



HoGent

Faculteit Bedrijf en Organisatie

Projecten 3, EVA, Examenperiode 3

Sander Brugge, Bram Vannevel, Dennis Noens, Ben Van Den Bossche

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:

-

Instelling: —

Academiejaar: 2016-2017

Derde examenperiode

Faculteit Bedrijf en Organisatie

Projecten 3, EVA, Examenperiode 3

Sander Brugge, Bram Vannevel, Dennis Noens, Ben Van Den Bossche

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:

-

Instelling: —

Academiejaar: 2016-2017

Derde examenperiode

Voorwoord

Dit verslag bevat de volledige beschrijving hoe wij, groep H4, binnen het opleidingsonderdeel Projecten 3 (derde examenperiode) de opdracht hebben trachten uit te werken. Onze groep bestaat uit de studenten Sander Brugge, Dennis Noens, Bram Vannevel en Ben Van den Bossche.

Inhoudsopgave

1	Inleiding	7
2	Plan van aanpak	9
3	Applicatie	11
3.1	Android	11
3.2	MEAN	12
4	Use Cases	13
4.1	Use cases - Android	13
4.2	Use Cases - MEAN	15
5	Diagrammen	19
5.1	Android App Class Diagram	19

5.2	Deployment Diagram	20
5.3	Entity Relationship Diagram	20
5.4	Structuur MEAN	22
6	Best Practices	23
6.1	Android	25
6.2	MEAN	26
7	Analyse	29
7.1	Sprints	29
	Bibliografie	41

1. Inleiding

De opdracht van het project bestaat eruit dat een oplossing dient gevonden te worden om mensen aan de hand van uitdagingen aan te zetten tot een veganistische levensstijl. Dit behoort behaald te worden door 21 dagen lang, elke dag, een uitdaging te presenteren aan een gebruiker. Zodat deze gemotiveerd en gestimuleerd wordt om hieraan deel te nemen. De reden dat dit 21 dagen moet zijn is omdat mensen, volgens EVA (2017), na deze periode van iets een gewoonte maken in de plaats van een opdracht.

Als oplossing voor deze onderneming wordt een Android applicatie ontwikkeld alsook een online dashboard geschreven met behulp van de MEAN technologieën, dit wordt verder besproken in de technische details van het respectievelijke hoofdstuk.

Verder bestond de opdracht eruit dat de voornamelijkste doelgroep vrouwen zijn, die instaan voor de bereiding van het eten in een huishouden. Maar dit betekent niet dat andere doelgroepen uitgesloten mogen worden. De oplossing moet dus breed aanvaardbaar zijn voor zo veel mogelijk partijen. Ook zal de voertaal van de applicaties Nederlands zijn. Maar met de mogelijkheid tot uitbreiding naar andere talen.

2. Plan van aanpak

Op dinsdag 4 juli kwam het team fysiek samen om de opzet van de oplossing te bespreken. Bij deze meeting werd gebrainstormd welke richting we willen uitgaan en hoe dit zal worden geïmplementeerd. Er werd begonnen met het bekijken van de online video door EVA (2017) en notities te nemen bij belangrijke punten die aangehaald werden door de vertegenwoordiger van Eva. Hierdoor kwam elk teamlid aan een eigen lijst en werd op basis daarvan iedereen zijn input besproken en vergeleken.

Op basis van een gezamenlijk verkregen lijst werd begonnen aan een groot aantal user stories en use cases die in de app zouden kunnen komen. Een groots idee dat daaruit naar voor kwam was een soort spel te implementeren waarbij mensen per regio kunnen proberen om de regio te zijn met de hoogste veganistische score van België. Na verder te beraadslagen hieromtrent kwamen we tot de conclusie dat dit gezien de tijdspanne, en ook het feit dat niemand binnen het team de design skills had om zoiets te realiseren, er toch een andere aanpak zal moeten volgen.

Er werd altijd in het achterhoofd gehouden dat dit ook een applicatie moet worden die aanspreekbaar is voor een heel breed publiek. Uiteindelijk kwamen we uit op drie grote delen. De applicatie zal bestaan uit de uitdagingen, waarbij de vooruitgang van de gebruiker zal gevolgd worden. Elke vijfde uitdaging zal ook een beloning bevatten die de gebruiker kan innen als hij de opdracht tot een succesvol einde weet te brengen. Het tweede hoofddeel bestaat uit een lijst met restaurants zodat iemand die geen zin heeft om deel te nemen aan de opdrachten toch goede locaties kan vinden om veganistisch te gaan eten.

Het derde deel, en ons paradepaardje, genaamd Vegagram is een soort van sociale media geïmplementeerd in de applicatie. Gebruikers zullen de mogelijkheid hebben tot het nemen van afbeeldingen en deze online te zetten. Dan hebben ze de keuze of ze deze publiek

beschikbaar wensen te maken, zodat andere gebruikers de afbeeldingen kunnen zien. Op de publiek beschikbare posts kan men ook aantonen wat ze van de afbeelding vinden door middel van een "vind ik leuk"knop.

Bij al dit hoort ook een beheerderskant deze staat online en is beschikbaar via volgende link (Dashboard, 2017). Dit staat toe om binnen EVA vzw alles omtrent de app te beheren. Dit omvat aanpassen, updaten, verwijderen en toevoegen van alles binnen de app. Hiervoor wouden we gaan voor een dashboard look and feel. Die intuïtief en eenvoudig te gebruiken is. Zo kan een administrator inloggen op het dashboard om daar, bijvoorbeeld, de lijst van restaurants aan te passen of een nieuwe toe te voegen. Voor een duidelijk overzicht te garanderen tonen we kleine gepagineerde lijsten met filter opties en duidelijke knoppen die toelaten de gewenste actie uit te voeren. De best practices en schema's met betrekking tot de beheerderskant zijn terug te vinden in de technische beschrijving.

3. Applicatie

GITHUB REPOSITORY: <https://github.com/BramVannevel/eva2017groepH4>

Onze github repository bestaat uit twee subrepositories, genaamd Android en MEAN. In dit hoofdstuk overlopen we bondig de architectuur van beide applicaties.

3.1 Android

Als men de mappenstructuur van de Android applicatie bekijkt is direct duidelijk wat waar thuishoort. Dit is al één van de best practices die gehanteerd wordt binnen dit project. De structuur ziet er als volgt uit:

- Adapters
- Data
- Factory
- Helper
- Interfaces
- Model
- Views
 - Fragments
 - Activities

Verder wordt binnen de Android applicatie gebruik gemaakt van een SQLite databank voor het opslaan van de gebruiker zijn vooruitgang. En wordt gebruik gemaakt van zeer gangbare en goed gedocumenteerde thrid party libraries, zoals Retrofit, OkHTTP, RXJava, Recyclerview, Cardview, etc.

3.2 MEAN

Voor de beheerderskant maken we gebruik van de MEAN-stack. Dit betekent dus dat er gebruik gemaakt wordt van een MongoDB, Express middleware, Angular Frontend en Node als runtime builder.

de mappenstructuur van het dashboard ziet er zo uit:

- controllers
- css
- factoriesAndServices
- filters
- img
- modals
- pages
- server

4. Use Cases

In dit hoofdstuk wordt besproken welke use cases het team heeft geïmplementeerd voor zowel de beheerderskant, als de Android applicatie.

4.1 Use cases - Android

Registreer en Log in

Aangezien de app met gebruiker gegenereerde content werkt heeft de app de mogelijkheid tot inloggen en registreren. Als gebruiker kan ik inloggen indien ik al een account heb of registreren in de app als ik dit wens. Eens ingelogd blijft de gebruiker ook ingelogd, zo moet iemand niet altijd inloggen als hij wenst gebruik te maken van de app. Er wordt ook feedback gegeven bij foutmeldingen van zowel de login als de registreer.

Use case registreer

1. Als gebruiker wens ik een nieuw account aan te maken.
2. Als gebruiker wens ik direct ingelogd te zijn na registratie.
 - (a) Als gebruiker wil ik feedback zien als het registreren mislukt is.
3. Als gebruiker wens ik ingelogd te blijven na registratie.

