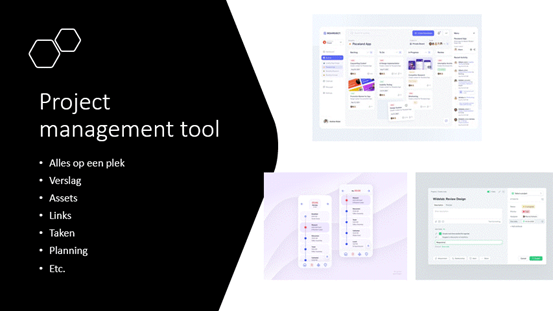
Project Management Software

Fontys S02 individueel project

Thijmen Brand | DB05-S02 | 11-06-2022

Een succesvol project begint altijd met een idee. Deze moet uiteindelijk duidelijk en helder zijn. Maar om te kunnen beginnen is het belangrijk om een globaal plan te schetsen. Dit niet een te veel detail of woorden. Hier komt een one-slide pitch goed van pas. Bij deze aanpak wordt je gedwongen in een korte vorm te formuleren wat je wilt gaan doen.

Met dit idee wil ik het process van het starten en onderhouden van een project gemakkelijker maken. Ik heb gemerkt bij mijn eigen projecten dat ik op veel verschillende plekker resources heb staan. Zo heb ik een word document voor mijn verslag, een ander document voor mijn documentatie. Dan sla ik in mijn brouwser of youtube handige links en video’s en mijn taken staan in trello. Dit zijn een heleboel verschillende plekken waardoor ik het af en toe moeilijk vind het overzicht te houden over wat waar staat. En dit probleem wil ik oplossen door het hele project process op een plek te combineren. Hier kan ik dus een verslag maken, Maar ook kan ik er andere bestanden zoals fotos, code of documentatie opslaan en bewerken. Daarnaast kan ik handige links of assets opslaan en makkelijk terug vinden, ik kan mijn taken handig op een rijtje zetten onder to-do, doing of done. En ik kan een dag/ week/ sprint planning maken.

Inhoud

[Product documentatie 2](#_Toc105844887)

[Code base 2](#_Toc105844888)

[Software requirements 2](#_Toc105844889)

[High level requirements 2](#_Toc105844890)

[MoSCoW requirements 4](#_Toc105844891)

[Minimal Viable Product (MVP) 7](#_Toc105844892)

[Non-functionele requirements 7](#_Toc105844893)

[Technische documentatie 7](#_Toc105844894)

[Begrippen 7](#_Toc105844895)

[Techniek onderbouwing 8](#_Toc105844896)

[Backend api 8](#_Toc105844897)

[Frontend 10](#_Toc105844898)

[Testen 10](#_Toc105844899)

[Unit tests 10](#_Toc105844900)

[End to End tests (E2E) 12](#_Toc105844901)

[Code quality 15](#_Toc105844902)

[Process documentatie 15](#_Toc105844903)

[Sprint planning 15](#_Toc105844904)

[V1 15](#_Toc105844905)

[V2 17](#_Toc105844906)

[Algemene documentatie 18](#_Toc105844907)

[Terug blik op sprint twee 18](#_Toc105844908)

[Sprint 3 19](#_Toc105844909)

[User authorization 19](#_Toc105844910)

[Terugblik op sprint 3 19](#_Toc105844911)

[Terugblik op sprint 4 21](#_Toc105844912)

[Terugblik op sprint 5 22](#_Toc105844913)

# Product documentatie

## Code base

**Vuejs**

Vuejs is een front-end javascript library. het focust zich vooral op simpliciteit en snelheid. Dit komt tot uiting in het starten van een nieuw project. bijna alles wordt voor je geregeld en verschillende opties kan je tijdens het initialiseren van het project al kiezen welke de vue CLI dan voor je configureert. Daarnaast in het gebaseerd op html templates waardoor het erg lijkt op gewoon html en het dus makkelijk op te pakken is.

**Typescript**

Om nog nettere en onderhoudbaardere code te schrijven maak ik bovenop vue nog gebruik van typescript. Dit is een javascript superset die er voor zorgt dat javascript strongly typed wordt. dit houdt in dat je types moet defineren.

**ASP.net web api 6**

Als backend maak ik gebruik van [asp.net](http://asp.net/) web api.

**SQL server**

Als database gebruik ik microsoft sql server. Dit omdat het goed samenwerkt met de [asp.net](http://asp.net/)

**E2E tests**

Om mijn applicatie flow en end to end tests uit te voeren gebruik ik de library playwright. Deze kan automatisch de frontend gebruiken op basis van scripts en dan door middel van assertions kan het voor verschillende dingen checken

## Software requirements

### High level requirements

* **Project verslag:** De gebruiken kan een verslag opstellen van het project. Hierin kan onderzoek in worden gedocumenteerd of iets anders. Dit is een plain tekst document waar verschillende tekst stijlen kunnen voorkomen. Alles kan hierin getypt worden zonder dat het door de server wordt uitgevoerd. Dit elimineert de optie om foutieve code in de website applicatie te injecteren. Ook bevat het de functie om automatisch op te slaan wanneer er een bewerking plaatsvind. Wanneer het opslaan of het uitlezen van het document mislukt zal er een duidelijke rode foutbalk verschijnen boven in het scherm met een unieke foutcode en de tekst die gekoppeld is aan deze unieke foutcode.
* **Taken:** De gebruiker moet de mogelijkheid hebben om verschillende taken aan te maken. Naast taken moet hij ook statussen voor deze taken aan kunnen maken, bijvoorbeeld to-do, doing en done. Deze taken kunnen een deadline hebben en deze deadlines kunnen aan een agenda worden bekeken. Ook kan een gebruiker een beschrijving bij een taak zetten.

De taak namen mogen geen speciale tekens bevatten bahalve vraagtekens en uitroeptekens. Verder mogen de titels een beperkte lengte van 50 tekens bevatten. Wanneer de gebruiker meer informatie bij een taak wil opslaan zal dit in de beschrijving gezet moeten worden. De beschrijving mag ook geen special tekens bevatten behalve een uitroepteken of een vraagteken.

