



ZIGGY MOENS – 2B1

SNEAKER STORE

WEBAPPLICATIES IV



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

Inhoudsopgave

Inloggegevens.....	3
Admin	3
Gebruiker.....	3
Beschrijving van de applicatie.....	4
Extra technologieën	5
Google Maps	5
Barcode	5
360 graden foto's	5
Structuur van de app	6
Screenshots van de applicatie	8
Login	8
Register.....	8
Index (<i>met cards als view</i>)	9
Index (<i>met table als view</i>).....	9
Index met filters toegepast.....	10
Filter naam in card view.....	10
Filter naam in table view	10
Filter brand in card view	11
Filter brand in table view	11
Sneaker detail page	12
Overzicht van de detail pagina.....	12
Bekijk de stock van elke sneaker per maat	12
Sorteer het stockoverzicht op maat of aantal	13
360 graden beeld van de sneakers	13
Verwijderen van een sneaker	14
Add Sneaker.....	15
Overzicht van de add sneaker pagina	15
Preview van de gegevens die zullen opgeslagen worden.....	15
Succesbericht na het toevoegen van een sneaker	16
Sneaker scan page	17
Overzicht.....	17
Resultaat na scannen van een sneaker.....	17
About page	18
Overzicht.....	18
Extra label bij marker op Google Maps.....	18
Page-not-found page.....	19
API-calls (Swagger)	20
Overzicht.....	20
Controllers	20
Account	20

Brands.....	21
Customers.....	21
Sneakers.....	22
Transactions.....	22
<i>Schema's</i>	23
Testen.....	25
<i>Site_connection.spec.js</i>	25
<i>Filter_tests.spec.js</i>	25
<i>Sneaker_add.spec.js</i>	26

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

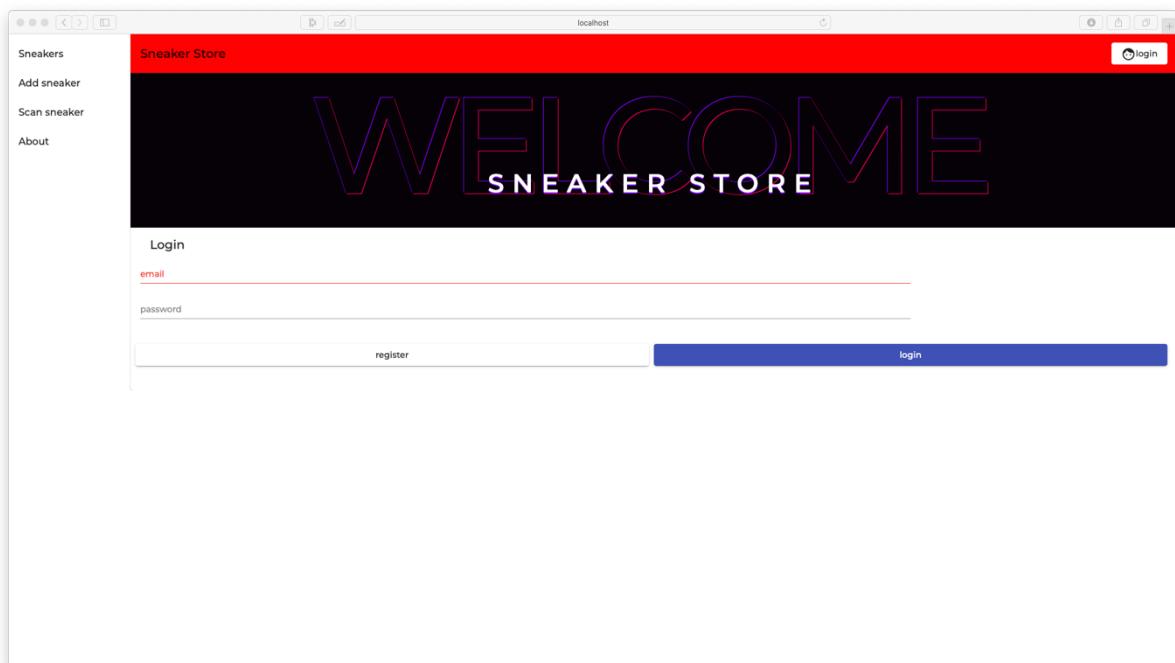
```
1920-b1-fe-ziggymoens/storeapp/src
├── app
│   ├── SelectivePreloadStrategy
│   ├── about-us
│   │   └── about-us.component
│   ├── app-routing.module
│   ├── app.component
│   ├── app.module
│   ├── http-interceptors
│   │   ├── AuthenticationInterceptor
│   │   └── index
│   ├── main-nav
│   │   └── 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
```