Use case registreer

1. Als gebruiker wens ik in te loggen.
 - (a) Als gebruiker wil ik feedback zien als het inloggen mislukt is.
2. Als gebruiker wens ik ingelogd te blijven.

Restaurants

De eerste Use case waar het team aan begonnen is, is het kunnen opzoeken van vegetarische restaurants. Hierbij moeten ze een korte beschrijving zien van het restaurant, de naam en locatie met de mogelijkheid tot het openen van Google maps. Hier gebruiken we een master - detail flow voor. Zo ziet de gebruiker een overzichtelijke lijst van de restaurants en kan doorklikken tot een detail scherm. De, verkorte, use case is dus als volgt.

1. Als gebruiker wens ik een lijst van restaurants te zien.
2. Als gebruiker wens ik meer details te zien van een specifiek restaurant.
3. Als gebruiker wens ik te kunnen navigeren naar een restaurant..
 - (a) als gebruiker wens ik dit in een dual pane layout te zien.

Challenges

Het hoofddoel van de app is mensen aanzetten tot vegetarisch eten door middel van 21 dagen lang een aantal uitdagingen te vervullen met betrekking tot een vegetarische levensstijl. Dit lost het team op door dagelijks een uitdaging aan te bieden aan de gebruiker, die deze dan kan starten. Ook is er de mogelijkheid tot een andere uitdaging door middel van een "ik wil een andere uitdaging" knop. Dit geeft de gebruiker wat meer vrijheid en kan zo een uitdaging die hij/zij minder ziet zitten vervangen door een andere. Het is wel zo dat de uitdagingen 21 opeenvolgende dagen moeten zijn. Vervolledigt hij één dag geen uitdaging moet de gebruiker opnieuw beginnen. Zo zetten we ze toch aan elke dag zelf maar gewoon eens te kijken. Een gebruiker kan ook hulp krijgen bij een uitdaging. Of als er geen uitdaging is die hij/zij wenst uit te voeren een optie tot te skippen. Het hoofdscherm toont alle dagen die al vervulledigd zijn tot nu toe, in kalender vorm. Ook ziet de gebruiker zijn vooruitgang tot de volgende beloning. De, verkorte, use case is als volgt.

1. Als gebruiker wens ik de uitdaging te bekijken van vandaag.
2. Als gebruiker wens ik meer uitleg te zien bij de uitdaging.
3. Als gebruiker wens ik de uitdaging te starten.
 - (a) als gebruiker wens ik een andere uitdaging te krijgen.
 - (b) als gebruiker wens ik de huidige uitdaging te stoppen.
4. als gebruiker wens ik de moeilijkheidsgraad te zien van de huidige challenge.
5. als gebruiker wens ik mijn huidige progressie te zien door middel van een dag.
6. als gebruiker wens ik te zien wanneer mijn volgende beloning er is.

Vegagram

Ons paradepaardje, en persoonlijke toevoeging aan het idee van uitdagingen is een sociaal media gebaseerde foto-deel activiteit. Hierbij kunnen gebruikers foto's nemen in app, deze bekijken of hij al dan niet geslaagd is om dan te beslissen hem te uploaden naar vegagram, niet up te loaden en terug te keren of opnieuw een foto te trekken. Als er een foto geupload wordt moet de gebruiker kiezen of hij deze foto publiek wenst of enkel privé. Indien deze publiek is kunnen alle andere gebruikers van Vegagram de foto bekijken en "vind

ik leuk-en. Ook kunnen gebruikers hun eigen afbeeldingen delen op Facebook met een omschrijving bij. Dit heeft als hoofddoel het aanzetten van vegetarisch eten door sociale druk.. Personen kunnen zo alles delen op Facebook en andere individuen aanzetten tot het gebruik van de app of een veganistische mindset. onderstaand de verkorte use case.

1. Als gebruiker wens ik alle posts in Vegagram te zien
2. Als gebruiker wens ik een afbeelding leuk te vinden
 - (a) als gebruiker wens ik een afbeelding niet meer leuk te vinden.
3. Als gebruiker wens alle beschikbare info te zien met betrekking tot de afbeelding.
4. als gebruiker wens ik zelf een post te maken.
5. als gebruiker wens ik zelf een afbeelding te uploaden.
 - (a) als gebruiker wens ik de genomen afbeelding te annuleren.
 - (b) als gebruiker wens ik de genomen afbeelding te annuleren en een nieuwe te maken
6. als gebruiker wens ik mijn eigen post te zien in Vegagram.

4.2 Use Cases - MEAN

Log in

Aangezien de webapplicatie beheerders nodig heeft die content uploaden, controleren, wijzigen,... is er de mogelijkheid voor een beheerder om in te loggen. Deze login is niet verbonden aan de android applicatie, aangezien een registratie in de android app de rol user oplevert. Als deze login mislukt wordt er feedback gegeven. De, verkorte, use case is als volgt.

1. Als beheerder wens ik in te loggen.
 - (a) als beheerder wil ik feedback zien als het inloggen mislukt is.
2. Als beheerder wens ik ingelogd te blijven.
 - (a) Als beheerder wens ik uit te loggen.

Restaurants

Indien we in de android app een lijst willen tonen van de vegetarische restaurants moeten we deze lijst voorzien door middel van een webapplicatie. Enkel een beheerder kan restaurants toevoegen aan de lijst, voorzien van een naam, telefoonnummer en adres(straat, huisnr, postcode, stad). Verder kan een beheerder ook restaurants verwijderen uit de lijst, of aanpassen. De, verkorte, use case is als volgt.

1. Als beheerder wens ik een lijst te zien met de verschillende restaurants.
 - (a) als beheerder wens ik te filteren in de lijst
2. Als beheerder wens ik een restaurant toe te voegen.
3. Als beheerder wens ik een restaurant te wijzigen.
4. Als beheerder wens ik een restaurant te verwijderen.
 - (a) Als beheerder wens ik punt 2, 3 en 4 te kunnen annuleren.

Challenges

Om gebruikers 21 dagen lang vegetarisch te laten eten maken we gebruik van challenges die een extra aanmoediging zijn om elke dag opnieuw voor vegetarische maaltijden te kiezen. Als een beheerder in de webapplicatie ingelogd is wenst hij zich te kunnen navigeren naar de challenges, en hiervan een lijst te zien. Enkel een beheerder kan een challenge toevoegen, voorzien van een titel, omschrijving, dag. En optioneel: een restaurant, gerecht en reward. challenges kunnen ook verwijderd of gewijzigd worden. De, verkorte, use case is als volgt.

1. Als beheerder wens ik een lijst te zien met de verschillende challenges.
 - (a) als beheerder wens ik te filteren in de lijst
2. Als beheerder wens ik een challenge toe te voegen.
3. Als beheerder wens ik een challenge te wijzigen.
4. Als beheerder wens ik een challenge te verwijderen.
 - (a) Als beheerder wens ik punt 2, 3 en 4 te kunnen annuleren.

Categorieën

Aangezien we met gerechten werken wensen we deze te kunnen onderverdelen in verschillende categorieën. Een ingelogde beheerder moet een lijst van de categorieën kunnen raadplegen, hier kan hij dan een categorie aan toevoegen of verwijderen. De, verkorte, use case is als volgt.

1. Als beheerder wens ik een lijst te zien met de verschillende categorieën.
 - (a) als beheerder wens ik te filteren in de lijst
2. Als beheerder wens ik een categorieën toe te voegen.
3. Als beheerder wens ik een categorieën te wijzigen.
4. Als beheerder wens ik een categorieën te verwijderen.
 - (a) Als beheerder wens ik punt 2, 3 en 4 te kunnen annuleren.

Gerechten

Als we mensen willen aanzetten om vegetarisch te eten hebben we natuurlijk vegetarische gerechten nodig, deze kunnen dan omvat worden in een challenge om de gebruikers extra te motiveren. Na het inloggen kan een beheerder zich navigeren naar een pagina waar hij een lijst met gerechten te zien krijgt. De beheerder kan dan kiezen om een gerecht toe te voegen, voorzien van een naam, categorie, mogelijke allergenen, en een omschrijving. Er kan ook gekozen worden om een gerecht te verwijderen of te wijzigen. De, verkorte, use case is als volgt.

