Afstudeerverslag Noah Pauw

Noah Pauw, 424595, HBO-ICT

Saxion Hogeschool te Deventer

Betabit – Apeldoorn

5 september 2022 – 10 februari 2023

Voorwoord

Tijdens de afronding van mijn opleiding tot software engineer heb ik een afstudeerstage gedaan bij Betabit in Apeldoorn. Ik heb voor hen gewerkt aan een applicatie om hun tweewekelijkse Betatalks podcast in te plannen. Deze applicatie is er gekomen om de planning van de podcasts eenvoudiger te maken en de communicatie tussen de gasten en de hosts op één plek te kunnen uitvoeren.

Ik wil graag mijn bedrijfsbegeleider Dinand ten Hooven bedanken voor zijn passie, hulp en begeleiding tijdens mijn stage.

Ook wil ik graag Luuk ten Böhmer bedanken voor alle hulp tijdens de stage en een gezellige collega te zijn geweest.

Verder bedank ik ook graag Daan Kulman voor het ontwikkelen van de API, het bouwen van de backend, mij altijd een lift willen geven naar Betabit en voor een gezellige en leerzame afstudeerstage.

Ook bedank ik Betabit, Luc en Olav Gimbrère voor de afstudeerplek die ze mij hebben aangeboden. Ik bedank hen voor de werklaptop die ze mij aangeboden hebben, de expertise van hun collega's waar ik gebruik van heb mogen maken en de fantastische en uitgebreide introductieperiode.

Tot slot wil ik ook graag Kevin Wilmink bedanken voor zijn begeleiding, feedback op mijn documenten en zijn steun vanuit school.

Samenvatting

Voor het laatste gedeelte van mijn opleiding heb ik een afstudeerstage gedaan bij het softwarebedrijf Betabit in Apeldoorn van 1 september 2022 tot en met 10 februari 2023. Ik heb voor Oscar van Tol en Rick van den Bosch (de hosts van de Betatalks podcast) een applicatie gebouwd waarmee zij gemakkelijk de tweewekelijkse Betatalks podcast kunnen inplannen. Oscar en Rick zijn twee Microsoft certified softwareontwikkelaars bij Betabit. Tijdens deze podcast wordt gepraat over ontwikkelingen in de softwarewereld, zoals het .NET framework, Microsoft Azure ontwikkelingen en veel meer.

Oscar en Rick voeren de voorbereiding van een podcast momenteel uit via een simpel Word document. De communicatie tussen de presentatoren en de gasten verliep hierdoor niet efficiënt waardoor beide partijen vaak niet van elkaar wisten wat er allemaal besproken zou gaan worden.

Over Betabit

Betabit is een softwarebedrijf afkomstig uit Rotterdam. Betabit is in 2002 opgericht door de broers Olav en Luc Gimbrère en telt inmiddels 7 kantoren door heel Nederland. Met de hulp van ruim 150 werknemers ben ik een leuke en vooral leerzame tijd tegemoet gegaan. Betabit specialiseert zich sinds 2009 volop in het ontwikkelen voor het Microsoft Azure platform. Dit is het cloudplatform van Microsoft zelf. Betabit is officieel Gold Partner van Microsoft *(Betabit, 2020)* en maakt daarom veel gebruik van Microsofts diensten.

Azure heeft veel te bieden en veel (ook niet in software gespecialiseerde) bedrijven vinden het interessant om hun applicaties op dit cloudplatform te laten draaien. Daar heeft Betabit consultants voor die bij de klant aan de slag gaan om hen te helpen bij de transitie. Betabit heeft werknemers rondlopen bij veel verschillende bedrijven, zoals a.s.r. verzekeringen, Interpolis, RGF Staffing en nog veel meer.

Het probleem

De hosts van de podcast maken momenteel gebruik van een Word document waarin de podcasts tot in detail uitgewerkt worden. Ze zijn op zoek geweest naar een applicatie die dit proces voor hen kon vereenvoudigen. Het leek de hosts een interessante en uitdagende afstudeeropdracht om zo een applicatie te ontwikkelen.

De opdracht

Betabit was zoals eerder vermeld op zoek naar een applicatie waarmee ze hun tweewekelijkse podcast kunnen inplannen. Omdat er misschien nog wat gaat veranderen of toegevoegd gaat worden aan de applicatie na mijn afstudeertraject, is Betabit ook benieuwd in welk framework de app het makkelijkst gebouwd kan worden. Ook daar heb ik een onderzoek naar gedaan.

Tijdens mijn stage heb ik een aantal prototypes gebouwd in verschillende frameworks. Ik heb de applicatie in React, Angular en Blazor gebouwd. Ik heb samen met de product owner voorafgaand aan de ontwikkeling van de prototypes requirements en wensen opgesteld voor de app. De product owner heeft het over het algemeen erg druk gehad tijdens de stage, waardoor we na overleg met hem de rol van product owner hebben gedeeld met mijn bedrijfsbegeleider en de regiomanager van Apeldoorn.

De product owner wilde aan het begin van het project ook graag een live-notities omgeving voor de applicatie. Zo konden de hosts, regie en de gasten van de podcast gezamenlijk notities achterlaten over de podcast. Hierin kunnen bijvoorbeeld onderwerpen of vragen in geplaatst worden. Omdat de hosts en gasten vaak op verschillende locaties zaten, leek het hen makkelijk om de notities op deze live-omgeving uit te delen.

Tot slot willen Oscar en Rick ook graag dat de applicatie gasten van de podcast in staat stelt om hun eigen geluidsopname van de podcast te uploaden in de applicatie. Zo beschikken zij later over de hoogst mogelijke geluidskwaliteit van beide kanten.

Na het veldonderzoek te hebben afgerond en een drietal prototypes te hebben gebouwd, is gebleken dat Angular de beste optie is voor Betabit om de applicatie verder mee te gaan bouwen. Dit is gebaseerd op de prioriteiten van de software developers van Betabit.

De backend is gebouwd in het .NET framework van Microsoft door een andere stagiair. Ik heb nauw met hem samengewerkt om de frontend op een goede manier te kunnen verbinden met de backend. Ook heb ik met behulp van Microsoft Azure Functions een verbinding gelegd tussen een serverless web-sockets dienst van Microsoft om gebruikers toegang te geven tot een live-omgeving. Op deze omgeving kunnen zij realtime notities met elkaar delen.

Afbakening

Ik houd mij alleen bezig met de ontwikkeling van de prototypes in de **frontend**. Ik bouw het Podcast Portaal in React, Angular en Blazor. Deze drie frameworks zijn gekozen door Betabit omdat hier al veel ervaring en interesse in is. Hierin implementeer ik een vorm van client-side authenticatie zodat Oscar van Tol en Rick van den Bosch kunnen inloggen met hun Betabit account. Verder ontwikkel ik ook een applicatie waarmee de hosts, gasten en regie met elkaar kunnen communiceren via een verbinding tussen de frontend en de Azure Web PubSub dienst van Microsoft.

Ik overleg samen met de backend-ontwikkelaar over de structuur en opbouw van de backend. Zo hebben wij afgesproken welke velden er voor gasten, notities en podcasts opgeslagen werden en hoe deze velden genoemd zouden gaan worden. Verder maken we afspraken over welke gegevens het Podcast Portaal moet kunnen bereiken en welke endpoints er nodig zijn in de API om bij deze gegevens te kunnen.

Ik houd mij *niet* bezig met de ontwikkeling van de backend. Ik schrijf geen code of tests voor de API. Ik stel verder ook niet de database in waar de frontend-applicatie gegevens mee uit kan wisselen. Wél stel ik de opslag voor bestanden die de gasten moeten kunnen uploaden in en de omgeving en verbinding met de Live Notities.

Werkwijzen

Tijdens de stage heb ik een aantal verschillende soorten onderzoek gedaan. Zo heb ik een paar veldonderzoeken gedaan om de achtergrond van de drie web-frameworks te onderzoeken. Om dit goed te doen heb ik een aantal punten opgesteld waar ik de frameworks op heb beoordeeld. Zo krijgen zowel ik als de volgende ontwikkelaars een objectief beeld van ieder framework.

Daarna ben ik van start gegaan met de ontwikkeling van een drietal prototypes. Deze prototypes zijn qua uiterlijk identiek aan elkaar. Wel moesten ze alle drie dezelfde functionaliteit kunnen bieden. Voor de beoordeling van deze prototypes heb ik wederom een aantal criteria opgesteld. Zo heb ik bijvoorbeeld gekeken naar hoeveel werk het is om podcasts uit een database te halen en deze weer te geven in het portaal.

Voor de ontwikkeling van deze prototypes heb ik gewerkt met een aangepaste vorm van de scrum-methode. Omdat ik alleen aan het project heb gewerkt waren alle ontwikkeltaken voor mijzelf bestemd. Ik heb gebruik gemaakt van Azure DevOps om een scrum-bord aan te maken en bij te houden. Op deze manier kon ik goed inzien of ik nog op schema lag.

Projectgrenzen

Tijdens mijn stage ontwikkel ik de frontend van het Podcast Portaal. Ik houd mij niet bezig met de ontwikkeling van de backend. Wél overleg ik met de ontwikkelaar van de backend over welke gegevens er verzonden moeten worden tussen de front en- backend.

Verder ontwikkel ik ook de WebSocket oplossing om op een liveomgeving meerdere clients notities met elkaar te laten uitwisselen. De clients kunnen verbinding maken via de frontend met een serverless Publish-Subscribe dienst van Microsoft om met elkaar te kunnen communiceren. Gebruikers kunnen via de applicatie verbinding maken met deze service en zo notities met elkaar uitwisselen. Ik houd mij niet bezig met de opbouw van de Web PubSub dienst van Microsoft in de backend. Dit wordt gedaan door de backend developer.

Eindresultaat

Aan het eind van het project heb ik een volledig uitgebouwde applicatie geleverd waarin Betabit hun podcast gemakkelijk kan inplannen met de gast en de regisseur. De applicatie is uiteindelijk volledig uitgebouwd met het Angular framework. Verder lever ik ook de drie prototypes op met daarin een advies en- beoordelingsrapport op.

Verder hebben ze van mij ook een algemeen adviesrapport over de toekomst van de applicatie ontvangen. Hierin staan onder andere de onderdelen die niet helemaal uitgewerkt zijn of juist delen die interessant zouden kunnen zijn voor de toekomst. De applicaties worden natuurlijk ook geleverd met tests en de bijbehorende documentatie.

Inhoud

1. Inleiding

In dit afstudeerverslag neem ik je mee in mijn afstudeerstage bij Betabit in Apeldoorn. Ik heb een applicatie ontwikkeld bij Betabit van 1 september 2022 t/m 10 februari 2023. Met deze applicatie kunnen Oscar van Tol en Rick van den Bosch, de hosts van de Betatalks podcast, gemakkelijk de tweewekelijkse Betatalks podcast inplannen. Ik heb een aantal prototypes gebouwd in frameworks waar Betabit al mee werkt om applicaties te bouwen. Dit waren React, Angular en Blazor.

De criteria voor de beoordeling van deze frameworks zijn opgesteld op basis van de requirements van de product owner en de prioriteiten van de softwareontwikkelaars van Betabit.

1.1 Betabit

Betabit is opgericht op 1 oktober 2002 door Luc en Olav Gimbrère. Het is een softwareconsultancybedrijf met vestigingen in Rotterdam, Amsterdam, Den Haag, Utrecht, Eindhoven, Apeldoorn en Belgrado in Servië.

Betabit is volledig gespecialiseerd in het ontwikkelen, adviseren en deployen van applicaties op het Microsoft Azure cloudplatform. Betabit is een officieel Gold Partner van Microsoft en maakt bijna exclusief gebruik van de Microsoft toolstack.

Bij Betabit worden werknemers opgeleid met het Azure Talent Program (ATP) tot volledige Azure developers. Na dit traject worden zij uitbesteed aan bedrijven die geïnteresseerd zijn in Azure, .NET of andere Microsoft diensten. Zij treden dan in dienst als consultant bij bedrijven zoals a.s.r. verzekeringen, RGF Staffing, Interpolis, de Rabobank en nog veel meer.

Naast het ontwikkelen van software voor bedrijven biedt Betabit ook opleidingen aan zoals het Azure Talent Programma (ATP) aan developers binnen en buiten Betabit. Softwareontwikkelaars van verschillende bedrijven kunnen bij Betabit het ATP volgen om Azure beter te begrijpen en hiermee aan de slag te gaan. Hiervoor heeft Betabit hun eigen e-learnings genaamd EasyGenerators.

Betabit heeft ook het BetaKids programma waar kinderen en jongeren kunnen leren programmeren. Verder is er ook de tweewekelijkse podcast genaamd BetaTalks waar ik een applicatie voor heb gebouwd. Tijdens deze podcast wordt er gepraat met verschillende softwareontwikkelaars over interessante onderwerpen in de software-wereld, zoals .NET, Azure en nog veel meer.

Introductieperiode

Iedere consultant of stagiair die aan de slag gaat bij Betabit krijgt eerst een introductieperiode van twee weken. Tijdens deze periode bezoeken zij alle kantoren in Nederland. Zo heb ik ook meegedaan met dit programma waarbij ik de vestigingen in Rotterdam en Utrecht heb bezocht. Ik heb van Betabit tijdens de introductie een werklaptop gekregen, een account en wat leuke extra's zoals een Bambook (Bambook, sd).

Verder heb ik tijdens de introductieperiode met beide oprichters van Betabit gesproken over de visie van Betabit en hoe Betabit precies in elkaar zit. De introductie heb ik als erg leerzaam, leuk en ook belangrijk ervaren. Het was erg interessant om precies te zien waar Betabit voor staat en welke cultuur er heerst. Ik stel het zeer op prijs dat de oprichters na 20 jaar nog steeds de tijd nemen om iedere nieuwe werknemer even te ontmoeten.

2. De afstudeeropdracht

Tijdens mijn afstudeerstage heb ik een applicatie gebouwd voor de tweewekelijkse Betatalks podcast. Dit is een podcast van Betabit die op Spotify, Google Podcasts, Apple Podcasts en YouTube terug te vinden is. De podcast wordt georganiseerd door Oscar van Tol en Rick van den Bosch.

2.1 Het probleem

Oscar en Rick liepen tegen een probleem aan tijdens de voorbereiding van de podcast. De voorbereiding werd gedaan in een Word document voor het begin van de podcast. De gasten wisten vaak niet welke onderdelen er allemaal besproken zouden gaan worden.

De communicatie verliep ietwat stroef en Betabit was op zoek naar een oplossing. Het leek de hosts van de podcast een interessante afstudeeropdracht om een applicatie te bouwen waarmee zij de podcasts makkelijker kunnen inplannen.

2.2 De applicatie

De Betatalks hosts wilden graag een applicatie waarin zij gemakkelijk podcasts konden inplannen. Zij moesten hier podcasts kunnen aanmaken, ophalen, wijzigen en annuleren. Ze nodigen ook graag gasten uit om te praten over een later bepaald onderwerp. Hiervoor moest er bij het aanmaken van de podcast automatisch een e-mail verzonden worden naar de gast. In deze e-mail zit ook een deeplink waarmee zij tijdelijk toegang krijgen tot het portaal. Alleen gasten, hosts en de regie van de podcasts krijgen toegang tot de app.

Verder is er ook een omgeving voor de gasten om hun eigen gegevens achter te laten, zoals hun adres voor een goodiebag en een stukje over henzelf voor de hosts. De gasten konden ook hun eigen opname opsturen via het portaal naar de hosts zodat zij de beste audiokwaliteit konden gebruiken.

Ook is er een notitieomgeving voor de hosts en regisseurs om met elkaar te communiceren voorafgaand aan een podcast. De gasten kunnen hier ook gebruik van maken als zij met de hosts willen communiceren. De notities moesten ook automatisch vormgegeven kunnen worden in markdown.

2.3 Doelstelling

Aan het einde van mijn afstudeerstage lever ik een aantal prototypes van de podcastplan-applicatie op. De applicatie zal worden gehost op het Microsoft Azure platform. Betabit ontvangt van mij een objectief rapport over de gebruikte frameworks/libraries en welk framework het beste bij de gestelde eisen van de applicatie past.

Het doel van het onderzoek is om een beeld te schetsen van de mogelijkheden dat ieder framework te bieden heeft én of deze van toepassing zijn op de eisen van Betabit. Hier vallen het bureauonderzoek en de prototypes onder.

Het doel is dat Betabit uiteindelijk een aantal werkende prototypes ontvangt waarmee de hosts de Betatalks podcast kunnen inplannen. Verder ontvangen zij daarbij een adviesrapport over de toekomst van de applicatie.

2.4 Afbakening

Ik werk tijdens mijn afstudeerstage *alleen* aan de frontend van de applicatie. Ik voer een onderzoek uit naar de drie door Betabit gekozen frameworks en onderzoek welk van deze frameworks het beste gebruikt kan worden voor de ontwikkeling van de applicatie. Ik houd mij niet bezig met de ontwikkeling van de backend. Ik werk echter wel nauw samen met de ontwikkelaar van de backend zodat ik de frontend goed kan afstemmen op de structuur van de backend.

3. Onderzoek naar frameworks

Aan het einde van mijn afstudeerstage lever ik de frontend van een applicatie op waar Betabit hun podcasts mee kan inplannen. In welk framework dit gebeurt gaat blijken uit de prototypes die ik ga bouwen. De applicatie zal verbinding maken met een API die gebouwd wordt door een andere stagiair binnen Betabit waar ik nauw mee samenwerk.

Omdat Betabit de applicatie waarschijnlijk wil (laten) uitbreiden en de ontwikkelaars van Betabit met verschillende frontend frameworks werken, doe ik een onderzoek naar welk van deze frameworks het beste aansluit op de wensen van Betabit.

De ontwikkelaars van Betabit werken vooral met de volgende frontend frameworks:

- **React** (JavaScript en TypeScript)
- **Angular** (JavaScript en TypeScript)
- **Vue** (JavaScript en TypeScript)
- **Blazor** (C#)

Er is binnen Betabit al veel ervaring met React, Vue en Angular. Er wordt minder gebruik gemaakt van Blazor. Er is echter wel veel interesse in Blazor omdat er ook veel ervaring is met .NET en C#. Daarom leek het Betabit een goed idee om één van deze frameworks te gebruiken voor de ontwikkeling van de applicatie.

3.1 Schrappen van het Vue prototype

Aan het begin van het project was het de bedoeling om *vier* prototypes te bouwen en één van deze applicaties uit te bouwen tot een volledige applicatie. Er is uiteindelijk gekozen om het Vue framework te laten vallen omdat er anders te weinig tijd zou zijn om voldoende diepgang te krijgen in ieder framework. Er is gekozen om de twee populairste JavaScript/TypeScript frameworks/libraries en een experimenteel en relatief nieuw C# framework te onderzoeken.

3.2 Hoofdvraag

Hoe bouw ik op basis van een frontend-framework dat goed past bij de wensen van Betabit en de requirements van de product owner een podcastportaal waarmee Betabit het voorbereiden van een podcast samen met de gasten op een gemakkelijke manier kan uitvoeren?

3.3 Deelvragen

1. Welke van de drie door Betabit uitgekozen frameworks past het beste bij de voorkeuren van de ontwikkelaars van Betabit voor de ontwikkeling van het Podcast Portaal?
    - Kwantitatieve survey: Wat vinden de ontwikkelaars van Betabit belangrijk bij het uitkiezen van een frontend framework? Ik ga een lijst met mogelijke criteria opstellen en rondvragen bij andere softwareontwikkelaars bij Betabit in welke criteria zij geïnteresseerd zijn.
    - Bureauonderzoek: Ik voer een onderzoek uit naar bestaande informatie over deze frontend-frameworks op basis van de opgestelde criteria.
    - Met deze onderzoeken kan ik concluderen welk framework het beste past bij de prioriteiten van ontwikkelaars bij Betabit. Dit zal worden beoordeeld op basis van een aantal criteria die later in dit hoofdstuk worden toegelicht.
2. Welke programmeertaal kan het beste gebruikt worden bij de JS/TS frameworks voor de applicatie?
    - Bureauonderzoek: Bij de eerste twee frameworks kan zowel JavaScript als TypeScript gebruikt worden. Bij dit onderzoek zoek ik de verschillen tussen de twee talen.
        - Ook onderzoek ik welke taal het beste gebruikt kan worden bij de ontwikkeling van het portaal.

- **Bureauonderzoek:** Wat zijn de voor- en nadelen van **JavaScript**?
- **Bureauonderzoek:** Wat zijn de voor- en nadelen van **TypeScript**?
- Dit onderzoek wordt uitgevoerd omdat het mogelijk is om voor React en Angular te kiezen uit TypeScript of JavaScript. Er wordt een veldonderzoek gedaan naar de verschillen tussen deze twee programmeertalen.

3. Hoe maak je een goede gebruikerservaring voor het podcastportaal op basis van algemene UX-principes?
    - **Bureauonderzoek:** Welke algemene UX-principes zijn er voor een goede gebruikerservaring?
    - **Bureauonderzoek**: Welke UX wetten/richtlijnen zijn er?
    - Dit onderzoek wordt uitgevoerd omdat mijn opdracht volledig gericht is op het ontwikkelen van de frontend van de applicatie. De applicatie moet makkelijk en efficiënt te gebruiken zijn. Daarom onderzoek ik welke ideologieën ik kan toepassen om de applicatie aan de hand van algemeen bekende ontwerpprincipes en denkwijzen te ontwikkelen.
4. Welk framework past het beste bij de ontwikkeling van de applicatie op basis van de requirements van de product owner?
    - **Prototype onderzoek:** Tijdens dit onderzoek worden drie prototypes gebouwd op basis van de requirements van de product owner. Er wordt voor ieder requirement gekeken wat de aanpak is om het requirement te implementeren.
    - Er wordt een aantal criteria opgesteld om de prototypes tijdens de ontwikkeling van de applicatie beoordelen op functionaliteit en moeilijkheidsgraad.
    - Er wordt objectief gekeken naar wat ieder framework biedt om bepaalde functionaliteit in te bouwen.
    - Verder vul ik dit aan met mijn eigen ervaring over de implementatie hiervan.

4. Deelvraag 1: Onderzoek naar frameworks

Deelvraag 1: Welke van de drie door Betabit uitgekozen frameworks past het beste bij de voorkeuren van de ontwikkelaars van Betabit voor de ontwikkeling van het Podcast Portaal?

Tijdens de afstudeerstage heb ik een deskonderzoek gedaan naar de verschillen tussen de drie voorgenoemde frameworks. Er is gekozen uit React, Angular en Blazor omdat er binnen Betabit al veel ervaring is met React en Angular. Verder zijn sommige ontwikkelaars bij Betabit ook benieuwd naar Blazor omdat er veel ervaring is met C# en .NET en Blazor hier gebruik van maakt. In de eerste instantie wilden we Vue als vierde framework onderzoeken, maar doordat er door tijdgebrek een tekort aan diepgang zou ontstaan is Vue uiteindelijk geschrapt.

Volledig onderzoek: Onderzoek naar frameworks

4.1 Onderzoeksopzet

Om een objectief beeld te krijgen van ieder framework is er een aantal criteria opgesteld. De criteria waarop ieder framework onderzocht gaat worden zijn als volgt:

- Installatie van het framework
- Officiële documentatie
- Community
- Performance
- Learning curve
- Toekomstplannen
- Tests

Voor het onderzoek wilde ik ook weten wat de softwareontwikkelaars van Betabit zelf belangrijk vonden voor het kiezen van een framework. Hiervoor heb ik een **enquête** opgesteld en deze gedeeld via Yammer. Yammer is een sociale mediaplatform van Microsoft waar werknemers binnen een bedrijf met elkaar kunnen communiceren.

De enquête is simpel. Alle punten worden benoemd en er wordt gevraagd aan andere softwareontwikkelaars in welke volgorde zij deze punten zouden zetten.

Op de enquête heb ik uiteindelijk 9 verschillende antwoorden gekregen. Met de antwoorden van andere werknemers kon er een gewicht aan ieder criteriumpunt gehangen worden. Zo werd van alle antwoorden de gemiddelde positie in de rij berekend en geprioriteerd.

Notitie: in verband met de Algemene Verordening Gegevensbescherming (AVG) worden de namen van de ontwikkelaars bij Betabit niet getoond.

Werknemer	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Softwareontwikkelaar 1	5	2	3	7	1	4	6
Softwareontwikkelaar 2	4	5	6	2	1	7	3
Softwareontwikkelaar 3	2	7	5	6	4	1	3
Softwareontwikkelaar 4	2	4	5	7	6	1	3
Softwareontwikkelaar 5	2	3	5	7	4	6	1
Softwareontwikkelaar 6	2	7	3	6	5	1	4
Softwareontwikkelaar 7	5	2	7	4	3	6	1
Regiomanager van Apeldoorn en Utrecht	3	2	5	4	6	7	1
Product owner	7	2	3	1	6	4	5

Tabel 1: Uitslag enquête over prioriteit van criteriumpunten

Met de criteriumpunten geprioriteerd is er een score toegekend aan ieder punt. In totaal zijn er 7 criteriumpunten die onderzocht gaan worden. In totaal zijn er per framework 100 punten te halen. Het framework met de meeste punten komt het beste uit de test.

