24 december 2021

Ilias Vynckier

Angular App

Mobile IoT Apps 2

Inhoud

[Inleiding 3](#_Toc91259618)

[Code 3](#_Toc91259619)

[Idee-material-design 3](#_Toc91259620)

[Idee-material-design.module.ts 3](#_Toc91259621)

[Idee-na.ts 3](#_Toc91259622)

[Idee-routing.module.ts 4](#_Toc91259623)

[Idee.ts 4](#_Toc91259624)

[Idee-toolbar 4](#_Toc91259625)

[Idee-toolbar.component.html 4](#_Toc91259626)

[Idee-toolbar.componetn.ts 5](#_Toc91259627)

[Idee-toolbar.compontent.css 5](#_Toc91259628)

[Idee-text-button 5](#_Toc91259629)

[Idee-text-button.component.html 5](#_Toc91259630)

[idee-text-button.component.ts 6](#_Toc91259631)

[Idee-text-button.component.css 6](#_Toc91259632)

[Idee-text-box 7](#_Toc91259633)

[Idee-text-box.component.html 7](#_Toc91259634)

[idee-text-box.component.ts 8](#_Toc91259635)

[Idee-text-box.component.css 9](#_Toc91259636)

[Idee-card 9](#_Toc91259637)

[Idee-card.component.html 9](#_Toc91259638)

[Idee-card.component.ts 9](#_Toc91259639)

[New-idee 10](#_Toc91259640)

[New-idee.component.html 10](#_Toc91259641)

[New-idee.component.ts 11](#_Toc91259642)

[New-idee.component.css 12](#_Toc91259643)

[Data-services 12](#_Toc91259644)

[State 13](#_Toc91259645)

[App.state.ts 13](#_Toc91259646)

[Idee.actions.ts 13](#_Toc91259647)

[Idee.effect.ts 14](#_Toc91259648)

[Idee.model.ts 14](#_Toc91259649)

[Idee.reducer.ts 15](#_Toc91259650)

[Idee.selector.ts 15](#_Toc91259651)

[Idee.state.ts 16](#_Toc91259652)

[Menu.actions.ts 16](#_Toc91259653)

[Menu.reducer.ts 16](#_Toc91259654)

[Menu.selector.ts 17](#_Toc91259655)

[Menu.state.ts 17](#_Toc91259656)

[App 18](#_Toc91259657)

[App-routing.module.ts 18](#_Toc91259658)

[App.component.html 18](#_Toc91259659)

[Resultaat 19](#_Toc91259660)

[LINK 20](#_Toc91259661)

[Besluit 20](#_Toc91259662)

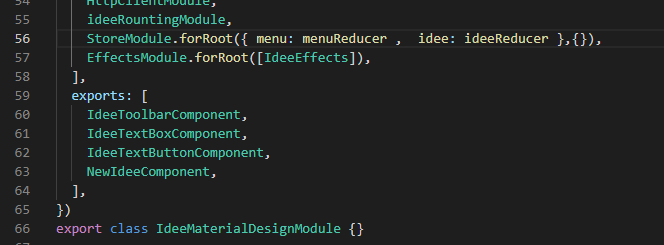
# Inleiding

In deze opdracht was het de bedoeling om een Angular app te maken. Ik had ervoor gekozen een app te maken die je kon gebruiken als je geen inspiratie had bij het bedenken van een idee om een app te maken. Er waren ook enkele vereisten we moesten gebruiken: Angular Material, Services, Inputs, Outputs, Forms, Routing en Navigation. De uitbreiding bestond eruit om ofwel testing toe te voegen of NgRx. Ik heb voor NgRx gekozen omdat ik dit niet zo goed verstond en door het in mijn app te gebruiken, wist ik dat ik het beter ging begrijpen.

# Code

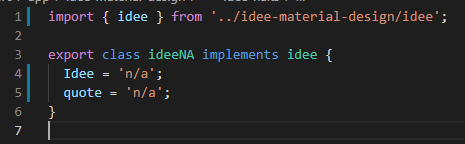
# Idee-material-design

### Idee-material-design.module.ts



In deze file worden alle modules geïmporteerd die we in de componenten gebruiken. In het bijzonder de Storemodule en Effects module.

### Idee-na.ts



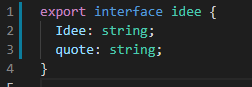
In deze file wordt de “Not Available” ideeën gemaakt. Deze wordt gebruikt in de idee-text-box component en idee-text-button component als @input.

### Idee-routing.module.ts



In deze file worden de routes van de componenten gedefinieerd. Samen met de routerlinks in de toolbar zorgen ze voor de navigatie in de app.