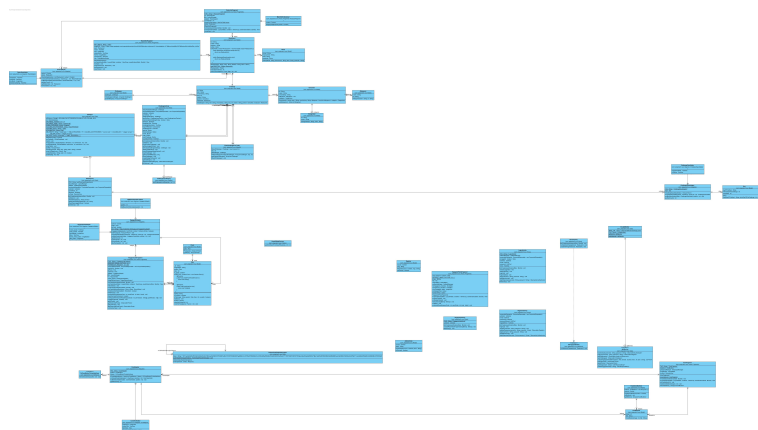
1. Als beheerder wens ik een lijst te zien met de verschillende gerechten.
 - (a) als beheerder wens ik te filteren in de lijst
2. Als beheerder wens ik een gerechten toe te voegen.
3. Als beheerder wens ik een gerechten te wijzigen.
4. Als beheerder wens ik een gerechten te verwijderen.

- (a) Als beheerder wens ik punt 2, 3 en 4 te kunnen annuleren.

5. Diagrammen

5.1 Android App Class Diagram

In dit hoofdstuk zijn alle diagrammen terug te vinden met betrekking tot de gebruikte applicaties. Als eerste is het klassendiagram terug te vinden van de Android applicatie. Daarna het deployment diagram, en als laatste het ERD van Mongo. Van de MEAN stack kan niet echt een "klassen"diagram gemaakt worden aangezien dit een module based applicatie is en geen Object Oriented applicatie. De structuur zal beschreven worden als laatste punt.

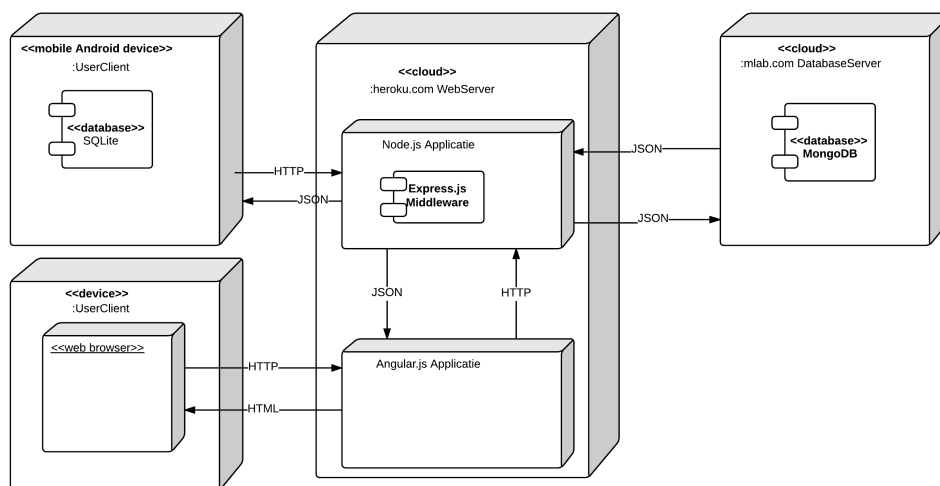


Figuur 5.1: Eva Applicatie Class Diagram

een grote afbeelding is terug te vinden op de GitHub repository via volgende link:

5.2 Deployment Diagram

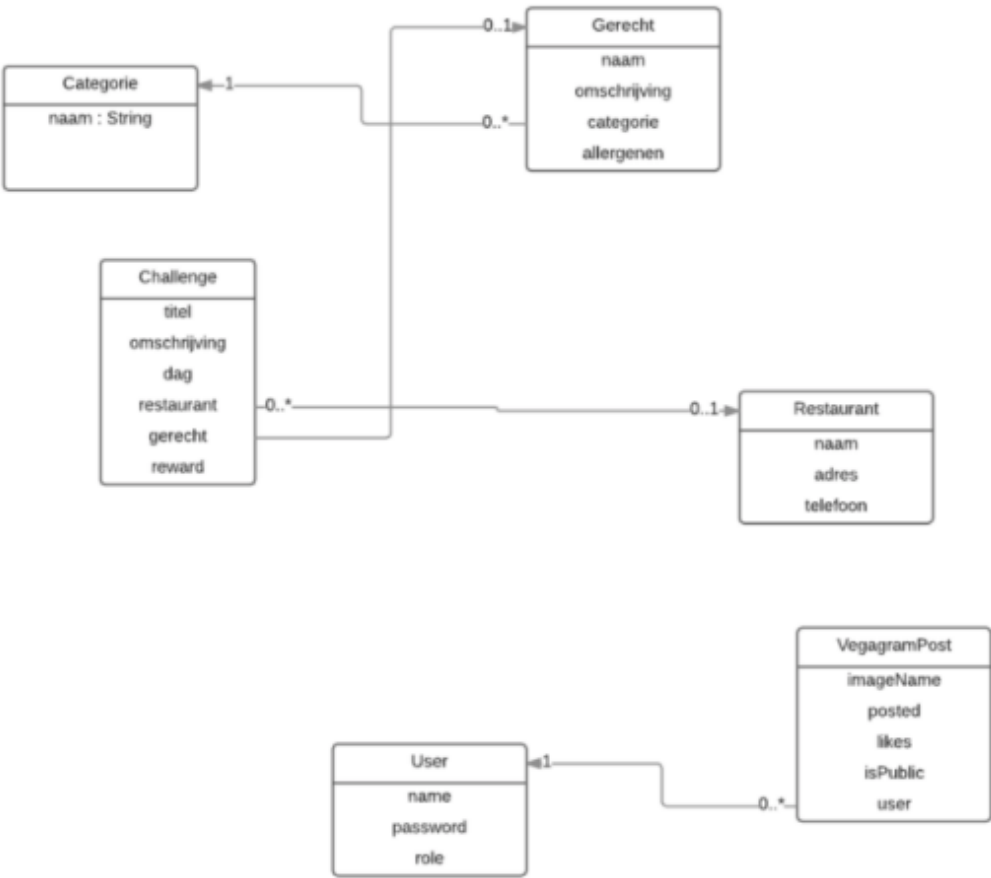
Volgende afbeelding toont de deployment diagram aan van de workflow van zowel de Android applicatie als de MEAN applicatie.



Figuur 5.2: Deployment Diagram

5.3 Entity Relationship Diagram

De volgende afbeelding toont aan hoe het ERD er uit ziet. Waar aan duidelijk te zien is wat de databank van SQLite bevat, alsook de MongoDB.



