FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Labo Gebruikersinterfaces
Reeks 5: Angular (deel 1)

1 Inleiding

- Angular is een front end framework om complexe web apps te ontwikkelen.
- Angular is een volledige rewrite van Angular.js. Angular wordt ook Angular2 of Angular2+ genoemd. Wij gebruiken hier versie 11. Angular.js en Angular zijn niet compatibel. Als je iets opzoekt over Angular moet je daarom zeker zijn dat je niet bij Angular.js documentatie terecht komt.
- Angular gebruikt Typescript, een superset van ES6. Met Typescript heb je static typing binnen Javascript. Typescript wordt dan gecompileerd naar Javascript.
- Indien je je laptop gebruikt, zorg je dat je volgende zaken geïnstalleerd hebt:
  1. NodeJS en npm: https://nodejs.org/en/download/
  2. Angular CLI:

     ```
     npm install -g @angular/cli
     ```

     zie https://angular.io/guide/quickstart

2 Getting started

1. Open een command line venster en navigeer naar de (lege) map waarin je het angular-project wil aanmaken (commando `cd`).
2. Maak een nieuw angular project aan, dat nieuwsberichten zal tonen. De naam van het project is dan ook `news`. Dus komt er:

   ```
   ng new news --routing
   ```
3. Er wordt misschien gevraagd of je een Angular applicatie wil maken in `strict` mode. In deze mode zal de Typescript compiler extra controles uitvoeren. Voor deze labo-opdracht is dit niet belangrijk en kan je dus kiezen voor `N`.
4. Kies voor CSS als stylesheet format.
5. Ga nu naar de aangemaakte map *news*.
6. Voer volgend commando uit om het project te compileren en de webserver te starten:

```
ng serve
```

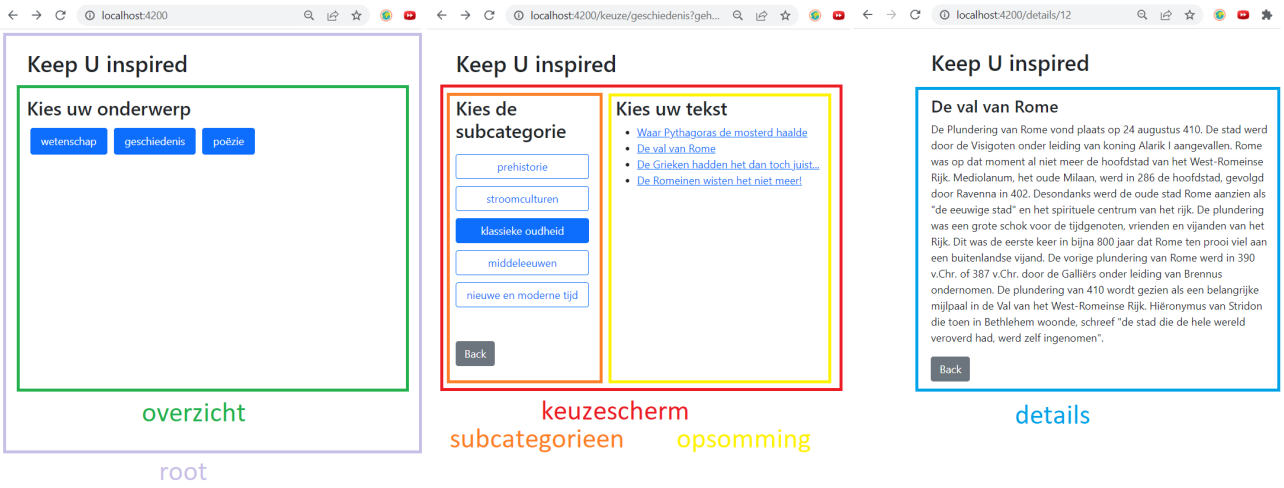
7. Open de browser en ga naar http://localhost:4200.
8. Open het project in een IDE naar keuze (voorkeur: IntelliJ) en verken de structuur. We gebruiken de IDE enkel om de bestanden aan te passen. Het compileren van de TypeScript en het runnen van de server worden automatisch door `ng` gedaan als je de bestanden opslaat.
9. Waar staat de html die de pagina beschrijft? Waar staat de CSS? Waar wordt de titel ingesteld? Wat is het verschil tussen de bestanden *index.html* en *app.component.html*? Welke code schrijf je waar? Merk op: de `<body>`-tag in *index.html* bevat maar één regel code; dat zal je ook later zo houden. En heb je in *app.component.html* de twee blokken commentaar gelezen over de *placeholder*? Die placeholder zal je straks vervangen.

