ZIGGY MOENS – 2B1

SNEAKER STORE

WEBAPPLICATIES IV

ZIGGY MOENS
2B1 - 201758095
Ziggy.moens@student.hogent.be

Inhoudsopgave

Inloggegevens

Admin

- admin@hotmail.com
- P@ssword1

Gebruiker

- customer@hotmail.com
- P@ssword1

Beschrijving van de applicatie

Voor dit project heb ik een webapplicatie geschreven dat een fictieve schoenenwinkel moet voorstellen namelijk Sneaker Store.
In de Sneaker Store bevinden zich tal van verschillende schoenen die men kan bekijken. Via detailpagina's is het mogelijk om de huidige stockaantallen te zien en de schoenen in 360 graden te bekijken.
Elke schoen is voorzien van een barcode wat het voor de winkel een stuk makkelijker maakt om de schoenen te stockeren en terug te vinden in de winkel.
Ook hebben mensen de mogelijkheid om extra schoenen toe te voegen, tijdens het proces van toevoegen kunnen de gebruikers een live preview zien van wat opgeslagen zal worden in de databank.
Voor het gebruik van de applicatie is een account verplicht, gebruikers die dit nog niet hebben kunnen zich registreren en zullen nadien de site kunnen gebruiken.

Extra technologieën

Google Maps (https://github.com/angular/components/releases/tag/9.0.0-rc.0)

Aan de hand van de Google Maps Component heb ik de hoofdzetel van de Sneaker Store weergegeven. Dit met een custom map met aangepast kleuren thema en special effects op de marker.
De installatie van deze component kan via *npm install @angular/google-maps*
Deze map implementeert ook de Google Maps Javascript API, meer info hierover vindt u hier: https://developers.google.com/maps/documentation/javascript/reference/

De Google maps implementatie is hier terug te vinden: http://localhost:4200/about

Barcode (https://github.com/efgiese/ngx-barcode6)

Aan de hand van deze barcode component heb ik de style codes van de sneakers weergegeven. Dit is een makkelijk iets om snel in een systeem iets terug te vinden.
Voor het installeren van de component kan je volgend commando gebruiken: *npm install --save ngx-barcode6 jsbarcode@3.11.0*
Bij deze installatie hebben we echter ook Jsbarcode (https://github.com/lindell/JsBarcode) nodig, deze installeren we met *npm install --save jsbarcode*

De barcodes kunnen in allerlei vormen worden gegenereerd maar ik deze situatie heb ik geopteerd om CODE128 te gebruiken aangezien deze alle tekens ondersteunt.

De barcodes zijn terug te vinden op http://localhost:4200/sneaker/scan of op een van de detailpagina's van de sneakers bv. http://localhost:4200/sneaker/detail/1

360 graden foto’s

Aan de hand van de mat-slider component heb ik een soort van 360 graden beeld laten maken van de sneakers. Dit door de gepaste afbeelding op te halen van het internet.
Dit is niet de meest optimale manier om dit op te lossen, indien dit project verder zou uitgewerkt worden zou hier best geopteerd worden op met observables te werken.

De 360 graden foto’s zijn terug te vinden op de detailpagina's van de sneakers zoals bv. http://localhost:4200/sneaker/detail/1

Structuur van de app

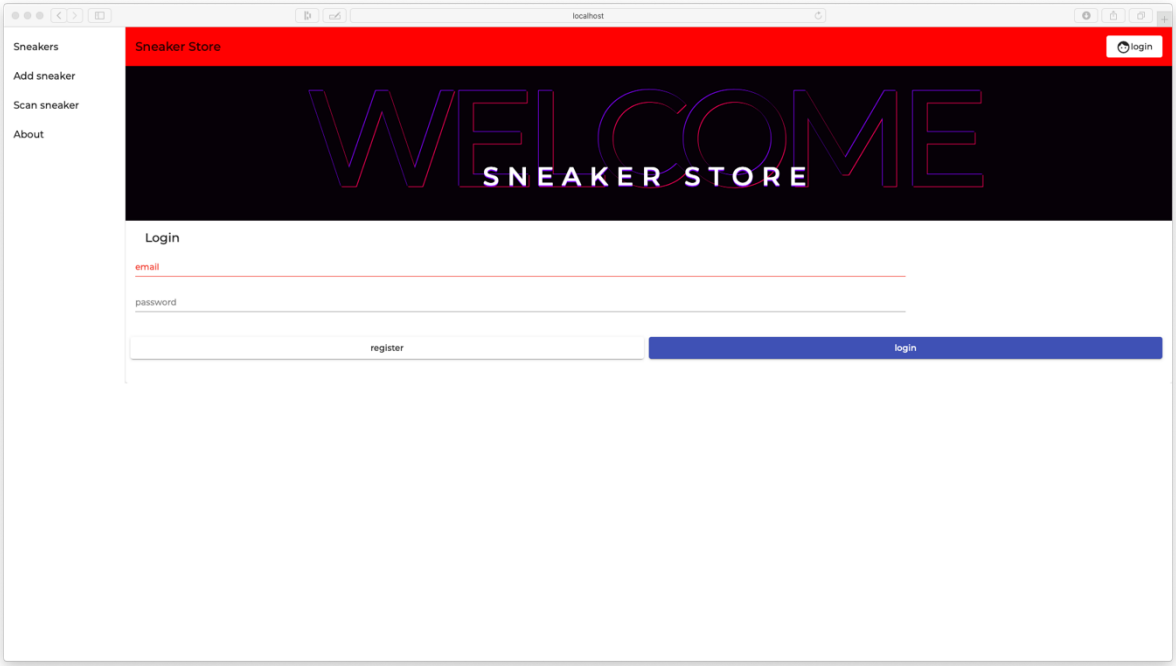
1920-b1-fe-ziggymoens/storeapp/src

```
├── app
|   ├── SelectivePreloadStrategy
|   ├── about-us
|   |   └── about-us.component
|   ├── app-routing.module
|   ├── app.component
|   ├── app.module
|   ├── http-interceptors
|   |   ├── AuthenticationInterceptor
|   |   └── index
|   ├── main-nav
|   |   └── main-nav.component
|   ├── material
|   |   └── material.module
|   ├── page-not-found
|   |   └── page-not-found.component
|   ├── sneaker
|   |   ├── SneakerResolver
|   |   ├── add-sneaker
|   |   |   └── add-sneaker.component
|   |   ├── brand
|   |   |   └── brand.component
|   |   ├── dataServices
|   |   |   ├── brand-data.service
|   |   |   ├── mock-sneakers
|   |   |   └── sneaker-data.service
|   |   ├── filters
|   |   |   ├── barcode-filter.pipe
|   |   |   ├── brand-filter.pipe
|   |   |   └── sneaker-filter.pipe
|   |   ├── models
|   |   |   ├── brand.model
|   |   |   ├── sneaker.model
|   |   |   └── stock.model
|   |   ├── sneaker
|   |   |   └── sneaker.component
|   |   ├── sneaker-detail
|   |   |   └── sneaker-detail.component
|   |   ├── sneaker-list
|   |   |   └── sneaker-list.component
|   |   ├── sneaker-scan
|   |   |   └── sneaker-scan.component
```

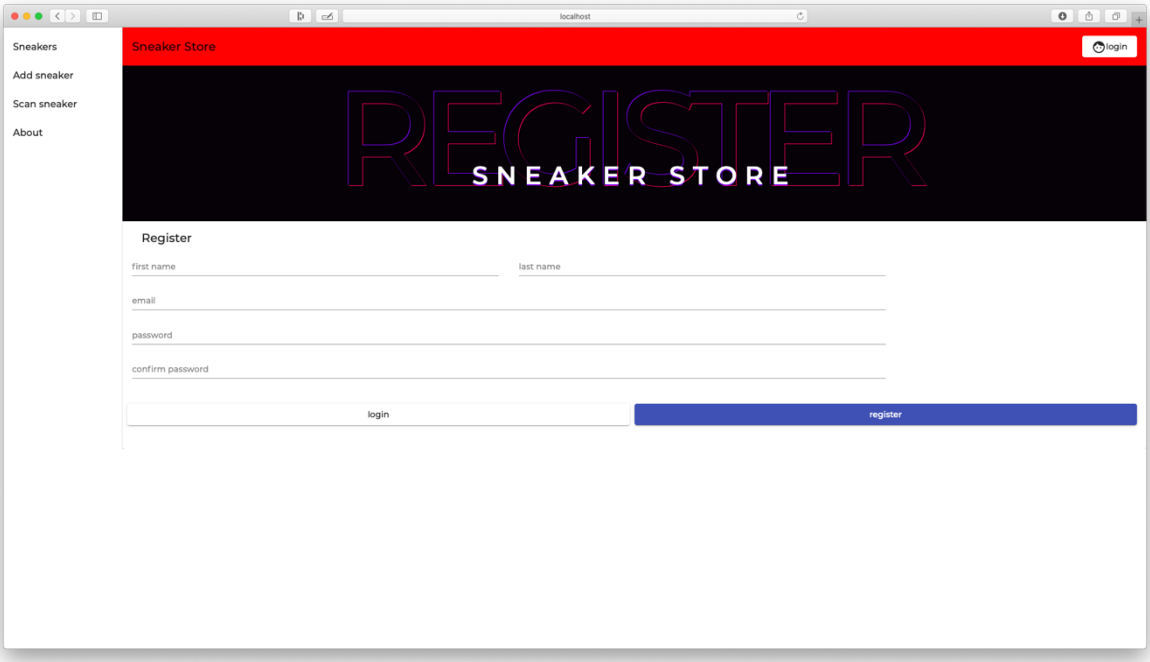
```
|   |   ├── sneaker.module
|   |   └── stock-sales
|   |       └── stock-sales.component
|   └── user
|       ├── auth.guard
|       ├── authentication.service
|       ├── login
|       |   └── login.component
|       ├── register
|       |   └── register.component
|       └── user.module
├── assets
|   └── images
|       ├── brands
|       |   ├── adidas.jpg
|       |   ├── airjordan.jpg
|       |   ├── default.png
|       |   ├── nike.jpg
|       |   ├── offwhite.jpg
|       |   └── supreme.jpg
|       └── sneakers
|           └── default.png
├── environments
|   └── environment
├── favicon.ico
├── index.html
├── main.ts
├── polyfills.ts
├── styles.css
└── test.ts

25 directories, 94 files
```

