|  |
| --- |
| IT Project - Einddocument  The Challenge Reisorganisatiesysteem v1.0 |
| **Opleiding: Programmeren**  **Academiejaar: 2020-2021**  **Voornaam en Naam student(en): Guning Deng**  **Voornaam en Naam projectbegeleiders: **Van Camp Vincent**** |

AP.BE

Inhoud

[IT Project - Einddocument 1](#_Toc72354031)

[The Challenge Reisorganisatiesysteem v1.0 1](#_Toc72354032)

[Inhoud 1](#_Toc72354033)

[Einddocument IT Project 2](#_Toc72354034)

[The challenge reisorganisatiesysteem v1.0 2](#_Toc72354035)

[1 Projectbeschrijving 2](#_Toc72354036)

[1.1 Projectinhoud 2](#_Toc72354037)

[1.2 Gebruikershandleiding 2](#_Toc72354038)

[1.3 Technische documentatie 2](#_Toc72354039)

[2 Projectrealisatie 3](#_Toc72354040)

[2.1 Linken en referenties 3](#_Toc72354041)

[2.2 Front-end 3](#_Toc72354042)

[2.3 Back-end 3](#_Toc72354043)

[2.4 Technische realisatie 3](#_Toc72354044)

[2.5 Projectbacklog 3](#_Toc72354045)

[2.6 Planning 4](#_Toc72354046)

[3 Reflecties 5](#_Toc72354047)

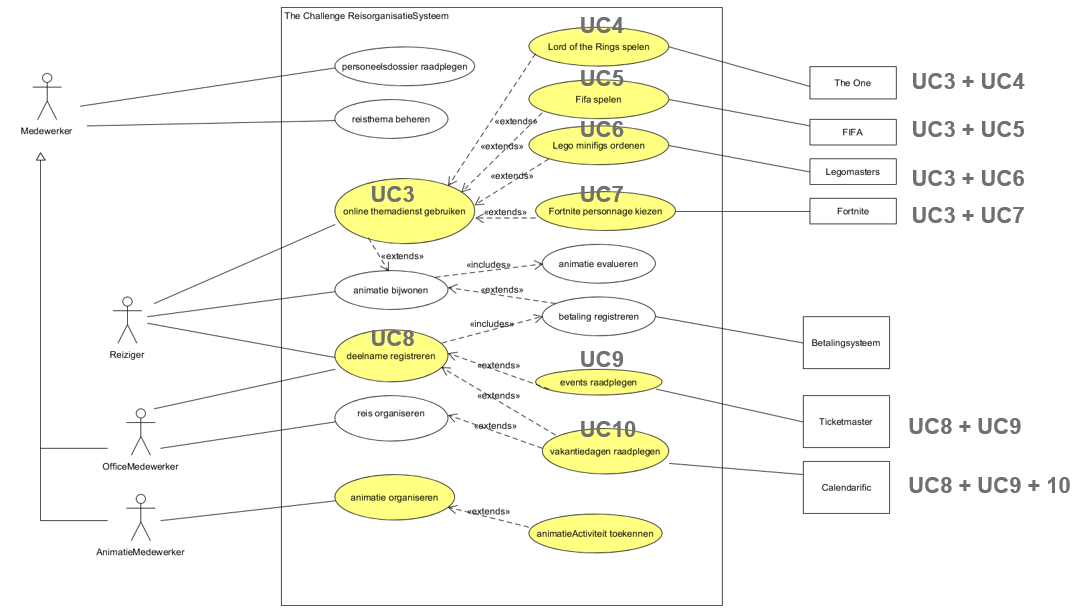
Einddocument IT Project

The challenge reisorganisatiesysteem v1.0

# Projectbeschrijving

## Projectinhoud

In het project is om een professionele webapplicatie te ontwikkelen. Twee delen, zogenoemde ‘Deelname Registreren’ (afkorting: UC 8) en ‘Events Raadplegen’ (afkorting: UC 9) van het project, worden door ons team gerealiseerd. Alle ‘Use Cases’ zijn voltooid.



