



Inhoudstafel

[Beschrijving van het probleemdomein 3](#_heading=h.gjdgxs)

[De zaakvoerder 3](#_heading=h.30j0zll)

[De klant 4](#_heading=h.1fob9te)

[Structuur van het programma 4](#_heading=h.3znysh7)

[Frontend klant 5](#_heading=h.2et92p0)

[Home pagina 5](#_heading=h.tyjcwt)

[Products pagina 5](#_heading=h.3dy6vkm)

[Login pagina 5](#_heading=h.1t3h5sf)

[Registratie pagina 5](#_heading=h.4d34og8)

[Checkout pagina 6](#_heading=h.2s8eyo1)

[Betaal pagina 6](#_heading=h.17dp8vu)

[Basket pagina 6](#_heading=h.3rdcrjn)

[Frontend Admin 6](#_heading=h.26in1rg)

[Admin pagina 6](#_heading=h.lnxbz9)

[Klanten pagina 7](#_heading=h.35nkun2)

[Order pagina 7](#_heading=h.1ksv4uv)

[Product beheer pagina 7](#_heading=h.44sinio)

[Backend 7](#_heading=h.2jxsxqh)

[Rest service 7](#_heading=h.z337ya)

[Google Firestore 10](#_heading=h.1y810tw)

[Uiteenzetting technologie 11](#_heading=h.4i7ojhp)

[Frontend 11](#_heading=h.2xcytpi)

[Angular 12](#_heading=h.1ci93xb)

[Angular Material 12](#_heading=h.3whwml4)

[Angular FireStore 12](#_heading=h.2bn6wsx)

[Git 12](#_heading=h.qsh70q)

[NodeJs 12](#_heading=h.3as4poj)

[XLSX 12](#_heading=h.1pxezwc)

[Backend 12](#_heading=h.49x2ik5)

[Dotnet Core 12](#_heading=h.2p2csry)

[Entity Framework 12](#_heading=h.147n2zr)

[Automapper 13](#_heading=h.3o7alnk)

[EPPlus 13](#_heading=h.23ckvvd)

[ExcelDataReader 13](#_heading=h.ihv636)

[iTextSharp 13](#_heading=h.32hioqz)

[Dotnet Mailkit 13](#_heading=h.1hmsyys)

[Uiteenzetting technologie 16](#_heading=h.28h4qwu)

[Backend 16](#_heading=h.nmf14n)

[Frontend 16](#_heading=h.37m2jsg)

[functionele vereisten 16](#_heading=h.41mghml)

[use case diagram 16](#_heading=h.2grqrue)

[overige (niet-)technische vereisten 16](#_heading=h.vx1227)

[kostprijs 16](#_heading=h.3fwokq0)

[overzicht van de architectuur 16](#_heading=h.1v1yuxt)

[interactiediagrammen 17](#_heading=h.4f1mdlm)

[dataschema’s 17](#_heading=h.2u6wntf)

[beschrijving van deployment mogelijkheden, installatie procedure 17](#_heading=h.19c6y18)

[referenties 17](#_heading=h.3tbugp1)

# Beschrijving van het probleemdomein

Als projectvoorstel heb ik gekozen om een webshop te maken dat de aankoop van voedingswaren mogelijk maakt. Ik heb dit onderwerp gekozen omdat ik zelf nog in een warenhuis gewerkt heb, en de aankoop online van voeding toch een must vind voor hedendaagse winkels. Zo kan men de bereikbaarheid vergroten en meer omzet draaien (alles draait rond cijfers). De webshop die ik gemaakt heb, zal ook gebruikt kunnen worden voor bv. Auto onderdelen, fietsen, enz… Voor dit eindwerk hou ik het bij voedingswaren.

Dit is een low level e-commerce system dat de mogelijkheid maakt om bestellingen te plaatsen als klant, bestellingen te verwerken, producten opladen en klanten beheren als zaakvoerder.