Screenshots van de applicatie

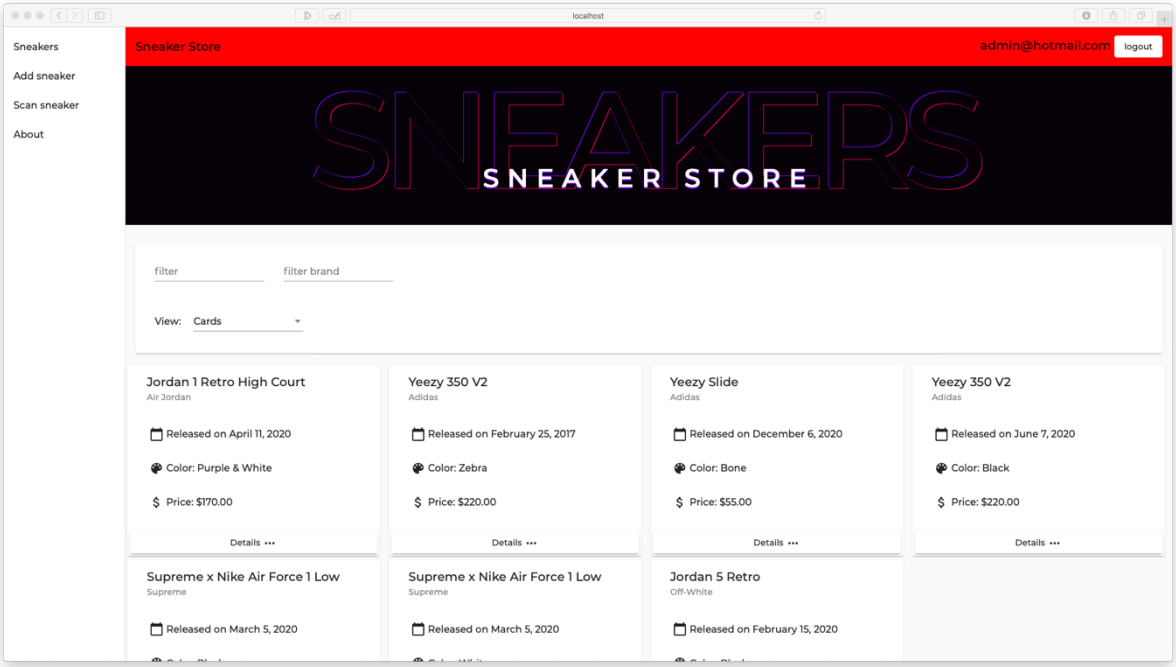
Login



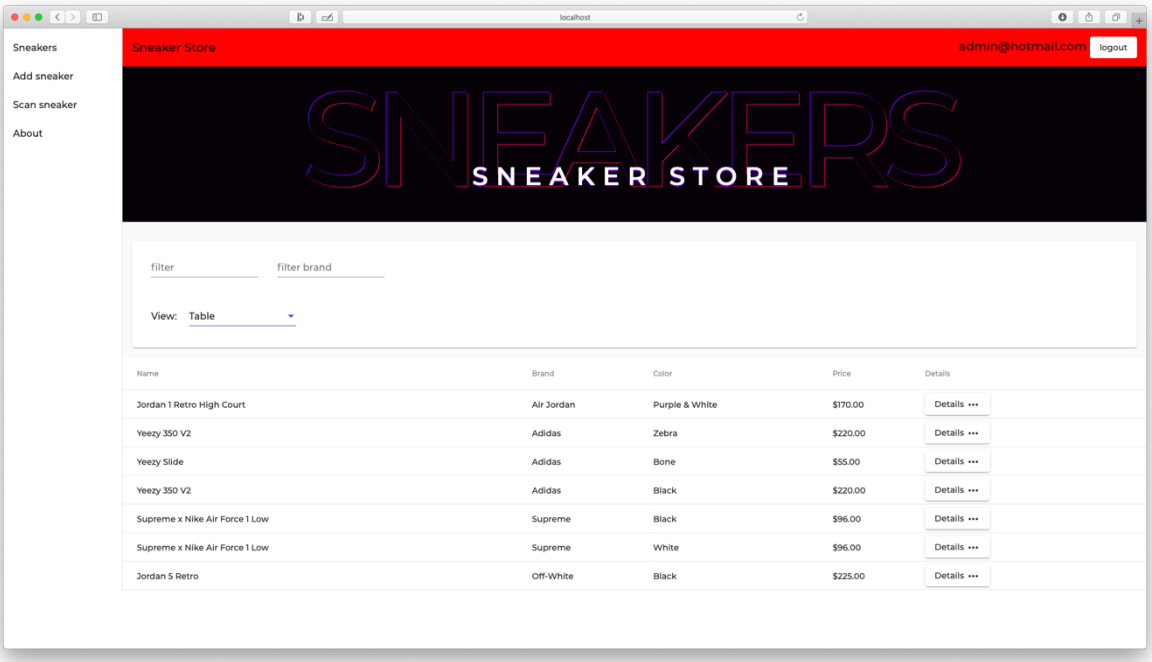
Register



Index (met cards als view)

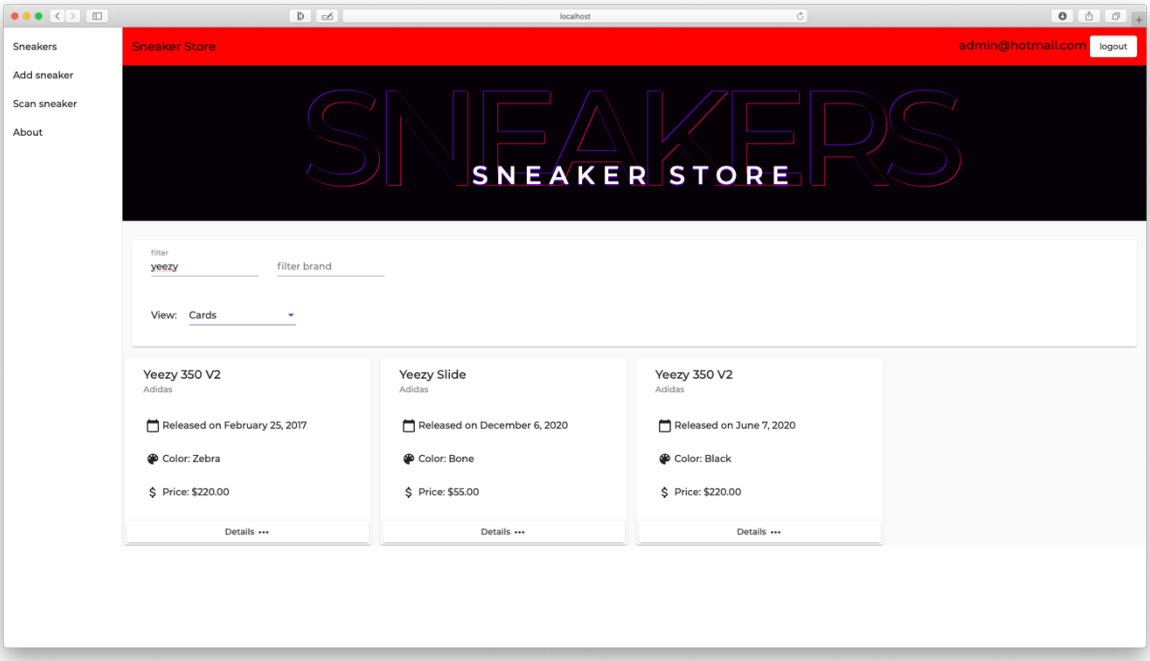


Index (met table als view)