### Idee.ts



De idee interface wordt gebruikt door ideeNA.

## Idee-toolbar

### Idee-toolbar.component.html



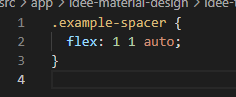
De toolbar is opgebouwd uit een mat-toolbar en mat-menu. Wanneer op het menu icoon wordt gedrukt, komt het menu tevoorschijn met daarin 2 keuzes “New” en “Idee”. Deze twee brengen je met behulp van de routerLink naar hun respectievelijke componenten.

### Idee-toolbar.componetn.ts



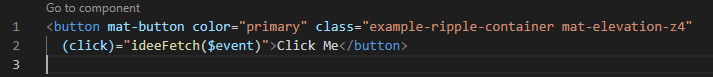
Wanneer er op het menu icon geklikt wordt, zal de MenuAction showMenu() gedispatched worden. Doordat het menu verdwijnt als je ergens anders op klikt, zal de state van het menu op andere plaatsen gedispatched worden zodat de state altijd correct is.

### Idee-toolbar.compontent.css



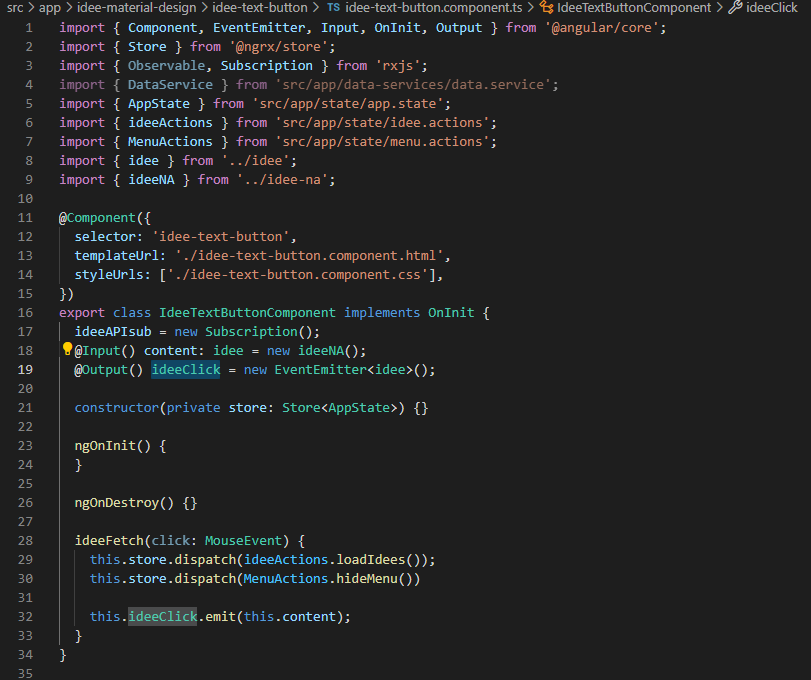
## Idee-text-button

### Idee-text-button.component.html



Als de button op de “hoofd component” wordt ingedrukt wordt de ideeFetch() functie uitgevoerd en wordt het event meegegeven.

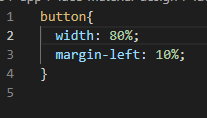
### idee-text-button.component.ts



Wanneer ideeFetch() wordt geactiveerd, zal er een dispatch gedaan worden naar de store. Door dit te doen, wordt loadIdees() geactiveerd. Deze haalt dan nieuwe data van de API en de nieuwe toestand wordt “opgeslaan” in de store. Hier wordt ook weer de state van de menu aangepast doordat het menu verdwijnt als je ergens anders dan het menu klikt.

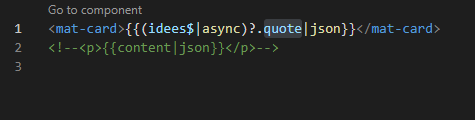
Ook wordt met ideeFetch() data geëmit. Dit is de ideeNA() data die we via de @Input() krijgen.

### Idee-text-button.component.css



## Idee-text-box

### Idee-text-box.component.html



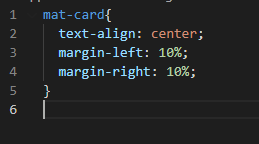
De ontvangen data van de api wordt in een mat-card getoond. Om deze te kunnen zien, moeten we gebruik maken van de async pipe. Op (idees$|async) kunnen we dan de .qoute gebruiken maar we moeten een ? gebruiken omdat deze technisch gezien ooit nul kan zijn. We moeten dan nog de Json pipe gebruiken anders krijgen we [object,object] te zien.