## De zaakvoerder

De zaakvoerder zal toegang krijgen tot de admin functionaliteiten. Zo zal de zaakvoerder verantwoordelijk zijn voor het opladen van de artikelen met de juiste hoeveelheid voorraad, zorgen voor een gepaste foto, benaming, referentie.

De zaakvoerder is verantwoordelijk voor het bezorgen van de bestellingen voor de klanten nadat de klanten hun bestelling bevestigd hebben.

De admin kan producten downloaden via excel, de producten aanpassen in excel en terug uploaden via excel. Er is ook een onderhoudsscherm om gekozen producten aanpassen of verwijderen.

Bij het downloaden van de excel zal men alle producten zien die aanwezig zijn in de database. Producten waarvan de voorraad onder 10 komen te staan zullen in het rood aangeduid staan om zo aan te geven dat de zaakvoerder zijn stock zal moeten bijvullen.

Inkomende orders verwerken gebeurt in een onderhoudsscherm waarbij men een overzicht heeft over alle openstaande orders. Het is aan de zaakvoerder om alle orders te verwerken. Ieder order beschikt over een PDF document met een overzicht van de totale bestelling. De zaakvoerder kan ervoor kiezen het document af te drukken of gewoon te selecteren welke orders er behandeld geweest zijn. Vervolgens moeten de orders gepickt worden, om die dan vervolgens nar de klant op te sturen.

## De klant

Bezoekers van de site kunnen vrij producten opzoeken en hun detail bekijken.

De klant zal zich moeten registreren om de aankoop te kunnen voltooien of om de producten die klaar staan aan te kopen op te slaan om bij het volgende bezoek aan de website gewoon verder te kunnen doen met de aankoop van producten. De producten die de klant geselecteerd heeft om te kopen worden bijgehouden in een basket. De volgende keer dat de klant zich inlogt kan hij/zij gewoon verder winkelen met de artikelen die men ervoor al gekozen heeft. Dit om een comfortabele en efficiënte winkelervaring aan te bieden. Nadat de klant zijn aankoop wenst te voltooien, dan zal een email gestuurd worden met de factuur/overzicht van de producten die men gekocht heeft op dat moment.

De klant heeft na zich geregistreerd te hebben de mogelijkheid om zich in te loggen met zij email en paswoord of met behulp van zijn Gmail account.

# Structuur van het programma

Het programma bestaat uit een front end (customer interface) en een backend. De gebruiker, namelijk de klant zal gebruik maken van de frontend. De backend wordt gebruikt om de frontend te voeden. Er zijn twee backend systemen, namelijk de REST service en de authenticatie service van Google firebase. Elk van hen is een microservice. Dat wil zeggen als de ene service uitvalt zal de andere nog kunnen werken. Dit om een volledige outage te voorkomen.

## Frontend klant

De frontend bestaat uit verschillende componenten met elk hun eigen functionaliteit. Ieder component maakt gebruik van een service die de data zal gaan ophalen of verwerken voor de component die de service gebruikt. Het programma start bij de “index.html” pagina. Daar wordt de hoofdcomponent aangeroepen (“app-root”) en die zal de verdere communicatie leggen tussen de componenten.

### Home pagina

De startpagina van de webshop is de homepagina waar men direct een overzicht heeft over de producten die in aanbieding zijn. Vanuit de homepagina kan men navigeren naar andere pagina’s via de navigation bar.

### Products pagina

Deze pagina is voorzien om alle producten weer te geven. Daar kan men een product aanklikken en zijn details bekijken. Details worden weergegeven in een apparte pagina. Als de klant ingelogd is zullen de “Buy” knoppen verschijnen en zal de klant een aantal kunnen ingeven.

### Login pagina

De loginpagina is voorzien om de klant zijn inloggegevens te laten invullen, of de Google login te gebruiken. Als de klant nog niet geregistreerd is dan zal men doorverwezen worden naar de registratiepagina.