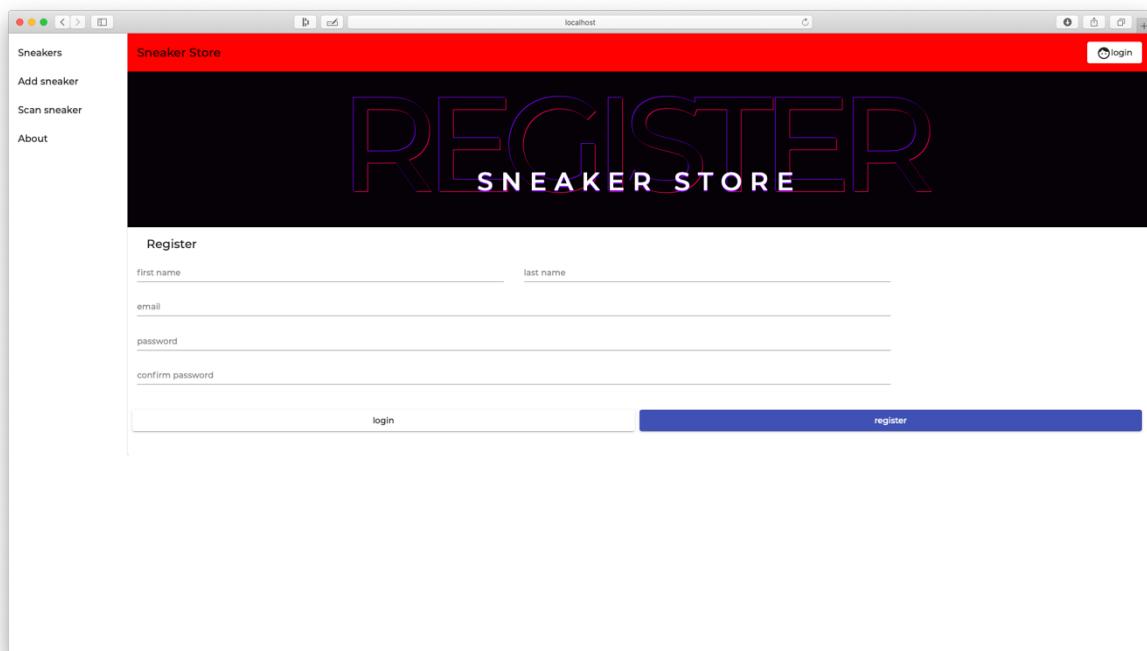
25 directories, 94 files

Screenshots van de applicatie

Login



Register



Index (met cards als view)

Name	Brand	Color	Price	Action
Jordan 1 Retro High Court	Air Jordan	Purple & White	\$170.00	Details ...
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...
Supreme x Nike Air Force 1 Low	Supreme		\$96.00	Details ...
Jordan 5 Retro	Off-White		\$225.00	Details ...

Index (met table als view)

Name	Brand	Color	Price	Action
Jordan 1 Retro High Court	Air Jordan	Purple & White	\$170.00	Details ...
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...
Supreme x Nike Air Force 1 Low	Supreme	Black	\$96.00	Details ...
Supreme x Nike Air Force 1 Low	Supreme	White	\$96.00	Details ...
Jordan 5 Retro	Off-White	Black	\$225.00	Details ...

Index met filters toegepast

Filter naam in card view

The screenshot shows a web browser window for the 'Sneaker Store' application. The header includes a sidebar with links for 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The main area features a large 'SNEAKERS SNEAKER STORE' logo. Below it is a search bar with 'filter' and 'yeezy' entered, and a 'filter brand' input field. A dropdown menu shows 'View: Cards'. The main content displays three cards for Yeezy sneakers:

Name	Brand	Color	Price	Action
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...

Filter naam in table view

The screenshot shows the same web browser window for the 'Sneaker Store' application. The sidebar and logo are identical. The main area shows the same search bar and 'View: Table' dropdown. The main content displays a table of Yeezy sneakers:

Name	Brand	Color	Price	Action
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...

Filter brand in card view

The screenshot shows a web browser window for 'localhost' with the title 'Sneaker Store'. The left sidebar has links for 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The main area features a large 'SNEAKERS SNEAKER STORE' logo. Below it are two search input fields: 'filter' (empty) and 'filter brand' (containing 'Adi'). A dropdown menu 'View: Cards' is selected. Three sneaker cards are displayed:

Name	Brand	Color	Price	Action
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...

