Winkel van de Toekomst

Wobbe 2000

Software Architecture Document

Versie 0.1

Auteurs: Malcolm Kindermans, Jeroen Kruis, Maurits van Mastrigt en Auke Willem Oosterhof.

Instituut: Hanzehogeschool Groningen

Datum: 19 januari 2015

Documenthistorie

Datum	Versie	Beschrijving	Auteur
18 januari 2015	0.1	Initiële versie	Projectgroep

Distributie

Naam	0.1
E. Nijkamp	×
Projectgroep	×

Accordering document

Namens Winkel van de Toekomst:

(handtekening)

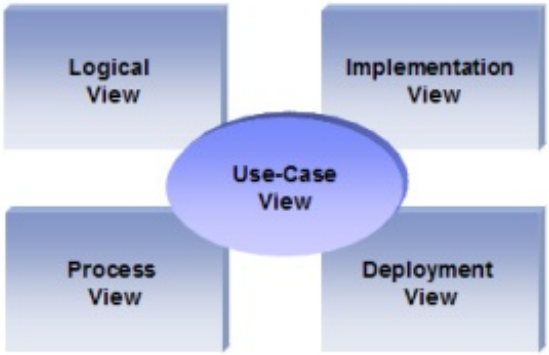
Inhoud

1. Inleiding

Dit hoofdstuk beschrijft het doel en de opbouw van dit document.

1.1. Doel van dit document

Dit document is geschreven om de architectuur te beschrijven van het Wobbe 2000 systeem, dat ontwikkeld is voor supermarktketen *Wobbe*. Om de verschillende aspecten van het systeem zo duidelijk mogelijk te belichten, zal het systeem beschreven worden vanuit een aantal verschillende architecturele views. De verschillende RUP views zijn uitgewerkt op basis van het RUP 4+1 model.



Het 4+1 view model stelt de verschillende belanghebbenden in staat om vanuit hun eigen perspectief de invloed van de gekozen architectuur te bepalen. De Process View (communicatie van processen) is niet als los hoofdstuk uitgewerkt, maar ondergebracht bij de hoofdstukken 3.3 en 5.

1.2. Referenties

Hieronder staan de referenties die gebruikt zijn voor deze SAD. Samen met de referentie is ook de vindplaats vermeldt.

Titel	Versie	Auteur(s)	Vindplaats
Vision document	0.1	Projectgroep	http://goo.gl/olmxoM
Software Development Plan	0.1	Projectgroep	http://goo.gl/KnP5NQ

1.3. Documentoverzicht

In onderstaande tabel zijn de belanghebbenden en het doel van alle hoofdstukken opgenomen.

Hoofdstuk	Belanghebbenden	Doel
2. Architecturele eisen	Software Architect	Overzicht van architectureel relevante requirements.
3. Logical View	Ontwikkelaars (t.b.v. technisch ontwerp)	Inzicht in de functionele structuur van de applicatie.
4. Implementation View	Ontwikkelaars (t.b.v. de bouw)	Inzicht in de technische structuur van de applicatie.
5. Deployment View	Systeembeheerders	Inzicht in de manier waarop de applicatie wordt gedeployed en de manier waarop de (interne en externe) communicatie plaatsvindt.

2. Architecturele eisen

Deze sectie beschrijft de software-eisen welke voor het ontwikkelen van de software-architectuur van belang zijn.