Wanneer een taak niet aangemaakt kan worden door bekende redenen, bijvoorbeeld een verkeerde datum (die al is verlopen) of een verkeerde titel of beschrijving zal dit direc tijdens de invoer duidelijk worden gemaakt door middel van rode letters onder de invoer. Wanneer het niet aangemaakt of bewerkt kan worden door onbekende redenen zal er een rode balk verschijnen binnen de modal die wederom een error code geeft met de corosponderende tekst.

* **Files:** Een gebruiker moet verschillende bestanden kunnen uploaden. Deze bestanden moeten daarnaast te in mappen te plaatsen zijn om ze te ordenen. Ook moet de gebruiker zelf tekst bestanden kunnen aanmaken en deze moeten net als het project verslag voorzien zijn van verschillende tekst stijlen. En ten derde moet een gebruiker bestanden kunnen verwijderen. Daarnaast moet de naam van een bestand aangepast kunnen worden en bestaande bestanden moeten bewerkt kunnen worden. Aan de hand van het mime type zal er een icon worden gekozen voor het bestand.

De namen van de bestanden en mappen mogen geen speciale tekens bevatten. Behalve een punt voor het Mime type. Binnen de file mag alles geplaats worden. Dit zal door de server niet geïnterpreteerd worden als code en daardoor ook niet uitgevoerd. Bestanden mogen enkel een beperkte grootte hebben, op deze manier zal de opslag niet te snel vol staan en zal het uploaden ook sneller gaan.

Wanneer er een foutieve titel aan een map of bestand word gegeven zal dit bij de invoer al duidelijk gemaakt worden doormiddel van een rode tekst onder de invoer balk. Wanneer een bestand te groot is zal er bij het bestand upload scherm een rode tekst verschijnen met de desbetreffende error. Wanneer er een onbekende fout is bij het uploaden zal dit door middel van een rode balk bekend worden gemaakt met een error code en een korte beschrijving van de code.

* **Beschrijving:** Een project kan een beschrijving hebben. Deze word enkel weergegeven bij het project overzicht.

Een gebruiker kan hier geen tekens invoeren die kunnen leiden tot foutieve code. Wanneer dit wel het geval is zal er geen error komen. Maar zal de tekst zonder speciale tekens worden opgeslagen.

Wanneer er een onbekende fout optreed bij het opslaan zal dit door middel van een rode balk boven aan het scherm bekend worden gemaakt Wederom met een error code en een korte beschrijving hierbij.

* **Links:** Een gebruiker moet de mogelijkheid hebben om links van webpagina’s op te slaan. Wanneer er op een van deze links geklikt wordt zal hij doorverwezen worden in een nieuw tabblad naar de desbetreffende internet pagina. Deze links moeten ook weer verwijderd of aangepast moeten kunnen worden.

Hier mogen enkel teksten worden ingevoerd die een valide url zijn. Wanneer dit niet het geval is kan de link niet worden opgeslagen en zal er onder de link invoer een rode tekst verschijnen met de vraag een valide url in te vullen. Ook de naam van het item mag geen speciale tekens bevatten en dit word direct bij invoer gecheckt. Wanneer er een foutieve invoer is zal dit direct onder de invoer bekend worden gemaakt door middel van een rode tekst.

* **Herinneringen:** Een gebruiker moet herinneringen aan kunnen maken. Deze kan dan op interval en tijd worden ingesteld om aan het project te werken, of om voor een bepaalde tijd een taak af te hebben. Daarnaast moet er een deadline herinnering ingesteld kunnen worden.

De data die bij deze herinneringen word ingevuld moeten in de toekomst liggen. Wanneer dit niet het geval is zal er direct een foutmelding onder de invoeg gegeven worden in rode tekst. Wanneer er door een onbekende fout geen herinnering kan worden toegevoegd zal er boven aan de pagina een rode foutbalk worden weer gegeven met een foutcode en een tekst.

* **Sprints:** Een gebruiker moet sprints kunnen aanmaken. Deze sprint kan dan een duur hebben waarna de volgende sprint start tot dat het project ten einde loopt. Aan deze sprint kunnen van taken worden toebedeeld die aan het einde van de sprint af moeten zijn.

Een sprint duur mag geen negatieve waarde hebben. Ook mag het niet boven een bepaalde waarden liggen. Wanneer dit wel het geval is krijg een direct een rode foutmelding.

* **Agenda:** Een gebruiker moet de mogelijkheid hebben om alle data die hij op slaat bij bijvoorbeeld taken, herinneringen en sprints in een agenda te zien. Ook kan de gebruiker van zelf agenda items aanmaken met weer individuele herinneringen.

