwam. Uiteindelijk heb ik dan toch een voorlopig datamodel kunnen opstellen op basis van deze gegeven data.



Op het eerste gezicht lijkt dit een zeer ingewikkeld datamodel, wat misschien ook wel het geval is, maar er is wel een duidelijke structuur te zien. Er zijn twee tabellen waar de meeste andere tabellen mee verbonden zijn: de Walkingtrail en Language tabellen. De WalkingTrail tabel is logisch, aangezien de volledige applicatie draait rond wandelroutes. De verbindingen met de Language tabel komen doordat er een taalfunctie Als geïntegreerd zijn in de applicatie en dit was de beste manier om dit te doen. Dit hebben we beslist na een meeting met iemand van het SQL-team, die ook zei dat dit de beste oplossing is voor de data en applicatie.

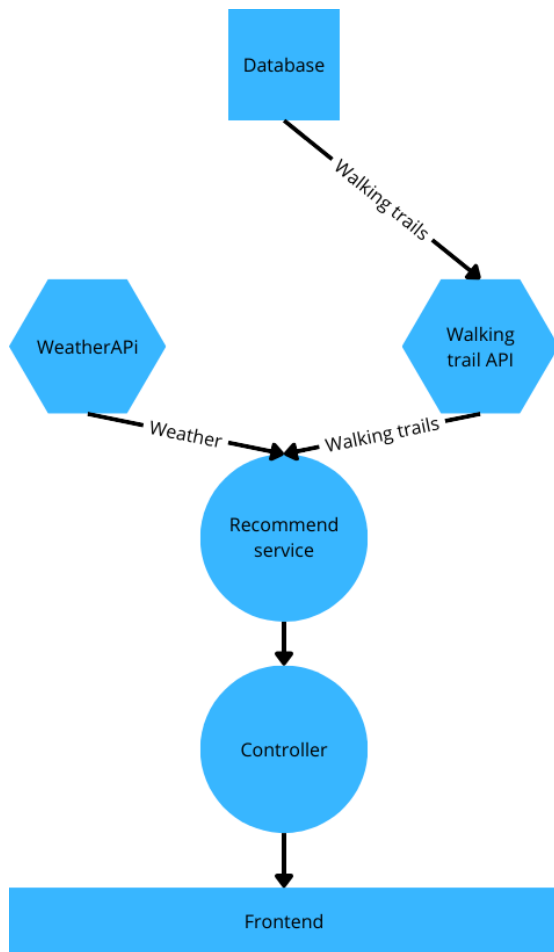
Een wandelroute heeft een begin -en eindpunt, waarvoor elks een tabel is voorzien. Een route kan ook verschillende ondergronden hebben, maar deze kunnen ook aan meerdere routes behoren, wat maakte dat ik gebruik Als maken van een associatietabel. Hetzelfde geldt voor de producten gelinkt aan een route. Deze producten houden de verschillende categorieën van producten in die aangeraden zijn voor een route. Een route is natuurlijk gelegen in één provincie, wat maakt dat een associatietabel hier niet nodig is. Ook is er een toegankelijkheid voor rolstoelen/kinderwagens, dit zijn de twee relaties tussen de route en de Accessibility tabel.

De Language tabel is zoals eerder vermeld verbonden met verschillende Translation tabellen. Deze tabellen bevatten de vertalingen van tekstuele data van de gerelateerde tabellen. Het zijn allemaal associatietabellen omdat een tabel in meerdere talen vertaald moet worden en een taal kan verbonden zijn met verschillende objecten van die tabel.

De attributen van de tabellen worden allemaal in snake_case opgeslagen, dit is om in de backend Flyway (database migratietool) te gebruiken om de entiteiten te linken met de tabellen.

2.3.2. Data flow

Wat ook altijd een duidelijk overzicht geeft is een data flow diagram, het laat zien door welke onderdelen de data verwerkt wordt en waar er gebruik wordt gemaakt van externe data.



Zoals te zien is op het diagram, wordt er eerst data gehaald uit de database door onze API. Deze wordt dan samen met data van een externe Weather API verwerkt door een service. Elke service heeft haar eigen controller die op zijn beurt dan endpoints gaan aanbieden die we in de frontend kunnen gebruiken.

3. Wie?

In dit onderdeel bespreek ik de klant/stageplaats, wie zijn ze? Waar staan ze voor? Hoe zit het bedrijf in elkaar? Dit geeft later ook een duidelijk beeld waarom ze deze applicatie wilden. De applicatie is een samenwerking van Yonderland/As Adventure en Natuurpunt. As Adventure sponsort namelijk enkele wandelroutes van Natuurpunt en wil deze nu meer toegankelijk maken via een applicatie.

3.1. Yonderland/As Adventure

As Adventure is door de meeste mensen in België wel gekend als een grote outdoor keten. Zij willen dat mensen buitenshuis leuke activiteiten en zorgen daarvoor met de verkoop van kwaliteitsvolle outdoor producten. Ze willen dit doel wel realiseren op een ecologische en duurzame manier.

As Adventure is op haar beurt dan weer een deel van de Yonderland groep. Yonderland is een overkoepelend bedrijf dat alle e-commerce en winkels onderhoudt van haar 7 bedrijven. Andere bekende bedrijven in Yonderland zijn Juttu en Bever, wat ook outdoor ketens/merken zijn.

