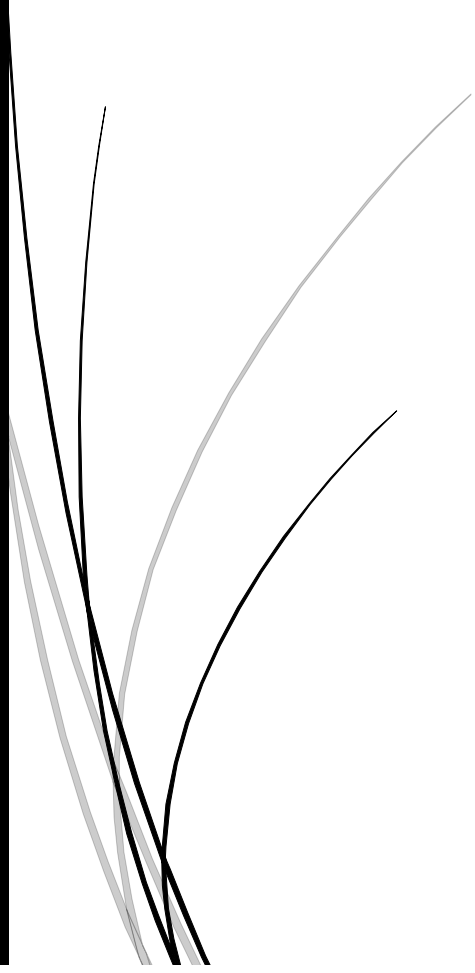
13-1-2017

Implementatie Ontwikkelstraat

IPSENH



RZN
REPOSITORY ZONDER NAAM

Versiebeheer

Versie	Datum	Omschrijving	Wie
0.1	28-11-2016	Opzet document	Anton
0.2	28-11-2016	Invulling Datamodel, back-end api, front end model	Anton
0.3	19-12-2016	Aanpassing model, beschrijving API en documentatie van de front-end	Anton
0.4	20-12-2016	Invulling gebruik externe API	Anton
0.5	24-12-2016	Invulling documentatie front-end	Anton
1.0	13-01-2017	Laatste controle en finalisering	Anton

Inhoud

Functionele Eisen

Angular2 Applicatie:

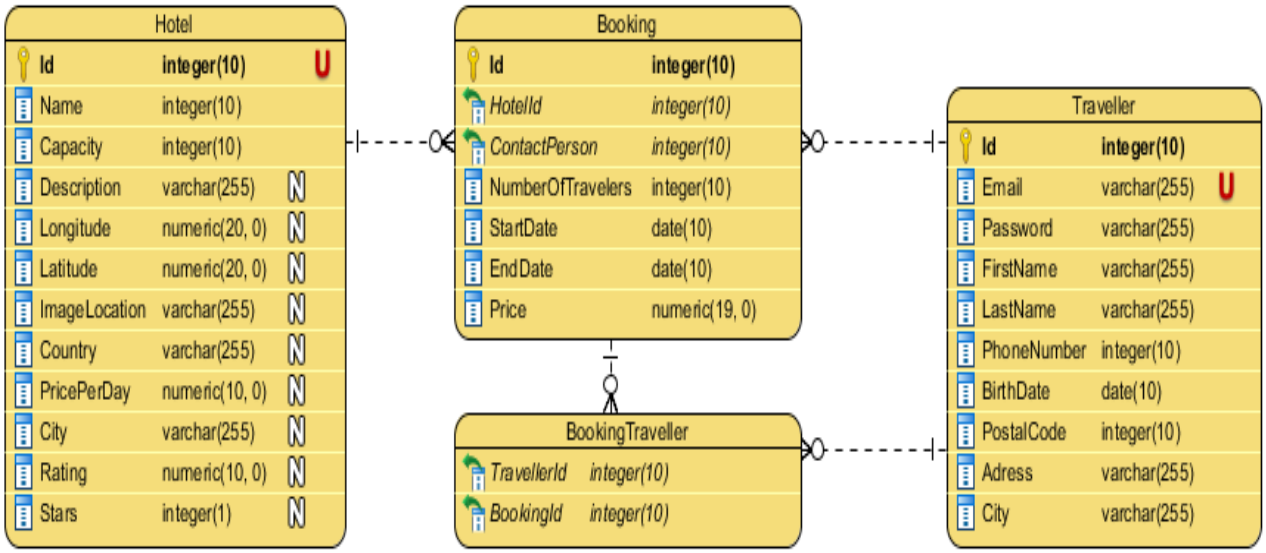
- Klant moet een boeking kunnen maken
- Klant moet een lijst van hotels in kunnen zien
- Klant moet kunnen registreren
- Klant moet zijn gegevens kunnen bijwerken

TravelPlanner Gegenereerde Applicatie:

- Medewerker moet een boeking kunnen maken voor een klant

Datamodel

Het datamodel is tijdens de ontwikkeling een aantal keer bijgesteld. Dit is het datamodel zoals die uiteindelijk geïmplementeerd hebben.



Documentatie Front-End

Implementatie

De front-end is gebouwd met Angular2. Wij hebben voor de verschillende schermen gebruik gemaakt van componenten en modules. De opbouw hiervan is terug te zien in het domein model. De functies zijn allemaal vrij triviaal.

Tijdens het programmeren hebben wij erop gelet dat we niet te lange, complexe methodes schrijven, zodat we kunnen voldoen aan een maximale cyclus complexiteit van 8.

Documentatie Externe API's

De front-end maakt gebruik van 2 externe API's : Google Maps API en een Temperature API.

Google Maps:
De google maps API hebben wij geimplementeerd met hulp van de Angular2 component. Deze API hebben wij gebruikt in de detail pagina van de hotels. Hier wordt aan de hand van een longitude en een latitude de locatie van het hotel weergeven.