Criteriumpunten

1.	Officiële documentatie	(positie 2.1 gemiddeld)	30 punten
2.	Learning curve	(positie 3.1 gemiddeld)	20 punten
3.	Tests	(positie 3.3 gemiddeld)	15 punten
4.	Community	(positie 4.2 gemiddeld)	12 punten
5.	Performance	(positie 4.8 gemiddeld)	10 punten
6.	Toekomstplannen	(positie 4.9 gemiddeld)	8 punten
7.	Installatie van het framework	(positie 5.9 gemiddeld)	5 punten

4.2 Het onderzoek

Tijdens het onderzoek ben ik op zoek gegaan naar informatie over ieder criteriumpunt. Ik heb gekeken naar bronnen die zijn geschreven door de ontwikkelaars van ieder framework, maar ook naar artikelen, vraagstukken, bestaande onderzoeken en interviews van andere ontwikkelaars en geïnteresseerden.

De criteria zijn algemeen opgesteld om te kijken welke onderdelen ontwikkelaars bij Betabit interessant zouden vinden om te onderzoeken. Hierbij is gekeken naar de gebruikers van de frameworks, de huidige staat en de toekomst van het framework.

Ieder onderdeel wordt grondig onderzocht. Voor onderdelen die te maken hebben met het gebruik van het framework wordt op een simpele omgeving een aantal handelingen uitgevoerd (bijvoorbeeld voor de installatie van ieder framework).

4.2.1 Puntentoekenning

Voor het toekennen van een score aan een criteriumpunt wordt gekeken naar de hoeveelheid beschikbare informatie, aantal opties en meningen van gebruikers uit verschillende bronnen ten opzichte van de andere frameworks. Een framework kan een bepaald punt net iets beter doen dan een ander framework.

4.3 Conclusie per framework

De drie frameworks zijn beoordeeld op de eerder genoemde criteriumpunten. Per criterium wordt er een aantal punten uitgegeven. Dit aantal is gebaseerd op wat de ontwikkelaars van Betabit belangrijk of juist minder belangrijk vinden. Ieder framework heeft sterke en zwakke kanten. Het is daarom voor dit onderzoek interessant om te kijken wat specifiek voor de wensen van Betabit handig is voor de ontwikkeling van het Podcast Portaal.

Criterium	React	Angular	Blazor	Gewicht
Beschikbare officiële documentatie	5,4	9,8	8,0	30
Learning curve	7,0	6,3	7,8	20
Tests	9,6	9,8	9,5	15
Community	8,2	8,5	3	12
Performance	9,5	8,5	6,2	10
Toekomstplannen	7,6	9,0	6,7	8
Installatie van het framework	9,4	8,2	8,8	5
Totaal	56,7	60,1	50	Max: 70

Tabel 2: Alle criteria van het veldonderzoek voor alle drie frameworks met ontvangen score per punt. Deze punten zijn ongewogen.

Omdat de punten verschillende gewichten hebben, wordt het totaal als volgt berekend:

```
T = T + score / 10 * gewicht
```

Waarbij T het totaal aantal punten is dat het framework heeft behaald.

Uit mijn onderzoek is gebleken dat Angular met 88,2 punten het beste aansluit op de voorkeuren van de ontwikkelaars van Betabit.

1.	Angular	86,7 punten
2.	React	74,7 punten
3.	Blazor	72,7 punten

De overige frameworks hebben ook goed gescoord en kunnen absoluut ook gebruikt worden voor de ontwikkeling van het Portaal.

4.4 Conclusie

Angular lijkt het beste te passen bij de prioriteiten van een aantal softwareontwikkelaars bij Betabit. Hiervoor is gekeken naar verschillende criteria die zijn geprioriteerd op basis van reacties op de enquête. Op deze manier is er een lijst gevormd met punten waar ieder framework op is beoordeeld. Op basis van deze criteria en prioriteiten vanuit Betabit is Angular als beste geëindigd.

Het is wellicht interessant dat React ruim zes punten meer heeft dan Blazor bij de ongewogen punten en dit verschil slechts 2 punten wordt na het toevoegen van gewichten aan de scores. Dit zou kunnen betekenen dat Blazor bepaalde onderdelen die Betabit belangrijk vindt beter doet dan React.

Angular heeft een aantal zeer sterke punten die goed aansluiten op de prioriteiten van Betabit. Bijvoorbeeld werd de documentatie van een framework als zeer belangrijk beschouwd over het algemeen. Angular biedt zeer uitgebreide documentatie om gebruikers goed op weg te helpen.

Het deskonderzoek is een onderdeel van een groot onderzoek waarbij naast het theoretische onderzoek naar de frameworks er ook drie prototypes gebouwd worden. Het Podcast Portaal wordt gebouwd in de drie voorgenoemde frameworks om te onderzoeken welk framework het beste gebruikt kan worden om het Podcast Portaal mee door te ontwikkelen.

Voor de volledige onderzoeksopzet en conclusies, zie Onderzoek naar frameworks.

5. Deelvraag 2: Onderzoek JavaScript vs. TypeScript

Deelvraag 2: Welke programmeertaal kan het beste gebruikt worden bij de React library en het Angular framework?

De eerste twee frameworks/libraries, React en Angular, kunnen gebruikt worden met zowel JavaScript als TypeScript. In dit onderzoek zoek ik de verschillen tussen de twee programmeertalen. Ook onderzoek ik welke van de twee talen het beste gebruikt kan worden voor de ontwikkeling van de applicatie.

5.1 Onderzoeksopzet

Tijdens dit onderzoek voer ik een bureauonderzoek uit naar bestaande informatie van online bronnen. Ik zoek uit wat de verschillen zijn tussen JavaScript en TypeScript. Vervolgens onderzoek ik de populariteit van beide talen. Tot slot onderzoek ik ook welke taal het beste gebruikt kan worden met React en Angular.

5.2 Verschillen tussen JavaScript en TypeScript

JavaScript is een programmeertaal die in 1995 door Brendan Eich is ontwikkeld voor Netscape (Mozilla, 2022). Het is een zeer populaire programmeertaal voor websites die meer dan statische inhoud willen vertonen, zoals video's, inhoud van externe bronnen en meer.

In 2012 is TypeScript uitgegeven na 2 jaar in ontwikkeling te zijn geweest bij ***Microsoft*** (Turner, 2014). TypeScript is een programmeertaal die is ontwikkeld in een poging de tekortkomingen van JavaScript aan te vullen (Hejlsberg, 2012).

Andere talen met dezelfde doelstelling als TypeScript (het verbeteren en vervangen van JavaScript) zijn PureScript, Reason, Elm en ClojureScript. TypeScript introduceert grote verbeteringen in JavaScript zoals type safety, abstract classes, overloading, generics en meer. (Agrawal, 2022) TypeScript is een wrapper voor JavaScript, wat inhoudt dat TypeScript hetzelfde functioneert als JavaScript met extra functies die de valkuilen van JavaScript proberen op te vullen.

5.2.1 Functionele verschillen tussen JavaScript en TypeScript

Functie	JavaScript	TypeScript
Bruikbaar voor frontend en backend	Ja	Ja
Type safety	Nee	Ja
Static typing	Nee	Ja
Classes	Ja	Ja
Class object georiënteerd (Megida, 2020)	Nee	Ja
Interface ondersteuning (ParTech Media, 2021)	Nee	Ja

Tabel 3: Functionele verschillen tussen JavaScript en TypeScript

5.2.2 Directe verschillen

	JavaScript	TypeScript
Ontwikkelaar	Door Brendan Eich (Netscape) in 1995	Door Microsoft in 2012
Type safety	Niet mogelijk om types te definiëren. (Mozilla, n.d.)	Verplicht om types te definiëren aan variabelen.
Populariteit *(StackOverflow)*	JavaScript was de meest gebruikte programmeertaal in 2022 volgens de StackOverflow top 20 survey. (Ramel, 2022)	TypeScript is de op vier na meest populaire programmeertaal. TypeScript is wel een van de 12e plaats in 2018 naar de 5e plaats gestegen in 2022. (Ramel, 2022)
Compilatie	Geen compilatie nodig. Directe interpretatie door browsers.	TypeScript moet eerst gecompileerd worden. (Dubey, 2022)

Learning curve	Dynamic typing maakt JavaScript een makkelijke taal omdat er weinig restricties zijn *(Kozlov, 2021)*	Vereist kennis van scripting, static typing en JavaScript.
Bestandstypen	.js	.ts en .tsx
Dynamic of static typing	Dynamic typing *(Kozlov, 2021)*.	Static typing

Tabel 4: Verschillen op het oppervlak tussen JavaScript en TypeScript

5.3 Performance

TypeScript is een "statically typed" programmeertaal terwijl JavaScript een "dynamically typed" taal is *(Kozlov, 2021)*. Bij static typing wordt een data type meegegeven aan de compiler. Bij dynamic typing kan dit data type tijdens runtime nog veranderen. Doordat de compiler van tevoren weet wat voor data type in een variabele zit, kan de code wanneer deze eenmaal gecompileerd is sneller uitgevoerd worden dan bij dynamic typing *(Bhatnagar, 2018)*.

5.4 Aanbevolen taal voor JavaScript frameworks

TypeScript is een superset voor JavaScript, wat inhoudt dat elk TypeScript programma eigenlijk JavaScript gebruikt onder de motorkap. TypeScript is een verbeterde en modernere versie van JavaScript met extra toegevoegde functies en syntax om JavaScript robuuster en beter te maken (Raval, 2022).

Zowel React als Angular bieden ontwikkelaars de mogelijkheid om frontends te bouwen met zowel JavaScript als TypeScript. Er zijn echter verschillen bij het ontwikkelen van een gebruikersinterface met JavaScript of TypeScript.

- **Performance**: voor performance redenen wordt aanbevolen om TypeScript te gebruiken in plaats van JavaScript. TypeScript biedt namelijk static typing t.o.v. JavaScript, wat het voor de compiler makkelijker maakt om de applicatie te compileren (White, 2020).
- **Populariteit**: JavaScript is een enorm populaire programmeertaal voor websites wereldwijd. Ruim 98% van alle websites gebruikt JavaScript (Raval, Top 10 JavaScript usage statistics to watch out for in 2022, 2022). TypeScript is op dit moment minder populair dan JavaScript, mede doordat TypeScript 17 jaar na de eerste versie van JavaScript uitgegeven werd (Turner, 2014).

5.4.1 TypeScript of JavaScript voor React en Angular

Bij de ontwikkeling van een applicatie met het React of Angular framework wordt aanbevolen dit met TypeScript te doen. TypeScript biedt de voordelen van JavaScript met betere leesbaarheid, verbeterde ondersteuning voor object-georiënteerd programmeren en type safety. (Bhati, 2022)

5.5 Conclusie

Omdat het Podcast Portaal een grotere applicatie wordt waarbij gegevens opgehaald worden van een backend, deze in de frontend geüpdatet moet kunnen worden en doorontwikkeld moet kunnen worden door andere teams, wordt **TypeScript** gebruikt om de applicatie te bouwen.

Door TypeScript te gebruiken is de code ook makkelijker te begrijpen voor het volgende team. Door gebruik te maken van static typing, classes en interfaces is de code ook leesbaarder voor anderen.

TypeScript groeit ook in populariteit *(Ramel, 2022).* Omdat TypeScript essentieel een verbeterde versie van JavaScript is met alle functies die JavaScript te bieden heeft, is het een waardevolle investering om TypeScript te leren.

Het kan meer tijd kosten om een applicatie te bouwen met TypeScript dan met JavaScript. JavaScript is namelijk een dynamisch getypeerde programmeertaal terwijl TypeScript een statisch getypeerde programmeertaal is. JavaScript verplicht ontwikkelaars niet om expliciet data types mee te geven aan de compiler. Hierdoor is het mogelijk om sneller een werkend prototype te ontwikkelen met JavaScript dan met TypeScript, maar kan de prestaties van de compiler verslechteren doordat de data types tijdens runtime kunnen veranderen.

Voor onderhoud, doorontwikkeling en leesbaarheid is TypeScript absoluut de betere optie. Door static typing is het makkelijker voor zowel de compiler als voor andere ontwikkelaars om te kunnen zien welke classes er worden gebruikt in de code.

Kortom, voor een applicatie die door andere developers gebruikt of beoordeeld gaat worden is TypeScript verreweg de beter optie. Ook om performance redenen scoort TypeScript hoger dan JavaScript omdat TypeScript een statisch getypeerde programmeertaal is. Verder is TypeScript in een notendop een verbeterde versie van JavaScript met alle functionaliteit die JavaScript al te bieden had.

6. Deelvraag 3: User Experience

Deelvraag 3: Hoe maak je een goede gebruikerservaring voor het podcastportaal op basis van algemene UX-principes?

6.1 Onderzoeksopzet

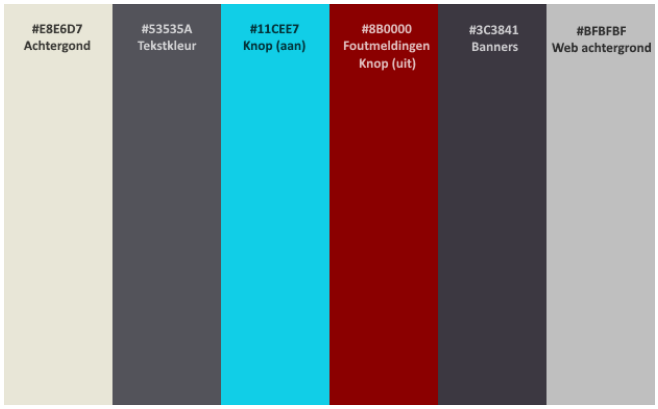
Voor het onderzoek naar de user experience van de applicatie voer ik een bureauonderzoek uit naar bekende UX principes. Hiervoor raadpleeg ik diverse online bronnen en onderzoek ik welke principes van toepassing zijn op het Podcast Portaal.

6.2 Inleiding

Het Podcast Portaal moet makkelijk in gebruik zijn. Ook heeft de product owner gevraagd om een unieke stijl te ontwikkelen voor de applicatie (zie requirement RQ20). Dit laatste wordt gedaan om de applicatie uiteindelijk open source uit te geven. Om het gebruik van de applicatie simpel, efficiënt en overzichtelijk te houden, maak ik gebruik van algemeen bekende UX-principes. Zo wordt het voor de gebruiker zo makkelijk mogelijk om gebruik te maken van de applicatie.

6.3 Unieke stijl

Om het Podcast Portaal een unieke stijl te geven hanteer ik een kleurenschema afkomstig van Coolors (Coolors, sd). Hier wordt automatisch een schema gegenereerd met passende kleuren. Dit kleurenschema is gecombineerd met een lichte invloed van de Betabit huisstijl om er toch nog een klein beetje branding aan toe te voegen.



Figuur 1: Kleurenschema van het Podcast Portaal

Het is uiteraard aan andere ontwikkelaars om de kleuren te veranderen. De kleuren staan met variabelen in de code aangeduid en kunnen gemakkelijk aangepast worden.

De applicatie maakt gebruik van het **Ubuntu** lettertype. Dit lettertype is open source en kan dus zonder licentie gebruikt worden (Adobe, sd). Ik heb overwogen om **Segoe UI** aan te raden aan de product owner omdat dit een lettertype is van Microsoft en Betabit officieel Gold Partner is van Microsoft. Echter is het verboden om Segoe UI te gebruiken buiten Microsoft producten op niet-Microsoft systemen (Basit et al, 2021). Omdat het Podcast Portaal een open source applicatie wordt, is gekozen voor een lettertype zonder licenties.

Afgeronde randen

Het Podcast Portaal maakt ook gebruik van afgeronde randen voor knoppen en sommige afbeeldingen. Het licht afronden van deze randen geeft een subtiele maar professionele draai aan het ontwerp. Verder kunnen de positieve aspecten van afgeronde knoppen psychologisch onderbouwd worden.

> "This is a classical conditioning principle where our brain is conditioned to think sharp objects can be harmful. A real-life example where sharp corners are considered harmful is when you're baby-proofing your house by using rounded fittings on sharp table corners." – (Subramaniyan, 2020)

Het menselijk brein interpreteert ronde hoeken als veilig en vriendelijk. Om deze stemming over te brengen in het ontwerp, zijn de knoppen in het Podcast Portaal lichtelijk afgerond.

Figuur 2: Een afgeronde en een niet afgeronde knop in het Podcast Portaal

6.4 UX wetten

Om de applicatie makkelijker te kunnen gebruiken, is een aantal UX "*best practices*" toegepast om het gebruik van de applicatie overzichtelijk te houden. Deze best practices zijn overgenomen van Laws of UX (lawsofux.com, sd). Er is gekozen om deze specifieke best practices te gebruiken omdat deze van toepassing waren op de applicatie.

- **Jakob's Law**: gebruikers besteden het merendeel van hun tijd op andere websites. Deze gebruikers vinden makkelijker hun weg als het Podcast Portaal op dezelfde manier werkt.
    - Jakob's wet is toegepast door te kijken naar wat alom bekende websites doen, zoals Twitter, Facebook, YouTube en Google. Zo maakt de applicatie gebruik van breadcrumbs zodat gebruikers zich kunnen oriënteren op de huidige pagina en gemakkelijk terug kunnen navigeren. Verder heeft de applicatie een uitvouwbaar menu met daarin een knop om direct terug naar de homepagina te navigeren.
    - Op alle populaire websites is het mogelijk om terug naar de homepagina te navigeren door te klikken op het logo van de website. Dit is bij het Podcast Portaal ook toegepast.
- **Miller's Law**: gebruikers kunnen gemiddeld rond de 7 (plus minus 2) onderdelen in hun werkgeheugen houden voordat zij het overzicht verliezen.
    - Dit is toegepast door maximaal vijf podcasts te tonen op de homepagina van het Podcast Portaal. Er is gekozen om maximaal vijf objecten te tonen omdat er ook al veel informatie in één podcast object kan staan.
- **Hick's Law**: de hoeveelheid benodigde tijd om beslissingen te maken neemt toe op basis van de hoeveelheid en complexiteit van keuzes.
    - Om het gebruik van het Podcast Portaal efficiënt te houden is er een beperkt aantal opties beschikbaar tijdens het gebruik. Zo zijn alle navigatieopties niet zichtbaar totdat de gebruiker het uitvouwbare menu opent. Verder zijn er te alle tijden maximaal 5 podcasts te zien op het homescherm voor de hosts. Dit zorgt ervoor dat zij niet ver hoeven te scrollen om de podcast te vinden die zij zoeken. Op deze manier wordt het overzicht beter bewaart.

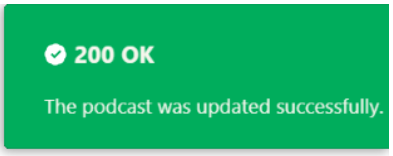
6.4.1 Toepassen UX-wetten in het ontwerp

Volgens Jakob's wet geven gebruikers de voorkeur aan websites die op dezelfde manier werken als andere websites die zij gebruiken. Dit is natuurlijk lastig te doen omdat iedere website anders is. Er zijn wel veel onderdelen die op veel websites voorkomen, zoals

- Hamburgermenu's
  - Een uitvouwbaar menu met een icoon dat lijkt op een hamburger. Dit icoon is op veel websites terug te vinden om een menu aan te duiden.
  - Er wordt ook gediscussieerd over de aanwezigheid van hamburgermenu's op niet-mobiele apparaten. Sommigen vinden dat het hamburgermenu de navigatie van een website juist saboteert. (Roselli, 2021)
  - Omdat er al snel veel podcasts, informatie en gastgegevens getoond worden op één pagina, is het een goede keuze om toch een hamburgermenu te implementeren. Het icoon is algemeen bekend en een gebruiker kan zich richten op zaken die niets te maken hebben met navigatie totdat zij willen navigeren.
- Breadcrumbs
  - Een vorm van navigatie die de structuur en hiërarchie van een webapplicatie aangeeft. Met breadcrumbs kunnen gebruikers makkelijker bijhouden waar zij zich bevinden op de website.
- Het F-patroon
  - Het meest voorkomende patroon waarin een gebruiker de inhoud van een webpagina bekijkt. Het patroon is aan het licht gekomen door een "eyetracking" studie uitgevoerd door NNGroup. (Babich, 2017)
  - Volgens het patroon leest een gebruiker een webpagina meestal als volgt:
    - Gebruikers lezen eerst horizontaal het dominantste deel van een blok inhoud
    - Vervolgens wordt er verticaal aan de linker kant van de pagina gezocht naar interessante punten in eerste zinnen van paragrafen. Een paragraaf die hen interessant lijkt wordt dan gelezen.
    - Vervolgens zoekt de gebruiker verticaal verder naar het volgende stuk inhoud.
    - In Fout! Verwijzingsbron niet gevonden. in de bijlagen is te zien hoe het Podcast P ortaal gebruik maakt van het F-patroon

6.5 Meldingen

Omdat het voor de gebruiker ook duidelijk moet zijn of een actie geslaagd is of niet (bijvoorbeeld het aanmaken van een podcast is niet gelukt omdat er een fout is opgetreden), worden er meldingen getoond rechtsboven in het scherm. Deze meldingen maken gebruik van een vast kleurenschema dat door de meeste websites wordt gebruikt. Deze kleuren geven in één oogopslag aan om wat voor type melding het gaat.



- Blauw: standaardmelding, geen fout of geslaagde actie
- Groen: melding van een geslaagde actie
- Geel: een waarschuwing
- Rood: een foutmelding

Zonder dat de gebruiker de melding hoeft te lezen wordt de aard van de melding iets duidelijker gemaakt. De kleur geeft direct aan om wat voor type melding gaat. Voor gebruikers met kleurenblindheid biedt het icoon voor de kop van de melding ook meer informatie.

6.5.1 Kleurenblindheid

Een algemene regel is dat kleur niet het enige medium moet zijn om informatie over te brengen. Om het voor gebruikers met kleurenblindheid iets duidelijker te maken wat de aard is van de melding, wordt er een klein icoon toegevoegd. In dit icoon wordt extra informatie gegeven over het type melding.

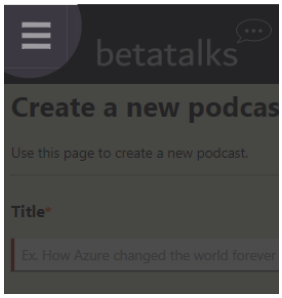
	**Algemene informatie**	(momenteel geen podcasts, verbonden met clouddienst)
	**Geslaagde actie**	(aanmaken podcast, opslaan notities, annuleren podcasts)

⚠	**Waarschuwing**	(verbinding met clouddienst is traag)
⊗	**Foutmelding**	(API kon niet bereikt worden, verbinding met clouddienst is weggevallen)

Figuur 3: Tabel met iconen die aangeven om wat voor type melding het gaat. Deze worden gebruikt om niet alleen afhankelijk te zijn van een kleurenschema voor meldingen

6.6 Navigatie

Om de gebruiker geen navigatie opties te hoeven tonen totdat deze wil navigeren, wordt er gebruik gemaakt van een uitvouwbaar `hamburgermenu`. Dit is een knop linksboven in het scherm (op sommige websites staat deze rechts) met daarin een hamburgericoon. Wanneer de gebruiker hier op klikt, worden er meerdere routes getoond waar de gebruiker naartoe kan navigeren.

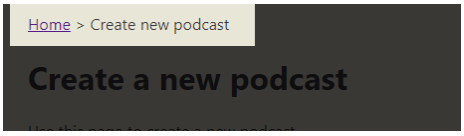


De gebruiker krijgt in dit menu alle pagina's te zien waar zij naartoe kunnen navigeren. Het menu is vervolgens ook makkelijk weer te sluiten met dezelfde knop.

Dit is gedaan omdat gebruikers snel het overzicht kunnen verliezen als zij met veel opties geconfronteerd worden (Hick's law). Verder wordt het op deze manier opgelost omdat veel gebruikers dit component al kennen van andere websites, zoals Twitter, YouTube en Facebook.

Breadcrumbs

Breadcrumbs is een bekend principe om overzichtelijke UI's te bouwen. Breadcrumbs worden bovenaan een webpagina getoond aan de gebruikers als een vorm van navigatie (Toonen, 2021). Deze breadcrumbs laten zien op welke pagina de gebruiker zich momenteel bevindt en waar deze pagina bij hoort. Gebruikers kunnen met deze breadcrumbs makkelijk navigeren naar bovenliggende pagina's.

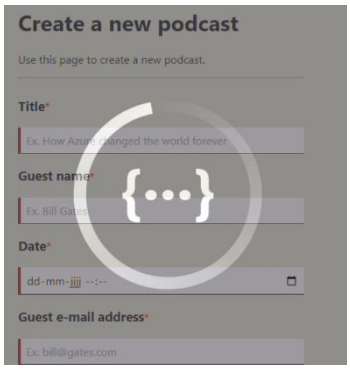


6.7 Laadbalken en spinners

Omdat de applicatie ook een aantal asynchrone taken gaat uitvoeren (HTTP-requests naar de backend en ontvangen en versturen van notities), is er een manier ontwikkeld om aan duidelijk te maken aan de gebruiker dat de applicatie een proces aan het uitvoeren is.