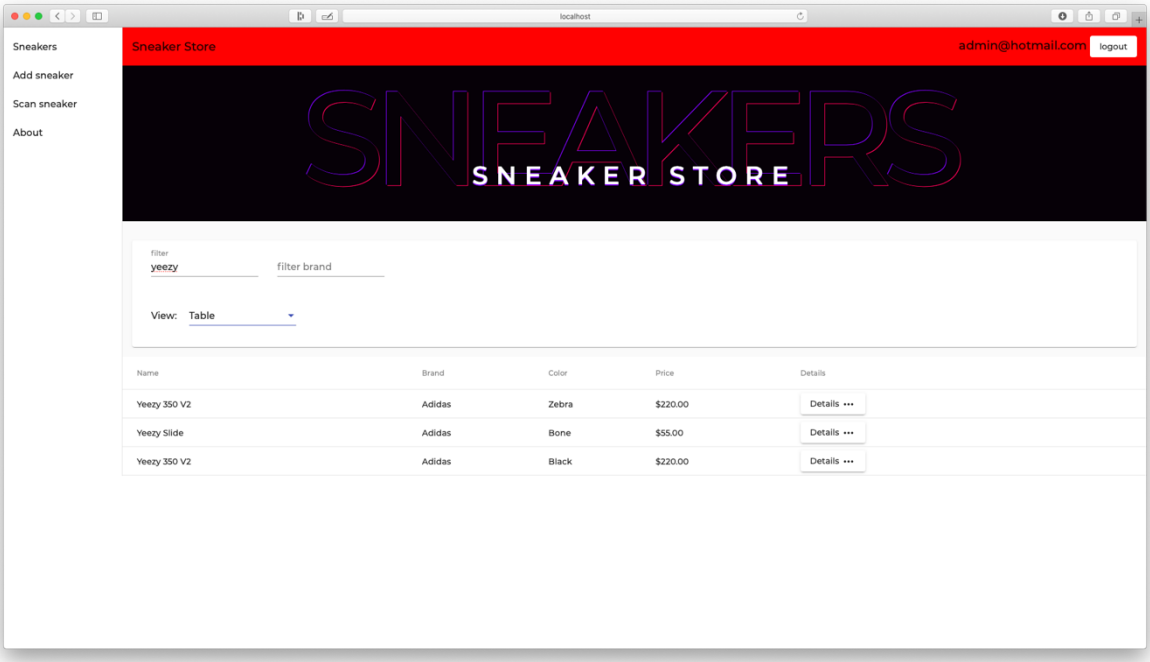


Index met filters toegepast

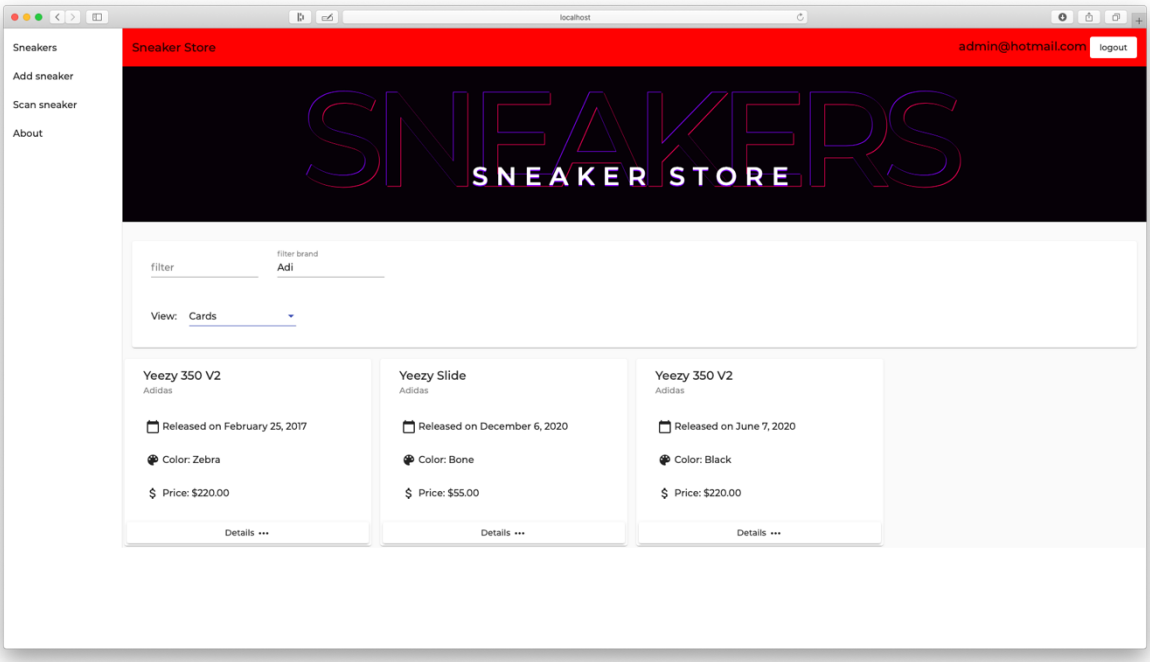
Filter naam in card view



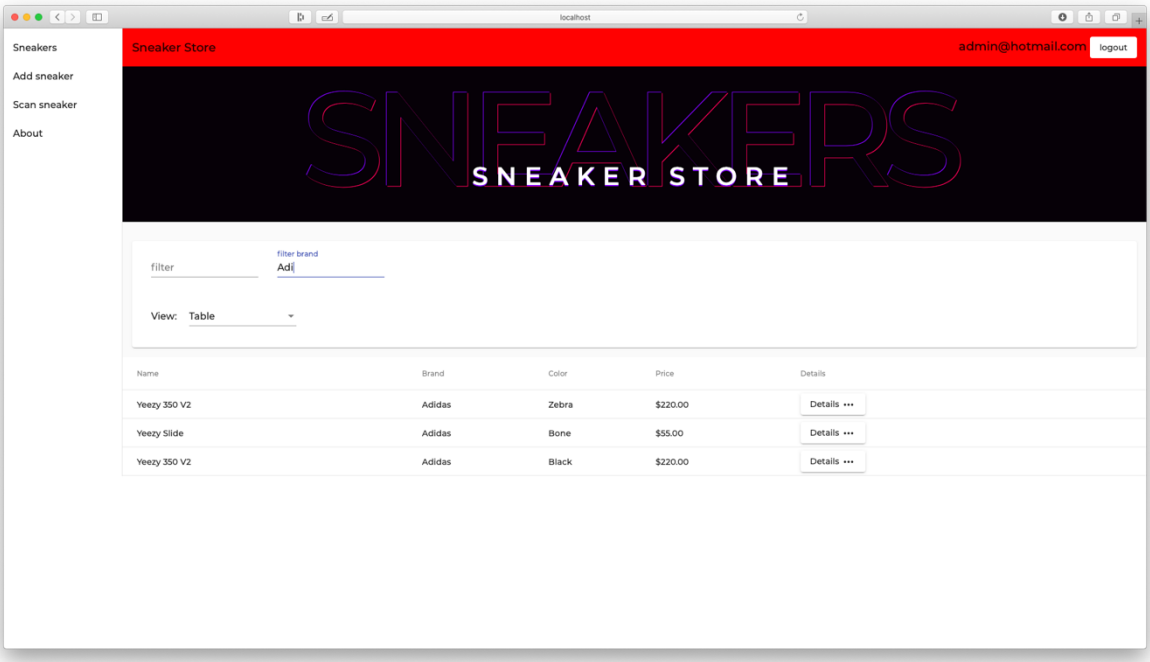
Filter naam in table view



Filter brand in card view

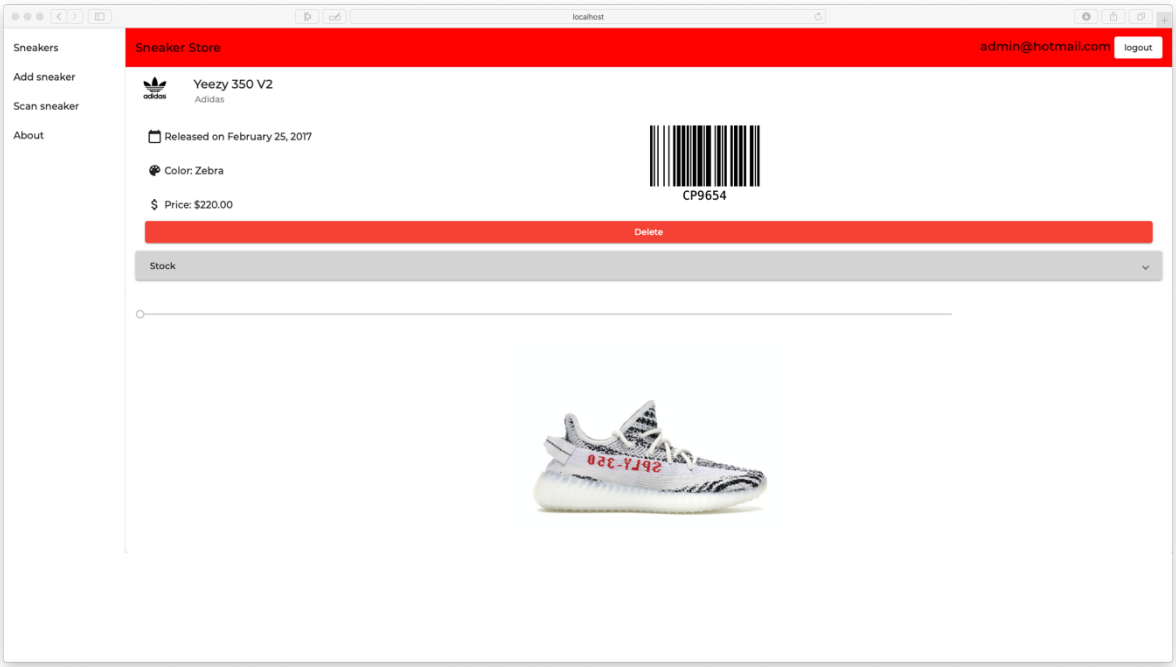


Filter brand in table view