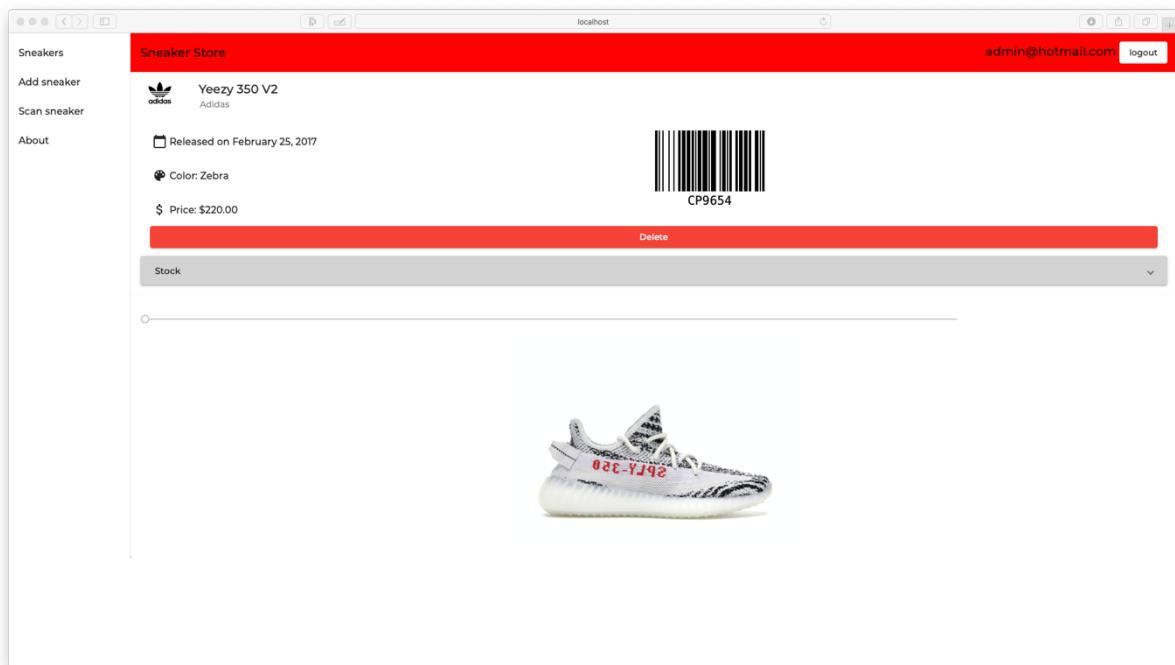
Filter brand in table view

The screenshot shows a web browser window for 'localhost' with the title 'Sneaker Store'. The left sidebar has links for 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The main area features a large 'SNEAKERS SNEAKER STORE' logo. Below it are two search input fields: 'filter' (empty) and 'filter brand' (containing 'Adi'). A dropdown menu 'View: Table' is selected. A table displays the filtered sneaker data:

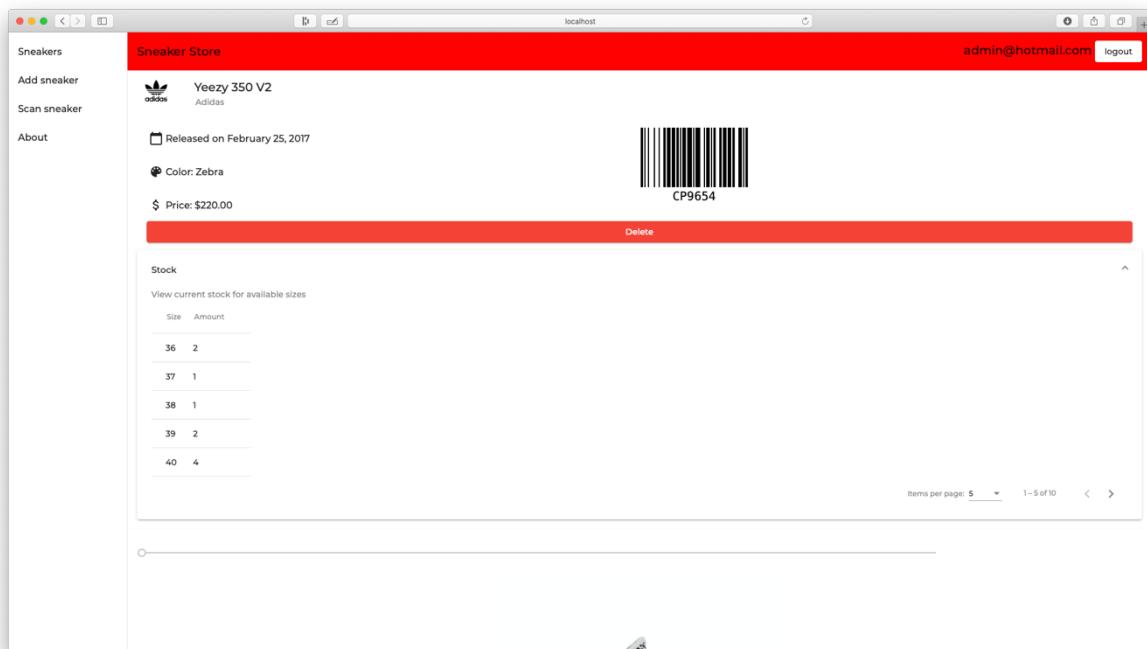
Name	Brand	Color	Price	Action
Yeezy 350 V2	Adidas	Zebra	\$220.00	Details ...
Yeezy Slide	Adidas	Bone	\$55.00	Details ...
Yeezy 350 V2	Adidas	Black	\$220.00	Details ...