‘Deelname Registreren’ (UC 8):

* Reiziger/officemedewerker geeft aan, aan een reis te willen deelnemen.
* Systeem vraagt de naam, geboortedatum, … van de reiziger.
* Systeem geeft lijst van de georganiseerde reizen.
* Reiziger/officemedewerker geeft aan, aan welke reis hij wil deelnemen.
* Systeem vraagt de deelname te bevestigen.
* Reiziger bevestigt deelname.
* Systeem registreert deelname.
* Einde UC.
* Reiziger/officemedewerker geeft aan events & attracties te willen opzoeken. Link naar UC 9 events raadplegen.

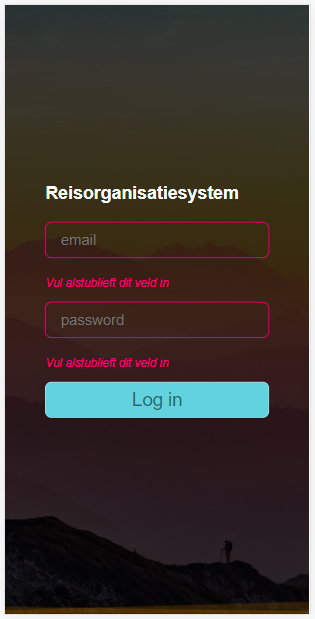
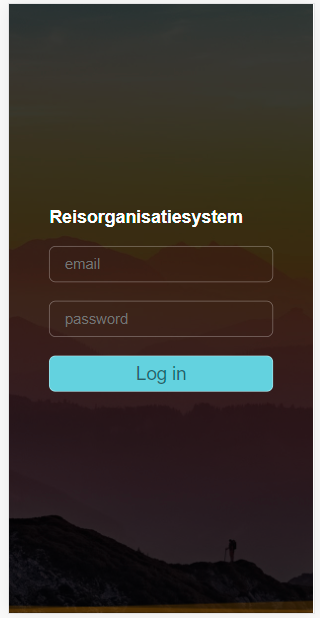
Events raadplegen (UC 9):

* Systeem vraagt de naam van een event of attractie.
* Reiziger/officemedewerker geeft de naam van een event of attractie op.
* Systeem vraagt de locatie (land of stad) van het event of attractie.
* Reiziger/officemedewerker geeft de locatie van het event of attractie op.
* Systeem geeft de lijst van de events & attracties.
* Reiziger/officemedewerker kiest een event of attractie.
* Systeem geeft prijzen en locatie van ticketverkoopagentschappen weer.
* Herhaal 1 t.e.m. 7 tot Reiziger/officemedewerker wenst te stoppen.
* Reiziger/officemedewerker kan het opzoeken van events en attracties annuleren.
* Geen events & attracties gevonden: Systeem geeft aan dat er geen attracties & events gevonden zijn

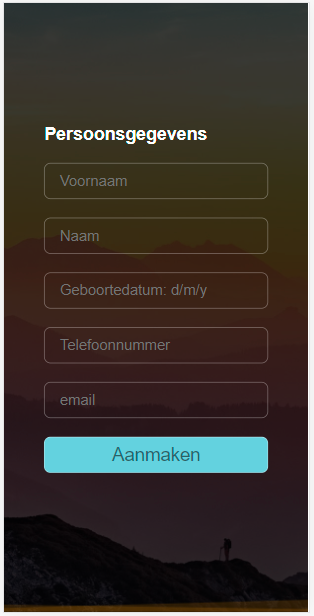
## Gebruikershandleiding

In UC 8 is het opstartscherm de aanmeldingspagina. Gebruiker kan zijn favoriete reis kiezen en bestellen.