### idee-text-box.component.ts



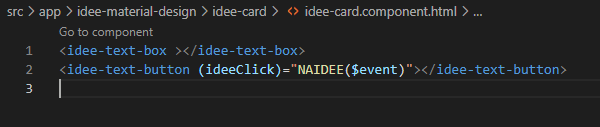
De nieuwe state wordt met de code op lijn 21 opgehaald. Als er nieuwe data binnengekomen van de API zal de state automatisch aangepast worden en doorgegeven aan de componenten die die state gebruiken. In de ngOnInit() zal er op de store geabonneerd worden om de state en aanpassingen van de state te krijgen. Met de code op lijn 32 zal een dispatch gedaan worden naar de store om data van de API op te halen.

### Idee-text-box.component.css



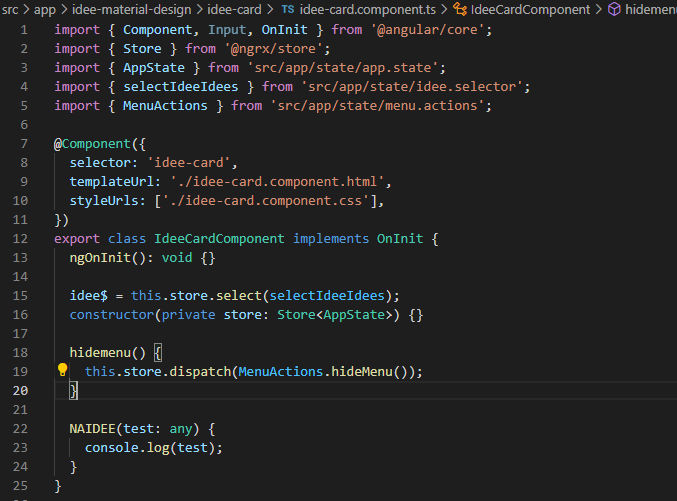
## Idee-card

### Idee-card.component.html



Doordat je maar 1 component kunt gebruiken in de routing module heb ik een extra component moeten aanmaken om de idee-text-box en de idee-text-button in 1 keer te tonen. Dit was een kleine aanpassing die niet veel extra code nodig had. Met (ideeClick)=”NAIDEE($event)” wordt de data die in de idee-text-button wordt geëmit “opgevangen”.

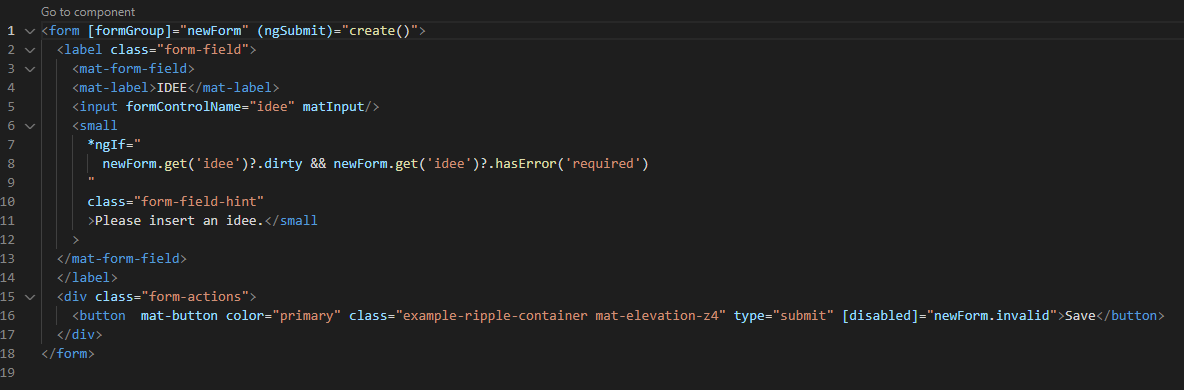
### Idee-card.component.ts



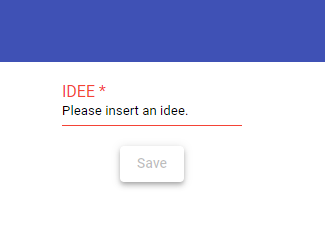
In deze file wordt het event van ideeClick in de console gelogd. Ook wordt de state van menu hier aangepast zodat er steeds de correcte state in de store zit.

## New-idee

### New-idee.component.html



Dit is de HTML voor het nieuwe ideeën form. Met \*ngIf wordt gekeken of de form al is aangeklikt en er data is ingevoerd. De submit knop zal enkel klikbaar zijn als je data in het input field hebt getypt.