Er zijn hiervoor twee welbekende methoden voor: een laadbalk en een spinner. Een laadbalk is een horizontale balk met daarin een anders gekleurde balk die van links naar rechts beweegt. Wanneer de balk volledig is ingevuld, is het proces voltooid en verdwijnt de balk. Dit wordt vooral gebruikt voor langere processen om aan te geven hoe ver het proces uitgevoerd is. Om te concluderen of een proces lang genoeg duurt om gebruik te maken van een laadbalk, wordt de 4-seconden regel toegepast. (Smirk, 2019)

Een spinner is een draaiend laadicoon dat net als de laadbalk aangeeft dat er een proces wordt uitgevoerd. Het verschil met de laadbalk is dat de gebruiker aan een spinner niet kan zien hoe ver het proces is en hoelang het nog ongeveer duurt. Een spinner wordt meestal bij kortere processen gebruikt. (Movement, UX, 2016)



Laadbalk of spinner voor de applicatie

Omdat het Podcast Portaal geen complexe processen uitvoert en bijna alle processen binnen 4 seconden afgerond zijn, wordt er gebruik gemaakt van een spinner.

Deze wordt bijvoorbeeld getoond wanneer de client een HTTP-request verstuurt naar de backend. Ook is deze te zien wanneer een podcast opgehaald wordt, een notitie wordt opgeslagen en wanneer een gebruiker inlogt met een deeplink.

Figuur 4: Podcast portaal spinner die aangeeft dat er een proces uitgevoerd wordt. Voor Betabit wordt het Betabit logo getoond, maar zodra de applicatie open source wordt uitgegeven wordt deze verwijderd

6.8 Conclusie UX Podcast Portaal

Voor een goede user interface is het belangrijk om te kijken naar algemeen bekende UX principes. Gebruikers besteden het grootste deel van hun tijd op andere websites, dus verwachten zij dat het Podcast Portaal ongeveer op dezelfde manier werkt.

Het toepassen van een aantal UX best practices helpt bij het efficiënt laten verlopen van de gebruikersinteractie. Er is gekozen om specifiek deze best practices toe te passen omdat deze van toepassing waren op de applicatie.

- **Jakob's Law**: gebruikers besteden het merendeel van hun tijd op andere websites. Deze gebruikers vinden makkelijker hun weg als het Podcast Portaal op dezelfde manier werkt
- **Hick's Law**: de hoeveelheid benodigde tijd om beslissingen te maken neemt toe op basis van de hoeveelheid en complexiteit van keuzes
- **Miller's Law**: gebruikers kunnen rond de 7 (plus minus 2) onderdelen in hun werkgeheugen kunnen houden voordat zij het overzicht verliezen

Om de meldingen beter voor gebruikers met kleurenblindheid te laten functioneren, is er aan het begin van iedere melding een icoon toegevoegd.

	Algemene informatie	(momenteel geen podcasts, verbonden met clouddienst)
	Geslaagde actie	(aanmaken podcast, opslaan notities, annuleren podcasts)
	Waarschuwing	(verbinding met clouddienst is traag)
	Foutmelding	(API kon niet bereikt worden, verbinding met clouddienst is weggevallen)

Tabel 5: Iconen om de bedoeling van een melding aan te geven ongeacht de kleur

Door gebruik te maken van notificaties in alom bekende kleuren kan de gebruiker in één oogopslag zien of een bepaalde handeling is gelukt.

- Blauwe melding: simpel bericht (bijvoorbeeld: er zijn geen podcasts gevonden)
- Groene melding: een handeling is goed afgerond (bijvoorbeeld: het aanmaken van een podcast is gelukt)
- Gele melding: een waarschuwing (bijvoorbeeld: de verbinding met Azure Web PubSub is traag)
- Rode melding: een foutmelding (bijvoorbeeld: de podcast kon niet aangemaakt worden)

De applicatie implementeert het F-patroon om de gebruikers op een gebruikelijke manier gebruik te laten maken van het Podcast Portaal. Het F-patroon is ontstaan uit een onderzoek van NNGroup. Gebruikers lezen over het algemeen eerst dominante delen van links naar rechts, gevolgd door het verticaal zoeken naar voor hen interessante paragrafen.

Ook maakt de applicatie gebruik van een spinner om aan te tonen dat een bepaalde handeling wordt uitgevoerd. De applicatie heeft ook een hamburgermenu aan de linkerzijde om de gebruiker te laten navigeren tussen alle routes. Met behulp van breadcrumbs behoudt de gebruiker het overzicht over de applicatie.

7. Projectorganisatie

Bij de ontwikkeling van het Podcast Portaal heb ik gebruik gemaakt van de **scrum** methode. Er is gekozen voor scrum omdat de applicatie uit veel verschillende features bestaat en deze met de scrum methode iteratief opgepakt konden worden. Dit was erg handig omdat de meeste frameworks volledig nieuw voor mij waren en ik niet goed van tevoren kon inschatten hoeveel tijd ik nodig zou hebben per framework.

7.1 Waarom scrum

Er is gekozen om voor de organisatie van het project gebruik te maken van de scrummethode. Deze methode werd gebruikt omdat alle ontwikkelaars bij Betabit gebruik maken van scrum. Ook is gekozen voor scrum omdat hierbij het uitvoeren van taken opgedeeld kon worden in korte iteraties/sprints.

Op deze manier was het ook makkelijker om bij te houden welke taken er uitgevoerd gingen worden en welke nog gedaan moesten worden. Dit maakte het ook weer eenvoudiger om met pull requests features samen te voegen met de live-versie van het project. Door nieuwe functionaliteit stuk voor stuk te beoordelen was het makkelijker voor de product owner om de functionaliteit te beoordelen.

Een andere optie voor de organisatie van het project was de Waterfall methode omdat de prototypes een vast aantal requirements hadden. Er is toch gekozen voor scrum omdat het niet goed in te schatten was hoeveel tijd er nodig was voor ieder onderzoek en prototype.

7.2 Communicatie met stakeholders

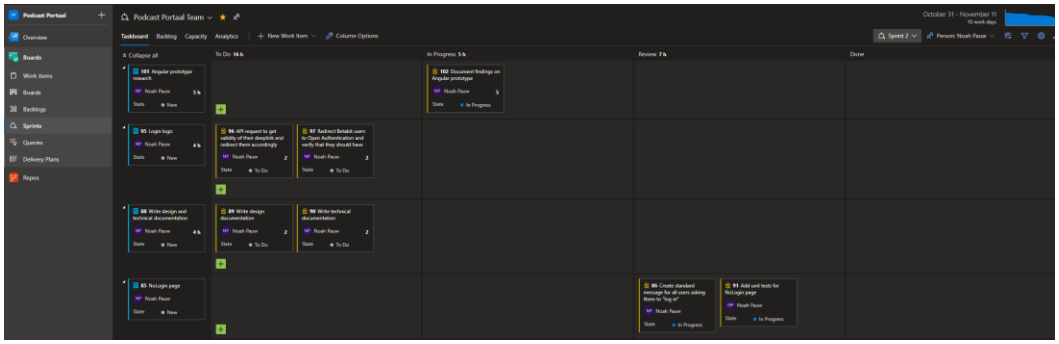
Voordat ik ben begonnen aan de ontwikkeling van de applicatie heb ik samen met één van de twee hosts om de tafel gezeten om te praten over de eisen van de app. Op basis van deze eisen heb ik een aantal requirements opgesteld en deze teruggestuurd naar de host. De hosts van de Betatalks podcast namen de rol van de product owner(s) aan.

Omdat beide hosts helaas weinig tijd hadden om de requirements te beoordelen, is de rol van product owner uiteindelijk overgedragen aan mijn bedrijfsbegeleider Dinand ten Hooven en de regiomanager van Apeldoorn, Emile Voogt. Ik kon bij hen terecht om de requirements door te nemen. De requirements (pagina 30, 9. Requirements) zijn SMART opgesteld om zo nauwkeurig mogelijk aan de slag te gaan.

7.3 Scrumbord

Omdat Betabit bijna exclusief gebruik maakt van Microsoft Azure producten, heb ik ook gebruik gemaakt van Azure DevOps om mijn project in te delen. Ik heb een nieuw project aangemaakt op de DevOps omgeving. Hier heb ik vervolgens een backlog aangemaakt.

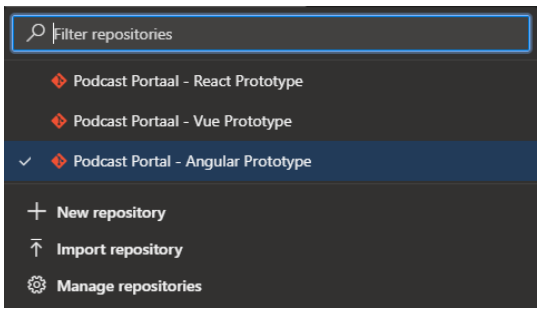
Omdat mijn projectteam alleen uit mijzelf bestond heb ik de rollen van *scrum master* en *development team* op mij genomen. Iedere dag maak ik gebruik van het scrumbord en kijk ik of taken waar ik mee bezig ben geweest afgerond kunnen worden en welke ik daarna zou kunnen oppakken.



Figuur 5: Een deel van de tweede sprint van het project in Azure DevOps.

7.4 Repositories

Alle code van het project is terug te zien op de repository die speciaal voor dit project is aangemaakt. Voor ieder framework is een aparte repository aangemaakt.



De repositories werden ook gehost op de Microsoft Azure DevOps omgeving. Op deze manier kon ik ook gemakkelijk teamleden binnen Betabit toevoegen aan het project die bereid waren om de code te beoordelen.

Aan het einde van mijn afstudeerstage heeft Betabit dan ook gelijk toegang tot alle repositories. Hierin is alle code, documentatie, tests en beoordelingscriteria te vinden.

Figuur 6: De repositories binnen het project. Ieder prototype heeft een eigen repository.

7.4.1 Pull requests

Wanneer ik een feature had afgerond maakte ik een pull request aan om de branch samen te voegen met de hoofdbranch. Ik wilde namelijk graag dat de code eerst beoordeeld zou worden. De code op de branch werd vervolgens nagekeken en beoordeeld door de product owner.

Als de code van een feature afgekeurd wordt moeten er eerst wijzigingen aangebracht worden voordat het opnieuw beoordeeld gaat worden. Pas wanneer de product owner de feature goed beoordeeld en accepteert wordt de feature samengevoegd met de hoofdbranch.

Figuur 7: Flowchart over het verloop van het samenvoegen van een nieuwe feature. Features worden pas samengevoegd na goedkeuring van de product owner

7.5 Sprint reviews

Voor de aanvang van de retrospectives blikken mijn begeleider (ook de product owner in dit project) en ik kort terug op het verloop van de sprint en welke resultaten hierbij behaald zijn. We kijken naar de voortgang van de applicatie, de burndown chart en de taken die nog openstaan indien van toepassing. Dit doen wij tweewekelijks voor de afloop van de huidige sprint.

7.6 Retrospectives

Aan het einde van iedere sprint heb ik een retrospective gehouden. Omdat er nog een andere stagiair was die bezig was met zijn eigen opdracht, hebben we samen na afloop van onze sprints een retrospective gehouden. Op deze manier konden we elkaar nog adviezen geven en bespreken hoe we tot actiepunten konden komen.

We hebben veel verschillende soorten retrospectives gehouden. In het begin hebben we Mad, Sad, Glads gehouden om aan te geven wat er goed en minder goed ging tijdens de afgelopen sprint. Daarna hebben we verschillende soorten retrospectives gehouden, zoals de M&M retrospective en de sailboat Agile retrospective (de le Maza, 2020).

De reden hiervoor was om elke keer een verse draai te geven aan de retrospectives. Verder is dit ook gedaan om specifieke problemen aan te kaarten. Bijvoorbeeld werd de sailboat retrospective gebruikt wanneer er duidelijke obstakels tijdens een sprint ondervonden werden.

7.6.1 Actiepunten

Na afloop van deze retrospectives hebben we een aantal actiepunten opgesteld. We hebben opgeschreven wat er precies misging tijdens de vorige sprint. Hierna hebben we samen besproken wat we hieraan zouden kunnen doen. Deze actiepunten werden toegevoegd aan één grote groeiende lijst. Deze lijst werd tijdens de volgende retrospective geraadpleegd om te kijken of we daadwerkelijk iets met de actiepunten hebben gedaan.

Zie Figuur 25: Actiepunten na afloop van de eerste sprint van het project voor de actiepunten naar aanleiding van de eerste sprint.

8. Kwaliteitsborging

Het is belangrijk om bij de oplevering van software aan te kunnen geven dat het product goed functioneert. Voor ieder prototype heb ik ook onderzocht hoe dit duidelijk gemaakt kan worden zodat ik kan garanderen dat ieder prototype ook goed werkt.

8.1 Unit tests

Ieder framework dat ik heb gebruikt voor de ontwikkeling van de prototypes maakt gebruik van een testing library. Voor de TypeScript frameworks heb ik gebruik gemaakt van de TypeScript testing library, Jasmine en Jest. Voor Blazor heb ik bUnit gebruikt.

8.2 Code kwaliteit

Ik heb gebruik gemaakt van verschillende soorten development tools om er voor te zorgen dat het testen ook goed uitgevoerd werd. Hiervoor heb ik gekeken naar een aantal tools waar ik al bekend mee was. Ik heb onderzocht of de frameworks een aantal van deze tools al geïntegreerd hadden. Dit heeft het testen van de applicaties een stuk inzichtelijker gemaakt door preciezer aan te geven waar de code nog op stuk kon lopen.

- TypeScript en C# code coverage:
  - Om duidelijk te maken welke code er allemaal wordt getest en welke functies en branches nog niet.
  - Het streven was om 100% code coverage te behalen. Mijn begeleider gebruikte hier altijd een metafoor voor over het vertrouwen hebben in een auto die bijvoorbeeld voor 80% goed functioneert en of ik daar akkoord mee zou gaan.
- SonarQube voor React en Angular (niet geïmplementeerd):
  - SonarQube is een open source (Fu, 2022) Code Quality Assurance programma waarmee de broncode van een applicatie geanalyseerd wordt. Het kan rapporten genereren over de kwaliteit van een codeproject (Malloy, 2020).
  - Om de kwaliteit van de code ook te controleren wordt er gebruik gemaakt van SonarQube. Met SonarQube is het mogelijk kwaliteit, code coverage en code smell te controleren, op te sporen en op te lossen.
  - Ik persoonlijk vind SonarQube een ideaal programma om de kwaliteit van de code zelf te waarborgen. Ik heb het helaas niet volledig kunnen implementeren omdat ik niet genoeg rechten heb op mijn werklaptop om SonarQube uit te voeren. Verder in het verslag wordt aangegeven wat mijn aanpak was geweest en waarom ik SonarQube een goed programma vindt.
- CI/CD (niet geïmplementeerd):
  - Ik heb met de product owner overlegd over het gebruik van Azure Pipelines. Het is mogelijk om in de repository omgeving een aantal processen automatisch uit te laten voeren om het project te testen (Kulla-Mader et al, 2023). Om dit te implementeren miste ik een aantal rechten waardoor ik deze pipeline niet kon aanmaken. In het adviesrapport beschrijf ik hoe dit eventueel alsnog gebouwd zou kunnen worden.
- nyc (Istanbul) code coverage:
  - Angular wordt standaard geleverd met code coverage. React heeft hiervoor een externe library nodig die dit mogelijk maakt. nyc (voorheen Istanbul) code coverage maakt dit alsnog mogelijk. Istanbul genereert een code coverage rapport waarin wordt aangetoond wat er precies getest wordt en wat niet.

8.3 Code coverage

Voor de drie prototypes zijn tests geschreven om de kwaliteit van de code te waarborgen. Het was de bedoeling om voor iedere class een aantal unit tests te schrijven die de functionaliteit ervan konden testen. Op deze manier was het dan ook makkelijker bij te houden of sommige features in de applicatie niet meer werkten na het toevoegen van nieuwe features.

Door code coverage toe te passen kon ik bijhouden welke functies en methodes allemaal getest werden in de applicatie. Met branch coverage is ook te zien of alle mogelijke scenario's getest worden, bijvoorbeeld door verschillende execution paths na beslissingen (bijvoorbeeld `if statements`) te testen *(Schults, 2021)*.

Om terug te komen op de metafoor die mijn begeleider eerder noemde: met hoeveel code coverage zou ik akkoord gaan als ik de applicatie in gebruik zou nemen? Uit onderzoek is gebleken dat ontwikkelaars gemiddeld met ongeveer 80% code coverage akkoord gaan *(Pittet)*.

Voor het Podcast Portaal is gestreefd naar **100% code coverage**. Voor de kwaliteit van de code is code coverage niet de beste metriek om te gebruiken. Dit komt doordat methodes die enkel aangeroepen worden maar verder niet getest worden, ook meegenomen worden in het code coverage rapport. Echter biedt code coverage wel inzicht over welke onderdelen van de code nog niet of onvoldoende getest worden *(Sealights, 2020)*.

In de bijlagen in Figuur 28 is een code coverage rapport te zien van het Angular project.

8.4 Code Quality Assurance met SonarQube

SonarQube is een applicatie dat rapporten kan opstellen met daarin een beoordeling van de geschreven code. Met SonarQube worden bijvoorbeeld code smells opgespoord. Code smells zijn karakteristieken in de code van een programma die een mogelijk dieper probleem aantonen *(Fowler, 2006)*. Een voorbeeld hiervan is bijvoorbeeld een naam van een functie die niets of weinig te maken heeft met de inhoud ervan. Een ander voorbeeld is (bijna) identieke code die op meerdere plaatsen in het programma terug te vinden is.

Volgens Pavel Mikula, een ontwikkelaar van SonarQube, is het niet mogelijk om de razor syntax van Blazor te testen met SonarQube *(Mikula, Pavel, 2022)*. Er is nog niet bekend of dit ooit mogelijk wordt gemaakt.

Helaas is SonarQube niet gebruikt tijdens de ontwikkeling van de prototypes. Ik kon het SonarQube programma niet op mijn werklaptop uitvoeren om beveiligingsredenen.

Wél is er in de React en Angular prototypes aangegeven hoe SonarQube alsnog gebruikt kan worden om de codekwaliteit te testen. Hiervoor is alleen de SonarQube applicatie nodig en een nieuw `npm command` (in het geval van React en Angular) en een speciaal bestand genaamd `sonar-project.properties` dat aangeeft op welke port SonarQube uitgevoerd wordt en welke bestanden er gecontroleerd moeten gaan worden.

Voor de implementatie van SonarQube in Angular, zie Figuur 34 in de bijlagen.

8.5 CI/CD met Azure Pipelines

Om de applicatie op de repository automatisch te builden en te testen wilde ik gebruik maken van Azure Pipelines voor Continuous Integration en uiteindelijk Continuous Deployment.

Om de code te testen wilde ik een aantal jobs instellen op het Azure Repos platform om de applicatie te builden en te testen. Hiervoor kan een YAML bestand aangemaakt worden met daarin een aantal jobs die uitgevoerd kunnen worden.

```
stages:
        - installation
        - test

installation:
        stage: installation
        script:
                - cd ./Podcast-Portal
                - npm install
test:
        stage: test
        script:
                - ng lint
                - ng test
                - ng test --code-coverage
```

Ik heb helaas niet de juiste rechten gekregen op de Azure DevOps omgeving om pipelines aan te maken. Ik heb gevraagd of ik deze rechten misschien nog zou kunnen krijgen, maar daarvoor moest een ander abonnement aangeschaft worden. Daarom is er besloten om niet met Azure Pipelines te gaan werken voor dit project.

8.6 nyc (Istanbul) code coverage

Angular ondersteunt code coverage standaard. React wordt niet geleverd met een implementatie van code coverage. Hiervoor moet een externe library gebruikt worden die geleverd wordt met het Jest framework *(Bennett, 2020)*.

Omdat een React applicatie wel standaard met het Jest framework geleverd, kan er direct gebruik gemaakt worden van nyc. Er wordt standaard een nieuw command in het `package.json` bestand aangemaakt genaamd coverage. Alleen als de applicatie is aangemaakt met het Jest framework kan dit command uitgevoerd worden. Door `npm run coverage` uit te voeren wordt automatisch de code coverage van het huidige project getest.

8.7 Code standards

Om er ook voor te zorgen dat de ontwikkelaar die verder gaat met de ontwikkeling van het portaal daadwerkelijk mijn keuzes kan begrijpen, heb ik mij verdiept in ieder framework door actief hun tutorials te volgen.

Het doel hierbij was het begrijpen van de basis van een framework en hoe hier volgens de ontwikkelaars applicaties mee gebouwd worden. Ervan uitgaande dat de meeste ontwikkelaars met deze frameworks gebruik maken van de aanbevolen ontwikkelmethoden, is het Podcast Portaal ook op deze manier opgebouwd.

Dit helpt mij ook met het toelichten van bepaalde keuzes en het sneller kunnen ontwikkelen van de prototypes.

9. Requirements

Op basis van de eisen van de product owner heb ik een aantal requirements opgesteld. Wel leidde het opstellen van deze requirements tot vragen van mijn kant.

Voordat er verder werd gegaan met het opstellen van deze requirements, was er een afspraak ingepland met de product owner in Rotterdam. Deze is helaas niet doorgegaan waardoor de vragen via de mail zijn opgestuurd.

Nadat hier antwoord op werd gegeven is het opstellen van de requirements verder gegaan.

Er is overlegd met de product owner over het maken van een account om toegang te krijgen tot de applicatie. Omdat de product owner het aanmaken van accounts niet nodig leek, is er gekozen om de hosts in te laten loggen met hun Betabit account met Azure Active Directory en OAuth 2.0. Verder is er ook de keuze gemaakt om gasten in te laten loggen met een tijdelijk geldige deeplink zodat zij ook geen account hoeven aan te maken.

De requirements zijn na het opstellen overlegd met de product owner. Hij is hiermee akkoord gegaan.

9.1 Requirements voor de applicatie

Req	Omschrijving	FR/NFR*)	Prioriteit
RQ01	De host moet de naam van de gast(en) kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ02	De host moet de datum en de tijd van de podcastopname kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ03	De host moet de topics van de podcast kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ04	De host moet de biografie van de gast(en) kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ05	De host moet commentaar kunnen achterlaten voor de gast van de huidige podcast via de applicatie tijdens het opstellen van een podcast.	FR	M
RQ06	De host moet een titel kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ07	De host moet het e-mailadres van de gast kunnen toevoegen via de applicatie.	FR	M
RQ08	De host moet een kopieerbare link naar de Zencastr omgeving van Betabit kunnen toevoegen aan de huidige podcast.	FR	M
RQ09	De host moet een TRQ (Totally Random Question) kunnen toevoegen aan de huidige podcast via de applicatie.	FR	M
RQ10	De hosts moet de gegevens van een bestaande podcast kunnen aanpassen	FR	M
RQ11	De host moet een bestaande podcast kunnen annuleren	FR	M
RQ12	De host en regisseur moeten kunnen inloggen met hun Betabit account.	FR	M
RQ13	De host moet een eigen notitieruimte kunnen gebruiken die niet zichtbaar is voor de gasten in het podcastportaal.	FR	M
RQ14	De gasten moeten hun geluidsopname kunnen uploaden via het podcastportaal naar de database van de applicatie	FR	M
RQ15	De gasten moeten hun woonadres kunnen achterlaten op het podcastportaal.	FR	M
RQ16	De gasten moeten hun eigen biografie kunnen aanpassen.	FR	M
RQ17	De gasten moet kunnen inloggen met een persoonlijke deeplink zodat zij tijdelijk en zonder account toegang krijgen tot de applicatie.	FR	M
RQ18	De applicatie moet de opname van de gasten alleen accepteren in het .wav formaat zodat Betabit de hoogste audiokwaliteit kan ontvangen.	NFR	M
RQ19	De applicatie moet de geldigheid van een deeplink kunnen verifiëren om te controleren of de gast toegang hoort te krijgen tot de applicatie.	FR	M
RQ20	De applicatie moet een eigen huisstijl gebruiken en niet die van Betabit om de applicatie open source te houden.	NFR	M
RQ21	De applicatie moet een single page application worden om de laadtijden van de applicatie te verlagen	NFR	M

RQ22	De Live Notities omgeving moet serverless notities kunnen uitwisselen.	FR	S
RQ23	De hosts moeten middels een serverless socketverbinding notities kunnen uitwisselen met de regisseur van de podcast in de applicatie die niet te zien zijn voor de gasten	FR	S
RQ24	De gasten moeten middels een serverless socketverbinding notities kunnen uitwisselen met de hosts van de podcast in de applicatie	FR	S
RQ25	De applicatie moet automatisch een e-mail versturen naar de gast met daarin commentaar voor de gast van de hosts, de datum en tijd van de podcast, de titel, onderwerpen en deeplink van de podcast nadat een podcast aangemaakt wordt.	FR	C
RQ26	De live notities moeten automatisch geparsed worden naar markdown zodat de notities makkelijker te lezen zijn.	FR	C
RQ27	De hosts moeten vanuit de regie een :AFRONDEN signaal kunnen ontvangen wanneer de podcast afgebouwd gaat worden	FR	C
RQ28	De applicatie moet meerdere talen ondersteunen omdat er ook veel gasten uit verschillende landen worden uitgenodigd.	NFR	W

*Figuur 8: Requirements voor de applicatie, geprioriteerd op basis van MOSCOW. *)FR = functioneel requirement, NFR = niet-functioneel requirement*

10. Product: Het Podcast Portaal

Tijdens mijn afstudeertraject ben ik bezig geweest met de ontwikkeling van het Podcast Portaal. In dit hoofdstuk licht ik toe hoe de applicatie is opgebouwd en hoe deze precies werkt.