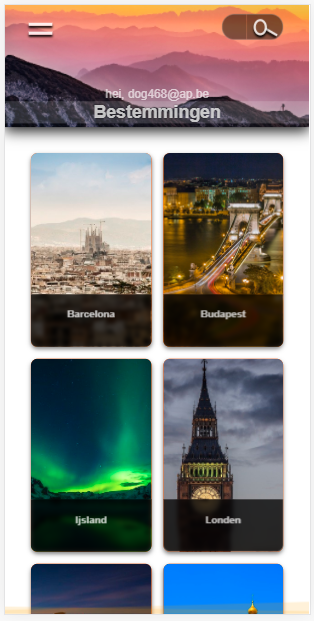
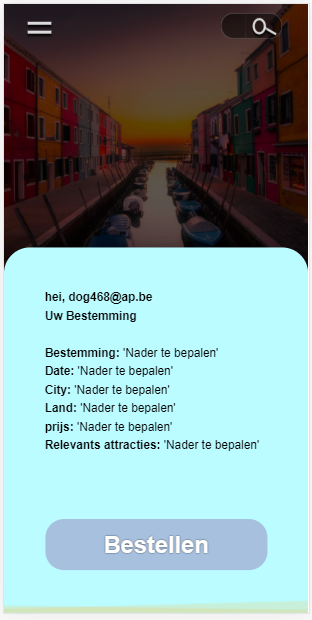
* De gebruiker kan zijn gebruikersnaam en wachtwoord gebruiken om toegang te krijgen tot de webapplicatie.

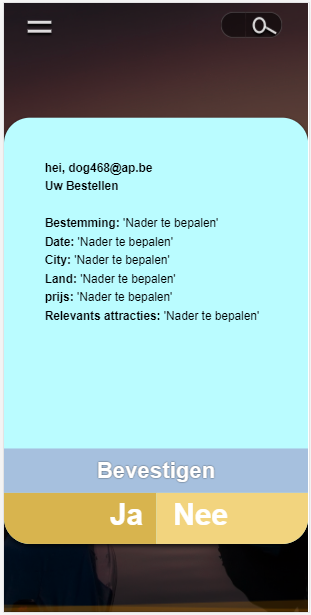
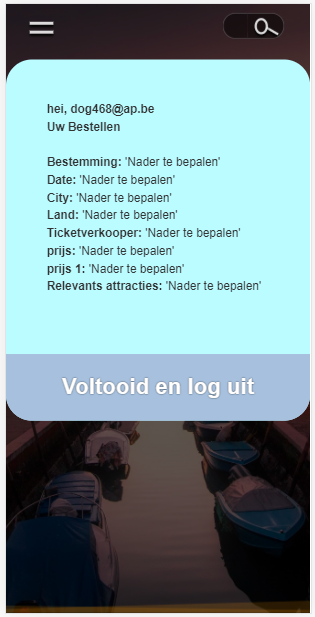


* Na het inloggen zullen de naam, geboortedatum, … van de reiziger worden geregistreerd. (De database is in de huidige fase nog niet klaar. De functionaliteit is niet beschikbaar.)

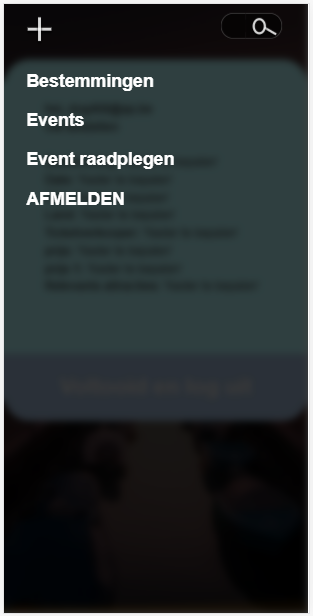


* Reiziger krijgt lijst van de georganiseerde reizen met foto's. De deelnemer kan zijn favoriete reis bestellen en ontvangt de bijbehorende gedetailleerde informatie en bevestiging.