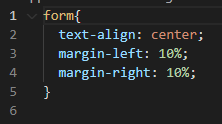


### New-idee.component.ts

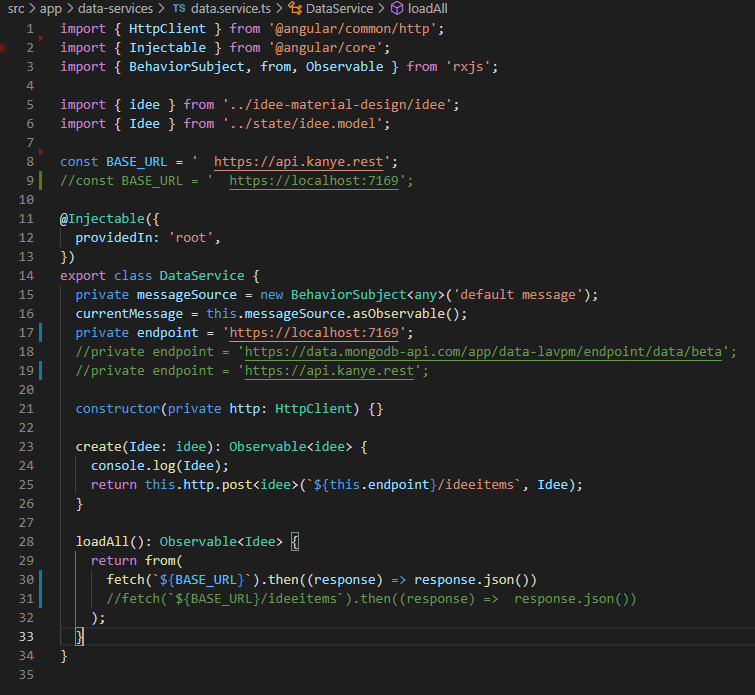


De data van de form wordt dan via de code op lijn 35 verzonden naar de API in de dataservice. Ik denk dat ik bookApiSubscription (door tijdsgebrek heb ik dit nog geen andere naam kunnen geven) niet meer nodig heb maar door tijdgebrek heb ik dit niet kunnen testen. De new-idee component is ook nog niet opgenomen in NgRx door tijdgebrek.

### New-idee.component.css



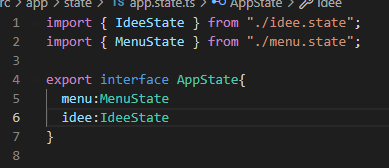
## Data-services



In data-service worden alle REST request naar de api uitgevoerd. Nu gebruik ik de kanye rest api maar als ik wat meer tijd had, wou ik mijn eigen api op punt stellen zodat ik die in deze app kon gebruiken. Met LoadAll() wordt een fetch naar de api gedaan en wordt data opgehaald. Met create() kan je een post doen naar de ideeAPI maar zoals ik al vermeld, had heb ik deze nog niet volledig kunnen afwerken.

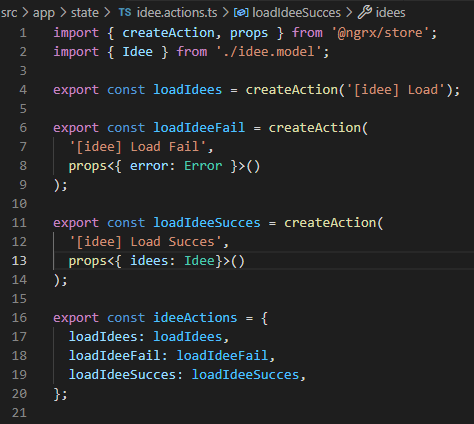
## State

### App.state.ts



In de appstate worden de verschillende states ‘samengevoegd’.

### Idee.actions.ts



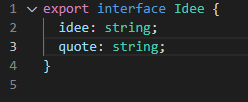
In deze file worden events beschreven die de api kan doen.

### Idee.effect.ts



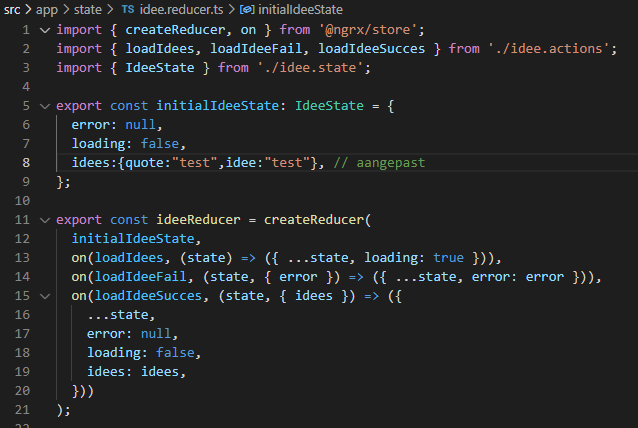
Met het effect kunnen we de api data in de store steken.