2.1. Niet-functionele eisen

De niet-functionele eisen zijn als volgt beschreven:

Bron	Naam	Architecturele relevantie	Geadresseerd in
Vision Document	Scanverwerkingstijd	Het scannen mag niet langer dan een bepaalde tijd duren, zodat de gebruiker niet lang hoeft te wachten voor het volgende product gescand kan worden.	§ 6.1
Vision Document	Platform ondersteuning	De mobiele app moet door meerdere platformen ondersteund worden, zodat de meeste klanten er gebruik van kunnen maken.	§ 6.1
Vision Document	Concurrent clients	Het systeem moet meerdere actieve clients tegelijk kunnen behandelen, zodat alle klanten in de supermarkt tegelijk hun producten kunnen scannen.	§ 6.1
Vision Document	Easy to deploy	Het systeem moet eenvoudig uit te rollen zijn bij andere filialen.	§ 6.1
Vision Document	Koppeling met voorraadsysteem	Er dient een koppeling te komen met het voorraadsysteem, zodat de klanten kunnen zien of een artikel nog op voorraad is.	§ 4.2
Vision Document	Responsive website	De website dient responsive gemaakt te worden, zodat deze op de meeste beeldschermen bekeken en gebruikt kan worden.	§ 4.4

2.2. Functionele eisen

In onderstaande tabel is zijn de functionele eisen opgenomen.

Bron	Naam	Architecturele relevantie	Geadresseerd in
Vision Document	Register	Gebruikers hebben een account nodig zodat ze hun boodschappenlijstjes kunnen bewaren.	§ 5.3
Vision Document	Log in (website)	Gebruikers moeten in kunnen loggen, zodat ze hun gegevens kunnen bekijken.	§ 5.3
Vision Document	Log in (mobile app)	Gebruikers moeten in kunnen loggen, zodat ze hun boodschappenlijstjes kunnen bekijken.	§ 5.3
Vision Document	Create shopping list	Gebruikers moeten nieuwe boodschappenlijstjes kunnen maken.	§ 5.3
Vision Document	Edit shopping list	Gebruikers moeten bestaande boodschappenlijstjes kunnen wijzigen.	§ 5.3
Vision Document	Show shopping list (app)	Gebruikers moeten hun boodschappenlijstje kunnen bekijken tijdens het winkelen.	§ 5.3
Vision Document	Scan product	Gebruikers moeten de producten uit de winkel kunnen scannen met hun smartphone.	§ 5.3
Vision Document	Generate receipt	Gebruikers moeten na het betalen van de producten een bon krijgen.	§ 5.3
Vision Document	Delete scanned product	Na het scannen van een product moet deze verdwijnen van het lijstje, zodat producten niet dubbel gekocht worden.	§ 5.3
Vision Document	Add scanned product	Wanneer een nieuwe product gescant wordt moet deze op de boodschappenlijst toegevoegd worden.	§ 5.3
Vision Document	Pay with NFC	De gebruiker moet contactloos kunnen betalen zodat het betalen makkelijker en sneller gaat.	§ 5.3
Vision Document	Show ads	De gebruiker moet persoonlijke advertenties te zien krijgen zodat er gerichter geadverteerd kan worden.	§ 5.3

3. Logical View

Deze sectie beschrijft de architecturele opbouw van het systeem.

3.1. Lagen

De opbouw van het systeem kan worden onderverdeeld in vier lagen.

Laag	Doel
Presentatie	De grafische interface voor de interactie met de eindgebruikers.
Service	Het communiceren tussen de presentatie- en de domeinlaag. Dit wordt gedaan m.b.v. een API.
Domein	De applicatielogica, zoals het presenteren en verwerken van gegevens.
Data	Het aanbieden en persisteren van gebruikersdata.