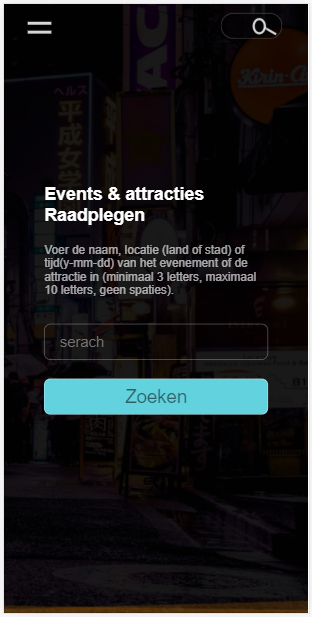
* Gebruiker kan blijven zoeken naar hun favoriete reis of het systeem veilig verlaten.



* Gebruiker kan naar UC 9 gaan om naar evenementen of attracties te zoeken.

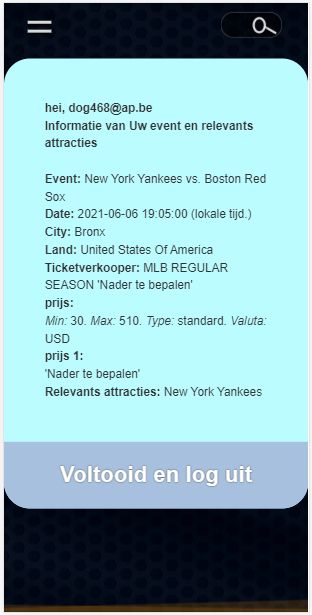
In UC 9 kunnen gebruikers kiezen of lijst van evenementen willen weergeven of zoeken naar specifieke evenementen en attracties.

* Zoeken naar specifieke evenementen en attracties. Hier gebruikt 'fuzzy search'. Gebruiker voert 3-10 letters en systeem geeft gevonden relevante evenementen of attracties weer. Als geen opgevraagde evenement of attractie gevonden is, toont systeem 'geen events & attracties gevonden'.



* Gebruiker klikt een event en systeem geeft relevante prijzen, locatie van ticketverkoopagentschappen en andere informatie weer.



* Gebruiker mag kiezen of verder zoeken of uitloggen.

## Technische documentatie

**Technische track:**

**Frontedn.**

HTML5: Alle pagina's voldoen aan HTML5-standaarden.

In ‘login’ pagina gebruikt ‘HTML5 Constraint Validation API’ om input waarden te valideren. (<https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Constraint_validation>)

CSS3: Grid lay-out + flex box lay-out + andere CSS-syntax worden gebruikt om responsieve pagina's realiseren.

Javascript: wordt gebruikt om ‘Hamburg menu’, ‘modal’, input validatie en ‘HTML5 Constraint Validation API’ te realiseren.

**Backend.**

Node.js + Express.js: lokale server aanmaken.

JSON: ‘Ticketmaster’ API.

Dependencies: body-parser, ejs, express-session en node-fetch.

Express is web framework for Node.js. ‘Express’ roep verschillende middleware op. De functie van de middleware: voer een willekeurige code uit. Wijzig de aanvraag- en antwoordobjecten. Beëindig de aanvraag-antwoordcyclus.

‘express-session’ wordt gebruikt om de aanmeldingsstatus van gebruiker te beheren. Als gebruiker login is, zal gebruikersinformatie in ‘session’ toevoegen. Toen gebruiker afmelden klikt, zal de ‘session’ vernietigen. Zonder legale ‘session’ is alleen ‘login’ route beschikbaar voor gebruiker.

Door ‘node-fetch' + ES6 destructuring + ES ‘optional chaining operation’ connecteert met ‘Ticketmaster API’ om gewenste data op te halen en filteren.

Door ‘body-parser' kan specifieke field van geposte data (HTML5 form) parseren.

‘EJS.js’ genereert HTML en met dynamische data.

**Dependencies & andere software:**

Node.js: <https://nodejs.dev/>

Express: <https://expressjs.com/>

Express-session: <https://www.npmjs.com/package/express-session>

Node-fatch: <https://www.npmjs.com/package/node-fetch>

Body-parse: <https://www.npmjs.com/package/body-parser>

# Projectrealisatie

## Linken en referenties

Vermeld hier:

* De url naar jouw gedeployde applicatie
* <http://nameless-stream-15098.herokuapp.com/login>
* De link naar de github repository met de definitieve versie
* <https://github.com/GuningDeng/Reisorganisatiesystem_Heroku.git>

## Front-end

**Hoe werd het front-end gerealiseerd?**

In de front-end fases wordt de basis van front-end (UC 8 en UC 9) met HTML5, CSS3 en Javascripp aangemaakt.