### Idee.model.ts



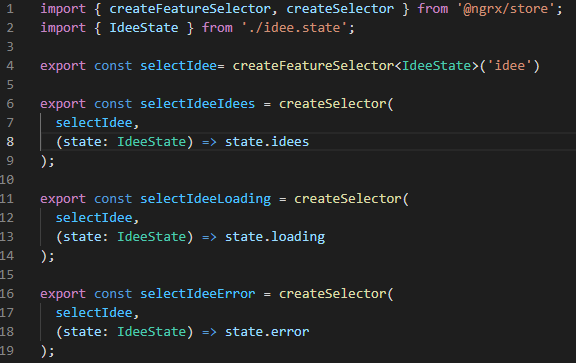
Idee model.

### Idee.reducer.ts



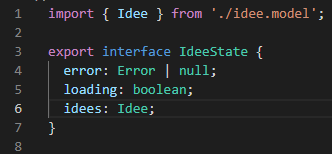
In de reducer van idee wordt als alles goed loopt de ontvangen data in de loadIdeeSucces state gestoken worden. Dit gebeurt met behulp van idee.effect.ts.

### Idee.selector.ts



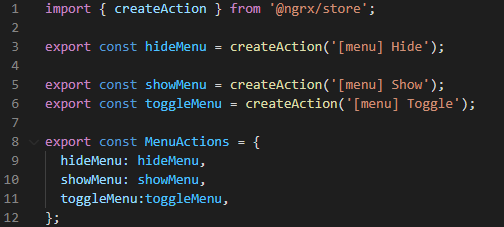
De selector van Idee. Hier wordt de laatste toestand van de state bijgehouden. Als er een probleem ontstaat bij het ontvangen van data van de API zal de error in de state van Idee geladen worden.

### Idee.state.ts



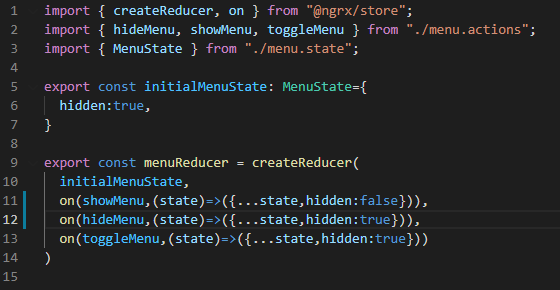
De “definiëring” van de idee state.

### Menu.actions.ts



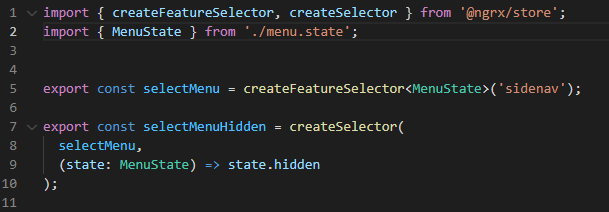
In deze file worden events beschreven die het menu kan doen.

### Menu.reducer.ts



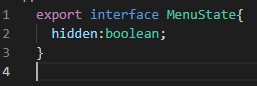
Hier zie je de reducer van Menu. Deze is verantwoordelijk voor het afhandelen van de transitie van een state tot de volgende state.

### Menu.selector.ts



De selector van Menu. Hier wordt de laatste toestand van de state bijgehouden.

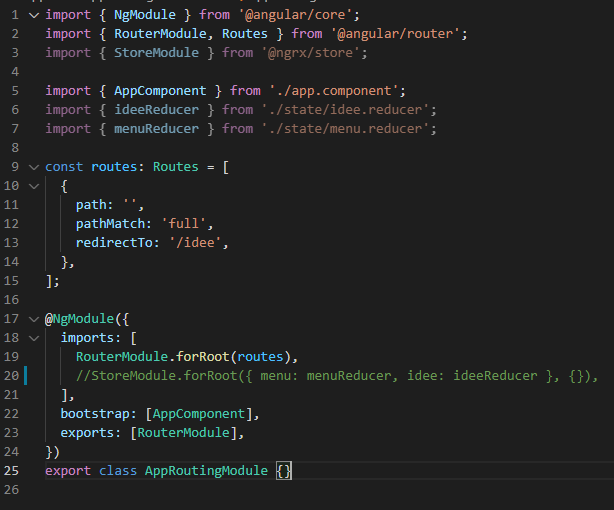
### Menu.state.ts



De interface van MenuState.