Temperature API:
We maken gebruik van deze weer API : https://openweathermap.org/api
De temperatuur API gebruiken we ook in de hotel-detail pagina. Hier halen we aan de hand van een longitude en een latitude de gegevens op: https://openweathermap.org/current#geo

We ontvangen dan de temperatuur, windrichting, omschrijving, luchtdruk en lucht vochtigheid.

Het antwoord dat we van de server krijgen hebben we verwerkt in onze applicatie. Dit is een voorbeeld antwoord dat we krijgen:

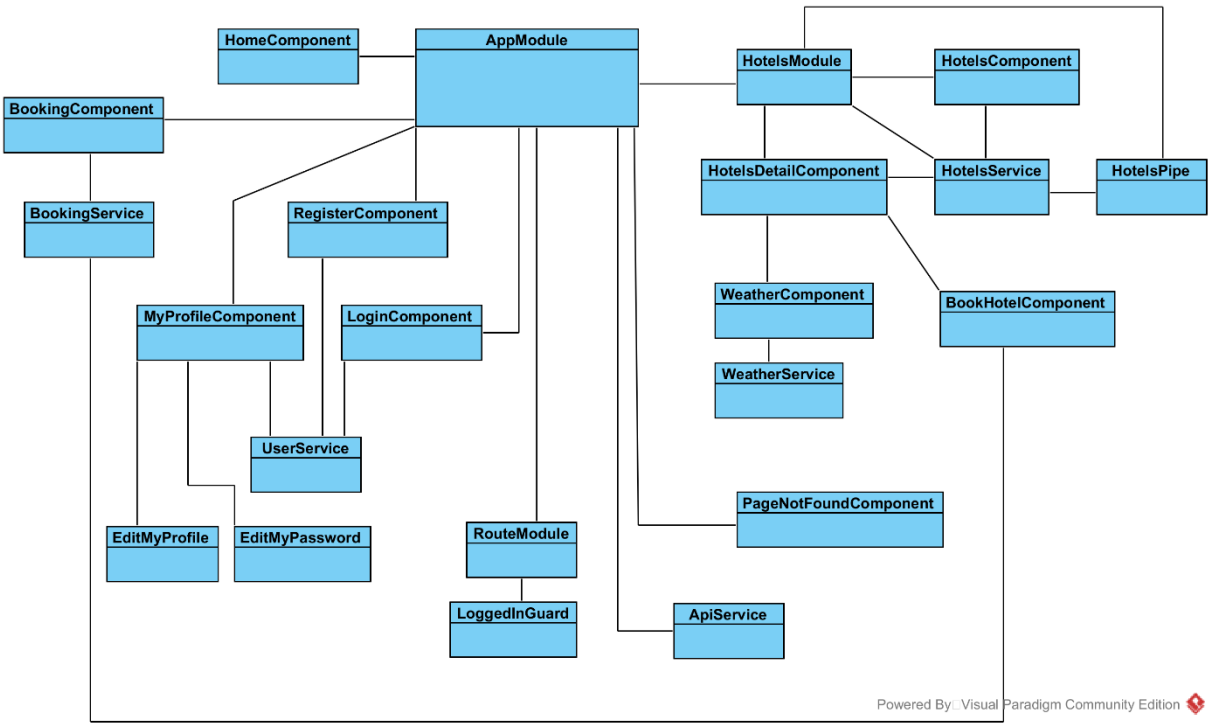
```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JP","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast
clouds","icon":"04n"}],
"main":{"temp":289.5,"humidity":89,"pressure":1013,"temp_min":287.04,"temp_
max":292.04},
"wind":{"speed":7.31,"deg":187.002},
"rain":{"3h":0},
"clouds":{"all":92},
"dt":1369824698,
"id":1851632,
"name":"Shuzenji",
"cod":200}
```

Testen

Om de applicatie te kunnen testen hebben we een SailsJS API opgezet. Deze API kunnen wij makkelijk de gegevens van bepalen die hij bevat wanneer hij wordt gestart. Zo kunnen wij eenvoudig end-2-end tests uitvoeren. Ook was dit nodig tijdens het ontwikkelen omdat de echte back-end niet goed kon worden aangesloten. Het testen gebeurt op de buildserver. We hebben Unit tests voor de back-end gemaakt, en e2e tests voor de front-end.

Domeinmodel

Om de applicatie te realiseren zijn we begonnen met te bedenken welke onderdelen onze applicatie nodig had. Hiervoor hebben we een domeinmodel opgesteld.