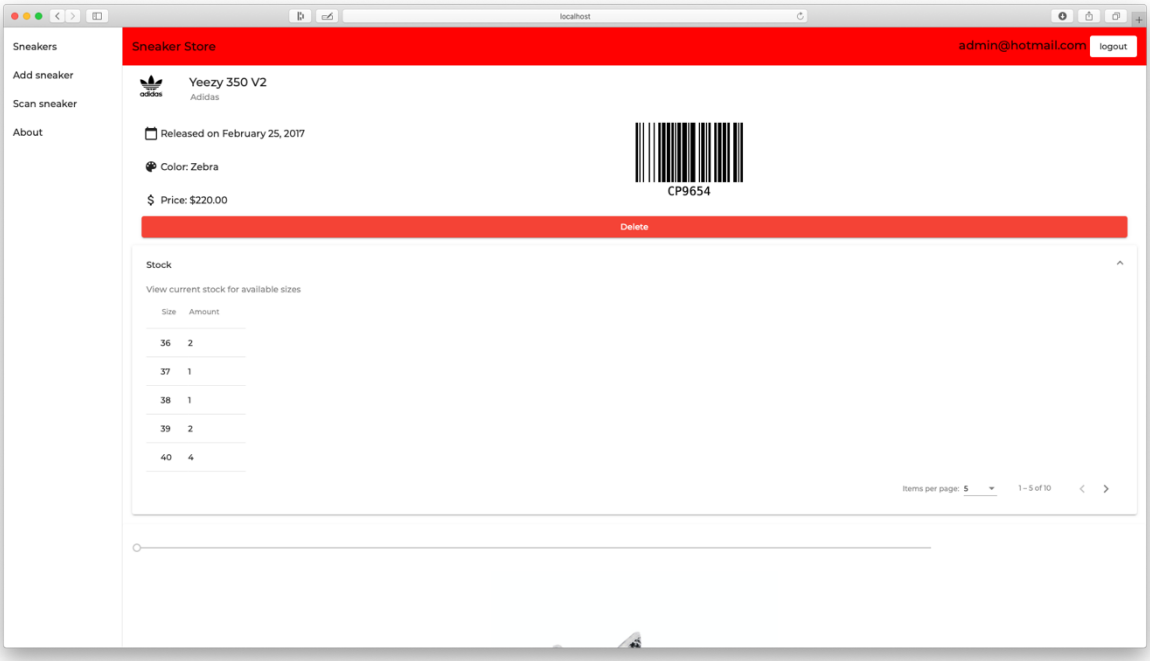


Sneaker detail page

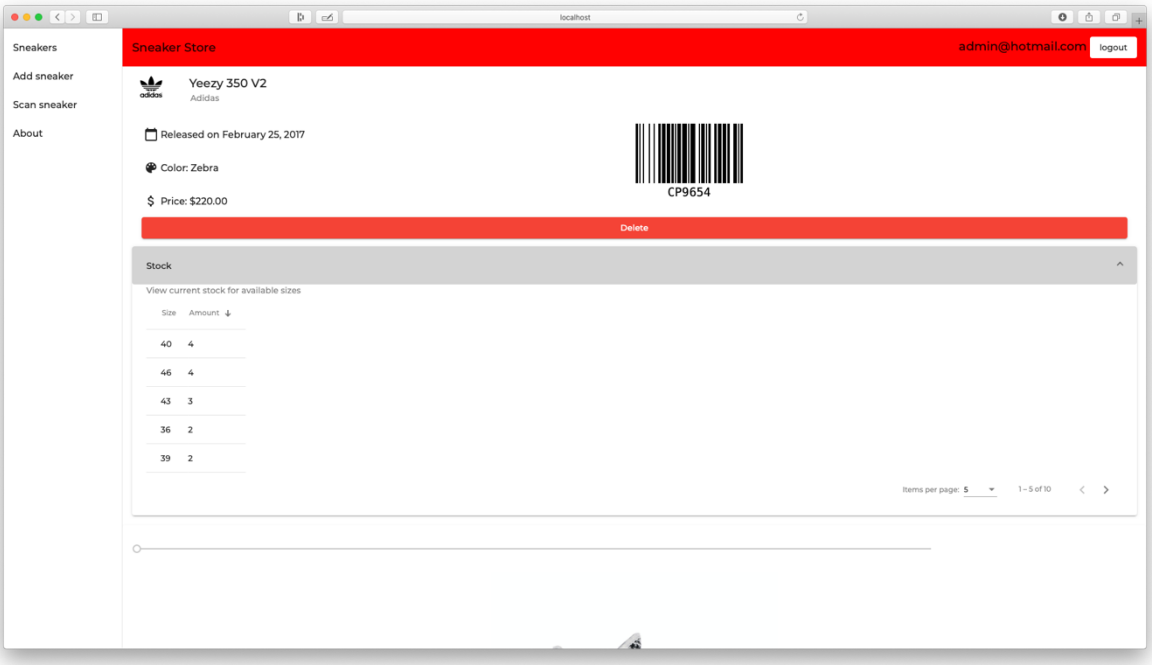
Overzicht van de detail pagina



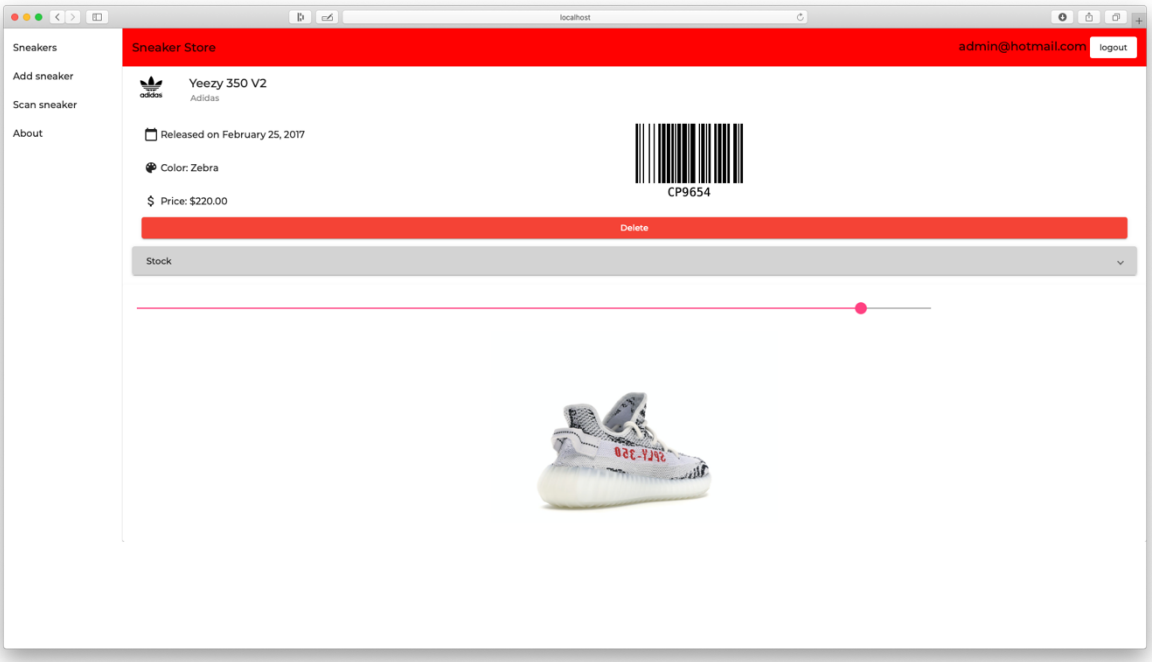
Bekijk de stock van elke sneaker per maat



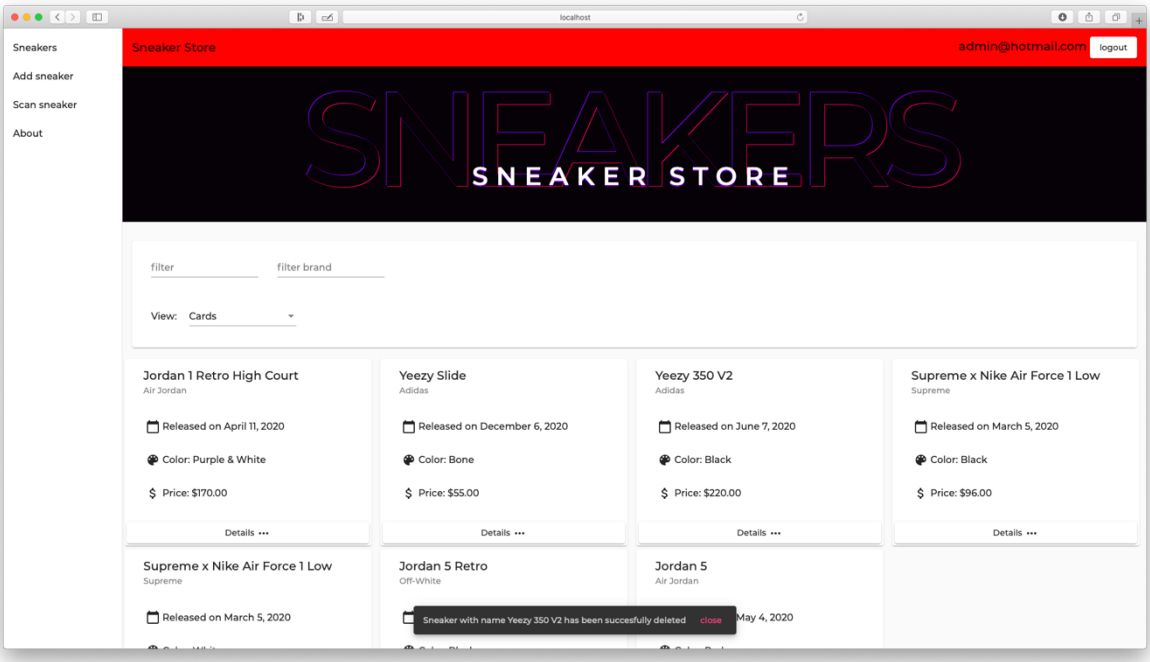
Sorteer het stockoverzicht op maat of aantal



