Het framework dat wij hebben gebruikt is Java Play 2. Dit framework heeft enkele belangrijke voordelen voor ons project, zoals de auto-refresh feature na wijzigingen in de code, een zeer goede debugger en het is Java en scala gebaseerd. Aangezien wij zeer vertrouwd zijn met Java en het ook een vereiste was van de klant dat de back-end in Java geschreven moest zijn, was de keuze vlug gemaakt. Het nadeel aan dit framework is wel dat het relatief nieuw is, waardoor er minder documentatie is, de API soms nog kan veranderen en er een kleinere community rond bestaat die ons zou kunnen helpen.

Het Play framework maakt gebruik van de MVC architectuur. In de model schrijven wij pure Java code, zoals klassen die gemapt worden op database tabellen via Ebean ORM. Java Play en Ebean werken zeer goed samen en zou dus makkelijk te gebruiken zijn dan bv. JPA. En de model bevat natuurlijk ook onze hulpklassen, zoals bv. voor de web API te modelleren. Voor de View waren wij eigenlijk vrij om te kiezen welk front-end framework wij wilden gebruiken. Omdat wij vorig semester met Angular hebben gewerkt en iedereen hier bekwaam in was, besloten we dit te gebruiken. Onze applicatie bestaat uit meerdere controllers die elk verschillende actions bevatten voor elke HTTP API call.

Een typische call om bv. gegevens uit de database te halen gaat als volgt: eerst wordt aan de front-end een asynchrone subscribe methode aangeroepen van de Service. Die Service helpt om het seperation of concerns principe uit te voeren. Hierna voert deze Service de HTTP call uit naar de back-end. Het Play framework voorziet hiervoor een routes bestand om de juiste link aan te spreken. De Controller voert de aanvraag en zal een JSON object versturen naar de front-end.

Nu aangezien die subscribe methode asynchroon is, zaten wij in het begin van ons project soms in moeilijkheden omdat sommige variabelen op undefined bleven te staan. Dit was vooral het geval bij meerdere asynchrone calls na elkaar en dat je erna bv. een globale variabele moet updaten.

Hier zien jullie een voorbeeld van zo’n probleem. Stel dat ik alle goedgekeurde aanvragen van enkele dokters aanvraag, en hierna wil ik een extra key toevoegen dat de id van een persoon mapt met zijn naam. Hier kan je nooit weten welke methode als eerste klaar zal zijn, dus waren sommige variabelen undefined. De oplossing hiervoor was alle methoden in een Observable array te steken. Een Observable object is hetgeen je subscribe methode terugggeeft en bevat de opgeroepen data. Nadat voor alle dokters de methode is opgeroepen, kan je gebruik maken van de forkJoin methode. Deze zorgt er dan voor dat er wordt gewacht tot alle methodes zijn afgewerkt, zodat je zeker geen undefined variabelen meer hebt. De dataArray bevat dus een array waarbij elke index data bevat van bovenstaande methode. Hierna kan je dan voor elke afgewerkte methode de naam toevoegen.

In ons project hebben wij ook zeer veel gebruik gemaakt van Angular Material. Zowel voor icoontjes, progress spinners, tabellen en onze navigatie bar. Het handige aan angular material is dat alles er heel clean uitziet en je met gewoon enkele opties mee te geven handige functies kan toevoegen.

Dit is een voorbeeld van zo’n angular material tabel. Deze tabel geeft alle aanvragen van een dokter weer met een detail view als er op geklikt werd. Je hoeft hiervoor enkel je data mee te geven en de kolommen te definiëren, meer niet. In het begin hadden wij hier nog opties om de rijen te filteren en te sorteren, maar dan hebben wij besloten om nog een detail view mee te geven van de vorige aanvragen. Er was hier geen ondersteuning voor in angular material, dus hebben wij dit opgelost met een beetje typescript en CSS. Iedereen weet natuurlijk dat CSS altijd exact doet wat je probeert te implementeren, dus daar hebben wij zeker geen meerdere uren aan verspild... Maar door die detailview werkte het sorteren en filteren niet meer naar behoren, dus hebben wij ervoor gekozen om dat weg te laten en alle aanvragen gesorteerd door te geven op Datum van aanvraag bij het meegeven van de data.