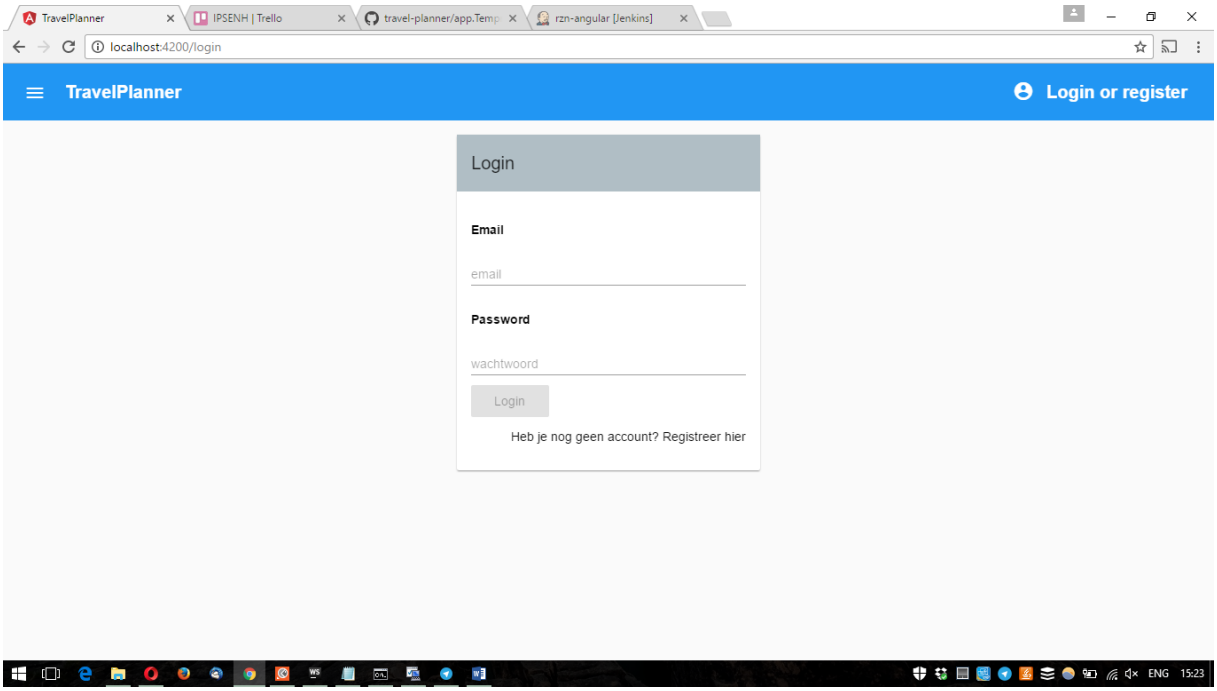
Interceptor

Er was een slordigheid bij het bouwen van de authenticatie van de applicatie. We moesten in iedere service een bepaald stuk code herhalen om de authenticatie toe te voegen en om de http request af te handelen. Om dit op te lossen heeft Anton een soort “Interceptor” geschreven. Dit is een klasse die uiteindelijk de call uitvoert. Dit hebben we op deze manier gedaan zodat er op 1 plaats de authenticatie kan worden toegevoegd en de http request kan worden afgehandeld om de code duplicatie te reduceren. In het UML diagram is deze terug te zien als “API Service”. Alle andere services maken gebruik hiervan om de uiteindelijke call te maken.

Ontwerp

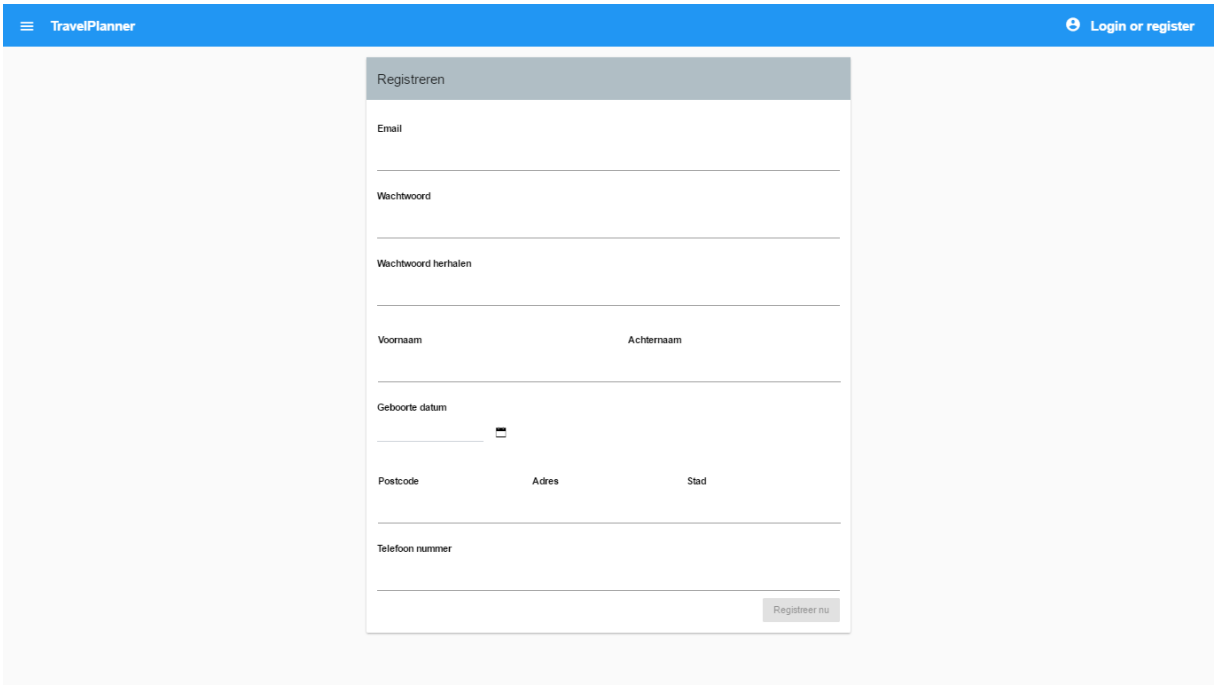
Loginscherm

Hier logt de gebruiker in met zijn gebruikersnaam (emailadres), of registreert de gebruiker een account.