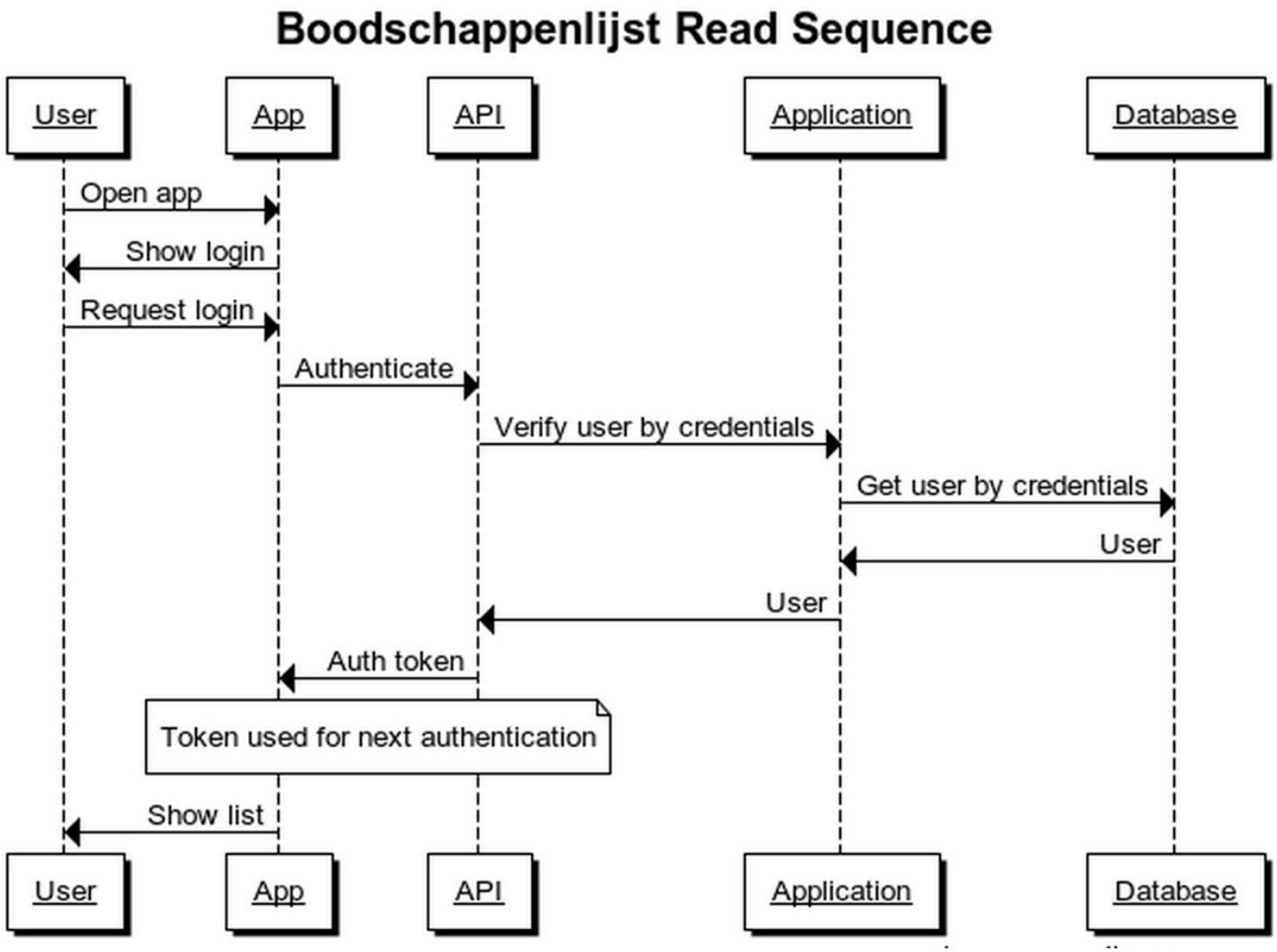
3.2. Deelsystemen

Het systeem bestaat uit drie deelsystemen, namelijk:

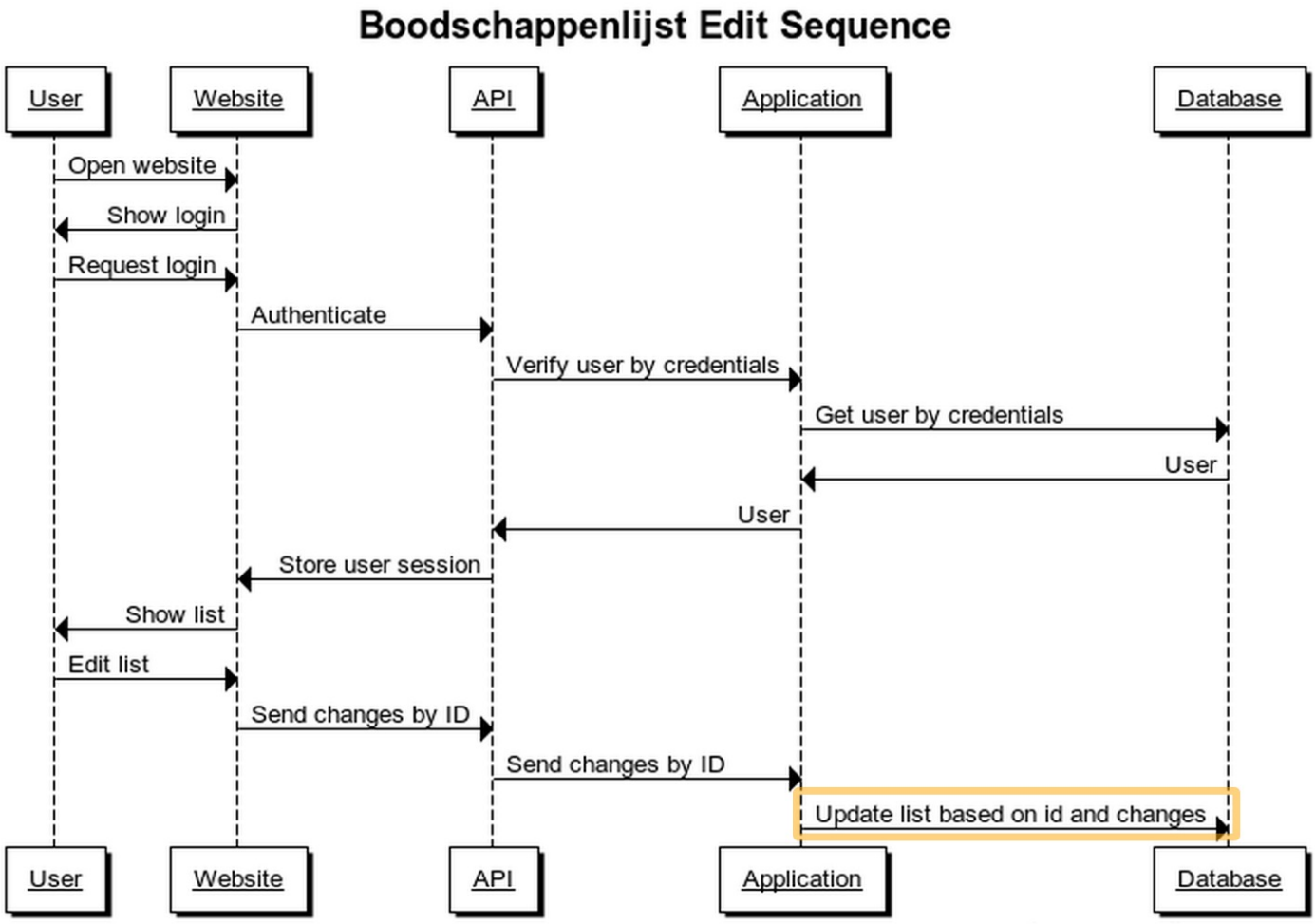
Deelsystemen	Doel
Smartphone	Dit apparaat heeft de app geinstalleerd. Hiermee kan de gebruiker acties uitvoeren op het systeem.
Personal Computer	Met behulp van de webbrowser kan de gebruiker inloggen op de website. Op de website kan de gebruiker een lijst aanmaken, aanpassen en verwijderen.
Webserver	Dit systeem verzorgt de API, die tevens wordt gebruikt door de mobiele app en de website. De website wordt aangeboden vanaf deze server.

