AFSTUDEERVERSLAG

Afstudeerstage Topicus

Mike Horstman

16-01-2022

1. INHOUD

2. VERSIEBEHEER

Versie	Onderwerp	Datum
0.1.0	Eerste opzet	05-10-2021
0.1.1	Onderdelen uit PVA toegevoegd	06-10-2021
0.1.2	Onderzoek	19-10-2021
0.1.3	Ontwerp	11-11-2021
1.0.0	Feedback verwerken	19-11-2021
1.0.1	Realisatie hoofdstuk	12-12-2021
1.0.2	Proces beschrijven	14-12-2021
2.0.0	Feedback verwerken	16-12-2021
3.0.0	Feedback op concept versie verwerken	07-01-2022
3.1.0	Proces aanvullen	11-01-2022
3.2.0	Afronden	16-01-2022

3. SAMENVATTING

ParnasSys is een leerling administratiesysteem voor het primaire onderwijs. Een onderdeel hiervan is het Digitale Rapport. Hierin kunnen leerkrachten rapporten voor leerlingen opstellen. Echter zouden leerkrachten meer vrijheid willen in hoe het rapport eruit komt te zien, zodat deze meer persoonlijkheid krijgt.

Dit afstudeerproject heeft als doel om een prototype te ontwerpen voor een rapport editor, zodat de leerkrachten zelf de opmaak en indeling van een rapport kunnen bepalen. Om dit te kunnen realiseren is het project begonnen met een onderzoek. In dit onderzoek is er gekeken naar de huidige staat van het Digitale Rapport en naar mogelijke technieken die van toepassing kunnen zijn voor de implementatie.

Uiteindelijk is er een prototype van het vernieuwde Digitale Rapport editor opgeleverd. Deze is gebouwd met het framework Angular. In deze editor kunnen leerkrachten zelf rapportdefinities ontwerpen, bestaande rapporten inladen en rapporten exporteren als PDF-bestand.

4. ACHTERGROND

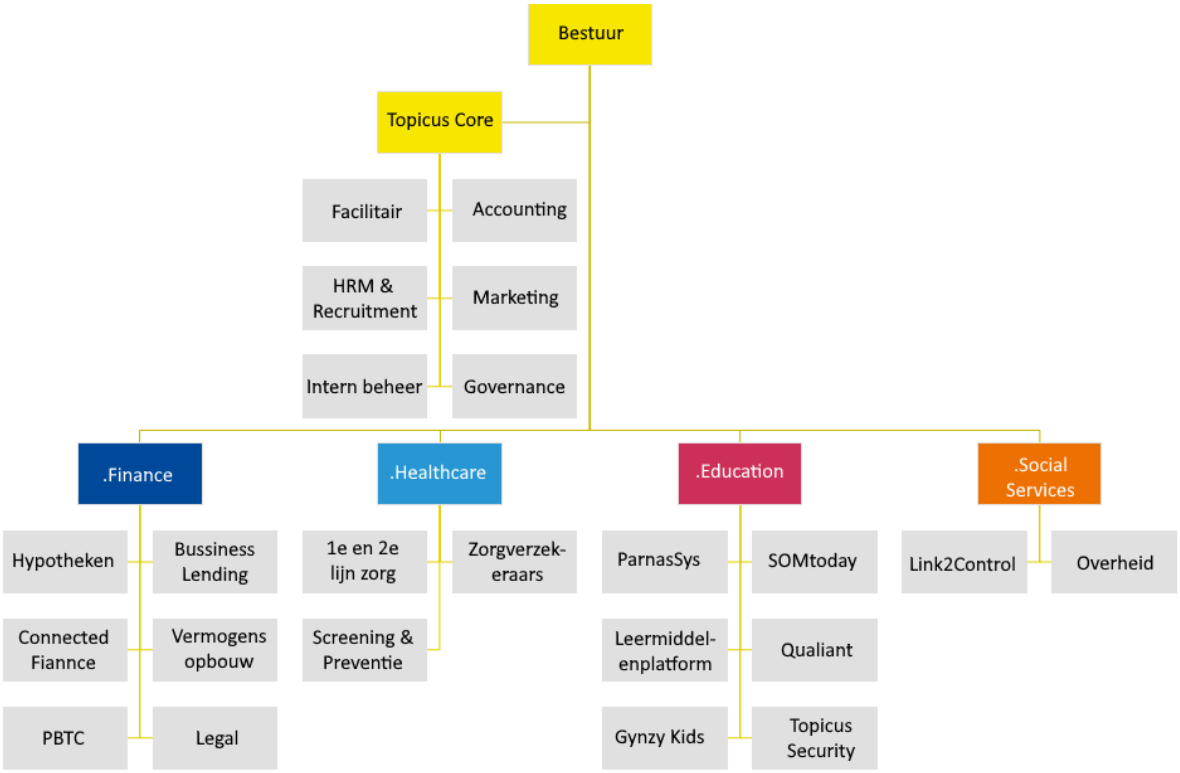
4.1 TOPICUS

Topicus is een softwarebedrijf met meerdere locaties door Nederland, waarvan er vier in Deventer gevestigd zijn. Het bedrijf is in 2001 opgericht door drie mannen en is tegenwoordig uitgegroeid tot ruim 1000 werknemers. Topicus houdt zich voornamelijk bezig met het maken van softwareplatformen die van belang zijn voor de samenleving. Denk hierbij aan systemen voor huisartsen, basisscholen, gemeenten, hypotheken en nog veel meer.
Topicus is onderverdeeld over vier verschillende divisies:

- Zorg
- Onderwijs
- Finance
- Sociaal

4.2 TOPICUS ONDERWIJS

Bij de onderwijs divisie van Topicus wordt er gewerkt aan platformen voor basisscholen en middelbare scholen. Deze IT-oplossingen bieden leerkrachten meer inzicht in resultaten, in vaardigheden en in de sociaal-emotionele ontwikkeling van leerlingen. Deze divisie bestaat uit ca 160 werknemers, waarvan ruim 60% zich bezighoudt met software engineering.



Figuur 1: Organogram Topicus

4.3 PARNASSYS

Topicus heeft meerdere systemen gemaakt voor het primaire onderwijs, waaronder ParnasSys. ParnasSys is een leerling administratiesysteem voor het primaire onderwijs, welke door ca 4 op de 5 scholen in Nederland wordt gebruikt. Hierdoor verwerkt het een enorme variatie aan gegevens van toets leveranciers en van leerkrachten over de voortgang van het kind.

4.4 STAGE

Als vierdejaars Software Engineering student van het Saxion Deventer zal ik gaan werken aan een opdracht voor de Divisie Topicus Onderwijs. Ik zal aan de slag gaan binnen de afdeling ParnasSys welke gevestigd is in Deventer. Het adres hiervan is Singel 9, Deventer.

BEGELEIDER

De bedrijfsbegeleider tijdens de afstudeerstage is Ted Roeloffzen. Ted is een Full stack developer binnen de onderwijs divisie van Topicus en ontwikkelt mee aan het ParnasSys systeem. Zijn opleidingsniveau is HBO en hij heeft 14 jaar werkervaring in dit vakgebied.

5. DE OPDRACHT

5.1 AANLEIDING

ParnasSys is een leerlingadministratiesysteem voor het primaire onderwijs waarin leerkrachten de gegevens van hun leerlingen bij kunnen houden. Deze applicatie is gemaakt in Java.

Figuur 2: Logo ParnasSys

Om ouders en leerkrachten inzage te geven in de voortgang en resultaten van hun kind is er het Digitale Rapport. Dit is een tool waarbij leerkrachten rapporten voor leerlingen kunnen aanmaken. Deze rapporten kunnen ingesteld worden aan de hand van een aantal configuratieopties. Ook kunnen de rapporten geëxporteerd worden als PDF.

Het huidige Digitale Rapport voldoet niet aan de eisen van leerkrachten, waardoor veel scholen een alternatief gebruiken. Leerkrachten zouden graag meer vrijheid willen hebben in hoe het rapport eruit komt te zien, zodat deze meer persoonlijkheid krijgt. Topicus wil daarom graag het Digitale Rapport vernieuwd hebben zodat het mogelijk wordt voor de gebruiker om het rapport zelf in te delen naar eigen wens.

5.2 DOELSTELLING

Het doel van het project is om een prototype te ontwerpen voor een editor van het Digitale Rapport. In deze editor zouden leerkrachten de opmaak en indeling van een rapport helemaal zelf moeten kunnen bepalen. Ook moeten deze rapporten geëxporteerd kunnen worden als PDF bestand en weergegeven kunnen worden in andere applicaties.

Om een zo goed mogelijk prototype te kunnen opleveren zal ik onderzoek gaan doen naar mogelijke technieken en libraries voor de implementatie van de editor. Het gebruik van Angular is hierbij de enige harde eis. Er zullen verschillende mogelijkheden met elkaar vergeleken worden om zo tot de beste oplossingen te komen.

5.3 RESULTATEN

Aan het einde van afstudeerperiode zijn de volgende documenten opgeleverd:

1. Een onderzoeksrapport met daarin:
    - een analyse van het huidige systeem
    - mogelijke technieken voor de implementatie van het prototype
    - mogelijke technieken voor de PDF export
    - mogelijk technieken voor het embedden in andere applicaties
2. Een functioneel en technisch ontwerp met daarin:
    - figma designs van de rapport editor
    - verschillende UML-diagrammen om het systeem te beschrijven
3. Een prototype van de rapport editor in Angular waarin de volgende functionaliteiten geïmplementeerd zijn:
    - het ontwerpen van rapporten
    - het exporteren van rapporten als PDF
    - het embedden van rapporten naar andere applicaties

5.4 HET ONDERZOEK

Een deel van de opdracht bestaat uit het doen van onderzoek, waarvan het doel is om extra inzicht te krijgen in de huidige staat van de applicatie en beschikbare technieken voor de implementatie van het prototype. Dit onderzoek zal uitgevoerd worden aan de hand van een hoofdvraag en een aantal deelvragen:

HOOFDVRAAG

De hoofdvraag zal beantwoord worden aan de hand van het onderzoek. Het is de bedoeling dat de onderzoeksresultaten gebruikt zullen worden voor de implementatie van de Digitale Rapport editor. De hoofdvraag luidt als volgt:

“Hoe kan het Digitale Rapport verbeterd worden zodat leerkrachten meer vrijheid krijgen in het uiterlijk van het rapport?”

DEELVRAGEN

Op basis van bovenstaande hoofdvraag zijn er een aantal deelvragen opgesteld. Deze vragen zullen stapsgewijs beantwoord worden zodat er uiteindelijk een antwoord komt op de hoofdvraag. Bij het opstellen van de deelvragen is er gekeken naar de requirements van de applicatie en naar wat er allemaal nog onduidelijk is.

- Wat zijn de huidige functionaliteiten van het Digitale Rapport?
- Welke technieken kunnen het beste gebruikt worden voor het visualiseren van de data?
- Welke technieken zijn geschikt voor het exporteren van PDF bestanden?
- Met welke bestaande technieken is het mogelijk om een digitale visualisatie uit te serveren naar andere applicaties binnen de ParnasSys suite?

Het doel van bovenstaande deelvragen is om een beter inzicht te krijgen in de huidige stand van zaken en om erachter te komen welke technieken er allemaal beschikbaar zijn voor de implementatie van de applicatie.

6. UITVOERING

De uitvoering van het afstudeerproject is verdeeld in verschillende fases. Deze fases zijn stapsgewijs doorlopen en hebben elk hun eigen resultaten opgeleverd. In de volgende hoofdstukken staat beschreven welke fases er doorlopen zijn, wat het proces hierbij was en welke resultaten hieruit zijn gekomen.

6.1 DE PROJECTFASES

FASE 1: HET ONDERZOEK

Deze fase bestaat uit het uitvoeren van een onderzoek. Aan de hand van de vooraf opgestelde hoofd- en deelvragen is dit onderzoek uitgevoerd en er is een conclusie opgesteld op basis van de resultaten.

In het hoofdstuk: *Onderzoek* zijn de bevindingen en resultaten te vinden.

FASE 2: HET ONTWERPEN VAN DE APPLICATIE

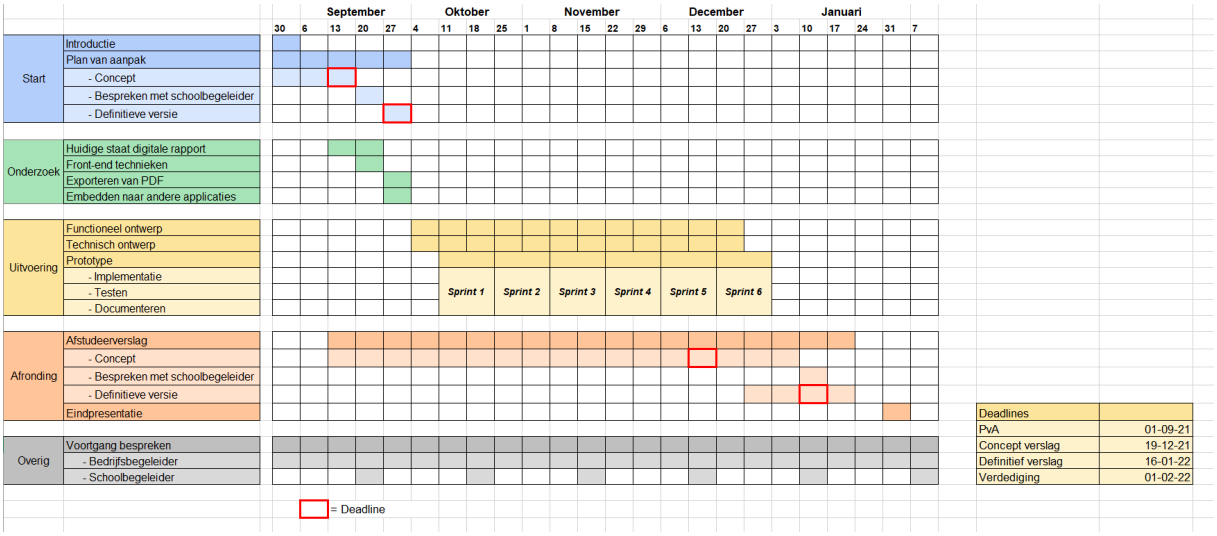
De tweede fase van het project bestaat uit het ontwerpen van wireframes en high-fidelity designs. Aan de hand van de user stories, die zijn opgesteld in de vorige fases, zijn deze ontwerpen voortgekomen.

In het hoofdstuk: *Ontwerp* zijn de ontwerpen en de gemaakt keuzes beschreven.

FASE 3: HET BOUWEN VAN EEN PROTOTYPE

De laatste fase bestaat uit het bouwen van een prototype voor de vernieuwde rapport editor. Tijdens deze fase is er gewerkt aan de implementatie van de designs in Angular op basis van de onderzochte technieken in de onderzoeksfase en de user stories.

In het hoofdstuk: *Realisatie* zijn de processen en resultaten beschreven.



Figuur 3: Planning

Bovenstaande figuur weergeeft de planning voor het gehele project. Deze is in het begin opgesteld en vervolgens gedurende de afstudeerperiode nagelopen. Een uitvergrote versie is te vinden in Bijlage K.

6.2 HET PROCES

In dit hoofdstuk zal het gehele proces beschreven worden wat er doorlopen is gedurende de afstudeerperiode. Zo zijn er verschillende stappen doorlopen om uiteindelijk tot het gewenste resultaat te komen.

TOTSTANDKOMING VAN DE USER STORIES

Aan het begin van de afstudeerperiode waren de requirements nog niet bekend en moesten deze nog opgesteld worden. Aan de hand van gesprekken met verschillende stakeholders en het analyseren van het huidige systeem zijn de requirements en user stories naar voren gekomen.

Vanuit diverse gesprekken is het duidelijk geworden wat de precieze opdracht in houdt en welke functionaliteiten er gewenst zijn. Ook werd hieruit duidelijk dat de functionaliteiten van het huidige systeem terug moeten komen in de nieuwe applicatie. Om deze functionaliteiten in kaart te krijgen is er tijdens het eerste onderzoek dat heeft plaatsgevonden onderzocht hoe het huidige Digitale Rapport in elkaar steekt en wat de functionaliteiten hiervan zijn.

Verder is er ook gekeken naar andere voorbeelden van Digitale Rapporten, om zo inspiratie op te doen. Easyrapport is een soortgelijke applicatie, welke ook gebruikt wordt op basisscholen. Hierin kunnen gebruikers hun eigen rapport maken. Door te kijken naar voorbeelden van deze rapporten zijn er nog een aantal functionaliteiten bij bedacht.

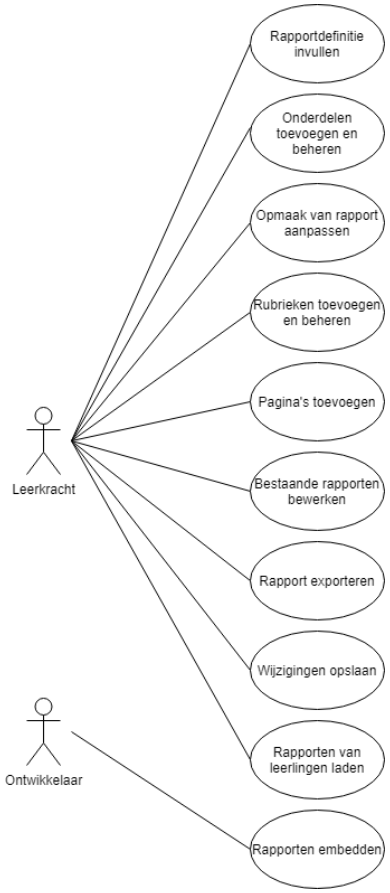
Algemeen

- Een rapportdefinitie kan ingevuld worden.
- Extra pagina's kunnen toegevoegd worden aan een rapport.
- De opmaak van het rapport kan aangepast worden.
- Rapporten kunnen geëxporteerd worden als PDF bestand.
- Bestaande rapporten kunnen bewerkt worden.
- Wijzigingen aan een rapport worden opgeslagen.
- Rapporten van specifieke leerlingen kunnen bewerkt worden.
- Rapporten kunnen weergegeven worden in andere applicaties.

Figuur 4: Een aantal functionaliteiten

Aan de hand van de functionaliteiten is vervolgens een usecasediagram opgesteld. Hiermee zou het duidelijk worden bij welke stakeholder een usecase hoort, welke usecases met elkaar verbonden zijn en welke er afhankelijk van elkaar zijn.
Omdat een aantal van de user stories vrijwel dezelfde functie hebben, is er voor gekozen om alleen de overkoepelende usecases op te nemen in het diagram. Een voorbeeld hiervan zijn de usecases voor het toevoegen van een grafiek, logo, legenda etc. Deze staan beschreven als 'Onderdelen toevoegen aan een rapport'.