### MoSCoW requirements

* **Must have**
  + 1.1.1: Als project manager wil ik in kunnen loggen om mijn projecten te beschermen tegen de buitenwereld.
  + 1.2.1: Als project manager wil ik een nieuw project aan kunnen maken om project details in op te slaan
  + 1.2.2: Als project manager wil ik een project die niet meer gebruikt wordt of waar de development mee gestopt is kunnen verwijderen om mijn account schoon en opgeruimd te houden.
  + 1.2.3: Als project manager wil ik een project een naam geven om te weten wat het project is en om het te onderscheiden van andere projecten
  + 1.3.1: Als project manager wil ik de mogelijkheid hebben om sprints een bepaalde duur te geven om zo mijn taken goed geordend in het tijdsbestek wat ik wil te kunnen doen
  + 1.4.1: Als project manager wil ik taken aan kunnen maken om op een rij te zetten wat er allemaal moet gebeuren
  + 1.4.2: Als project manager of developer wil ik de naam van een taak kunnen bewerken om eventuele fouten op te lossen
  + 1.4.3: Als project manager wil ik onnodige taken of outdated taken kunnen verwijderen om de werkomgeving schoon te houden.
  + 1.4.4: Als project manager wil ik taken in kunnen plannen in sprint planningen om te weten wanneer de taak aan de beurt komt.
  + 1.4.5: Als project manager wil ik taken uit te planning kunnen verwijderen om eventuele fouten op te lossen of om oude taken weg te halen.
  + 1.5.1: Een gebruiker kan van sprint weergave wijzigen, uur planning / dag planning / week planning / sprint duur planning 1.5.1
* Should have
  + 2.1.1: Als project manager wil ik een project een beschrijving kunnen geven om een beter beeld te krijgen en te geven aan developers over wat het project precies inhoudt
  + 2.1.2: Als project manager wil ik de naam van mijn project kunnen veranderen om eventuele fouten op te lossen, of veranderingen door te voeren
  + 2.1.3: Als project manager wil ik de beschrijving van een project veranderen om eventuele fouten op te lossen of om een aanpassing te maken wanneer de project details veranderen
  + 2.2.1: Als project manager wil ik een bestand verwijderen om verwarring met onnodige documenten te voorkomen
  + 2.2.2: Als project manager wil ik bestanden kunnen aanmaken om informatie globaal bereikbaar te hebben.
  + 2.3.1: Als developer of project manager wil ik een url kunnen opslaan om later te kunnen gebruiker of om een handige pagina op te slaan voor later
  + 2.3.2: Als developer of project manager wil ik de titel van een link kunnen aanpassen om eventuele fouten op te lossen
  + 2.3.3: Als developer of project manager wil ik een link kunnen verwijderen om al gebruikte of niet meer relevante informatie op te schonen
  + 2.4.1: Als project manager of developer wil ik een taak een beschrijving kunnen geven om een duidelijker beeld te krijgen wat een specifieke taak inhoudt
  + 2.4.2: Als project manager wil ik een taak een deadline geven om te weten wanneer iets af moet zien.
  + 2.4.3: Als project manager of developer wil ik taken kunnen schuiven tussen containers wanner de status is veranderd
  + 2.5.1: Als project manager wil ik verschillende taak container aanmaken om taken te ordenen in status
  + 2.4.4: Als project manager of developer wil ik de weergaven van mijn taken kunnen wisselen om een overzichtelijk beeld te krijgen van de taken.
  + 2.4.5: Als project manager of developer wil ik bij een taak kunnen zien wanneer en of deze staat ingepland om te weten of het nog ingepland moet worden en wanneer deze aan de beurt komt.
* Could have
  + 3.1.1: Als gebruiker wil ik handige bestanden kunnen opslaan om zo documentatie te schrijven of belangrijke bestanden globaal bereikbaar te hebben.
  + 3.1.2: Als project manager of developer wil een opgeslagen bestanden bekijken om informatie over het project of andere project gerelateerde documenten te lezen
  + 3.1.3: Als project manager wil ik een bestand kunnen bewerken om mijn developers de meest actuele informatie te geven
  + 3.1.4: Als project manager wil ik mappen aan kunnen maken om een overzichtelijke en schonen werk omgeving te creëren
  + 3.1.5: Als project manager wil ik bestanden in mappen kunnen plaatsen om een overzichtelijke omgeving te creëren
  + 3.1.6: Als project manager wil ik een onnodige map kunnen verwijderen om verwarring met informatie te voorkomen
  + 3.2.1: Als project manager wil ik requirements en user stories kunnen exporteren naar taken om op deze manier makkelijk en snel een planning te kunnen maken
  + 3.3.1: Als project manager of developer wil een taak een tag kunnen geven om te weten in welke categorie deze valt
  + 3.3.2: Als project manager wil ik tags aan kunnen maken om categoriën bij te houden
  + 3.4.1: Als developer wil ik kunnen bijhouden hoelang iets duurt om in de toekomst een beter beeld te krijgen over te tijd wat dingen innemen
  + 3.4.2: Als developer wil ik een naam aan deze meting geven om duidelijk te weten waar de meting voor was
  + 3.4.3: Als developer wil ik een beschrijving aan de meting geven om een gedetailleerde beschrijving te krijgen van waar de meting voor was en of er enige bijzonderheden waren.
  + 3.4.4: Als developer wil ik een bepaalde taak aan de meting kunnen handen om inzicht te krijgen in hoelang een bepaalde taak duurt.
  + 3.5.1: Als project manager wil ik een project Wiki kunnen maken om te documenteren of een gedetailleerde beschrijving te geven van het project
  + 3.5.2: Als project manager wil ik verschillende pagina’s aan de wiki hangen om verschillende onderwerpen toe te lichten
  + 3.5.3: Als project manager wil ik verschillende elementen aan de wiki toevoegen om een duidelijk beeld te geven

### Minimal Viable Product (MVP)

* Een gebruiker kan een project aanmaken
* Een gebruiker kan een sprints planning zien
* Een gebruiker kan een taak maken
* Een gebruiker kan een taak aan een sprint toevoegen

### Non-functionele requirements

* De applicatie staat 3 inlog pogingen toe
* Wanneer de inlog pogingen op zijn wordt het account geblokkeerd
* Wanneer het account geblokkeerd is zal er een reset mail naar het geregistreerde mail adres gestuurd worden
* De applicatie hoeft enkel op desktop te functioneren
* Wanneer er een bekende fout op treed zal deze duidelijk getoond worden met beschrijving wat er fout ging.
* Wanneer er een onbekende fout op treed zal de foutcode getoond worden waarmee contact gelegd kan worden met de admin
* Een gebruiker ken .txt bestanden openen

## Technische documentatie

### Begrippen

PMS: Project management systeem. Je project waar ik momenteel aan werk en waar je verschillende projecten kan beheren.

sut: System under test.

### Techniek onderbouwing

De applicatie zal gaan bestaan uit een front- en een backend. Deze twee componenten zullen strikt gescheiden blijven. Dit is voordelig omdat op deze manier de frontend makkelijk aan te passen of te updaten is zonder dat dit complicaties met zich mee brengt. De enige manier waarop deze twee lagen namelijk afhankelijk van elkaar zijn is een http endpoint die makkelijk te veranderen zou zijn. Wanneer deze manier niet toegepast zou worden en de front- en backend grotendeels met elkaar verweven is, word het heel lastig om iets makkelijk aan te passen aan een van beide delen.