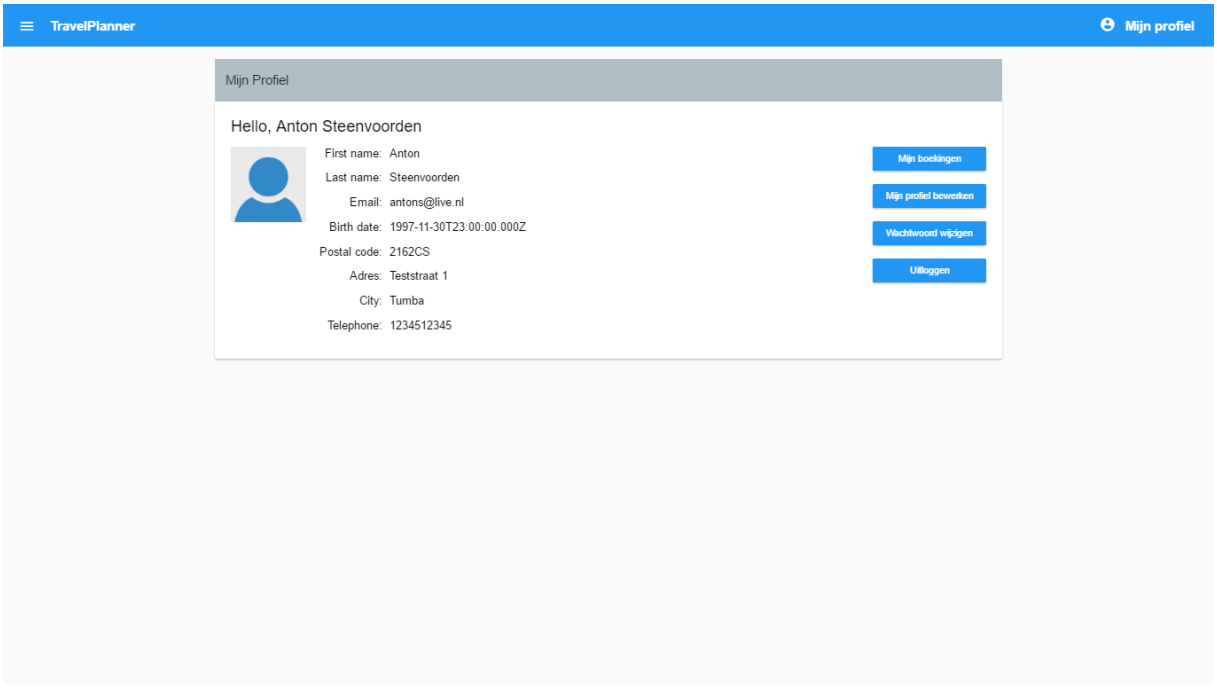
Registreer scherm

Hier registreert de gebruiker een nieuw account.



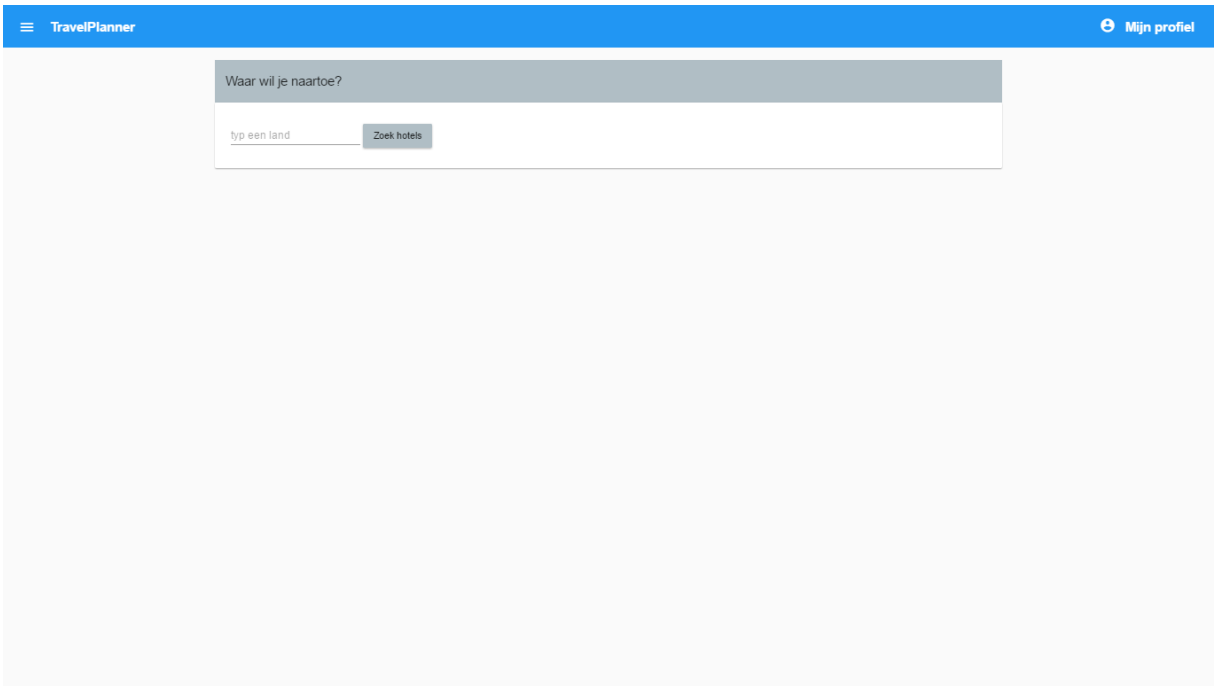
Mijn profiel

Als je bent ingelogd kun je jouw gegevens zien. Hier kun je ook uitloggen en gegevens wijzigen. Vanaf hier kan je ook naar jouw boekingen gaan