Een vergrote versie van dit diagram is te vinden in Bijlage E.



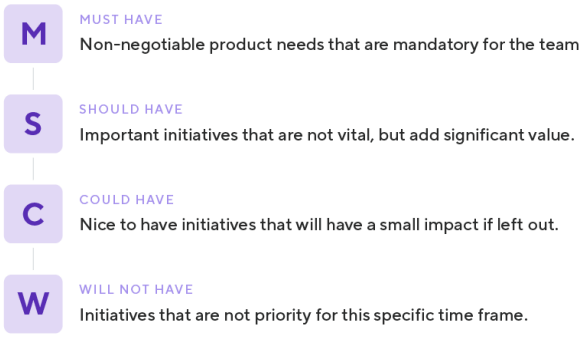
Figuur 5: Usecasediagram voor het Digitale Rapport

Nadat de user stories opgesteld waren heeft de bedrijfsbegeleider er nog even naar gekeken ter controle voor het geval er nog iets over het hoofd gezien was. Uit zijn feedback bleek dat sommige stories iets te globaal opgezet waren en dat ze iets te algemeen konden zijn.

US09	Als leerkracht wil ik kiezen hoe de resultaten weergegeven worden, zodat ik de beste optie kan toepassen.	**Beetje te algemeen**
US10	Als leerkracht wil ik een rapport exporteren naar PDF, zodat ik deze naar ouders kan sturen.	**Heel goed**
US11	Als ontwikkelaar wil ik een rapport uitserveren naar andere applicaties, zodat ik deze niet opnieuw hoef te bouwen.	**Wat bedoel je precies?**
US12	Als leerkracht wil ik het gemiddelde cijfer per subrubriek ophalen, zodat ik	

Figuur 6: Feedback begeleider op user stories

Na het verwerken van deze feedback werden er nog prioriteiten gesteld aan de verschillende user stories. Dit is gedaan met behulp van de bedrijfsbegeleider zodat er niet zomaar aannames gedaan zouden worden. Aan de hand van het MoSCoW principe heeft elke story een prioriteit gekregen, wat zorgde voor een duidelijk overzicht over welke taken het eerst opgepakt moesten worden. De uiteindelijke user stories zijn terug te vinden in Bijlage F en in het functionele ontwerp.



Figuur 7: MoSCoW

ONDERZOEK UITVOEREN

Voordat er begonnen kon worden aan het implementeren van het prototype moest er een onderzoek uitgevoerd worden. Op basis van de onderzoeksresultaten konden er conclusies getrokken worden welke hielpen bij de implementatie.

Voorafgaand aan de afstudeerperiode waren er al een aantal onderzoeksvragen beschreven in de opdrachtomschrijving. Er zou onderzoek gedaan moeten worden naar welk framework het beste gebruikt kan worden voor het prototype, welke techniek gebruikt kan worden voor het exporteren naar PDF en hoe het mogelijk is om een rapport te embedden in andere applicaties.

Na het voeren van een aantal gesprekken was het al vrij duidelijk dat ze bij Topicus een voorkeur hadden voor Angular als framework, aangezien ze dit voor meerdere projecten gebruiken. Doordat dit een van de requirements werd, hoefde hier ook geen onderzoek meer naar gedaan te worden. In plaats hiervan is er in overleg met de bedrijfsbegeleider besloten om dit onderzoek te vervangen voor onderzoek naar bepaalde technieken die van belang kunnen zijn voor de implementatie. De volgende onderzoeksvragen zijn uiteindelijk tot stand gekomen:

- Q1. Wat zijn de huidige functionaliteiten van het Digitale Rapport?
- Q2. Welke technieken kunnen het beste gebruikt worden voor het visualiseren van de data?
- Q3. Welke technieken zijn geschikt voor het exporteren van PDF bestanden?
- Q4. Met welke technieken is het mogelijk om een digitale visualisatie te embedden naar andere applicaties binnen de ParnasSys suite?

ONDERZOEKSAANPAK

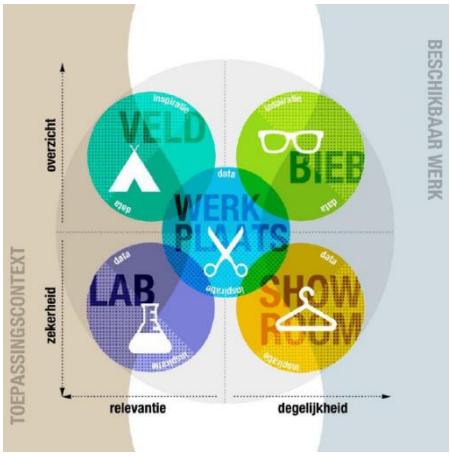
Het onderzoek is volgens een methodische werkwijze uitgevoerd. Dit houdt in dat er voor elke deelvraag een apart onderzoek gedaan is met een eigen onderzoeksmethode. Er is hierbij gebruik gemaakt van het DOT-framework. Dit framework bestaat uit 3 lagen:

De kennisdomeinen waarover je informatie naar boven kunt halen (Wat)
Deze laag bevat drie verschillende kennisdomeinen:

1. Beschikbaar werk: Alles wat er al gedaan is dat relevant is voor de oplossing.
2. Toepassingscontext: Alles wat relevant is aan de context waarin de oplossing gebruikt gaat worden.
3. Innovatieruimte: Voor onderzoek aan de oplossing zelf

De onderzoeksstrategieën en de methoden die je gebruikt om de nodige informatie te verkrijgen (Hoe)
Deze laag bestaat uit vijf verschillende onderzoeksstrategieën:

1. Bieb: Oriëntatie op beschikbaar werk.
2. Veld: De behoeften en wensen van gebruikers vaststellen.
3. Werkplaats: Achterhalen hoe bepaalde dingen werken.
4. Lab: Oplossingen testen.
5. Showroom: Oplossingen vergelijken met beschikbaar werk.



Figuur 8: Het DOT-Framework

De afwegingen die duidelijk maken waarom je kiest voor bepaalde onderzoeksactiviteiten (Waarom)
Deze laag bestaat uit drie afwegingen. Door een keuze te maken tussen deze afwegingen is het makkelijker om de juiste onderzoeksmethodes te kiezen en om uit te kunnen leggen wat je wilt bereiken.

1. Degelijkheid of Relevantie
2. Overzicht of zekerheid
3. Inspiratie of data

Door antwoord te geven op deze basisvragen zal het gemakkelijker zijn om het onderzoek te structureren en erover te communiceren.
Op welk gebied van de methodische aanpak de vragen zich bevinden, is beschreven in onderstaande tabel.

Vraag	Wat?	Hoe?	Waarom?
Q1	Beschikbaar werk	Veld	• Relevantie • Overzicht • Data
Q2	Beschikbaar werk/ innovatieruimte	Bieb/ Werkplaats	• Relevantie • Degelijkheid • Data
Q3	Beschikbaar werk	Bieb	• Relevantie • Degelijkheid • Data
Q4	Beschikbaar werk/ toepassingscontext	Bieb/ Veld	• Relevantie • Degelijkheid • Data

Tabel 1: Gebieden methodische aanpak

7. ONDERZOEK

In dit hoofdstuk is het onderzoek beschreven dat er tijdens de afstudeerstage is uitgevoerd. De vooraf opgestelde onderzoeksvragen zijn onderzocht en op basis van de resultaten zijn er conclusies getrokken welke zullen helpen met het implementeren van het prototype. Een uitgebreide versie van die onderzoek is vinden in het Onderzoeksrapport.

7.1 WAT IS DE HUIDIGE STAAT VAN HET DIGITALE RAPPORT?

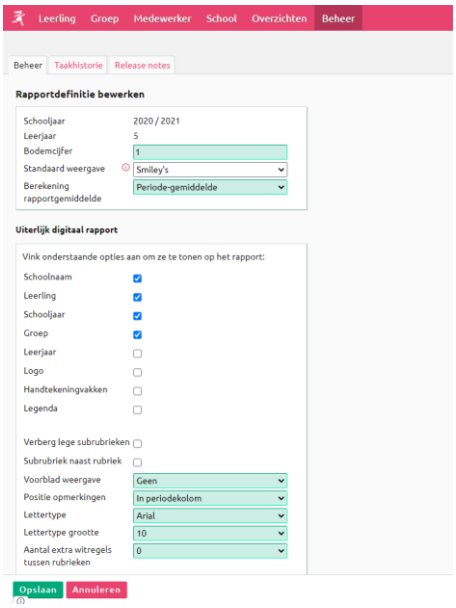
Allereerst is er onderzocht wat de huidige situatie is omtrent het Digitale Rapport. Het doel hiervan was om een beter inzicht te krijgen in het gehele proces en het structuur van het systeem. Ook werden hierdoor de functionaliteiten duidelijk die terug moeten keren in het nieuwe prototype.

DE PROCESSEN IN DE HUIDIGE APPLICATIE

Om de verschillende processen in kaart te brengen is er door middel van een testomgeving van de ParnasSys applicatie onderzocht hoe het aanmaken van een rapport werkt. Tijdens dit onderzoek zijn er meerdere rapporten aangemaakt met verschillende configuraties, zodat alle mogelijke opties duidelijk werden. Hetzelfde is gedaan voor het bewerken en afdrukken van een rapport. Er is ervoor gekozen om op basis van deze resultaten een aantal activity diagrammen op te stellen. De reden hiervoor is dat dit soort diagrammen geschikt zijn om de activiteiten van een systeem weer te geven. Ook is dit gedurende de opleiding veel aan bod gekomen waardoor er al enige ervaring mee was.

Eén van de uitgewerkte handelingen is het aanmaken van een nieuw rapport. De voornaamste redenen om dit proces in kaart te brengen is het feit dat er veel verschillende keuzeopties van toepassing zijn en dat een soort gelijk proces zal plaats vinden in het nieuwe prototype. Hetzelfde geldt voor het aanpassen en afdrukken van een rapport.

Een vergrote versie van deze activity diagrammen is te vinden in Bijlage A en Bijlage B.



Figuur 9: Het aanmaken van een rapport in ParnasSys

Het aanmaken van een rapport

Wanneer een leerkracht een nieuw rapport wil aanmaken gaat deze naar de beheerpagina in de ParnasSys omgeving en drukt op de knop “Rapportdefinitie toevoegen”. **(1)**

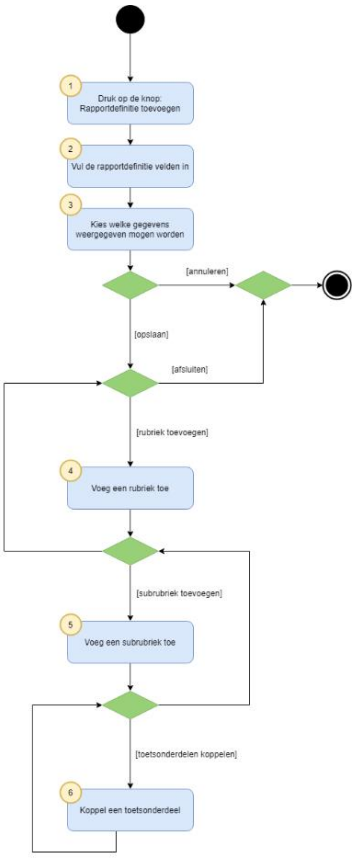
In de nieuwe pagina die verschijnt kunnen de rapportdefinitie velden ingevoerd worden. (Leerjaar, bodemcijfer, standaard weergave, etc.) **(2)**

Ook kan er aangevinkt worden welke gegevens er weergegeven moeten worden op het rapport en welke er verborgen moeten blijven. **(3)**

Wanneer een leerkracht toch geen rapport wil aanmaken, kan het proces gestopt worden en wordt de flow beëindigd. De andere optie is het opslaan van het nieuwe rapport. Wanneer dit gedaan wordt, is er nog de optie om rubrieken toe te voegen of om het proces te stoppen. **(4)**

Nadat er een rubriek wordt toegevoegd, is er de mogelijkheid om subrubrieken hieraan toe te voegen. **(5)**

Wanneer dit het geval is, komt de mogelijkheid er om toetsonderdelen te koppelen aan de subrubriek. **(6)**



Figuur 10: Activity diagram aanmaken rapport

Het bewerken en afdrukken van een rapport

Wanneer een leerkracht een rapport van een leerling wil afdrukken, zal deze naar de ParnasSys pagina van de leerling moeten gaan. **(1)**

De nieuwe pagina die verschijnt weergeeft het rapport van de desbetreffende leerling. De leerkracht kan wisselen tussen rapporten door een ander schooljaar te selecteren. **(2)**

Nu volgen er twee mogelijkheden; het bewerken van het rapport of het afdrukken van het rapport.

Wanneer het rapport bewerkt dient te worden, zal deze eerst ontgrendeld moeten worden door op het slotje te drukken. **(3)**

Nadat er op het bewerk potloodje geklikt is **(4)** kan het rapport bewerkt worden. **(5)**

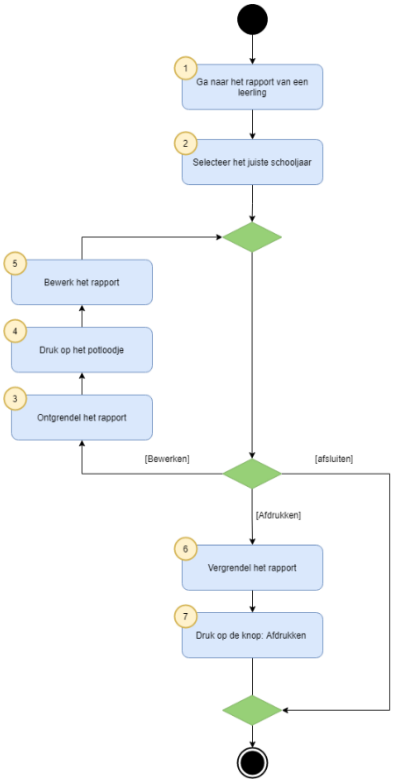
De volgende dingen kunnen aangepast worden:

- Het plaatsen van een opmerking onderaan het rapport
- Het plaatsen van een opmerking bij de rubrieken
- Het aanpassen van de weergeving van de resultaten

De leerkracht krijgt nu de mogelijkheid om de wijzigingen op te slaan of te annuleren, waarna de flow weer terug gaat naar de eerste keuzemogelijkheid.

Bij het afdrukken van een rapport zal deze eerst vergrendeld moeten worden door weer op het slotje te drukken. **(6)**

Hierna kan er op de afdruk knop gedrukt worden om het rapport te exporteren als PDF bestand. **(7)**

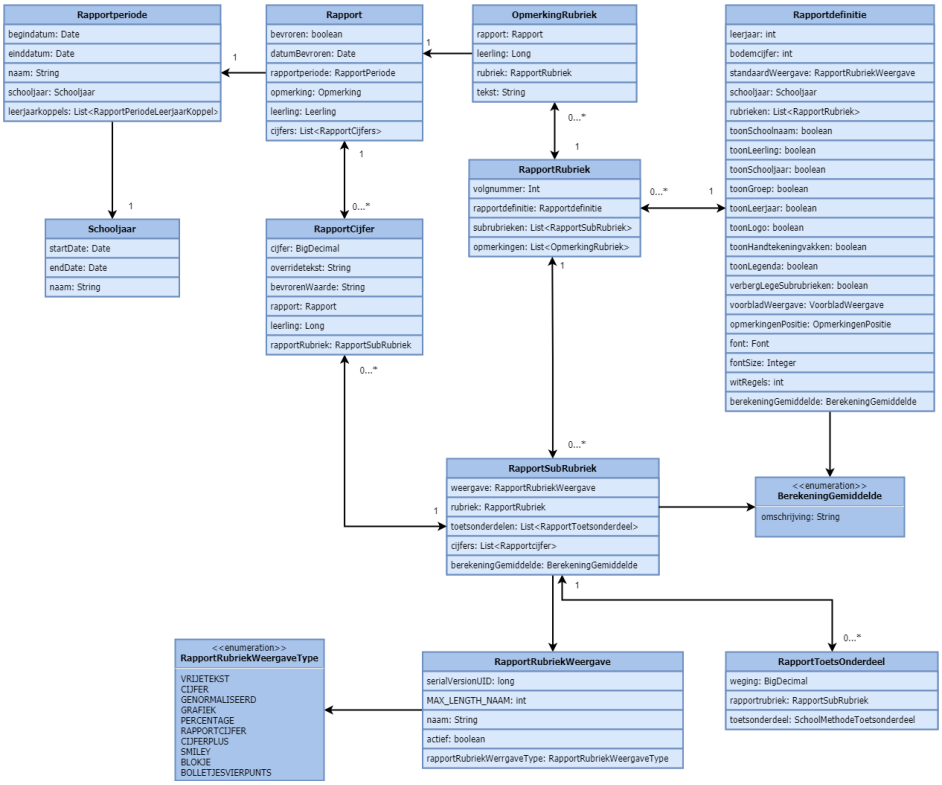


Figuur 11: Activity diagram bewerken en afdrukken rapport

DE STRUCTUUR VAN DE HUIDIGE APPLICATIE

Om meer duidelijkheid te krijgen over de structuur van het huidige rapport is er een classdiagram opgesteld. Door middel van dit diagram is er in een oogopslag te zien welke classes er relevant zijn voor de rapport editor en welke connectie deze met elkaar hebben.

Hiervoor is eerst aandachtig de code doorgespit en zijn verschillende entiteiten onderzocht die betrekking hebben tot het rapport. De applicatie van ParnasSys is extreem groot, waardoor het lastig was om er doorheen te spitten om zo de relevante classes te vinden. Hier is dan ook veel tijd in gaan zitten.



Figuur 12: Classdiagram van de entiteiten

De meest belangrijke class is het Rapport. Deze is voorzien van onder andere een rapportperiode en een lijst met cijfers. Ook heeft deze een vergrendeling en een datum van vergrendeling.
De Rapportperiode geeft aan in welke periode een bepaald rapport valt. Dit wordt gedaan door middel van een schooljaar en een begin- en einddatum. Ook kan de leerkracht bepalen uit hoeveel leerjaren een periode bestaat.
Een rapport kan meerdere rubrieken hebben. Onder deze rubrieken vallen weer meerdere subrubrieken. Aan een subrubriek kunnen toets onderdelen gekoppeld worden, van welke de resultaten getoond zullen worden op het rapport. De leerkracht kan uit verschillende manieren kiezen om de resultaten te tonen (cijfers, smileys, bolletjes, percentages, etc.).
Een rubriek bevat ook een rapportdefinitie. Deze definitie bevat alle configuratiemogelijkheden waaruit de leerkracht kan kiezen om zo te kunnen bepalen hoe een rapport eruit komt te zien.
Zowel aan een rubriek als aan een rapport kan een opmerking toegevoegd worden. Deze kan de leerkracht zelf invoeren per leerling. Een vergrote versie van dit diagram is te vinden in Bijlage C.