3.2. Natuurpunt

Natuurpunt is een vrijwilligersorganisatie die in staat voor het beschermen alsook het genot van de natuur. Natuur is hun prioriteit wanneer ze samen werken met andere bedrijven. Zij willen wandelroutes populairder en toegankelijker maken om zo buitenactiviteiten te bevorderen.

4. Waarom?

Aangezien dat deze applicatie een samenwerking is tussen As Adventure en Natuurpunt en dit twee ander soort bedrijven zijn, gaan de doelen van deze applicatie ook iets anders zijn natuurlijk. As Adventure is een bedrijf dat probeert winst te maken terwijl dit voor Natuurpunt niet hun eerste streefdoel is.

4.1. Toename in verkoop

De voornaamste reden voor de ontwikkeling van deze app voor As Adventure is natuurlijk de toeneming in verkoop van wandelkleding. De applicatie zou moeten zorgen voor een extra toegangspunt tot de webshop, wat automatisch leidt tot een toename in verkoop. Dit is natuurlijk een van de hoofddoelen voor een bedrijf. Bijkomend bevordert de app mogelijk ook buitenactiviteit en dit is ook iets waar As Adventure voor staat.

4.2. Natuurpunt

Natuurpunt is een vrijwilligersorganisatie, wat betekent dat hun eerste streefdoel niet per se winst maken is. Zij focussen zich meer op het genot van de natuur en met deze app worden mensen aangespoord om meer buiten te komen en te genieten van de natuur. De gebruikers kunnen mogelijk ook nog anderen aansporen waardoor wandelpaden misschien meer gebruikt worden.

5. Waar?

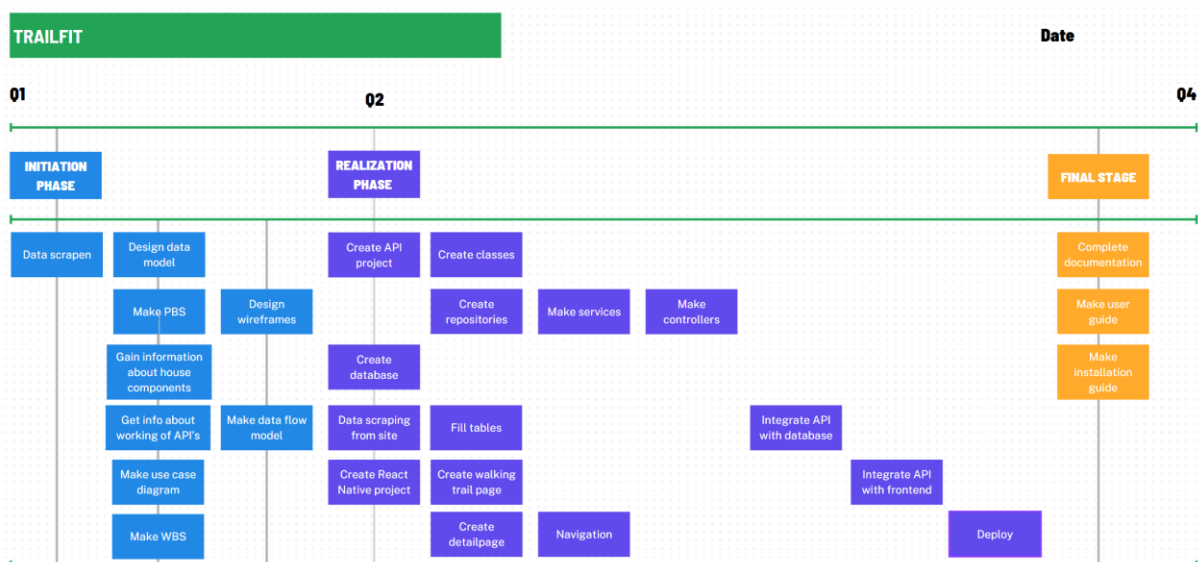
Yonderland is een groep die op verschillende plekken in de wereld gevestigd is, daardoor was een van de vereisten een taalfunctie. Zo kunnen al hun klanten en medewerkers deze applicatie gebruiken. De applicatie ondersteunt voorlopig wel enkel wandelroutes in België, wat zorgt voor toch een beperkter publiek van mensen. Het is wel de bedoeling dat de app beschikbaar is in de App Store en Play Store, wat zorgt dat iedereen er wel toegang tot heeft.

6. Wanneer?

Aan de start van elk project is het aangeraden om een planning op te stellen en dit is precies wat ik ook heb gedaan. We hadden al een soort 'planning' meegekregen vanuit de school, zo waren de eerste 3 weken "initiation phase", wat inhoudt dat we het project moesten gaan analyseren en een projectplan moesten gaan opstellen. De weken daarna waren dan de "realisation phase", deze diende dan om het project uit te werken zoals gepland in de "initiation phase". Natuurlijk was dit maar een richtlijn en mochten we onze eigen planning maken, dus ben ik begonnen met het opstellen van een Work Breakdown Structure.