Als frontend techniek werk ik met Vue.js. Dit is een javascript library welke zich focust op de het zo simpel mogelijk maken maar toch vele features bieden. Zo zijn er duizenden packages voor te vinden en is het een no-config of low-config framework. Dit houdt in dat alles voor je wordt geregeld voor de vue CLI tool. Deze installeert de juiste packages en configureert ze naar jouw behoefte. En wanneer je toch iets meer wil aanpassen kan dit in de Vue.Config file. Daarnaast maakt hebt gebruik van html templates in combinatie met JSX. Dit is een voordeel omdat het duidelijker is om te lezen maar toch krachtig en snel is. Daarnaast is de documentatie erg uitgebreid wat het makkelijk maakt om met vue te beginnen.

### Backend api

De backend van PMS bestaat is gebouwd als een restful web-api met verschillende lagen om op deze manier de kwaliteit en onderhoudtbaarheid van de applicatie te waarborgen. De eerste laag is de Controller laag, deze ontvangt ieder http request met de bijbehorende data, hier wordt de data gefilterd en gecontroleerd. Vervolgens gaat de data naar de Business laag waar de data verder gefilterd en gevormd word naar de vorm waarop het bruikbaar is. Wanneer dit is gebeurt word de data wederom door gestuurd naar de laatste laag en dit is de data laag. Deze voegt met wat laatste checks de data toe aan de database. Dit dezelfde proces werkt ook omgekeerd met wanneer er data wordt opgevraagd.

De Controller laag: Deze laag heeft een aantal verschillende controllers. Voor iedere hoofd functionaliteit een. Zo is er bijvoorbeeld een project controller, een taak controller en een sprint controller welke allemaal zorgen voor zijn eigen taken.

De “SprintsController.cs” heeft twee functies welke ook endpoints zijn. Dit zijn “AddSprint” en “GetSprintsByProject” De namelijk speken voor zich.

De “TasksController.cs” heeft vier functies welke wederom ook endpoints zijn. Deze zijn “AddTask”

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijvingOm de samenhang tussen front- en back-end te visualiseren heb ik een high level data flow model gemaakt. Hier gaat de data van de DOM (gebruikers presentatie en interactie) naar de store waar de data klaargemaakt wordt om naar de backend te sturen. Vervolgens wordt het via de API middleware die zorgt dat de juiste endpoint en http methode gebruikt word naar de backend gestuurd. Daar wordt het ontvangen door de API laag die checkt of de gebruiker is geautoriseerd waarna de data door gestuurd word naar de validatie laag. Hier wordt de data gecontroleerd op juistheid. Daarna wordt het doorgestuurd naar de executie laag waar de benodigde business logica wordt toegepast. En ten slotte wanneer alles goed is gegaan wordt het naar de data laag gestuurd welke de database operaties uitvoert.

### Frontend

De frontend bestaat uit een aantal belangrijke componenten die samenwerken om de data weer te geven in mooie visuele pagina’s. Het aller belangrijkste zijn de templates. Dit defineerd hoe een pagina er uit ziet en wat er allemaal op staat. In diezelfde file als de template staat ook het script. Dit is gekoppeld aan die file en dat gaat over de logica op die pagina; Bijvoorbeeld het ophalen van data, kleine berkeningen doen etc. Onder deze weergave laag zit de store. Dit is een centrale opslagruimte voor alle data en complexe(re) logica. en als laatste hebben we de API laag. Deze maakt de daadwerkelijke api requests naar de backend. En al deze onderdelen plus nog modellen, helper functie en andere onderdelen maken de frontend.

## Testen

### Unit tests

Om de kwaliteit van mijn software te garanderen schrijf ik voor iedere backend functie een unit test. Dit doe ik echter alleen voor mijn business layer waar ik de logica af handel. De andere lagen maken namelijk gebruik van standaard functies en geven alleen maar data door waardoor er niks fout kan gaan.

Ik gebruik voor mijn unit tests twee verschillende libraries. Dit zijn Moq en NUnit. De Moq library zorgt ervoor dat ik gemakkelijk een depandancy kan isoleren. Dit is nodig zodat mijn functie die een test ondergaat geen andere dependancies in de applicatie gebruikt (24-03-2009, Van Fosson, T.).

In mijn unit tests houdt ik en bepaalde structuur aan om de overzichtelijkheid te bewaren. Eerst maak ik namelijk mijn test variabelen en mijn verwachter uitkomst. In dit geval test ik op een exception bij null. dus mijn test variabelen is null. De functie die ik test geeft een boolean terug en ik verwacht een false terug te krijgen wanneer het project niet is aangemaakt, dus is de expected output false.

//Arrange

Project project = null;

bool expected = false;

Daarna ga ik mijn dependancy klaar maken. Omdat ik dependancy injection gebruik om een meerlaagse structuur te creeëren moet ik de benodigde depandancys mocken. Dit zorgt maakt een soort van neppe kopie van de benodigde depancancy. Daarna ga ik de mock klaarmaken en de benodigde functie aanroepen. en ten slotte maak ik een instantie van de class waar de functie die ik test onderdeel van is en geeft ik de gemockte depandancy mee.

var RepoMock = new Mock<IProjectRepo<Project>>();

RepoMock.Setup(x => x.Create(project)).ReturnsAsync(expected);

var sut = new ServiceProject(RepoMock.Object);

Wanneer ik de hele test heb klaargemaakt ga ik daadwerkelijk mijn test variabelen in de functie stoppen en kijken wat eruit komt. Wanneer dit is wat ik verwacht, is mijn unit test geslaagd. Wanneer dit niet zo is heb ik of een bug te pakken, of ik heb gewoon slechte code geschreven (Waar we niet vanuit gaan natuurlijk 😜).

En de gehele test functie ziet er dan als volgt uit

[Test] public void ProjectService\_ProjectIsNull\_ThrowArgurmentNullException()

{

//Arrange Project

project = null;

bool expected = false;

var RepoMock = new Mock<IProjectRepo<Project>>(); RepoMock.Setup(x => x.Create(project)).ReturnsAsync(expected);

var sut = new ServiceProject(RepoMock.Object);

//Assert var ex = Assert.ThrowsAsync<Exception>(async () => await sut.AddProject(project));

Assert.That(ex.Message, Is.EqualTo("Project cant be null"));

}

### End to End tests (E2E)

Om zeker te weten dat de front-end goed samenwerkt met de back-end en dat alles doet wat hij moet doen maak ik gebruik van end to end tests. Deze tests maken gebruik van de front-end en simuleren dan een gebruiker die de applicatie gebruikt. Er zijn meerdere manieren om dit te doen. Je kan het handmatig doen. Maar ook met een library. Ik doe het doormiddel van een library genaamd playwright. Deze library simuleert op basis van instructies een front-end gebruiker en checkt dan of het alles kan uitvoeren en of wat er verwacht wordt ook daadwerkelijk gebeurt. Zo kan je bijvoorbeeld checken voor tekst of een url of dat er een element bestaat op de pagina.