DE FUNCTIONALITEITEN VAN DE HUIDIGE APPLICATIE

Eén van de requirements van de nieuwe rapport editor is dat deze dezelfde functionaliteiten van het huidige systeem zal behouden. Door middel van onderzoek naar de applicatie en een testomgeving zijn er een aantal functionaliteiten gevonden. Tijdens het definiëren van de opdracht werd er van vele kanten gezegd dat het huidige Digitale Rapport niet veel voorstelde.

Na zelf onderzoek gedaan te hebben naar de beschikbare functionaliteiten bleek dit inderdaad te kloppen. Bij bijvoorbeeld het aanmaken van een nieuw rapport kan een leerkracht alleen maar een aantal opties selecteren, zonder te kunnen kiezen hoe het er uiteindelijk uit komt te zien. Het uitzoeken en opsommen van de functionaliteiten ging daarom erg voorspoedig.

De volgende functionaliteiten hebben betrekking op het Digitale Rapport:

Het aanmaken van een rapport

Een van de belangrijkste functionaliteiten van het Digitale Rapport is het aanmaken van een nieuw rapport voor een bepaald leerjaar. Bij het aanmaken van een rapport kunnen er een aantal configuratieopties gekozen worden. Dit zijn onder andere een aantal standaard gegevens zoals het leerjaar, de schoolnaam etc. Ook kan er gekozen worden welke elementen er weergegeven wel en niet weergegeven mogen worden.

Het toevoegen van rubrieken aan een rapport

Een andere functionaliteit die het huidige Digitale Rapport bevat is het toevoegen van rubrieken en subrubrieken aan een rapport. Bij het toevoegen van een rubriek kan er gekozen worden of de rubriek wel of niet actief is. Ook zal de naam van de rubriek ingevoerd moeten worden.

Aan elke rubriek kunnen er subrubrieken toegevoegd worden. Ook hierbij kan de naam ingevoerd worden en is er een keuze of deze wel of niet actief moet zijn. Verder kan de gebruiker nog kiezen op welke manier de resultaten weergegeven moeten worden.

Optie	Element	Verklaring	Keuzemogelijkheden
Leerjaar	Tekstvak	Het leerjaar waar het rapport voor bedoeld is.	-
Bodemcijfer	Tekstvak voor getallen	Het cijfer dat minimaal gehaald dient te worden.	-
Standaard weergave	Selectielijst	De manier waarop de resultaten weergegeven worden.	*Cijfers – Tekst – Bolletjes – Progress bar – Smileys - Percentages*
Berekening rapportgemiddelde	Selectielijst	De manier waarop het gemiddelde berekend zal worden.	*Periode-gemiddelde – Voortschrijdend gemiddelde*

Figuur 13: Aantal configuratieopties in ParnasSys

Een tabel met alle configuratieopties is te vinden in Bijlage D.

7.2 WELKE TECHNIEKEN KUNNEN HET BESTE GEBRUIKT WORDEN VOOR HET VISUALISEREN VAN DE DATA?

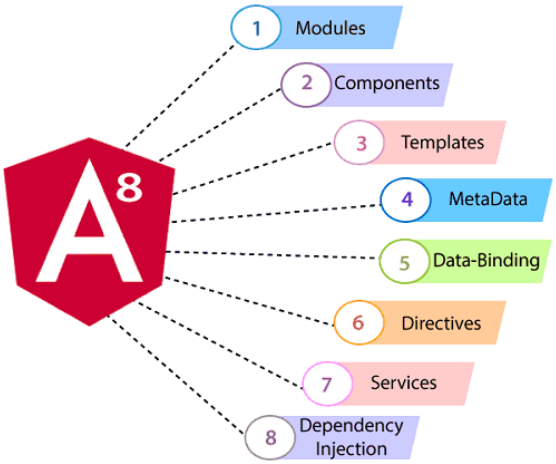
Een van de eisen vanuit Topicus is dat de nieuwe rapport editor gebouwd zal worden in het framework Angular. In dit hoofdstuk is er verder gekeken naar alle mogelijkheden die Angular biedt en welke onderdelen van belang kunnen zijn voor de implementatie van het prototype.

ANGULAR

Angular is een front-end framework ontwikkeld door Google. Angular is de opvolger van AngularJS en is gebaseerd op TypeScript. Angular helpt met het bouwen van responsive websites en applicaties. Het is volledig uitbreidbaar en werkt goed in combinatie met andere libraries.
Angular is een TypeScript framework. TypeScript is afkomstig van JavaScript maar is meer gericht op objectgeoriënteerd programmeren. Angular kan worden toegevoegd aan een HTML-pagina door middel van een <script>-tag en breidt de HTML-attributen uit met Directives.
Angular is een structurele framework voor dynamisme webapplicaties. Hiermee kan HTML gebruikt worden als sjabloontaal en kan de syntaxis uitgebreid worden om de componenten duidelijk en compact weer te geven. Door middel van gegevensbinding wordt er veel van de code geëlimineerd en bespaart het een hoop werk.

BELANGRIJKE ONDERDELEN VAN ANGULAR

Onderstaande diagram weergeeft de verschillende onderdelen van Angular.



Figuur 14: Angular onderdelen

Modules
Angular applicaties zijn modulair en bestaan voornamelijk uit modules genaamd **NgModules.** Deze functioneren als containers voor stukken code die gerelateerd zijn aan elkaar. Zo kunnen ze bijvoorbeeld componenten en service providers bevatten. Functionaliteit van de module kan gedeeld worden met andere modules.

Componenten
Componenten in Angular zijn de bouwstenen waaruit een applicatie bestaat. Elk component bevat de volgende onderdelen:

- Een TypeScript class voor de data en logica
- Een HTML-template die aangeeft wat er wordt weergegeven op de pagina
- Een CSS-selector die definieert hoe de component in een template wordt gebruikt
- CSS-stijlen die worden toegepast op de template

Templates
Elke component heeft een HTML-template welke aangeeft hoe de component wordt weergegeven. Een Angular template geeft een gebruikersinterface weer in de browser, net als gewone HTML, maar met meer functionaliteit.

Data-binding
Data-binding is de automatische synchronisatie van data tussen model en view componenten. Data-binding kan gebruikt worden om dingen te specificeren zoals de bron van een afbeelding, de status van een knop of gegevens voor een bepaalde gebruiker.

Directives
Directives zijn markeringen op DOM (Document Object Model)-elementen zoals elementen, attributen en CSS. Deze kunnen worden gebruikt om aangepaste HTML-tags te maken en de verschijning van de DOM-elementen aan te passen. Angular heeft ingebouwde directives zoals **ngBind** en **ngModel.**

Services
Services zijn verantwoordelijk voor de data van een component. Een component kan bepaalde taken laten uitvoeren door een service, zoals het verkrijgen van data van een server en het verwerken van gebruikers invoer.
Door het scheiden van de view (Component) en de data (Service) zal de applicatie veel schoner en overzichtelijker zijn.

ANGULAR DIRECTIVES

Met directives kunnen aangepaste en herbruikbare HTML-achtige elementen en attributen gespecificeerd worden die data-binding definiëren. Enkele van de meest gebruikte directives zijn:

- **ngClass:** Laat CSS classes dynamisch laden.
- **ngStyle:** Voegt een set HTML-stijlen toe of verwijdert ze.
- **ngModel:** Voegt data-binding in twee richtingen toe aan een HTML form element.
- **ngIf:** Reguliere if-statement waarmee het volgende element kan worden weergegeven op basis van de voorwaarden.
- **ngSwitch:** Instantieert voorwaardelijk één template uit een reeks keuzes, afhankelijk van de waarde van een selectie.
- **ngFor:** Herhaald een element voor elke item in een lijst.

ANGULAR MATERIAL

Angular Material is een UI component library voor Angular. De componenten van deze library helpen bij het bouwen van aantrekkelijke, consistente en functionele webpagina's en applicaties terwijl ze voldoen aan moderne webontwerpprincipes. Het is dan ook geïnspireerd op Google Material Design.
Hieronder een aantal componenten en technieken die van toepassing kunnen zijn op het project:

DRAGGEN EN DROPPEN VAN ELEMENTEN

Een van de functionaliteiten voor het vernieuwde Digitale Rapport editor is het draggen en droppen van elementen. Op deze manier kunnen leerkrachten zelf de gewenste elementen op een plek naar keuze slepen in het rapport.
De **DragDropModule** biedt een manier om eenvoudig interfaces voor draggen en droppen te maken, met ondersteuning voor bijvoorbeeld slepen, sorteren binnen een lijst, animaties, aangepast slepen en vergrendeling langs de as. (Angular Material, sd)

PORTAL

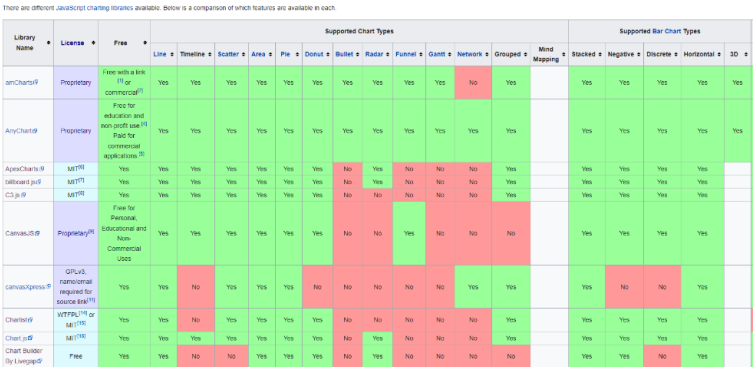
Een van de gevraagde functionaliteiten is een live weergave van het rapport tijdens het bewerken. Bij het kiezen van componenten en elementen zullen deze dynamisch geladen moeten worden. De **portals package** biedt een flexibel systeem voor het omzetten van dynamische content in een applicatie.
Een portal is een stukje UI dat dynamisch kan worden weergegeven in een open slot op de pagina. Dit stukje UI kan drie verschillende dingen zijn:

- Component
- TemplateRef
- DOM-element

Het open slot wordt de PortalOutlet genoemd. (Angular Material, sd)

HET GENEREREN VAN GRAFIEKEN

Eén van de requirements was dat een leerkracht een grafiek met resultaten moet kunnen weergeven op het rapport. Dit zal gaan om standaard grafieken als lijn- en staafgrafieken. Om erachter te komen welke library hier het meest geschikt voor was is er een vergelijking gemaakt op basis van een aantal criteria.



There are different JavaScript charting libraries available. Below is a comparison of which features are available in each.

Library Name	License	Free	Supported Chart Types													Supported Bar Chart Types				
			Line	Timeline	Scatter	Area	Pie	Donut	Bullet	Radar	Funnel	Gantt	Network	Grouped	Mind Mapping	Stacked	Negative	Discrete	Horizontal	3D
amCharts	Proprietary	Free with a link or commercial	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes		Yes	Yes	Yes	Yes	Yes
AnyChart	Proprietary	Free for education and non-profit use. Paid for commercial applications.	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
ApexCharts	MIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes		Yes	Yes	Yes	Yes	
billboard.js	MIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes		Yes	Yes	Yes	Yes	
C3.js	MIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes		Yes	Yes	Yes	Yes	
CanvasJS	Proprietary	Free for Personal, Educational and Non-Commercial Uses	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No	No		Yes	Yes	Yes	Yes	
canvasXpress	GPLv3, noncommercial required for source link	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes		Yes	No	No	Yes	
Chartist	WTFPL or MIT	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes		Yes	Yes	Yes	Yes	
Chart.js	MIT	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	Yes		Yes	Yes	Yes	Yes	
Chart Builder By Livejapi	Free	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No	No	No		Yes	Yes	No	Yes	

Figuur 15: Beschikbare libraries (Comparison of JavaScript charting libraries, 2021)

Allereerst moest er een kleine selectie gemaakt worden van de 50+ libraries die beschikbaar waren. Deze selectie kon gemaakt worden op basis van het type licentie en de prijs die eraan verbonden is. Het idee hierachter was dat het geen zin zou hebben om zoveel libraries met elkaar te gaan vergelijken terwijl een belangrijk punt toch wel is dat de library geen geld zou kosten of copyright zou hebben. Door op deze manier een vergelijking te maken bleven er nog vijf libraries over om mee verder te gaan:

- ApexCharts.js
- Billboard.js
- Chartist.js
- Chart.js
- Plotly.js

De uiteindelijk overgebleven 5 libraries zijn vervolgens met elkaar vergeleken op basis van een aantal uitgekozen criteria waarvan ik dacht dat die belangrijk zouden zijn voor het uitkiezen van een goede library. Doordat je niet gelijk de eerste de beste optie kiest, maar door middel van criteria meerdere opties met elkaar vergelijkt zal de kans groter zijn dat de juiste keuze gemaakt wordt.

Om de populariteit te meten is er gekeken naar het aantal downloads van iedere library. Ook is er bijvoorbeeld gekeken naar de verschillende soorten grafieken die beschikbaar zijn en welke aanpassingen er gedaan kunnen worden. De uitkomsten van dit onderzoek zijn hieronder te vinden.

HET AANTAL DOWNLOADS

Door te kijken naar hoe vaak een library gedownload is, kan er afgelezen worden hoe populair een bepaalde library is.

	ApexCharts.js	Billboard.js	Chartist.js	Chart.js	Plotly.js
Number of downloads	9.989.356	368.486	4.417.052	70.877.646	5.146.748

Tabel 2: Aantal downloads met npm van 22-09-2020 t/m 22-09-2021 (npm-stat, sd)

Op basis van bovenstaande gegevens kunnen de volgende conclusies getrokken worden:

- Chart.js is veruit het populairst met meer dan 70 miljoen downloads in een jaar tijd.
- Billboard.js heeft vergeleken met de andere libraries veruit de minste downloads

ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
☆☆☆☆	☆	☆☆☆	☆☆☆☆☆	☆☆☆

AANGELEVERDE GRAFIEKEN EN DIAGRAMMEN

Het aantal beschikbare grafieken en diagrammen is een belangrijke criteria om op te letten. Hoe meer variatie er is in diagrammen hoe meer keuzevrijheid een leerkracht heeft.

	ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
Lijndiagram					
Staafdiagram					
Cirkeldiagram					
Bellendiagram					
Vlakdiagram					
Spreidingsdiagram					
Radardiagram					

Tabel 3: Aangeleverde grafieken

Wel beschikbaar
Niet beschikbaar

Op basis van bovenstaande gegevens kan het volgende geconcludeerd worden:

- Over het algemeen worden alle standaard diagrammen ondersteund door de libraries.
- Chartist.js is de enige library die geen ondersteuning biedt voor twee toch wel belangrijke diagrammen.

ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
☆☆☆☆☆	☆☆☆☆☆	☆☆☆	☆☆☆☆☆	☆☆☆☆☆

RENDER TECHNOLOGIE

Voor het renderen van de diagrammen zijn er twee verschillende mogelijkheden; op basis van SVG of op basis van HTML5 Canvas. SVG en Canvas zijn data visualisatie technologieën die gebruikt kunnen worden voor het weergeven van grafieken.

SVG

SVG is used to describe Scalable Vector Graphics, a retained mode graphics model that persist in an in-memory model that can be manipulated through code results in re-rendering. Similar to HTML, SVG is built into the document using elements, attributes, and styles. (Microsoft: SVG vs canvas: how to choose, 2016)

SVG is een XML-bestandsindeling die is ontworpen om vectorafbeeldingen te maken. Een van de grootste voordelen is de schaalbaarheid. Schaalbaar zijn heeft het voordeel dat een vectorafbeelding vergroot of verkleint kan worden terwijl de kwaliteit en scherpte behouden blijft.
SVG gebruikt een behouden benadering, waarbij er eenvoudig tekeninstructies gegeven kunnen worden. De grafische API van de browser maakt hiervan een in-memory model van de uiteindelijke uitvoer en vertaalt dit in tekenopdrachten.

HTML5 CANVAS

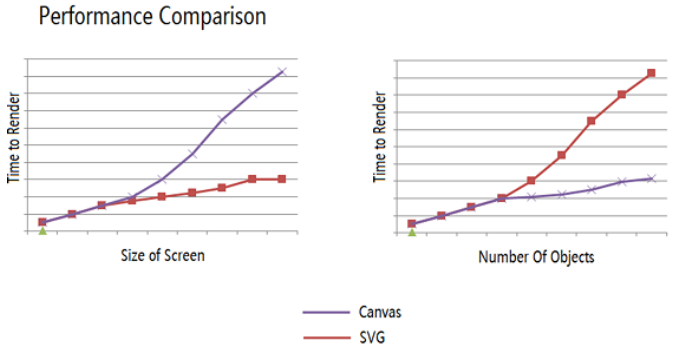
Another approach to provide a richer graphical experience for users provided by the <canvas> tag … Immediate mode graphic rendering is a "fire and forget" model that renders graphics directly to the screen and then subsequently has no context as to what was done. (Microsoft: SVG vs canvas: how to choose, 2016)

Het <canvas>-element onthult een oppervlak waar er pixel voor pixel afbeeldingen gemaakt en bewerkt kunnen worden door middel van JavaScript. Wanneer een afbeelding op het canvas getekend is, dan wordt deze niet meer bijgehouden door het canvas. De gebruiker moet zelf de tekeninstructies uitwerken en specificeren wat er moet worden bijgewerkt.

SVG	Canvas
Goede schaalbaarheid	Slechte schaalbaarheid
Geeft een betere performance met het gebruik van een klein aantal objecten op een groter scherm.	Voor een groter aantal objecten op een klein scherm geeft het een betere performance
Kan bewerkt worden door middel van script en CSS	Kan alleen bewerkt worden door middel van script

Tabel 4: Verschil tussen de render technologieën

Door te kijken naar de performance van beide technieken en naar de toepassing waarin deze gebruikt zal worden, gaat de voorkeur uit naar een render technologie op basis van SVG. SVG geeft aan een betere performance te hebben bij het gebruik van een klein aantal objecten op een groter scherm, en dat is precies de toepassing waar deze voor gebruikt zal worden aangezien een rapport weinig grafieken zal bevatten.