Sneaker detail page

Overzicht van de detail pagina



Bekijk de stock van elke sneaker per maat



Sorsteer het stockoverzicht op maat of aantal

The screenshot shows a web-based application for managing sneaker inventory. The main navigation menu on the left includes 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The current page displays a sneaker card for 'Yeezy 350 V2' by 'Adidas'. The card includes the release date (February 25, 2017), color (Zebra), price (\$220.00), and a barcode labeled 'CP9654'. Below the card is a 'Stock' section titled 'View current stock for available sizes', which lists the following inventory:

Size	Amount
40	4
46	4
43	3
36	2
39	2

At the bottom right of the stock section, there are pagination controls: 'Items per page: 5', '1-5 of 10', and navigation arrows.

360 graden beeld van de sneakers

This screenshot shows the same sneaker card for 'Yeezy 350 V2' as the previous one, but with a large image of the sneaker in the center. A red dot on a horizontal line indicates the camera's position, suggesting a 360-degree view feature. The rest of the interface is identical to the first screenshot, including the sidebar menu and the stock table.

Verwijderen van een sneaker

The screenshot shows a web application for managing sneakers. The top navigation bar includes links for 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The user is logged in as 'admin@hotmail.com'. The main header features the text 'SNEAKERS SNEAKER STORE' in large, stylized letters. Below the header, there are two search/filter input fields: 'filter' and 'filter brand'. A dropdown menu 'View: Cards' is set to 'Cards'. The main content area displays a grid of four sneaker cards:

Jordan 1 Retro High Court Air Jordan	Yeezy Slide Adidas	Yeezy 350 V2 Adidas	Supreme x Nike Air Force 1 Low Supreme
📅 Released on April 11, 2020 ⌚ Color: Purple & White 💵 Price: \$170.00 Details ...	📅 Released on December 6, 2020 ⌚ Color: Bone 💵 Price: \$55.00 Details ...	📅 Released on June 7, 2020 ⌚ Color: Black 💵 Price: \$220.00 Details ...	📅 Released on March 5, 2020 ⌚ Color: Black 💵 Price: \$96.00 Details ...
Supreme x Nike Air Force 1 Low Supreme	Jordan 5 Retro Off-White	Jordan 5 Air Jordan	
📅 Released on March 5, 2020 Details ...	🕒 Sneaker with name Yeezy 350 V2 has been successfully deleted close	May 4, 2020	

A modal dialog box is open over the third card, stating 'Sneaker with name Yeezy 350 V2 has been successfully deleted' with a 'close' button.

Add Sneaker

Overzicht van de add sneaker pagina

The screenshot shows a web browser window for a 'Sneaker Store' application. The left sidebar has links for 'Sneakers', 'Add sneaker' (which is active), 'Scan sneaker', and 'About'. The main content area has a large 'SNEAKERS SNEAKER STORE' logo. Below it is a form titled 'add sneaker' with fields for 'name*', 'color*', 'price*', 'Choose a date', 'barcode*', and 'brand'. A 'NO IMAGE AVAILABLE' placeholder image is displayed next to the brand field. At the bottom of the form are 'add sneaker' and 'Reset' buttons. To the right of the form, there is a preview section showing a circular placeholder with 'NO IMAGE AVAILABLE' text, a release date of 'January 1, 2020', color 'unknown', price '\$0.00', and a small Air Jordan logo.

Preview van de gegevens die zullen opgeslagen worden

This screenshot shows the same 'Add sneaker' page after filling out the form. The 'name*' field contains 'Jordan 5', 'color*' is 'Red', 'price*' is '\$129.99', 'Choose a date' is '5/4/2020', 'barcode*' is '2341GGH21', and 'brand' is 'Air Jordan'. The 'add sneaker' button is now highlighted in red. To the right, a preview card for 'Jordan 5' by 'Air Jordan' is shown, featuring the Air Jordan logo, the sneaker name, its color, its price (\$129.99), and a barcode.

Succesbericht na het toevoegen van een sneaker

The screenshot shows a web-based application for managing a sneaker store. On the left, a sidebar menu includes 'Sneakers', 'Add sneaker', 'Scan sneaker', and 'About'. The main content area has a large header 'SNEAKERS SNEAKER STORE'. Below it, a form titled 'add sneaker' contains fields for 'name *' (with 'Jordan 5' entered), 'color *' (empty), 'price *' (empty), 'Choose a date' (empty), 'barcode *' (empty), and 'brand' (empty). Buttons for 'add sneaker' and 'Reset' are present. A green success message at the bottom of the form area states: 'a sneaker with name Jordan 5 has been successfully added.' A 'Return' button is also visible. To the right of the form, there is a circular placeholder with the text 'NO IMAGE AVAILABLE'. Further down, product details are displayed: 'Released on January 1, 2020', 'Color: unknown', 'Price: \$0.00', and a barcode with the identifier '2341GGH21'.