6.1. Work Breakdown Structure

In dit schema verdeel je alle taken binnen het project over de tijdsperiode die je krijgt. Het is wel de bedoeling dat je een logische volgorde en structuur aanhoudt. Zo heb je een duidelijke manier van werken en weet je wat er moet gebeuren voordat je aan een andere taak begint.



In de eerste weken heb ik ingepland om vooral veel info te verzamelen over alle delen van het project en beslissingen te maken over wat ik waarvoor ga gebruiken. Ook ging ik al wat diagrammen opstellen die het project duidelijker maakten en ging ik de nodige data verzamelen zodat ik goed voorbereid was op de realisatiefase.

Ik dacht in de realisatiefase eerst alle gedeeltes al te creëren zodat ik eventueel verder kon met een ander deel Als ik ergens vastzitten. Na enkele weken werken, dacht ik dan aan het integreren van deze gedeeltes.

Ik heb de structuur die de school suggereerde grotendeels behouden, ik denk dat ik misschien enkele dagen eerder ben begonnen met de realisatiefase en ik heb nog een afwerkingsfase toegevoegd.

Uiteindelijk heb ik mijn planning niet 100% gevolgd.

7. Wat?

Het algemene idee van het project was het maken van een applicatie die wandelkleding kan aanraden aan mensen die geïnteresseerd waren in een wandelroute die gesponsord is door As Adventure.

7.1. Beschrijving

De uiteindelijke applicatie is een overzicht van alle wandelroutes die gesponsord zijn door As Adventure, waarbij geïnteresseerde gebruikers verder kunnen klikken op een wandelroute om dan meer gedetailleerde informatie te zien. Het is ook mogelijk om de applicatie te gebruiken in 5 verschillende talen (Nederlands, Frans, Engels, Duits en Spaans). De gebruiker kan op de detailpagina suggesties voor wandelkleding aanvragen en eventueel navigeren naar het startpunt van de wandelroute. Via de suggesties kan de gebruiker dan naar de webshop van As Adventure navigeren.

7.2. Naam

De uiteindelijke naam voor de applicatie is TrailFit, deze heb ik mogen verzinnen aangezien er nog geen naam was. Ik wou iets wat duidelijk maakte wat de app juist deed, iets met wandelroute en kleding. De Trail komt dus van wandelroute en de Fit kan twee betekenissen hebben. Enerzijds Fit als in "Ik ben fit want ik ga veel wandelen" en anderzijds Fit uit outfit, want de app geeft suggesties over een outfit voor tijdens het wandelen.

7.3. Logo

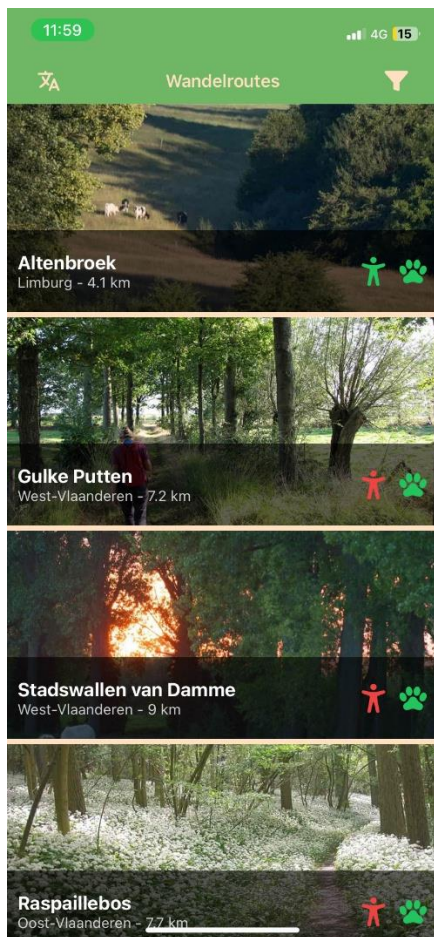
Een applicatie moet natuurlijk ook een logo hebben en aangezien er nog geen logo was, mocht ik er een maken. Ik heb samengezeten met mijn zus, die een vak grafisch design heeft, waardoor zij mij hier perfect mee kon helpen. We zijn gegaan voor een logo dat direct de essentie van de applicatie weergeeft.



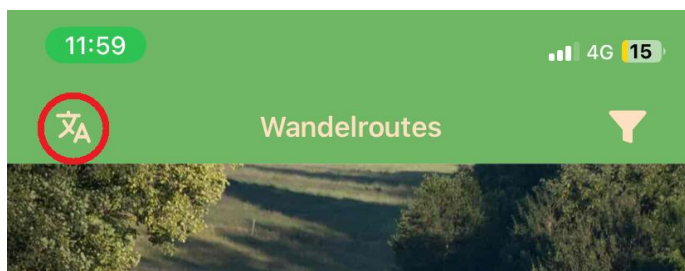
De lege kapstok in het logo staat voor de suggesties in kleding die de app de gebruikers geeft. Deze is leeg omdat ze momenteel nog geen idee zouden hebben wat ze moeten aandoen. TrailFit lijkt ook aan de kapstok te hangen omdat na het gebruik van de app, je kapstok wel gevuld kan zijn. Binnenin de kapstok zie je dan nog bergen, die staan voor de natuur rond de wandelroutes.