Alles begint met een

text.describe(name: string, async ({page}) => {});

Binnen deze callback functie worden de instructies gegeven die op de pagina gebeuren. zo kan je bijvoorbeeld een page.goto() gebruiken om naar een specifiek url te gaan. een page.fill() om een form in te vullen of een page.locator().click() om op een element te clicken.

Om goede test cases te maken voor de end to end tests ben ik gaan kijken naar de eerder opgestelde use cases. Dit zijn namelijk klein opgeschreven requirements die beschrijven hoe een deel van de applicatie zou moeten werken en dit is dan ook heel makkelijk om te zetten naar instructies voor de test.

Ik heb iedere user story gemarkeerd met een nummer

MUST have: 1.x.x

SHOULD have: 2.x.x

COULD have: 3.x.x

Deze test cases zet ik in een text matrix om overzichtelijk te hebben wat al werkt. Wat nog niet werkt en wat nog niet is geimplementeerd

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQ | Num | Input | Expected output | Passed |
| Login | 1.1.1 | Email: [testUser@thijmenbrand.nl](mailto:testUser@thijmenbrand.nl) Password: Welkom01 | Redirect naar <http://localhost:8080/#/home> | ✅ |
| Project | 1.2.1 | 1. Klik “Add new project” 2.Vul “TestProject” in het text vel in 3. Klik “Create” | Tabel row met de naam “TestProject” komt onderaan in de tabel te staan | ✅ |
|  | 1.2.2 | No implementation | Het project is verwijdered en gebruiker wordt geredirect naar <http://localhost:8080/#/home> | ❌ |
|  | 1.2.3 | Zie test 1.2.1 | Zie test 1.2.1 | ✅ |
|  | 2.1.1 | No implementation | In het textarea staat de zojuist ingevulde text | ❌ |
|  | 2.1.2 | No implementation | 1. navigeer naar <http://localhost:8080/#/home> 2. In de tabel staat de zojuist aangepaste naam met nieuwe waarde | ❌ |
|  | 2.1.3 | No implementation | In het textarea staat de bewerkte text | ❌ |
| Sprints | 1.3.1 | 1. Navigeer naar <http://localhost:8080/#/home> 2. Klik op een willekeurig project 3. Navigeer naar “Sprints” 4. Klik op “Add Sprint” 5. Vul bij “Sprint name” “testSprint” in 6. Selecteer timerange vandaag - over twee weken 7. Klik op “Create” | De sprint wordt toevoegd aan de dropdown rechtsboven in het scherm. | ✅ |
| Taken | 1.4.1 | 1. Navigeer naar <http://localhost:8080/#/home> 2. Klik op een willekeurig project 3. Navigeer naar “Sprints” 4. Klik op het plusje rechts naast TODO 5. Vul “TestTaak” in | De taak met naam “TestTaak” wordt toegevoegd aan de todo container. | ✅ |
|  | 1.4.2 | 1. Navigeer naar <http://localhost:8080/#/home> 2. Klik op een willekeurig project 3. Navigeer naar “Sprints” 4. Klik op de taak “TestTaak” 5. Pas de taak naam aan naar “TestTaakBewerking” 6. Klik op de “Save” knop | De taak naam wordt aangepast in de container en luidt nu “TestTaakBewerking” | ✅ |
|  | 1.4.3 | 1. Navigeer naar <http://localhost:8080/#/home> 2. Klik op een willekeurig project 3. Navigeer naar “Sprints” 4. Klik op de taak “TestTaakBewerking” 5. Klik op de drie puntjes naast de “Save” knop 6. Klik op de “Delete” knop | De taak met naam “TestTaakBewerking” bestaat niet langer en staat niet meer op de pagina. | ✅ |
|  | 1.4.4 | 1. Navigeer naar “Sprints” 2. Klik op een wilekeurige taak 3. Onder het kopje “Schedule” klik op de date selector 4. Kies wilekeurige data in de toekomst 5. Klik op de knop “Save” | De taak staat nu bij de pagina “Calender” op de horizontale calender | ✅ |
|  | 1.4.5 | Not implemented | De taak staat niet langer op bij de pagina “Calender” op de horizontale calender en heeft ook geen tijd planning meer | ❌ |
|  | 2.4.1 | 1. Klik op een willekeurige taak 2. Typ “Dit is een test beschrijving” in het tekstvak onder description 3. Klik op de “Save” knop | De taak heeft beschrijving “Dit is een test beschrijving” Wanneer je de taak opent zal deze beschrijving te zien zijn | ✅ |
|  | 2.4.2 | Zie test 1.4.4 | Zie test 1.4.4 | ✅ |
|  | 2.4.3 | 1. Pak een willekeurige taak vast 2. Sleep de taak naar een willekeurige container | De taak staat nu in de zojuist heen gesleepte container | ✅ |
|  | 2.5.1 | Not implemented | Een extra container is aan het einde van de andere containers toegevoegd met de ingevoerde naam | ❌ |
|  | 2.4.4 | Not implemented | Alle containers worden niet langer naast elkaar weergegeven maar boven elkaar. | ❌ |
|  | 2.4.5 | 1. Klik op een willekeurige taak 2. Kijk bij Schedule | Er staat een begin en eind datum en tijd | ✅ |
| Planning | 1.5.1 | 1. Navigeer naar <http://localhost:8080/#/home/><ProjectNaam>/calender 2. Klik links boven in de pagina op “Week” | De tijdstappen veranderen van uren naar dagen | ✅ |
| Assets | 2.2.1 | Not implemented |  | ❌ |
|  | 2.2.2 | Not implemented |  | ❌ |
|  | 2.3.1 | Not implemented |  | ❌ |
|  | 2.3.2 | Not implemented |  | ❌ |
|  | 2.3.3 | Not implemented |  | ❌ |

### Code quality

Om de kwaliteit van mijn code te garanderen gebruik ik een tool genaamd SonarLint. Deze checkt iedere regel code om te controleren of er geen rare naming conventions gebruikt wordt, dat classes niet te lang worden enzovoorts.