* Het prototype van UI/UX is door **Adobe XD** aangemaakt.
* Alle pagina’s zijn volgens gebruik scenario aangemaakt.
* Pagina bevat tekst, formulier (HTML5 form) of foto’s. Alle pagina’s zijn responsief en beschikbaar voor mobiele telefoons, tablets en PC’s.
* Met behulp van "**Grid**" en "**Flex box**" kan elke pagina responsief zijn.
* CSS3 syntax voor foto adaptief is: **img: {object-fit: }.**
* ‘**js/ham.js**’ is gebruikt om ‘Hamburg menu’ te realiseren. **‘classList.toggle (‘classname’)’** voegt een element nieuwe ‘class parameter’ toe of verwijdert. Op een element klikt en wijzigt de positie van een groep elementen. Op deze manier wordt het menu weergegeven of verborgen.
* Alle input waarden worden gevalideerd.
* De validatie van ‘search input’ (pagina: events, event en raadplegen).

Functie ‘**preventDefault()**’: Voorkom dat formulieren worden ingediend als input leeg of input waarde illegale is.

Functie ‘**match()**’: Valideer of input waarde legaal is.

* ‘**HTML5 Constraint Validation API**’ om input waarden te valideren (inlog pagina). (<https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Constraint_validation>)

**In welke fases is deze verlopen?**

Van sprint 1 tot sprint 4 is deze verlopen.

**Fase 1**: Analyseren van relevante document. (Sprint 1)

**Fase 2**: Analyseren gebruiker scenario's. (Sprint 1)

**Fase 3**: Bevestigen technische track. (Sprint 1)

**Fase 4**: Maak een prototype van UI/UX. En nuttige documenten afronden. (Sprint 2)

**Fase 5**: Taakverdeling. (Sprint 2)

**Fase 6**: Development en reviews. (Sprint 3-4)

**Be taakverdeling:**

Ieder teamlid doet zijn best.

**Aerts Yarno**: Pagina index.html, login.html, raadplegen.html, bevestig.html

**Guning Deng**: Pagina profiel.html, bestemmingen.html, bestemming.html, bestellen.html, events.html, event.html, ‘ham.js’, ‘inputvalidate.js’.

## Back-end

**Hoe werd de back-end gerealiseerd?**

In de fase wordt UC 8 & UC 9 van de webapplicatie volledig herbouwd.

De functionaliteiten van ‘Read’ wordt gerealiseerd: login & afmelden, Dynamische weergave-informatie en events raadplegen.