3.3. Realisatie van functionele eisen

De hoofdfunctionaliteit van het systeem bestaat uit het bekijken en aanpassen van een boodschappenlijstje. Onderstaande sequence diagrammen beschrijven zowel het bekijken van een boodschappenlijst via de mobiele app als het aanpassen van een boodschappenlijst via de website.



Deze afbeelding beschrijft het bekijken van een boodschappenlijst via de mobiele app.



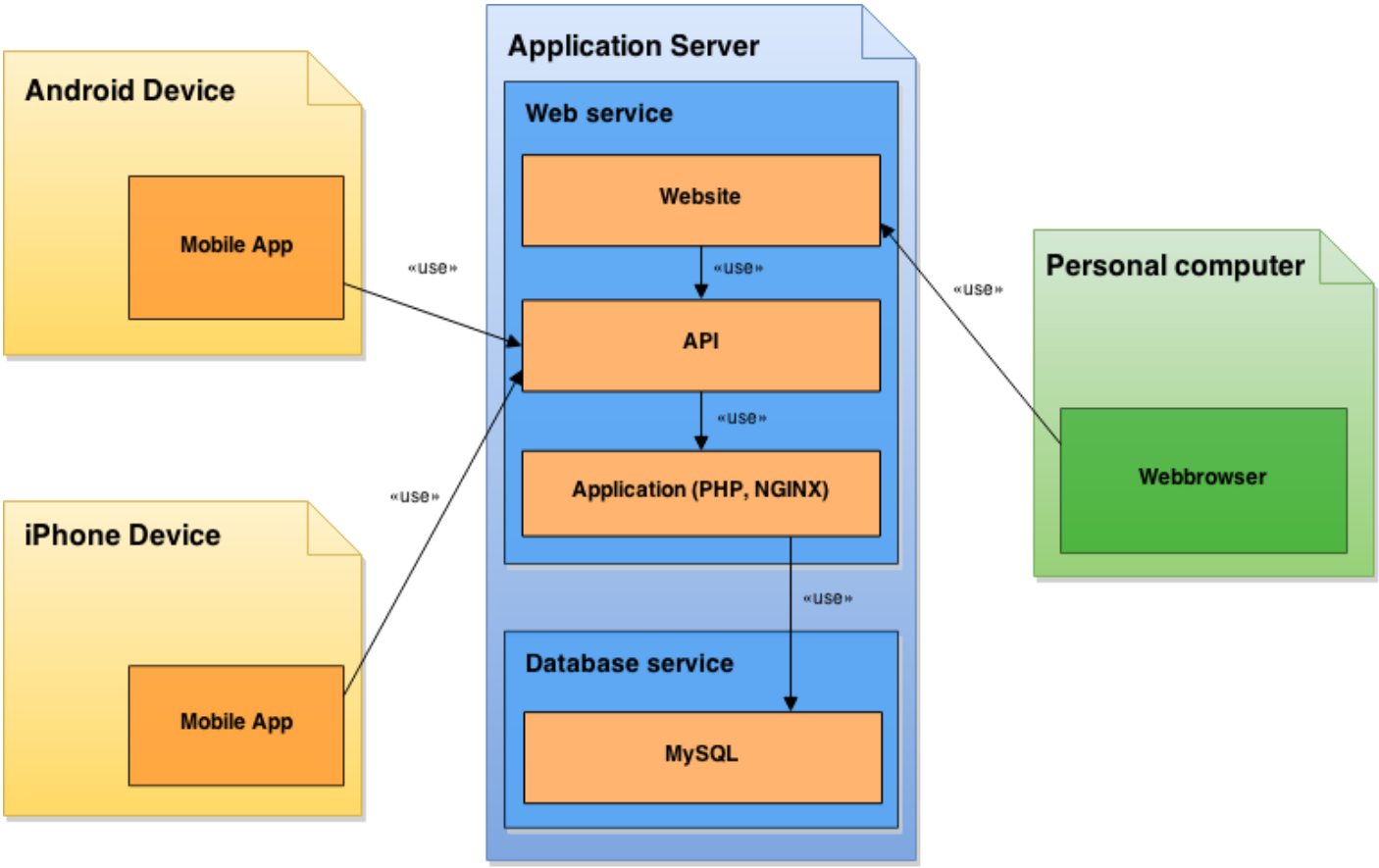
Deze afbeelding beschrijft het aanpassen van een boodschappenlijst via de website.