Figuur 16: Verschil in performance tussen SVG en Canvas

ApexCharts.js (SVG)	Billboard.js (SVG)	Chartist.js (SVG)	Charts.js (Canvas)	Plotly.js (SVG)
☆☆☆☆☆	☆☆☆☆☆	☆☆☆☆☆	☆	☆☆☆☆☆

VERRIJKING EN ONDERHOUD

Het kijken naar de GitHub repository van de desbetreffende library kan helpen met het maken van een keuze. Aan de hand van de activiteit en populariteit van een repository kunnen er een aantal dingen gegarandeerd worden:

- De library zal in de loop van de tijd worden onderhouden
- De library zal verbeterd worden met nieuwe functies

	ApexCharts.js	Billboard.js	Chartist.js	Chart.js	Plotly.js
Stars	10700	4900	12700	54800	14000
Watchers	127	85	357	1400	286
Forks	824	310	2600	11200	1600
Contributors	105	141	68	381	167
Open issues	748	129	352	98	1125
Last commit	22/09/2021	17/09/2021	01/11/2019	14/09/2021	16/09/2021

Tabel 5: GitHub repositories statistieken (23-09-2021)

Goed
Slecht

Op basis van bovenstaande gegevens kunnen de volgende dingen geconcludeerd worden:

- Over het algemeen hebben al deze libraries goede GitHub statistieken. Er kan verwacht worden dat ze in de loop van de tijd onderhouden blijven worden.
- Chartist.js wordt de laatste tijd niet zo goed onderhouden. De laatste commit is dan ook al bijna twee jaar geleden geweest, terwijl er nog redelijk wat issues openstaan.
- ApexCharts.js, Chartist.js en Plotly.js hebben veel openstaande issues.

ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
☆☆☆	☆☆☆☆	☆☆	☆☆☆☆☆	☆☆☆

ONTWERP EN AANPASSINGSOPTIES

Het vergelijken van ontwerp- en aanpassingsmogelijkheden is lastig. Elke library biedt aangepaste opties en het is moeilijk om ze te vergelijken.

	ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
Titel					
Legenda					
Zoomen					
Responsive					
Kleuren aanpassen					
Animaties					

Tabel 6: Ontwerp en aanpassingsopties per library

Wel beschikbaar
Beschikbaar via een plugin
Niet beschikbaar

Uit bovenstaande gegevens is te concluderen dat:

- De meeste libraries ondersteunen de belangrijkste aanpassingsopties.
- De diagrammen van Chartist.js bevatten geen titel bovenaan het diagram. De legenda en zoom functionaliteit moet geïnstalleerd worden via een plugin.
- De diagrammen van Billboard.js zijn uit zichzelf niet responsive.

ApexCharts.js	Billboard.js	Chartist.js	Charts.js	Plotly.js
☆☆☆☆☆	☆☆☆	☆☆	☆☆☆☆	☆☆☆☆☆

7.3 HOE KAN HET DIGITALE RAPPORT GEËXPORTEERD WORDEN NAAR PDF?

Een belangrijke vraag bij het bouwen van het prototype is de vraag hoe een rapport geëxporteerd kan worden als een PDF bestand. Hierbij moet er bijvoorbeeld rekening gehouden worden met de opmaak van een rapport en dat deze behouden wordt. In dit hoofdstuk is er onderzoek gedaan naar mogelijke oplossingen voor dit probleem en naar de huidige manier van exporteren.

HET HUIDIGE EXPORTEREN VAN HET DIGITALE RAPPORT NAAR PDF

PDF export gaat door middel van **Velocity Templates (.vm)** in Java.
Op basis van deze templates wordt de indeling van het rapport bepaald. Dit is een voorgedefineerd HTML bestand.
Velocity is een op Java gebaseerde template engine. In deze templates kan verwezen worden naar methodes die zijn gedefinieerd in Java code. Op deze manier staat de opmaak van de het document vast, en kan alleen de data dynamisch zijn.

MOGELIJKE OPLOSSINGEN

Het exporteren van bestanden naar PDF kan via twee verschillende manieren gedaan worden; door middel van Server-side en Client-side export.

SERVER-SIDE EXPORTEREN

Bij server-side exporteren wordt de PDF gemaakt in de backend van de applicatie en vervolgens naar de gebruiker teruggestuurd. Een voordeel hieraan is dat er veel meer opties zijn qua opmaak en indeling van het PDF-bestand.

Wkhtmltopdf

Wkhtmltopdf is een uitgebreide command-line tool om HTML naar PDF te renderen met behulp van de QT Webkit Engine. Deze tool is erg gemakkelijk te gebruiken. Na het downloaden van de juiste build kan deze gelijk uitgevoerd worden.

```
$ wkhtmltopdf https://wikipedia.org/wiki/Wiki demo.pdf
```

Hiervoor is wel een HTML-bestand nodig, welke op de juiste manier opgesteld is. Zo moet de juiste styling en scripts in dit bestand staan. Ook is het mogelijk om een link naar een webpagina mee te geven in plaats van een HTML-bestand.

Voordelen	Nadelen
+ De tekst kan gemarkeerd worden	- Coderingsfouten kunnen optreden
+ Tekst en SVG-bestanden zijn schaalbaar	- Maakt gebruik van server resources
+ Kan meerdere pagina's goed verwerken	
+ Kleine bestandsgrootte	

Tabel 7: De voor- en nadelen van Wkhtmltopdf

CLIENT-SIDE EXPORTEREN

Bij client-side exporteren wordt de PDF gemaakt binnen de client. Een voordeel hieraan is dat er niet eerst requests heen en weer gestuurd hoeven te worden naar de server maar dat alle logica in de browser gebeurt.

JsPDF

JsPDF is een package die het gemakkelijk maakt om PDF-documenten te genereren met een klik op een knop. Deze package werkt door middel van het invoegen van elementen in een nieuw pdf-bestand. Dit kunnen onder andere teksten, afbeeldingen en vormen zijn.

```
$ var doc = new jsPDF()
$ doc.text('Jspdf text!', 10, 10)
$ doc.save('test.pdf')
```

Aan elk element wordt een x en y locatie meegegeven. Deze waardes bepalen de plek van het element op de pdf-pagina. Om op deze manier HTML om te zetten naar PDF is erg omslachtig, aangezien alle HTML-elementen stuk voor stuk opnieuw nagemaakt moeten worden in JavaScript. Daarbij zal niet alle styling opnieuw overgenomen kunnen worden.

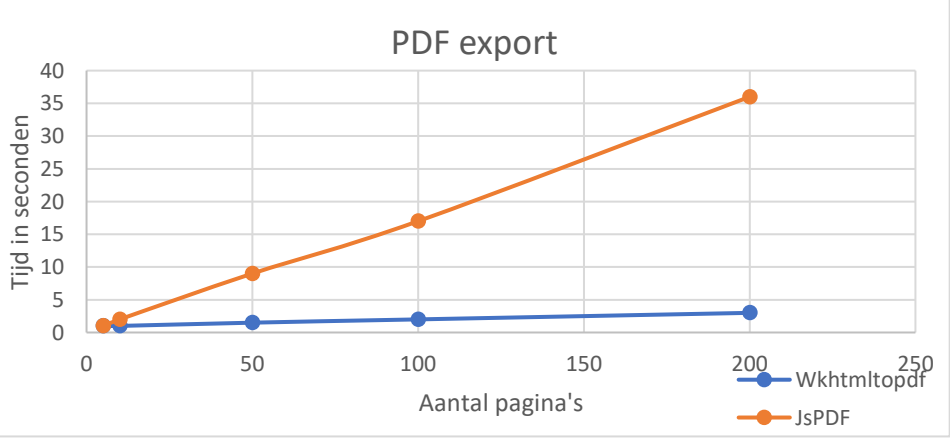
Een oplossing hiervoor is door gebruik te maken van een extra library: **HTML2Canvas.**

HTML2Canvas is een script waarmee er screenshots gemaakt kunnen worden van een webpagina of delen ervan. Deze screenshot is gebaseerd op de DOM en is misschien niet 100% nauwkeurig.
Het script doorloopt de DOM van de pagina waarop het is geladen. Vervolgens verzamelt de informatie over de elementen om er vervolgens een weergave van de pagina van te bouwen. Het script maakt dus niet echt een screenshot van de pagina, maar bouwt er een weergave van op basis van de eigenschappen die het uit de DOM leest.
Als gevolg hiervan kan het alleen eigenschappen weergeven die het begrijpt, wat betekent dat veel CSS-eigenschappen niet zullen werken. Dit is een groot nadeel aangezien de PDF het liefste zo accuraat mogelijk wordt weergegeven.

Voordelen	Nadelen
+ Maakt gebruik van client resources + Zeer flexibel + Slaat alle CSS-inhoud op omdat het een screenshot maakt van de sectie	- Tekst en SVG-bestanden zijn niet schaalbaar - Tekst kan niet gemarkeerd worden - Inhoud kan wazig worden - Extreem hoge bestandsgrootte doordat het uit afbeeldingen bestaat. - Meerdere pagina's zijn vrijwel onmogelijk. - Bij veel pagina's kan het lang duren

Tabel 8: De voor- en nadelen van jsPDF in combinatie met HTML2Canvas

Onderstaande grafiek weergeeft het tijdsverschil tussen beide opties wanneer het aantal pagina's toeneemt. Te zien is dat bij een kleine oplage er niet veel tijdsverschil zit. Echter wanneer het aantal pagina's oploopt doet JsPDF er veel langer over dan Wkhtmltopdf.
Bij de server-side optie blijft de tijd aardig gelijk, terwijl bij JsPDF de het tijdsverschil lineair toeneemt.



Figuur 17: Tijdsverschil tussen de libraries bij een groei van het aantal pagina's

7.4 MET WELKE TECHNIEKEN IS HET MOGELIJK OM EEN DIGITALE VISUALISATIE TE EMBEDDEN OF UIT TE SERVEREN NAAR ANDERE APPLICATIES BINNEN DE PARNASSYS SUITE?

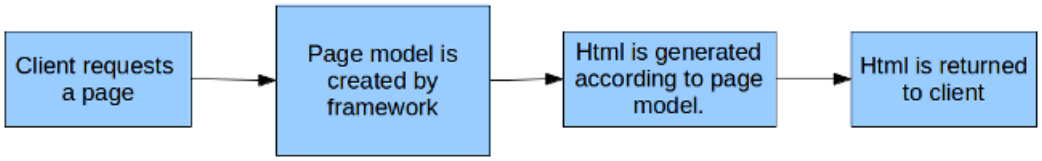
Het Digitale Rapport moet kunnen weergegeven worden in andere applicaties van ParnasSys. Op deze manier kan het component hergebruikt worden zonder dat deze opnieuw gebouwd hoeft te worden.
In dit hoofdstuk zijn de verschillende mogelijkheden voor het embedden van het rapport in andere applicaties onderzocht. Hiervoor is er gekeken hoe de ParnasSys applicatie in elkaar steekt zodat hier rekening mee gehouden kan worden met het maken van een keuze.

PARNASSYS APPLICATIE

Het uiteindelijke Digitale Rapport zal weergegeven moeten worden in andere ParnasSys applicaties. Om erachter te komen aan welke eisen de techniek die hiervoor gebruikt gaat worden moet voldoen, is er onderzoek gedaan naar één van de ParnasSys applicaties.
Applicaties binnen de ParnasSys suite zijn gebouwd in Java. Voor de front-end is gebruik gemaakt van het framework Wicket.

Wicket is een Java web development framework. Wicket is gebaseerd op componenten waardoor het in schil contrast staat met enkele andere oplossingen voor webprogrammering.

Elke pagina in een Wicket bestaat uit een Pagina model en een HTML-bestand. Het model van de pagina wordt aan de serverzijde gebouwd en op basis hiervan wordt de HTML gegenereerd en gestuurd naar de client.



Figuur 18: Voorbeeld page requests in Wicket

BESCHIKBARE TECHNIEKEN

Voor het embedden van het Digitale Rapport in andere applicaties, zijn er meerdere opties beschikbaar.

IFRAMES

Een iFrame is een stukje HTML-code dat een frame uit een andere website in kan sluiten. Deze kunnen dan op de eigen website weergegeven worden. Hetzelfde kan gedaan worden bij webapplicaties. Een iFrame kan dus een verwijzing hebben naar een pagina uit de Angular applicatie.

Voordelen	Nadelen
+ JavaScript encapsulation + De embedded pagina blijft intact	- Geen ondersteuning voor dynamische componenten - Content in iFrames wordt niet gevonden door Google 's search engine

Tabel 9: Voor- en nadelen van iFrames

WEBCOMPONENTS MET ANGULAR ELEMENTS

Webcomponents zijn een set web platform APIs waarmee er herbruikbare aangepaste elementen gemaakt kunnen worden (Custom Elements). Deze elementen kunnen vervolgens gebruikt worden binnen HTML-pagina's en webapplicaties.
Custom components bouwen voort op de Web Component standaarden en kunnen worden gebruikt met elke JavaScript bibliotheek of -framework dat met HTML werkt.

Benodigde packages

Voor het werken met web-components heeft Angular een eigen package beschikbaar gesteld. Met behulp van Angular Elements is het eenvoudig om Angular componenten te verpakken in een Web Component. Angular Elements is een package welke zorgt voor een brug tussen het Angular Component en de ingebouwde DOM API.
@angular/elements

Browser support

Webcomponents worden in de meeste moderne browsers ondersteund. Voor ondersteuning op oudere browsers zal gebruik gemaakt moeten worden van Polyfills.

Chrome	Edge	Firefox	Opera	Safari	IE

Tabel 10: Browser ondersteuning

Supported natively
Need polyfills
Not supported

Voordeel

Het aanmaken van web-components binnen Angular is erg eenvoudig met behulp van de Angular Elements library. Binnen een paar stappen is het component omgezet naar een webcomponent.
Bied ondersteuning voor meerdere frameworks, zo dus ook Angular.
Stijl-inmenging wordt voorkomen doordat webcomponent een eigen stijl-context bieden. Styling van buiten het component zal niet toegepast worden.

Nadelen

Angular Elements richt zich alleen op het verpakken van Angular componenten in web-components. Bij het verpakken van grotere delen van een applicatie zal de communicatie en navigatie tussen web-componenten nog zelf geregeld moeten worden door de ontwikkelaar.
Voor het aaneenschakelen van de webcomponent bestanden zijn externe packages nodig.

Voordelen	Nadelen
+ Eenvoudig + Ondersteuning voor meerdere frameworks + Style encapsulation + Applicatieonderdelen kunnen hergebruikt worden	- Het verpakken van grotere applicatie-onderdelen in Angular kan voor problemen zorgen - Externe packages vereist - Vereist extra maatregelen voor communicatie en navigatie

Tabel 11: Voor- en nadelen van Webcomponents

7.5 CONCLUSIE

In voorgaande hoofdstukken zijn meerdere mogelijke technologieën benoemd die gebruikt kunnen worden voor het visualiseren, embedden en exporteren van het Digitale Rapport.

Het visualiseren van de data

Het framework dat gebruikt gaat worden voor de applicatie is Angular. Dit front-end framework is uitermate geschikt voor het maken van dynamische webapplicaties. Er zal gewerkt worden op basis van componenten, wat betekent dat er voor elk onderdeel een nieuw component gemaakt zal worden. Een van de voordelen van Angular componenten is dat elke component een eigen verantwoordelijkheid heeft en dat wijzigingen in het ene component mindere impact hebben op de rest.
Het ophalen van de data uit de backend zal gaan via Angular Services. Hierdoor ontstaat er een mooie scheiding tussen de view en data.
Angular Material zal gebruikt worden voor andere componenten zoals tabellen, buttons en input fields. Ook functionaliteiten zoals portals en het draggen en droppen van elementen zullen gebruikt worden.

Pdf-export

Samen met Topicus is er besloten om voor dit project als requirement mee te nemen dat de uiteindelijk oplossing gebruikt gaat maken van server-side pdf-export voor het omzetten van HTML-bestanden naar PDF-documenten. De reden hiervoor is dat er in andere ParnasSys applicaties ook gebruik wordt gemaakt van deze techniek. Bovendien is het belangrijk dat de PDF-bestanden van goede kwaliteit zullen zijn en dat het niet te veel tijd zal kosten.
Aangezien de opdracht bestaat uit het bouwen van een prototype voor de rapport editor in Angular valt het exporteren naar PDF via de server buiten de scope van de opdracht. In plaats van dat ik er zelf voor zorg dat de PDF-bestanden gegenereerd worden zal ik mijn taak zijn om in de frontend de juiste HTML-bestanden te prepareren zodat deze opgestuurd kunnen worden naar de backend en daar ongezet kunnen worden naar PDF door middel van Wkhtmltopdf. Deze HTML-bestanden zullen de juiste CSS moeten bevatten zodat de opmaak van de geëxporteerde rapporten overeen komt met het origineel.

Embedden naar andere applicaties

IFrames zijn geschikt voor het embedden van bijvoorbeeld websites of webapplicaties in een andere applicatie. Het Digitale Rapport zal een Angular component worden, welke dynamisch opgesteld zal worden. Hier biedt iFrame echter geen ondersteuning voor.

Voor het embedden van het rapport naar andere applicaties zal gebruik worden gemaakt van de Angular Elements library. Voor zover te vinden is, zijn web-components in combinatie met deze library de enige geschikte oplossing. De voornaamste nadelen zitten hem in het embedden van grotere applicaties in tegenstelling tot enkele componenten. Dit is gelukkig niet van belang voor deze opdracht, aangezien het Digitale Rapport waarschijnlijk uit één enkele component zal bestaan.

Figuur 19: Logo's van de gekozen technieken

Grafieken

Voor het weergeven van grafieken zal er gebruik worden gemaakt van ApexCharts.js. Deze library kwam als beste naar voren tijdens het vergelijken op basis van een aantal criteria. ApexCharts.js biedt ondersteuning aan een groot aantal verschillenden grafieken en diagrammen en heeft genoeg aanpasopties om de gebruiker tevreden te stellen.
Op basis van criteria komt ApexCharts.js als beste uit de 5 gekozen libraries, gevolgd door Plotly.js en Chart.js. Het verschil qua scores tussen deze libraries is nihil en daarom is het lastig om een keuze te maken tussen deze libraries.
Charts.js is ver uit de meest populaire library. Zo is deze het vaakst gedownload en wordt het beste onderhouden. Het nadeel is echter de render technologie. Als enige maakt deze library gebruik van een technologie op basis van HTML5 Canvas, terwijl, qua performance, de voorkeur uit gaat naar SVG.
Apexcharts.js en Plotly.js lopen eigenlijk vrij gelijk op. Op vrijwel alle criteria komen de scores ongeveer gelijk uit. Dit maakt het lastig om een keuze te maken tussen deze libraries. Om toch tot een conclusie te kunnen geraken is er nog gekeken naar de beschikbare documentatie van beide libraries.