* In de fase is een webserver aangemaakt. De applicatie is op basis van de webserver gebouwd.
* Alle functionaliteiten van de webapplicatie zijn afhankelijk van de server.
* Volgens gebruik scenario’s zijn deze routes (in **index.js**) gecreëerd.
* Route (‘/’) (method: get) overeenkomstige pagina: index.
* Route (‘/login’) (method: get) overeenkomstige pagina: login.
* Route (‘/doLogin’) (method: post) dient ‘login request’ in.
* Route (‘/profile’) (method: get) overeenkomstige pagina: profile.
* Route (‘/bestemmingen’) (method: get) overeenkomstige pagina: bestemmingen.
* Route (‘/bestemming’) (method: get) overeenkomstige pagina: bestemming
* Route (‘/bestellen’) (method: get) overeenkomstige pagina: bestellen.
* Route (‘/bevestig’) (method: get) overeenkomstige pagina: bevestig.
* Route (‘/events’) (method: get) overeenkomstige pagina: events.
* Route (‘/event’) (method: get) overeenkomstige pagina: event.
* Route (‘/raadplegen’) (method: get) overeenkomstige pagina: raadplegen.
* Route (‘/dosearch’) (method: post) om ‘search’ functie te realiseren.
* Nuttige gegevens zijn via ‘Ticketmaster’ API dynamisch opgehaald.
* In aparte module (**model/database.js**) wordt data via ‘Ticketmaster’ API opgehaald. Na het filteren en verwerken van de data zal bruikbaar data ‘export’ worden.
* Door '**node-fetch**' connecteert met 'Ticketmaster' API.
* Door '**ES6 Destructuring**' om bepaalde field op te halen.
* ES6 '**map()**' wordt gebruikt om gegevens te filteren en verwerken.
* '**module.exports**' biedt een interface waarmee andere modules API-gegevens kunnen lezen.
* Alle pagina’s van front-end worden met EJS herbouwd om dynamische webpagina’s weer te gegeven.
* Als er mogelijk is, gebruiken alle pagina’s modules te vormen. Er zijn drie gemeenschappelijke module: **head.ejs**, **header.ejs** en **footer.ejs**.
* Autorisatie. (In huidige delen zijn login en afmelden).
* In route(‘/doLogin’) gebruikt functie ‘**find()**’ om gebruiker te valideren. Als de gebruiker toegang heeft, gaat dan naar de route ('/profile') en voeg de gebruikersinformatie toe in de 'session'. Anders gaat naar route(‘/login’).
* In route(‘/logout’) vernietigt ‘session’ en bezoeker kan niet naar vorige pagina terugkeren.
* Alle input waarden zullen worden gevalideerd.
* De authenticatie van input waarden is door functie ‘**match()**’ gerialiseerd.
* Events raadplegen op trefwoord.

Door trefwoord kan relevante events zoeken. Als event(s) is(zijn) gevonden, geeft(geven) de gevonden events weer. Anders heeft melding ‘Geen zoek events gevonden!’ weer.

* Functie ‘**selectMatchItem**’: ‘Fuzzy search’.
* Functie ‘**getAEvent**’: om een specifiek event te vinden.
* Ontdubbelen: Door ‘**reduce()**’ verwijdert de dubbele events.

**In welke fases is dit verlopen?**

In sprint 5 is deze verlopen.

**Fase 1**: Webserver wordt aangemaakt. (Sprint 5)

**Fase 2**: Volledige en nieuwe project is aangemaakt. Routes wordt aangemaakt. (Sprint 5)

**Fase 3**: Connectie met ‘Ticketmaster’ API is gerealiseerd en module ‘database.js’ is afgerond. (Sprint 5)

**Fase 4**: Alle front-end pagina’s zijn herbouwd met EJS. (Sprint 5)

**Fase 5**: Alle volledige routes en functionaliteiten zijn voltooid en review is uitgevoerd. (Sprint 5)

**Wie deed wat / hoe is de taakverdeling gebeurd onder de teamleden?**

Alle delen zijn door Guning Deng gerealiseerd. (Andere leden afwezig)

**Welke library’s werden gebruikt? Waarom? Waren er alternatieven die je hebt overwogen?**

* Express.js: webapplicatie framework. Alternatief: Koa 2.
* Node.js body parsing middleware. Parseer inkomende 'request bodies' in een middleware voor uw handlers. Alternatief: geen betere middleware is gevonden in het scenario.
* express-session: session middleware om gebruikersinformatie op te slaan. Alternatief: nee.
* node-fetch: Een lichtgewicht module die window.fetch naar Node.js brengt. Alternatief: geen betere middleware is gevonden in het scenario.
* EJS: een eenvoudige 'template language' waarmee u 'HTML markup' kunt genereren met gewoon JavaScript. Alternatief: art-template.