3 Doel

In dit labo bouwen we een website from scratch op. Het is echter wel nuttig om te weten waar we naartoe willen.

3.1 Opbouw van de site

Figuur 1 toont de drie verschillende verschijningsvormen van die ene webpagina die het project bevat.



Figuur 1: Drie screenshot van website

In kleur wordt aangegeven uit welke componenten deze drie views bestaan.

root-component

De titel **Keep U inspired** is overal gelijk en staat dus in de rootcomponent. De bestanden (voor html, css en typescript (ts)) die tot de rootcomponent behoren, hebben allemaal een naam die start met `app.component`.

overzicht, keuzescherm en details

Via routing zorgen we ervoor dat de rootcomponent één van deze drie componenten als kind heeft: `overzicht`, `keuzescherm` of `details`. Merk de verschillen in de url van de drie screenshots.

Als er informatie van de ene component (bvb `Overzicht`) naar de andere (bvb `Keuzescherm`) doorgegeven moet worden, dan zal die extra informatie verpakt zitten in de naam van het pad. Dat bestaat dan uit een vast gedeelte en een parameter van primitief type (bvb de naam van de categorie die gekozen werd in `Overzicht`). De 'ontvangende' component (bvb `Keuzescherm`) kan dan aan het actieve pad deze parameter opvragen.

subcategorieen en opsomming

De component `keuzescherm` heeft zelf twee kindcomponenten: `subcategorieen` en `opsomming`.

Als er informatie van de ene component (`subcategorieen`) naar de andere component (`opsomming`) doorgegeven moet worden, zal dat gebeuren aan de hand van attributen en events die toegevoegd worden aan de `<app-subcategorieen>`- en `<app-opsomming>`-tags.

3.2 Werking van de site

Als de gebruiker naar de pagina surft, krijgt hij/zij eerst het overzicht te zien: een aantal categorieën van teksten (*wetenschap, geschiedenis, poëzie*). Dat is te zien in het eerste screenshot in Figuur 1.

Als er één van deze categorieën gekozen wordt, verandert de view naar een volgend keuzescherm. Dat staat op het tweede screenshot in Figuur 1. Links op dat keuzescherm staan de subcategorieën van de gekozen categorie, rechts staat in het begin niets. Zodra de gebruiker een subcategorie aanklikt, komt hier een lijst te staan van teksttitels uit de gekozen subcategorie.

Kiest de gebruiker een titel, dan verandert de view naar het laatste scherm. Dat is te zien in het derde screenshot van Figuur 1. Titel en inhoud van de tekst worden getoond.

3.3 Planning

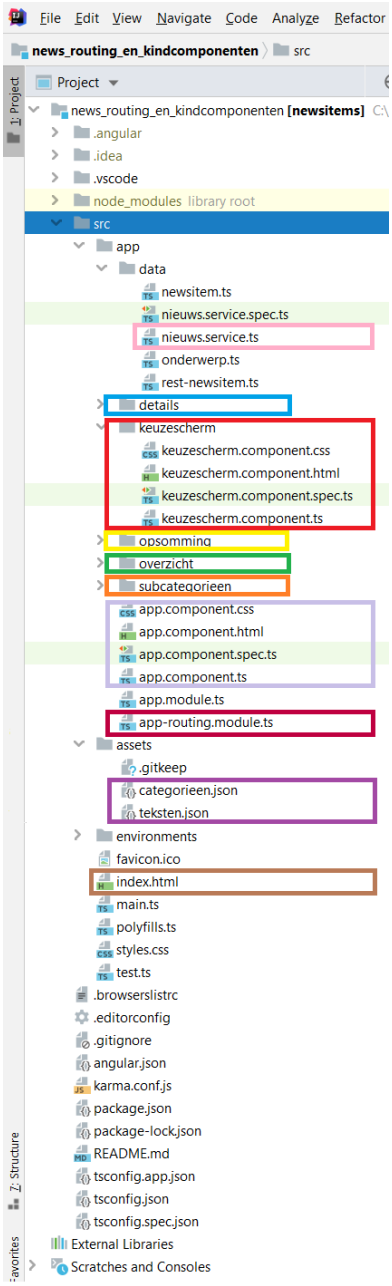
Voor dit project kunnen jullie vrij zelfstandig te werk gaan, of de werkwijze volgen die hieronder (soms summier, soms meer gedetailleerd) aangegeven wordt.