De documentatie van ApexCharts.js is erg uitgebreid. Zo bevat deze een installatie handleiding, een beschrijving van alle beschikbare soorten diagrammen en alle mogelijke aanpassingsopties.
Ook is er een documentatie die laat zien hoe de integratie werkt met verschillende frameworks (Angular, React en Vue).
Voor elke diagram is er een demo beschikbaar. Dit maakt het gelijk duidelijk hoe de desbetreffende diagram te implementeren is.

De documentatie van Plotly.js is minder uitgebreid en duidelijk. Hij bevat dan wel een installatie handleiding, maar deze is niet speciaal op maat gemaakt voor Angular. Verder bevat de documentatie wel een beschrijving van de beschikbare diagrammen en aanpassingsopties.

Over het algemeen oogt de documentatie van ApexCharts.js overzichtelijker en aantrekkelijker dan die van Plotly.js. Op basis hiervan en de criteria zal er gekozen worden voor ApexCharts.js als library voor grafieken en diagrammen.

	ApexCharts.js	Billboard.js	Chartist.js	Chart.js	Plotly.js
Downloads	4	1	3	5	3
Render technologie	5	5	5	1	5
Verrijking en onderhoud	3	4	2	5	3
Aangeleverde diagrammen	5	5	3	5	5
Ontwerp- en aanpassingsopties	5	3	2	4	5
Totaal	**22**	**18**	**15**	**20**	**21**

Tabel 12: Resultaten grafiek onderzoek

8. ONTWERP

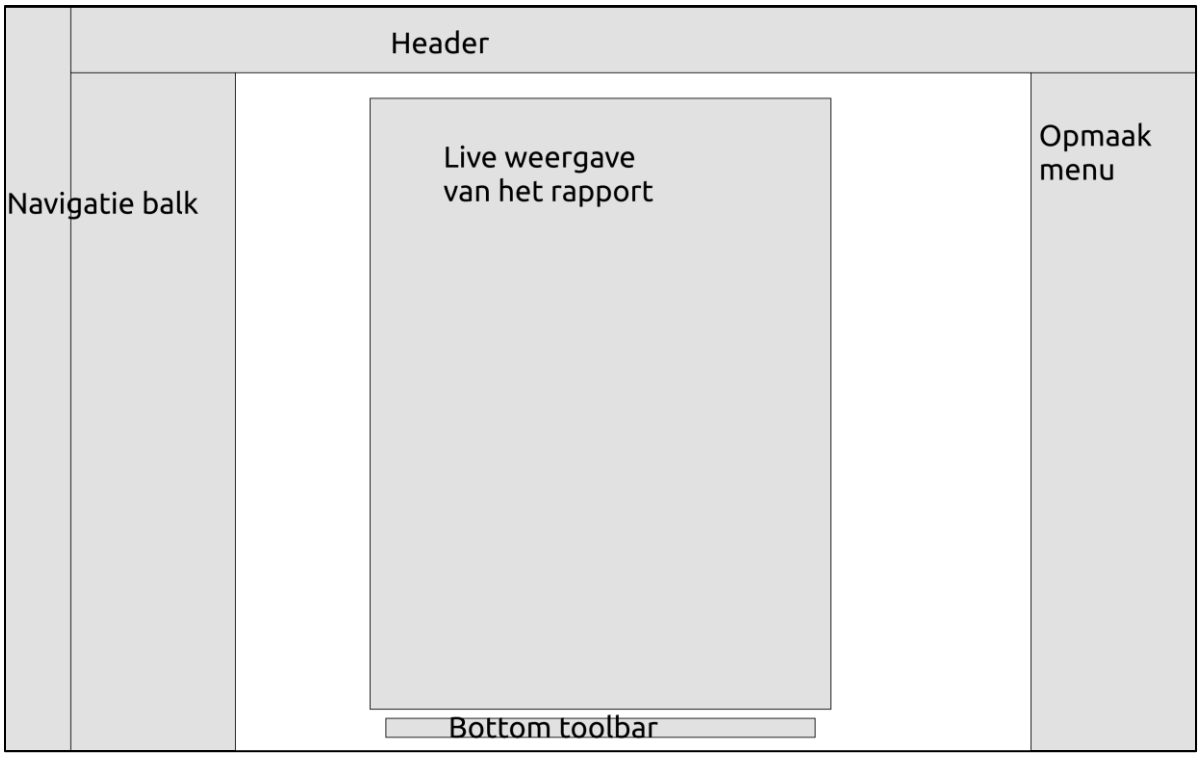
In dit hoofdstuk is het ontwerp beschreven dat gemaakt is voor de rapport editor. De gemaakte keuzen komen hierbij ook aan bod. Alle wireframes en schermontwerpen zijn te vinden in Bijlage G.

8.1 SCHERMONTWERPEN

Nadat de onderzoeksfase afgerond was kon er begonnen worden aan het ontwerpen van de applicatie. Door middel van Figma zijn er wireframes en high-fidelity ontworpen, gebruikmakend van dezelfde stijl die de huidige ParnasSys applicatie heeft.
Er is gekozen om dezelfde styling aan te houden als de ParnasSys applicatie. Dit was niet zozeer een eis maar aangezien de nieuwe rapport editor onderdeel zal uitmaken van ParnasSys leek het me wel zo logisch om dit consistent te houden. Na het ontwerp te hebben laten zien aan de bedrijfsbegeleider vond hij het ook de juiste keuze.

Voordat de schermontwerpen uitgewerkt werden was het handig om eerst een schermindeling te bedenken met de benodigde onderdelen. Het idee was om de applicatie te laten bestaan uit één enkele scherm aangezien een gebruiker zich alleen maar zal bezig houden met het bewerken van een rapport. Het hebben van meerdere schermen is dus onnodig. Het scherm bevat de volgende onderdelen:

- Navigatiebalk
- Header
- Live weergave van het rapport
- Bottom-toolbar
- Opmaakmenu



Figuur 20: Indeling van het applicatiescherm

Een van de eisen was om een live weergave van het rapport te laten zien. Bij het bewerken hiervan zal deze dus ook constant te zien zijn. Om dan toch meerdere handelingen te kunnen uitvoeren, zoals het invoegen van onderdelen, is er voor gekozen om een side-bar te maken.

In elk van deze onderdelen kan een leerkracht bepaalde handelingen uitvoeren. In onderstaande hoofdstukken staan de verschillende onderdelen beschreven, met daarbij de bijbehorende schermontwerpen en functionaliteiten. In de bijlagen zijn de vergrootte versies van alle ontwerpen te vinden.

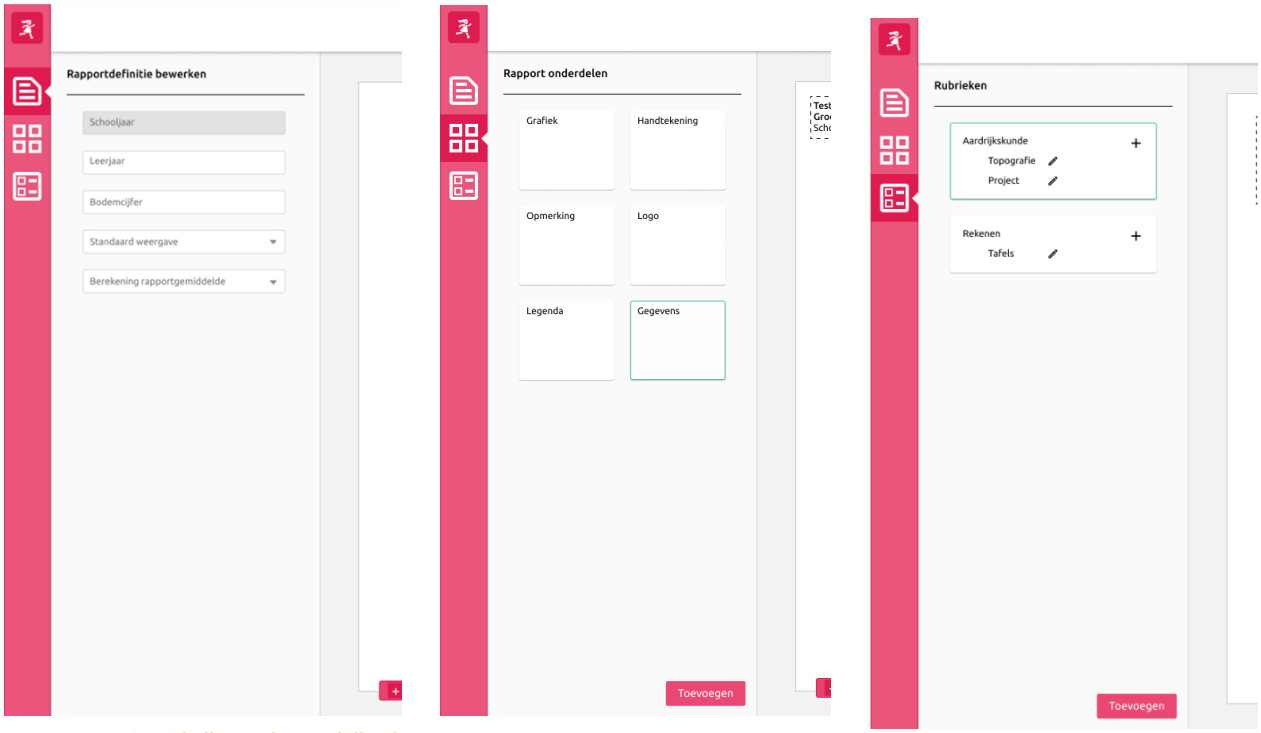
Het uiteindelijke ontwerp is tot stand gekomen door inspiratie te halen uit andere document-editors zoals Microsoft Word. De gekozen indeling is dan ook deels overgenomen hieruit. Een overzicht van de afgedekte user stories per schermonderdeel is te vinden in het functionele ontwerp en in Bijlage L.

NAVIGATIEBALK

Via de navigatiebalk aan de zijkant van het scherm kan de leerkracht wisselen tussen verschillende content. Door middel van het klikken op één van de tabs zal er een ander menu weergegeven worden waar vervolgens bepaalde handelingen uitgevoerd kunnen worden.

- Invullen van de rapportdefinitie.
- Selecteren van rapportonderdelen.
- Beheren van rubrieken.

Een van de gemaakte keuzes was het gebruik van een navigatiebalk aan de zijkant van het scherm. Door middel van tabjes kan de gebruiker wisselen tussen de content. Doordat de functionaliteiten onderverdeeld zijn over verschillende tabjes, zal het voor een gebruiker overzichtelijker zijn waar die mee bezig is. De volgorde van de opties op de navigatiebalk is bewust gekozen. Bij het aanmaken van een nieuw rapport moeten een leerkracht eerst de rapportdefinitie invullen, alvorens er ontworpen kan worden.



Figuur 21: Navigatiebalk met de verschillende content

HET INVULLEN VAN DE RAPPORTDEFINITIE

Voordat er een nieuw rapport ontworpen kan worden, zal de rapportdefinitie ingevuld moeten worden. De rapportdefinitie bestaat uit een aantal standaard gegevens over het rapport zoals het schooljaar, leerjaar, bodemcijfer, standaardweergave van resultaten en de manier waarop het gemiddelde berekend zal worden.

Om deze definitie te kunnen invoeren zal de leerkracht naar het juiste kopje op de navigatiebalk moeten navigeren. Vervolgens komt er een menu tevoorschijn welke bestaat uit een aantal invoervelden. Pas wanneer elk van deze velden ingevuld zijn kan de leerkracht beginnen met het ontwerpen van een rapport lay-out. Een aantal van deze definitie waardes komen namelijk terug bij het ontwerpen. Een voorbeeld hiervan is de standaardweergave van de schoolresultaten. Bij het invoegen van een rubriek of legenda wordt er rekening gehouden met deze weergave en het is dus handig als deze al ingevuld is.

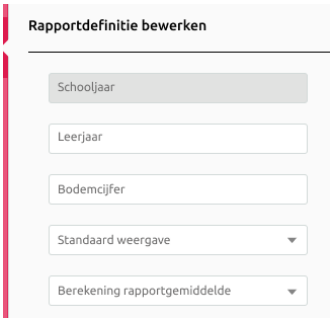
HET INVOEGEN VAN RAPPORT ONDERDELEN

Wanneer een leerkracht begint met het ontwerpen van een rapport, is het rapport nog helemaal leeg. Het toevoegen van onderdelen aan het rapport kan gedaan worden via het rapportonderdelen-menu, welke geopend wordt via de navigatiebalk. De leerkracht kan één van de onderdelen selecteren en op de 'toevoegen' knop drukken. Vervolgens verschijnt dit onderdeel op de rapport pagina. Het rapport kan de volgende onderdelen bevatten:

- Grafiek
- Handtekening vak
- Opmerkingenveld
- Logo
- Legenda
- Rapportdefinitie

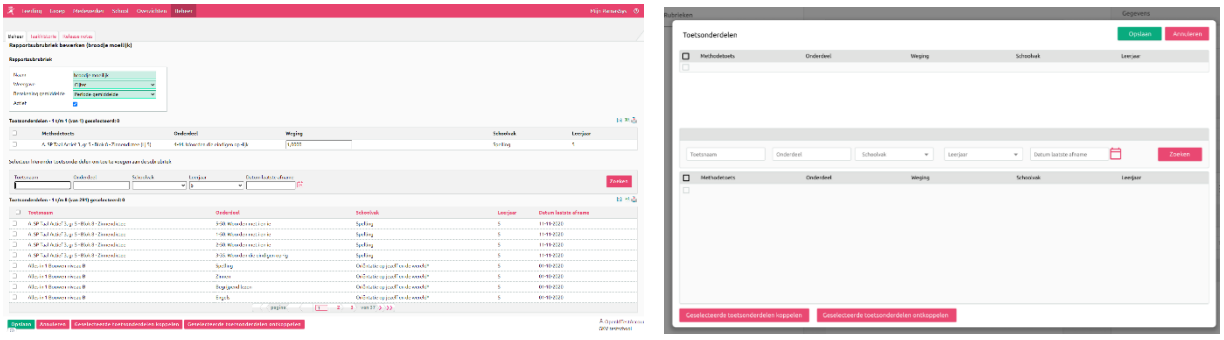
HET BEHEREN VAN RUBRIEKEN

Aan een rapport kunnen rubrieken toegevoegd worden. Een leerkracht kan in het rubrieken-menu een nieuwe rubriek toevoegen door op de 'toevoegen' knop te klikken. Vervolgens kunnen er extra subrubrieken toegevoegd worden aan de rubrieken.
Van elke subrubriek wordt het gemiddelde resultaat weergeven op het rapport. Hoe de weergave hiervan eruit ziet wordt bepaald door de leerkracht bij het invoeren van de rapportdefinitie.



Figuur 22: Het invullen van de definitie

Voor het koppelen van toetsonderdelen aan een subrubriek, is er gekozen om dit door middel van een pop-up te doen. Met een pop-up wordt ergens de focus op gelegd; de achtergrond wordt vaak donkerder, waardoor hetgeen op de pop-up meer opvalt. Binnen de huidige applicatie wordt dit gedaan door middel van een nieuwe pagina waarin de gebruiker een aantal toetsonderdelen kan selecteren uit een tabel. Het leek mij het beste om deze functionaliteit zo te laten aangezien het gewoon prima werkt zo. Het verschil bij het nieuwe ontwerp is dat deze functionaliteit in een pop-up plaatsvindt in plaats van een nieuwe pagina. Aangezien dit een vrij grote functionaliteit betreft waar 2 tabellen aan te pas komen was het ook geen optie om dit in een side-bar weer te geven.



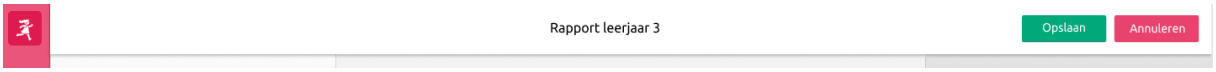
Figuur 23: Vergelijking selecteren van toetsonderdelen tussen oude en nieuwe ontwerp

HEADER

De header weergeeft de titel van het rapport. Zo is het duidelijk voor de leerkracht voor welke leerjaar hij een rapport aan het bewerken is. Verder heeft de header twee knoppen; voor het opslaan en het annuleren van de wijzigingen aan een rapport.

WIJZIGINGEN OPSLAAN EN ANNULEREN

Wanneer een leerkracht een nieuw rapport heeft ontworpen of een bestaand rapport heeft bewerkt, kan hij ervoor kiezen om de wijzigingen op te slaan, of om deze te annuleren. Dit kan gedaan worden door op de groene of roze knop in de header te klikken. Vervolgens wordt de applicatie afgesloten.



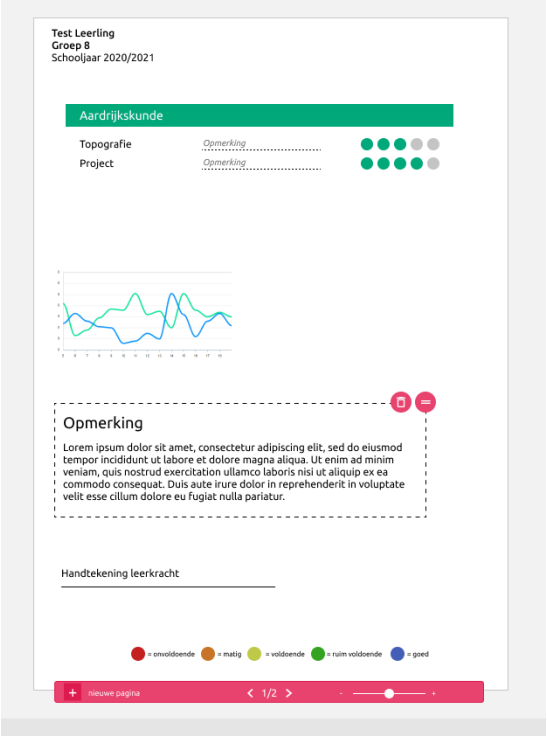
Figuur 24: Header

LIVE WEERGAVE VAN HET RAPPORT

De live weergave van het rapport bevindt zich in het midden van het scherm. Dit is een a4 pagina welke te vergelijken is met Word. De toegevoegde rapportonderdelen zullen verschijnen op deze pagina, waarna een leerkracht deze kan verplaatsen naar een gewenste plek.

RAPPORTONDERDELEN VERPLAATSEN EN VERWIJDEREN

De positie van onderdelen op het rapport kunnen verplaatst worden over de gehele pagina. De leerkracht kan dit doen door te klikken op dit onderdeel om deze vervolgens te verslepen naar een andere positie.
Bij het selecteren van één van de onderdelen, verschijnt er een ‘prullenbak’ -icoontje. Door hier op te klikken zal het onderdeel verwijderd worden van het rapport.