10.1 Afbakening

In de volgende tabel is te zien welke onderdelen van de applicatie ik heb gebouwd. Verder wordt er aangetoond waar ik bij geholpen of geadviseerd heb en welke onderdelen ik niet heb gebouwd.

Zelf ontwikkeld
Geadviseerd/ samengewerkt
Niet door mij ontwikkeld

Figuur 9: Legenda over afbakening van ontwikkeltaken

Ontwikkelen van een prototype van de applicatie in de React library
Ontwikkelen van een prototype van de applicatie in het Angular framework
Ontwikkelen van een prototype van de applicatie in het Blazor framework
Ontwerpen van een UI/UX op basis van UX best practices en algemene ontwerpprincipes*
Gegevens kunnen ophalen, plaatsen, aanpassen en verwijderen uit de backend vanuit de frontend
Configureren van de Azure Web PubSub dienst in Microsoft Azure
Ontwikkelen van een WebSocket programma in TypeScript en C# om vanaf de client-side notities te versturen naar een clouddienst en te verwerken (bijvoorbeeld het ontvangen van een notitie van een andere client)
Ontwikkelen van de live notities omgeving met WebSockets
Onderzoek doen naar het maken van een applicatie in verschillende web-frameworks/libraries om deze op een conventionele manier op te bouwen**
Bureau en- enquête onderzoek uitvoeren naar React, Angular en Blazor op algemeen niveau
Configureren Azure Blob Storage opslag en containers om bestanden te uploaden naar de cloud
Het ontwerpen van de structuur van de backend (class-structuur, API endpoints, clouddienst-configuratie)
Onderzoek doen naar beschikbare typen databases voor de backend
Adviseren in mogelijke opties voor de serverless Live Notities omgeving
Hosten van de prototypes op het Azure cloudplatform
De volledige ontwikkeling van de Podcast Portaal API in het .NET framework
Genereren van een deeplink voor de gasten om in te loggen op de applicatie
Opstellen van de Azure Table Storage database van de applicatie
OAuth 2.0 afhandeling in de backend door de API (bijvoorbeeld Oscar van Tol mag wél bij de applicatie omdat hij een host is van de podcast)

Tabel 6: Afbakening van de ontwikkeling van het Podcast Portaal; wat heb ik wel en niet gemaakt

* Deze ontwerpprincipes worden op pagina 19 Deelvraag 3: User Experience behandeld

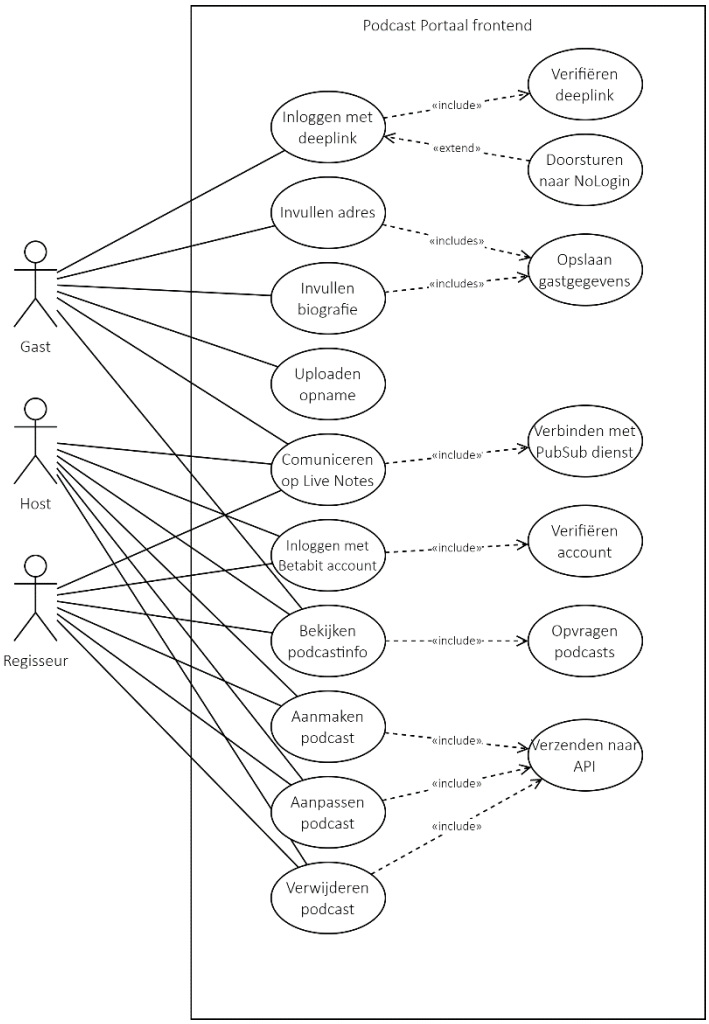
** Met een conventionele manier opbouwen doel ik op het ontwikkelen van een applicatie op basis van tutorials, documentatie en andere applicaties die gebouwd zijn met hetzelfde framework. Op deze manier kunnen ontwikkelaars bij Betabit met ervaring in dat framework beter aan de slag

10.2 Actors

Er zijn drie verschillende soorten actors die gebruik kunnen maken van de applicatie:

- **Gasten**: Worden uitgenodigd door de hosts van de podcast
  - Volgens requirement RQ17 kunnen de gasten gebruik maken van de applicatie zonder hiervoor een account te hoeven aanmaken
- **Hosts**: De presentatoren van de podcast
  - Volgens requirement RQ12 hoeven de hosts geen account aan te maken om in te loggen op de applicatie. Zij kunnen met behulp van Azure Active Directory gebruik maken van hun eigen account
- **Regisseurs**: De regie van de podcast

10.3 Use case diagram



Figuur 10: Use case diagram van de applicatie met gebruikers, hosts en regisseurs

Deze verschillende rollen zijn van groot belang voor de applicatie. Zo kunnen er bijvoorbeeld **geen** luisteraars van de podcast inloggen op het portaal. Alleen gasten, hosts en regisseurs krijgen toegang tot de applicatie.

Gasten

De gasten krijgen van de hosts een deeplink toegestuurd via de mail (requirement RQ25). Hierin staat wat informatie over de inhoud van de podcast, de titel en de datum en tijd. De gast kan op hun eigen omgeving wel wat gegevens aanpassen. Zo kunnen zij hun adres achterlaten via het portaal als zij een goodiebag thuisgestuurd willen krijgen (requirement RQ15). Verder kunnen zij ook een stukje tekst over zichzelf schrijven (requirement RQ16). Tot slot kunnen zij ook via een live-omgeving notities uitwisselen met de hosts en regie (requirement RQ24).

Verder kunnen de gasten ook hun eigen geluidsopname uploaden naar de regie/hosts van de podcast via de applicatie (requirement RQ14). Dit wordt gedaan om de hoogst mogelijke audiokwaliteit te hebben van beide partijen. Het is dan ook om die reden dat gebruikers alleen lossless .WAV bestanden kunnen uploaden (requirement RQ18). Dit type bestand biedt namelijk de beste geluidskwaliteit.

Wanneer de deeplink die ze hebben ontvangen van de hosts verloopt hebben ze geen toegang meer tot het portaal.

Hosts

De hosts zijn de presentatoren van de podcast. Zij zijn het visitekaartje van Betatalks en praten samen met de gasten over verschillende onderwerpen in de softwarewereld.

De hosts krijgen toegang tot het portaal door in te loggen met hun eigen Betabit account (requirement RQ12). Bij het inloggen wordt een OAuth 2.0 proces in gang gezet. Hierbij wordt gekeken of de huidige werknemer toegang hoort te krijgen tot het portaal. Dit wordt gedaan omdat ook met een Betabit account niet iedere werknemer toegang hoort te krijgen tot de applicatie. De backend regelt de authenticatie van de huidige gebruiker met een OAuth 2.0 oplossing. Nadat de gebruiker heeft ingelogd met hun Betabit account en zij toegang hebben tot het portaal (e.g. alleen de hosts en regie) worden zij doorverwezen naar de homepagina van het portaal.

De hosts kunnen zelf alle podcasts zien, nieuwe podcasts inplannen, bestaande podcasts aanpassen (requirement RQ10) of annuleren (requirement RQ11) en communiceren met de gasten en regisseurs via een live-omgeving (requirement RQ23).

Regisseurs

De regisseurs zitten op de achtergrond. Zij communiceren nauw met de presentatoren voor en tijdens de opnames. Zij loggen op dezelfde manier in als de presentatoren. Verder hebben ze ook dezelfde rechten als de presentatoren. Zij mogen dus ook podcasts inplannen, aanpassen en annuleren.

Regisseurs moeten wel via het notitieplatform signalen kunnen doorsturen. Zo wil de product owner bijvoorbeeld dat zij een :AFRONDEN signaal kunnen versturen dat aangeeft dat de podcast langzaamaan moet beginnen met afbouwen (requirement RQ27).

Deze functionaliteit is alleen beschikbaar voor de regisseurs van de podcasts. De hosts hoeven geen signalen door te sturen.

10.4 Toegang tot de applicatie

De gasten krijgen van de hosts een deeplink toegestuurd via de e-mail. Volgens requirement RQ25 moet deze e-mail automatisch aangemaakt worden na het inplannen van een podcast. Hiermee krijgen zij automatisch toegang tot de applicatie. Deze deeplink is geldig tot een (aantal) dag(en) na de opname van de podcast. Daarna zal deze verlopen en heeft de gast geen toegang meer tot het portaal.

De hosts en regisseur(s) hoeven geen eigen account te maken voor de applicatie. Zij krijgen toegang tot de applicatie met hun Betabit account. In “Inloggen met OAuth 2.0 en Azure Active Directory” op pagina 35 wordt hier verder op ingegaan.

De gast wordt met de deeplink direct naar het portaal doorgestuurd. Wanneer er een inlogpoging met een deeplink wordt gedaan, voert de applicatie een HTTP-request uit naar een API die speciaal ontwikkeld is voor de applicatie. Hierna zal in de Azure Table Storage database van de applicatie gekeken worden of deze deeplink bestaat. Als deze niet bestaat zal de gebruiker doorverwezen worden naar de NoLogin pagina.

Als de deeplink wél bestaat wordt deze doorgegeven aan de API. Vervolgens wordt er in de frontend gecontroleerd of de deeplink nog geldig is (requirement RQ19). Er zit een vervaldatum in het deeplink-object. Deze zal worden vergeleken met de huidige datum.

Overige medewerkers van Betabit en luisteraars van de podcast krijgen *geen* toegang tot het portaal. Nadat een gebruiker inlogt wordt er in de backend gecontroleerd of het account van de medewerker hoort bij een host of regisseur. Zo niet, dan wordt de gebruiker niet ingelogd.

10.4.1 Afbakening

Het ophalen van de deeplink en het endpoint om de deeplink naartoe te sturen heb ik niet zelf gebouwd. Dat onderdeel is gebouwd door de ontwikkelaars van de backend. De frontend is *alleen* verantwoordelijk voor:

- Het verzenden van de deeplink naar de backend
- Het controleren van de datum en validiteit van de deeplink
- Het afhandelen van een (on)geldige deeplink

Zie Figuur 30 in de bijlagen voor een sequence diagram over het validatieproces van de deeplink.

10.5 Inloggen met OAuth 2.0 en Azure Active Directory

De product owner heeft ervoor gekozen om gebruikers van de applicatie niet op te willen slaan in de database. Dit is niet nodig omdat de hosts en regie al een eigen Betabit account hebben en zich kunnen authentiseren met OAuth 2.0 en Azure AD. Er hoeft dus geen account aangemaakt te worden voor de applicatie (requirement RQ12).

De gasten krijgen ook geen eigen account om mee in te loggen, maar een deeplink waarmee zij automatisch tijdelijk toegang krijgen tot het portaal (requirement RQ17).

Als een medewerker van Betabit die geen toegang hoort te krijgen tot het podcast portaal probeert in te loggen, dan wordt dit afgevangen door de backend. De gebruiker zal dan een `401 Unauthorized` of een `403 Forbidden` terugkrijgen en doorverwezen worden naar een lege homepagina.

10.5.1 Afbakening

Voor het inloggen wordt gebruik gemaakt van het bestaande Microsoft Azure Active Directory systeem en het OAuth autorisatieprotocol. Hiervoor wordt alleen een URL aangeroepen die de gebruiker na het inloggen weer terugverwijst naar de frontend.

De frontend die ik heb ontwikkeld is alleen verantwoordelijk voor:

- Het doorverwijzen van de gebruiker naar de loginpagina van Microsoft

Het validatieproces in de backend en de OAuth oplossing van Microsoft vallen buiten de scope van mijn project.

Voor de volledigheid is in Figuur 32 in de bijlagen de volledige flow van het inloggen in te zien in een sequence diagram. Ik heb mij alleen bezig gehouden met het doorverwijzen van de gebruiker.

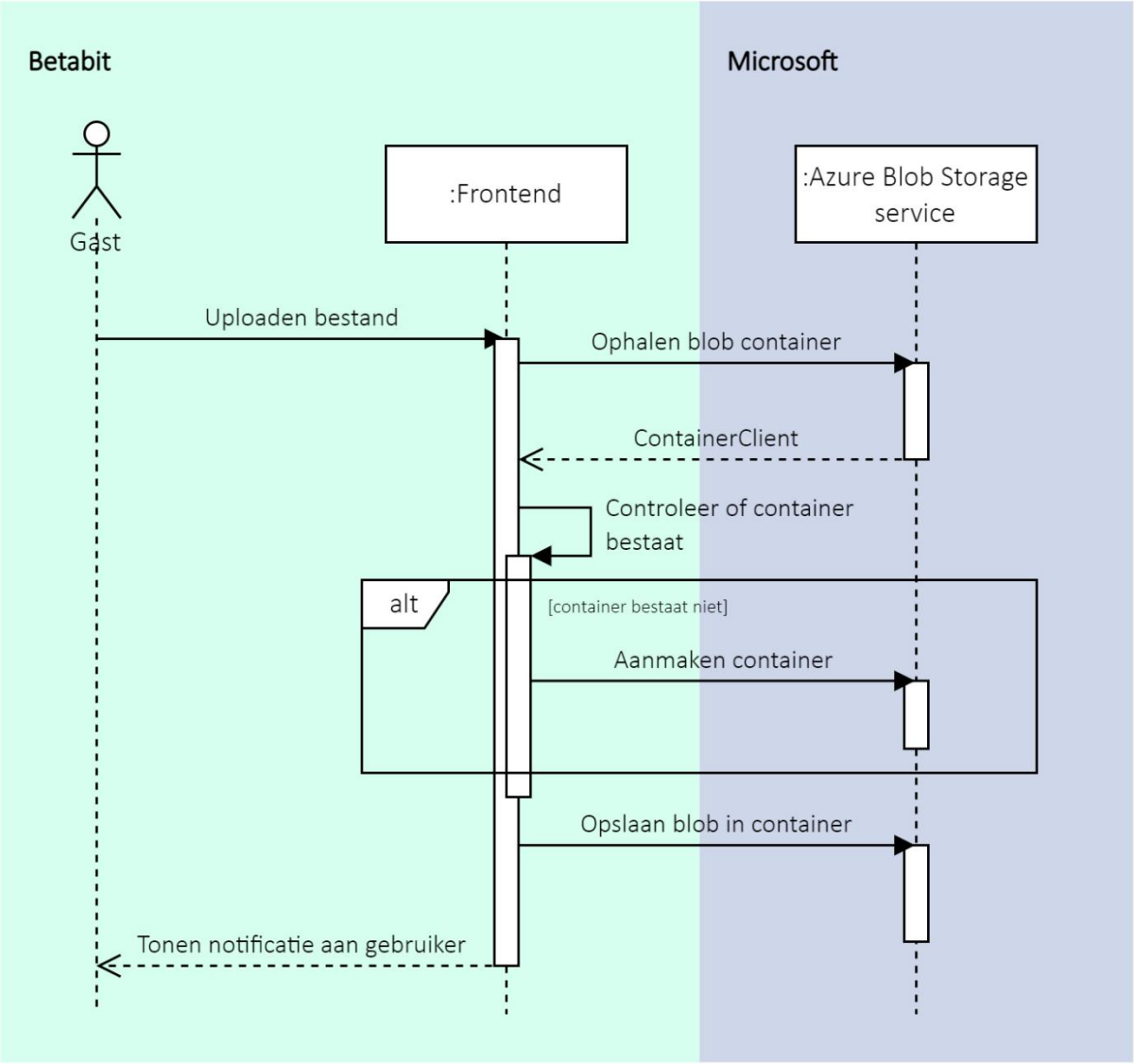
10.6 Bestanden uploaden naar Azure Blob Storage

De product owner wilde graag dat gasten met het portaal geluidsbestanden kunnen uploaden (requirement RQ14). Dit is mogelijk gemaakt door bestanden te versturen naar Azure Blob Storage.

Azure Blob Storage is een dienst van Microsoft waar (grote) bestanden opgeslagen kunnen worden in de cloud van Microsoft. Er is gekozen om Blob Storage te gebruiken voor dit project omdat de opnamebestanden van de gasten zeer groot kunnen worden. Deze bestanden moeten namelijk in het .WAV formaat geüpload worden (requirement RQ18). Dit zijn grote geluidsbestanden die al gauw meerdere honderden megabytes in grootte kunnen worden. (Arthur Fox, n.d.)

Blob Storage biedt de mogelijkheid deze bestanden van dit formaat op te slaan en op te halen. Blob Storage maakt gebruik van **blob containers** om bestanden in op te slaan. De frontend controleert of een blob container bestaat. Hiervoor wordt een request gedaan naar de API van een Storage Service. Als deze niet bestaat wordt er een nieuwe container aangemaakt. Daarna worden de bestanden opgeslagen in de container. (Microsoft, 2022)

Dit proces loopt asynchroon in de frontend en maakt gebruik van een Shared Access Signature (SAS) token (jimmart-dev, et al., 2022) en Access Keys voor authenticatie (rickvdbosch, et al., 2022).



Figuur 11: Sequence diagram over het uploaden van bestanden naar Azure Blob Storage. Gasten kunnen geluidsopnames uploaden met de applicatie. Deze wordt vervolgens in een container opgeslagen op het Azure platform.

10.7 Architectuur

De applicatie maakt gebruik van bekende UX principes, zoals breadcrumbs om het overzicht te bewaren, responsive elementen en algemene design features waar gebruikers gewend aan zijn geraakt, zoals hamburger-menu's, knoppen die in of -uitgeschakeld zijn en onderlijnde links.

De navigatie van de applicatie moet op een simpele en duidelijke manier kunnen gebeuren. Met behulp van de breadcrumbs weet de gebruiker waar zij op dat moment zijn. Verder kan er genavigeerd worden via het zij-menu.

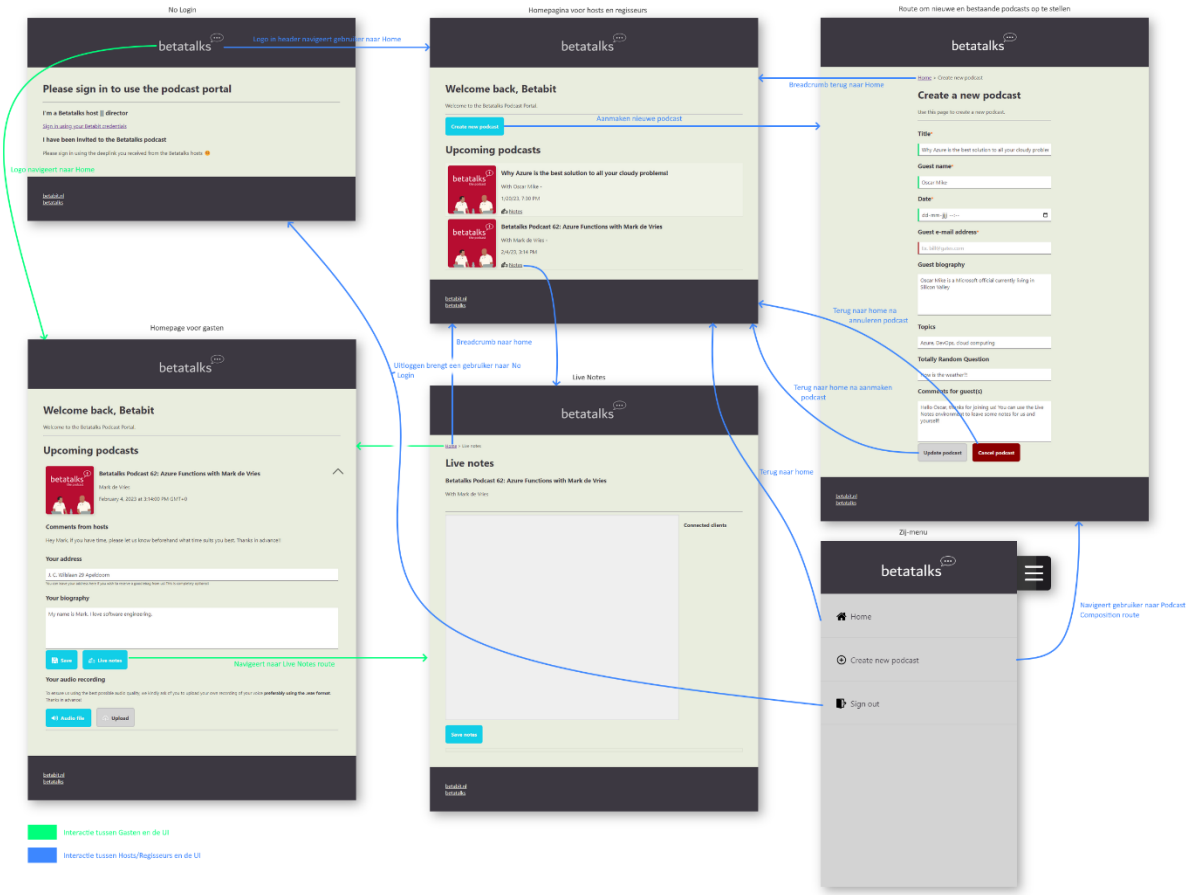
Wanneer een host een nieuwe podcast wil inplannen, zal deze naar de podcast compositie pagina worden gestuurd. Wanneer de host klikt op een bestaande podcast, zal deze ook naar de compositie pagina worden doorverwezen. Hier zullen de velden alvast ingevuld zijn en kan de podcast aangepast worden.

De applicatie wordt een single page application (SPA). Een SPA is een implementatie van een web applicatie die de inhoud hiervan bijwerkt met JavaScript APIs. Een SPA kan zorgen voor prestatieverbeteringen doordat er geen nieuwe pagina's gedownload hoeven te worden van een server (Mozilla, sd).

Een SPA maakt gebruik van router. Routes zijn pagina's die aan het begin van het bezoeken van de applicatie worden ingeladen. Zo hoeven de HTML, CSS en TypeScript/C# codes niet allemaal opnieuw opgehaald te worden zodra er genavigeerd wordt tussen pagina's.

10.7.1 Routes

Notitie: alle gebruikers beginnen bij de **Homepage Route** (vierde route voor de gasten) na het inloggen of navigeren met een geldige deeplink. Als een host uitlogt of een gebruiker probeert in te loggen met een ongeldige deeplink, zullen zij worden doorverwezen naar route 1 (No Login) Routes alleen voor gasten zijn lichtgroen en routes voor hosts en regisseurs zijn blauw.



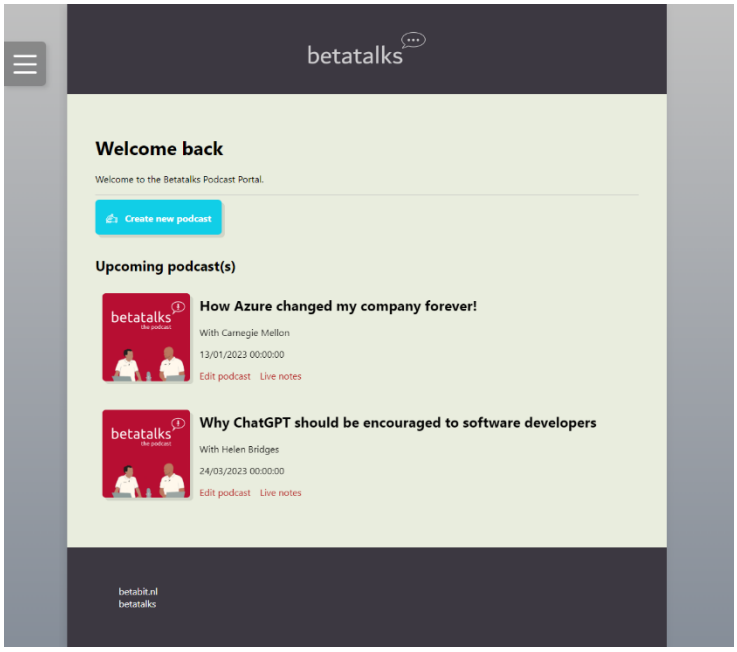
Figuur 12: Navigatie tussen "routes" en welke relaties pagina's tussen elkaar hebben voor gasten en hosts

10.8 Wireframes

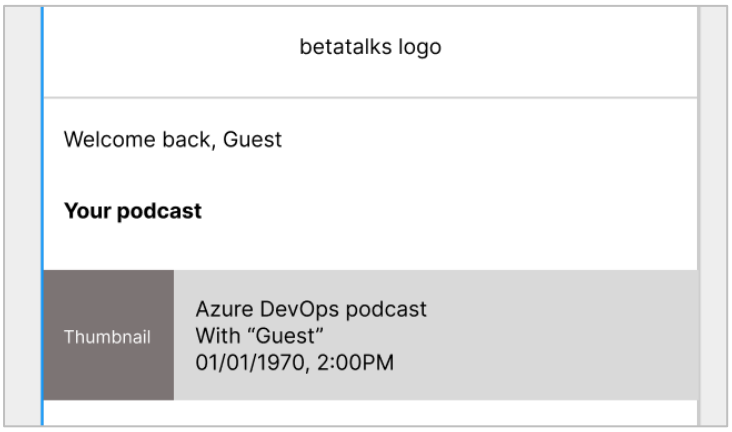
Voordat ik begon met de ontwikkeling van de prototypes, heb ik een aantal schetsen opgesteld. Hierbij heb ik vooral gekeken naar de gebruikerservaring en de huisstijl van Betabit. Er is een aantal wireframes opgesteld om de product owner een idee te geven van de structuur en opstelling van de applicatie.