Figuur 5.3: Entitiy Relationship Diagram

De SQLite databank bevat maar één klasse die er als volgt uit ziet



Figuur 5.4: SQLite ERD

5.4 Structuur MEAN

De structuur van de MEAN applicatie bestaat uit de gevolgde best practices, zo hebben we volgende controllers

- RestaurantDetailModalController
- RestaurantModalController
- authController
- categorieModalController
- challengeController
- challengeDetailModalController
- challengeModalController
- gerechtDetailModalController
- gerechtModalController
- restaurantsController

Deze maken gebruik van de volgende Factories en services

- authService
- beheer-factory
- challenge-factory
- restaurants-factory

Dit alles werkt samen om alles te genereren dat getoond wordt in de html van de volgende pagina's

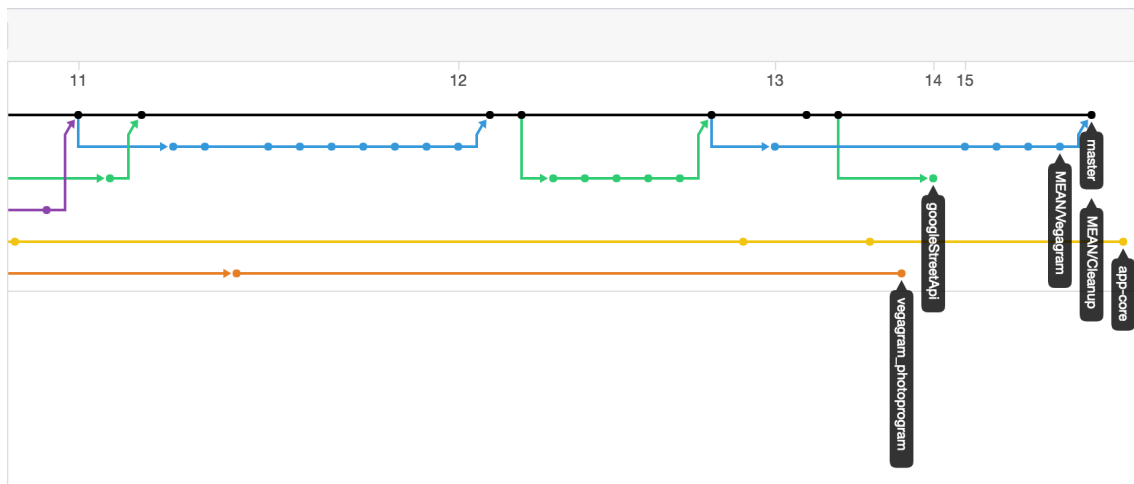
- allergenen
- categorieën
- challenges
- gerechtBeheer
- restaurants

en volgende modals

- categorieModal
- challengeDetailModal
- challengeModal
- gerechtenDetailModal
- gerechtModal
- restaurantModal
- restaurantDetailModal

6. Best Practices

In dit hoofdstuk zijn alle gehanteerde best practices terug te vinden van de twee applicaties. Algemene best practices zijn het gebruik van Github en daarbinnen werken met Git Flow. Zo hebben we voor alle belangrijke use cases een aparte branch aangemaakt die pas op de master branch mogen na dat iemand anders de pull request reviewed.




Als men surft naar: <https://github.com/BramVannevel/eva2017groepH4/pulls?utf8=%E2%9C%93&q=is%3Apr%20is%3Aclosed%20> zijn ook alle gesloten pull requests zichtbaar.

En andere best practice, wat ook aangehaald zal worden in het hoofdstuk met betrekking tot de analyse is onze communicatie. Voor teamcommunicatie hebben we gebruik gemaakt van de handige tool Slack. Hiermee kunnen teams eenvoudig communiceren met elkaar. Slack staat ook toe dat men andere tools integreerd. Zo hebben wij Trello geïntegreerd in Slack waardoor iedereen kan zien wanneer een ticket klaar is. Ook GitHub werd toegevoegd

zodat daar direct duidelijk is wanneer een pull request of code review gevraagd werd door een teamlid.

De Trello updates zagen er zo uit


Yesterday



Trello APP 9:21 AM

- Card moved: "ANDROID: implement calendar like features in the core" from list "to do" to list "done"
- Card moved: "ANDROID: make use of refactoring tools like Android lint/Android monitor to look for errors" from list "to do" to list "done"
- Card moved: "ANDROID: start implementing the REST api's from the backend now that it's online" from list "to do" to list "done"

Today




Trello APP 9:17 AM

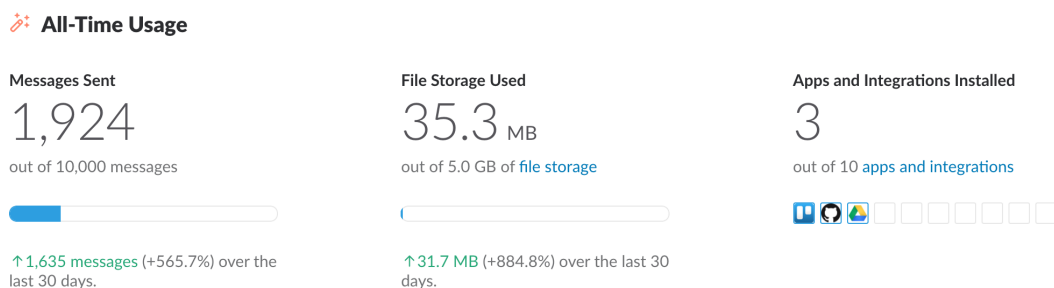
- Card moved: "WEB: implement a backend that is capable of receiving images and returns the url" from list "in progress" to list "done"
- Card moved: "ANDROID: start to combine every branch" from list "to do" to list "done"
- Card moved: "ANDROID: update the UI to fit with the backend (eva UI)" from list "to do" to list "done"

+

Message #trello



Binnen slack is het ook eenvoudig om de statistieken te bekijken, zoals hieronder getoond.



Als laatste hebben we ook de scrum en agile principes gerespecteerd en hebben we een scrummaster aangewezen en in iteraties van één week gewerkt per sprint. Dit is allemaal terug te vinden onder het hoofdstuk met betrekking tot de Analyse.

6.1 Android

Binnen Android zijn er veel best practices en normen die gehanteerd en gevolgd kunnen worden om een goede Look and Feel te garanderen voor een app. Zoals het material design. Ook op implementatieniveau zijn er belangrijke verschillen tussen een goede, onderhoudbare app en een minder goede.

De belangrijkste best practice die achter de schermen gebeurt, maar direct vertaalt naar de user experience van de app is dat alles dat netwerk en databank gerelateerd is asynchroon gebeurt. Voor de netwerk communicatie bekomen we dit door gebruik te maken van OkHttp, Retrofit en RxJava. Dit zorgt voor een asynchrone manier van data ophalen dat perfect in overeenkomst is met de Android Lifecycle van activities en fragments. De reden dat RxJava hierin betrokken wordt, kan het best aangetoond worden aan de hand van onderstaand voorbeeld.

Als een gebruiker de app opstart en nieuwe uitdagingen ophaalt kan het zijn dat, bijvoorbeeld door het roteren van het apparaat of het openen van een andere applicatie, de activity die de data wil opvragen er niet meer is. Indien men enkel Retrofit gebruikt zal dit leiden tot een crash. Want de asynchrone bevraging zal de data proberen afleveren aan een activity die er niet meer is. Door gebruik te maken van RxJava kunnen we effectief werken met een observer pattern en kunnen activities hun subscriben op data, en als ze destroyed worden unsubscribe zodat de data weet dat ze daar niet naar toe kunnen. Ook heeft RxJava enkele methodes, zoals defer, die kunnen ingeschakeld worden om enkel de data te beginnen opvragen als het effectief nodig is.

Een tweede best practice is het gebruik van een SQLite databank om de gebruiker zijn vooruitgang bij te houden. Zo kan de gebruiker ook offline zijn challenges en vooruitgang bekijken.

Als derde code best practice wordt ook gebruik gemaakt van instancebundles om opgehaalde data in de activity te houden zolang die niet destroyed is. Zo hoeft de applicatie niet altijd een netwerk call te maken als de gebruiker verwisselt van schermen om daarna terug te keren naar het eerste scherm. Zoals bijvoorbeeld het geval is in de Vegagram activity om de posts bij te houden.

Ook wordt veel gebruik gemaakt van fragments zodat een tablet versie ook een makkelijk te implementeren mogelijkheid is. Verder rond UI wordt ook veel gebruik gemaakt van de (relatief) nieuwe ConstraintLayout omdat deze veel performanter is dan RelativeLayout en aanverwanten. Voor het tonen van lijsten gebruikt de applicatie RecyclerViews met als rij een cardview, die opgevuld worden aan de hand van het viewholder patroon. Voor creaties met meerdere mogelijkheden gebaseerd op een input werden meerdere factories geïmplementeerd.

een overzicht van gebruikte patterns:

- Factory pattern
- Observer pattern

- Viewholder pattern
- Singleton pattern
- Builder pattern

kleine, en niet gebruikersgevoelige info wordt opgeslaan in de shared preferences zodat die snel en eenvoudig overal aanspreekbaar zijn, bijvoorbeeld een Json Web Token (jwt).