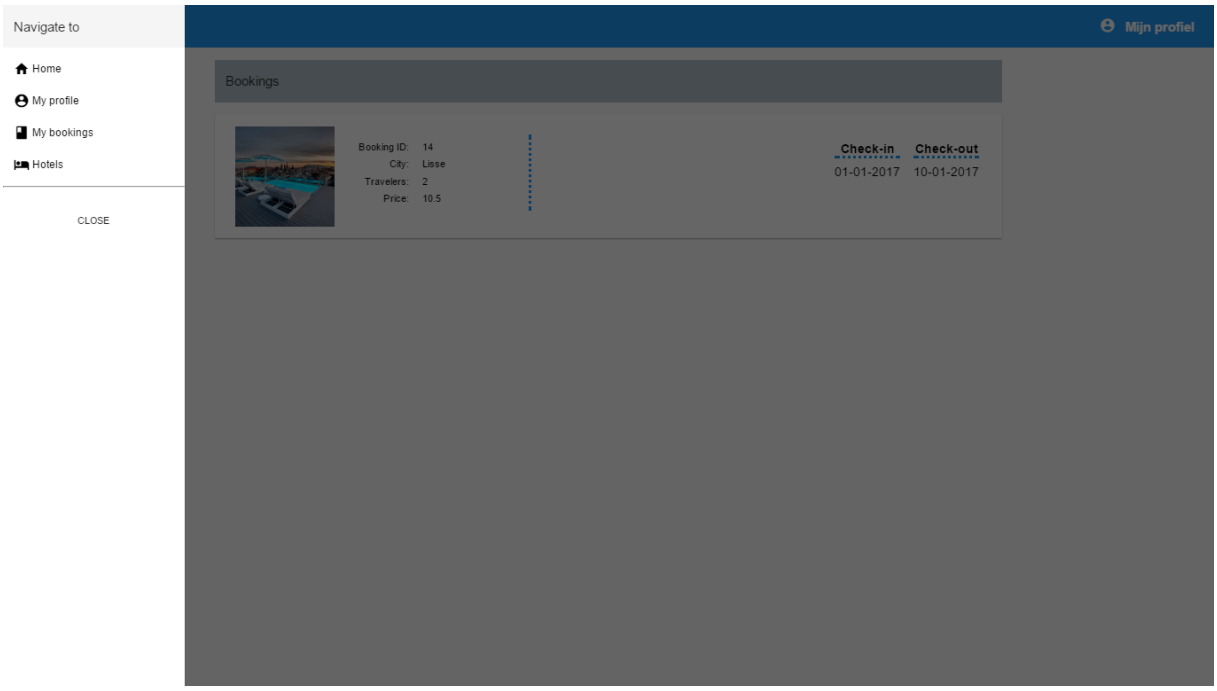
Dashboard

Hier vul je een land in waar je naartoe wilt. Je krijgt dan alleen de hotels uit dat land die in de database zitten te zien.