10.8.1 Homepagina voor hosts

Als voorbeeld gebruik ik hier de homepagina van het portaal wanneer een host inlogt.



Figuur 13: Homepagina van het podcastportaal exclusief voor hosts en regisseurs



Figuur 14: De podcastlijst die de gast te zien krijgt.

Volgens requirement RQ20 wordt er geen gebruik gemaakt van de huisstijl van Betabit voor het uiterlijk van de applicatie.

De product owner wil juist een unieke stijl omdat het portaal uiteindelijk open source beschikbaar gesteld moet worden.

Omdat ik nog geen kleurenthema voor ogen had, heb ik een aantal wireframes gemaakt om een duidelijk beeld te schetsen over het gebruik van de app.

Op de homepagina kunnen de hosts direct alle podcasts bekijken. Vanaf hier kunnen zij ook navigeren naar de opstelpagina voor podcasts.

Door te klikken op een podcast in de lijst kunnen zij een bestaande podcast aanpassen of annuleren.

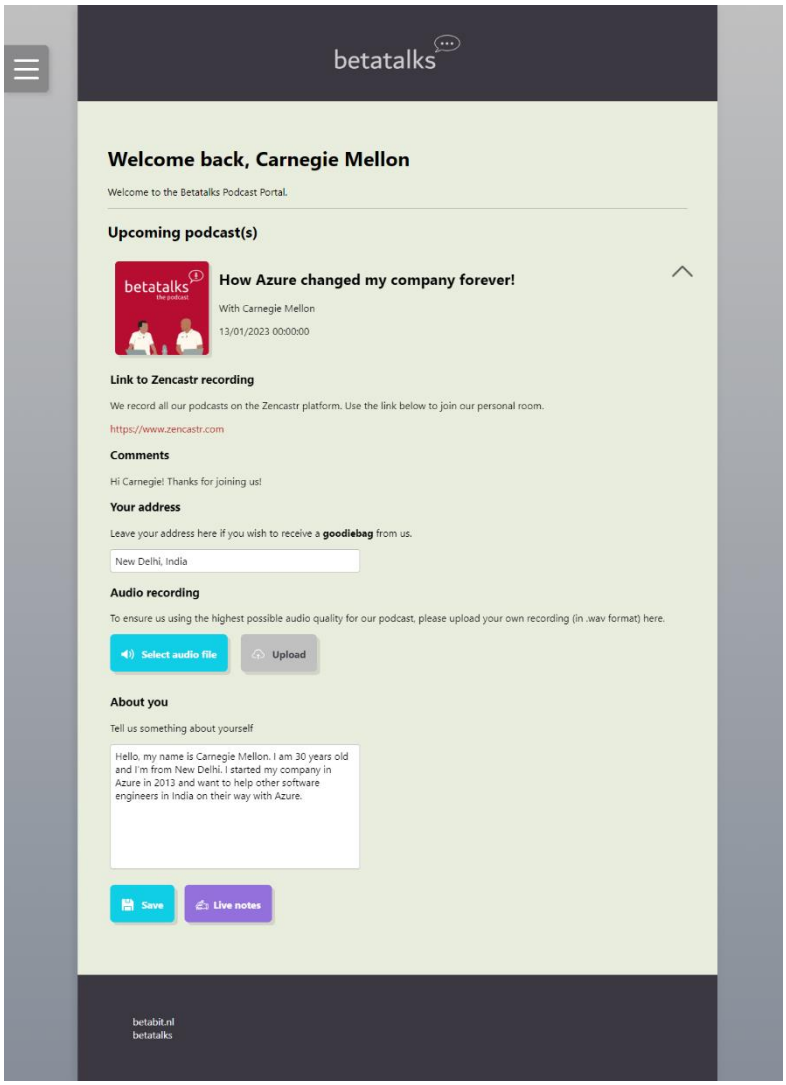
Linksboven zit een knop waarmee er een menu uitgevouwen kan worden. Hiermee kunnen alle gebruikers het portaal navigeren.

De gast krijgt ongeveer dezelfde pagina te zien. De route zal hetzelfde zijn. Er zal geen knop zijn om een nieuwe podcast aan te maken en er zal waarschijnlijk één podcast getoond worden, tenzij deze gast alvast meerdere keren is ingepland.

Wanneer de gast op een podcast klikt, zal deze worden doorverwezen naar een speciale gastpagina. Daar kunnen zij hun eigen biografie aanpassen en hun adres achterlaten mochten ze dat willen.

10.8.2 Bewerken van gastgegevens

De gasten kunnen via het podcastportaal hun eigen gegevens aanpassen. Volgens requirement RQ15 en RQ16 moeten de gasten in ieder geval hun adres achter kunnen laten en hun eigen biografie kunnen aanpassen. Het adres invoeren wordt gedaan omdat Betabit graag een goodiebag opstuurt naar de gast na afloop van de podcast.



Figuur 15: Route waar de gasten hun eigen gegevens kunnen aanpassen en zich in kunnen lezen over de datum en inhoud van de podcast.

Uploaden geluidsopname

Het is ook mogelijk voor de gasten om hier hun eigen geluidsopname te uploaden (requirement RQ14).

De frontend van het portaal laat gebruikers alleen bestanden in het Waveform (.wav) formaat uploaden omdat dit de hoogst mogelijke geluidskwaliteit biedt (Chiba, 2021) (requirement RQ18).

Dit bestand wordt daarna geüpload naar een Azure Blob Storage opslag-container vanuit de frontend.

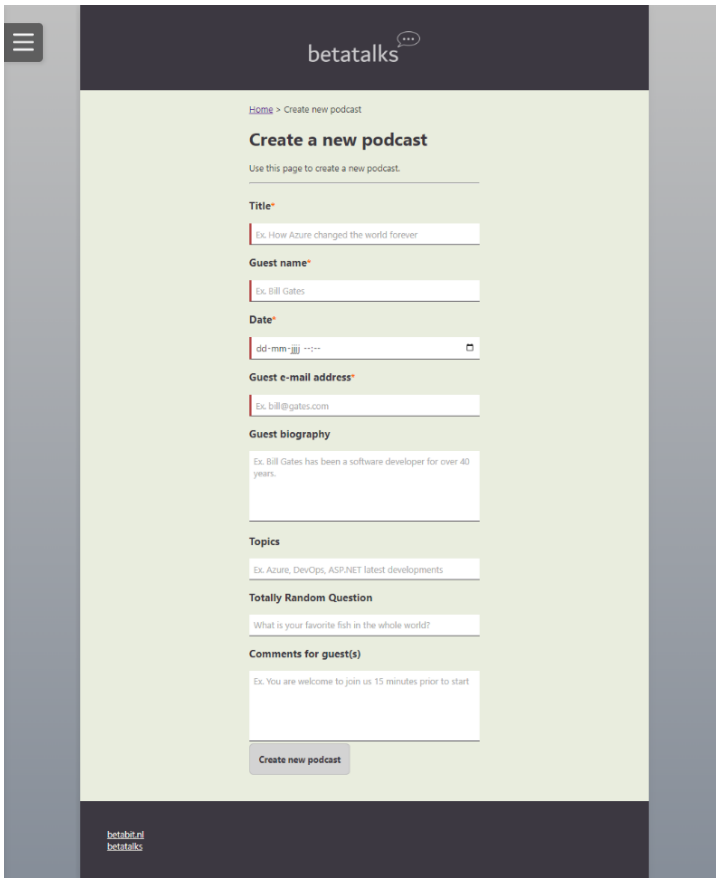
10.8.3 Podcasts inplannen

De hosts kunnen zelf podcasts inplannen. Hiervoor worden zij naar de podcast compositie pagina doorverwezen. Hier kunnen zij alle velden invullen die belangrijk zijn voor de hosts van de podcast. Bovenaan de pagina zijn breadcrumbs te vinden zodat de gebruiker in één oogopslag ziet waar zij ook alweer zijn.

Requirements

Notitie: Het opslaan van de link naar de Zencastr (requirement RQ08) is niet mogelijk omdat de database geen veld geconfigureerd heeft om deze op te kunnen slaan.

Volgens de requirements RQ01 tot en met RQ09 moet er een aantal velden beschikbaar zijn voor het aanmaken van een podcast.



Figuur 16: Compositiepagina waar hosts en regisseurs (bestaande) podcasts kunnen inplannen/aanpassen

De product owner heeft een lijst opgeleverd met de velden die altijd worden opgeslagen bij het inplannen van een podcast. Dit zijn:

- Titel
- Naam van de gast
- Datum met tijd (in UTC)
- E-mailadres van de gast
- Biografie van de gast
- Podcast topics
- De TRQ (Totally Random Question)
- Optioneel commentaar voor de gast

De gast kan na het inplannen van de podcast zelf de biografie aanpassen en antwoorden op het commentaar van de hosts.

Wanneer de hosts/regisseurs een podcast willen aanpassen komen zij op dezelfde pagina uit. Het verschil hierbij is dat de velden vooraf ingevuld zijn en zij de aanpassingen kunnen opslaan. Deze worden dan doorgevoerd naar de database.

Opslaan in de backend

Wanneer de host de podcast inplant of wijzigt, wordt er een verzoek gedaan naar de API van het portaal. Om de gebruiker duidelijk te maken dat er daadwerkelijk iets gebeurt, komt er een zwarte overlay met het Betabit logo en een draaiende cirkel in beeld. Wanneer deze verdwijnt wordt er of een groene melding of een foutmelding getoond.

10.8.4 Live notes

Volgens requirement RQ23 wil de product owner een omgeving waar de hosts en regie van de podcast notities met elkaar kunnen uitwisselen (requirement RQ22). Deze ruimte moet niet zichtbaar zijn voor de gasten. Zij krijgen een eigen ruimte waar zij ook realtime notities kunnen uitwisselen met de hosts.

De applicatie maakt ook gebruik van de WebSocket API (Mozilla, n.d.) en de Microsoft Azure Web PubSub dienst. Volgens requirement RQ22 moeten hosts middels een serverless socketverbinding kunnen communiceren met de gasten, regie en andere hosts van de podcast. Verder moeten de gasten ook gebruik kunnen maken van deze omgeving. De notities tussen de hosts en regisseurs moeten niet zichtbaar zijn voor de gasten.

Azure Web PubSub

Azure PubSub is een dienst van Microsoft die het mogelijk maakt om met het Publish-Subscribe model realtime (Liangying et al, 2023) chatberichten uit te wisselen tussen meerdere clients. Azure Web PubSub is een volledig door Microsoft beheerde dienst die ondersteuning biedt voor serverloze websockets. (Microsoft, 2022)

Omdat de Live Notities eis ook inhield dat de notitiefunctionaliteit serverless geïmplementeerd moest worden, is er gekozen voor Azure Web PubSub. Niet alleen omdat het serverless socketverbindingen ondersteunt (requirement RQ22), maar ook omdat Betabit veel ervaring heeft met Azure diensten.

Verschillende omgevingen

Er zijn binnen de applicatie twee aparte omgevingen waar live notities uitgewisseld kunnen worden. Zo is er een notitieomgeving voor hosts en regisseurs en een aparte omgeving voor gasten, hosts en regisseurs. Zo blijven de notities die bestemd zijn voor de communicatie tussen de hosts en regisseurs gescheiden van de gast.

De actors maken verbinding met een Azure Web PubSub dienst in de cloud. In de frontend wordt hier de WebSocket API voor gebruikt. Dit is een manier om two-way communicatie te realiseren tussen een browser en een server. Er is gekozen om de WebSocket API te gebruiken omdat de documentatie van Azure Web PubSub aanraadt om deze te gebruiken. (vicancy & dbradish-microsoft, 2023)

Markdown ondersteuning

De product owner wil dat de Live Notes omgeving *Markdown* ondersteunt (requirement RQ26) . Markdown is een lightweight markup taal die gebruikt kan worden om plain-text documenten vorm te geven. Het is één van de populairste markup talen ter wereld. (Markdown Guide, n.d.)

```
// Parse to markdown using regex
const toHTML = this.notesArea.innerText
  .replace('//n', '<br/>')
  .replace(/^#### (.*$)/gim, '<h4>$1</h4>')
  .replace(/^### (.*$)/gim, '<h3>$1</h3>')
  .replace(/^## (.*$)/gim, '<h2>$1</h2>')
  .replace(/^# (.*$)/gim, '<h1>$1</h1>')
  .replace(/\*\*(.*)\*\*/gim, '<b>$1</b>')
  .replace(/\~\~(.*)\~\~/gim, '<s>$1</s>')
  .replace(/\*(.*)\*/gim, '<i>$1</i>');

console.log(toHTML)

this.markdownField.innerHTML = toHTML.trim();
```

Figuur 17: Markdown parser met regular expressions in TypeScript in het Angular prototype

Markdown met regular expressions

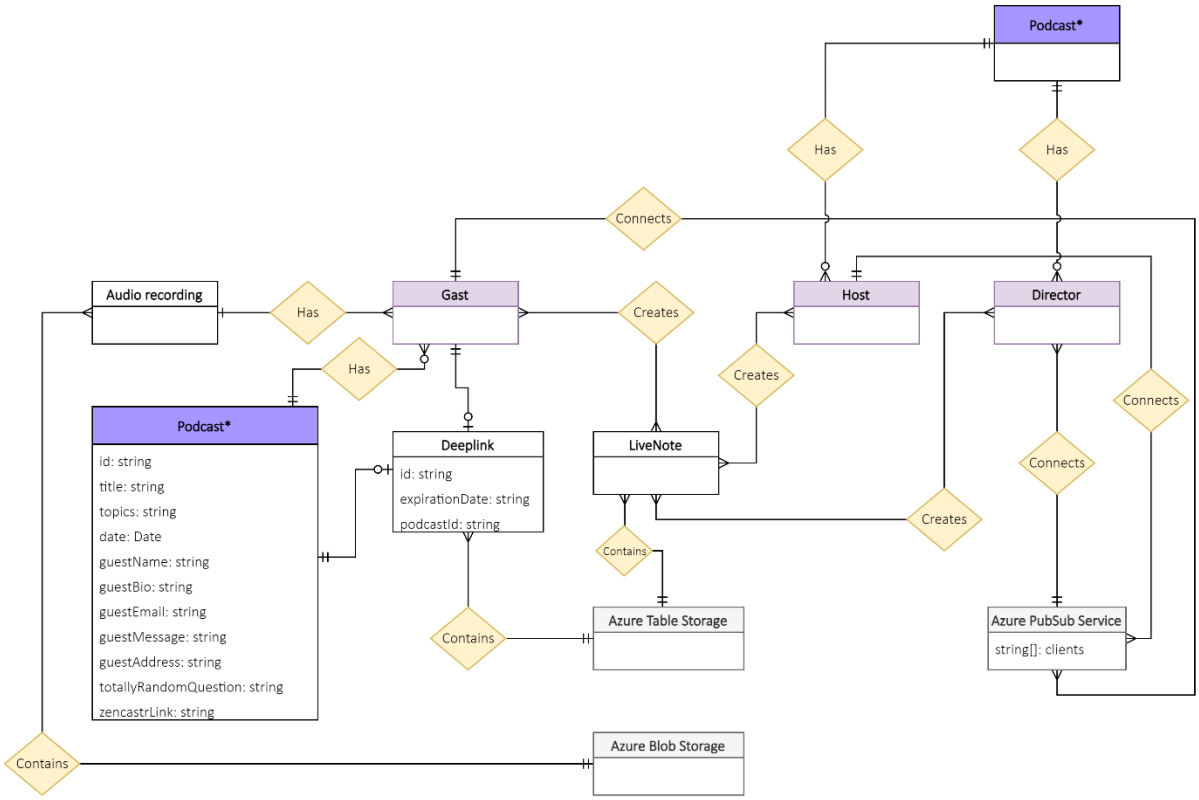
De Live Notes omgeving in de applicatie ondersteunt ook markdown. Door gebruik te maken van regular expressions worden de notities automatisch omgezet naar markdown in een apart tekstveld. Een regular expression is een combinatie van karakters waar patronen mee gevonden kunnen worden. (Hope, 2022)

Omdat markdown een eigen syntax gebruikt (Markdown Guide, n.d.), worden regular expressions gebruikt om deze syntax te vinden in een notitie.

Zie Figuur 31 in de bijlagen voor een schermafbeelding van de Live Notes pagina.

10.9 Entity Relation Diagram

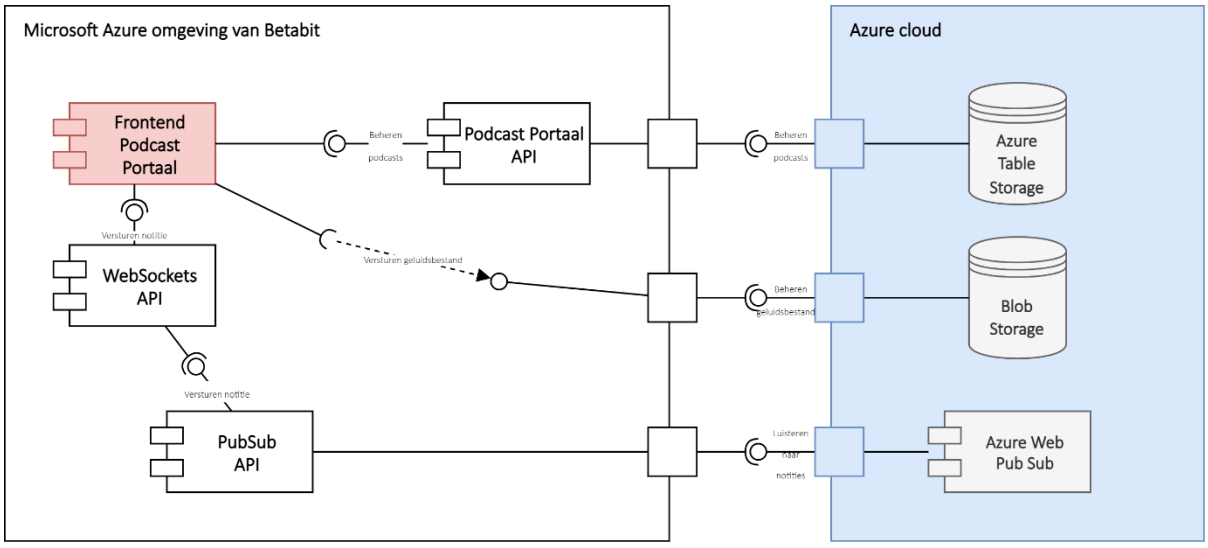
Het portaal heeft een aantal entiteiten met onderlinge relaties. Zo maakt bijvoorbeeld de gast gebruik van een deeplink die opgeslagen staat in een database. De gast gebruikt deze link om in te kunnen loggen. Verder kan het portaal podcasts ophalen uit de database als deze er zijn.



Figuur 18: Entity Relation Diagram waarin te zien is hoe verschillende entiteiten met elkaar communiceren en wat hun relaties zijn

10.10 Component diagram

Om de relaties tussen alle onderdelen van de applicatie weer te geven is het volgende component diagram opgesteld.



11. Prototypes

Welk framework past het beste bij de ontwikkeling van de applicatie op basis van de requirements van de product owner?

De applicatie wordt gebouwd in drie verschillende frameworks. Eerder was het idee om de applicatie in vier frameworks te bouwen, maar door tijdgebrek en een tekort aan diepgang is gekozen voor drie frameworks. De prototypes zijn gebouwd in:

- React met TypeScript
- Angular met TypeScript
- Blazor met C# [en JavaScript interop]

Het Vue prototype is te komen vervallen om meer ruimte te maken voor het Blazor prototype. Omdat er al twee frameworks gebruikt zijn met TypeScript en het interessant is om een C# framework te gebruiken, is gekozen om deze te laten vallen.

Als eerste werd het prototype in React ontwikkeld. React is een zeer populaire library die wordt ontwikkeld door Meta . Sinds 2013 wordt React uitgegeven als open source (Hámori, 2022) en is het alsmaar blijven groeien tot één van de populairste frontend libraries ter wereld. (Valishery, 2022)

Na een aantal weken is de ontwikkeling van het protoype in Angular van start gegaan. Angular is een veelgebruikt frontend framework. In 2022 stond Angular op de vijfde plek in de top tien meest gebruikte web frameworks (Salnik, 2022).

Tot slot is uiteindelijk ook de ontwikkeling van het Blazor prototype begonnen. Blazor is een framework dat wordt ontwikkeld door Microsoft. Het maakt gebruik van C# in de frontend in plaats van JavaScript of TypeScript. Wel biedt het de mogelijkheid om JavaScript te gebruiken met behulp van JavaScript interop. Hiermee kunnen JavaScript functies aangeroepen worden vanuit .NET methodes en andersom. (guardrex, thiennn, & diberry, 2023)

11.1 Beoordelingscriteria voor prototypes

Omdat veel requirements van de product owner op dezelfde manier geïmplementeerd worden in de drie prototypes, wordt er vooral gekeken naar de grote verschillen per prototype. Bijvoorbeeld is de implementatie voor de WebSocket verbinding, het uploaden van een geluidsbestand en het inloggen met Azure Active Directory in alle drie prototypes gelijk. Daarom wordt er vooral gekeken naar de volgende criteria.

- De implementatie van forms om podcasts aan te maken (40 punten)
  - Ingebouwde elementen bij het framework
  - Input validation
  - Workload: hoeveel werk is het om een form te bouwen
  - Omzetten invoervelden naar een object
- Dynamisch tonen van gegevens (podcasts uit de backend, binnenkomende notities) (10 punten)
  - De implementatie van (two-way) data binding
- Opsturen en ophalen data uit de backend (30 punten)
  - Structuur van API calls in de frontend
  - Omgang met data: hoe wordt data afkomstig van een externe bron gebruikt

Notitie: Dit onderzoek is een samenvatting van het Frontend Prototypes onderzoek.

11.2 Puntentoekenning

Ieder onderdeel wordt in een matrix beoordeeld per framework. Hier wordt een kleur aan toegekend. De frameworks kunnen op basis van deze kleur punten verdienen.

De puntentoekenning wordt gebaseerd op het aantal beschikbare opties, de implementatie hiervan, hoe uitgebreid de implementatie is en hoe ik het implementeren zelf heb ervaren. De prototypes worden alle drie op dezelfde manier ontwikkeld. Ze gebruiken dezelfde styling en dezelfde routes.

11.3 Scorematrix

De prototypes kunnen punten verdienen op basis van hun implementatie van een requirement.

	0 punten
	3 punten
	6 punten
	8 punten
	10 punten

Tabel 7: Legenda voor het kleurenschema van het puntensysteem

De puntenmatrix somt alle criteriapunten op waar het prototype op beoordeeld wordt. Er wordt eerst objectief gekeken naar de implementatie van ieder onderdeel. Later wordt er ook gekeken naar mijn eigen meningen en hoe ik de ontwikkeling van de prototypes heb ervaren.

Een puntenmatrix kan er als volgt uitzien:

	Prototype 1	**Prototype 2**	**Prototype 3**
Criterium 1	Uitstekend **(10)**	Goed **(8)**	Gemiddeld **(6)**
Criterium 2	Slecht **(0)**	Uitstekend **(10)**	Matig **(3)**
Punten toegekend	10 punten	18 punten	9 punten

Figuur 19: Voorbeeld van een scorematrix met een aantal criteria per framework

11.4 Opbouw van de prototypes

Alle prototypes worden op dezelfde manier gebouwd. Eerst wordt er gekeken naar de conventionele manier waarop een applicatie gebouwd wordt met een framework. Op deze manier is het makkelijker om van start te gaan en zijn de code en structuur leesbaarder voor de volgende ontwikkelaar. Hiervoor wordt de officiële documentatie van ieder framework/library geraadpleegd.

Vervolgens worden alle componenten op de volgende volgorde gebouwd:

- No Login
- Homepage voor hosts
- Homepage voor gasten
- Podcast Composition pagina
- Live Notes pagina

Zie Figuur 12: Navigatie tussen "routes" en welke relaties pagina's tussen elkaar hebben voor gasten en hosts om een overzicht van alle routes in te zien.