360 graden beeld van de sneakers

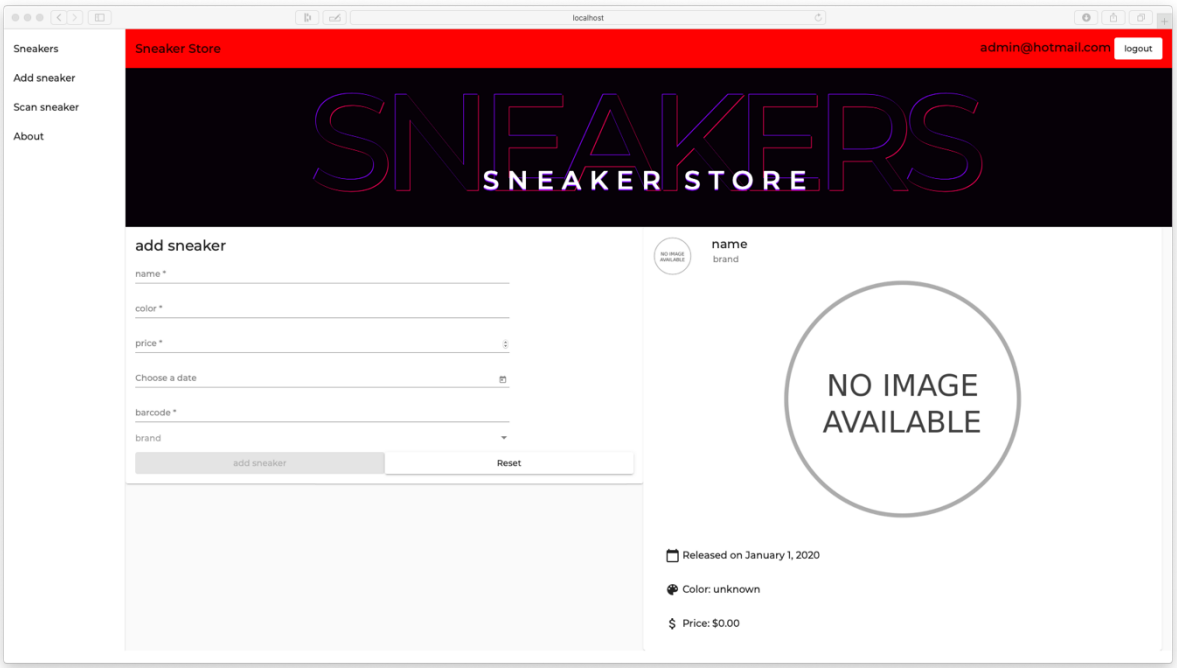


Verwijderen van een sneaker

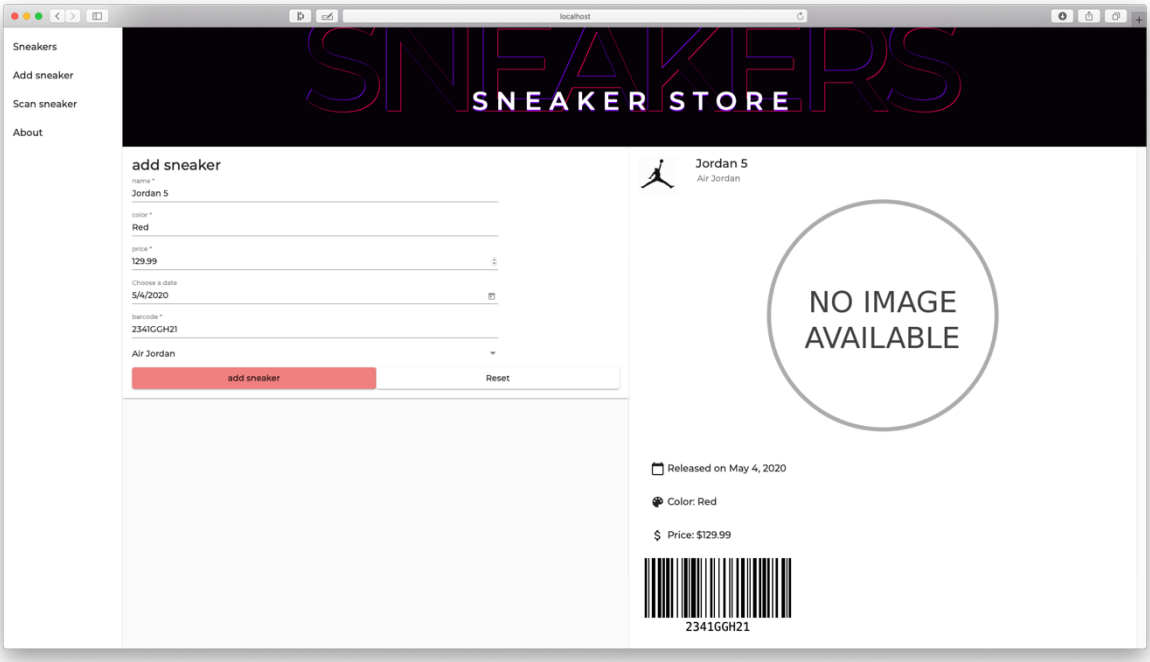


Add Sneaker

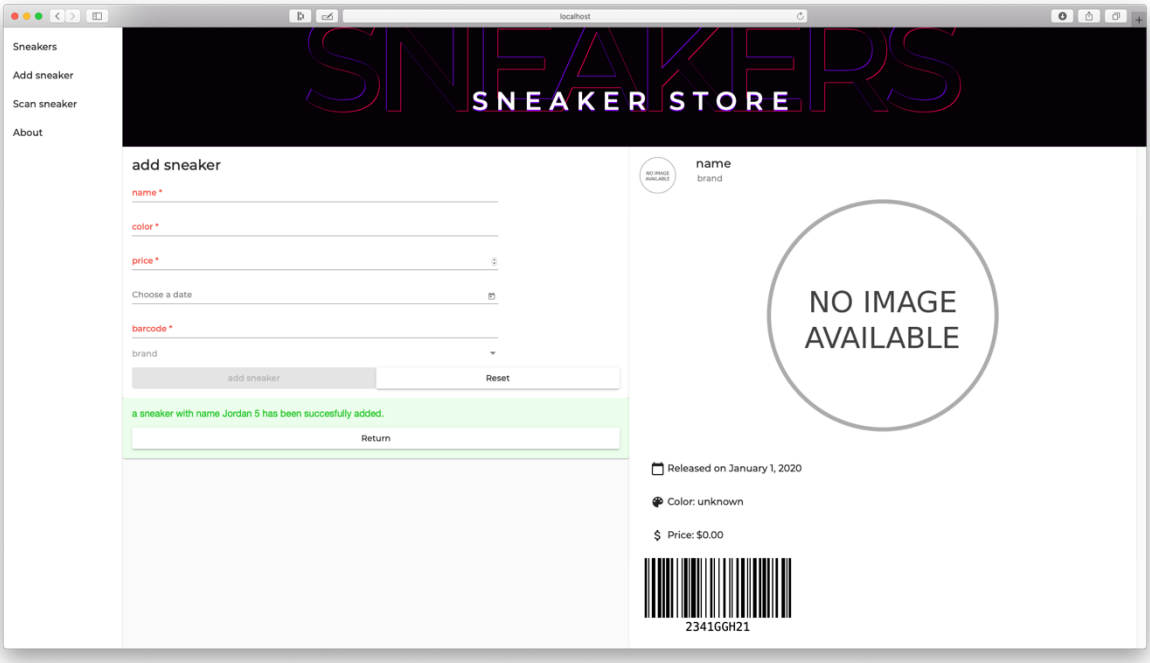
Overzicht van de add sneaker pagina



Preview van de gegevens die zullen opgeslagen worden

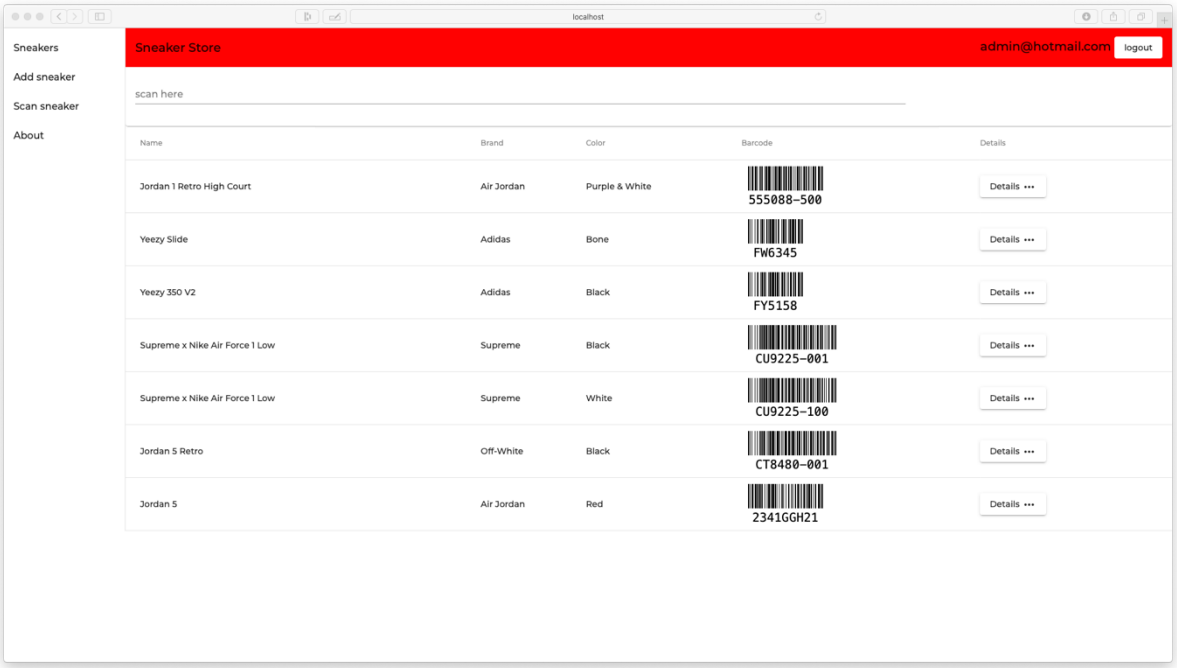


Succesboodschap na het toevoegen van een sneaker