7.4. Schermen

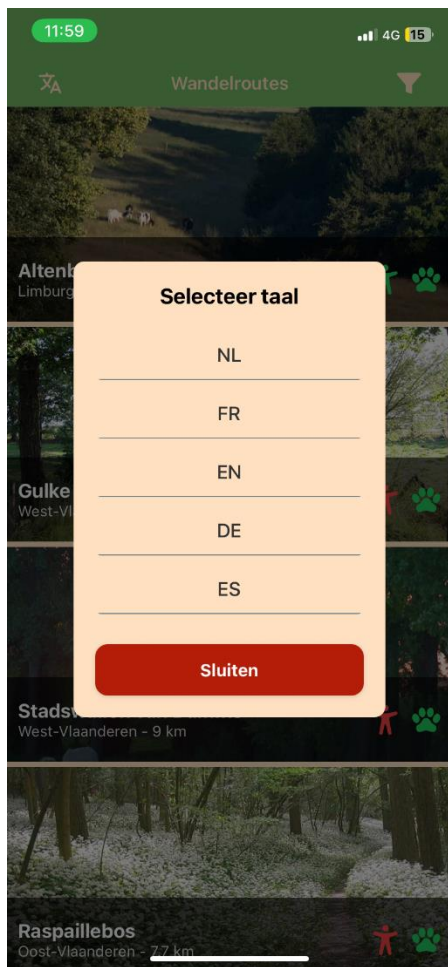
De volledige app bestaat uit 6 schermen en een errorscherm.



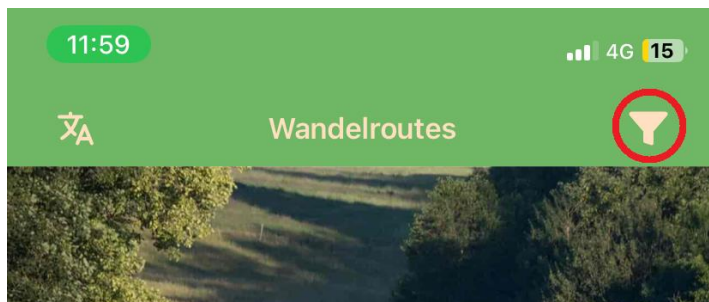
Dit is het beginscherm, hier kunnen de gebruikers door de verschillende wandelroutes scrollen en er eventueel één selecteren. Elke wandelroute bestaat uit een kaart met de naam, provincie en afstand van die route. Rechts op de kaart zijn er twee iconen te zien, eentje laat zien of de route in speelnatuur gelegen is en de andere of dat honden welkom zijn.



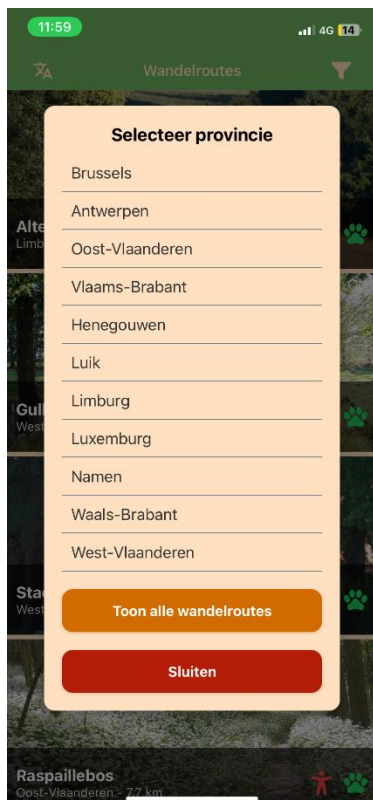
Links bovenaan het scherm is er een taal-icoontje te zien, wanneer hier op wordt gedrukt, wordt het taal menu geopend.



Hier heeft de gebruiker de optie om zijn/haar app naar een andere taal te veranderen.



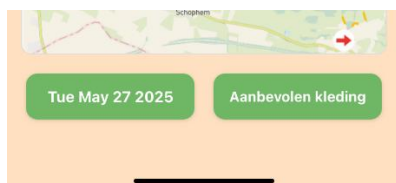
Recht bovenaan het beginscherm is een filtericoontje te zien, wordt deze ingedrukt, dan opent het provinciemenu.



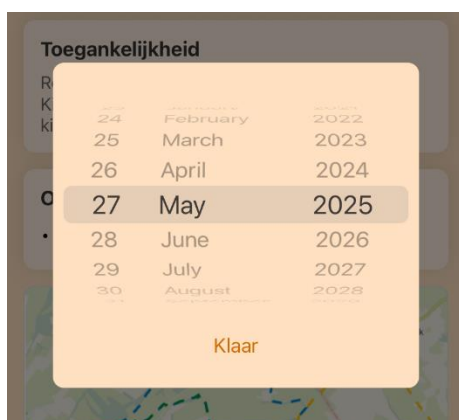
Hier worden alle provincies in België weergegeven, wil de gebruiker bijvoorbeeld een wandeling gaan doen in Antwerpen, dan selecteert hij/zij “Antwerpen” en dan filtert de app alle routes op enkel provincie Antwerpen. Mocht de gebruiker dan terug alle routes willen zien, dan kan hij/zij op de “Toon alle wandelroutes” en dan worden alle wandelroutes terug getoond.