11.4.1 Single page application (SPA)

De applicatie wordt een single page application (SPA), wat inhoudt dat het gebruik gaat maken van routes in plaats van losse pagina's volgens een multiple page application (MPA).

Routes zijn pagina's die aan het begin van het bezoeken van een website ingeladen worden. Door de pagina's vooraf in te laden hoeft alle markup en styling maar één keer opgehaald te worden in plaats van iedere keer na

het wisselen van pagina (Neoteric, 2016). Het ontwikkelen van een SPA in plaats van een MPA was ook een requirement van de product owner omdat een SPA veel sneller zou zijn dan een MPA omdat er maar weinig routes nodig zijn.

11.5 Persoonlijke voorkeur

Mijn persoonlijke voorkeur voor de ontwikkeling van het Podcast Portaal gaat uit naar het *Angular* framework. Ik vind de workflow en de hoeveelheid opties die Angular te bieden heeft op het gebied van forms, tests, input validation en data-binding zeer goed. Ik heb de ontwikkeling van alle drie prototypes als enorm leerzaam ervaren. Het was zeer interessant om het Podcast Portaal op drie compleet verschillende manieren te ontwikkelen.

Ondanks dat alle drie frameworks prima toepasbaar zijn voor de ontwikkeling van de applicatie, gaat mijn persoonlijke voorkeur toch uit naar Angular.

11.6 Uitslagen per criteriumpunt

Aan de hand van de vooraf bepaalde criteria heeft ieder prototype een aantal punten ontvangen. Deze punten zijn toegekend op basis van hun implementatie van het requirement en in hoeverre dit in verhouding staat met de andere prototypes.

	React	Angular	Blazor
Forms implementatie	23 / 40	34 / 40	29 / 40
Dynamisch tonen van gegevens uit de backend	6 / 10	8 / 10	10 / 10
Verbinden met de backend	24 / 30	26 / 30	28 / 30
Totaal	53 / 80	68 / 80	67 / 80

Figuur 20: Puntentoekenning uitslag van de prototypes

11.7 Conclusie

Angular komt nog maar net als beste uit de criteria naar voren. Blazor en React hebben elk zeer sterke kanten. Voor de ontwikkeling van het Podcast Portaal biedt Angular de sterkste kanten naar mijn mening. Dit komt doordat Angular's implementatie van het dynamisch tonen van gegevens (podcasts) in de frontend afkomstig uit de backend zeer uitgebreid is.

React, Angular en Blazor zijn alle drie zeer goede frameworks/libraries om op een goede manier een applicatie te bouwen voor Betabit. Uit mijn onderzoeken komt Angular vaak goed naar voren, maar dat wil niet zeggen dat Blazor en React niet gebruikt kunnen worden. Alle drie prototypes hebben sterke en zwakke punten getoond. In dit geval komt Angular goed uit de test voor de ontwikkeling van het Podcast Portaal.

12. Tests

Om ervoor te zorgen dat de opgeleverde code naar behoren werkt is er een aantal test cases opgesteld om de kwaliteit van de applicatie te waarborgen. De code is per framework getest met het test-framework dat het beste aansluit bij het framework.

Gebruikte testframeworks

Frontend framework	Test framework
React	Jest, Puppeteer
Angular	Jasmine + Karma
Blazor	bUnit, PlayWright for .NET

Voor ieder prototype is gestreefd naar 100% code coverage om er zeker van te zijn dat alle code uiteindelijk goed werkt. Ieder framework heeft een eigen implementatie voor het uitvoeren en schrijven van tests.

12.1 Teststrategie

De applicatie wordt met behulp van verschillende test-frameworks getest. Er worden functionele en niet-functionele tests geschreven om verschillende onderdelen van de applicatie te testen. De prototypes worden getest met **unit tests**. Verder wordt er ook gebruik gemaakt van externa applicaties om **e2e tests** uit te voeren in de frontend.

12.2 Code coverage

Code coverage in React is toegepast met behulp van de nyc (Istanbul) library. Deze library zit ingebouwd bij het Jest framework waar React automatisch gebruik van maakt. Met code coverage kan gecontroleerd worden welke onderdelen van een component getest worden en welke nog niet. Code coverage heeft geen invloed op de kwaliteit van de tests, maar biedt wel inzicht over de hoeveelheid code die nog getest moet worden.

12.3 Tests in React

Voor het React prototype heb ik gebruik gemaakt van het Jest test framework. Jest is een test-framework ontwikkeld door Facebook. Het wordt gebruikt voor het testen van JavaScript/TypeScript applicaties. Het is één van de populairste test-frameworks voor JavaScript applicaties.

Tests in React worden in één bestand, namelijk

```
// Attempt to open hamburger menu and succeed when side menu changes class to expanded
test('Try to open hamburger menu', () => {
    render(<App />);

    const hamburgerButton = screen.getByTestId("hamburgerButton");
    const sideMenu = screen.getByTestId("sideMenu");

    expect(hamburgerButton).toBeInTheDocument();
    expect(sideMenu).toBeInTheDocument();

    // Click hamburger icon to open side menu
    fireEvent.click(hamburgerButton);
    expect(sideMenu).toHaveClass("Hamburger-menu");

    // Fire click event again to check whether menu is collapsed again
    fireEvent.click(hamburgerButton);
    expect(sideMenu).toHaveClass("Hamburger-menu Hide");
});
```

Figuur 21: Een test geschreven in het Jest framework voor het React prototype

12.3.1 Linting met ESLint

Om te controleren of de code ook volgens de standaard van het huidige framework is geschreven, wordt gebruik gemaakt van het **ESLint node package**. Hiervoor gebruik ik een ESLint plugin speciaal voor React. Vóór iedere push naar de repository, voer ik een command uit waarmee ik een rapport terugkrijg waarin te zien is of de code nog voldoet aan de standaard van React.

Voor de configuratie van de ESLint plugin gebruik ik een preset met daarin de aanbevolen standaard voor React. Deze is op de repository (Harband et al, n.d.) van de plugin te vinden.

12.3.2 E2E tests met Jest en Puppeteer

Puppeteer is een Node.js library die gebruikt kan worden om de applicatie te testen vanuit het perspectief van een gebruiker. Het kan gebruikt worden om onder meer de handelingen die een gebruiker zou kunnen uitvoeren te automatiseren.

Puppeteer tests worden op ongeveer dezelfde manier als Jest testbestanden opgebouwd. Puppeteer kan een nieuwe browser instantie openen en vervolgens naar de frontend navigeren. Vervolgens kunnen er instructies toegevoegd worden aan een testbestand. In dit testbestand staan de handelingen die een gebruiker zou kunnen verrichten in de applicatie.

12.4 Tests in Angular

Testen in Angular gaat op een soortgelijke manier als in React. Het gebruikt immers dezelfde programmeertaal met dezelfde syntax. Angular maakt standaard gebruik van een combinatie van Jasmine en Karma. Jasmine is een populair test-framework dat wordt gebruikt om JavaScript en TypeScript te testen. Karma wordt gebruikt om de code te kunnen testen in verschillende web-browsers. Het resultaat van het uitvoeren van een test wordt in een browser getoond.

```typescript
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { By } from '@angular/platform-browser';
import { BetatalksFooterComponent } from './betatalks-footer.component';

describe('BetatalksFooterComponent', () => {
  let component: BetatalksFooterComponent;
  let fixture: ComponentFixture<BetatalksFooterComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [BetatalksFooterComponent]
    })
      .compileComponents();

    fixture = TestBed.createComponent(BetatalksFooterComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('Test whether footer consists of two links', () => {
    const linkToBetabit = fixture.debugElement.query(By.css('#linkToBetabit'));
    expect(linkToBetabit).not.toBeNull();
  });

  it('Test whether link to Betatalks exists', () => {
    const linkToBetatalks = fixture.debugElement.query(By.css('#linkToBetatalks'));
    expect(linkToBetatalks).not.toBeNull();
  });
});
```

Figuur 22: Voorbeeld van tests in Angular. Angular maakt gebruik van Jasmine en Karma.

Bij het uitvoeren van het test command in Angular, gaat het framework op zoek naar bestanden met het `SPEC.TS` formaat.

Deze bestanden worden gebruikt om de code te testen. De uitkomst hiervan is te zien in een webbrowser die automatisch geopend wordt door Jasmine en Karma.

Ieder component in Angular wordt gegenereerd met een eigen `SPECT.TS` bestand. Hiermee kunnen alle componenten afzonderlijk van elkaar getest worden.

Bij het uitvoeren van de tests in Angular kan er ook een aantal flags meegegeven worden, zoals `–code-coverage`. Hier wordt automatisch een code coverage rapport gegeneerd in HTML. Zie Fout! Verwijzingsbron niet gevonden. voor een voorbeeld van het r apport van Angular over de code coverage van het prototype.

12.5 Tests in Blazor

Voor het Blazor prototype worden ook unit tests en e2e tests geschreven. Bij het unit testen wordt gebruik gemaakt van `bUnit`. Hierbij worden componenten gerendered, de output en state van componenten getest, triggers en event handlers getest en assertions gebruikt om verwachte output te testen.

Voor het e2e testen wordt gebruik gemaakt van `PlayWright for .NET`. Dit is een framework dat gebruikt kan worden om Razor componenten in de browser te testen. Hierbij kan de gebruikersinteractie gesimuleerd worden om te kijken of de UI nog naar behoren werkt.

bUnit wordt niet officieel ondersteund of ontwikkeld door Microsoft. bUnit gebruikt echter wel dezelfde syntax als XUnit, een populair test framework voor C# programma's.

12.6 Verschillen tussen frameworks

Er is een aantal verschillen tussen het testen van de code in de drie frameworks.

	React	Angular	Blazor
Framework	Jest, Puppeteer	Jasmine, Karma	bUnit, PlayWright
Structuur	Komt met één testbestand, maar alle bestanden met de .test.tsx suffix worden getest.	Aanbevolen om een testbestand per component en service te maken	Aanbevolen om een testbestand per component en service te maken
Programmeertaal	JavaScript/TypeScript	JavaScript/TypeScript	C#/Razor syntax
Ingebouwde code coverage	Ja*	Ja	Nee
Linter ondersteuning	Ja	Ja	Ja

Figuur 23: Verschillen tussen de drie frameworks op het gebied van testen.

13. Conclusie

13.1 Requirements

De meeste requirements zijn geïmplementeerd in het Podcast Portaal. Een paar requirements zijn helaas niet volledig ontwikkeld en ontbreken nog in de eindapplicatie.

Behaald
Geïmplementeerd met commentaar
Niet behaald

Tabel 8: Legenda van het resultaat van de requirements

Req	Omschrijving	FR/NFR*)	Prioriteit
RQ01	De host moet de naam van de gast(en) kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ02	De host moet de datum en de tijd van de podcastopname kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ03	De host moet de topics van de podcast kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ04	De host moet de biografie van de gast(en) kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ05	De host moet commentaar kunnen achterlaten voor de gast van de huidige podcast via de applicatie tijdens het opstellen van een podcast.	FR	M
RQ06	De host moet een titel kunnen toevoegen via de applicatie aan de huidige podcast.	FR	M
RQ07	De host moet het e-mailadres van de gast kunnen toevoegen via de applicatie.	FR	M
RQ08	De host moet een kopieerbare link naar de Zencastr omgeving van Betabit kunnen toevoegen aan de huidige podcast.	FR	M
RQ09	De host moet een TRQ (Totally Random Question) kunnen toevoegen aan de huidige podcast via de applicatie.	FR	M
RQ10	De hosts moet de gegevens van een bestaande podcast kunnen aanpassen	FR	M
RQ11	De host moet een bestaande podcast kunnen annuleren	FR	M
RQ12	De host en regisseur moeten kunnen inloggen met hun Betabit account.	FR	M
RQ13	De host moet een eigen notitieruimte kunnen gebruiken die niet zichtbaar is voor de gasten in het podcastportaal.	FR	M
RQ14	De gasten moeten hun geluidsopname kunnen uploaden via het podcastportaal naar de database van de applicatie	FR	M
RQ15	De gasten moeten hun woonadres kunnen achterlaten op het podcastportaal.	FR	M
RQ16	De gasten moeten hun eigen biografie kunnen aanpassen.	FR	M
RQ17	De gasten moet kunnen inloggen met een persoonlijke deeplink zodat zij tijdelijk en zonder account toegang krijgen tot de applicatie.	FR	M
RQ18	De applicatie moet de opname van de gasten alleen accepteren in het .wav formaat zodat Betabit de hoogste audiokwaliteit kan ontvangen.	NFR	M
RQ19	De applicatie moet de geldigheid van een deeplink kunnen verifiëren om te controleren of de gast toegang hoort te krijgen tot de applicatie.	FR	M
RQ20	De applicatie moet een eigen huisstijl gebruiken en niet die van Betabit om de applicatie open source te houden.	NFR	M
RQ21	De applicatie moet een single page application worden om de laadtijden van de applicatie te verlagen	NFR	M
RQ22	De Live Notities omgeving moet serverless notities kunnen uitwisselen.	FR	S
RQ23	De hosts moeten middels een serverless socketverbinding notities kunnen uitwisselen met de regisseur van de podcast in de applicatie die niet te zien zijn voor de gasten	FR	S
RQ24	De gasten moeten middels een serverless socketverbinding notities kunnen uitwisselen met de hosts van de podcast in de applicatie	FR	S

RQ25	De applicatie moet automatisch een e-mail versturen naar de gast met daarin commentaar voor de gast van de hosts, de datum en tijd van de podcast, de titel, onderwerpen en deeplink van de podcast nadat een podcast aangemaakt wordt.	FR	C
RQ26	De live notities moeten automatisch geparsed worden naar markdown zodat de notities makkelijker te lezen zijn.	FR	C
RQ27	De hosts moeten vanuit de regie een :AFRONDEN signaal kunnen ontvangen wanneer de podcast afgebouwd gaat worden	FR	C
RQ28	De applicatie moet meerdere talen ondersteunen omdat er ook veel gasten uit verschillende landen worden uitgenodigd.	NFR	W

Tabel 9: Requirements van de product owner waarin wordt aangegeven welke requirements zijn behaald en welke niet

Commentaar:

Het is mogelijk voor de hosts en gasten om gebruik te maken van de live notitie omgeving van de applicatie. Echter doordat de functionaliteit voor de gast ontbreekt om in te loggen met de deeplink is het nog niet mogelijk om de notities van de hosts en regie af te schermen voor de gasten.

13.2 Competenties

In het Plan van Aanpak is een aantal HBO competenties ingeschat.

	Manage/Control	Analyseren	Adviseren	Ontwerpen	Realiseren
Gebruikersinteractie	2	2	2	2	2
Organisatieprocessen	2	1	1	2	2
Infrastructuur	1	1	1	1	1
Software	2	3	2	3	3
Hardware interfacing	1	1	1	1	1

Figuur 24: Mijn persoonlijke competenties op basis van het HBO-I model

Toelichtingen

Softwareanalyse op niveau 3: Op basis van de eisen van de product owner is er een lijst met requirements opgesteld. Deze lijst is vervolgens overlegd en uiteindelijk goedgekeurd door de product owner. Er is een onderzoek gedaan naar de verschillende mogelijkheden voor de ontwikkeling van de applicatie.

Softwareontwerp op niveau 3: De applicatie is ontworpen conform standaard methodiek zoals het opstellen van een klasse diagram, ERD en een componentdiagram. Er is onderzoek gedaan naar de werking van ieder framework om de code op een logische manier op te stellen zodat hier gemakkelijk verder mee gewerkt kan worden. Ook wordt er beschreven waarom bepaalde keuzes gemaakt zijn tijdens de ontwikkeling hiervan en op wat voor manier de applicatie is opgebouwd.

Softwarerealisatie op niveau 3: Er is op een logische manier omgegaan met het schrijven van de code voor de applicatie. Alle code is voor het samenvoegen op de hoofdbranch eerst met een pull request nagekeken. De code is voorzien van commentaar over gemaakte keuzes en werking hiervan. Er zijn unit en **e2e** tests geschreven om de kwaliteit van de applicatie en code te waarborgen. Er is een live notities omgeving ontwikkeld, een authenticatie-implementatie met OAuth 2.0

13.3 Eindresultaat

Ik ben zelf zeer tevreden over het eindresultaat. Het was interessant om nieuwe frameworks te leren en een volledige applicatie te bouwen met dynamische inhoud. De applicatie stelt de hosts van Betabit in staat om podcasts in te plannen en internationaal te communiceren met andere gebruikers van de applicatie, iets waar ik zelf nog maar weinig ervaring mee heb.

13.4 Adviesrapport

Naar mijn mening is Angular de beste optie voor Betabit om de ontwikkeling van het Podcast Portaal voort te zetten. De ontwikkeling van het Angular prototype heb ik als gestroomlijnd en efficiënt ervaren. Niet alleen biedt Angular uitstekende documentatie, voor een applicatie als het Podcast Portaal biedt Angular ook two-way data binding, static typing met TypeScript, een goed testplatform en een grote community van developers.

Ik raad hierbij Betabit aan om de ontwikkeling van het Podcast Portaal voort te zetten in het Angular framework als hier nog features aan toegevoegd of gewijzigd gaan worden. Ik raad Angular ook aan omdat het de naar mijn mening beste implementatie heeft van het vertonen van gegevens uit de backend in de frontend. Uit mijn eigen onderzoek blijkt ook dat Angular goed aansluit bij de wensen van een aantal ontwikkelaars bij Betabit. Daarom raad ik Angular aan bij Betabit.

13.5 Verbeteringen

Om aan te tonen wat er allemaal verbeterd kan worden in het Podcast Portaal is er een adviesrapport opgesteld. Dit rapport bestaat uit onderdelen die nog niet (goed) geïmplementeerd zijn, nog niet af zijn en wat wellicht interessant zou zijn voor Betabit.

	Verbeteringsadvies
	Niet geïmplementeerd
	Idee

- ✍ **Verbeteren van de Live Notes functionaliteit**
    - De Live Notes omgeving heeft momenteel een probleem. Het is mogelijk om met meerdere clients tegelijk aan één bestand te werken. Een onderdeel dat nog niet goed werkt is het synchroniseren van de clients. Op dit moment wordt er wanneer een client aan het typen is, gewacht met het versturen van gegevens om prestatie gerelateerde redenen. Dit kan ervoor zorgen dat de wijzigingen die andere clients gemaakt hebben overschreven worden. Dit gebeurt niet wanneer er om de beurt getypt wordt.
    - Er kan worden gekeken naar een oplossing om alleen de **gewijzigde** sectie door te sturen naar de Azure Web PubSub service. Dit zorgt ervoor dat alleen stukken tekst waar een gebruiker mee bezig is geweest aangepast wordt. Dit levert ook verbeterde prestaties op omdat er minder gegevens verzonden en ontvangen hoeven te worden.
- ✍ **Positie van de cursor in het notitieveld verbeteren**
    - Op dit moment wordt de positie van de cursor in het notitieveld berekend op een inconsistente manier. De code die de positie berekent van de cursor in het document houdt bijvoorbeeld geen rekening met breaklines (witregels) en kan daardoor verkeerd worden vertoond bij andere clients.
    - De reden dat de cursor positie op deze manier berekent moet worden is omdat er gebruik gemaakt wordt van een `contenteditable DIV`. Dit is een HTML element waar de gebruiker de inhoud van kan wijzigen.
    - Omdat dit element anders werkt dan het ingebouwde `TEXTAREA` element, is het niet mogelijk om direct de positie van de cursor op te vragen. Het was helaas niet mogelijk om het TEXTAREA element te gebruiken omdat er veel bugs optraden tijdens het gezamenlijk aanmaken van notities met meerdere clients. De cursor van een client springt bijvoorbeeld naar het begin van het tekstveld wanneer een andere client iets typt.
    - Wellicht is het interessant om het `DIV` element te vervangen door een `CANVAS` element en een eigen implementatie te bouwen van het tekstveld.
- ✍ **Het uploaden van bestanden wordt in de frontend uitgevoerd**
    - De huidige versie van het Podcast Portaal stelt gebruikers in staat om geluidsbestanden te uploaden. Dit proces wordt echter in de frontend uitgevoerd. In de huidige versie zijn dan ook SAS tokens terug te vinden om toegang te krijgen tot een lokale Blob Storage Emulator.
    - Als de applicatie live gedeployed gaat worden waarbij gebruik gemaakt gaat worden van een Blob Storage opslagdienst op het Azure platform, dan moet er om serieuze beveiligingsrisico's te

vermijden een endpoint komen in de API dat de authenticatie van de gebruiker regelt. Door deze SAS tokens in de frontend op te slaan, zouden alle gebruikers er in theorie bij kunnen. Ook is hier een URL naar de opslagomgeving en de opslagcontainers te zien die een gebruiker eigenlijk niet hoort te zien.

- **Het betrekken van het publiek tijdens een uitzending**
    - Het is misschien interessant om luisteraars van de podcast mee te laten communiceren tijdens een uitzending. Gebruikers kunnen bijvoorbeeld meedoen met een soort chat die ook met de bestaande WebSocket en Web PubSub implementatie zou kunnen werken.
- Niet geïmplementeerd: deeplink functionaliteit
    - Het is voor de gasten momenteel niet mogelijk om in te loggen op het Podcast Portaal. Dit komt doordat er geen functionaliteit is ingebouwd voor het genereren, ophalen en verifiëren van een deeplink.
    - Ik wil graag aan Betabit aanraden om in de Azure Table Storage opslag alle deeplinks op te slaan die worden gegenereerd voor alle gasten. Hierbij moet een uniek ID, de naam van de gast, het ID van de podcast waar ze voor uitgenodigd zijn en een vervaldatum opgeslagen worden. In de Podcast Portaal API kan eventueel ook gelijk gecontroleerd worden of een deeplink nog geldig is.
        - Ik zou ook aanraden om al deze gegevens te encrypten in de database. Omdat ik mij met de frontend bezig houd weet ik niet of dit al gebeurt, maar zo niet, dan zou ik wel aanraden om deze gegevens onleesbaar te maken om eventuele beveiligingsrisico's te vermijden.
- Niet afgekregen: volledige uitwerking Angular applicatie
    - Het is helaas niet gelukt Betabit een uitgewerkte versie van de applicatie te geven. Ik wilde na het onderzoek naar de frotend frameworks afgerond te hebben graag aan de slag met de verdere ontwikkeling van de applicatie in Angular. Helaas door tijdgebrek is dit niet helemaal gelukt. Ook doordat de backend niet volledig is uitgewerkt mist een aantal features.

14. Reflectie

Vragen om hulp

In de eerste twee maanden van mijn afstudeerstage heb ik het moeilijk gevonden om vragen te stellen aan mijn begeleider. Ik wist niet precies wat ik moest vragen en vond het lastig om te vragen om hulp wanneer ik dat nodig had. Bijvoorbeeld kwam ik af en toe vast te zitten met taken die ik graag zelf wilde oplossen en daarbij niet om hulp wilde vragen omdat ik er waarschijnlijk wel uit zou komen.

Mijn begeleider heeft aangegeven dat ik vragen moest gaan stellen als ik ergens te lang mee vast zou zitten. Met "te lang" hebben we uiteindelijk besloten niet langer dan een uur te wachten met vragen na tegen een probleem aangelopen te zijn. Ook heb ik gevraagd aan mijn begeleider om af en toe even een moment in te plannen om samen even over het verloop van de stage te praten. Hier kwamen wat interessante gesprekken uit voort, mede over hoe mijn begeleider mijn inzet als stagiair ervaart en actiepunten die ik zou kunnen ondernemen om te verbeteren.

Gelukkig is het voor mij makkelijker geworden om mijn begeleider om hulp te vragen wanneer ik dat nodig had. Voor mij was het een grote drempel om mijn begeleider niet van zijn werk af te leiden en speelde mijn eigen instelling over alles zelf willen oplossen ook een rol hierin. Ik heb geleerd dat ik vragen kan stellen zonder dat iemand daar direct last van heeft. Komt het een keertje niet uit? Dan kom ik een andere keer terug. Niet alleen was dit een belangrijk leermoment tijdens de stage, maar ook voor toekomstige ervaringen in het bedrijfsleven. Het is voor een team juist makkelijker als hen wordt gevraagd om hulp.

Planning, ontwikkeling van features en de onafgemaakte backend

Het project is helaas niet volledig volgens de planning gegaan. De planning was opgebouwd op basis van het onderzoeken en het ontwikkelen van prototypes over drie frontend frameworks en een library. Het Vue framework is in de derde maand van de stage laten vallen met als doel de overige drie frameworks/libraries uitgebreider te kunnen onderzoeken. Ook heeft de ontwikkeling van sommige features wat langer geduurd dan de bedoeling was doordat er nog veel onderdelen nieuw waren voor mij en de backend nog niet helemaal klaar was.

Door het Vue framework niet langer uit te voeren heb ik de planning gewijzigd. De duur van de onderzoeken naar ieder framework is verlengd met ongeveer twee weken doordat hier extra tijd voor was gekomen. Op deze manier kon er meer diepgang in het onderzoek verwerkt worden door er meer tijd aan te besteden. Verder heeft de ontwikkeling van de Live Notities omgeving en het uploaden van bestanden naar Azure Blob Storage langer geduurd dan ik had ingepland. Het werken met WebSockets in TypeScript en C# was nieuw voor mij, maar door het volgen van Microsofts officiële handleiding en het kort onderzoeken naar de theorie achter een socketverbinding is het uiteindelijk toch gelukt om de omgeving volledig in te bouwen.