Op de front-end heeft dit geholpen een aantal rare functies te verbeteren. Deze had ik dan bijvoorbeeld slordig opgezet met niets zeggende variabelen en slechte tanary operations.

In de backend heeft dit er voor gezorgd dat ik nu een nog betere structuur heb. heb namelijk bij ieder domein een extra sublaag toegevoegd welke de parameters valideert. Daarnaast heb ik een validation middleware toegevoegt omdat ik heel veel standaar checks had voor null of 0. Dit wordt nu allemaal geregeld door een globale validationhandler. En ten slotte heb ik een aantal custom exceptions toegevoegd om een beter inzicht te krijgen in wat er mis ging.

Na met sjors gesproken te hebben over deze vernieuwde structuur heb ik besloten mijn eigen validation middleware toch te dumpen en gebruik te maken van Guard Clauses. Dit is een library om voor simpele dingen de checken, zoals null, empty of simpele parameters. En aangezien ik op veel plekken alleen voor null en 0 check kan dit gemakkelijk met Guard clauses. Daarnaast heeft een ingebouwde functie hoogst waarschijnlijk betere checks en protectie dan mijn eigen functies.

# Process documentatie

## Sprint planning

### V1

* Sprint 1
  + **Week 1**
  + Intro bootcamp
  + Project idee bedenken
  + Project idee beschrijven
  + Project requirements stellen
  + Planning maken
  + **Week 2**
  + Prototype: Project aanmaken
  + Prototype: Sprints plannen
  + Prototype: Taken maken
  + **week 3**
  + Unit test: Project aanmaken
  + Unit test: Sprints plannen
  + Unit test: Taken maken
* Sprint 2
  + **Week 1**
  + Development: Inloggen / registreren
  + Unit test: Inloggen / registreren
  + Development: Project Aanmaken / Verwijderen / Aanpassen
  + Unit test: Project Aanmaken / Verwijderen / Aanpassen
  + **Week 2**
  + Development: Bestanden uploaden / verwijderen
  + Unit test: Bestanden uploaden / verwijderen
  + **Week 3**
  + Development: Bestanden Aanmaken / bestanden bewerken
  + Unit test: Bestanden Aanmaken / bestanden bewerken
* Sprint 3
  + **Week 1**
  + Development: Mappen aanmaken
  + Unit test: Mappen aanmaken
  + Development: Bestanden ordenen in mappen
  + Unit test: Bestanden ordenen in mappen
  + **Week 2 & 3**
  + Development: Sprint planning
  + Unit test: Sprint planning
  + Development: Sprint weergave
  + Development: Taken toevoegen aan sprint planningSprint 4
* Sprint 4
  + **Week 1**
  + Development: Taken aanmaken / verwijderen
  + Unit test: Taken aanmaken / verwijderen
  + Development: Taken bewerken / duur geven / due date koppelen
  + Unit test: Taken bewerken / duur geven / due date koppelen
  + **Week 2**
  + Development: Requirements document exporteren naar taken
  + Unit test: Requirements document exporteren naar taken
  + Development: Taken toevoegen aan sprint planning
  + Unit test: Taken toevoegen aan sprint planning
  + **Week 3**
  + Development: Time tracking
  + Unit test: Time tracking
  + Development: Time tracking opslaan en naam geven
  + Unit test: Time tracking opslaan en naam geven
* Sprint 5
  + **Week 1**
  + Development: Time tracking weergeven in schema
  + Development: Time tracking koppelen aan taak
  + Unit test: Time tracking koppelen aan taak
  + **Week 2**
  + Development: Project wiki aanmaken
  + Unit test: Project wiki aanmaken
  + Development: Verschillende pagina’s koppelen aan project wiki
  + Unit test: Verschillende pagina’s koppelen aan project wiki
  + **Week 3**
  + Development: Tekst document als project wiki instellen
  + Unit test: Tekst document als project wiki instellen
  + Development: Integratie tussen verschillende componenten polishen
* Sprint 6
  + **Week 1,2&3**
  + Finishing touch en bug fixes

### V2

* Sprint 2
  + **Week 1**
  + Development: Sprint instellingen
  + Unit test: Taken / Sprint
  + **Week 2**
  + Unit test: Inloggen / registreren
  + **Week 3**
  + Development: Inloggen / registreren
* Sprint 3
  + **Week 1**
  + Development: agenda
  + **Week 2 & 3**
  + Development: agenda
  + Unit test: agenda
* Sprint 4
  + **Week 1**
  + Development: Bestanden uploaden
  + **Week 2**
  + Development: Bestanden uploaden
  + **Week 3**
  + Development: bestanden uploaden
  + Unit test: Bestanden uploaden
* Sprint 5
  + **Week 1**
  + Development: Time tracking
  + **Week 2**
  + Development: Time tracking
  + Development: Settings
  + **Week 3**
  + Unit test: Time tracking
  + Unit test: Settings
* Sprint 6
  + **Week 1&2&3**
  + Bugs fixen, refractoren en presentatie voorbereiden.

## Algemene documentatie

# Terug blik op sprint twee

Ik begon sprint twee met een al staande web app. Ik kon projecten aanmaken, ophalen en selecteren. Echter was ik niet tevreden met de opzet van mijn backend waardoor ik deze besloot te veranderen. Wat ik namelijk in het verleden had was een eenlaagse structuur waar een request vanuit de api binnen kwam en direct in de database werd gezet. Dit is natuurlijk geen goede aanpak en heb dit dan ook veranderd naar een drie laags model waarbij iedere laag een aparte taak heeft. Zo komt er een request binnen bij de API laag. Vanuit hier gaat de data naar de tweede laag waar er wordt gekeken of de data correct is, de data wordt gevormd naar een vorm dat in de database past en andere benodigde dingen worden met de data gedaan. Wanneer dit allemaal klopt gaat de data naar de laatste laag; de data laag. En deze zorgt er enkel voor dat de data in de database terecht komt.

