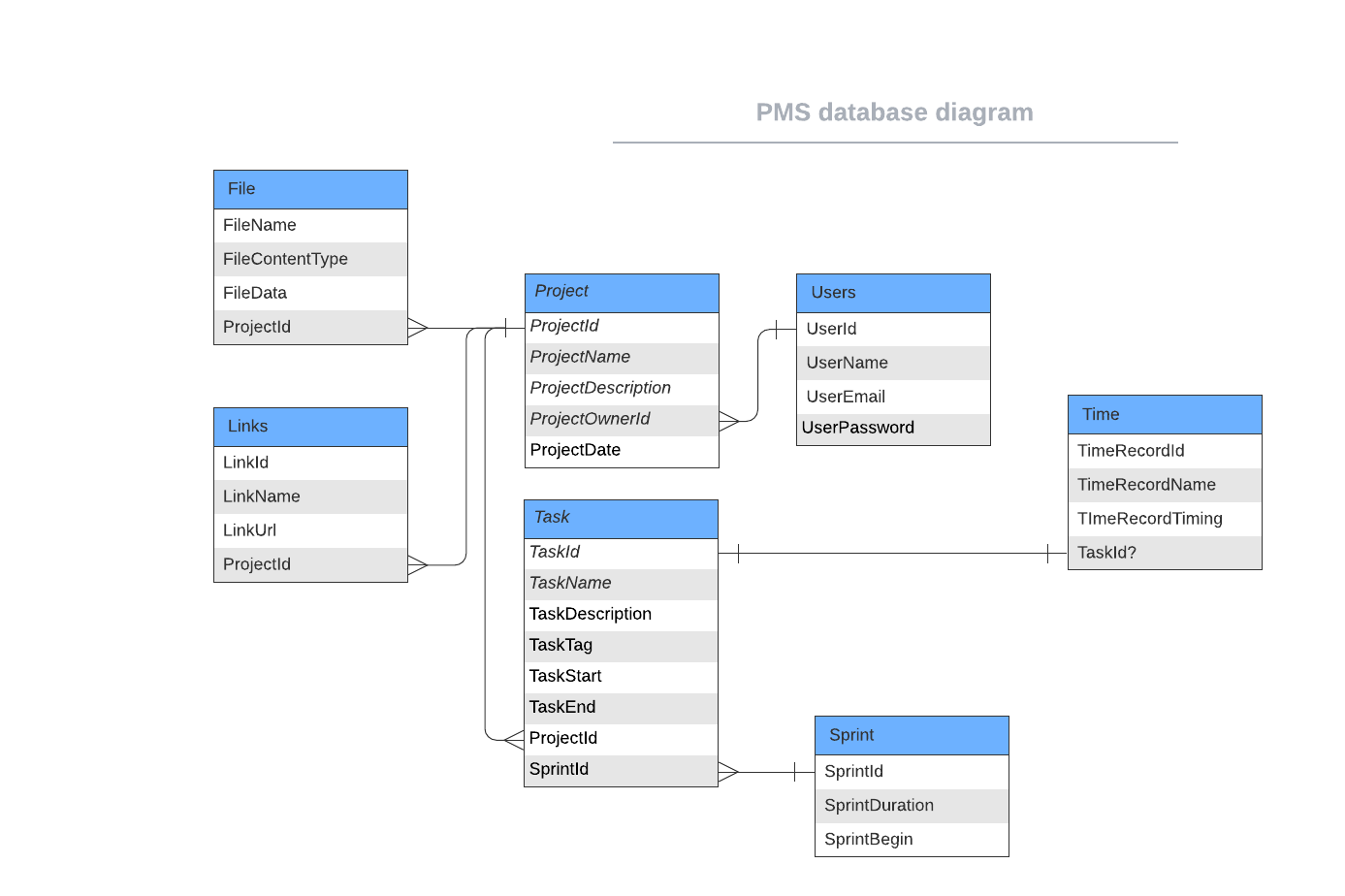
PMS documentatie

# Data

Om alle functionaliteiten goed tot zijn recht te laten komen is het van belang goed te weten hoe verschillende data typen en sources tot elkaar in verband staan. Hiervoor heb ik een UML diagram gemaakt.



# Begrippen

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

# 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

*Updated: 09-03-2022*

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”

# Backend-Api

## [GET]api/Projects

Om alle projecten met bijbehorende data op te vragen

## [GET]api/Projects/{id}

Om een individueel project met bijbehorende data op te vragen

## [POST]api/Projects

Om een nieuw project toe te voegen.

De body bestaat uit de naam en de beschrijving

## [PUT]api/Projects/

Om een project zijn gegevens aan te passen.

In de body moet het id, de nieuwe naam en de nieuwe beschrijving meegeven worden

## [DELETE]api/Projects/{id}

Om een project te verwijderen.