Verder had ik voor de stage nog nooit gewerkt met Microsoft Azure. Ik wist nog helemaal niet hoe ik bestanden kon uploaden naar een opslagomgeving op het Azure platform. Ik heb mijn begeleider en een aantal collega's op het kantoor gevraagd of zij enige ervaring hadden met Azure Blob Storage. Er bleek gelukkig veel ervaring te zijn met Blob Storage. Na het hebben gekregen van een korte uitleg over hoe Blob Storage precies werkt en hoe het gebruikt kan worden, is ook deze feature gelukkig volledig geïmplementeerd.

Ook kwam de ontwikkelaar van de backend in de problemen met de planning. Het was tot de vierde maand van de stage niet mogelijk om de API aan te roepen. Om dit op te lossen heb ik tijdelijk gebruik gemaakt van een **mock API** met daarin hetzelfde opslagschema dat we gebruikt hebben voor de daadwerkelijke API. Op deze manier kon ik de applicatie wel testen en hoefde ik alleen de URL van de API aan te passen naar de API van het Podcast Portaal.

Uiteindelijk is de ontwikkelaar van de backend gestopt met zijn stage. Hierdoor was een groot aantal features van de applicatie nog niet beschikbaar. Het aanmaken en verifiëren van deeplinks (RQ19) was nog niet mogelijk. Verder was er nog geen endpoint voor het opslaan van geluidsopnames (RQ14). Ook was het opslaan en aanpassen van podcasts en gastgegevens door niet-geïmplementeerde CORS-policies in de backend niet mogelijk.

Nadat de backend ontwikkelaar was gestopt met zijn stage, is overlegd wie de taken in de backend zou gaan proberen af te maken. Mijn begeleider, de begeleider van de gestopte stagiair en een collega in Apeldoorn hebben een aantal features geprobeerd op te pakken. Zo kijken mijn begeleider en een collega naar het genereren en gebruiken van de deeplink functionaliteit.

Ik heb zelf besloten om de Azure Blob Storage functionaliteit te ontwikkelen. Omdat er geen endpoint was voor het uploaden van bestanden naar de backend, wordt het versturen van een bestand momenteel direct vanuit de frontend geregeld. In het adviesrapport aan Betabit raad ik dan ook aan dat zij een endpoint aanmaken in de API om via de API bestanden te uploaden.

Planning met documenten

Tijdens de stage ben ik zeer druk bezig geweest met het schrijven van mijn afstudeerverslag en het ontwikkelen van de prototypes. In de laatste weken ben ik ook druk geweest met het voorbereiden van een aantal presentaties op verschillende kantoren van Betabit. Hierdoor kwam ik al snel toch in tijdnood.

De volgende keer ga ik vragen of ik de presentaties op een ander datum zou kunnen doen in plaats van vóór de deadline van het verslag. Dit geeft mij meer tijd om de presentatie voor te bereiden en kan ik mij meer richten op het afstudeerverslag.

14.1 Ervaring bij Betabit

Ik heb mijn afstudeerstage bij Betabit als leuk en zeer leerzaam ervaren. Mijn hart gaat uit naar het ontwerpen van websites in de frontend dus het was erg interessant om een uitdagende opdracht in de frontend te kunnen uitvoeren. Ik heb veel geleerd van mijn stage bij Betabit. Niet alleen over het ontwikkelen van software, maar ook over mijzelf. Ik heb geleerd hoe ik beter kan plannen en dat ik vaker om hulp kan vragen.

Kortom, ik heb een zeer leerzame tijd gehad bij Betabit in Apeldoorn en ik hoop dat zij veel gebruik zullen maken van het Podcast Portaal.

Appendix A: Versiegeschiedenis

Versie	Opleverdatum	Aanpassingen
0.1	05/09/2022	Start afstudeerverslag
0.2	23/11/2022	Feedback begeleider toegepast
Concept	07/01/2023	Feedback toegepast, informatie aangevuld, veld-onderzoek gescheiden
Definitief verslag	22/01/2023	Uitwerken onderzoeken, uitleg afbakening project, aanvullen tests, product, projectorganisatie en adviesrapport.

Appendix B: Begrippenlijst

Actiepunten
Een lijst met handelingen op basis van problemen waar tegenaan is gelopen tijdens een sprint. Deze handelingen beschrijven hoe een probleem voortaan voorkomen kan worden. De actiepunten worden tijdens iedere retrospective weer naar voren gehaald om te beoordelen of deze ook gebruikt zijn.

ATP
Azure Talent Program: Een cursus waarin ontwikkelaars worden opgeleid tot echte Azure ontwikkelaars.

Azure
Azure is de officiële cloudomgeving van Microsoft. Het wordt ontwikkeld door Microsoft sinds 2008. Via dit platform kunnen verschillende soorten internetdiensten aangeboden worden, zoals het hosten van databases en webapplicaties.

Azure AD
Azure AD (Azure Active Directory) is de ingebouwde oplossing voor het beheren van identiteiten in Microsoft 365 en Azure.

Backlog
Een lijst met taken en bugs voor op het sprintbord. Deze taken worden voorafgaand aan de komende iteratie ingepland. De hoeveelheid wordt gebaseerd op de sprint velocity van de vorige iteratie.

Betatalks
De Betatalks podcast is een tweewekelijkse podcast over verschillende onderwerpen in de softwarewereld. De podcast wordt gehost door twee software developers van Betabit. Zij nodigen hier ook altijd gasten voor uit.

Branch
Een “tak” op de repository waar alleen code voor een specifieke feature in zit. Deze branch moet uiteindelijk samengevoegd worden met de rest van de code. Dit gebeurt alleen wanneer deze is goedgekeurd door andere teamleden.

Breadcrumbs
Een visueel hulpmiddel dat aangeeft waar de gebruiker zich bevindt op een website, bijvoorbeeld:
Home > Artikelen > Artikel 32. De breadcrumbs dienen ook als navigatiemiddel. De gebruiker kan klikken op de links in de breadcrumbs om terug te navigeren.

Code smell
Code smell is een verzamelnaam voor verschillende problemen die met code te maken hebben. Hier vallen bijvoorbeeld lange functies, duplicate code, ongebruikte code of eindeloze switch statements. (Code Smells, n.d.)

e2e
Het testen van een frontendapplicatie door handelingen van een gebruiker automatisch uit te laten voeren door een script.

IDE
Integrated Development Environment: een applicatie om software mee te bouwen.

JS
JavaScript - Een programmeertaal om onder meer webapplicaties mee te bouwen.

MOSCOW principe
MOSCOW is een manier om eisen van de opdrachtgever te prioriteren. De medeklinkers van MOSCOW vormen de verschillende prioriteiten voor iedere eis. M = Must have, S = Should have, C = Could have en W = Won’t have/Would be nice to have.

OAuth 2.0
Een open standaard voor autorisatie. Een manier van authentiseren via een derde partij.

Product owner
De opdrachtgever van het project. De product owner stelt wensen op, biedt feedback aan op het product tijdens de ontwikkeling en is verantwoordelijk voor de communicatie met de stakeholders binnen het project.

Pull request
Een verzoek aan andere teamleden om code te beoordelen. Wanneer deze code wordt goedgekeurd kan dit onderdeel samengevoegd worden met de rest van de code.

Refactor
Een aantal wijzigingen (in de code) om deze duidelijker, korter of simpeler te maken. Bijvoorbeeld een aantal naamswijzigingen van een paar variabelen of een bepaalde methode die korter had gekund.

Repository
Een online omgeving waar alle code en documentatie van het huidige project te vinden is.

Requirement
Een eis van de opdrachtgever voor het product. Deze wordt zo duidelijk mogelijk opgesteld zodat deze procesmatig en met bewijs afgevinkt kan worden.

Retrospective
Een moment aan het eind van iedere sprint waarin wordt gereflecteerd op het verloop van de sprint. Dit kan op veel verschillende manieren. Er wordt gekeken naar wat er goed ging, wat er minder goed ging en hoe dit opgelost kan worden tijdens de volgende sprint.

Scrum
De scrum methode is een Agile framework waarmee projecten op basis van iteraties ingepland kunnen worden. Er wordt gewerkt met verschillende rollen om projecten in sprints in te delen. Deze lengte van deze sprints verschilt per project.

Sprint velocity
Het aantal punten dat tijdens een sprint is “verbrand”. Iedere taak bestaat uit een aantal punten. Deze punten worden aan het einde van een sprint tijdens de retrospective geëvalueerd. Op basis hiervan kan de volgende sprint ingepland worden omdat dan duidelijk is hoeveel punten haalbaar zijn binnen het team.

TS
TypeScript - Een programmeertaal/wrapper gebaseerd op JavaScript die de fundamentele problemen, zoals het ontbreken van OOP en type safety oplost.

UI
User Interface, of de gebruikersinterface. Alles wat met het gebruik van een applicatie te maken heeft.

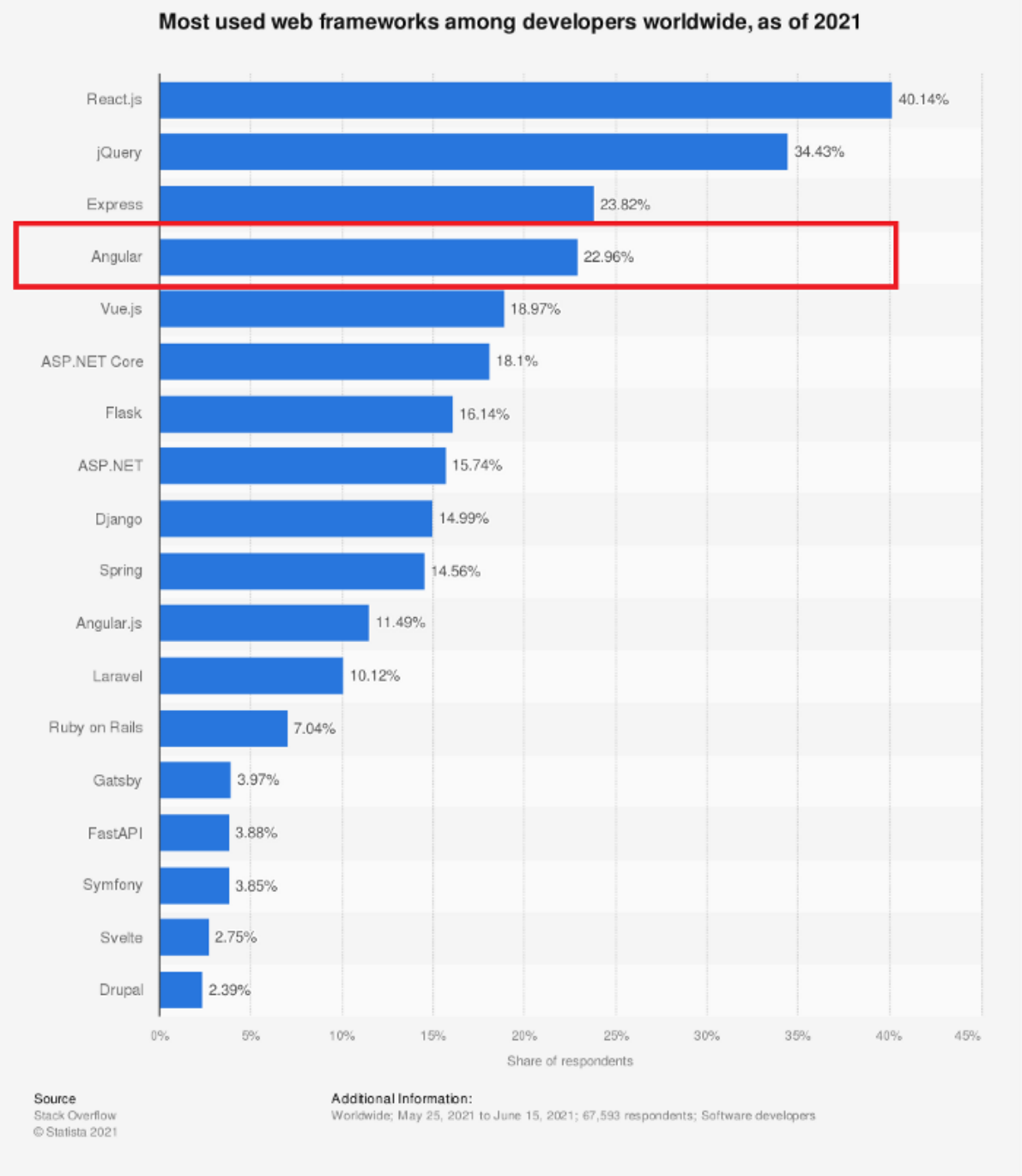
UX
User Experience of de gebruikerservaring. De manier waarop een gebruiker het gebruik van een applicatie ervaart.

Appendix C: Bijlagen

Figuur 25: Actiepunten na afloop van de eerste sprint van het project

```
Actiepunten

Retrospective 28 oktober 2022 - Einde van Sprint 1

        - Plannen van tevoren:
[x]         - Sprint velocity meenemen in planning.
[ ]         - Houden aan een vaste hoeveelheid story points.
[ ]         - Liever te weinig dan te veel. Dan kan er altijd nog wat werk bij.
        - Diepgang framework:
[ ]         - Meer verdiepen in conventionele werkwijzen per framework om te voorkomen
            dat ik verouderde werkwijzen toepas.
[x]         - Verdiepen in test framework dat bij dit framework past.
[x]         - Code netjes houden met separation of concerns UI van functioneel scheiden
[x]             - Te doen door code vaker te laten reviewen
        - Reviewen
[x]         - Pull requests klein houden.
[x]         - Vaker vragen om review en feedback om daadwerkelijk stories naar Done te slepen.
[x]         - Code laten reviewen door Luuk en Dinand
        - Daily Standup
[ ]           Daily standups organiseren van tevoren en daadwerkelijk te houden.
                - Problemen tijdens de vorige werkdag
                - Wat je nog gaat doen vandaag
                - Of het werk dat je nog hebt staan af gaat komen
        - Testen
[ ]         - Test Driven Development uitvoeren door tests te schrijven wanneer een nieuwe functie
              wordt toegevoegd. Dit voorkomt technical debt later
[x]         - Onderzoek doen naar bijkomend testframework en welke opties daar zijn
        - Grootte PR
[x]         - Eerder pull requests aanmaken en laten reviewen
[x]         - Pull requests klein houden: minder functionaliteit
[x]             - Eén functie per pull request
[x]             - Niet wachten met meer features inbouwen
                    - Features inbouwen kan ook tijdens het wachten op goed of -afkeuring
```

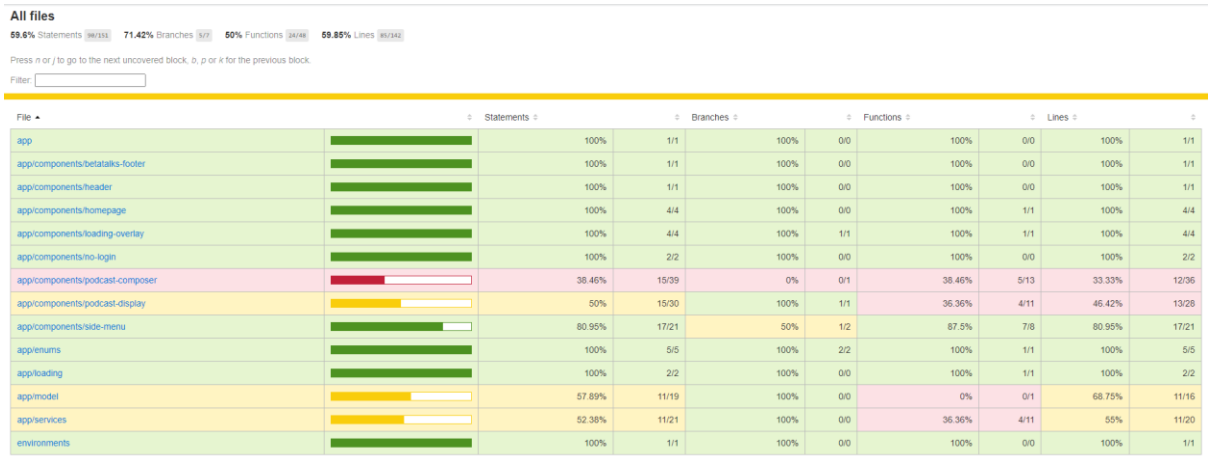
Figuur 26: Populariteit van ieder frontend framework wereldwijd



Figuur 27: Podcast service in het Angular framework om verzoeken te versturen naar de Podcast Portaal API

```typescript
@Injectable()
export class PodcastService {
    constructor(private http: HttpClient) { }

    // Retrieve all podcasts from the database
    getPodcasts(): Observable<Podcast[]> {
        return this.http.get<Podcast[]>(environment.apiURL);
    }

    // Get a podcast based on the specified ID
    // The ID will be provided by either the host or by the guest when they log on
    getPodcastById(podcastID: number): Observable<Podcast> {
        return this.http.get<Podcast>(environment.apiURL + podcastID.toString());
    }

    // Create a new podcast based on the data that was passed
    createNewPodcast(podcast: Podcast): Observable<Podcast> {
        return this.http.post<Podcast>(environment.apiURL, podcast, httpOptions);
    }

    // Cancel current podcast by performing a DELETE request to server
    cancelPodcast(podcastID: number): Observable<Podcast> {
        return this.http.delete<Podcast>(environment.apiURL + podcastID.toString());
    }

    // Updates an exisiting podcast by performing a PUT HTTP request to the server
    updatePodcast(podcast: Podcast): Observable<Podcast> {
        return this.http.put<Podcast>(environment.apiURL + podcast.id.toString(), podcast, httpOptions);
    }
}
```

Figuur 28: Angular test code coverage rapport in HTML waarin te zien is welke onderdelen van de code wel en niet getest worden



All files

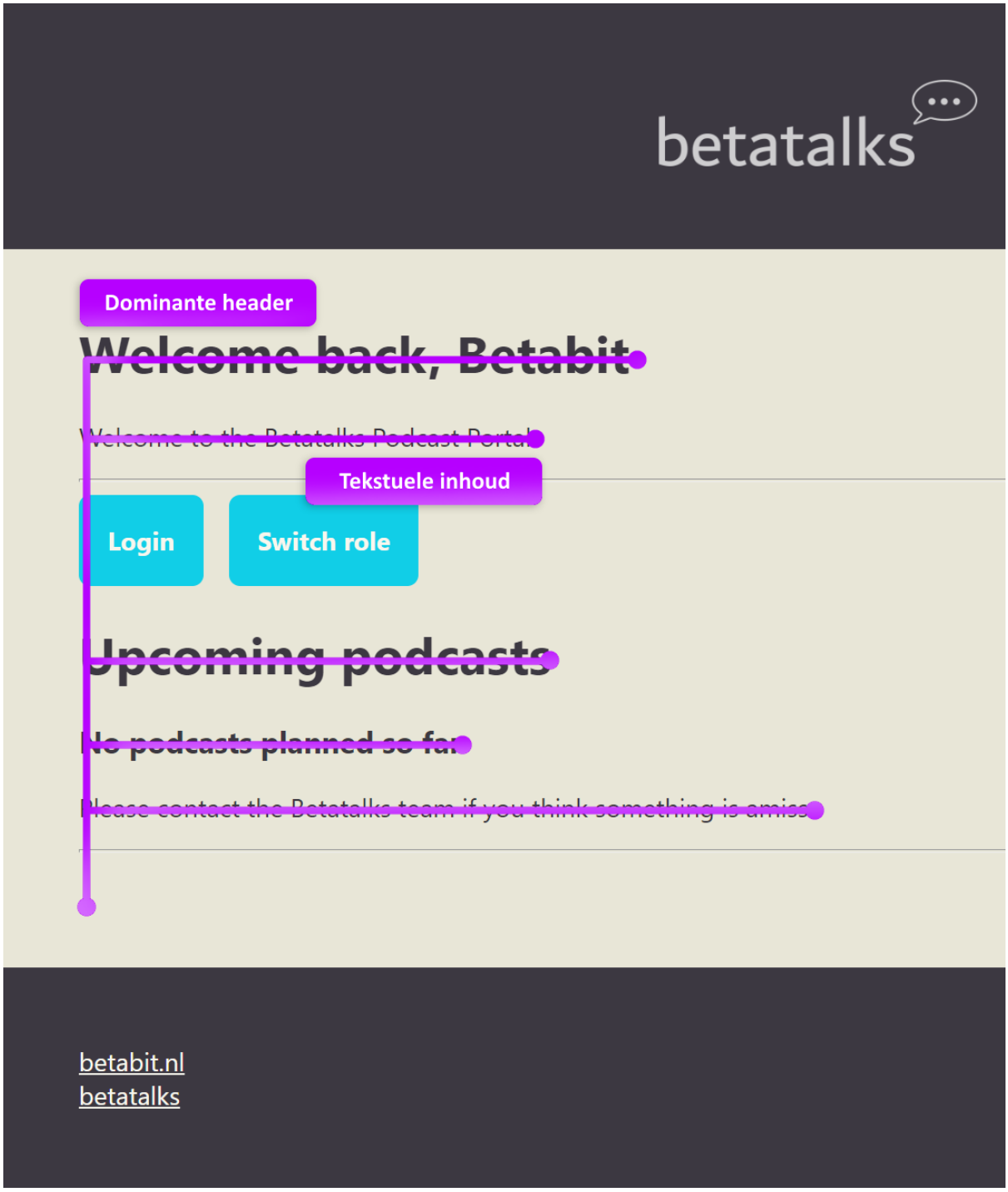
59.6% Statements 94/151 **71.42%** Branches 5/7 **50%** Functions 24/48 **59.85%** Lines 85/142

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

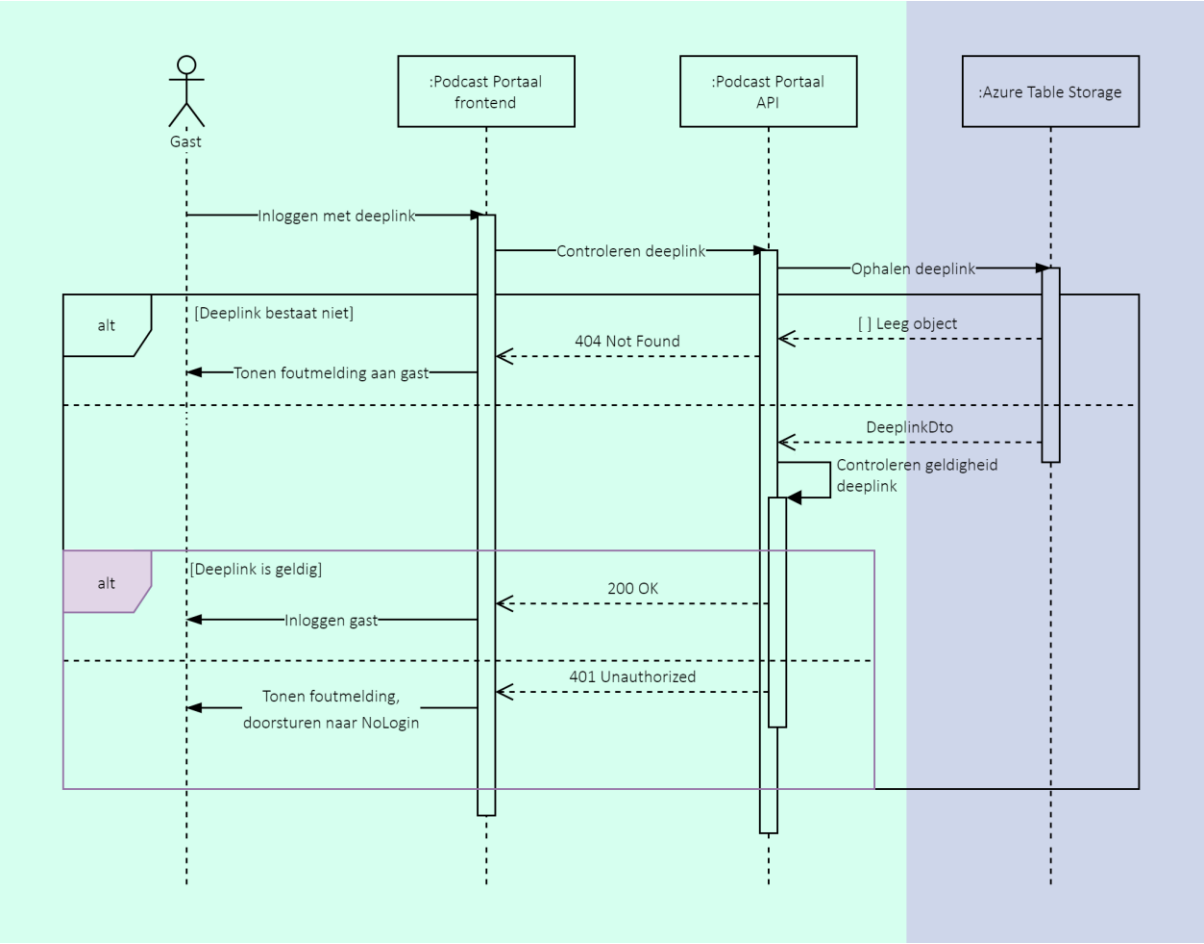
Filter:

File	Statements		Branches		Functions		Lines	
app	100%	1/1	100%	0/0	100%	0/0	100%	1/1
app/components/betatalks-footer	100%	1/1	100%	0/0	100%	0/0	100%	1/1
app/components/header	100%	1/1	100%	0/0	100%	0/0	100%	1/1
app/components/homepage	100%	4/4	100%	0/0	100%	1/1	100%	4/4
app/components/loading-overlay	100%	4/4	100%	1/1	100%	1/1	100%	4/4
app/components/no-login	100%	2/2	100%	0/0	100%	0/0	100%	2/2
app/components/podcast-composer	38.46%	15/39	0%	0/1	38.46%	5/13	33.33%	12/36
app/components/podcast-display	50%	15/30	100%	1/1	36.36%	4/11	46.42%	13/28
app/components/side-menu	80.95%	17/21	50%	1/2	87.5%	7/8	80.95%	17/21
app/enums	100%	5/5	100%	2/2	100%	1/1	100%	5/5
app/loading	100%	2/2	100%	0/0	100%	1/1	100%	2/2
app/model	57.89%	11/19	100%	0/0	0%	0/1	68.75%	11/16
app/services	52.38%	11/21	100%	0/0	36.36%	4/11	55%	11/20
environments	100%	1/1	100%	0/0	100%	0/0	100%	1/1

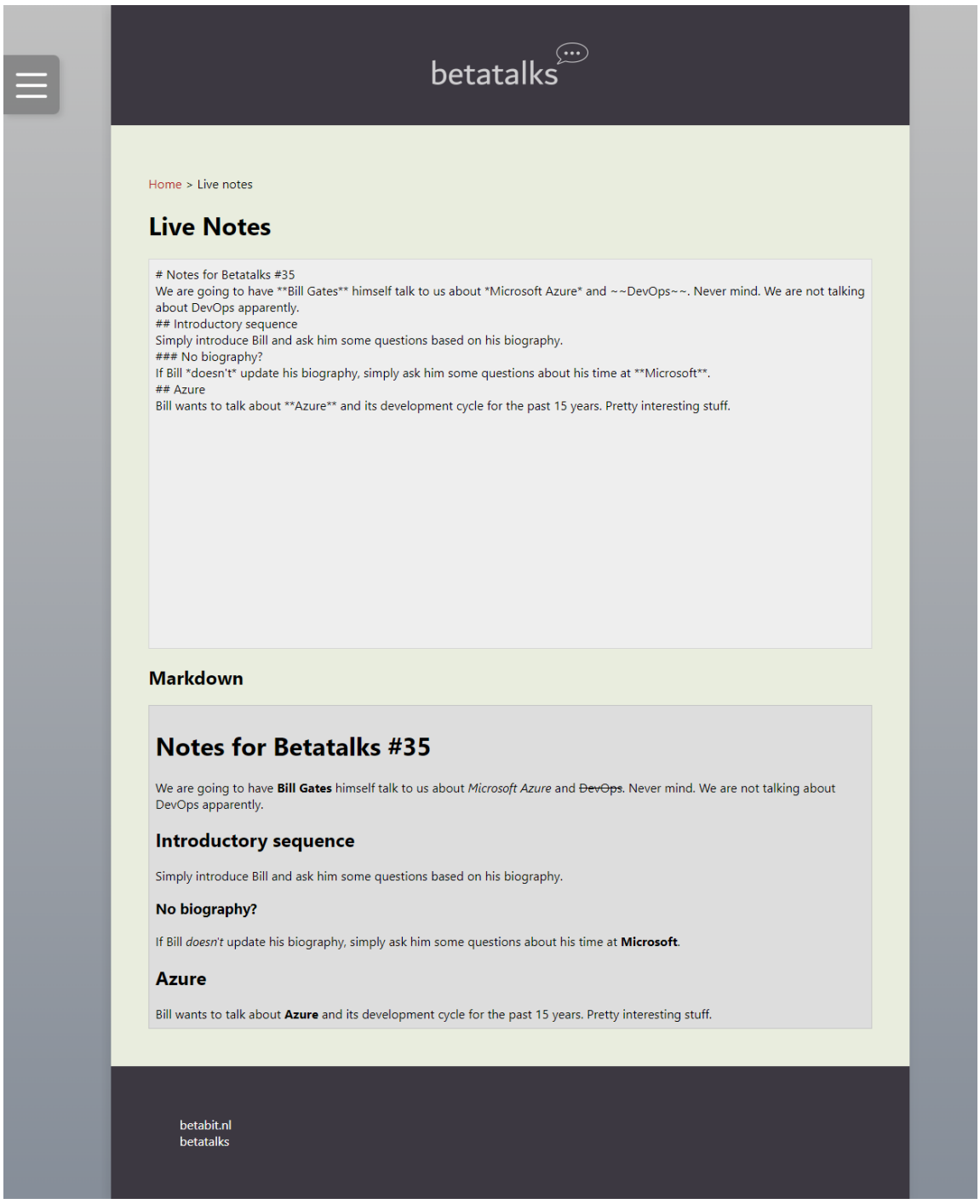
Figuur 29: F-layout in het Podcast Portaal. De applicatie is opgedeeld in headers en aparte inhoud-boxes. De lezer leest de dominante onderdelen vaak eerst en zoekt daarna vertikaal naar het volgende interessante onderdeel.



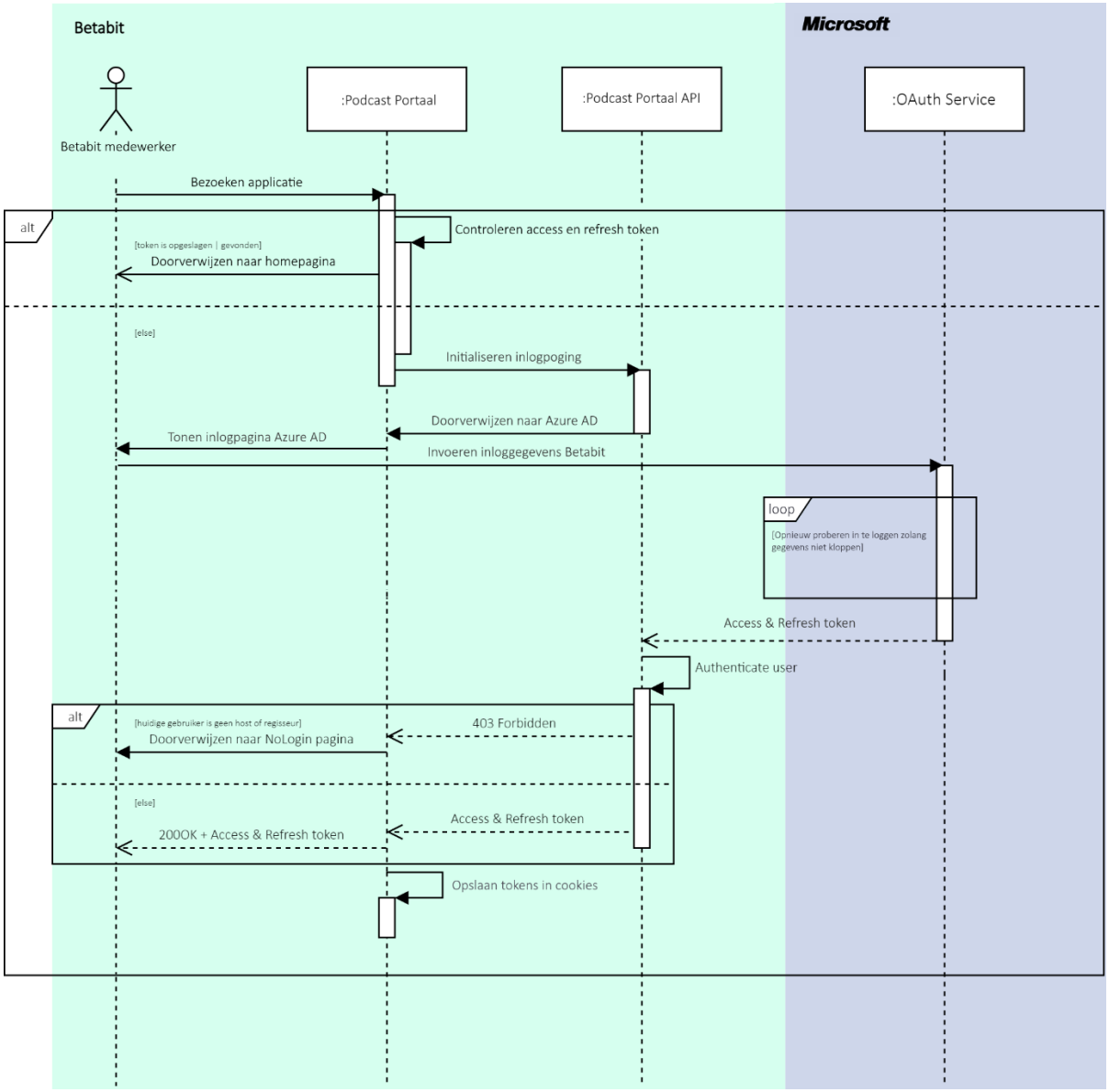
Figuur 30: Sequence diagram over het inloggen met gebruik van een deeplink. De frontend is verantwoordelijk voor het versturen van de deeplink naar de API. De frontend is verantwoordelijk voor het controleren van de geldigheid als de deeplink in de database staat

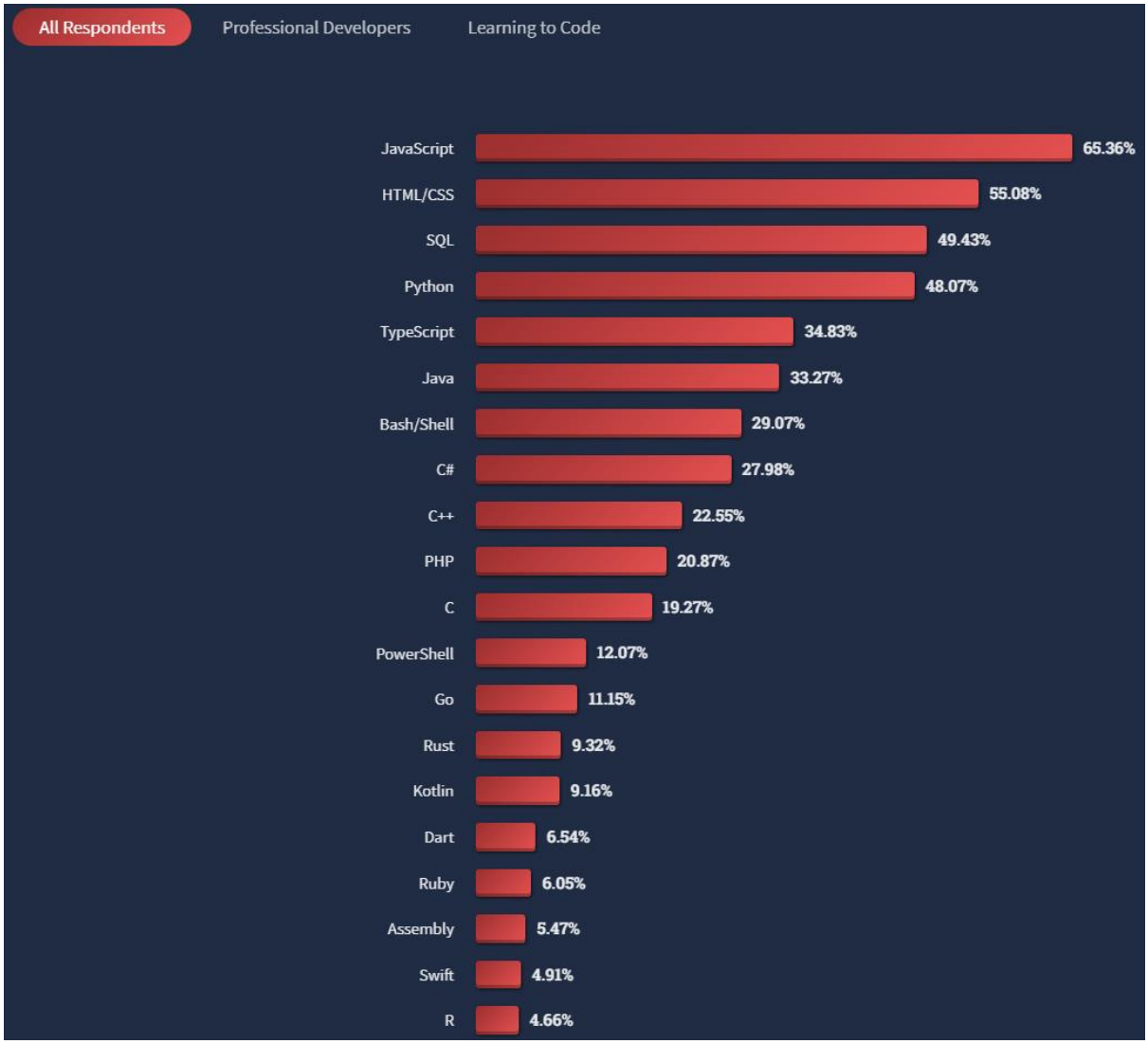


Figuur 31: Live Notes omgeving waar gasten, hosts en regisseurs met elkaar kunnen communiceren. De omgeving ondersteunt ook markdown om tekst vorm te geven.



Figuur 32: Sequence diagram voor het inloggen van medewerkers op het podcast portaal. Op deze manier kunnen de hosts en regisseurs inloggen zonder een account aan te hoeven maken.





Figuur 33: Top 20 meest gebruikte programmeertalen volgens de jaarlijkse StackOverflow survey (2022) JavaScript staat bovenaan en TypeScript op plaats vijf.

sonar-project.properties

```
1   sonar.host.url=http://localhost:9000
2   sonar.login=admin
3   sonar.password=admin
4   sonar.projectKey=podcast-portal
5   sonar.projectName=podcast-portal
6   sonar.projectVersion=0.1
7   sonar.sourceEncoding=UTF-8
8   sonar.sources=src
9   sonar.exclusions=**/node_modules/**
10  sonar.tests=src
11  sonar.test.inclusion=**/*.spec.ts
12  sonar.typescript.lcov.reportPaths=coverage/lcov.info
```

Figuur 34: SonarQube properties bestand met configuratie voor de SonarQube applicatie en welke bestanden er gecontroleerd gaan worden.

Bibliografie

Adobe. (n.d.). *Open Sans*. Retrieved from Adobe Fonts: https://fonts.adobe.com/fonts/open-sans

Agrawal, H. (2022, Augustus 18). *What is TypeScript and why should you use it?* Retrieved from Contentful: https://www.contentful.com/blog/what-is-typescript-and-why-should-you-use-it/

Arthur Fox. (n.d.). *WAV or MP3: Which Is The Superior Audio Format?* Retrieved from mynewmicrophone.com: https://mynewmicrophone.com/wav-or-mp3-which-is-the-superior-audio-format/

Asiuwhu, E. (2022, April 28). *Blazor vs. React: Minimize JavaScript in your SPAs*. Retrieved from blog.logrocket.com: https://blog.logrocket.com/blazor-vs-react-minimize-javascript/

Babich, N. (2017, April 5). *F-Shaped Pattern for Reading Contennt*. Retrieved from uxplanet.org: https://uxplanet.org/f-shaped-pattern-for-reading-content-80af79cd3394

Bambook. (n.d.). *Bambook*. Retrieved from bambook.org: https://www.bambook.org/nl/

Basit et al. (2021, November 13). *Font redistribution FAQ for Windows*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/typography/fonts/font-faq

Bennyboy1973. (2022, Mei 18). *Is it possible to learn and use Blazor without prior knowledge or experience with Razor*. Retrieved from Quora: https://stackoverflow.com/questions/72296649/is-it-possible-to-learn-and-use-blazor-without-prior-knowledge-or-experience-wit

Bhati, N. (2022, Juli 29). *React with TypeScript or JavaScript*. Retrieved from Emizentech: https://www.emizentech.com/blog/react-with-typescript-or-javascript.html

Borrelli, P. (2021, Oktober 26). *Angular vs. React vs. Vue.js: Comparing performance*. Retrieved from blog.logrocket.com: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/

Chiba, R. (2021, Oktober 6). *How to Choose the Best Audio File Format and Codec*. Retrieved from filestack: https://blog.filestack.com/thoughts-and-knowledge/audio-file-format-codec

Clark, A., Abramov, D., Kassens, J., Savona, J., Story, J., Tan, L., . . . Huang, X. (2022, Juni 15). *React Labs: What we have been working on*. Retrieved from reactjs.org: https://reactjs.org/blog/2022/06/15/react-labs-what-we-have-been-working-on-june-2022.html

Code Smells. (n.d.). Retrieved from Refactoring Guru: https://refactoring.guru/refactoring/smells

Coolors. (n.d.). *coolors.co*. Retrieved from coolors.co: coolors.co

de le Maza, M. (2020, Oktober). *Master the sailboat retrospective in 4 steps*. Retrieved from Miro: https://miro.com/guides/retrospectives/how-to-run-sailboat-retrospective

Dubey, B. K. (2022, Mei 19). *Difference between TypeScript and JavaScript*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/

Front-end Frameworks. (2020, Januari 1). Retrieved from 2020.stateofjs.com: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/

Fu, J. (2022, September 14). *Exploring SonarQube With Create React App*. Retrieved from Better Programming: https://betterprogramming.pub/exploring-sonarqube-with-create-react-app-e0b7e0e5658c

guardrex, thiennn, & diberry. (2023, Januari 11). *ASP.NET Core Blazor JavaScript interoperability (JS interop)*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/aspnet/core/blazor/javascript-interoperability/?view=aspnetcore-7.0

Hámori, F. (2022, Mei 31). *The History of React.js on a timeline*. Retrieved from RisingStack: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/

Harband et al. (n.d.). *eslint-plugin-react*. Retrieved from npmjs: https://www.npmjs.com/package/eslint-plugin-react

Hejlsberg, A. (2012, Oktober 5). *What is TypeScript and why with Anders Hejlsberg*. Retrieved from hanselminutes.com: https://www.hanselminutes.com/340/what-is-typescript-and-why-with-anders-hejlsberg

Hope, C. (2022, Oktober 18). *Regex*. Retrieved from computerhope.com: https://www.computerhope.com/jargon/r/regex.htm

jimmart-dev, tamram, stevenmatthew, alexbuckgit, Amrinder-Singh29, v-kents, . . . mariekekortsmit. (2022, November 8). *Grant limited access to Azure Storage resources using shared access signatures (SAS)*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview

Kapoor, A. (2022, Januari 24). *Angular Roadmap: The Past, Present and Future*. Retrieved from enlear.academy: https://enlear.academy/angular-roadmap-the-past-present-future-212d8b98591b

Kulla-Mader et al. (2023, Januari 4). *What is Azure Pipelines?* Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines

lawsofux.com. (n.d.). Retrieved from Laws of UX: https://lawsofux.com/

Liangying et al. (2023, Januari 13). *Tutorial: Create a chat app with Azure Web PubSub service*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/azure-web-pubsub/tutorial-build-chat?tabs=csharp

Malloy, S. (2020, September 2). *An introduction on using SonarQube*. Retrieved from Crest Data Systems: https://www.crestdatasys.com/blogs/an-introduction-on-using-sonarqube/

Markdown Guide. (n.d.). *Basic Syntax*. Retrieved from markdownguide.org: https://www.markdownguide.org/basic-syntax/

Markdown Guide. (n.d.). *What is Markdown?* Retrieved from markdownguide.org: https://www.markdownguide.org/getting-started/

Megida, D. (2020, Februari 13). *Object Oriented Programming in JavaScript*. Retrieved from freecodecamp.org: https://www.freecodecamp.org/news/how-javascript-implements-oop/

Microsoft. (2022, Augustus 6). *Azure Web PubSub*. Retrieved from azure.microsoft.com: https://learn.microsoft.com/en-us/azure/azure-web-pubsub/overview

Microsoft. (2022, Augustus 11). *Introduction to Azure Blob Storage*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Movement, UX. (2016, November 16). *Progress bars vs. Spinners: When to Use Which*. Retrieved from medium.com: https://uxmovement.medium.com/progress-bars-vs-spinners-when-to-use-which-1107a8ecb522

Mozilla. (2022, September 14). *What is JavaScript*. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

Mozilla. (n.d.). *JavaScript Type Safety Systems*. Retrieved from mozilla.github.com: https://mozilla.github.io/firefox-browser-architecture/text/0009-type-safety-systems.html

Mozilla. (n.d.). *SPA (Single-page application)*. Retrieved from Developer Mozilla: https://developer.mozilla.org/en-US/docs/Glossary/SPA

Mozilla. (n.d.). *The WebSocket API*. Retrieved from developer.mozzila.org: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

ParTech Media. (2021, December 22). *TypeScript vs JavaScript - Key differences*. Retrieved from Partech: https://www.partech.nl/nl/publicaties/2021/12/typescript-vs-javascript---key-differences#

Podcast Portaal repository. (n.d.). *Podcast Portaal repository*. Retrieved from Azure DevOps: https://dev.azure.com/npauw/Podcast%20Portaal

Ramel, D. (2022, Juni 28). *TypeScript Vaults Ahead of Java to Crack Stack Overflow Top 5*. Retrieved from Visual Studio Magazine: https://visualstudiomagazine.com/articles/2022/06/28/typescript-so.aspx

Raval, N. (2022, September 20). *Top 10 JavaScript usage statistics to watch out for in 2022*. Retrieved from Radixweb: https://radixweb.com/blog/top-javascript-usage-statistics

Raval, N. (2022, Oktober 13). *TypeScript vs. JavaScript: The Difference You Should Know*. Retrieved from radixweb.com: https://radixweb.com/blog/typescript-vs-javascript

React. (n.d.). *Forms*. Retrieved from React: https://reactjs.org/docs/forms.html

React. (n.d.). *Updating Objects in State*. Retrieved from React Docs Beta: https://beta.reactjs.org/learn/updating-objects-in-state

rickvdbosch, jimmart-dev, stevenmatthew, tamram, v-alje, navba-MSFT, . . . bandersmsft. (2022, Juli 14). *Manage storage account access keys*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/storage/common/storage-account-keys-manage?tabs=azure-portal

Roselli, A. (2021, November 7). *Avoid the Hamburger Menu for Desktop Layouts*. Retrieved from adrianroselli.com: https://adrianroselli.com/2016/01/avoid-the-hamburger-menu-for-desktop-layouts.html

Saini, R. (2019, Juli 27). *React vs. Angular vs. Vue.js*. Retrieved from medium.com: https://rahul-saini.medium.com/react-vs-angular-vs-vue-js-which-one-should-i-learn-in-2019-23ec05a49f78#:~:text=Angular%20has%20a%20steep%20learning%20curve%2C%20considering%20it,terms%20of%20understanding%20how%20the%20front%20end%20works.

Salnik, R. (2022, Augustus 25). *Angular vs. React. Which JS Framework Is Better in 2022?* Retrieved from brocoders: https://brocoders.com/blog/angular-vs-react/

SlashData. (2022, September). *Size of programming communities worldwide as of 2022*. Retrieved from Statista: https://www.statista.com/statistics/1241923/worldwide-software-developer-programming-language-communities/#:~:text=According%20to%20the%20survey%2C%20the,programming%20language%20in%20the%20world.

Smirk. (2019, April 23). *The Four-Second Rule: Page Speed, UX & SEO*. Retrieved from smirknewmedia.com: https://smirknewmedia.com/page-speed-and-seo/

Subramaniyan, S. (2020, Juli 30). *The Rounded User Experience*. Retrieved from uxplanet.org: https://uxplanet.org/the-rounded-user-experience-ff7a1898ab33

Team, i. (2021, April 28). *Is Blazor the future in Web App Development?* Retrieved from ifourtechnolab.com: https://www.ifourtechnolab.com/blog/is-blazor-the-future-in-web-app-development

Toonen, E. (2021, Februari 10). *What are breadcrumbs? Why are they important for SEO?* Retrieved from Yoast: https://yoast.com/breadcrumbs-seo/

Trends.Builtwith. (2022, Oktober 1). *TypeScript Usage Statistics*. Retrieved from Trends.Builtwith.com: https://trends.builtwith.com/javascript/TypeScript#:~:text=TypeScript%20Competitors%20and%20Similar&text=Get%20a%20list%20of%2054%2C273,to%20sites%20in%20this%20list.

Turner, J. (2014, April 2). *Announcing TypeScript 1.0*. Retrieved from Microsoft Dev Blogs: https://devblogs.microsoft.com/typescript/announcing-typescript-1-0/

Valishery, L. S. (2022, Augustus 9). *Most usede web frameworks among developers worldwide, as of 2022*. Retrieved from statista: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

vicancy, & dbradish-microsoft. (2023, Januari 13). *Tutorial: Publish and subscribe messages between WebSocket clients using subprotocol*. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/azure/azure-web-pubsub/tutorial-subprotocol?tabs=csharp#publish-messages-from-client

White, M. (2020, Augustus 24). *Why Static Typing & Why is TypeScript so popular?* Retrieved from section.io: https://www.section.io/engineering-education/typescript-static-typing

Wozniewicz, B. (2019, Februari 1). *The Difference Between a Framework and a Library*. Retrieved from freecodecamp.org: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/