Zoals eerder aangehaald is de mappenstructuur ook zeer intuïtief en makkelijk uitbreidbaar bijvoorbeeld om bepaalde classes terug te vinden. Een meer algemene code best practice is ook dat de code zo geschreven is dat ze makkelijk uitbreidbaar en aanpasbaar is. Door middel van factories, interfaces en design keuzes. Een voorbeeld van zo een design keuze is het startscherm. Hier zien we vier knoppen die in een grid terug te vinden zijn. Initieel werden die geïmplementeerd door middel van een TableLayout en dan vier maal hard gecodeerde knoppen in cardviews toegevoegd, maar na een pull request en analyse kwamen we tot het besef dat dit niet uitbreidbaar nog onderhoudbaar is. In de app wordt nu gebruik gemaakt van een RecyclerView met dynamisch gegenereerde knoppen gebaseerd op één klasse die ze kent. Stel dat EVA nog functionaliteit wenst toe te voegen of extra knoppen kan dit op één plaats en wordt alles overal aangepast.

Een laatste best practice is het feit dat er gebruik gemaakt werd van de resource bundles. Zo zitten alle strings in de string resource alsook alle colors. De gebruikte iconen binnen de applicatie zijn afkomstig van de vectors die terug te vinden zijn en gratis te gebruiken, onder de Apache Licentie. Dit heeft als voordeel dat elk icoon op elke smartphone er goed uit ziet.

Binnen Android werken we ook veel met afbeeldingen, zoals de afbeeldingen die kunnen getrokken worden en de Vegagram Posts die getoond worden van andere gebruikers die publiek zijn. De best practice, zoals beschreven in onderstaande sectie, bestaat uit het feit dat geen afbeeldingen direct mee komen via de GET calls. We slaan de afbeeldingen op op de server, en sturen een url naar de afbeelding mee met de call. Dit bespaart tijd (lengte van de duur van de call), geheugen en data van de gebruiker zijn toestel. Daarna kunnen de foto's ingelezen worden via libraries zoals Glide of Picasso en zo door.

6.2 MEAN

Ook voor het webgedeelte werd rekening gehouden met best practices. Zo wordt er gebruik gemaakt van Webpack om alle statische resource bestanden te bundelen. Om de CSS overzichtelijk te houden werd er gebruik gemaakt van Sass. Binnen angular werd de functionaliteit netjes opgesplitst aan de hand van controllers, factories en services en wordt voor de navigatie tussen de verschillende paginas gebruik gemaakt van angular-ui-router. Om de verschillende endpoints voor de netwerkooperaties overzichtelijk te houden, werd er gebruik gemaakt van een individuele module met constanten. Ook met security werd rekening gehouden. Zo maakt het webgedeelte gebruik van een JSON Web Token voor de authenticatie en worden alle wachtwoorden eerst gehashed alvorens ze worden gepersisteerd. Om ervoor te zorgen dat enkel admins kunnen inloggen in de webapplicatie,

werd gebruik gemaakt van gebruikersrollen. Enkel gebruikers met de rol 'admin' kunnen succesvol inloggen in het webgedeelte. Ook wie van buitenaf (bijvoorbeeld via postman) een HTTP request stuurt naar een endpoint dat enkel bedoeld is voor admins, krijgt indien hij dit doet met een account met de rol 'user' een 403 Unauthorized response. Op deze manier blijven alle gegevens binnen het bereik van de juiste personen.

Voor het mongoDB gedeelte hebben we, door dat we werken met afbeeldingen, ook rekening gehouden met de persistentie en efficiëntie van de databank. Zo worden afbeeldingen via een POST als binary opgeslagen op de server zelf (via /uploads/MulterNaamVanDeAfbeelding). Enkel de link naar de afbeelding wordt opgeslaan in de databank. Op die manier blijft deze snel en responsief. En is het mogelijk om de afbeeldingen te laden door gewoon de link te gebruiken. Dit heeft ook als voordeel dat er geen afbeeldingen mee gestuurd worden via de GET call van de vegagram posts. Dat zou te veel data verbruiken en te veel geheugen consumeren van de Android applicatie.

7. Analyse

In dit hoofdstuk wordt alles omtrent de analyse uitgelegd en beschreven.

Voor we begonnen aan de effectieve implementatie werd een backlog gemaakt en hebben we gebruik gemaakt van wekelijkse sprints, die starten op woensdag en eindigden precies zeven dagen later. Sander Brugge nam de rol op zich van scrummaster, hij besliste welke tickets die week werden opgenomen en dan kon onderling beslist worden wie wat deed. Ook besliste hij welke tickets de hoogste prioriteit hadden en wat zeker af moest zijn.

Elke week maakte hij een nieuwe sprint aan in Trello, en zorgde ervoor dat via burndown for Trello de vooruitgang kon bekeken worden en dat er een burndown gegenereerd werd. Elke woensdag op het einde van een sprint hielden we ook een retrospective, om te kijken wat goed ging, wat minder goed ging en wat we gaan doen. Deze zijn allemaal terug te vinden in volgende sectie.

Alle communicatie verliep via de professionele team communicatie tool Slack. Aangezien iedereen binne het team relatief ver van elkaar wonen. Voor quality assurance werd gebruik gemaakt van Pull Requests op Github. Waar iedereen commentaar kon schrijven, verbeteringen kon voorstellen of meer uitleg vragen over bepaalde stukken code.