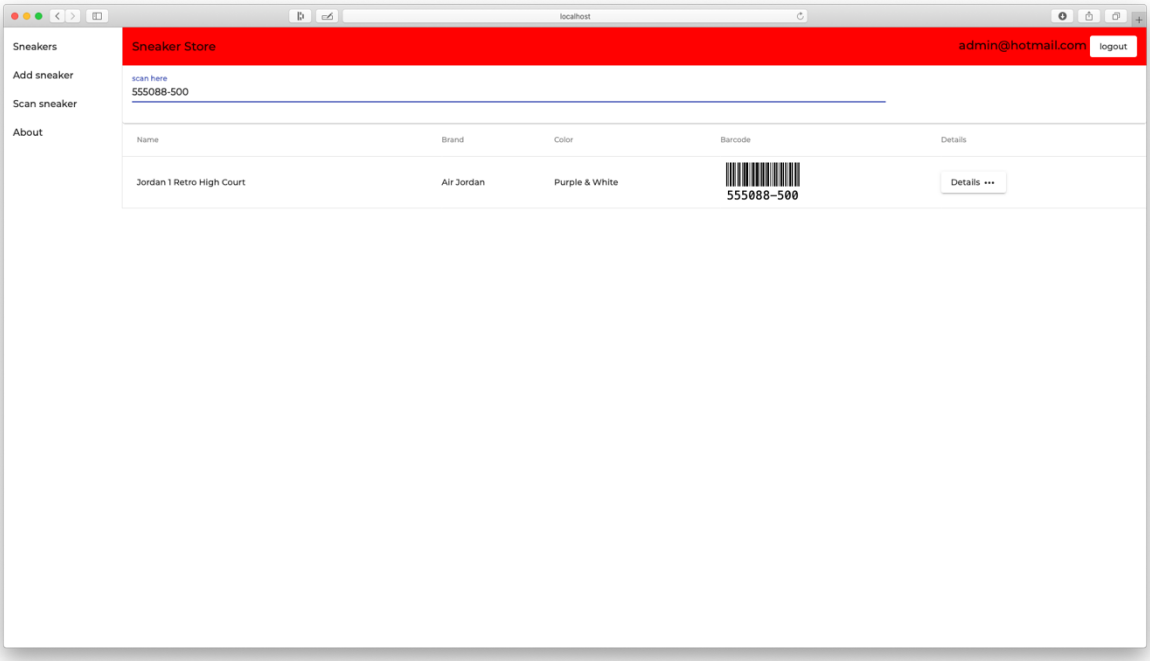


Sneaker scan page

Overzicht

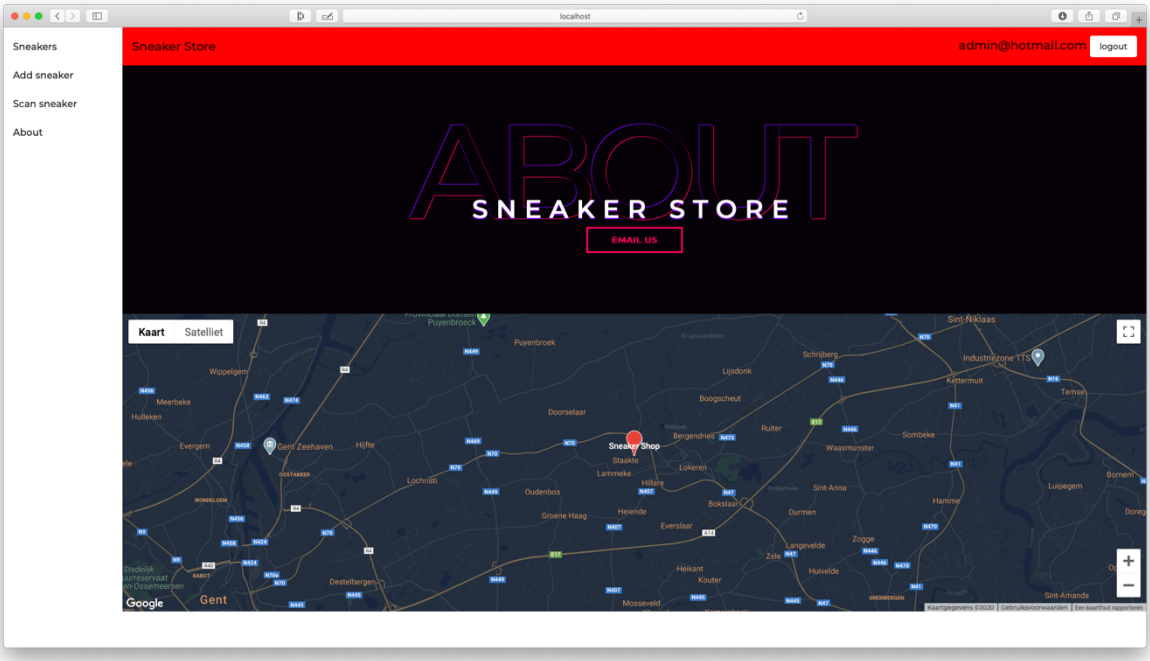


Resultaat na scannen van een sneaker

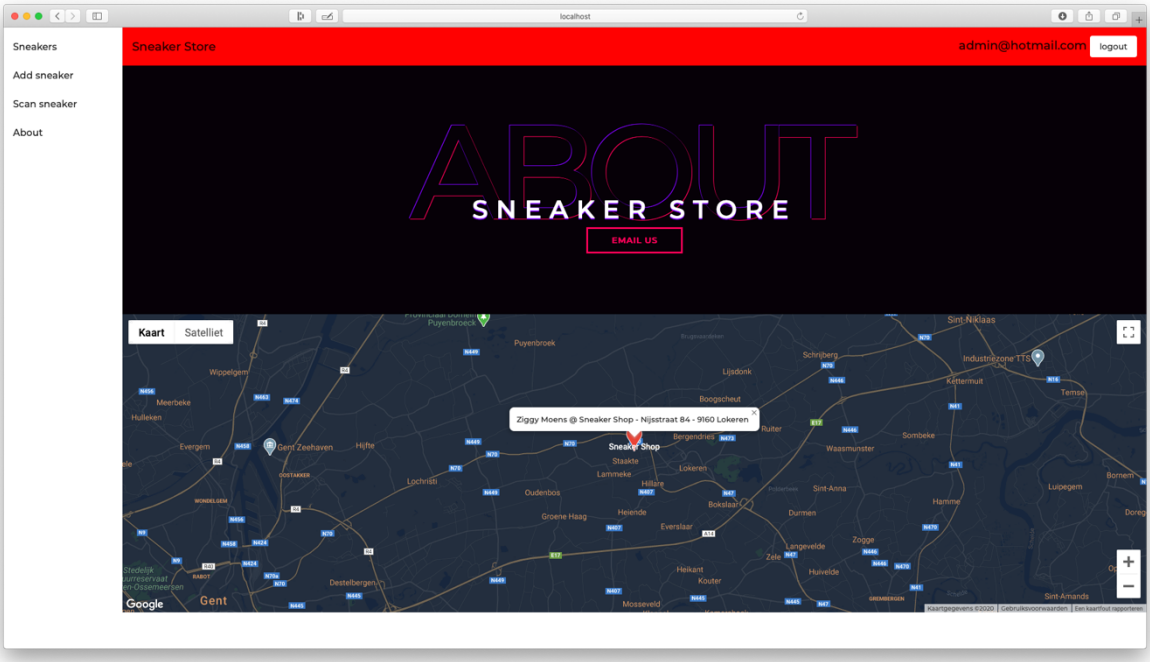


About page

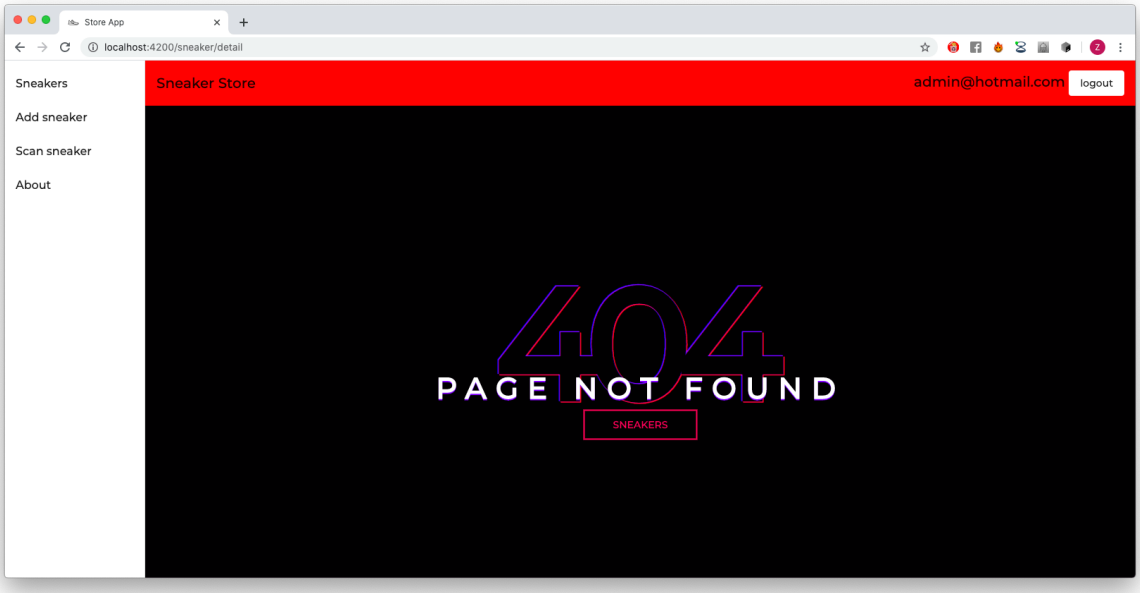
Overzicht



Extra label bij marker op Google Maps

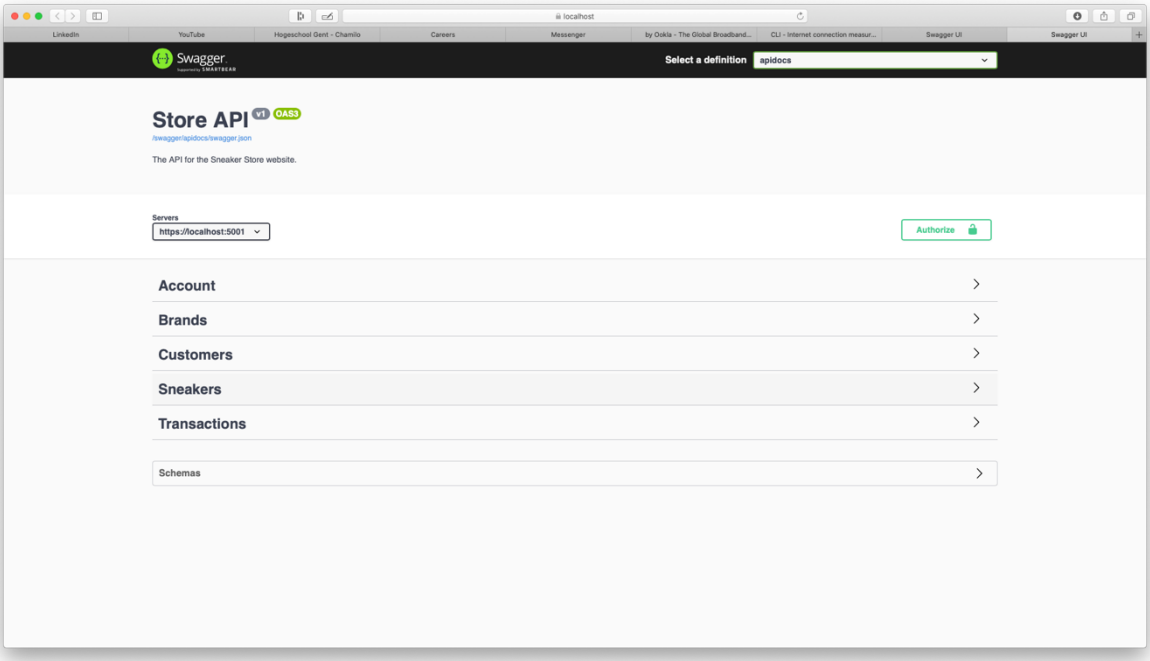


Page-not-found page



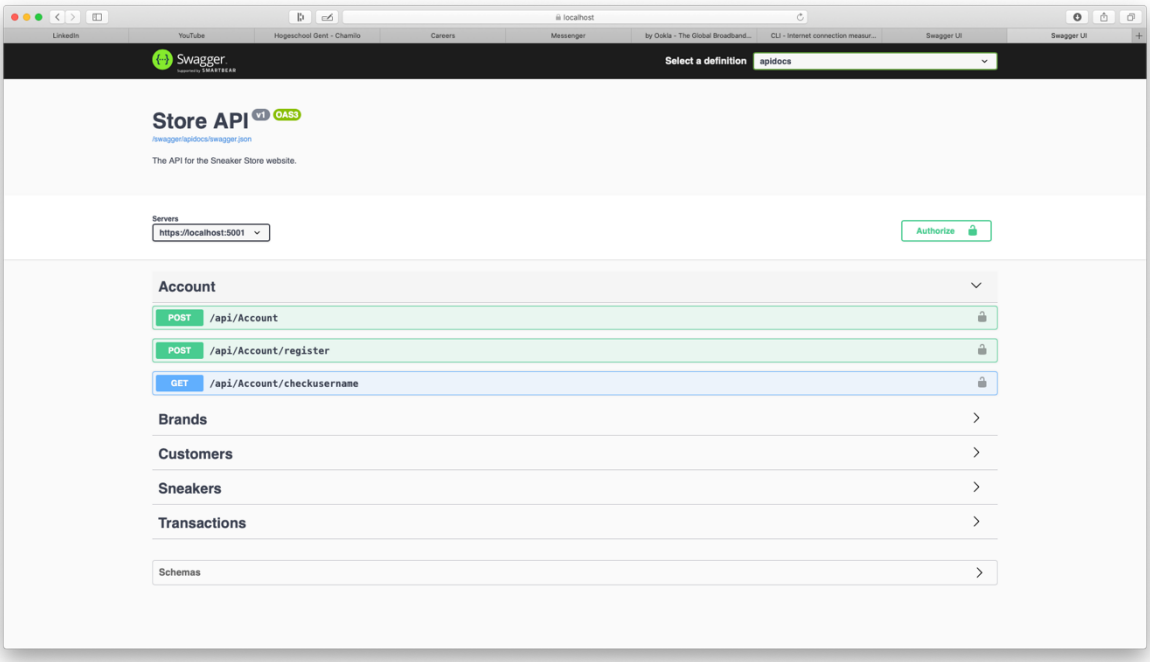
API-calls (Swagger)