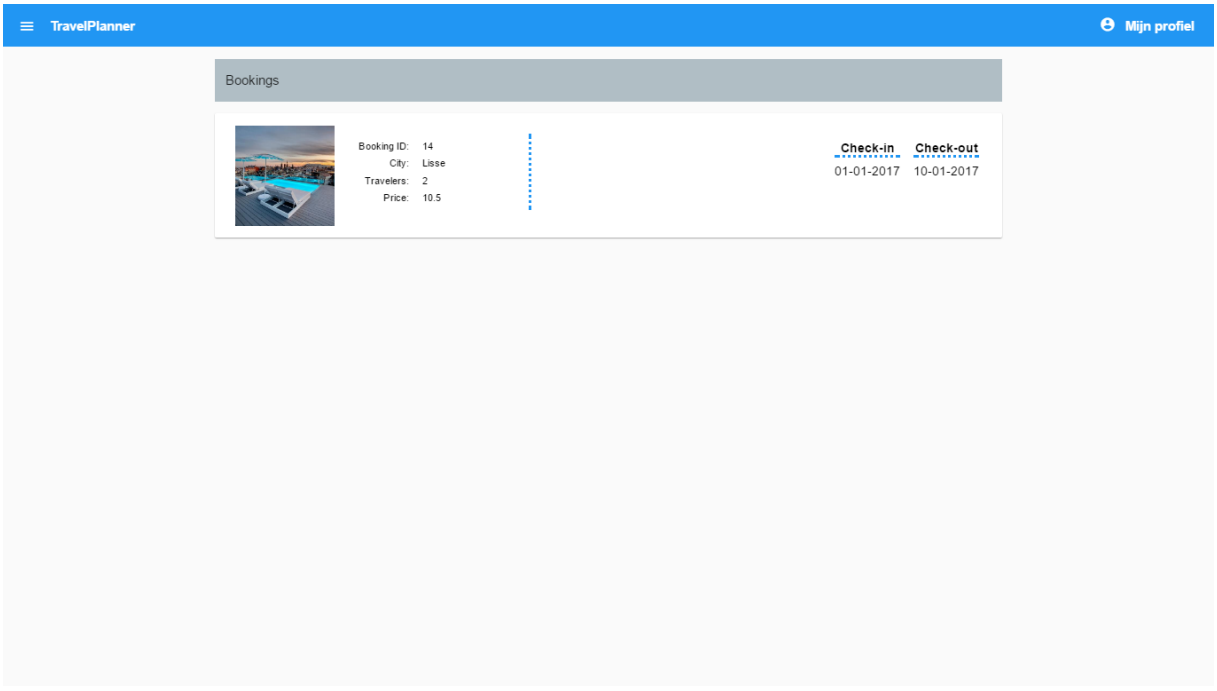
Navigatie

Menu waarmee je naar de verschillende pagina's kunt



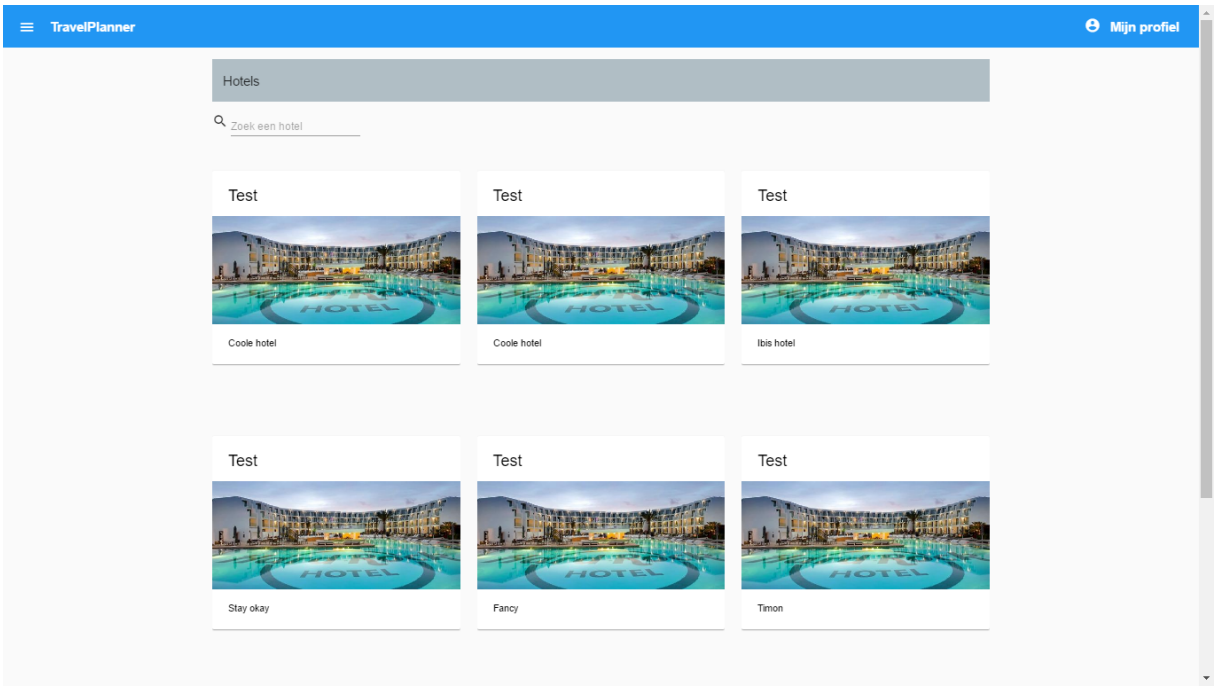
Mijn boekingen

Hier kun je al jouw boekingen bekijken



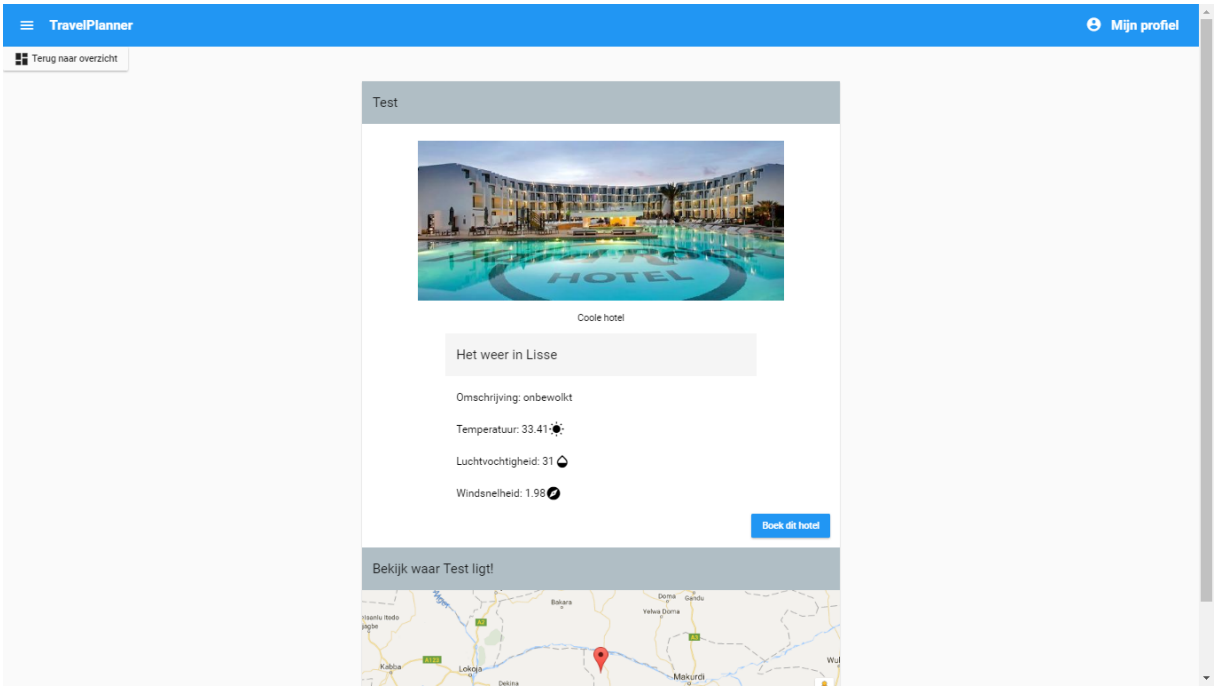
Hotels

Hier is een overzicht van de hotels die je kunt boeken



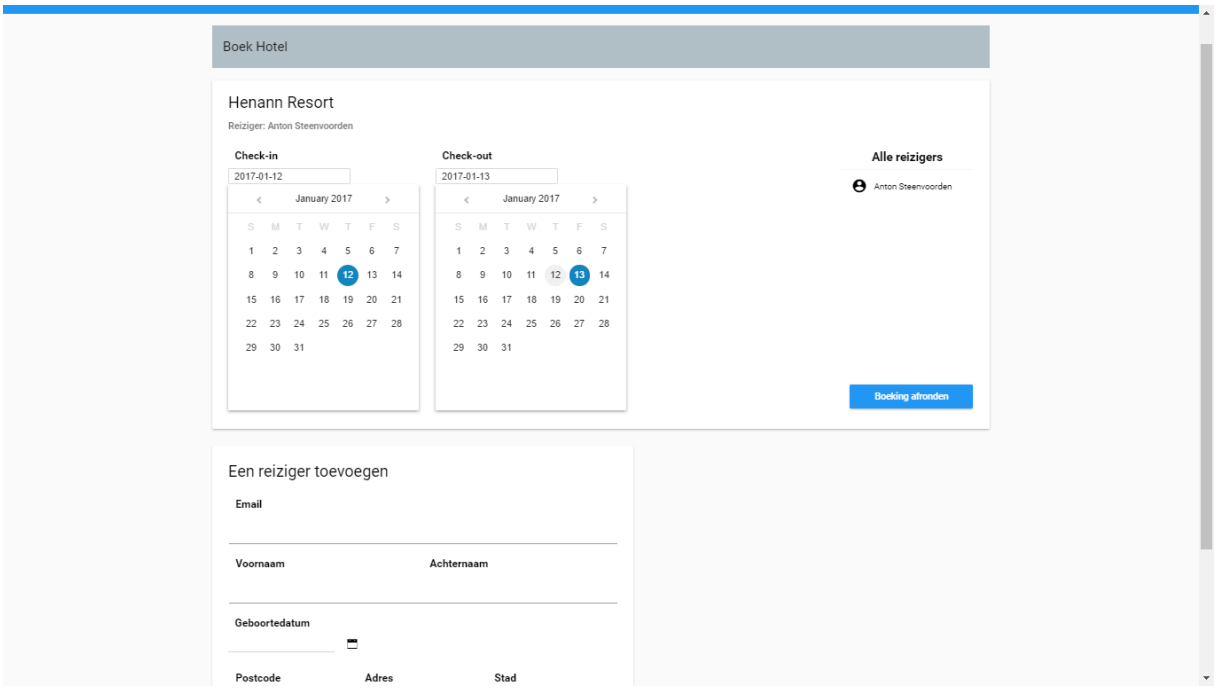
Hotel detailpagina

Hier zie je de gegevens van het hotel, vanaf hier kan je ook boeken.