### Registratie pagina

De registratiepagina is voorzien om de klant zijn persoonlijke gegevens te laten invullen. In verschillende stappen zal de klant volgende gegevens moeten invullen:

* Voornaam en achternaam
* Straat en huisnummer
* Postcode, gemeente en land
* Telefoonnummer
* E-mail adres en een zelfgekozen paswoord

Alle velden moeten ingevuld worden om de registratie te kunnen voltooien.

### Checkout pagina

De checkout pagina geeft een overzicht van alle items die de klant op dat moment zal bestellen. Het totale bedrag per orderlijn en het volledige bedrag van de bestelling worden daar weergegeven. De klant zal daar alsnog de mogelijkheid hebben om producten te verwijderen. Vanuit deze pagina zal de klant verder kunnen naar de betaalpagina.

### Betaal pagina

De betaalpagina is enkel voorzien van een knop om de bestelling te bevestigen. Een afbeelding van een dummy betaalpagina wordt weergegeven ter simulatie. Nadat de bestelling bevestigd is zal de klant teruggestuurd worden naar de homepagina.

### Basket pagina

De basket geeft een overzicht van alle producten die de klant gekozen heeft. Het is hetzelfde principe als de checkout pagina, maar dan kleiner. Deze pagina komt tevoorschijn bij het hoveren over het basket logo. Dit is enkel mogelijk als de klant ingelogd is.

## Frontend Admin

De admin heeft alle functionaliteiten die de klant heeft. Het admin gedeelte is ervoor bedoeld om de webshop te voorzien van data, en om de zaakvoerder bestellingen te laten verwerken.

### Admin pagina

Deze pagina is voorzien om een overzicht te geven over alle functionaliteiten die de admin heeft. De admin zal als klant geregistreerd moeten zijn in het systeem, en zal hij als admin moeten geselecteerd staan. Om als admin geselecteerd te zijn dan zal de programmeur van dit programma dit moeten wijzigen.

De admin pagina bestaat uit 3 delen:

* Klanten
* Orders
* Producten

Bij klanten krijg je een overzicht van alle geregistreerde klanten in het systeem. Daar is het niet mogelijk om klantgegevens te verwijderen. In het order gedeelte zal je een overzicht krijgen van alle openstaande of gesloten orders. En in het producten gedeelte zal je een overzicht hebben van alle producten, en kan men daar CRUD bewegingen toepassen via Excel of via het onderhoudsscherm.

### Klanten pagina

Deze pagina voorziet enkel een overzicht van de geregistreerde klanten. Bij het klikken op een klant zal de rij uitschuiven en zal het adres worden weergegeven.

### Order pagina

Deze pagina geeft een overzicht van alle openstaande of gesloten orders. Er is een knop voorzien om te wisselen tussen open en gesloten orders. De admin zal daar elke order zien in een tabel en bij het aanklikken van een order zullen de bijhorende orderlijnen uitschuiven. Een knop is voorzien bij elke order om de volledige order te sluiten, of kan de admin ieder orderlijn apart sluiten. Dit om een overzicht te bewaren welke lijnen van het order er al verwerkt zijn. Ook heeft men de mogelijkheid om de factuur van die order te downloaden in PDF format.

### Product beheer pagina

Deze pagina is voorzien van een overzicht van alle producten. Daar kan men ook doorklikken naar de detail van ieder product en daar eventuele aanpassingen aanbrengen. De admin kan op deze pagina een excel downloaden van alle producten in het systeem. De Excel kan aangepast worden en terug opgeladen worden in het systeem. Deze manier van werken zal het meest efficiënt zijn. Kleinere aanpassingen kunnen alsnog gedaan worden in het detailscherm van een product. De admin kan ook zelf nieuwe producten ingeven en of producten verwijderen.