Overzicht

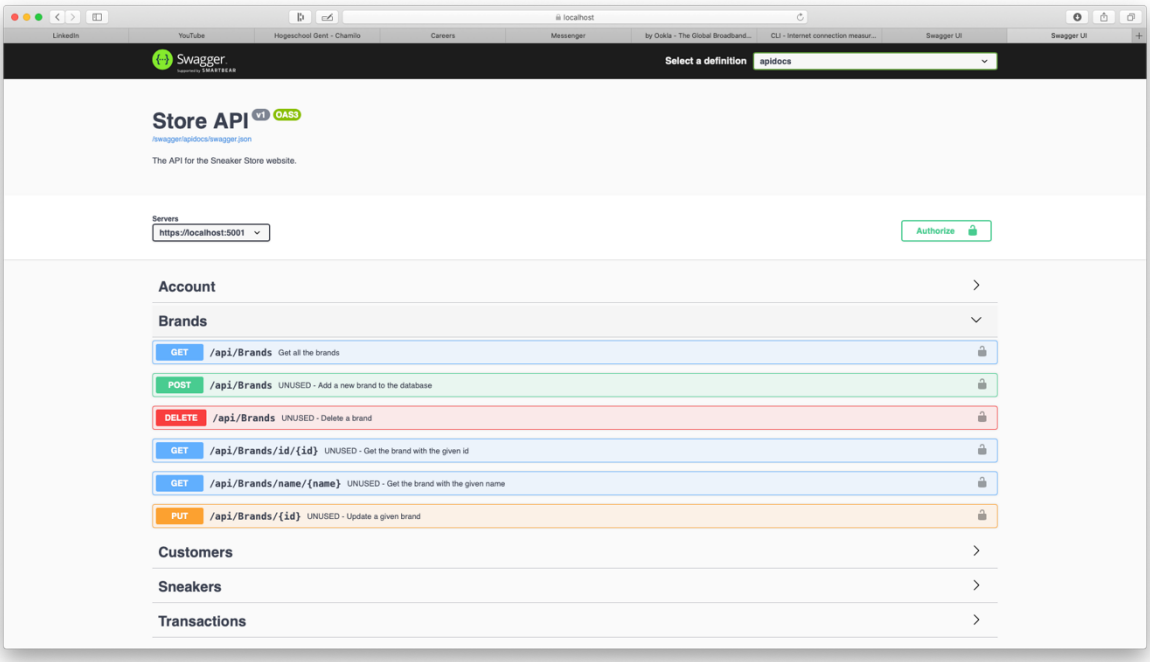


Controllers

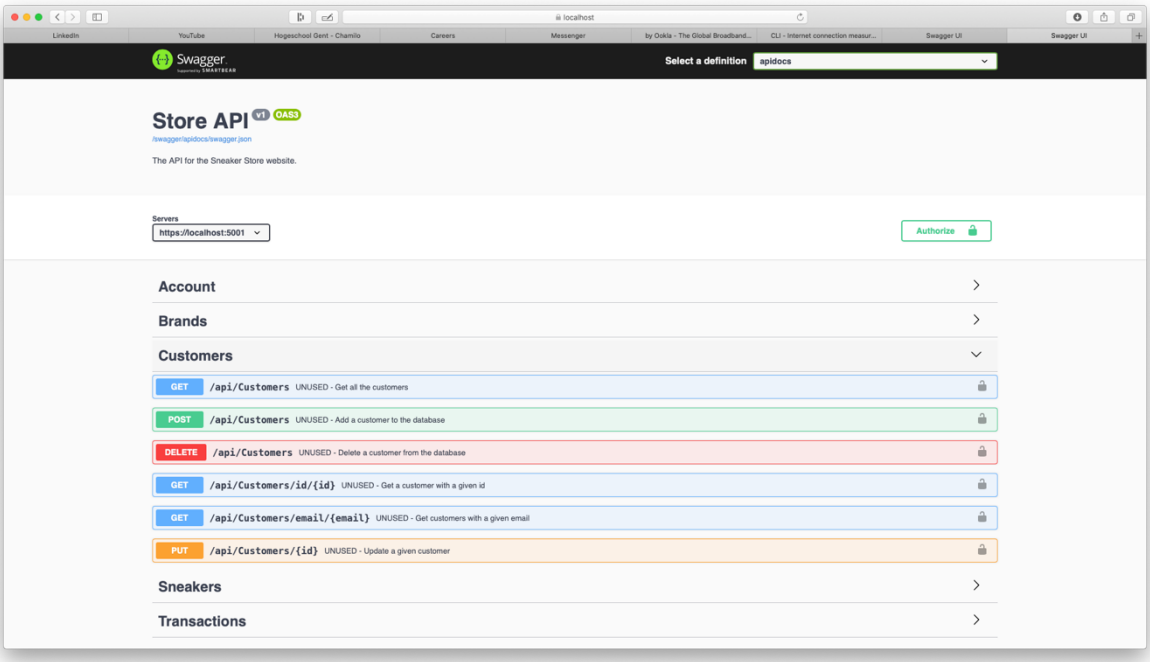
Account



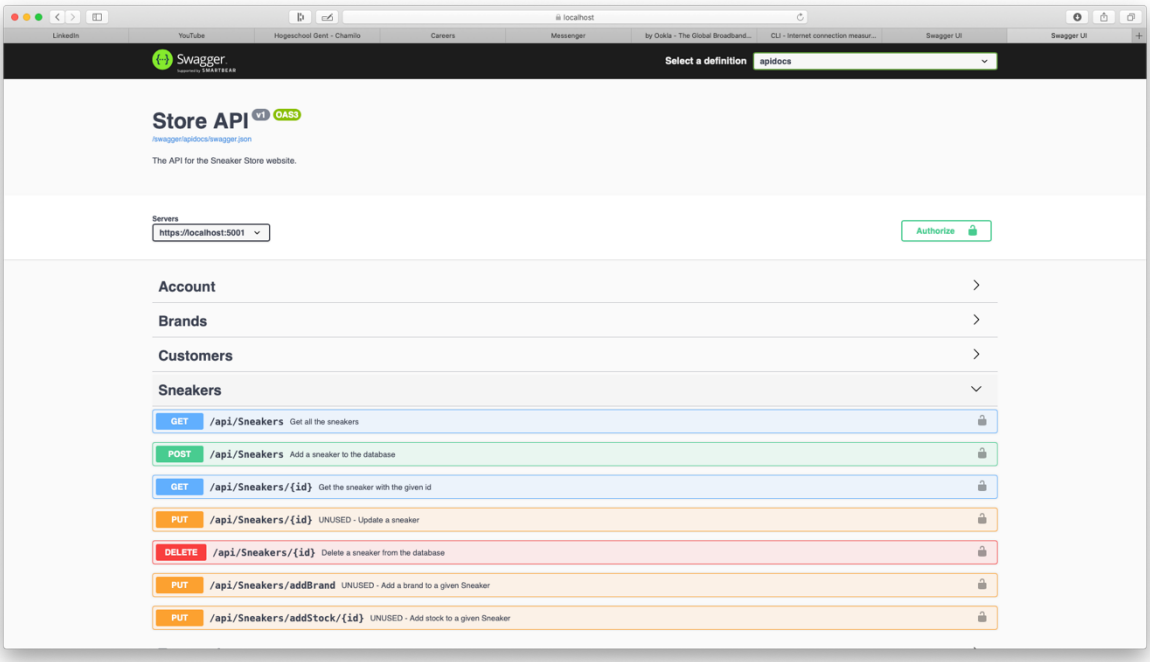
Brands



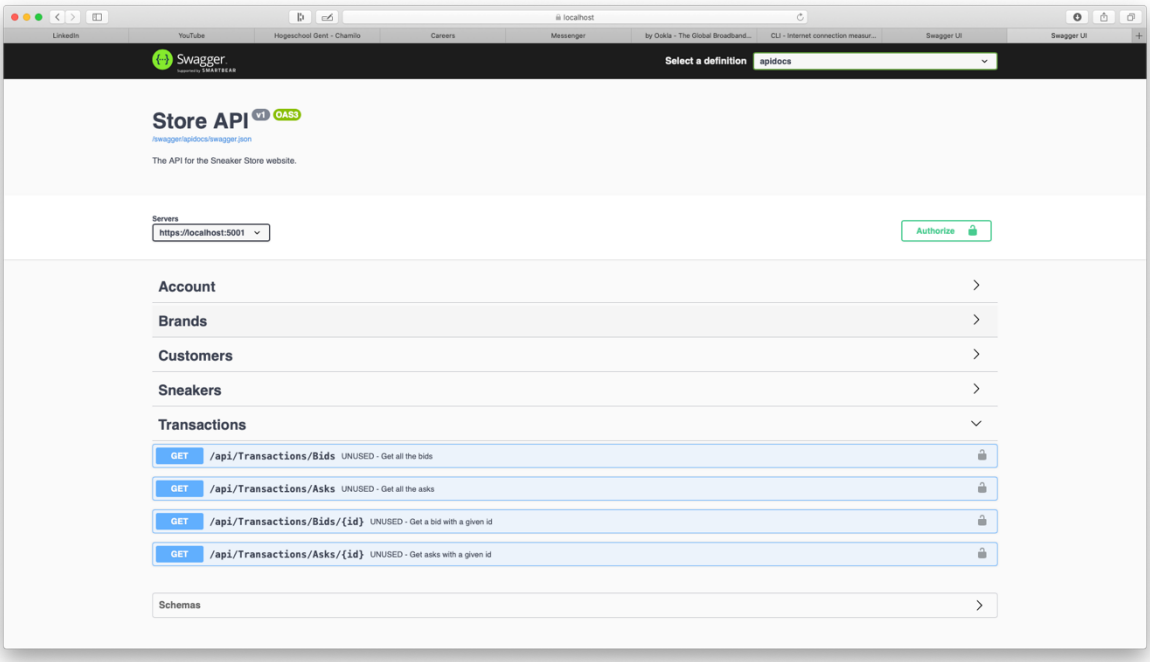
Customers



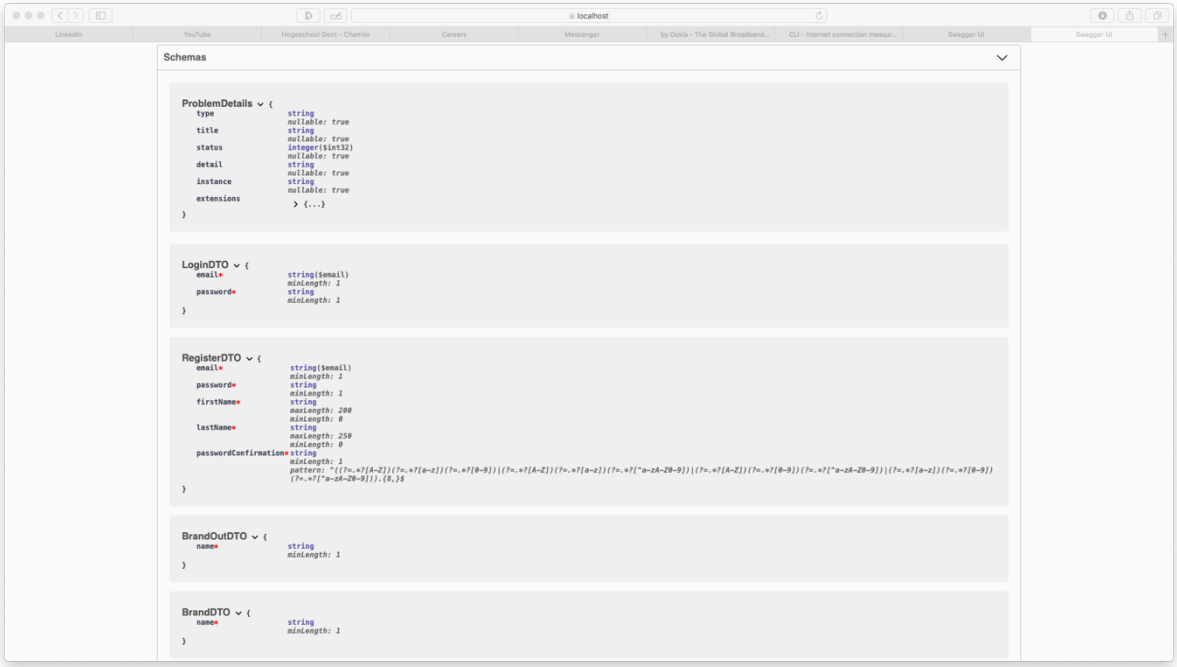
Sneakers

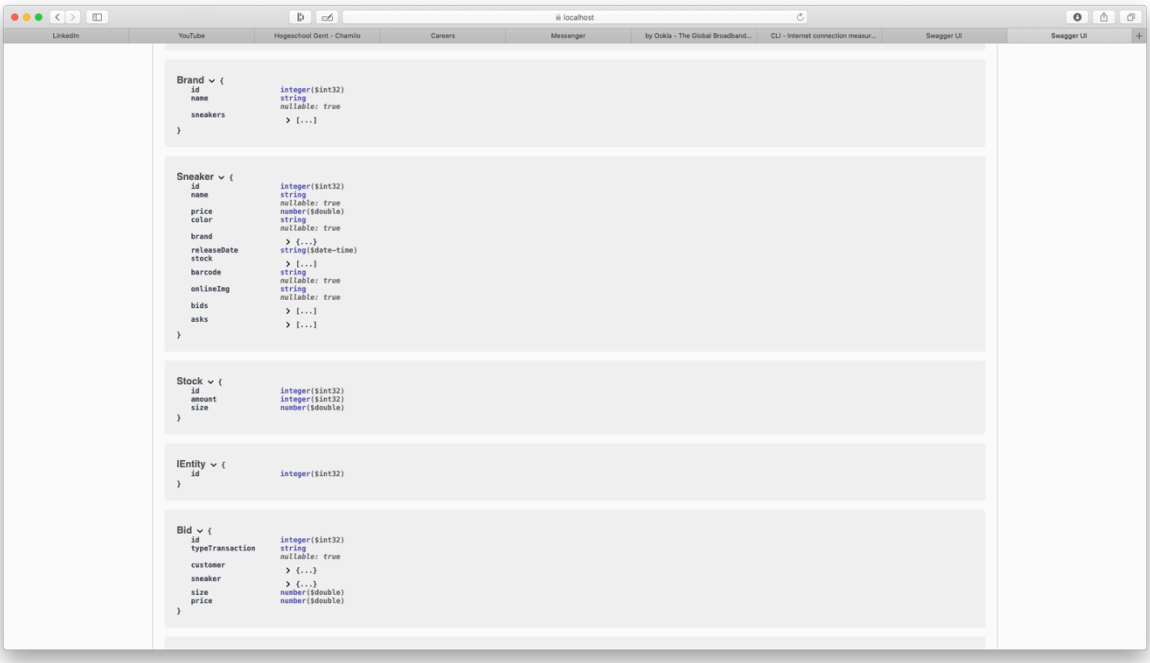


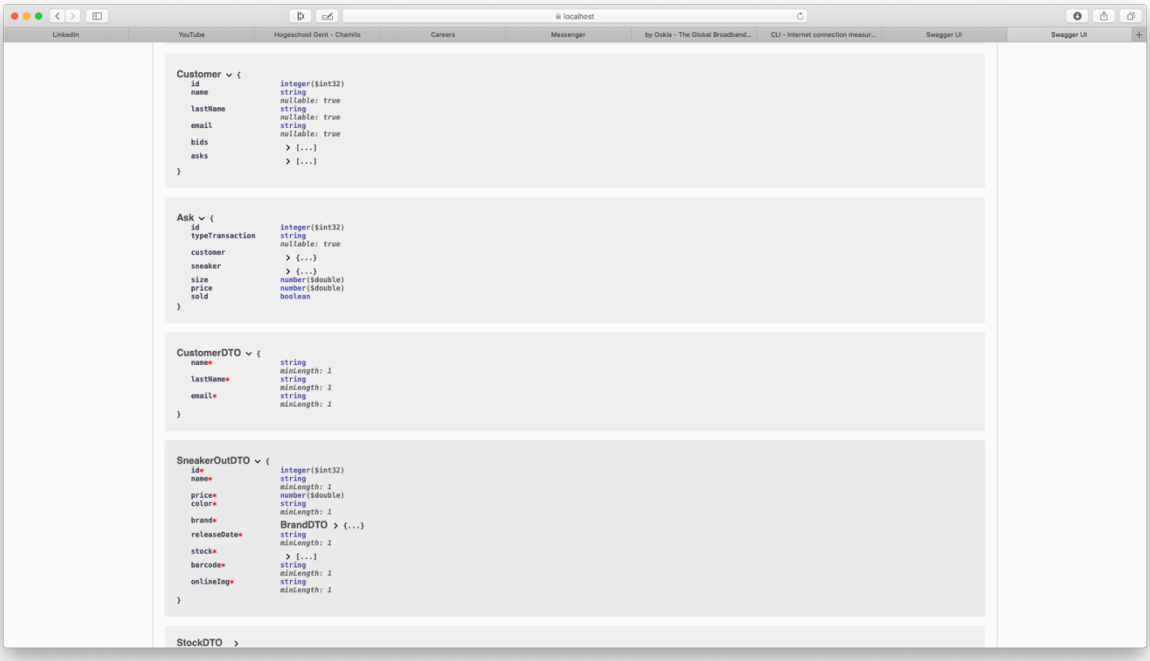
Transactions

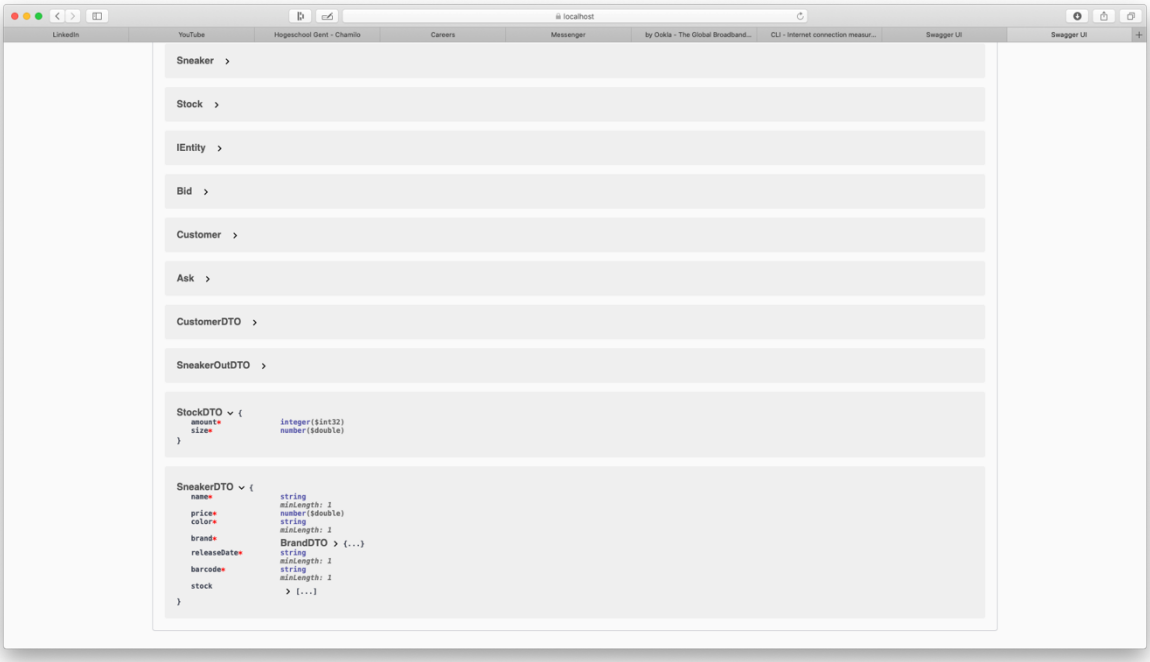


Schema's

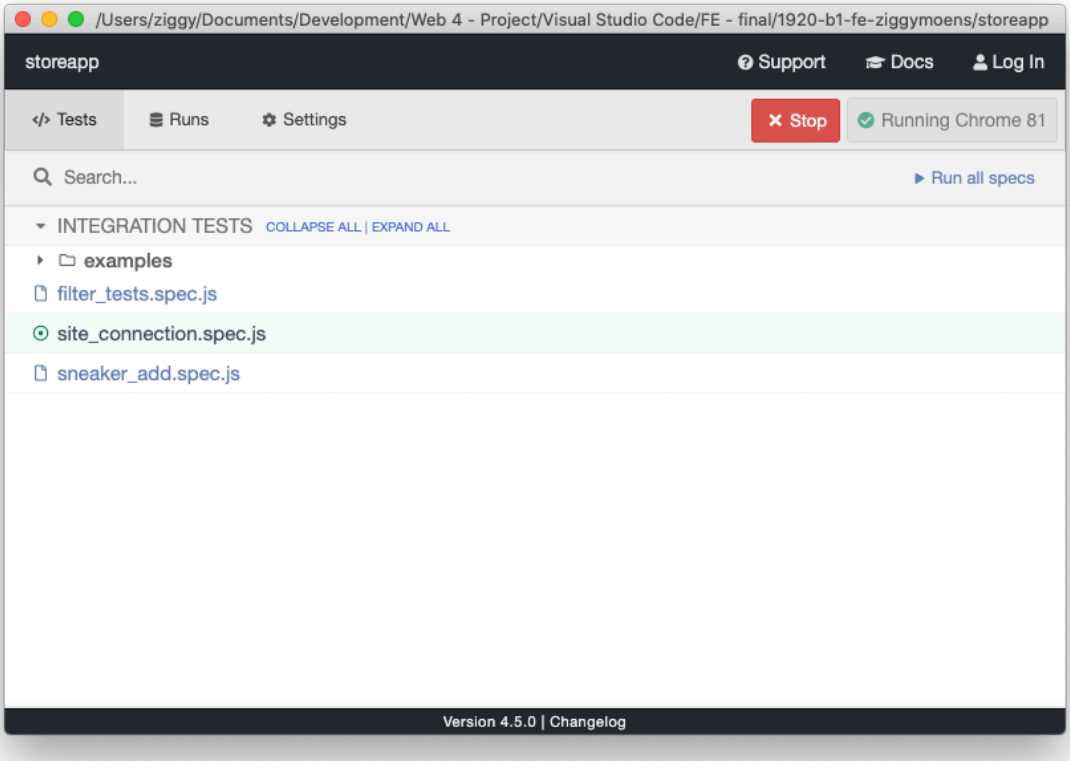

Schemas
ProblemDetails
type string nullable: true
title string nullable: true
status integer($int32) nullable: true
detail string nullable: true
instance string nullable: true
extensions > {...}
LoginDTO
email* string($email) minLength: 1
password* string minLength: 1
RegisterDTO
email* string($email) minLength: 1
password* string minLength: 1
firstName* string maxLength: 200 minLength: 0
lastName* string maxLength: 250 minLength: 0
passwordConfirmation* string minLength: 1
BrandOutDTO
name* string minLength: 1
BrandDTO
name* string minLength: 1


Brand
id integer($int32)
name string nullable: true
sneakers > [...]
Sneaker
id integer($int32)
name string nullable: true
price number($double)
color string nullable: true
brand > {...}
releaseDate string($date-time)
stock > [...]
barcode string nullable: true
onlineImg string nullable: true
bids > [...]
asks > [...]
Stock
id integer($int32)
amount integer($int32)
size number($double)
IEntity
id integer($int32)
Bid
id integer($int32)
typeTransaction string nullable: true
customer > {...}
sneaker > {...}
size number($double)
price number($double)


Customer {
id integer($int32)
name string nullable: true
lastName string nullable: true
email string nullable: true
bids > [...]
asks > [...]
}