Sneaker scan page

Overzicht

The screenshot shows a web-based application titled "Sneaker Store". On the left, there is a sidebar with links: "Sneakers" (which is active), "Add sneaker", "Scan sneaker", and "About". The main content area has a red header bar with the title "Sneaker Store" and the user information "admin@hotmail.com" and "logout". Below this, there is a section labeled "scan here" with a text input field containing the barcode "555088-500". The main table lists seven sneakers:

Name	Brand	Color	Barcode	Details
Jordan 1 Retro High Court	Air Jordan	Purple & White	555088-500	Details ...
Yeezy Slide	Adidas	Bone	FW6345	Details ...
Yeezy 350 V2	Adidas	Black	FY5158	Details ...
Supreme x Nike Air Force 1 Low	Supreme	Black	CU9225-001	Details ...
Supreme x Nike Air Force 1 Low	Supreme	White	CU9225-100	Details ...
Jordan 5 Retro	Off-White	Black	CT8480-001	Details ...
Jordan 5	Air Jordan	Red	2341GGH21	Details ...

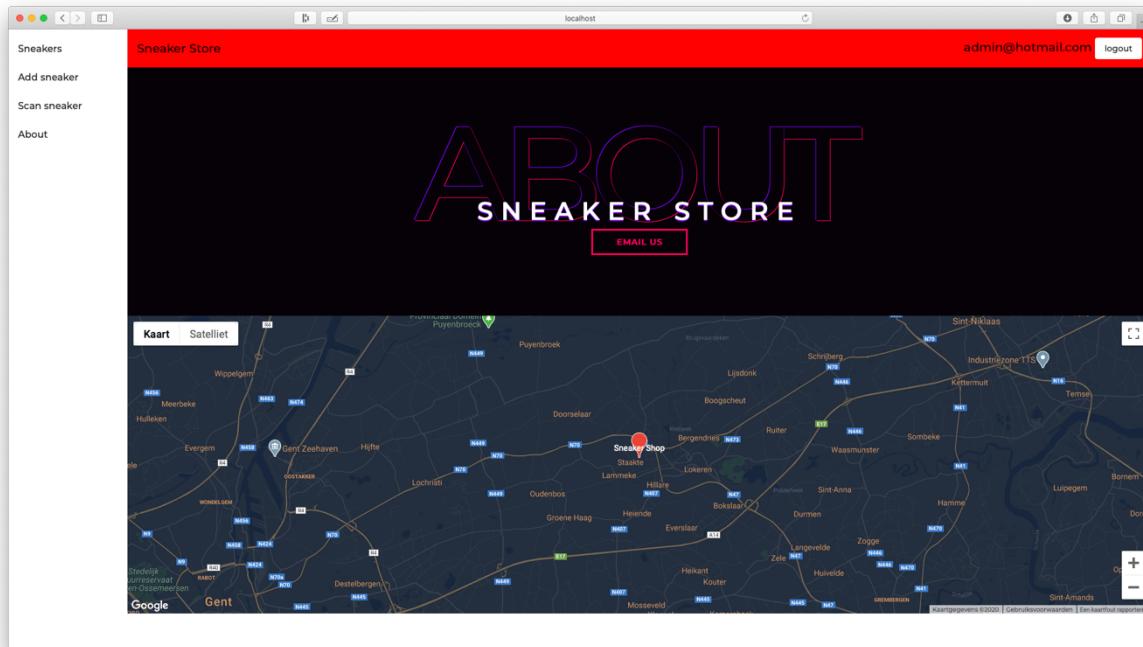
Resultaat na scannen van een sneaker

The screenshot shows the same "Sneaker Store" application after scanning the barcode "555088-500". The main content area now displays the details for the "Jordan 1 Retro High Court" sneaker, which has the barcode "555088-500" displayed above the table.