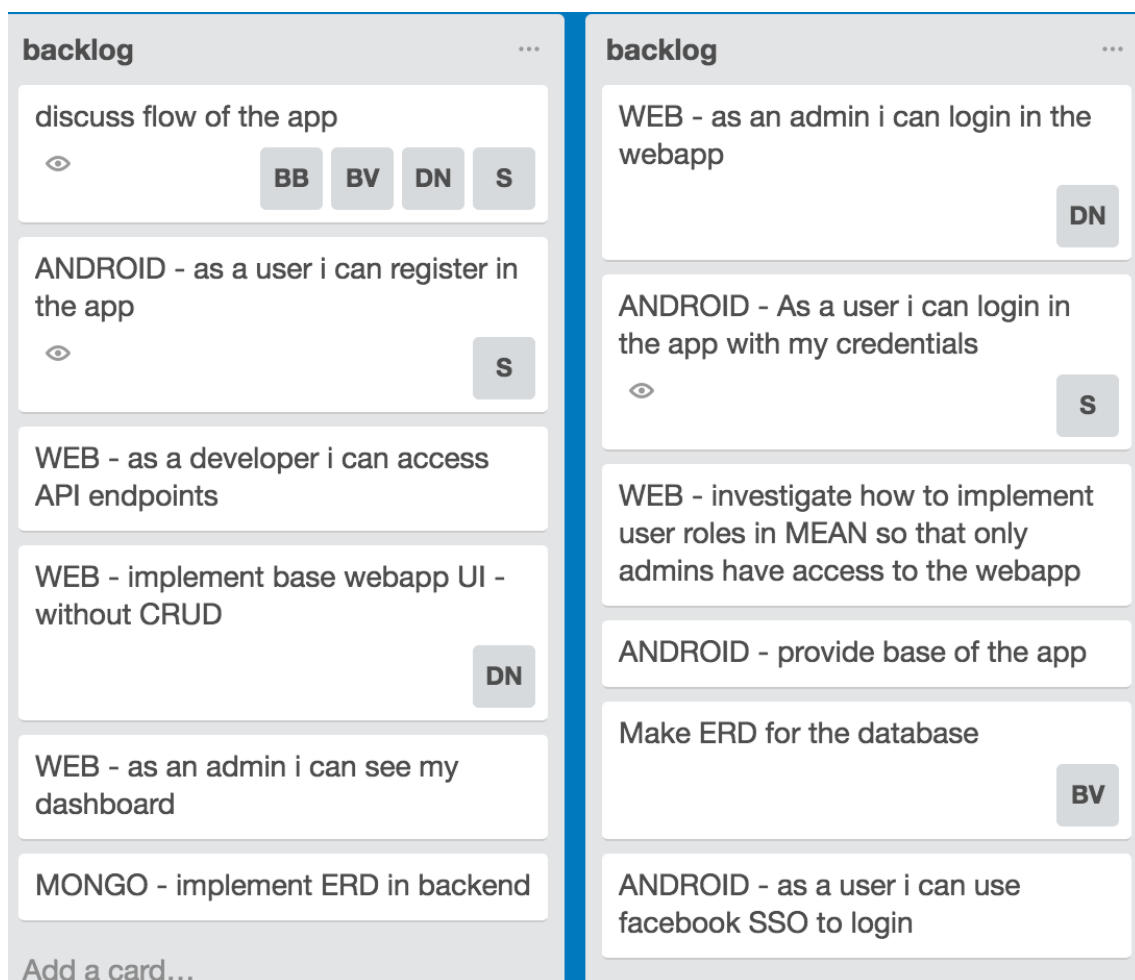
7.1 Sprints

in onderstaande sectie zijn de backlog, burndown en retrospective terug te vinden per sprint. De taken werden onderling verdeeld en besproken via Slack.

Sprint 1

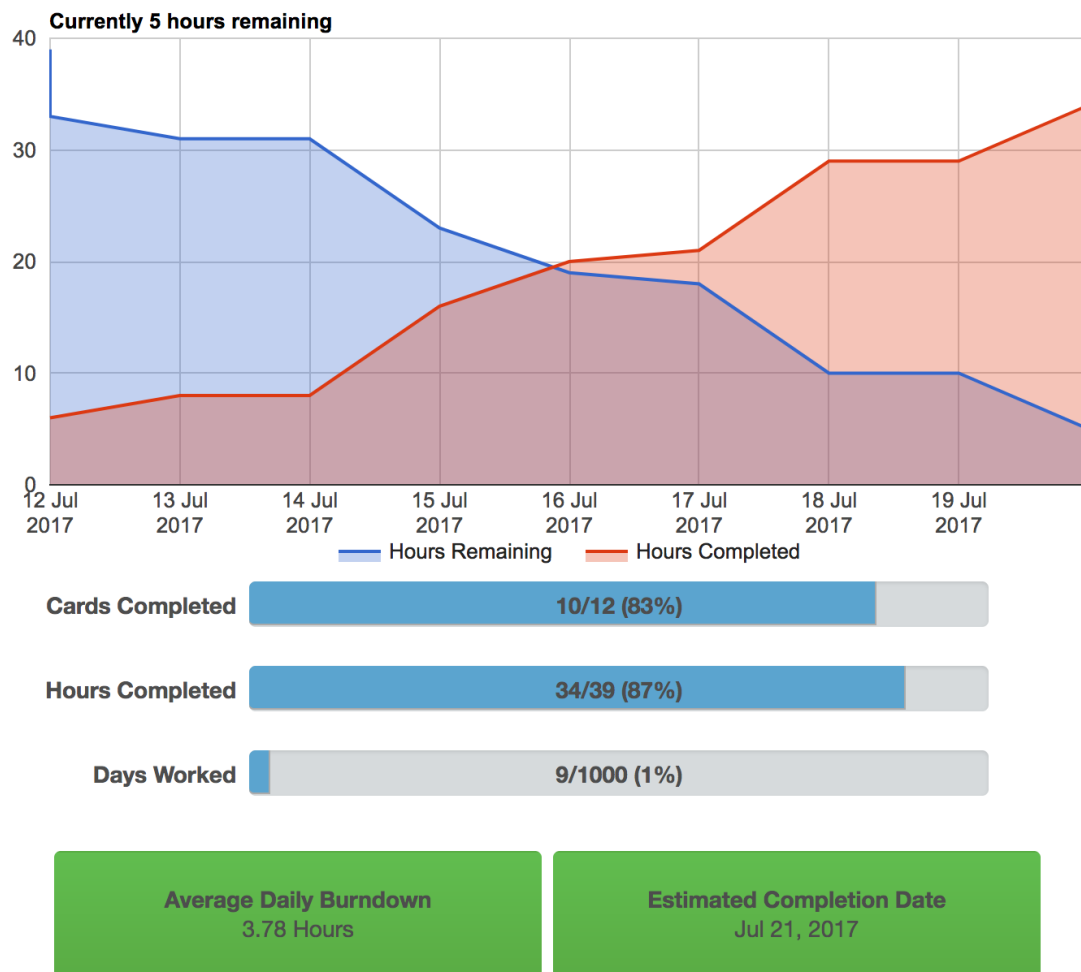
De eerste week hebben we fysiek samengezeten om de startup te bespreken en te discussiëren wat we van de app verwachten. Na de tijd te hebben genomen om de opdracht te bekijken en te beluisteren, kwamen we tot een algemeen idee. Zoals beschreven in het hoofdstuk plan van aanpak. Op basis van ons idee hebben we een backlog gemaakt met de belangrijkste use cases. Voor ons was het sociale aspect zeer belangrijk. Zo kan een gebruiker inloggen via Facebook en Vegagram posts delen op Facebook om andere mensen aan te zetten tot het gebruik van de app of een veganistische levensstijl.

De backlog van de eerste sprint. Hier is direct zichtbaar dat alles in de backlog gestoken wordt, zowel functionele requirements maar ook niet functionele. Een voorbeeld hiervan is het bespreken van de app. Ook hier moet tijd voor gerekend worden in de sprint.



Figuur 7.1: backlog sprint 1

De burndown is niet perfect omdat dit de eerste week was en er iets te veel hooi op de vork werd genomen.



Figuur 7.2: burndown sprint 1

Op het einde van een sprint vond ook altijd een retrospective klaar, daarvan is een kort verslag te vinden hieronder.

Sprint 1: What did we do good

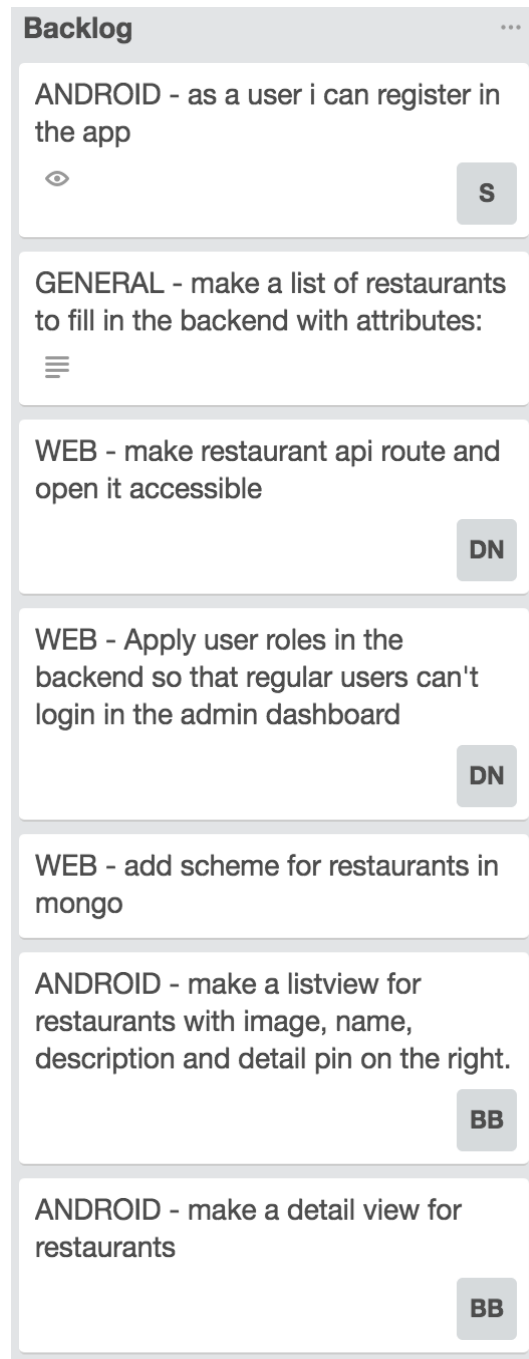
- Begonnen met goed idee
- Fysiek samen zitten om het plan te bespreken
- Vertrokken van uit de basis (ERD/goede afspraken)
- Agile werken
- Best practices respecteren van in het begin