Figuur 25: Live weergave van het rapport

BOTTOMBAR

Onder in het scherm bevindt zich een soort bottom-toolbar. Het doel van dit component is het aanpassen van de weergave van het rapport. Er kunnen extra pagina's toegevoegd worden aan een rapport, gewisseld worden tussen de pagina's en er kan in- of uitgezoomd worden.

PAGINA TOEVOEGEN AAN RAPPORT

Een leerkracht kan ervoor kiezen om een rapport meerdere pagina's groot te maken. Dit kan gemakkelijk gedaan worden door op het 'plusje' te drukken in de bottom-toolbar. Vervolgens er gewisseld worden tussen de verschillende pagina's door middel van de pijltjes.

ZOOMEN

Om het gebruiksgemak te vergroten kan een leerkracht de live rapport weergave in- en uitzoomen. Dit kan gedaan worden door middel van de zoom-slider.

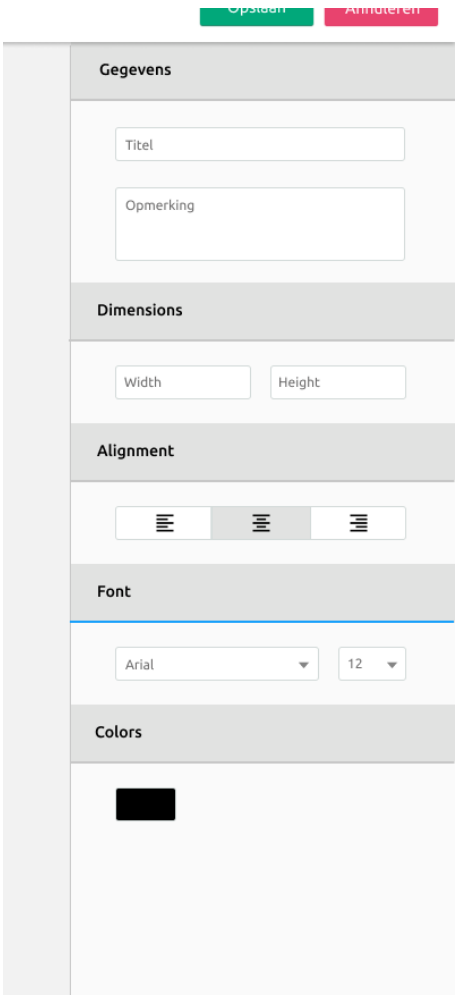
Figuur 26: BottomBar

OPMAAKMENU

Het stylingmenu aan de rechterkant van het menu zorgt ervoor dat de opmaak van het rapport en de verschillende rapportonderdelen aangepast kunnen worden. Dit menu bestaat uit verschillende invoervelden welke verschillen op basis van het geselecteerde onderdeel. Wanneer een leerkracht één van deze velden invult zullen de wijzigingen gelijk zichtbaar worden op de live weergave van het rapport.

OPMAAK VAN RAPPORTONDERDELEN AANPASSEN

Een leerkracht heeft de mogelijkheid om de opmaak van de rapportpagina en bijbehorende onderdelen aan te passen. Wanneer er één van de rapportonderdelen geselecteerd wordt, zullen er in het stylingmenu de juiste invoervelden verschijnen. In deze velden kan de leerkracht bepaalde waardes aanpassen om zo de opmaak van het geselecteerde onderdeel te veranderen.



Figuur 27: Ontwerp van het opmaakmenu

8.2 GRAFISCH ONTWERP

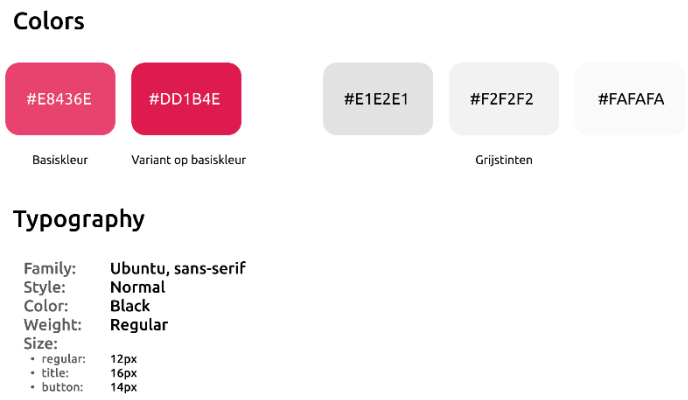
In dit hoofdstuk worden de grafische kenmerken van het design besproken.

KLEUREN

De gebruikte kleuren in de designs zijn overgenomen van de ParnasSys applicatie. De reden hiervoor is dat er zo consistentie is tussen beide applicaties en het voor de gebruiker voelt als dezelfde applicatie, aangezien de rapport editor hier onderdeel van zal zijn.
Het kleurmodel dat hier gebruikt wordt is een monochromatisch model. Monochromatische kleurmodellen zijn afgeleid van een enkele basiskleur en uitgebreid met tinten hiervan. De rest wordt aangevuld met wit en grijs. Het gebruik van een monochromatische kleur zorgt voor een sterk gevoel van visuele samenhang. De relatieve afwezigheid van tintcontrast kan worden gecompenseerd door variaties in toon en de toevoeging van textuur. In het geval van dit ontwerp is er ook een enkele basiskleur met daarbij een secondaire variant in een andere tint. Voor de rest wordt er gebruik gemaakt van verschillende tinten grijs.

TYPOGRAFIE

Het font dat gebruikt is in de designs is overgenomen van de ParnasSys applicatie. De Ubuntu Font Family is het standaardlettertype voor de huidige ontwikkelingsversies van het Ubuntu-besturingssysteem. Deze Font Family is opgenomen in de Google Fonts-directory waardoor deze gemakkelijk beschikbaar is voor web typografie. Onderstaande styling-card weergeeft de verschillende kleuren die zijn gebruikt en het gebruikte font, met bijbehorende toepassingen.



Figuur 28: Stylingcard van de gemaakte designs

GESTALT PRINCIPLES

Bij het maken van de schermontwerpen is er rekening gehouden met de Gestalt Principles. Gestalt Principles zijn wetten van menselijke waarneming die beschrijven hoe mensen vergelijkbare elementen groeperen, patronen herkennen en complexe beelden vereenvoudigen wanneer we objecten waarnemen.

Principle of Similarity

De principle of similarity houdt in dat wanneer dingen op elkaar lijken, de hersenen ze samen groeperen en laat denken dat ze dezelfde functie hebben. In het ontwerp is dit onder andere terug te zien in de navigatiebalk. De tabjes op de navigatiebalk zien er vrijwel allemaal hetzelfde uit. Ze hebben dezelfde kleur, formaat en dezelfde soort icoontjes. Hierdoor zal de gebruiker door hebben dat ze allemaal dezelfde functie hebben.

Principle of Common region

De principle of common region is sterk gerelateerd aan nabijheid. Het stelt dat wanneer objecten zich in hetzelfde gesloten gebied bevinden, de hersenen ze als groepen waarnemen. In het ontwerp is dit terug te zien bij de navigatiebalk en het opmaakmenu. Beiden hebben een border en een andere achtergrondkleur waardoor het duidelijk is dat deze componenten los staan van het omliggende gebied.

Principle of Figure ground

De principle of figure ground stelt dat mensen instinctief objecten waarnemen als of op de voorgrond of op de achtergrond. In de bottom-toolbar is dit terug te zien doordat er een box-shadow is toegevoegd. Hierdoor treedt dit component meer naar de voorgrond dan de rest van de componenten.

9. REALISATIE

In dit hoofdstuk wordt alles beschreven met betrekking tot de realisatie. De volgende onderdelen komen hierbij aan bod:

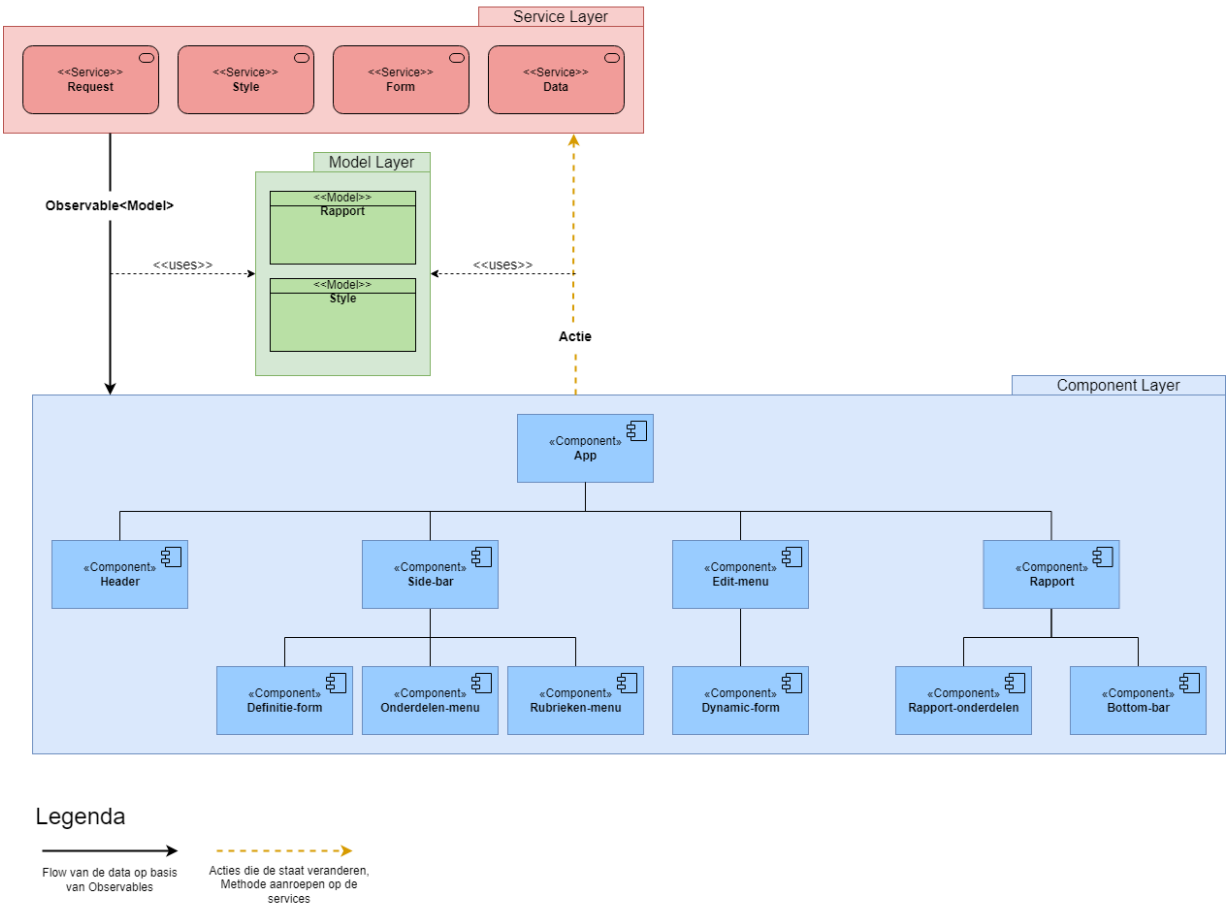
- De applicatie structuur
- Talen en libraries
- Het realisatieproces
- Gebruikte technieken

9.1 APPLICATIE STRUCTUUR

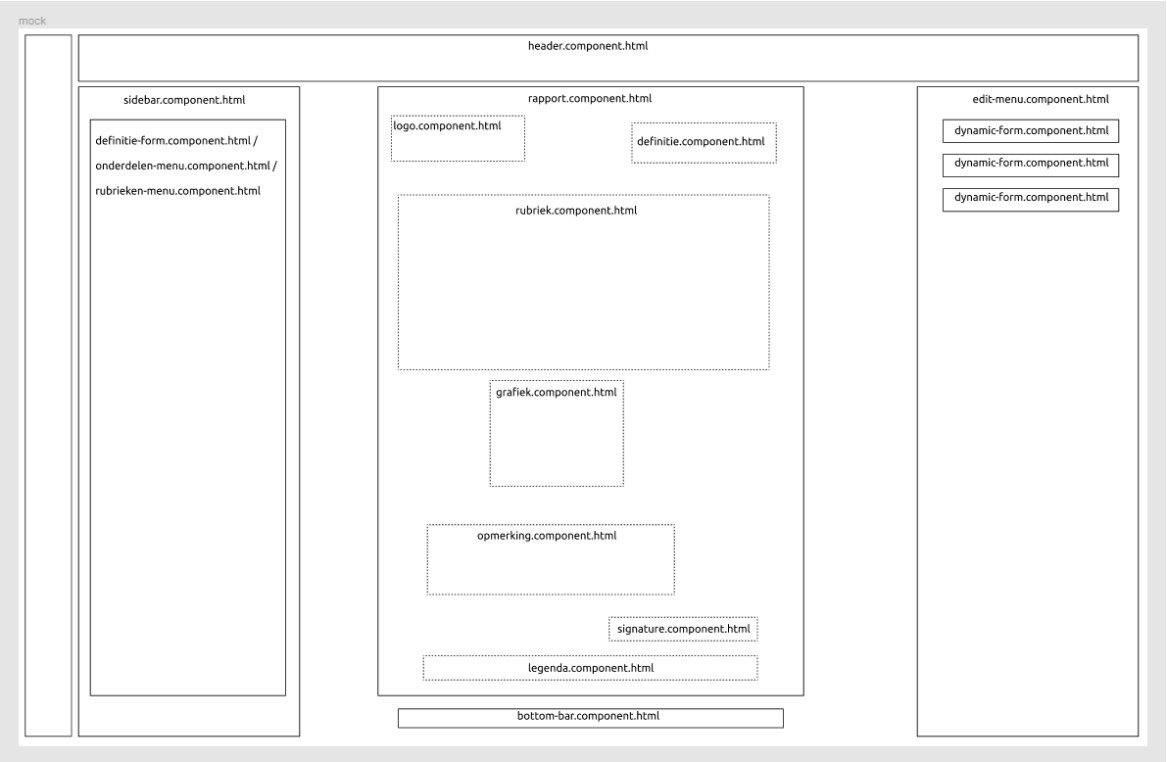
De structuur van de applicatie is vrij eenvoudig en duidelijk. Het bestaat uit Componenten, Entiteiten en Services. De componenten zijn de verschillende onderdelen die te zien zijn op het scherm. De data halen ze uit de Services

In onderstaande architectuur diagram is de interactie tussen de Componenten en Services weergegeven. In de Component Layer is de hiërarchie tussen de verschillende Componenten beschreven. Zoals te zien is vallen alle componenten onder één parent-component, aangezien de applicatie ook uit een enkel scherm bestaat. Vervolgens zijn er vier componenten welke de gehele indeling van het scherm innemen, met daaronder nog wat sub-componenten. De volledige schermindeling qua componenten is te zien in figuur 29.

Onderstaande diagram weergeeft ook de dataflow tussen de componenten en services. Data stroomt van de services naar de componenten door middel van Observables. Componenten kunnen een abonnement hebben op bepaalde service-onderdelen. De opgehaalde data wordt gecodeerd met behulp van de modelobjecten. De acties die de staat wijzigen, worden geïmplementeerd als methoden in de services.



Figuur 29: Architecture diagram



Figuur 30: Schematische weergave van de componenten op het scherm

COMPONENTEN

De applicatie heeft redelijk wat componenten. Elk component bestaat uit een aantal bestanden zoals een TypeScript, CSS en HTML bestand.

Componenten	Beschrijving
Base	Een basis component met een aantal variabelen en methoden die relevant zijn voor meerdere componenten binnen de applicatie. De meeste componenten extenden vanuit deze component en daarom is deze niet te zien op de schematische weergave.
BottomBar	De bottom-toolbar onder in het scherm waarmee er acties uitgevoerd kunnen worden op een rapport.
DynamicForm	Verschillende soorten invoervelden die gebruikt kunnen worden in een formulier.
EditMenu	Menu aan de rechterzijde van het scherm waarin de styling van onderdelen aangepast kunnen worden.
Header	De header boven in het scherm.
Rapport	De liveweergave van het rapport waar onderdelen op geplaatst en verslepen kunnen worden.
Definitie	Rapportonderdeel welke standaardgegevens weergeeft (schooljaar, leerling naam etc.)
Grafiek	Rapportonderdeel welke resultaten weergeeft in een grafiek.
Legenda	Rapportonderdeel welke de cijferschaal weergeeft.
Logo	Rapportonderdeel welke de logo van een school weergeeft.
Opmerking	Rapportonderdeel waarin een leerkracht een opmerking kan plaatsen.
Rubriek	Rapportonderdeel waarin de resultaten van bepaalde vakken weergegeven worden.
Signature	Rapportonderdeel waarin gebruikers een handtekening kunnen zetten.
Sidebar	Menu aan de linkerzijde van het scherm waarin verschillende acties uitgevoerd kunnen worden.
ToetsOnderdeel	Pop-up waarin toetsonderdelen aan een subrubriek gekoppeld kunnen worden.

Tabel 13: Componenten

SERVICES

De services binnen de applicatie worden gebruikt voor het ophalen van data en voor de communicatie tussen bepaalde componenten.

Services	Beschrijving
FormService	Service die gebruikt wordt om form-layouts op te halen
StyleService	Service die gebruikt wordt als communicatiepunt tussen verschillende componenten om zo styling-data door te geven.
DataService	Service die gebruikt wordt voor het ophalen van pagina gerelateerde data.
RequestService	Service die gebruikt wordt om rapportdata op te halen van de database.