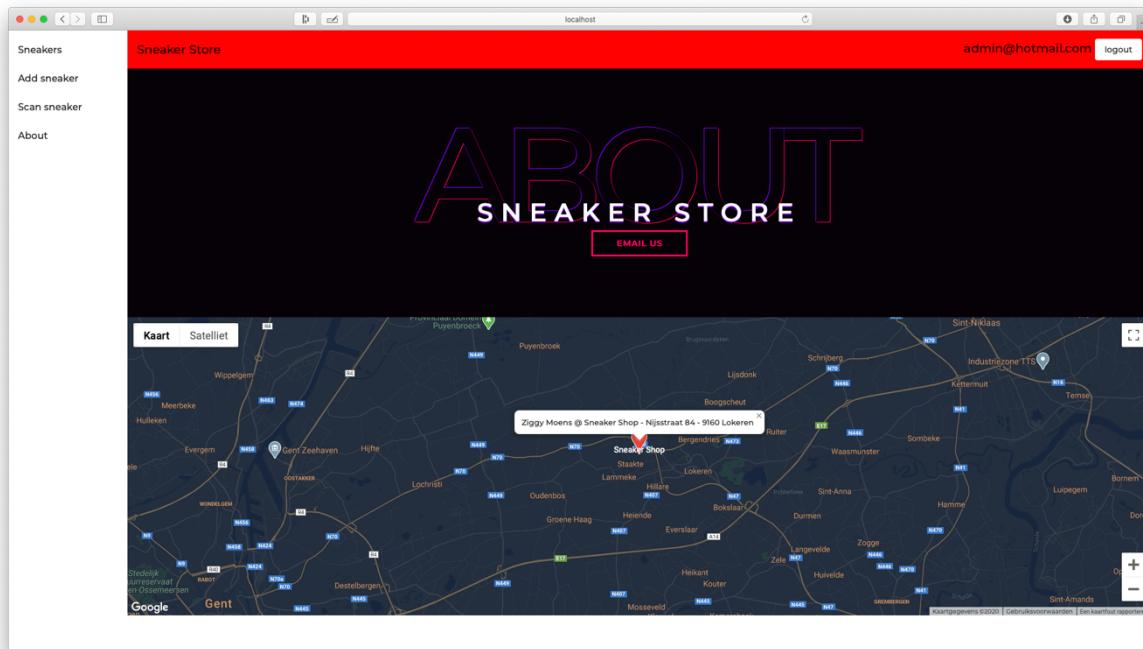
Name	Brand	Color	Barcode	Details
Jordan 1 Retro High Court	Air Jordan	Purple & White	555088-500	Details ...

About page

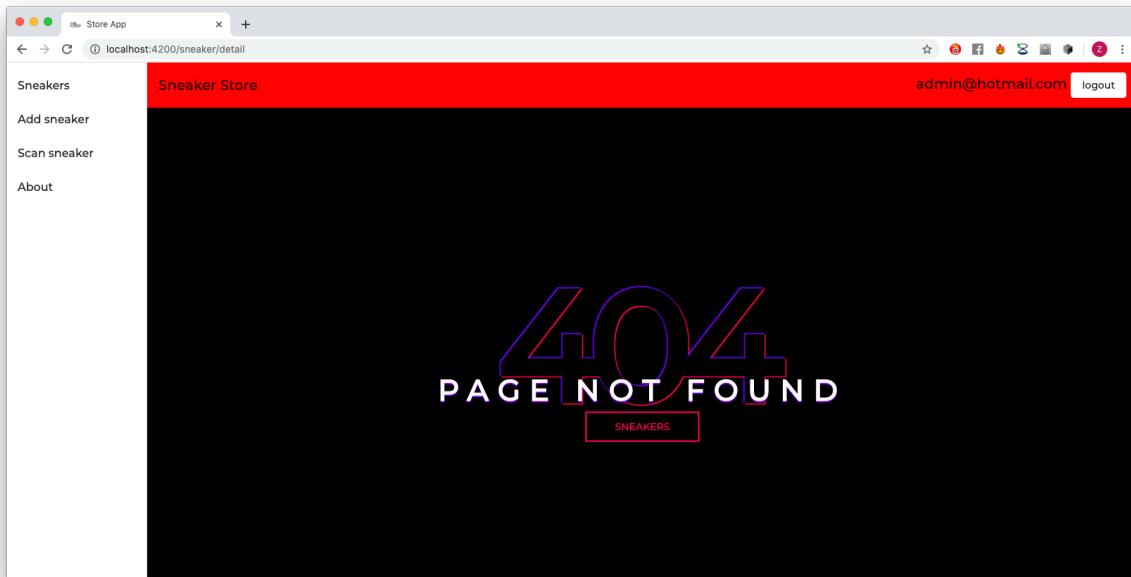
Overzicht



Extra label bij marker op Google Maps



Page-not-found page



API-calls (Swagger)

Overzicht

The screenshot shows the Swagger UI interface for the 'Store API'. At the top, there is a navigation bar with links to LinkedIn, YouTube, Hogeschool Gent - Chamilo, Careers, Messenger, by Dokla - The Global Broadband..., CLI - Internet connection measure..., Swagger UI, and a dropdown menu for 'Select a definition' set to 'apidocs'. Below the navigation bar, the title 'Store API' is displayed with a 'v1' version indicator and an 'OAS3' badge. A sub-link 'Swagger/apidocs/swagger.json' is shown. A descriptive text states: 'The API for the Sneaker Store website.' On the left side, there is a sidebar with a 'Servers' dropdown set to 'https://localhost:5001' and an 'Authorize' button with a lock icon. The main content area lists several API endpoints under categories: 'Account', 'Brands', 'Customers', 'Sneakers', and 'Transactions'. Each category has a right-pointing arrow. Below these categories is a 'Schemas' section with a right-pointing arrow.