Wie zelfstandig aan de slag wil, vindt hieronder een aantal tips.

- Loop niet in zeven sloten tegelijk.
- Maak een plan op van wat je eerst wil uitwerken, en wat foutloos moet lopen voor een volgende functionaliteit geïmplementeerd wordt.
- Laat bepaalde details achterwege, als ze het debuggen van de belangrijke zaken niet in de weg staan. Zo kan je al wel meteen een service gebruiken om gegevens op te halen, maar zet daar voorlopig enkel hardgecodeerde waarden in. Dus zorg eerst dat je weet hoe de service te *gebruiken*, voor je ze ook helemaal *functioneel* gaat maken.
- Kies op voorhand of je meteen Bootstrap wil toepassen, of voorlopig zonder css wil werken. Ga leentjebuur spelen in de vorige projecten.

- Om alle newsitems tegelijk op te halen, is inlezen van een json-bestand een optie. Van zodra er specifieke newsitems opgehaald moeten worden, kan dat niet meer met een json-bestand en is er een REST API nodig.
- Scan de rest van de opgave, om te weten wat waar uitgelegd wordt. Doorloop de tekst grondiger als je aan dat onderwerp toe bent.

4 Projectstructuur



De structuur van het project zal er uiteindelijk uitzien zoals hiernaast. De componenten werden in dezelfde kleur aangeduid als in Figuur 1.

Het typescript-bestand `nieuws.service.ts` bevat de klasse `NieuwsService` die newsitems aanlevert. In het json-bestand `teksten.json` staan er newsitems opgesomd. Die worden door de service ingelezen. We kozen ervoor om de categorieën in een apart json-bestand op te sommen. Je zou die categorieën ook uit het bestand `teksten.json` kunnen halen, maar zoals gezegd: loop niet in zeven sloten tegelijk.

Het bestand `index.html` is het enige html-bestand met de tags `<html>`, `<head>` en `<body>`. De code die in de body staat, is niet langer dan wat hieronder te vinden is:

```
<body>
<div class="container">
  <div class="row">
    <app-root></app-root>
  </div>
</div>
</body>
```

Merk op: hier staan twee `<div>`-tags, die te maken hebben met het gebruik van Bootstrap.

Het bestand `app-routing.module.ts` bevat alle code die nodig zal zijn voor routing. Hier worden dus de paden gedefinieerd, en hun bestemming.

5 Tips bij foutmeldingen

Bij foutmeldingen controleer je of de link naar `styles.css` er nog staat (die mag weg), en of de code `<base href="/">` te vinden is in `index.html` (dat moet er wél staan).
Als IntelliJ bovenaan in het venster met de code een waarschuwing geeft over TSLint, dan klik je helemaal rechts op diezelfde regel, en laat je dit automatisch installeren. Daarna IntelliJ

afsluiten en opnieuw opstarten.

6 Angular componenten aanmaken

Om een nieuwe component aan te maken (met bijhorende map en bestanden), gebruik je in de Terminal (zie onderaan IntelliJ) of in een commandline-venster volgende code:

`ng generate component componentnaam  (of korter: ng g c componentnaam)`

Doen we dit voor de component `keuzescherm`, dan staat er binnen de map `src/app` nu een map `keuzescherm` met daarin:

1. *keuzescherm.component.html*: html template van de component
2. *keuzescherm.component.scss*: (s)css styling van de component
3. *keuzescherm.component.spec.ts*: unit tests voor de component
4. *keuzescherm.component.ts*: De javascript (typescript) code voor de component.

Dit kan je alvast doen voor alle componenten.