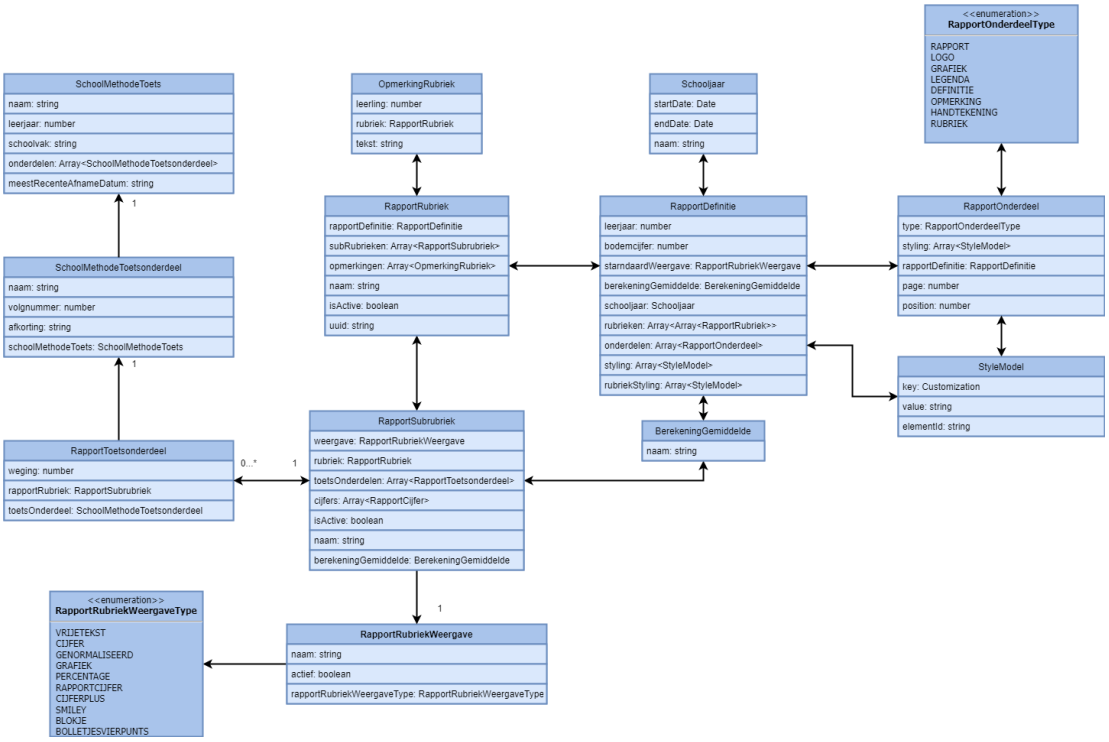
Tabel 14: Services

ENTITEITEN

De entiteiten in de applicatie komen grotendeels overeen met de entiteiten uit ParnasSys en worden gebruikt voor het ophalen en opslaan van data uit de database.

Entiteiten	Beschrijving
RapportDefinitie	Alle gegevens van een rapport.
RapportRubriek	De rubrieken van een rapport.
RapportSubrubriek	De subrubrieken die bij een rubriek horen.
RapportToetsonderdeel	De toetsenonderdelen die gekoppeld kunnen worden aan een subrubriek.
RapportCijfer	Het cijfer van een rapport.
Schooljaar	Het jaar waarin een leerling zit.
SchoolMethodeToets	Een toetsen die bij een toetsonderdeel horen.
OpmerkingRubriek	De opmerkingen die bij een rapport horen.
RapportRubriekWeergave	De manier waarop resultaten weergegeven worden.
BerekeningGemiddelde	De manier waarop het gemiddelde resultaat berekend wordt.

Tabel 15: Entiteiten



Figuur 31: Class diagram

Bovenstaande Class diagram weergeeft de verschillende entiteiten die de applicatie heeft. Deze entiteiten zijn grotendeels overgenomen uit de huidige ParnasSys applicatie aangezien deze applicaties uiteindelijk ook dezelfde data gaan gebruiken. Echter zijn er wel een aantal nieuwe entiteiten toegevoegd, welke noodzakelijk zijn om een rapportdefinitie op te kunnen slaan.

De meest belangrijke class is de RapportDefinitie. Deze bevat namelijk alle configuratiemogelijkheden die de opmaak van een rapport bepalen. Een definitie heeft een array met de verschillende styling opties, die bij het inladen van een rapport toegepast kunnen worden. Ook heeft een definitie een lijst met rapportOnderdelen. Een rapportonderdeel is een element dat op een rapportpagina geplaatst kan worden (grafiek, logo etc.).
Een rapportdefinitie kan meerdere rubrieken hebben. Onder deze rubrieken vallen weer meerdere subrubrieken. Aan een subrubriek kunnen toetsonderdelen gekoppeld worden, van welke de resultaten getoond zullen worden op het rapport. Elk rapportOnderdeel heeft zijn eigen styling array en een verwijzing naar de bijbehorende definitie.

Een rapport kan meerdere rubrieken hebben. Onder deze rubrieken vallen weer meerdere subrubrieken. Aan een subrubriek kunnen toets onderdelen gekoppeld worden, van welke de resultaten getoond zullen worden op het rapport. De leerkracht kan uit verschillende manieren kiezen om de resultaten te tonen (cijfers, smileys, bolletjes, percentages, etc.).
Zowel aan een rubriek als aan een rapport kan een opmerking toegevoegd worden. Deze kan de leerkracht zelf invoeren per leerling. Een vergrote versie van dit diagram is te vinden in Bijlage I.

9.2 TALEN EN LIBRARIES

De applicatie is gemaakt in het framework **Angular 12**, aangezien dit de nieuwste versie was op het moment van implementatie. **Angular Material** is gebruikt voor een aantal componenten zoals het tabellen, invoervelden en het draggen & droppen van onderdelen. Als taal is er gebruik gemaakt van **TypeScript**. Voor het weergeven van grafieken op een rapport is de library **ApexCharts.js** gebruikt.
Het testen van de front-end code is gedaan door middel van het framework **Karma**.

Taal/ framework	Versie
Angular	12
Angular Material	12
TypeScript	4.3.5
Karma	6.3.0
ApexCharts.js	3.28

Tabel 16: Gebruikte talen en frameworks

9.3 HET PROCES

Tijdens de realisatiefase is er gewerkt volgens de Scrummethode. Scrum is een agile werkwijze dat gebruikt kan worden om op een effectieve manier software te ontwikkelen. Er is gekozen om te werken volgens deze werkwijze omdat er al veel ervaring vanuit de opleiding mee was. Ook is deze methode uitermate geschikt voor het structureren van het ontwikkelproces, doordat de taken van tevoren al worden ingedeeld. Doordat Scrum normaal uitgevoerd wordt in teamverband, was het niet mogelijk om de gehele methode te volgen. Vandaar dat er gekozen is om bepaalde delen van Scrum te gebruiken:

- **Sprints:** korte periodes van een aantal weken waarin een deel van het werk verricht wordt.
- **Backlogs:** geprioriteerde lijsten met alle resterende werkzaamheden.
- **Scrumboard:** alle backlog items komen hierop te staan verdeeld over verschillende statussen.
- **Stand up:** dagelijkse moment waarbij de voortgang en eventuele problemen besproken worden.

Volgens de planning zou er gewerkt worden via sprints die elk 2 weken duren. Aan het begin van elke sprint werden de taken ingedeeld die er gedurende die periode opgepakt zouden worden. Hieronder is er per sprint beschreven wat er is uitgevoerd en wat de resultaten hiervan waren:

SPRINT 1

De eerste sprint begon met het vullen van de backlog, het toewijzen van punten aan de user stories en het afmaken van het design. Vanuit Topicus stond er al een GitHub repository klaar die gebruikt kon worden en dus hoefde die niet meer aangemaakt te worden.
Deze sprint zou voornamelijk bestaan uit het opzetten van de applicatie en het ontwikkelen van de eerste componenten zoals een header en een sidebar. De volgende functionaliteiten zijn er uiteindelijk gerealiseerd gedurende deze sprint:

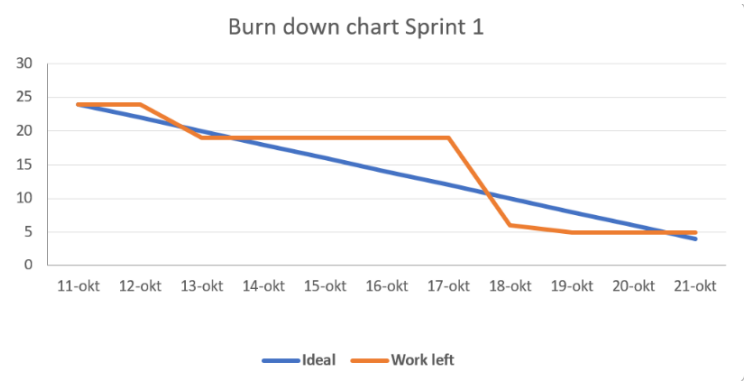
- Een header component.
- Een side-bar component.
- Een live weergave van het rapport.
- Een bottom-toolbar component.
- Een menu om de opmaak van componenten aan te passen.
- Het toevoegen van pagina's aan een rapport.

Het implementeren van bovenstaande functionaliteiten ging voornamelijk vrij voorspoedig. Doordat er van tevoren een ontwerp opgesteld was kon dit eigenlijk 1 op 1 overgenomen worden. Voor de functionaliteit voor het invoegen van extra pagina's aan een rapport was het nog wel eventjes zoeken naar een juiste methode. Er moest namelijk gewisseld kunnen worden tussen meerdere pagina's welke elk hun eigen content zouden laten zien. Uiteindelijk is er bedacht om een Array bij te houden met het aantal pagina's waarvan er voor elke pagina een DIV-element gegenereerd wordt met een eigen elementId op basis van de index. Aan de hand van een currentPage value wordt er dan besloten of een pagina DIV wel of niet zichtbaar moet zijn.

```
<div *ngFor="let page of pages; index as i">
    <div [hidden]="(currentPage-1)!==i" class="rapport-page" id="{{'rapport-page-'+i}}" (click)="onSelect($event, Onderdelen.rapport, this.styling)">
        <div class="inner-page" id="{{'inner-page-'+i}}">
            <p>pagina {{i + 1}}</p>
            <app-rubriek></app-rubriek>
        </div>
    </div>
</div>
```

Figuur 32: Code voorbeeld van rapport html

Uiteindelijk is het niet gelukt om de functionaliteit voor het invullen van een rapportdefinitie op te pakken gedurende deze sprint en deze moest doorgeschoven worden naar de volgende sprint. De reden van deze vertraging is dat er veel tijd ging zitten in het testen van de code. Aangezien er getest werd volgens het Test Driven Development principe, werden de testen van tevoren opgesteld. Er was nog geen ervaring met deze testmethode en daarom kostte het redelijk wat tijd om hiermee te beginnen.



Figuur 33: Burn down chart sprint 1

SPRINT 2

Tijdens deze sprint is er begonnen met het afronden van de functionaliteiten die de vorige sprint niet zijn afgekomen. Verder stond deze sprint voornamelijk in het teken van het kunnen invoegen van onderdelen aan een rapport. Zo is er een menu gemaakt waaruit een gebruiker onderdelen kan selecteren die vervolgens op een rapport geplaatst kunnen worden. Ook is er bezig geweest met de functionaliteit om de verschillende rapportonderdelen te kunnen verplaatsen over de pagina. Uit het eerder uitgevoerde onderzoek is er voortgekomen dat dit zou gaan door middel van Drag & drop van Angular Material, aangezien dit makkelijk is te implementeren en ook nog eens perfect geschikt is voor de situatie. Hier zijn dan ook geen problemen mee ondervonden.

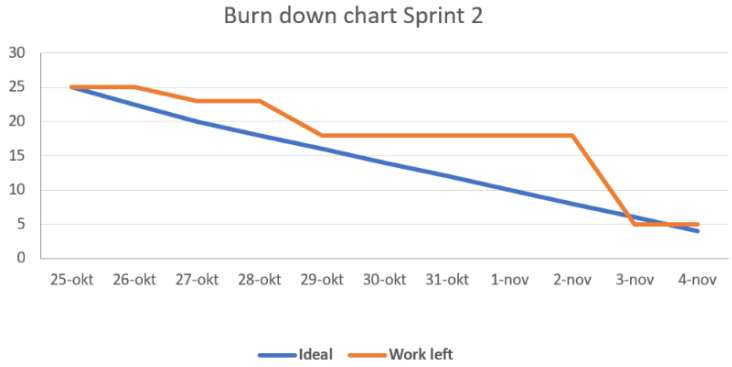
Aangezien alle verschillende rapportonderdelen een eigen styling kunnen hebben, moest er een manier bedacht worden om dit zo praktisch mogelijk op te stellen. Doordat dit een functionaliteit is die op meerdere plekken gebruikt zou worden moest er iets bedacht worden zodat er geen dubbele code zou ontstaan. Na overleg gehad te hebben met de bedrijfsbegeleider is er besloten om dit door middel van een Service te doen, aangezien dit de manier in Angular is om data te delen onder meerdere componenten. In het volgende hoofdstuk wordt het technische aspect beter beschreven.

Het werken volgens het Test Driven Development principe ging deze sprint minder goed. Omdat er voornamelijk complexere functionaliteiten geïmplementeerd werden waarvan het nog niet helemaal duidelijk was het gemaakt zou worden, was het lastig om hier van tevoren al testen voor te bedenken. Dit was natuurlijk geen ramp aangezien de testen gewoon achteraf opgesteld konden worden, alleen was dat niet volgens het TDD principe. Uiteindelijk is er wel gezorgd voor 100% code coverage zodat alsnog alle code getest was.

De volgende functionaliteiten zijn er geïmplementeerd deze sprint:

- Het invullen van een rapportdefinitie.
- Rapportonderdelen kunnen selecteren uit een menu.
- De positie van rapportonderdelen veranderen op een rapport.
- Een logo toevoegen aan een rapport.
- De opmaak van een logo aanpassen.

De totale backlog van deze sprint bestond uit 5 user stories (25 punten). Deze zijn allemaal binnen de sprint afgerond.



Figuur 34: Burn down chart sprint 2

SPRINT 3

In de derde sprint is er verdergegaan met het invoegen van rapportonderdelen aan een rapport. Zo moesten er nog een aantal componenten gebouwd worden die vervolgens aan een pagina gehangen konden worden. Dit gaat over de volgende onderdelen:

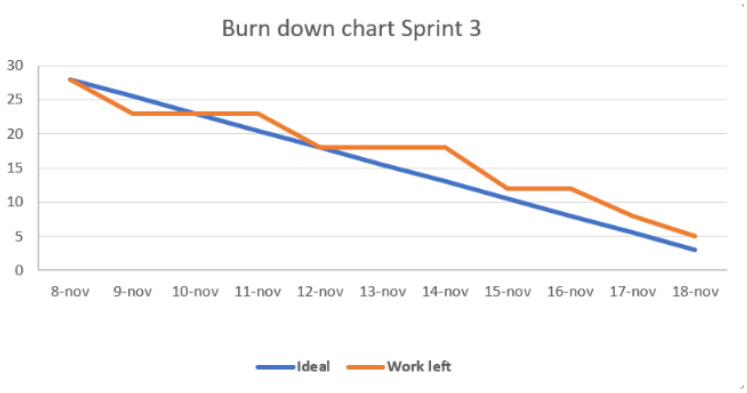
- Handtekeningregel
- Grafiek met resultaten
- Legenda
- Opmerkingenveld

Van elk van deze onderdelen moest de styling weer aangepast kunnen worden. Hier was in de vorige sprint al een bepaalde functionaliteit voor gebouwd welke ook toegepast kon worden op de nieuwe rapportonderdelen waardoor hier niet veel werk meer in ging zitten. Echter moesten er wel een aantal wijzigingen aangebracht worden zodat het zo modulair mogelijk was en elke rapportonderdeel dezelfde stylingfunctionaliteit kon gebruiken.

Voor de implementatie van het grafiek component werd er gebruik gemaakt van een library; ApexCharts.js. Uit het onderzoek is gebleken dat dit de beste optie was voor het genereren van grafieken. Na korte tijd kon er dan ook al een grafiek gemaakt worden, echter was er nog geen data om weer te geven. Het was lange tijd onduidelijk wat een grafiek nou eigenlijk moest laten zien en waar deze data vandaan moest komen. De bedrijfsbegeleider is hier ook niet uitgekomen en er is was uiteindelijk besloten om dit onderdeel maar even links te laten liggen.

Verder is er deze sprint nog gewerkt aan het verwijderen van rapportonderdelen van een rapport en is de feedback van de vorige sprint verwerkt. Deze feedback ging voornamelijk over de code kwaliteit en een aantal dingetjes die iets netter geschreven konden worden.

De totale backlog van deze sprint bestond uit 10 user stories (28 punten). Hiervan zijn er 23 behaald. De reden hiervoor is dat het nog niet duidelijk was welke data een grafiek zou moeten weergeven en deze daarom nog maar even over te slaan.

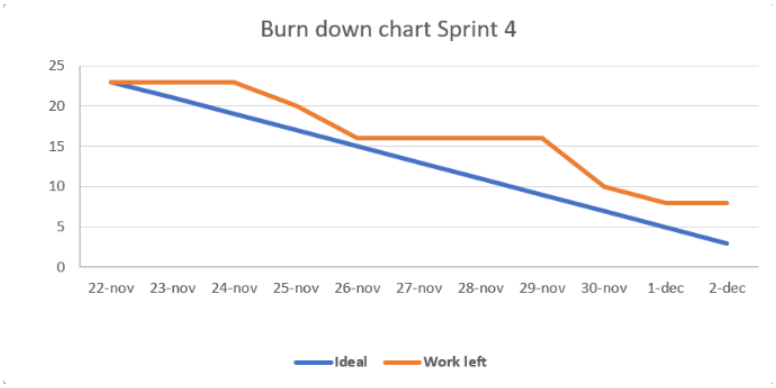


Figuur 35: Burn down chart sprint 3

SPRINT 4

Deze sprint stond in het teken van rubrieken en subrubrieken. Er is een menu gerealiseerd waarin gebruikers rubrieken kunnen beheren en toevoegen. Voor het koppelen van toetsonderdelen aan een rubriek is er een pop-up gemaakt. Ook moest de styling van een rubriek aangepast kunnen worden, maar deze functionaliteit kon grotendeels hergebruikt worden vanuit de vorige sprints.
Een ander belangrijk aspect waar deze sprint aandacht aan is besteed is het opslaan van een rapportdefinitie in de database. De huidige manier van het opslaan was niet meer relevant en er moest een nieuwe manier bedacht worden. Zo is er nagedacht over het type database en de manier waarop de data het beste opgeslagen zou kunnen worden. Door middel van Postman zijn er een aantal request uitgevoerd op de ParnasSys applicatie zodat er een beter beeld kwam van de opbouw van de data.

De totale backlog van deze sprint bestond uit 5 user stories (23 punten). Deze zijn allemaal binnen de sprint afgerond.

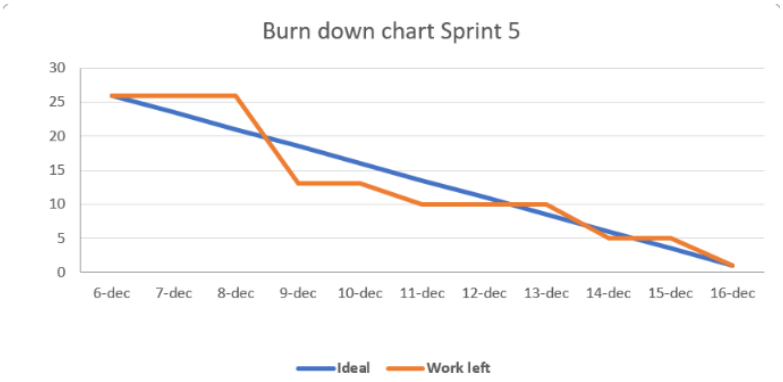


Figuur 36: Burn down chart sprint 4

SPRINT 5

Tijdens deze sprint is er voornamelijk gewerkt aan het inladen en opslaan van data. Er is een koppeling gemaakt met een Node.js backend en PostgreSQL database zodat er rapportdefinities opgeslagen en ingeladen kunnen worden. In de vorige sprint was er al nagedacht over de manier waarop dit zou gaan gebeuren.
Ook is er gedurende deze sprint gewerkt aan het embedden van het rapport in andere applicaties doormiddel van Webcomponents.