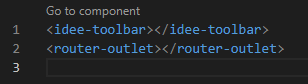
## App

### App-routing.module.ts



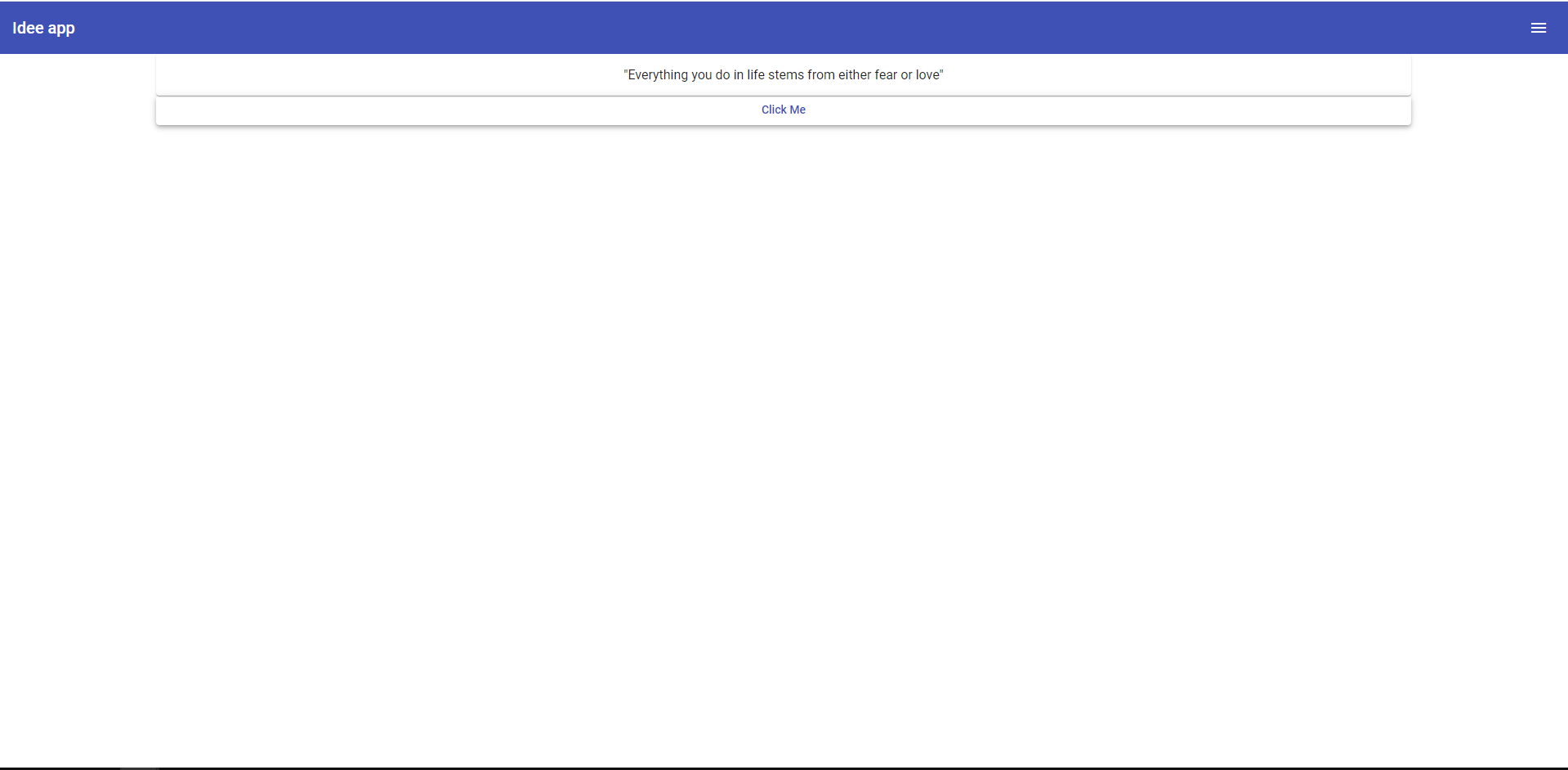
De routing module voor app. Als men de app opent, zal men direct doorverwezen worden naar /idee.