Eenmaal de gebruiker dan een wandelroute heeft gekozen, komt hij/zij op de detailpagina. Hier worden meer gegevens van de wandelroute weergegeven zoals een beschrijving, de ondergronden, hoe toegankelijk de route is voor rolstoelen en kindervan en een kaartje van de route.



Onderaan deze pagina zijn twee knoppen te zien, de linker knop opent een datepicker.



Hier kan de gebruiker een datum selecteren waarop hij/zij graag de wandeling zou willen doen.

De rechterknop kan ingedrukt worden wanneer de juiste datum geselecteerd is (standaard is vandaag). Dit opent dan de suggesties.




Hier kan de gebruiker dan categorieën kleding zien die door de app aangeraden zijn. Als de gebruiker geïnteresseerd zijn in een type kleding, dan kan hij/zij op de “Kopen” knop drukken en wordt de gebruiker naar de webshop gestuurd waar dan de juiste filters al toegepast zijn.

8. Hoe?

Tijdens het ontwikkelen van de applicatie heb ik gebruik gemaakt van bepaalde methodes, code practices en structuren van code. Dit om het ontwikkelen makkelijker te maken, maar ook om de code later leesbaar en duidelijk te maken voor toekomstige developers.

8.1. SQL-scripts

Om het later toevoegen van data aan de databank zo makkelijk mogelijk te maken, maken zen bij Yonderland gebruik van 'version scripts'. Dit zijn scripts die de verschillende versies van de database bevatten. Zo heb ik tijdens mijn project gebruik gemaakt van twee version scripts:

 `V1_CreateTables.sql`

 `V2_FillTables.sql`

In het V1 script worden alle tabellen met de juiste attributen aangemaakt en in het V2 script worden deze opgevuld. Moet er later een modificatie gebeuren aan een tabel of ergens anders in de database, dan kan er een V3 script worden aangemaakt dat deze tabel update. Zo moet niet de hele database gedropt worden om iets kleins aan te passen.





8.2. Modificatie

Er wordt vanuit Yonderland ook aangeraden om attributen als `user_modified` en `date_modified` toe te voegen aan elke tabel. Deze kunnen dan veranderd worden wanneer een admin of developer iets van data aanpast. Zo is er makkelijk te zien wanneer wie iets heeft aangepast wanneer er mogelijk iets fout loopt.

```
@LastModifiedDate
@CreatedDate
private Date dateModified;
private String userModified;
```

8.3. Repository – Service – Controller

Zoals we ook in enkele vakken op school hebben geleerd, wordt er bij de meeste services bij Yonderland gebruik gemaakt van het 'repository – service – controller patroon'. Hier gaat de repository data ophalen uit de database. De service maakt dan gebruik van haar repository om de data te verwerken, in dit proces kan eventueel nog gebruik gemaakt worden van een bijhorende mapper. De controller heeft één of meerdere endpoints waarvan de data in de juiste vorm komt uit de service. De belangrijkste entiteiten hebben deze 3 kern layout.

-  `WalkingTrailRepository`
-  `WalkingTrailMapper`
-  `WalkingTrailService`
-  `WalkingTrailController`

8.4. Entiteiten en DTO's

De meeste services bij Yonderland bestaan uit 3 geconnecteerde projecten. Er is voor dit project bijvoorbeeld:

- Een trailfit-service

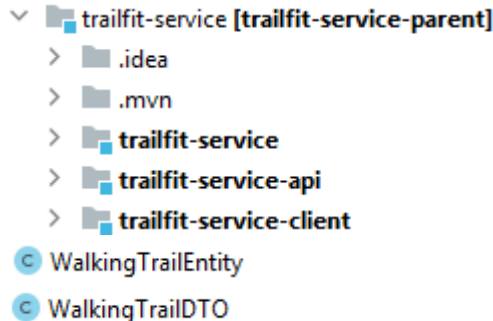
Dit is de innerlijke werking van de backend zelf. Hierin zitten de entiteiten, repositories, mappers, services enz.

- Een trailfit-service-api

Hier zit alles in wat te maken heeft met de opgehaalde data, bij dit project waren dat enkel de DTO's.

- Een trailfit-service-client

Dit project wordt gebruikt als de gebruiker de api met een post request wil aanspreken en was hier dus niet van toepassing.