## Backend

### Rest service

Voor de REST service wordt er gebruik gemaakt van verschillende controllers die aangeroepen kunnen worden door een url request te zenden. Alle CRUD bewegingen zijn aanwezig voor volgende entiteiten:

* Klanten
* Producten
* Orders
* BasketItems

Klantgegevens worden opgeslagen in de database. Iedere klant heeft een adres dat gekoppeld is met de klant. De tabel producten heeft algemene naamgeving voor zijn velden. Zo kan je gelijk welke soort producten opladen. Basket items zijn ervoor bedoeld om de gekozen artikelen van de klant bij te houden totdat de klant zijn aankoop bevestigd. Nadat de klant bevestigd worden de basket items uitgelezen van die klant, en van iedere basket item wordt er een orderlijn aangemaakt. Na deze bewerking worden alle basket items verwijderd. Orders worden aangemaakt nadat de klant zijn order heeft bevestigd. Na het bevestigen van een order zal het systeem een email sturen naar de klant met een overzicht van het order in een PDF bestand, en een dankwoord in de mail zelf. De REST service is niet voorzien van beveiliging.

##### Products

In de products controller kan men volgende REST requests aanspreken:

|  |  |  |
| --- | --- | --- |
| Request | Url | Actie |
| Get | /api/products | Alle producten in database |
|  | /api/products/ + id | Een product via id |
|  | /api/products/DownloadExcel | Lijst met producten in Excel downloaden |
| Post | /api/products | Product in het systeem plaatsen |
|  | /api/products/list | Lijst met producten toevoegen in het systeem |
|  | /api/products/uploadExcel | Excel lijst met producten uploaden in het systeem |
| Put | /api/products/ + id | Product updaten via id |
| Delete | /api/products/ + id | Product verwijderen via id |

##### Klanten

In de customer controller kan men volgende REST requests aanspreken:

|  |  |  |
| --- | --- | --- |
| Request | Url | Actie |
| Get | /api/Customers/ + uid | Klant ophalen met UID |
|  | /api/customers | Alle klanten ophalen |
| Post | /api/customers | Nieuwe klant toevoegen |
| Put | /api/Customers/ + uid | Klant updaten met UID |
| Delete | /api/Customers/ + uid | Klant verwijderen met UID |

##### Orders

In de products controller kan men volgende REST requests aanspreken:

|  |  |  |
| --- | --- | --- |
| Request | Url | Actie |
| Get | /api/orders/ + status | Orders ophalen van een bepaalde status |
|  | /api/orders/openlines/ + ordId | Open orderlijnen van een order ophalen via ordId |
|  | /api/orders/closedlines/ +ordId | Gesloten orderlijnen van order ophalen via ordId |
|  | /api/orders/invoiceOrder/ + ordId | Factuur als PDF downloaden |
| Post | /api/orders/ + custId | Order plaatsen |
| Put | /api/Orders/ + ordId + "/closeline/" + lineNo | Orderlijn sluiten |
|  | /api/Orders/ + ordId + "/openline/" + lineNo | Orderlijn heropenen |
|  | /api/Orders/close/ + ordId | Order sluiten (incl orderlijnen) |
|  | /api/Orders/open/ + ordId | Order openen (incl orderlijnen) |
| Delete |  | Orders kunnen niet verwijderd worden |

##### BasketItems

In de basket controller kan men volgende REST requests aanspreken:

|  |  |  |
| --- | --- | --- |
| Request | Url | Actie |
| Get | /api/BasketItem/ + custId | Alle basketItems teruggeven |
| Post | /api/BasketItem | Basketitem opslaan |
| Put | /api/BasketItem | Lijst met basketitems aanpassen |
| Delete | /api/BasketItem/ + id | Item uit basket verwijderen |

##### Documenten

###### PDF

Bij alle orders die geplaatst worden zal er een PDF document aangemaakt en opgeslagen worden op de server. Dit als backup en voor boekhoudkundige doeleinden. De admin kan op vraag al die documenten verkrijgen. Of via de frontend kunnen alle documenten opgevraagd worden per order.