7 Routing

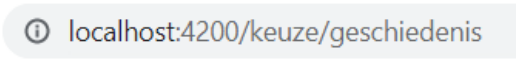
1. Bekijk het bestand `app.component.html`. Daar zal alvast de titel `Keep U inspired` komen. Wat daaronder getoond wordt, wordt bepaald door routing. Dus bevat dit bestand slechts twee regels code:

   ```
   <h1>Keep U inspired</h1>
   <router-outlet></router-outlet>
   ```

2. Definieer de paden in het bestand `app-routing.module.ts`.

   (a) Om bij de start meteen de component `overzicht` te zien, moet je het lege pad instellen, dus met waarde `''`.

   (b) Als de gebruiker een categorie kiest in de `OverzichtComponent`, dan wordt de gekozen categorie meegegeven in de padnaam, en wordt er genavigeerd naar de `KeuzeschermComponent`. Je kan de naam van het pad (met de meegegeven parameter) zien in de url van dat deel van de webpagina.

   In bovenstaande url zie je `geschiedenis` in woorden staan. Allicht is het een beter idee om hier de id van de categorie mee te geven, zodat opzoeken van de juiste subcategorieën makkelijker gaat.

3. Omdat de overgang van `OverzichtComponent` naar `KeuzeschermComponent` via navigatie gebeurt, bevatten de drie blauwe buttons (of tekstvelden, of list-items) in de OverzichtComponent een attribuut dat instaat voor de navigatie. In gewone html zou dat het attribuut `href` zijn, in dit geval gebruiken we het attribuut `routerLink`. Er wordt dus niet gewerkt met `onclick`.

4. Houd in eerste instantie de `KeuzeschermComponent` nog heel eenvoudig. Je voorziet dus voorlopig nog geen kinderen (`SubcategorieenComponent` en `OpsommingComponent`).

Toon op die pagina wel welke categorie op de vorige pagina geselecteerd werd. Die informatie kan je opvragen aan het actieve pad (het pad langs waarheen de gebruiker op de huidige component geraakt is).

5. Wacht nog even met de routing van het `Keuzescherm` naar de `Details`.

   Wil je dit tóch al in orde maken, voorzie in de `KeuzeschermComponent` een paar (=twee) hardgecodeerde titels. Wordt op één van die titels geklikt, dan wordt er naar de `KeuzeschermComponent` genavigeerd, mét de titel als parameter.

8 Structural directives

Heb je tot nu toe zonder lussen gewerkt om een klein aantal gelijkaardige html-elementen te implementeren? Heb je met andere woorden ergens een variant staan van onderstaande code?

```
<li><a routerLink="...">wetenschap</a></li>
<li><a routerLink="...">geschiedenis</a></li>
<li><a routerLink="...">po&euml;zie</a></li>
```

Dan is het tijd om *structural directives* te gebruiken, op z'n minst `*ngFor`. Daarvoor declareer je in de TypeScriptklasse die bij de component hoort, een instantievariabele van het type list. Initialiseer deze instantievariabele ook - *hoe* dat gebeurt speelt nog niet zoveel rol. Dus hardcoderen met de drie woorden `"wetenschap","geschiedenis","poëzie"` is voorlopig goed genoeg.

9 Service en REST API

Data voor de website worden aangeleverd door een service. In het screenshot van de projectstructuur (blz 4) vind je het bestand `nieuws.service.ts`. (Dit bestand zal je zelf nog moeten aanmaken, genereer het aan de hand van `ng generate`.) Daarin staat de klasse `NieuwsService`, die in eerste instantie alleen een methode heeft om alle nieuwsitems tegelijk op te halen. Dat kan je nog implementeren door het inlezen van een json-bestand.

Later wordt de service uitgebreid: specifieke nieuwsitems kunnen op id opgevraagd worden; en voor een gegeven categorie kan een lijst van subcategorieën opgevraagd worden.

Dat wil dus zeggen dat er gegevens opgevraagd worden aan de hand van een parameter. In dat geval hebben we een REST API nodig. Kijk voor de uitleg hiervan naar vorig labo. Op onderstaande links vind je twee REST API's die je kan gebruiken.

- nieuwsitems
- categorieën met hun subcategorieën

Voor deze API's is enkel de GET methode geactiveerd om alle items of één individueel item op te halen a.d.h.v. een id. Je kan deze al eens uit proberen in de URL-balk van je browser.