Ask {
id integer($int32)
typeTransaction string nullable: true
customer > {...}
sneaker > {...}
size number($double)
price number($double)
sold boolean
}
CustomerDTO {
name* string minLength: 1
lastName* string minLength: 1
email* string minLength: 1
}
SneakerOutDTO {
id* integer($int32)
name* string minLength: 1
price* number($double)
color* string minLength: 1
brand* BrandDTO > {...}
releaseDate* string minLength: 1
stock* > [...]
barcode* string minLength: 1
onlineImg* string minLength: 1
}
StockDTO >


Sneaker >
Stock >
IEntity >
Bid >
Customer >
Ask >
CustomerDTO >
SneakerOutDTO >
StockDTO {
amount* integer($int32)
size* number($double)
}
SneakerDTO {
name* string minLength: 1
price* number($double)
color* string minLength: 1
brand* BrandDTO > {...}
releaseDate* string minLength: 1
barcode* string minLength: 1
stock > [...]
}

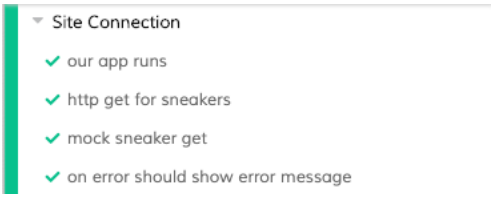
Testen



Site_connection.spec.js

In deze testfile wordt getest of:

- De app werkt en opstart
- De sneakers succesvol kunnen worden opgehaald uit de API
- De mock-data van de sneakers succesvol gebruikt kan worden
- Indien een GET-call mislukt de site de gepaste foutboodschap toont



Filter_tests.spec.js

In deze testfile wordt getest of:

- Op de index pagina:
  - De naamfilter correct werkt
    - Zowel voor Card- als Tabelview
  - De brandfilter correct werkt
    - Zowel voor Card- als Tabelview
- Op de scan pagina:
  - De scanfilter correct werkt

Al deze tests zijn uitgevoerd op mock-data

Sneaker_add.spec.js

In deze testfile wordt getest of:

- De velden correct ingevuld kunnen worden en de correcte preview gegeven wordt
- De submit knop de juiste bevestigingsboodschap geeft
- De reset knop de velden correct leegmaakt