Hotel boeken

Hier kun je het hotel boeken en met wie je gaat.



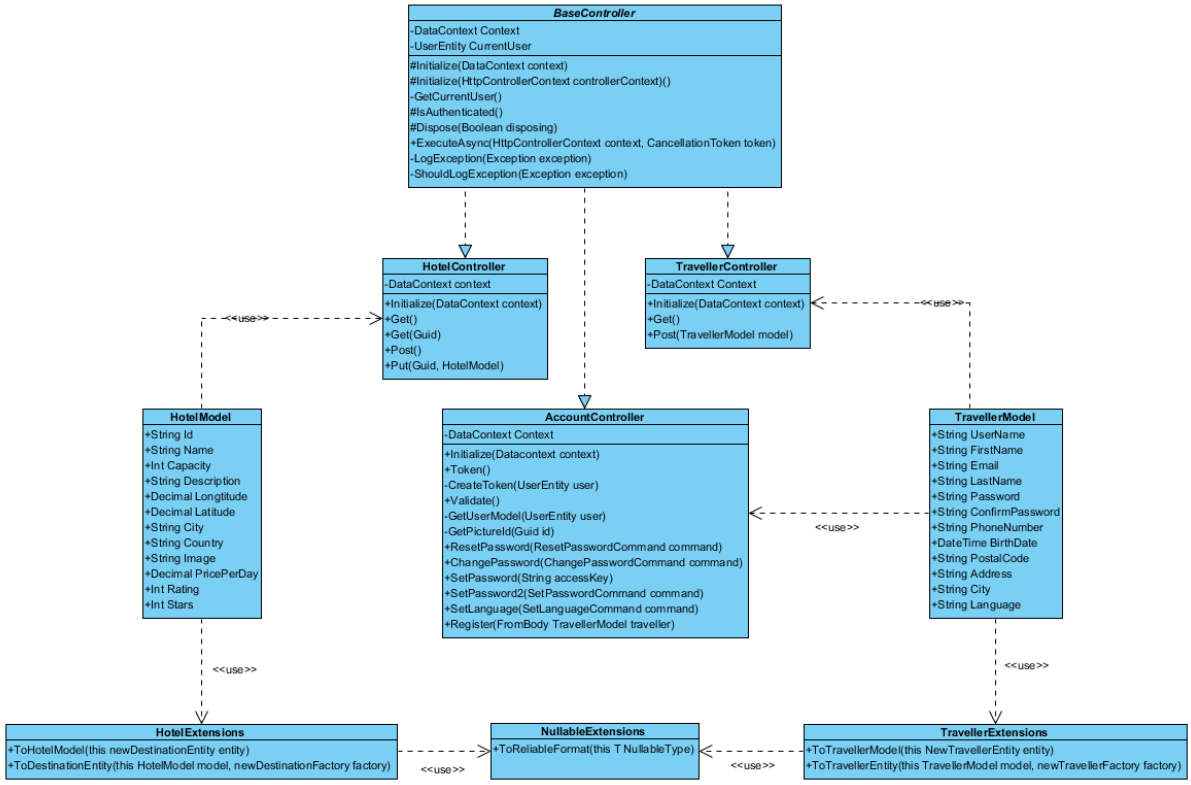
Documentatie Back-end / API

Implementatie

De back-end is op de gegeneerde code doorgebouwd van de 42TravelPlanner die wij hebben gemaakt via de designer. Het datamodel is een representatie van de entiteiten die wij hebben toegevoegd met hun attributen. Om de many-to-many relatie te ondersteunen hebben we een tussen-entiteit toegevoegd.

Om de back-end beschikbaar te maken via http en daarbij de data terug te geven in JSON formaat hebben we gebruik gemaakt van het ASP.NET Web API framework. We hebben onze toevoegingen vooral in de Controllers en Models mappen van het App.RestAPI project gedaan. Hier hebben wij de end-points afgehandeld en hier handelen wij autorisatie en het teruggeven van de data af.

UML van de back-end



Beschrijving

In de back-end hebben we een klasse Base Controller waar de rest van de controllers overerven. Deze basecontroller bevat de benodigdheden voor het afvangen van http calls en teruggeven van data.

End Points + Toegankelijkheid

De API ondersteund de volgende endpoints:

Het basis pad is: /api

/hotels - GET, POST

- Opvragen van alle hotels, nieuw hotel aanmaken


```
[
 {
  "Name": "test hotel numero 1",
  "Id": "e441340a-26fa-4823-aea4-da6de9994087",
  "CreatedOn": "2017-01-12T15:39:12.093",
  "ModifiedOn": "2017-01-12T15:39:12.093",
  "Status": false,
  "OwnerId": "1baee08d-7102-4190-8a19-4f25fce7be26",
  "ModifiedById": "1baee08d-7102-4190-8a19-4f25fce7be26",
  "CreatedById": "1baee08d-7102-4190-8a19-4f25fce7be26",
  "OwningBUId": "f048f3cd-ca92-46dd-ad82-58d683959d0d",
  "Capacity": 40,
  "Description": "geen zin in moeilijke beschrijvingen",
  "Imagelocation": "",
  "Country": "Holland",
  "Priceperday": 5,
  "City": "Sassum",
  "Rating": 3,
  "Stars": 5,
  "Latitude1": 30,
  "Longitude1": 46,
  "SelectionData": ""
 }
]
```