###### Excel

De excel die gegenereerd wordt zal enkel opgeslagen worden in het systeem waar de request gemaakt werd.

### Google Firestore

Deze service leek me handig om authentificatie te garanderen. Deze service is gratis en heeft de mogelijkheid om de pro versie te gebruiken tegen betaling. De gebruikers hun wachtwoorden worden opgeslagen zonder dat iemand hun wachtwoord kan achterhalen. Dit supporteerd ook het resetten van wachtwoorden. Elke nieuwe klant zijn inloggegevens worden opgeslagen in de Firestore database. De data staat gelinkt met de REST service. Dus geen authentificatie wil zeggen dat de klant niet geregistreerd is in het systeem van de werkgever. Ook de schaalbaarheid van deze service is makkelijk uit te breiden. Ook handige features zijn het kunnen aanmelden met je Google login of eventueel Facebook login, en zoveel meer. In deze implementatie is het enkel voorzien om te kunnen inloggen met username en password of met je Google login.

##### Authentication

De frontend is afgeschermd met een “Auth” service. Dit wil zeggen zolang de gebruiker zich niet heeft ingelogd kan hij niet navigeren naar bepaalde pagina’s van de webshop. De meeste apps moeten de identiteit van een gebruiker kennen. Door de identiteit van een gebruiker te kennen, kan een app gebruikersgegevens veilig opslaan in de cloud en dezelfde gepersonaliseerde ervaring bieden op alle apparaten van de gebruiker.

Firebase-verificatie biedt backend-services, gebruiksvriendelijke SDK's en kant-en-klare UI-bibliotheken om gebruikers voor uw app te authenticeren. Het ondersteunt authenticatie met behulp van wachtwoorden, telefoonnummers, populaire federatieve identiteits providers zoals Google, Facebook en Twitter, en meer.

Firebase Authentication kan eenvoudig worden geïntegreerd met uw aangepaste backend.

Als u een gebruiker bij uw app wilt aanmelden, krijgt u eerst authenticatiegegevens van de gebruiker. Deze gegevens kunnen het e-mailadres en wachtwoord van de gebruiker zijn, of een OAuth-token van een federatieve identiteitsprovider (Google, Facebook, …). Vervolgens geeft u deze gegevens door aan de Firebase Authentication SDK. De backend-services zullen vervolgens die inloggegevens verifiëren en een reactie aan de klant retourneren.

Na een succesvolle aanmelding heeft u toegang tot de basisprofiel-informatie van de gebruiker en kunt u de toegang van de gebruiker beheren tot gegevens die zijn opgeslagen in andere Firebase-producten. U kunt ook het verstrekte authenticatietoken gebruiken om de identiteit van gebruikers in uw eigen backend-services te verifiëren.

# Uiteenzetting technologie

Voor dit project leek het me interessant om te kiezen voor Angular als front-end en C# als backend. Angular is een framework die de programmeur voorziet om iedere pagina als een component laat maken zodat er een duidelijk structuur ontstaat bij het implementeren. Angular gebruikt Typescript als programmeertaal, en is een aftakking van JavaScript.

## Frontend

Gebruikte technologien en libraries:

* Angular
* Angular material
* Angular FireStore modules
* Github
* Git Bash
* Node.js
* XLSX

### Angular

Een voorbeeld hiervan is:

### Angular Material

### Angular FireStore

### Git

### NodeJs

### XLSX

## Backend

Gebruikte technologien en libraries:

* Dotnet Core
* Entity framework
* Automapper
* EPPlus
* ExcelDataReader
* iTextSharp
* Dotnet Mailkit