What could we have done better

- Beter inplannen van de tickets
- Beter verdelen van taken
- Rekenen op meer tijd voor mooiere UI

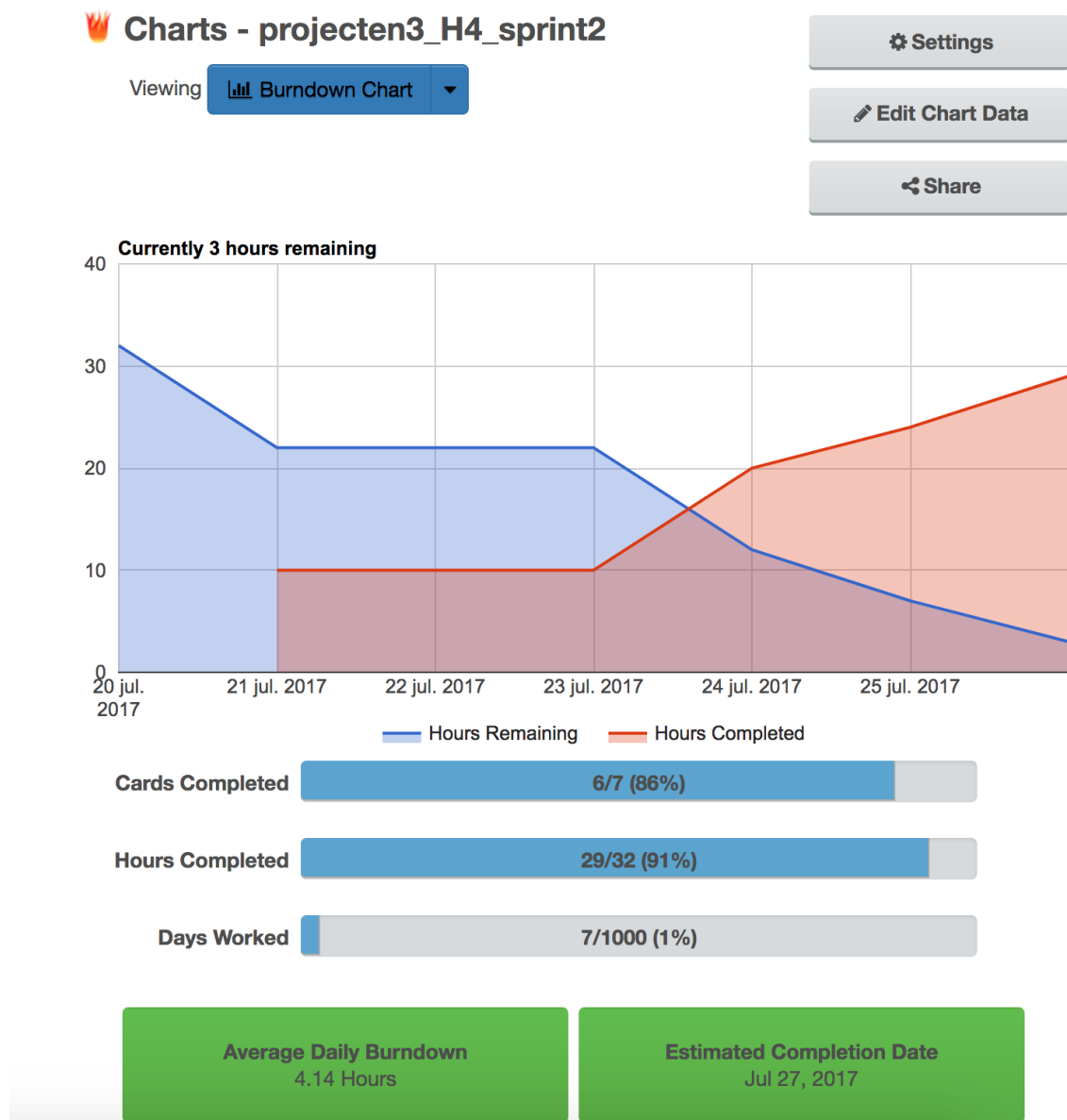
Sprint 2

De backlog van de tweede sprint. Dit ziet er op eerste zicht een veel kleinere backlog uit maar de tickets die beschreven zijn waren veel zwaarder en groter van omvang. Hier werd de basis van alles geïmplementeerd dus dit moest ook correct zijn van in het begin.



Figuur 7.3: backlog sprint 2

Dit ziet er ook al beter ingeschat uit dan de eerste sprint, we maken snel vooruitgang als team!



Figuur 7.4: burndown sprint 2

Ook hier was na het einde van deze sprint een retrospective, die gehouden werd via Slack.

Sprint 2:

What did we do good

- met branches werken
- pull requests om elkaar code te reviewen
- goed de sprint gepland
- goede communicatie dmv Slack

what could we have done better

- betere UI van in begin
- sommige zaken (zoals properties in pojo's) bespreken in slack ipv in commentaar in de code
- Javadoc toevoegen test

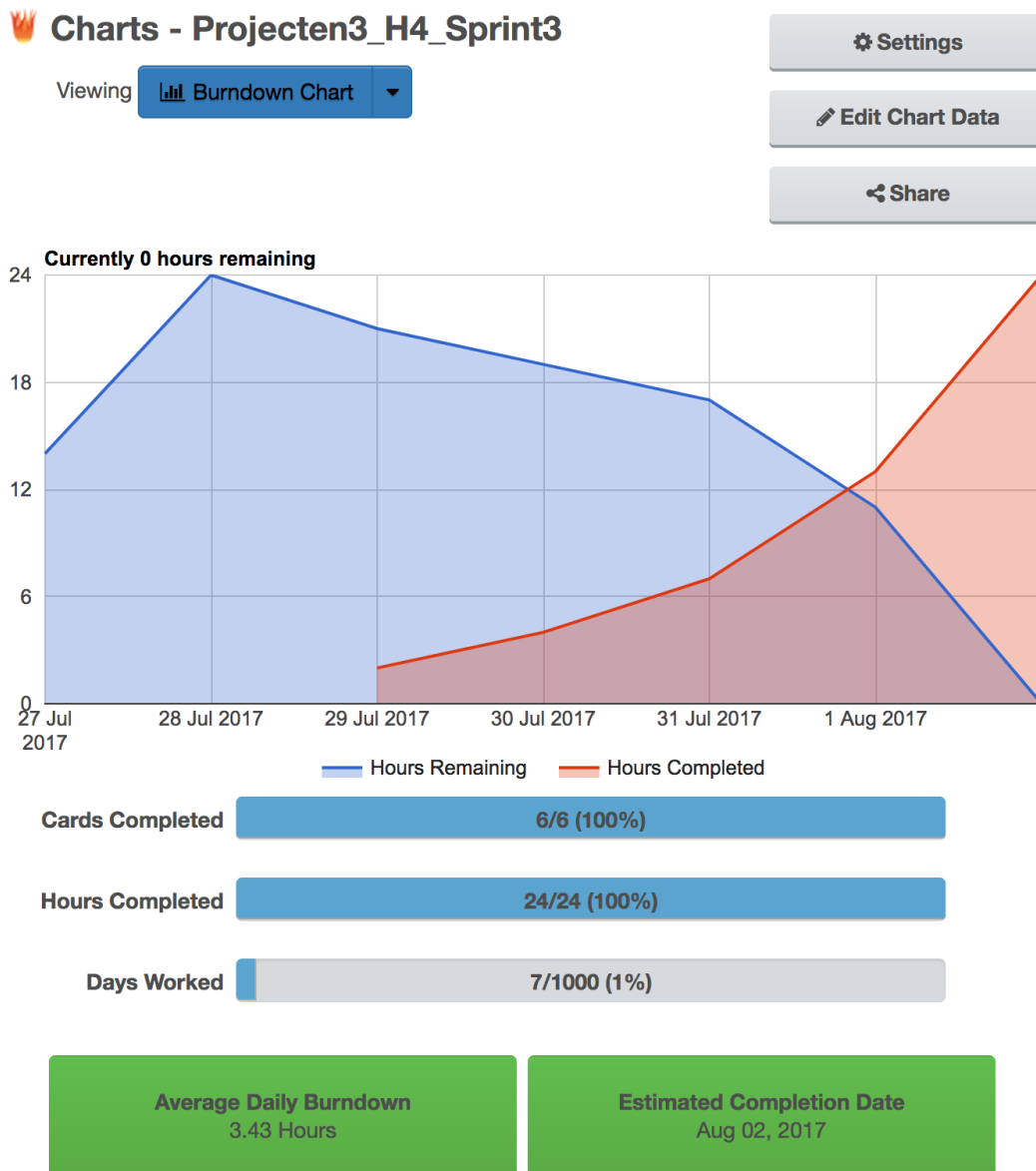
Sprint 3

De backlog van de derde sprint.