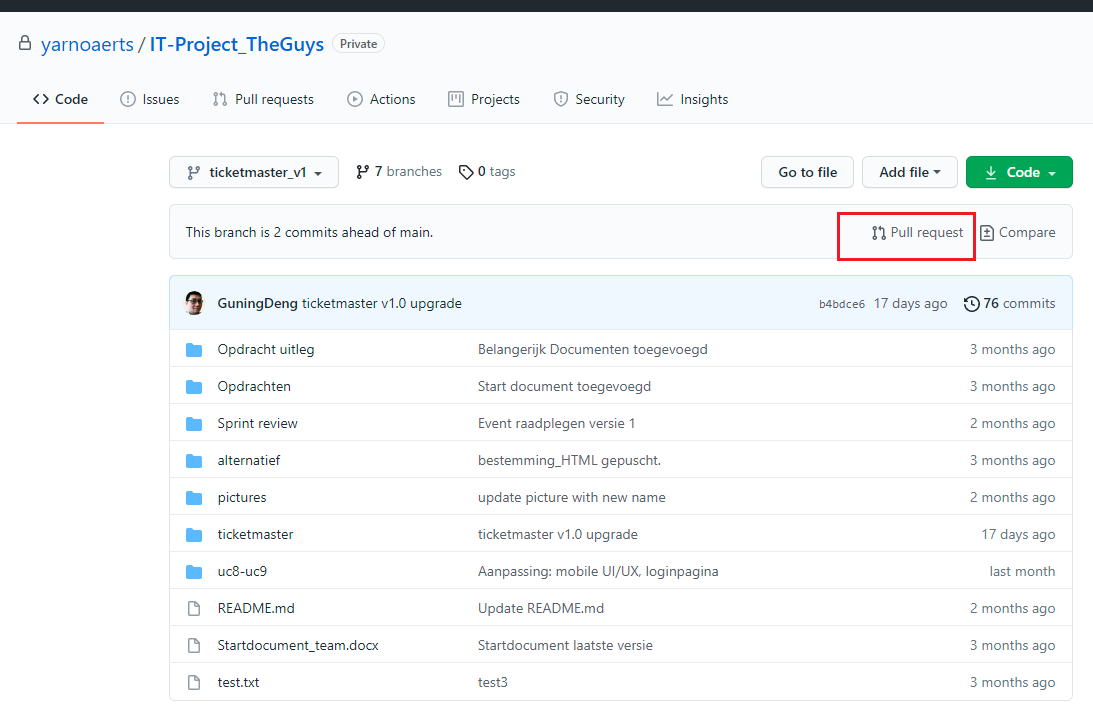
## Technische realisatie

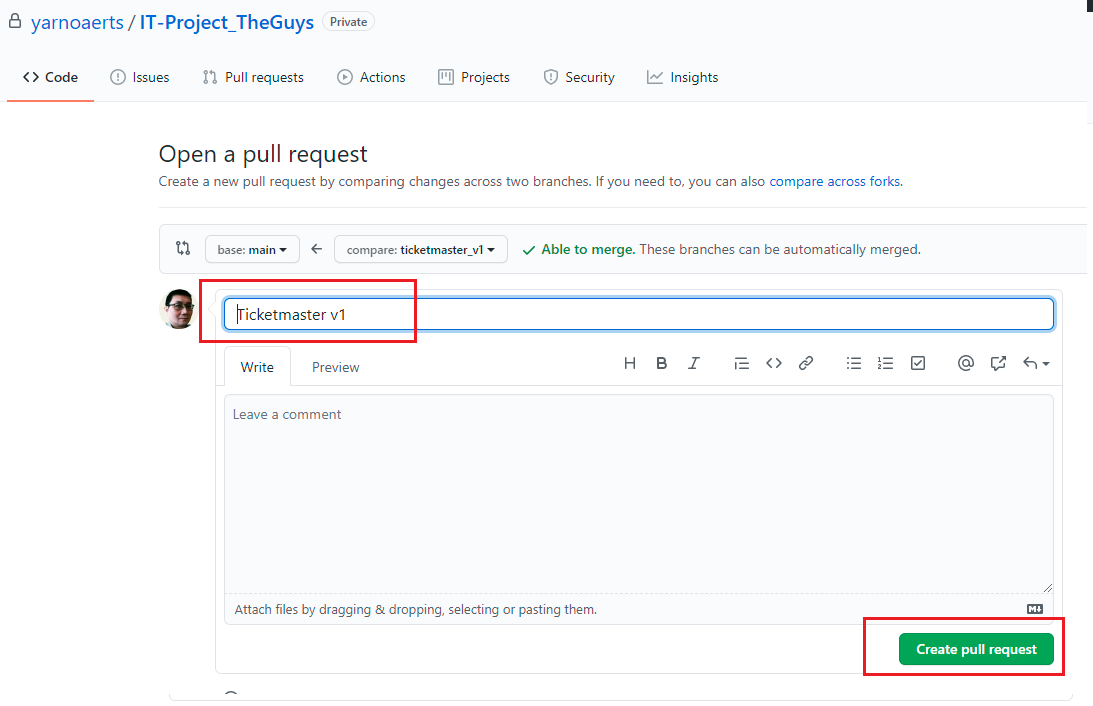
Elke lid kan binnen git-repository een nieuwe branch aanmaken om zijn versie te bewerken: **git branch [branchname]**