### Dotnet Core

### Entity Framework

### Automapper

### EPPlus

### ExcelDataReader

### iTextSharp

### Dotnet Mailkit

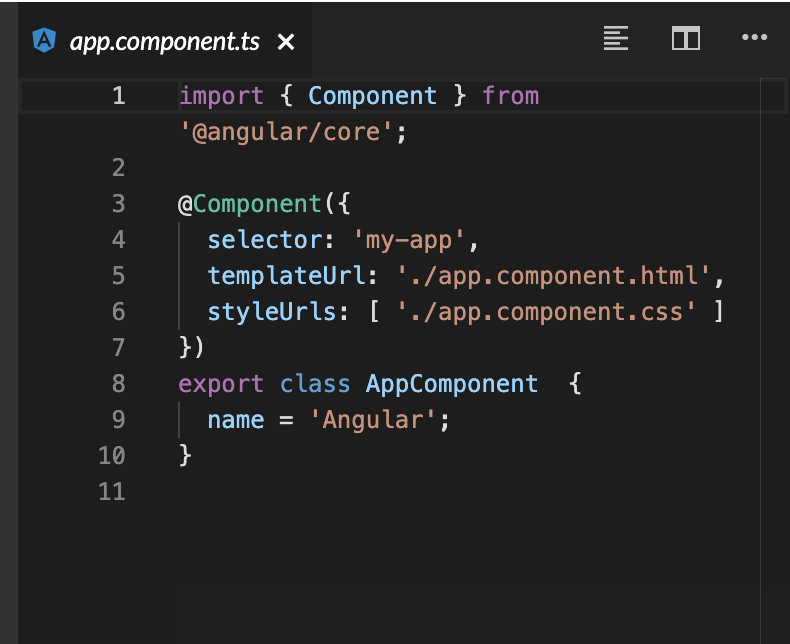


De “app-root” roept de “app-parrent” aan, die op zijn beurt de “app-child” aanroept. Dus alle 3 de componenten zijn op dit moment ingeladen in de browser.

Ieder component bestaat uit 4 delen:

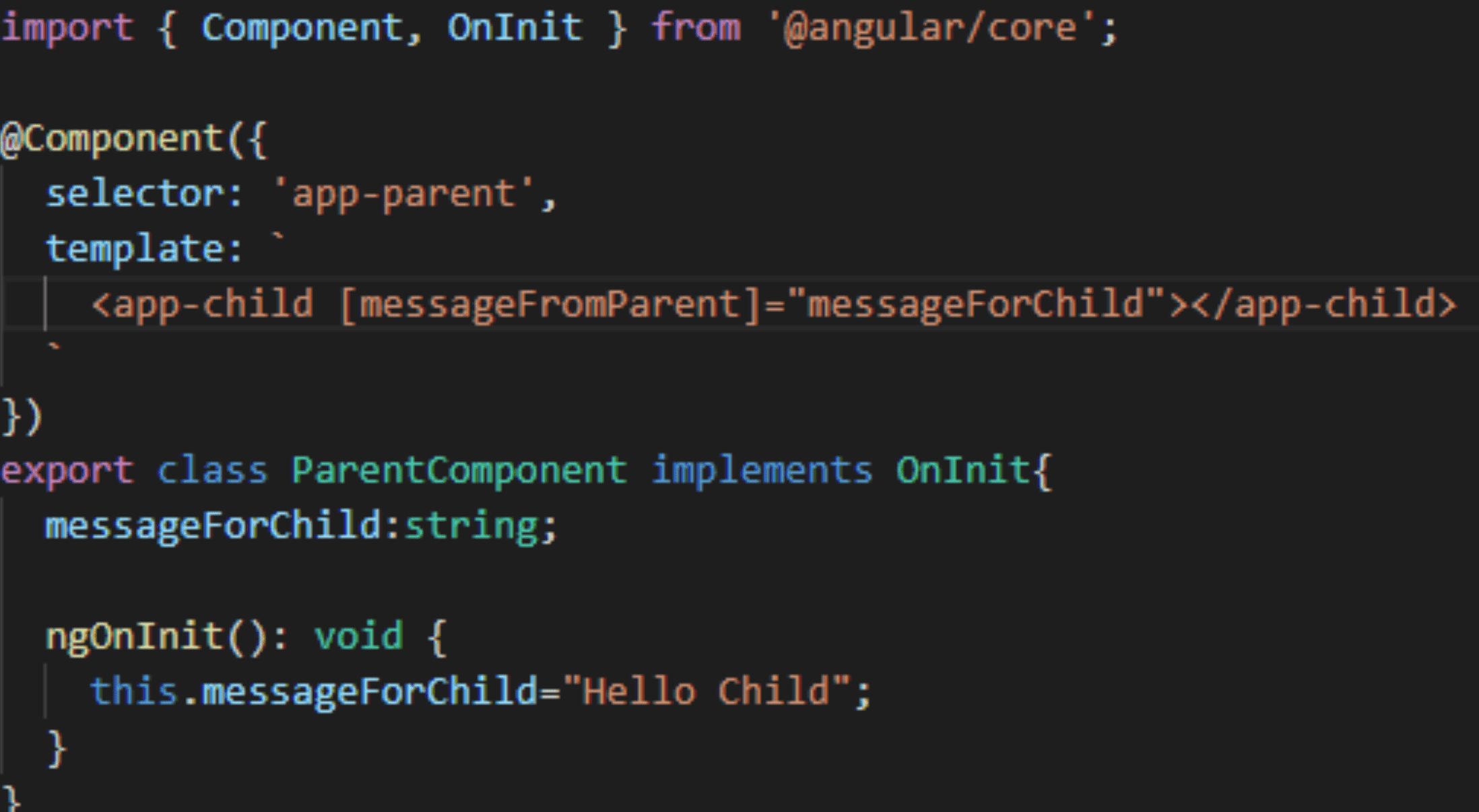
1. CSS pagina
2. HTML pagina
3. Spec.ts pagina
4. TS pagina