Controllers

Account

This screenshot shows the same Swagger UI interface as the previous one, but it focuses on the 'Account' controller. The 'Account' category in the sidebar is expanded, revealing three API endpoints: 'POST /api/Account' (green button), 'POST /api/Account/register' (green button), and 'GET /api/Account/checkusername' (blue button). The other categories ('Brands', 'Customers', 'Sneakers', 'Transactions') and the 'Schemas' section are collapsed. The 'Servers' dropdown and 'Authorize' button are also visible at the top.

Brands

The screenshot shows the Swagger UI interface for the 'Store API' (version 1.0.0, OAS3). The top navigation bar includes links for LinkedIn, YouTube, Hogeschool Gent - Chamilia, Careers, Messenger, and a 'Select a definition' dropdown set to 'apidocs'. The main content area displays the 'Brands' section under the 'Account' category. It lists several API endpoints:

- Brands**
 - GET /api/Brands** Get all the brands
 - POST /api/Brands** UNUSED - Add a new brand to the database
 - DELETE /api/Brands/{id}** UNUSED - Delete a brand
 - GET /api/Brands/{id}** UNUSED - Get the brand with the given id
 - GET /api/Brands/{name}** UNUSED - Get the brand with the given name
 - PUT /api/Brands/{id}** UNUSED - Update a given brand

Other sections visible include Customers, Sneakers, and Transactions.

Customers

The screenshot shows the Swagger UI interface for the 'Store API' (version 1.0.0, OAS3). The top navigation bar includes links for LinkedIn, YouTube, Hogeschool Gent - Chamilia, Careers, Messenger, and a 'Select a definition' dropdown set to 'apidocs'. The main content area displays the 'Customers' section under the 'Account' category. It lists several API endpoints:

- Customers**
 - GET /api/Customers** UNUSED - Get all the customers
 - POST /api/Customers** UNUSED - Add a customer to the database
 - DELETE /api/Customers/{id}** UNUSED - Delete a customer from the database
 - GET /api/Customers/{id}** UNUSED - Get a customer with a given id
 - GET /api/Customers/{email}** UNUSED - Get customers with a given email
 - PUT /api/Customers/{id}** UNUSED - Update a given customer

Other sections visible include Brands, Sneakers, and Transactions.

Sneakers

The screenshot shows the Swagger UI interface for the 'Store API' (version 1.0.0, OAS3). The top navigation bar includes links for LinkedIn, YouTube, Hogeschool Gent - Chamilia, Careers, Messenger, by Ookla - The Global Broadband..., CLI - Internet connection measure..., and Swagger UI. A dropdown menu 'Select a definition' is set to 'apidocs'. The main title is 'Store API' with a 'v1' badge and an 'OAS3' badge. Below it is the URL '/swagger/api/docs/swagger.json'. A sub-header states: 'The API for the Sneaker Store website.' A 'Servers' dropdown is set to 'https://localhost:5001'. An 'Authorize' button with a lock icon is visible. The API documentation is organized into sections: Account, Brands, Customers, and Sneakers. The 'Sneakers' section is expanded, showing the following endpoints:

- GET /api/Sneakers** Get all the sneakers
- POST /api/Sneakers** Add a sneaker to the database
- GET /api/Sneakers/{id}** Get the sneaker with the given id
- PUT /api/Sneakers/{id}** UNUSED - Update a sneaker
- DELETE /api/Sneakers/{id}** Delete a sneaker from the database
- PUT /api/Sneakers/addBrand** UNUSED - Add a brand to a given Sneaker
- PUT /api/Sneakers/addStock/{id}** UNUSED - Add stock to a given Sneaker

Transactions

The screenshot shows the Swagger UI interface for the 'Store API' (version 1.0.0, OAS3). The top navigation bar and 'Select a definition' dropdown are identical to the previous screenshot. The main title is 'Store API' with a 'v1' badge and an 'OAS3' badge. Below it is the URL '/swagger/api/docs/swagger.json'. A sub-header states: 'The API for the Sneaker Store website.' A 'Servers' dropdown is set to 'https://localhost:5001'. An 'Authorize' button with a lock icon is visible. The API documentation is organized into sections: Account, Brands, Customers, Sneakers, and Transactions. The 'Transactions' section is expanded, showing the following endpoints:

- GET /api/Transactions/Bids** UNUSED - Get all the bids
- GET /api/Transactions/Asks** UNUSED - Get all the asks
- GET /api/Transactions/Bids/{id}** UNUSED - Get a bid with a given id
- GET /api/Transactions/Asks/{id}** UNUSED - Get asks with a given id

A 'Schemas' section is also present.