8.5. MapStruct

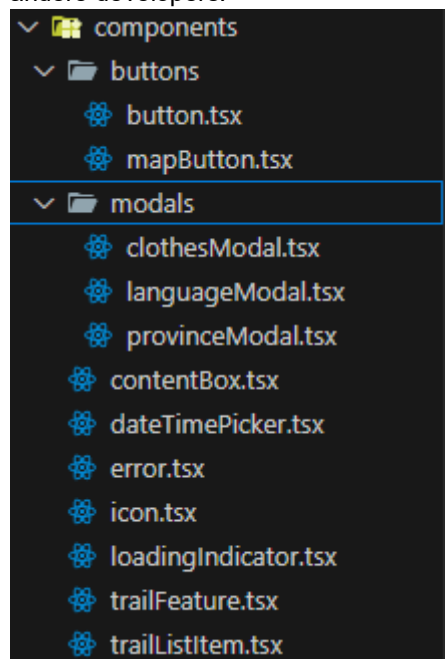
Ik heb in de mappers gebruik gemaakt van MapStruct, dit is een tool dat de meeste attributen automatisch mapt. Ze hadden dit vanuit Yonderland aangeraden en in ben er aangenaam door verrast.

```
@Mapping(target = "name", source = "translations", qualifiedByName = "mapNameTranslation")
@Mapping(target = "provinceName", source = "province.translations", qualifiedByName = "mapProvinceName")
@Mapping(target = "wheelchairAccessibility", source = "wheelchairAccessibility.translations", qualifiedByName = "mapAccessibility")
@Mapping(target = "strollerAccessibility", source = "strollerAccessibility.translations", qualifiedByName = "mapAccessibility")
WalkingTrailOverviewDTO toDto(WalkingTrailEntity walkingTrail, @Context Long languageId);
```

Zoals je hier ziet, zijn er maar enkele attributen die ik specifiek moet vernoemen, de rest wordt automatisch gemapt door MapStruct.

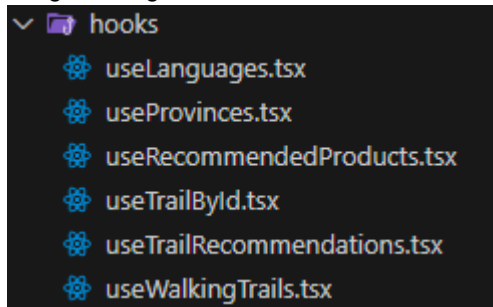
8.6. Componenten

In de frontend, heb ik zoals we op school ook altijd gedaan hebben, gebruik gemaakt van veel kleinere componenten. Dit zorgt dat de componenten herbruikbaar zijn, maar dit leidt ook tot duidelijkere code voor andere developers.



8.7. API requests

Net zoals voor de componenten, heb ik voor de API requests ook aparte bestanden aangemaakt. Zo kunnen ook deze hergebruikt worden en wordt het request niet overal voluit geschreven. Dit werd ook aangemoedigd vanuit Yonderland zelf.



Dit is een voorbeeld van een van de requests:

```

import { useQuery } from "@tanstack/react-query"; // React Query hook for data fetching
import { API_BASE_URL } from "@config/config"; // Base API URL for backend calls
import { WalkingTrailOverviewDTO } from "@types/types"; // Trail overview type definition

// Hook to fetch walking trails, optionally filtered by language and province
export const useWalkingTrails = (
  languageId?: number,
  provinceId?: number
) => {
  return useQuery<WalkingTrailOverviewDTO[], Error>({
    // Unique query key depends on language and province to manage caching
    queryKey: ["walkingtrails", languageId, provinceId],

    queryFn: async () => {
      // Require languageId to perform query
      if (!languageId) throw new Error("Language ID is required");

      // Choose URL based on whether provinceId is provided
      const url = provinceId
        ? `${API_BASE_URL}/walkingtrail?provinceId=${provinceId}&languageId=${languageId}`
        : `${API_BASE_URL}/walkingtrail/overview?languageId=${languageId}`;

      // Fetch data from backend API
      const response = await fetch(url);

      // Throw error if response is not OK
      if (!response.ok) {
        const errorMessage = await response.text();
        throw new Error(`Failed to fetch trails: ${errorMessage}`);
      }

      const data = await response.json();

      // Validate that the response is an array (expected format)
      if (!Array.isArray(data)) {
        throw new Error("Invalid trail data: Expected an array");
      }

      return data;
    },

    enabled: !!languageId, // Enable query only if languageId is defined
    retry: 1, // Retry once on failure (handles transient issues)
    staleTime: 1000 * 60 * 5, // Cache data for 5 minutes
    refetchOnWindowFocus: false, // Disable refetch on window focus to reduce unnecessary calls
  });
};

```

8.8. App layout

Voor de algemene layout heb ik gebruik gemaakt van de algemene documentatie van expo, zo heb ik een layout pagina waar de header gevormd wordt en waar de navigatie naar de andere pagina's uitgeschreven staat.


```

▼ app
  ▼ details
    [id].tsx
    _layout.tsx
    index.tsx

```

```

headerLeft: () => (
  <TouchableOpacity
    onPress={() => setLanguageModalVisible(true)}
    style={styles.iconButton}
    hitSlop={{ top: 16, bottom: 16, left: 16, right: 16 }}
  >
    <CustomIcon
      name="language"
      color={colors.asOrange100}
      size={ICON_SIZE.MEDIUM}
    />
  </TouchableOpacity>
),

// Province selector button on the right
headerRight: () => (
  <TouchableOpacity
    onPress={() => setProvinceModalVisible(true)}
    style={styles.iconButton}
    hitSlop={{ top: 16, bottom: 16, left: 16, right: 16 }}
  >
    <CustomIcon
      name="funnel"
      color={colors.asOrange100}
      size={ICON_SIZE.MEDIUM}
    />
  </TouchableOpacity>
),