Elke lid mag nuttige branch pullen naar locatie en verder werken: **git pull** (vernieuw local branch) of **git fetch** (verkijg remote repository).

Op een gegeven moment willen we de wijzigingen die we hebben aangebracht integreren in een andere branch, typisch de master branch. We doen dit door gebruik te maken van een **pull-request**. Met behulp van een pull-request brengen we anderen op de hoogte dat we wijzigingen hebben aangebracht in een branch. Nadat de wijzigingen in de branch zijn goedgekeurd kunnen ze worden samengevoegd met de master branch.

**GitHub-flow** en **pull-request**.





Aan het eind van deployment worden nuttige branches samengevoegd naar master branch: **git merg**.

(<https://git-scm.com/docs)>

## Projectbacklog

**Is de volledige projectbacklog die initieel was opgesteld volledig afgewerkt?**

De volledige projectbacklog is volledig afgewerkt.

**Zijn jullie afgeweken van de initiële backlog?**

Nee.

## Planning

**Welk was voor jullie de belangrijkste sprint? Wat hebben jullie daarin gerealiseerd?**

Sprint 1 is de belangrijkste sprint en die is gerealiseerd.

**Zouden jullie – achteraf bekeken – de sprints anders hebben ingepland? Hoe dan? Waarom?**

Nee.

# Reflecties

**Welk was voor jou individueel de grootste challenge? Hoe ben je daarmee omgegaan?**

De grootste challenge voor mij is de afwezigheid van teamleden.

Ik heb meer backlog overgenomen.

**Noem 2 leerpunten die je hebt bijgeleerd.**

* Eerst is een team met goede motivatie nuttig.
* Tweede is goede noodplan een must.

**Noem 2 werkpunten die je voor jezelf hebt gedefinieerd op basis van het verloop van het project, waar je in de toekomst belang aan hecht om zeker nog bij te leren.**

* Volgens scenario's zal het project volledig herbouwen. 'control' module zal worden toegevoegd. Meer gebalanceerde tussen verschillende modules. In het project is te veel 'business logica' in routes. Hier zal andere 'route stijl' gebruiken en 'business logica' kan naar controlemodule worden verplaats.
* Security en ‘Multi-user concurrency’.

**Hoe ervaarde je het werken in teamverband aan dit project? Zou je voorkeur ernaar uitgaan om het project alleen te realiseren, mocht je dit traject kunnen overdoen? Waarom wel / niet?**

**Hoe ervaarde je het werken in teamverband aan dit project?**

* Ik heb veel positieve punten ervaren.

**Zou je voorkeur ernaar uitgaan om het project alleen te realiseren, mocht je dit traject kunnen overdoen? Waarom wel / niet?**

* Ja. In back-end fase heb ik alleen al het project herbouwd.
* Het project is in huidige fase alle eenvoudige functionaliteiten gerealiseerd:
* Alleen ‘Read’ van ‘CURD’ is gerealiseerd.
* Gebruikersrol is alleen bezoeker met de functionaliteiten van ‘Read’.
* Geen database verbindt.
* Er zijn heel weinige data om te bewerken.