De CSS zit in een aparte file, zodat dit een beter overzicht behoudt aan de implementatie. Ook de HTML die de CSS gebruikt zit apart voor een beter gestructureerd overzicht te hebben. De functionaliteiten van het programma zitten in de .ts pagina. De “.ts” staat voor TypeScript pagina. Zonder de ts pagina zal het programma niet werken omdat op die plaats alles wat behoort tot het component ingeladen wordt. Het ook mogelijk is om de CSS in de HTML pagina te implementeren, of zelfs de HTML code in de .ts file te implementeren. Maar dit zaait verwarring bij verder onderhoud van het programma.



**Figure 1 Voorbeeld van een component** [**https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi64N2Y-PjmAhXRDOwKHRELAYUQjB16BAgBEAM&url=https%3A%2F%2Fblog.hackages.io%2Fangular-component-interaction-with-input-output-and-eventemitter-72526422b95c&psig=AOvVaw2kwxKMxP7rwgy**](https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi64N2Y-PjmAhXRDOwKHRELAYUQjB16BAgBEAM&url=https%3A%2F%2Fblog.hackages.io%2Fangular-component-interaction-with-input-output-and-eventemitter-72526422b95c&psig=AOvVaw2kwxKMxP7rwgy)

* **Selector**: de naam van de component dat kan gebruit worden in andere componenten.
* **TemplateUrl**: het pad naar het HTML bestand.
* **styleUrl**: het pad naar het CSS bestand



**Figure 2 Voorbeeld html code in .ts bestand** [**https://i.imgur.com/bbChQww.png**](https://i.imgur.com/bbChQww.png)

Hier zie je dus dat het ook mogelijk is om html te schrijven in de template selector. De app-child die daar dan aangeroepen wordt zal ook ingeladen worden.

# functionele vereisten

## use case diagram

# overige (niet-)technische vereisten

## kostprijs

# overzicht van de architectuur

# interactiediagrammen

# dataschema’s

# beschrijving van deployment mogelijkheden, installatie

# procedure

# referenties