```

8.9. Vertaling statische data

De applicatie Als beschikbaar zijn in 5 talen. Voor de vertalingen van de dynamische data hebben we gebruik gemaakt van de language tabel in de backend zoals eerder vermeld. Natuurlijk moet alle statische data in de applicatie ook vertaald worden. Ik dan wat opties overwogen en samengezeten met de verantwoordelijke voor het frontend gedeelte in Yonderland en we hebben besloten i18Next te gebruiken.

```

return (
  <I18nextProvider i18n={i18n}>
    <QueryClientProvider client={queryClient}>
      <LanguageProvider>
        <ProvinceProvider>
          <RootLayoutContent />
        </ProvinceProvider>
      </LanguageProvider>
    </QueryClientProvider>
  </I18nextProvider>
);

```

De I18nextProvider maakt gebruik van bestanden voor elke taal om dan een bepaald woord te vertalen.

```

i18n.use(initReactI18next).init({
  debug: true, // Enable debug mode (set to false in production)
  fallbackLng: 'nl', // Fallback language if user's preferred language is unavailable
  supportedLngs: ['nl', 'fr', 'en', 'de', 'es'], // List of supported languages
  defaultNS: 'translation', // Default namespace for translations
  lng: 'nl', // Initial language to use
  resources: {
    nl: { translation: nl },
    fr: { translation: fr },
    en: { translation: en },
    de: { translation: de },
    es: { translation: es },
  },
  interpolation: {
    escapeValue: false, // React already escapes values to prevent XSS
  },
  react: {
    useSuspense: false, // Disable suspense to avoid loading issues in React Native
  },
});

```

Dit is een voorbeeld van een bestand voor een bepaalde taal:

```

{
  "accessibility": "Toegankelijkheid",
  "checkShop": "Check ons aanbod",
  "close": "Sluiten",
  "dateMaxLimit": "De weersgegevens zijn alleen nauwkeurig voor de komende 5 dagen...",
  "dateMinLimit": "Deze datum is al voorbij...",
  "details": "Details",
  "done": "Klaar",
  "description": "Beschrijving",
  "loading": "Laden...",
  "noSpecificClothing": "Geen aanbevelingen... Maar onze shop zit vol met kwaliteitsuitrusting!",
  "noTrailsAvailable": "Geen wandelroutes beschikbaar voor deze provincie",
  "pastDate": "Datum ligt in het verleden!",
  "provinceSelect": "Selecteer provincie",
  "provinces": "Provincies",
  "selectLanguage": "Selecteer taal",
  "shopShort": "Kopen",
  "showAllTrails": "Toon alle wandelroutes",
  "stroller": "Kinderwagen:",
  "suggestedClothes": "Aanbevolen kleding",
  "surfaces": "Ondergrond",
  "tooFarDate": "Te ver in de toekomst!",
  "trailNotFound": "Wandelroute niet gevonden",
  "trails": "Wandelroutes",
  "tryDifferentProvince": "Probeer een andere provincie",
  "wheelchair": "Rolstoel: "
}

```

Aan gezien de applicatie niet al te groot is en er dus niet super veel vertaald moet worden, leek ons dit de beste oplossing.

8.10. Achtergrondafbeeldingen + kaarten

De afbeeldingen voor de producten worden simpelweg gewoon van de site gehaald, maar dit ging moeilijk met de achtergrondafbeeldingen en kaarten voor de routes. De oplossing die ik gebruikt heb is om een bestand te maken met het pad naar de images in de frontend.

```
export const imageSources: Record<number, any> = {
  1: require('../assets/images/backgrounds/trail1.jpg'),
  2: require('../assets/images/backgrounds/trail2.jpg'),
  3: require('../assets/images/backgrounds/trail3.jpg'),
  4: require('../assets/images/backgrounds/trail4.jpg'),
  5: require('../assets/images/backgrounds/trail5.jpg'),
  6: require('../assets/images/backgrounds/trail6.jpg'),
  7: require('../assets/images/backgrounds/trail7.jpg'),
  8: require('../assets/images/backgrounds/trail8.jpg'),
  9: require('../assets/images/backgrounds/trail9.jpg'),
  10: require('../assets/images/backgrounds/trail10.jpg'),
  11: require('../assets/images/backgrounds/trail11.jpg'),
  12: require('../assets/images/backgrounds/trail12.jpg'),
};

export const mapImageSources: Record<number, any> = {
  1: require('../assets/images/maps/trail1.png'),
  2: require('../assets/images/maps/trail2.png'),
  3: require('../assets/images/maps/trail3.png'),
  4: require('../assets/images/maps/trail4.png'),
  5: require('../assets/images/maps/trail5.png'),
  6: require('../assets/images/maps/trail6.png'),
  7: require('../assets/images/maps/trail7.png'),
  8: require('../assets/images/maps/trail8.png'),
  9: require('../assets/images/maps/trail9.png'),
  10: require('../assets/images/maps/trail10.png'),
  11: require('../assets/images/maps/trail11.png'),
  12: require('../assets/images/maps/trail12.png'),
};
```

Dit is wel vervelend moest er een nieuwe wandelroute bijkomen, maar dit leek de beste en goedkoopste oplossing.