Schema's

```

Schemas

ProblemDetails ✎ {
    type string
    message string
    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
    pattern: "((?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])(?s.*?[^A-Z])(?s.*?[^a-z])(?s.*?[^0-9])){6,}§"
}

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 > (...) string(date-time)
    releasedate string(date-time)
    stock > (...) string
    barcode string
    nullable: true
    onlineling string
    nullable: true
    bids > [...] string
    asks > [...] string
}

Stock ✎ {
    id integer($int32)
    amount integer($int32)
    size number($double)
}

IEntity ✎ {
    id integer($int32)
}

Bid ✎ {
    id integer($int32)
    typeTransaction string
    nullable: true
    customer > (...) string
    sneaker > (...) string
    size number($double)
    price number($double)
}

```

```

Customer ~ {
    id          integer($int32)
    name        string
    mutable: true
    lastName    string
    mutable: true
    email       string
    mutable: true
    bids        > [...]
    asks        > [...]
}

Ask ~ {
    id          integer($int32)
    typeTransaction string
    mutable: true
    customer    > [...]
    sneaker     > [...]
    size        number($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 > [...]
    releasedDate* string
    minLength: 1
    stock*      > [...]
    barcode*   string
    minLength: 1
    onlineImg* string
    minLength: 1
}

StockDTO >

```

```

Sneaker >

Stock >

IEntity >

Bid >

Customer >

Ask >

CustomerDTO >

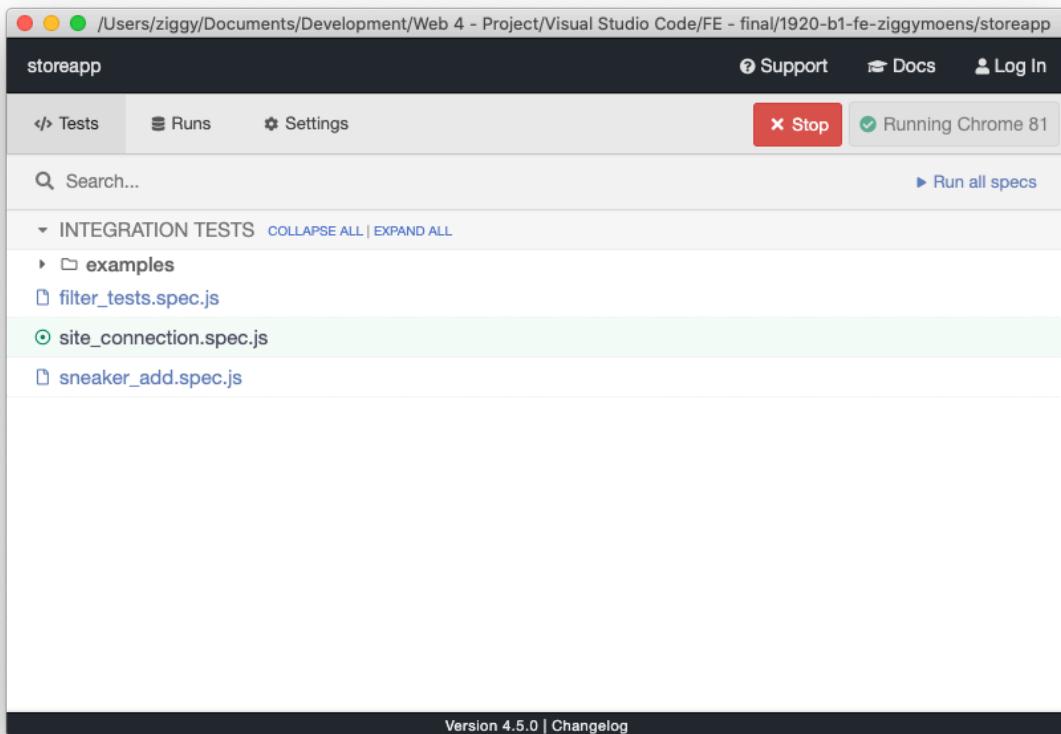
SneakerOutDTO >

StockDTO ~ {
    amount*    integer($int32)
    size*     number($double)
}

SneakerDTO ~ {
    name*       string
    minLength: 1
    price*      number($double)
    color*     string
    minLength: 1
    brand*     BrandDTO > [...]
    releasedDate* 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 foutbericht toont

- Site Connection
 - ✓ our app runs
 - ✓ http get for sneakers
 - ✓ mock sneaker get
 - ✓ on error should show error message

[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

Filter tests
✓ Name filter - 1
✓ Name filter - 2
✓ Brand filter - 1
✓ Brand filter - 2
✓ Scan filter - 1
✓ Scan filter - 2

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 bevestigingsbericht geeft
- De reset knop de velden correct leegmaakt

Sneaker add tests
✓ Open sneaker add page
✓ Name
✓ Color
✓ Price
✓ Date
✓ Barcode
✓ Brand
✓ Check sumbit
✓ fill
✓ Check reset