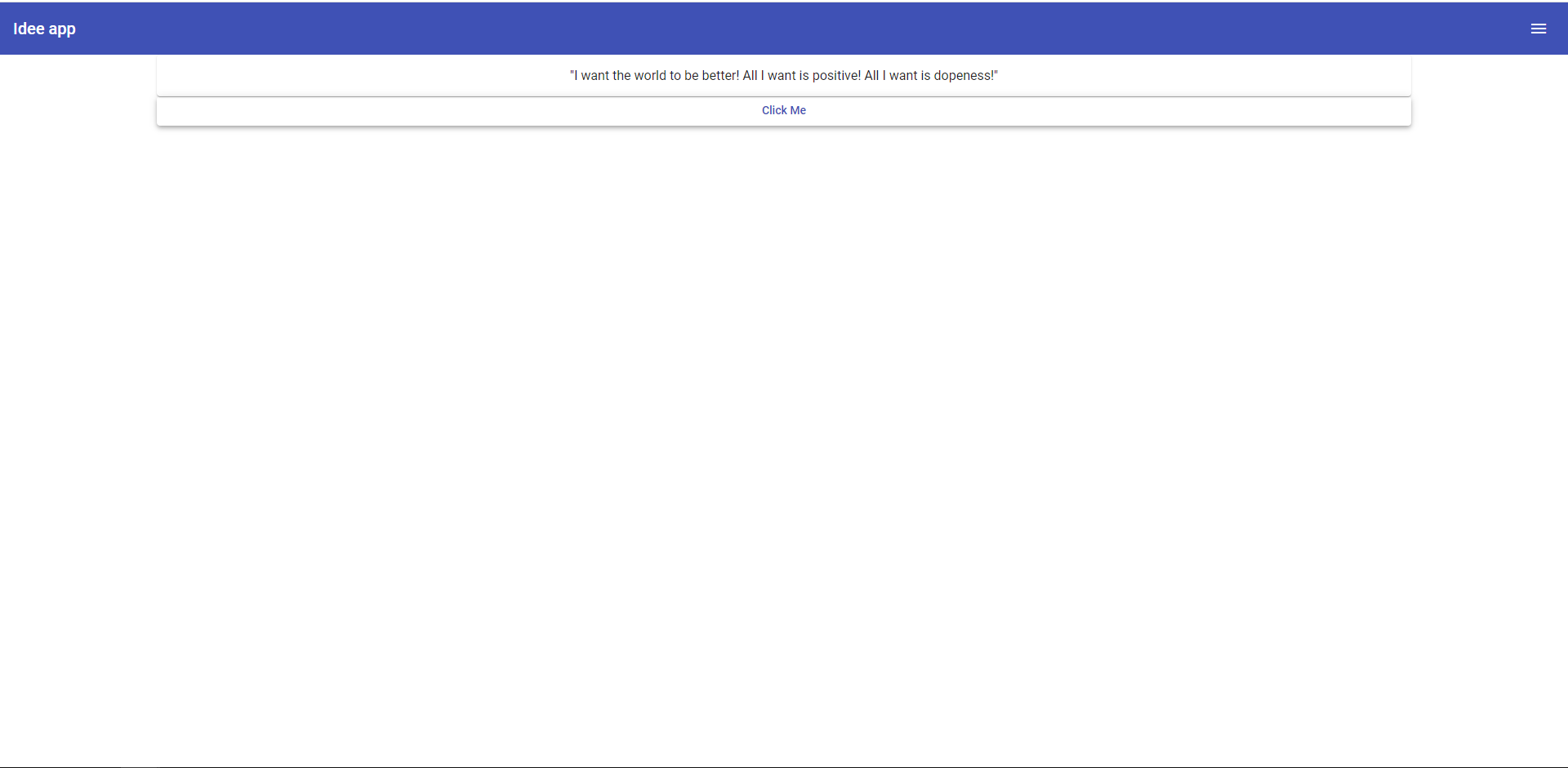
### App.component.html

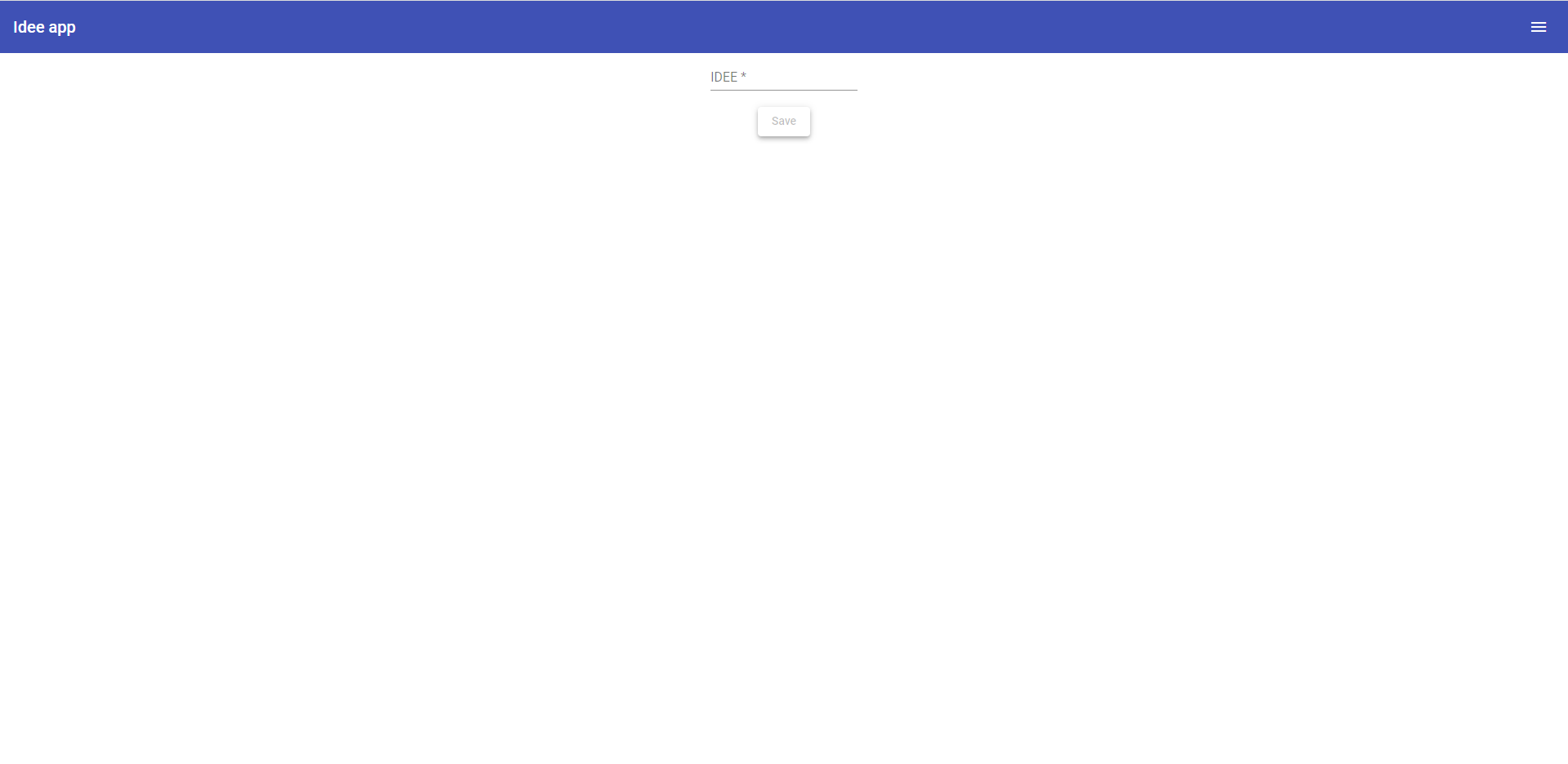


Doordat de toolbar op elke “pagina” mag blijven, staat deze in de app.component.html.

# Resultaat







# LINK

Github: <https://github.com/Ilias-Vynckier/Idee-app.git>

Azure: <https://testideeapp.azurewebsites.net/>

# Besluit

Deze opdracht heeft mij leren appreciëren hoe handig Angular kan zijn. Maar Angular blijft wel een steile leercurve hebben. Het kan snel heel veel worden waardoor men het overzicht kan verliezen. Door tijdsgebrek heb ik jammer genoeg niet mijn eigen API kunnen toepassen maar ik heb de code zodanig gemaakt dat de omschakeling vrij minimaal blijft. Toen we de gastles van Delaware kregen over NgRx verstond ik er helemaal niets van en zag het nut niet in om dit te gebruiken. Maar met het te gebruiken in mijn app heb ik het nut ontdekt. Het is niet eenvoudig en nogal complex maar het maakt de code meer overzichtelijk. In plaats van te werken met observables en inputs en outputs moet je enkel de state “updaten”. Je kan deze dan overal gaan ophalen/gebruiken. Dit heeft ervoor gezorgd dat de code om nieuwe data op te halen door te klikken op de button eenvoudiger is geworden. Nu klik je op de knop, de state wordt geüpdatet en NgRx zorgt ervoor dat de tekst in de text-box aangepast wordt.

Door tijdsgebrek is dit verslag ook niet naar de standaard die ik zou willen.