8.11. Navigatie naar Google Maps en Waze

Net zoals bij de componenten en de api requests heb ik ook de navigatie naar Google Maps en Waze in aparte bestanden gezet voor hergebruik en leesbaarheid van de code.

```
TS openGoogleMaps.ts
TS openWaze.ts
```

8.12. Kleurpallet

De kleurpallet van de applicatie mocht ik zelf kiezen, al Als ik wel huiskleuren gebruiken. Deze heb ik van Figma overgezet naar een bestand, zodat ik deze makkelijk kan gebruiken in de applicatie. Ik heb niet enkel de kleuren toegevoegd die ik heb gebruikt, maar in één keer alle kleuren. Als Yonderland dan ooit de kleurpallet willen veranderen, kan dit makkelijk gebeuren.

```
export default {  
  // Brand AS Adventure  
  // Green  
  asGreen50: '#F2F9F1',  
  asGreen100: '#E2F0E0',  
  asGreen200: '#C5E2C0',  
  asGreen300: '#A8D3A1',  
  asGreen400: '#8EC685',  
  asGreen500: '#70B765',  
  asGreen600: '#549C49',  
  asGreen700: '#3E7336',  
  asGreen800: '#294C24',  
  asGreen900: '#152612',  
  asGreen950: '#0B150A',  
  
  // Grey  
  asGrey50: '#F4F5F6',  
  asGrey100: '#E0EBEC',  
  asGrey200: '#D5DADC',  
}
```

9. Besluit

Uiteindelijk heb ik een werkende applicatie afgeleverd die alle originele functionaliteiten heeft. Gaandeweg het project zijn er nog enkele bijgekomen die de applicatie net dat tikkeltje meer geven. Natuurlijk zijn er nog altijd verbeteringen/veranderingen die aangebracht kunnen worden.

Zo staan de images voor de background van de walkingtrails momenteel in het front-end terwijl het misschien beter is om deze remote ergens op te slagen. Dit zorgt ervoor dat de app minder opslag inneemt.

De huidige kaart op het detailscherm is gewoonweg een foto, dit zou natuurlijk veel beter zijn als dit een echte kaart zij waarbij je kon inzoomen. Dit was helaas onmogelijk met de data (coördinaten) die ik had, want hiervoor waren véél meer coördinaten vereist.

Een goede extra functionaliteit zou bijvoorbeeld een user management system zijn. Gebruikers hebben momenteel geen account nodig, wat zorgt voor gemakkelijker gebruik, maar we hebben ook geen feedback of informatie over de gebruikers. Als de applicatie met accounts werken, dan konden we bijvoorbeeld data ophalen, bijvoorbeeld het aantal mensen dat naar de webshop doorklikt. Ook zouden we dan een feedbacksysteem op de wandelpaden kunnen implementeren waarbij de user dan de wandeling kan liken.

TrailFit is momenteel nog niet beschikbaar in de App of Play Store omdat deze nog op bepaalde dingen gecheckt Als worden, zowel op IT-vlak als op het business vlak.

LITERATUURLIJST

- **Confluence (documentatieplatform)**

In Yonderland maken ze gebruik van Confluence (documentatieplatform) voor documentatie, dus hier heb ik voor dingen zoals de opstart van een project en de configuratie van bepaalde dingen al veel info kunnen vinden. Ook staat de uitleg van het project en haar functionaliteiten hierop.

- **BitBucket**

Yonderland werkt met BitBucket als repository managementsysteem. Als ik dus voorbeelden van andere projecten wou zien, heb ik deze hierop kunnen vinden.

- **Mapstruct**

Op de site van Mapstruct heb ik kunnen vinden hoe een mapper die gebruik maakt van Mapstruct in elkaar zit.

- **Flyway (database migratietool)**

Op de site van RedGate Flyway (database migratietool) was te vinden hoe database migraties worden gedaan met behulp van Flyway (database migratietool) in Spring Boot.

- **Expo**

Aangezien ik een expo-router project gebruik voor mijn frontend, heb ik de opstart en verschillende andere documentatie op de expo site gebruikt.

- **React Native**

De frontend is geschreven in React Native, dus bij onzekerheden of gebrek aan kennis, ben ik de officiële site van React Native gaan raadplegen.

- **Stack Overflow (ontwikkelaarsforum)**

Tijdens het project ben ik op enkele errors gebotst die ik niet direct opgelost kreeg. Uiteindelijk heb ik de oplossing meestal op Stack Overflow (ontwikkelaarsforum) gevonden.

- **Collega's**

Bij vragen over de inhoud van het project of best practices binnen het bedrijf (die niet of Confluence (documentatieplatform) staan), ben ik een collega gaan raadplegen. Dit was meestal mijn stagebegeleider, ook kon ik hier terecht voor technische vragen, maar Als deze niet genoeg kennis hebben van een bepaald onderwerp, dan werd ik doorverwezen naar iemand in het juiste vakgebied.

- **ChatGPT (AI-hulpmiddel)**

Als ik op alle andere bronnen niet kunnen vinden wat ik zocht, dan schreef ik een prompt in ChatGPT (AI-hulpmiddel), die me dan Meestal wel verder kon helpen. Ik gebruikte ChatGPT (AI-hulpmiddel) vooral voor het vertalen of genereren van data. Om geen tijd te verliezen met repeterende taken heb ik ook gebruik gemaakt van ChatGPT (AI-hulpmiddel).