4. Implementation View

Deze sectie beschrijft de technische invulling van de logical view.

4.1. Packagestructuur

Het onderstaande diagram geeft weer hoe de verschillende deelsystemen verband houden met elkaar:



Dit diagram geeft weer hoe de mobiele applicatie interacteert met de API en hoe een gebruiker via een webbrowser gebruik kan maken van de wbsite. De onderliggende laag, met name de applicatielaag, biedt alle functionaliteiten en presenteert de (MySQL) databasegegevens, aan de API en de website, in het juiste formaat.

4.2. Invulling van lagenstructuur

Deze paragraaf beschrijft de technische invulling van de, in de logica view, onderscheiden lagen.

4.2.1. Presentatielaag

De presentatielaag omvat de deelsystemen die een interface bieden aan de eindgebruikers van het systeem. Dit wordt gedaan door middel van een mobiele applicatie en een website. Beide onderdelen communiceren, via de service laag, met de domein-/applicatielaag voor het ophalen en muteren van gegevens.

- Mobiele Applicatie

De mobiele applicatie moet een cross-platform applicatie zijn die eenvoudig is in gebruik. Hierdoor is er voor bestaande oplossingen als Cordova en Onsen UI gekozen (zie paragraaf 4.3). Dit neemt werk uit handen, waardoor er meer focus gelegd kan worden op de gebruikerservaring, terwijl de applicatie toch grafisch aantrekkelijk blijft.

- Website

Net als de mobiele applicatie, is ook de website met bestaande componenten geïmplementeerd. Voor een aantrekkelijk grafische interface is er Twitter Bootstrap gebruikt.

Het serveren van de webistepagina’s wordt gedaan met behulp van een webservice. Hierbij is er gekozen voor het Laravel framework, omdat dit framework makkelijk is in gebruik, geen stijle leercurve heeft en er al de nodige ervaring in de groep aanwezig was. De keuzecriteria zijn dus met name ontwikkelsnelheid en onderhoudbaarheid van het systeem.

4.2.2. Servicelaag

De communicatie gebeurd met behulp van HTTP-requests. Hierbij wordt het RESTful design principe gebruikt. Dit houdt in dat alle entiteiten op één uniforme manier, via de Application Programming Interface (API), benaderd kunnen worden. Dit vereenvoudigd de communicatie (HTTP-requests) en het opsporen en oplossen van fouten.

Het HTTP-protocol heeft veel overhead, maar biedt daarentegen veel structuur en garandeert een stabiele omgeving. Tevens wordt dit protocol door webbrowsers gebruikt, waardoor er veel informatie over te vinden is, wat het implementeren eenvoudiger maakt.

4.2.3. Domeinlaag

De applicatielogica is volledig uitgewerkt in de PHP scripttaal. Er is hierbij gekozen voor een framework, omdat hiermee sneller en transparanter ontwikkeld kan worden. Dit is voordelig voor alle ontwikkelaars. Tevens is het gekozen framework zeer expressief en biedt het veel mogelijkheden. Dit voorkomt dat ontwikkelaars onnodig tijd besteden aan het “opnieuw uitvinden van het wiel”, wat een gevaar voor elk project is. Het gebruiken van bestaande tools stelt de ontwikkelaars in staat zich meer te richten op de functionaliteiten en minder op implementatie technieken.