/hotels/{id}- GET, PUT

- Wijzigen bestaande hotels, specifieke opvragen (detailpagina)


```
{
   "Name": "test hotel numero 1",
   "Id": "e441340a-26fa-4823-aea4-da6de9994087",
   "CreatedOn": "2017-01-12T15:39:12.093",
   "ModifiedOn": "2017-01-12T15:39:12.093",
   "Status": false,
   "OwnerId": "1baee08d-7102-4190-8a19-4f25fce7be26",
   "ModifiedById": "1baee08d-7102-4190-8a19-4f25fce7be26",
   "CreatedById": "1baee08d-7102-4190-8a19-4f25fce7be26",
   "OwningBUId": "f048f3cd-ca92-46dd-ad82-58d683959d0d",
   "Capacity": 40,
   "Description": "geen zin in moeilijke beschrijvingen",
   "Imagelocation": "",
   "Country": "Holland",
   "Priceperday": 5,
   "City": "Sassum",
   "Rating": 3,
   "Stars": 5,
   "Latitude1": 30,
   "Longitude1": 46,
   "SelectionData": ""
 }
```


/bookings - GET, POST

- Opvragen alle namen (id's)
- Krijgt traveller (id) en hotel (id) mee


```
[
 {
   "Name": "",
   "Id": "f25fa31c-fb9d-4e48-9d93-4fdfb99d5255",
   "CreatedOn": "2017-01-12T16:33:15.553",
   "ModifiedOn": "2017-01-10T11:56:19.18",
   "Status": false,
   "OwnerId": "75da64e1-d8a7-43cf-b29b-05c287c49450",
   "ModifiedById": "75da64e1-d8a7-43cf-b29b-05c287c49450",
   "CreatedById": "75da64e1-d8a7-43cf-b29b-05c287c49450",
   "OwningBUId": "00000000-0000-0000-0000-000000000000",
   "HotelId1": "e441340a-26fa-4823-aea4-da6de9994087",
   "ContactpersonId": "896f7e62-89a0-4cc9-951a-438a641747dc",
   "Numberoftravelers": 2,
   "Startdate": "2017-01-18T00:00:00",
   "Enddate": "2017-01-21T00:00:00",
   "Price": 10.5,
   "TravellerId": "00000000-0000-0000-0000-000000000000",
   "SelectionData": ""
 }
]
```

/bookings/{id}- GET, PUT

- Krijgt detailpagina

- Wijzigingen in boeking


```
[
 {
  "id": "f25fa31c-fb9d-4e48-9d93-4fdfb99d5255",
  "contactPerson": "896f7e62-89a0-4cc9-951a-438a641747dc",
  "numberOfTravellers": 2,
  "hotelID": "e441340a-26fa-4823-aea4-da6de9994087",
  "startDate": "2017-01-18T00:00:00",
  "endDate": "2017-01-21T00:00:00",
  "price": 10.5,
  "createdAt": "2017-01-12T16:33:15",
  "updatedAt": "2017-01-10T11:56:19"
 }
]
```


/account/validate – GET
- Hiermee krijg je de user gegevens terug. Deze call wordt gebruikt om mee in te loggen.


```
{
 "userName": "Administrator",
 "firstName": "contactFirstName",
 "email": "",
 "lastName": "contactLastName",
 "password": "VeryGoodPass!",
 "phoneNumber": "00",
 "birthDate": "0001-01-01T00:00:00Z",
 "postalCode": "",
 "address": "",
 "city": ""
}
```

/travellers/ - GET
- Haalt alle travellers op, mag alleen admin


```
[
 {
  "Name": "",
  "Id": "896f7e62-89a0-4cc9-951a-438a641747dc",
  "CreatedOn": "2017-01-12T16:32:27.483",
  "ModifiedOn": "2017-01-12T16:32:27.483",
  "Status": false,
  "OwnerId": "00000000-0000-0000-0000-000000000000",
  "ModifiedById": "00000000-0000-0000-0000-000000000000",
  "CreatedById": "00000000-0000-0000-0000-000000000000",
  "OwningBUId": "00000000-0000-0000-0000-000000000000",
  "Email": "test@test.nl",
  "Password": "VeryGoodPass!",
  "Firstname": "contactFirstName",
  "Lastname": "contactLastName",
  "Birthdate": "0001-01-01T00:00:00Z",
  "Phonenumber": 0,
  "Postalcode": "2162CS",
  "Adress": "Govert testrtraat 2",
  "City": "Lisse",
  "SelectionData": ""
 }
]
```

Toevoegen nieuwe componenten

Front-end

Om nieuwe componenten aan de front-end toe te voegen is niet heel erg ingewikkeld. Hier kan gebruik worden gemaakt van de Angular-CLI. Deze Command Line Interface biedt een aantal generatie commando's aan die kunnen worden gebruikt om het skelet op te zetten van een gewenste feature. https://cli.angular.io Je moet vervolgens zelf de implementatie uitschrijven maar omdat alles heel modulair is opgebouwd is dit erg prettig om te doen. Een voorbeeld van het toevoegen van een nieuwe Angular2 Module is: "ng generate module voorbeeldModule" hier maakt angular-cli dan een mapje voor en zorgt dat in de files de juiste dingen geïmporteerd zijn zodat er gelijk kan worden geprogrammeerd aan de nieuwe module.

Back-end

Om aan de back-end nieuwe componenten toe te voegen kan gebruik worden gemaakt van de designer van de 42WindMills. Zij genereren dan nieuwe database velden en bijhorende Data Layer Logica. De RESTAPI moet dan nieuwe controllers en models aanmaken en deze kunnen vervolgens worden aangesloten op de gegenereerde code. Om de API uit te breiden moet in het project App.RestAPI nieuwe controllers of functies worden toegevoegd. Dit is waar de API alle logica en calls afhandelt.