Figuur 7.5: backlog sprint 3

Zoals te zien aan de burndown van deze sprint hadden we te weinig opgenomen voor deze sprint. De scrummaster heeft dan beslist dat er extra tickets bij genomen mochten worden. Dit zorgt voor een piek in de grafiek.



Figuur 7.6: burndown sprint 3

Op het einde van de sprint werd opnieuw een retrospective gehouden. Dit zal aanhouden gedurende de volgende sprints ook.

sprint 3:

What did we do good

- beter rekening gehouden met de UI en javadoc
- coderefactoring van bij de start zodat dat later geen onduidelijke code wordt/blijft
- verder werken via git flow

What could we have done better

- UI testing
- unit testing
- goed webgedeelte en android app laten afstemmen qua properties test

Sprint 4

De backlog van de vierde sprint.

Deze sprint was perfect ingepland en is dan ook tot een goed einde geraakt.



Figuur 7.7: backlog sprint 4

Zoals te zien in de burndown ging het werk vlot. Dit is te zien aan de tickets die snel en consistent naar beneden gaan.



Figuur 7.8: burndown sprint 4

Ook hier wordt opnieuw een retrospective besproken.

Sprint 4:

What did we do good:

- goede communicatie voor backend/frontend
- coderefactoring van bij de start zodat dat later geen onduidelijke code wordt/blijft
- pair programming
- verder werken via git flow

What could we have done better

- UI testing
- unit testing test


Sprint 5

De backlog van de vijfde sprint.

done ...

WEB-bug: Fix a bug where if a user refreshes, his token is invalidated.

ANDROID: start to combine every branch

ANDROID: update the UI to fit with  the backend (eva UI)

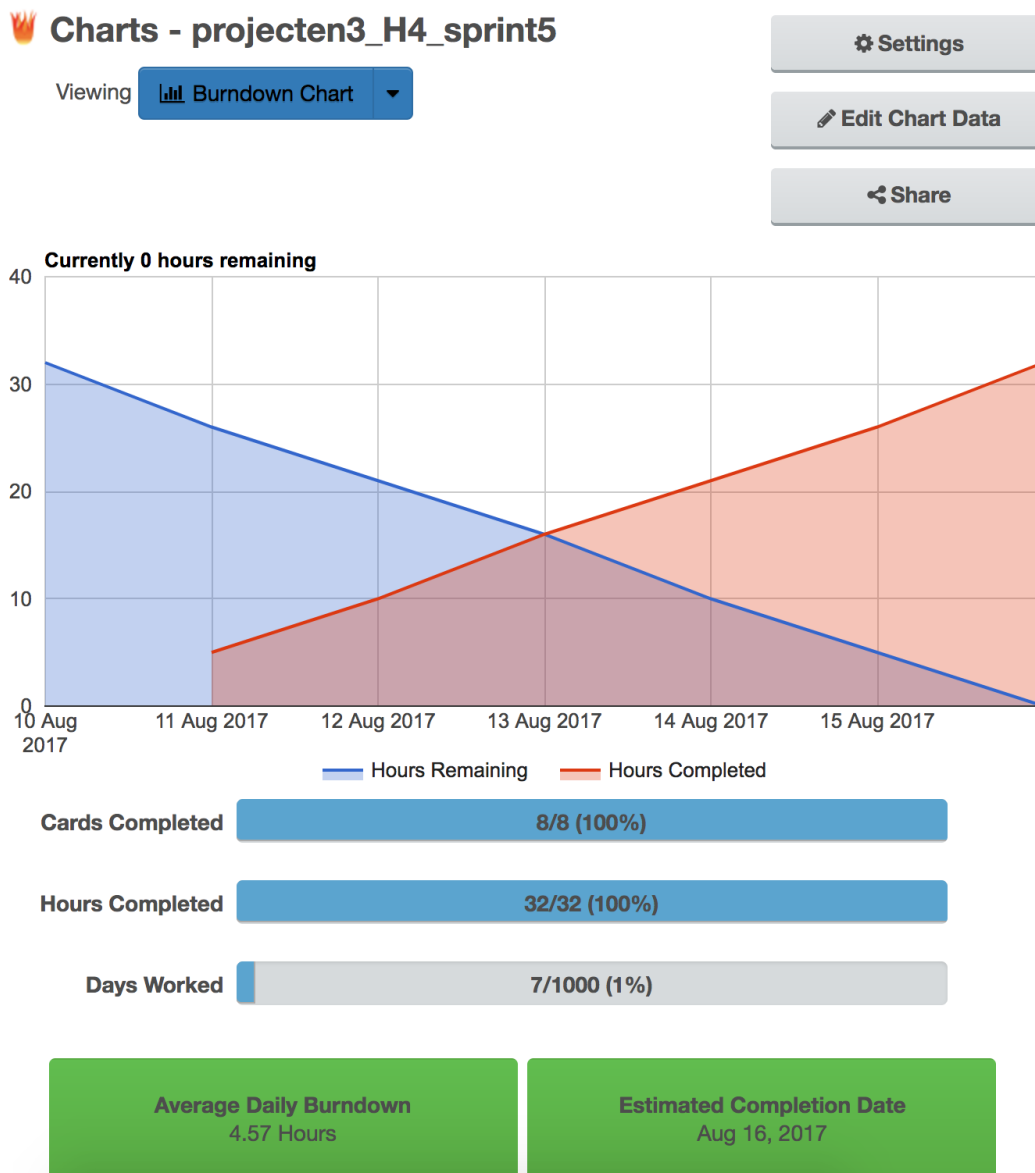
WEB: implement a backend that is capable of receiving images and returns the url

WEB: implement recipes REST api's

ANDROID: start implementing the REST api's from the backend now that it's online

ANDROID: make use of refactoring tools like Android lint/Android monitor to look for errors

ANDROID: implement calendar like features in the core



sprint 5:

What did we do good:

- duidelijke taakverdeling
- implementatie met best practices in het achterhoofdpair programming
- verder werken via git flow
- duidelijke communicatie voor design app in overeenkomst met de backend

What could we have done better

- UI design
- nog extra javadoc toevoegen
- beta testing

Bibliografie

Dashboard. (2017). Beheerderskant. Verkregen van <https://evabeheer.herokuapp.com/>

EVA. (2017). Opdracht Projecten 3. Verkregen van <https://www.youtube.com/watch?v=D6S3MS4Xh6c>

Lijst van figuren

5.1	Eva Applicatie Class Diagram	19
5.2	Deployment Diagram	20
5.3	Entitiy Relationship Diagram	21
5.4	SQLite ERD	21
7.1	backlog sprint 1	30
7.2	burndown sprint 1	31
7.3	backlog sprint 2	32
7.4	burndown sprint 2	33
7.5	backlog sprint 3	34
7.6	burndown sprint 3	35
7.7	backlog sprint 4	36
7.8	burndown sprint 4	37

Lijst van tabellen