Na dit te hebben geïmplementeerd heb ik me vooral gefocust op de front-end van mijn applicatie. Ik heb deze sprint de taak pagina geïmplementeerd met alle bijbehorende functies. Dit is niet helemaal wat ik voor deze sprint voor ogen had om te doen. Ik had verwacht dat ik sneller verschillende features zou kunnen implementeren, maar dit viel helaas tegen. Daarom zal ik mijn planning ook aanpassen naar deze nieuwe inzichten.

[Sprint planning](https://www.notion.so/Sprint-planning-00ba6af2707e4ecab7a68b00039a5ab1)

Ook liep ik tegen een ander probleem aan. Dit was namelijk dat ik unit testen heel erg stom vond en het niet graag deed en daarnaast was ik zo gefocust op de front-end dat ik on-the-go verschillende backend features implementeerde waardoor ik unit testen totaal over het hoofd zag. Dit is iets wat ik ook in de volgende sprint bewuster mee bezig wil zijn en al het achterstallig werk wil inhalen.

# Sprint 3

### User authorization

Een belangrijk onderdeel van mijn applicatie is de mogelijkheid om in te loggen. Een project moet namelijk beschermd zijn van ongewenste gebruikers en de buiten wereld. Om in te loggen zijn er een aantal verschillende methoden en lagen van veiligheid. Een veel gebruikte manier is het meesturen van een api key. Dit is een lange string van random letters en characters. Een tweede manier is het gebruik van OAuth met OpenId. Dit is niet alleen authenticatie maar ook authorizatie. Dus je kan er gemakkelijk verschillende lagen in bouwen. En wanneer een van de twee dan ook niet correct wordt geimplementeerd kan het zorgen voor een mindere veiligheid. Alhoewel wanneer het just wordt geimplementeerd is het juist een van de best opties voor authenticatie.

Where I left of: User authentication werkt. Echter wanneer een user inlogt kan hij alles van iedereen zien. Dus een user moet alleen zijn eigen dingen kunnen zien. (Claims?) <https://www.youtube.com/watch?v=fhWIkbF18lM>

Ideeën om te implementeren

[x] Wanneer je via het menu naar een ander project gaat. Moet er een functie worden aangeroepen die iedere store (behalve de project en user store) leeg maakt.

[x] Better project support. dus gebaseerd op het id van een project steeds de project data ophalen.

[x] Switchen tussen projecten is nog heel wonky

### Terugblik op sprint 3

Sprint drie stond veelal in het teken van zorgen dat al het achterstallig onderhoud gedaan werd, zoals bugs en unit testing. Dit heb ik dan ook gedaan. Daarnaast stond er een grote nieuwe feature op de planning genaamd authenticatie. Ik wilde graag mijn applicatie beveiligen van ongewenste gebruikers. Dit heb ik gedaan ana de hand van eerder gedocumenteerde authenticatie. Dit verliep eigenlijk best soepel. Ik had in het begin een beetje moeite met de daadwerkelijke implementatie van de backend logina. Ik volgde namelijk een artikel welke goed uitlegde wat er gebeurde, alleen dit was niet ontworpen voor de applicatie architectuur die ik gebruik. Dus met wat andere artikelen en eigen kennis heb ik dit wel werkend gekregen. Op de aanpak heb ik feedback gevraagd aan sjors. Hij gaf me nog een aantal verbeter punten zoals het verplaaten van de daadwerkelijke authenticatie logica vanuit mijn business laag, naar de api laag. Ik heb het gevoel dat ik een goede sprint heb gehad en dat ik heb bereikt wat ik wilde. Toch zijn er een aantal puntjes die ik zou willen verbeteren, deze heb ik hierboven genoteerd.

Ook waren er dingen die beter konden. Zo merk ik dat ik me heel erg vast bijt in geselecteerde projecten en onderwerpen. Ik krijg een tunnel visie naar deze dingen en focus niet meer op dingen die ook belangrijk zijn, zoals algoritmen of de calculus waar ik naast de opleiding nog mee bezig ben. Daarnaast merk ik dat ik het erg leuk vind om nieuwe features uit te rollen en hier mijn tijd in te steken, maar wat ik vergeet / geen zin in heb is het zorgen dat alles getest is en dat het veilig is.

De actiepunten die hieruit volgen zijn dat ik beter moet gaan priotizeren. Ik kan bijvoorbeeld een planning maken waarin ik ieder onderdeel opneem en mij hier dan ook aan houden. Het tweee punt is echt waarborgen dat wanneer ik een functie schrijf, hier ook meteen een unit test voor schrijf.

[ ] Wanneer een gebruiker een project kiest, zet hij een claim bij de gebruiker die de projectID bevat. Op deze manier hoeft deze niet iedere keer meegestuurd te worden

[ ] Op de sprint planning kan je alle sprints een overzicht zien van de taken en de nog niet ingedeelde taken. En wanneer je bij de sprint taken inplant op een tijd en datum die buiten de sprint data valt, dan wordt er eerst een waarschuwing gegeven en dan overgeplaatst naar de andere sprint.

[x] loading animation van element plus

[ ] Een globale planning waar je alle items van ieder project kan zien.

[x] De workload op een taak kunnen instellen.

[ ] Op basis van de workload van taken kan de software zelf een sprint planning maken.

[ ] Sub taken. Dus binnen een taak kan je meerdere taken weer toevoegen. En voordat deze naar done gesleept mag worden moeten alle sub taken worden afgevinkt. (Even bedenken hoe dit met taak dependancy moet werken)

[ ] Mogelijkheid om een taak categorie toe te voegen (zoals bug, documentatie of feature)

[ ] Wanneer een taak niet voor het sprinteinde is afgerond komt er een knopje met “move to next sprint”

**Feedpulse**

Ik ben recent begonnen met e2e testing. Ik had hier een aantal vragen over en heb die aan mark gesteld. Ik begon namelijk met een flow door mijn applicatie. Maar vroeg me af wat ik nu concreet moet testen. Mark had hier als antwoord op dat ik mijn user stories of use cases heel makkelijk als test cases kan schrijven. Daarnaast vroeg ik me af of de manier waarop ik het deed uberhoupt goed was. Nu gebruik ik namelijk een library genaamd playwright die automatisch van instructies test. Hij had hier als opmerking op dat er niet één goede manier was. maar dat ik het zeker op deze manier kon doen. Een nadeel kon wel zijn dat wanneer ik iets in de flow of elementen veranderde dat ik dat in mijn tests ook moet veranderen. Ik zou het ook handmatig kunnen doen en de resultaten opschrijven in een test matrix. En ten slotte vroeg ik wat hij vond van de voortgang van mijn algoritmiek project. Dit vond hij goed en dapper dat ik zo’n complex onderwerp koos.