4.2.4. Datalaag

Voor het persisteren van data zijn er veel opties. Omdat de gegevens het goed in een relationeel model passen is er de keuze gemaakt voor een relationele database. Hierbij is er gekozen voor MySQL, vanwege het gemak in gebruik. Er is veel documentatie beschikbaar en er zijn weinig ontwikkelaars die niet met dit softwarepakket gewerkt hebben. Ook dit scheelt ontwikkeltijd, waardoor er meer gericht kan worden op de functionaliteiten in plaats van implementatie. Tevens is MySQL zeer ver doorontwikkeld, waardoor dingen zoals stabiliteit en hardware specificatie eigenlijk geen rol meer spelen.

4.3. (Her)gebruik van componenten en frameworks

Deze paragraaf beschrijft de componenten en frameworks die in het project zijn gebruikt.

4.3.1. Presentatielaag

De presentatielaag is op te delen in de mobiele applicatie en de website.

Mobiele App

De volgende componenten/frameworks zijn gebruikt bij het ontwikkelen van de mobiele applicatie:

- Cordova – Vanwege de mogelijkheid om te ontwikkelen zonder gebruik te maken van de native APIs van devices;
- Onsen UI – Vanwege de mogelijkheid om simpel een native “*look and feel*” te creëren in een webomgeving. Daarnaast heeft Onsen UI een betere performance voor mobiele apparaten dan een regulier HTML5 UI Framework.

Website

De volgende componenten/frameworks zijn gebruikt bij het ontwikkelen van de website:

- Twitter Bootstrap – Vanwege de moeilijkheidsgraad om er een mooi product mee te maken;
- Laravel - Laravel biedt zelf een ingebouwde template engine. Vanwege het gemak is deze gebruikt.

4.3.2. Servicelaag

De service laag bestaat uit:

- RESTful API – Vanwege de eenvoud van de benadering is er voor gekozen om de API RESTful te implementeren. Complexiteit wordt hierdoor zoveel mogelijk verminderd. Deze API is geïmplementeerd in het Laravel framework. Zie hiervoor ook het kopje “Domeinlaag”.

4.3.3. Domeinlaag

De domeinlaag is de applicatielogica en valt te omvatten in de term “back-end”. Hieronder valt de mogelijkheid om boodschappenlijstjes te beheren en de applicatielogica. Hiervoor is het volgende framework gebruikt:

- Laravel – Vanwege de hoeveelheid documentatie die er over beschikbaar is en de duidelijke MVC-structuur die het biedt.

4.3.4. Datalaag

De datalaag bestaat uit het volgende onderdeel:

- MySQL – Als database voor de opslag van gegevens, vanwege het sterke relationele karakter van onze data en de kleine pool van proefpersonen.

5. Deployment View

Voor deployment wordt Docker gebruikt. Met Docker is het mogelijk om de software in losse containers te draaien. Al deze containers werken met elkaar samen. Deze containers kunnen op dezelfde fysieke servers draaien, maar ze kunnen ook op verschillende servers gedeployed worden.

De mobiele app zal worden aangeboden aan de opdrachtgever. kan de app zelf verspreiden. Na de testfase zal de applicatie in de downloadwinkels van de verschillende platformen verschijven.

5.1. Web- en applicatieserver

HTTP-requests voor de website of de API worden afgehandeld door een Nginx. Nginx draait in een Dockercontainer. Poort 80 van de hostmachine is verbonden met poort 80 van deze container. Nginx stuurt de requests door naar de naar de PHP-applicatie die een reponse genereert.

5.2. Databaseserver

De webapplicatie gebruikt een MySQL-database. De database draait in een eigen container en is benaderbaar via poort 3306. De Dockercontainer moet geconfigureerd worden zodat deze poort bereikbaar is voor andere containers.

5.3. Deploymentdiagram

In onderstaande afbeelding is weergegeven hoe de diverse onderdelen met elkaar verband houden.