### Terugblik op sprint 4

Begin sprint vier was ik erg tevreven over feedback die ik kreeg op sprint drie. Mijn documentatie was goed en mijn persoonlijke project was in orde. Het enige waar zeker nog veel aan moest gebeuren was het algoritmiek onderwerp waar ik nog weinig aan had gedaan. Dit heb ik dan ook deze sprint goed opgepakt naar eigen zeggen.

* **Good**
  + **Algoritmiek:** Ik heb een eigen algoritmiek project gekozen waar ik erg in ben geïnteresseerd. Het is moeilijk en complex maar wel zeer interessant wat me gemotiveerd houdt.
  + **Betere communicatie met docenten:** In eerdere sprints merkte ik dat ik het lastig vond om vaak genoeg feedback te vragen. Hier ben ik extra op gaan letten en heb naar mijn mening goede en nuttige feedback gevraagd waar ik veel mee kon.
* **Bad**
  + **Niet goed genoeg naar de feedback van vorige sprint gekeken:** Net voor de sprint oplevering ben ik nog even gaan kijken naar de feedback van vorige sprint. Ik kwam erachter dat ik een onderwerp over het hoofd zag. Deze heb ik dan ook niet helemaal meer kunnen implementeren in deze sprint waar ik toch van baal.
  + niet genoeg afwisseling: Ik heb gemerkt dat ik deze sprint veel bezig ben geweest met het onderwerp algoritmiek. Ik was hier in het begin erg gefocust op maar naar het eide toe begon ik deze toch kwijt te raken omdat het erg eentonig werd.
* **Action**
  + Na terug gekeken te hebben naar de afgelopen sprint en wat goed en minder goed ging heb ik me bedacht dat ik beter moet gaan kijken naar de feedback die ik terug kreeg op de vorige sprint en deze aan het begin van de nieuwe sprint implementeer en mee aan de slag ga. Daarnaast wil ik meer gaan afwisselen tussen algoritmiek en het persoonlijke project om de motivatie te houden en productief te blijven

**Sprint 5**

[x] Rauwe sql queries uitvoeren

[ ] Playwright tests schrijven

[x] Skip-gram implementatie maken

[x] Code quality tool gebruiken

[x] Exception middleware

[x] Validation middleware

[x] Business layer

[x] Project

[x] Sprint

[x] Tasks

[x] Users

[x] DataLayer

[ ] github pipelines instellen

### Terugblik op sprint 5

Deze sprint stond veelal in het teken van algoritmiek en kleine afrondingen van het project. ik heb nog wat aanpassingen gemaakt om hopelijk to komen tot een gemiddeld gevorderde uitkomst. Ik heb veel aan algoritmiek gedaan en heb uitgebreid en concreet onderzoek opstart. Om een zo goed mogelijke toelichting te geven op het onderwerp. Maar daarnaast ook om voor mijzelf inzicht te geven in hoe dit complexe onderwerp nu in elkaar steekt. Ik begon namelijk bij het skip-gram model en bedacht dat ik wel kan proberen te begrijpen hoe dat werk. Maar wanneer ik niet weet wat er onder de moterkap gebeurt leer ik er nog steeds niet veel van. Dus ben ik gaan kijken naar achterliggende theoriën en onderpen zoals neurale netwerken en machine learning. En vanaf daar bouw ik ieder stukje stuk voor stuk op om dadelijk tot een waardig skip-gram model te komen.

* **Good**
  + **Delta programma:** Ik heb dit semester erg mijn best gedaan en heb veel dingen geleerd en gedaan. En dit heeft zich uitbetaald nadat ik van robbert (Semester coach) hoorde dat hij mij graag wilde aandragen voor het delta programma. Dit is denk ik uitdagender dan het gewone curriculum. Ik heb dit semester namelijk gemerkt dat ik best wat voor loop op sommige mede studenten.
  + **Algoritmiek:** Ik ben heel veel kennis opgedaan van AI en machine learning. Ik heb me echt in dit onderwerp gebeten en probeer hier iets moois van te maken. Ik heb een groot maar in eigen mening handig en duidelijk onderzoek opgesteld wat zeker nog niet klaar is.
  + **Bijna klaar:** Ik heb dit semester met niet al te veel moeite bijna alles al afgerond. Hier heb ik dan wel tijd ingestoken en dan ben ik er ook trots op dat het allemaal solide staat.
* **Bad**
  + **Motivatie voor het project:** Ik merk dat wanneer ik langere tijd aan een project werk. Ik de interesse kwijt raak. Ik heb drie a vier sprint heel veel tijd gestoken in mijn persoonlijke project wat ik nog steeds een heel interessant idee vind. Maar na een tijd heb ik er minder en minder zin in. Ook kijk ik om mij heen naar andere tools die iets soortgelijks doen die gemaakt zijn door grote bedrijven. Deze zijn zo geweldig en dan vind ik de mijne daar niet mooi / goed genoeg voor. Dus dan ook iedere keer wanneer ik eraan werk wil ik veel te veel tijd steken in details en vind ik het aan het einde van de rit nog steeds niet goed genoeg wat ook niet bevorderen is voor de motivatie.
* **Action**
  + Na terug gekeken te hebben wat ik mooi en minder mooi vond aan deze sprint zie ik dat ik moet werken aan mijn perfectionisme. Ik wil dat alles perfect is wat me nu steeds vaker in het nadeel werkt. Ik weet nog niet hoe ik dit wil aanpakken maar de bewustwording is een eerste stap.