De totale backlog van deze sprint bestond uit 4 user stories (26 punten). Deze zijn allemaal binnen de sprint afgerond.



Figuur 37: Burn down chart sprint 5

9.4 GEBRUIKTE TECHNIEKEN

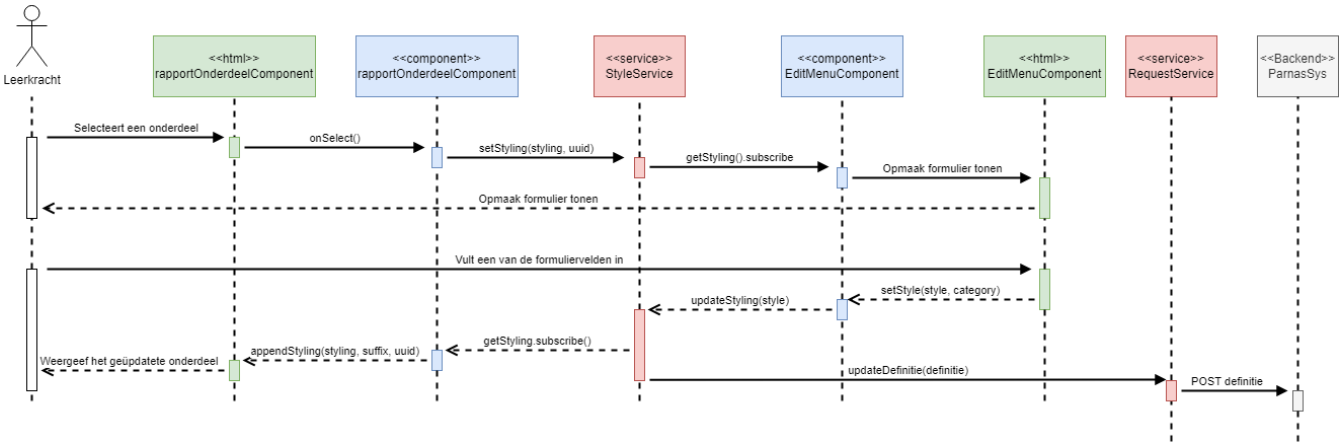
HET STYLEN VAN ONDERDELEN

Een gebruiker kan de opmaak van een rapportonderdeel aanpassen. Wanneer er op een van de onderdelen geklikt wordt, veranderden de invoervelden in het EditMenuComponent. Bij het invullen van een van de invoer-velden zal de opmaak van het onderdeel op de live-weergave veranderen.
Voor het veranderen van de opmaak van een rapportonderdeel wordt er gebruikt gemaakt van een StyleService.
Vrijwel alle componenten in de applicatie hebben een styling en uuid variabele waarmee de opmaak van het desbetreffende component aangepast kan worden. Dit gebeurt door middel van een StyleService.

Variabele	Type	Beschrijving
styling	Array<StyleModel>	Bevat de styling voor het desbetreffende component.
uuid	String	Elke component heeft een unieke uuid zodat deze van elkaar te onderscheiden zijn.

Tabel 17: Variabele beschrijving

De StyleService heeft dezelfde variabelen, dus bij het selecteren van een van de rapportonderdelen wordt de styling en uuid variabelen in de StyleService vervangen voor dezelfde variabelen in het geselecteerde onderdeel.



Figuur 38: Sequence diagram stylen van onderdelen

Wanneer een gebruiker een van de onderdelen op het rapport selecteert, wordt er een call gedaan naar de StyleService met als parameters de styling en uuid variabelen van desbetreffende onderdeel-component.
De StyleService slaat vervolgens de meegegeven styling en uuid op als de “huidige waardes”. Doordat de styling een Observable is, kunnen andere componenten hier op abonneren.
De EditMenuComponent heeft een abonnement op de StyleService en elke keer wanneer de styling verandert, wordt deze ook in de EditMenu aangepast zodat deze overeenkomt met de service (en dus de styling variabele bevat van het geselecteerde onderdeel-component). Vervolgens wordt er in de EditMenuComponent een dynamisch formulier opgesteld, op basis van de styling.
Een gebruiker kan een van de invoervelden invullen, waarna de styling verandert op basis van de invoer. De EditMenuComponent stuurt de geüpdatete styling terug naar de StylingService.
In alle componenten die een abonnement hebben op de StyleService wordt er gecontroleerd of de StyleService.uuid overeenkomt met hun eigen uuid. Als dit het geval is wordt de geüpdatete styling toegepast op het component. Hierdoor wordt de rapportweergave live geüpdatet bij het aanpassen van de styling,

```
styling: StyleModel[] = [
    { key: Customization.text, value: this.text, elementId: 'header' },
    { key: Customization.fontSize, value: '12', elementId: 'header' },
    { key: Customization.color, value: '#000000', elementId: 'header' },
    { key: Customization.backgroundColor, value: '#000000', elementId: 'line' },
    { key: Customization.height, value: '1', elementId: 'line' },
    { key: Customization.width, value: '250', elementId: this.mainElement }
];
```

Figuur 39: Code voorbeeld van StyleModel

```
export class Customization {
    static readonly padding = new Customization( label: 'padding', suffix: 'px');
    static readonly fontSize = new Customization( label: 'fontSize', suffix: 'px');
    static readonly fontFamily = new Customization( label: 'fontFamily', suffix: '');
```

Figuur 40: Voorbeeld code van Customization class

```
export interface StyleModel {
    key: Customization;
    value: string;
    elementId: string;
}
```

Figuur 41: Voorbeeld code van StyleModel interface

De StyleModel interface heeft 3 verschillende variabelen:

- **Key:** de key is van het type Customization en hiermee wordt er aangeduid over welk soort CSS het gaat en welke suffix er aan toegevoegd moet worden.
- **Value:** de waarde van de styling.
- **elementId:** de id van het element waaraan de styling toegepast zal worden.

De onderstaande methode, **appendStyling(),** zorgt voor het toevoegen van styling aan de elementen. Deze methode maakt gebruik van de styling array van het desbetreffende component en het unieke uuid. Allereerst wordt er gecontroleerd of de meegegeven uuid overeenkomt met de **currentUuid** die opgeslagen is in de StyleService. Wanneer dit het geval is wordt de styling doorlopen en voor elke item wordt de styling toegepast aan het element met het meegegeven **elementId**. Aangezien deze methode door vrijwel elke component gebruikt wordt, is er gekozen om deze te plaatsen in een BaseComponent die vanuit elke component kan worden uitgebreid. Dit voorkomt dubbele code.

```
/**
 * Appends the given styling on the element with the given elementId.
 *
 * @param styling
 * @param suffix
 * @param uuid
 */
appendStyling(styling: StyleModel[], suffix: string, uuid: string = this.uuid): void {
    if (this.styleService.currentUuid === uuid) {
        this.styling = styling;
        styling.forEach(style => {
            let element = document.getElementById( elementId: style.elementId + '-' + suffix);
            element!.style[style.key.label] = style.value + style.key.suffix;
        })
        this.setAlignment(this.mainElement + '-' + suffix);
        this.selectElement( elementId: this.mainElement + '-' + suffix);
    } else {
        this.deselectElement( elementId: this.mainElement + '-' + suffix)
    }
}
```

Figuur 42: Voorbeeld code appendStyling() methode

OPSLAAN VAN DATA

Voor het opslaan van een rapportdefinitie zijn er meerdere mogelijkheden. Aangezien er in de huidige applicatie nog niet gebruik werd gemaakt van rapportonderdelen en styling, zal dit niet gaan werken met de nieuwe editor en moesten er veranderingen gedaan worden aan de `rapport_definitie` tabel. Hiervoor zijn er een aantal ideeen bedacht.

Idee 1: Rapportonderdelen als JSON-field

Het eerste bedachte idee was om aan de huidige `rapport_definitie` tabel een kolom "`rapportOnderdelen`" van het type JSON toe te voegen. Hierin zal dan een array opgeslagen worden met alle onderdelen (en bijbehorende styling) die bij het rapport horen.

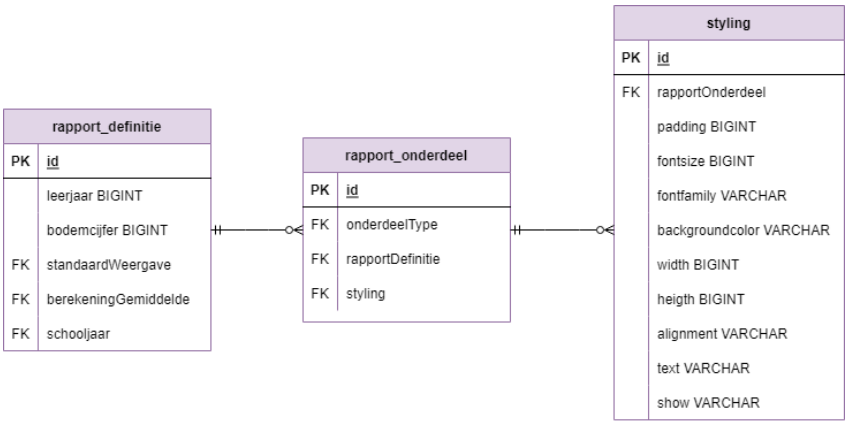

```
[
  {
    "type":"logo",
    "id":"754899",
    "styling":{
      "font":"12",
      "backgroundColor":"#ffffff",
      "width":"60"
    }
  },
  {
    "type":"grafiek",
    "id":"754899",
    "styling":{
      "font":"12",
      "backgroundColor":"#ffffff",
      "width":"60"
    }
  }
]
```


Figuur 44: Voorbeeld van rapportonderdeel JSON-data

Figuur 43: Databasemodel idee 1

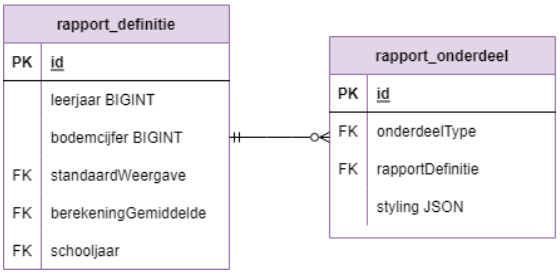
Idee 2: Aparte tabel voor rapportonderdelen en styling

Een ander idee was om twee nieuwe tabellen aan te maken; "`rapport_onderdeel`" en "`styling`". Voor elke nieuwe rapportonderdeel wordt er dan een nieuw item aan de tabel toegevoegd, met een foreign key naar de `rapport_definitie`. Hetzelfde geldt voor de `styling` tabel.



Figuur 45: Databasemodel idee 2

Idee 3: Aparte tabel voor rapportonderdelen en styling als JSON-field
Het laatste idee was een combinatie van bovenstaande opties. Er wordt een nieuwe tabel "`rapport_onderdeel`" gemaakt welke een foreign key heeft naar de `rapport_definitie`. In plaats van dat er een aparte `styling` tabel gemaakt wordt met een verwijzing naar `rapport_onderdeel`, zal er aan de `rapport_onderdeel` tabel een `styling` kolom toegevoegd worden van het type JSON.



Figuur 46: Databasemodel idee 3

Database
Er zijn twee soorten databases die gebruikt konden worden voor het opslaan en ophalen van de data; relationele databases en NoSQL databases.

Bij relationele databases worden de gegevens opgeslagen in tabellen met rijen en kolommen. De verschillende tabellen kunnen met elkaar worden verbonden door een kolom toe te voegen met daarin een verwijzing (foreign key) naar een record in een andere tabel. De tabellen hebben een vast schema en gebruiken SQL om gegevens te beheren.

NoSQL databases zijn vrijwel het tegenovergestelde en verwijzen naar krachtige niet-relationele gegevensarchieven. In plaats van tabellen met genormaliseerde gegevens samen te voegen, slaat NoSQL ongestructureerde gegevens op, vaak in key/value paren of JSON-documenten. NoSQL databases blinken uit in hun gebruiksgemak, schaalbaarheid en beschikbaarheid.

Aangezien een rapportdefinitie een dynamisch aantal onderdelen en styling kan bevatten, waardoor deze erg groot kan groeien, zou de voorkeur uitgaan naar een NoSQL database. In dit geval zou een rapportdefinitie opgeslagen kunnen worden als een grote JSON, in plaats van kolommen te maken voor elke verschillende waardes.
Echter is het uiteindelijk de bedoeling dat de rapport editor geïntegreerd zal worden in de ParnasSys applicatie, welke gebruik maakt van PostgreSQL. En dus is in overleg met de bedrijfsbegeleider besloten dat de data opgeslagen zal worden in een PostgreSQL relationele database.

Bij het gebruik van een relationele database is het meest logische om gewoon voor elke waarde een kolom aan te maken in plaats van een JSON-object. Het gebruik van JSON in een database brengt voor- en nadelen met zich mee.
Zo kunnen er bij JSON geen foreign keys gemaakt worden en is het lastig om relaties te leggen tussen bepaalde data. Ook is een blob van JSON data lastig en traag om op te halen. Het duurt dan ook erg lang om te zoeken in de data omdat er elke keer een tekst-search uitgevoerd moet worden. Een voordeel van JSON data is dat het gemakkelijk is om een andere key/value paar in een JSON-field te steken.

Om deze redenen is er in overleg met de bedrijfsbegeleider besloten om voor het derde idee te kiezen en gewoon een apart `rapport_onderdeel` tabel te maken. De reden dat de styling een JSON-field zal zijn is dat het eigenlijk niet nodig is om hier een nieuwe tabel voor te maken en omdat de styling van een rapportonderdeel nogal kan variëren en dus nooit dezelfde velden heeft.

HET INVOEGEN VAN ONDERDELEN

Aan een rapport kunnen onderdelen toegevoegd worden. Dit zijn bepaalde componenten zoals een logo, legenda, grafiek, handtekeningvakken en opmerkingen. Dit gebeurt in het RapportComponent door middel van een `ComponentFactoryResolver`.
ComponentFactoryResolver is een simpel register dat gebruikt kan worden om componenten dynamisch aan te maken.
Op basis van de onderdeelType parameter dat wordt meegegeven aan de addOnderdeel() methode wordt er in een switch case het juiste component gecreëerd.
Vervolgens wordt het DOM-element van de pagina opgehaald door middel van bijbehorende elementId en de huidige pagina. Het zojuist gecreëerde component zal vervolgens met de HTML DOM appendChild() methode gekoppeld worden aan de pagina.

```
addOnderdeel(onderdeelType: Onderdelen): void {
    switch (onderdeelType) {
        case Onderdelen.logo:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(LogoComponent);
            break;
        case Onderdelen.grafiek:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(GrafiekComponent);
            break;
        case Onderdelen.legenda:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(LegendaComponent);
            break;
        case Onderdelen.definitie:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(DefinitieComponent);
            break;
        case Onderdelen.opmerking:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(OpmerkingComponent);
            break;
        case Onderdelen.handtekening:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(SignatureComponent);
            break;
        case Onderdelen.rubriek:
            this.factory = this.componentFactoryResolver.resolveComponentFactory(RubriekComponent);
            break;
    }

    let page = document.getElementById( elementId: 'rapport-page-' + (this.currentPage - 1));
    let innerPage = document.getElementById( elementId: 'inner-page-' + (this.currentPage - 1));
    this.componentRef = this.factory.create(this.injector, projectableNodes: []);
    this.componentRef.instance['boundary'] = innerPage;

    const pagePadding = page?.style.padding;

    this.componentRef.location.nativeElement.style.cssText = 'position:absolute; left:' + pagePadding
                                                            + '; top:' + pagePadding + '; width:200;';

    innerPage?.appendChild(this.componentRef.location.nativeElement);
    this.dataService.components.push(this.componentRef);

    this.app.attachView(this.componentRef.hostView);
}
```

Figuur 47: Code voorbeeld addOnderdeel() methode

In eerste instantie was het de bedoeling om het dynamische component te koppelen door middel van `ViewContainerRef()` van Angular. `ViewContainerRef` is een soort DOM-container waar nieuwe gegenereerde componenten in geplaatst kunnen worden. Helaas was deze gemakkelijkere oplossing niet mogelijk in de applicatie doordat er rekening gehouden moest worden met het hebben van meerdere rapportpagina's. Zoals te zien is in onderstaande voorbeeld wordt een `ViewContainerRef` gekoppeld aan een @ViewChild welke luistert naar veranderingen in een DOM element.

```
export class AppComponent {
    @ViewChild('container', { read: ViewContainerRef })
    container!: ViewContainerRef;

    constructor(private componentFactoryResolver: ComponentFactoryResolver) {
    }

    add(): void {
        const dynamicComponentFactory = this.componentFactoryResolver
                                            .resolveComponentFactory(RapportOnderdeelComponent);
        const componentRef = this.container.createComponent(dynamicComponentFactory);
    }
}
```

Figuur 48: Code voorbeeld

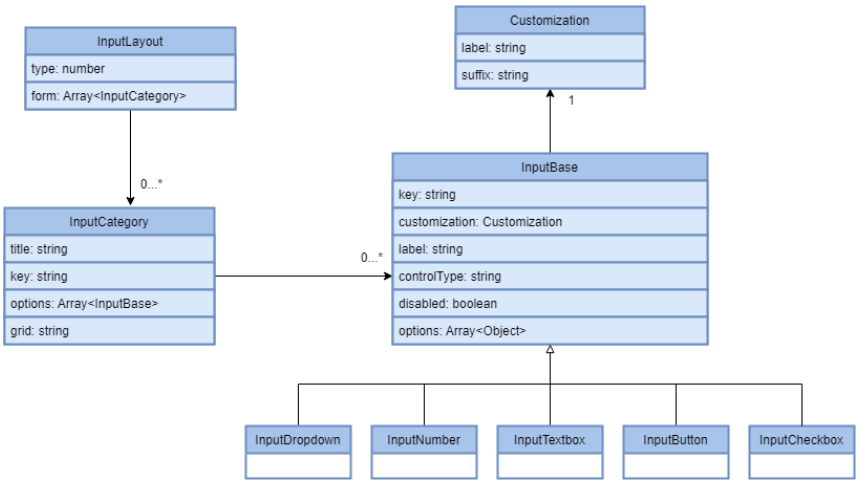
Aangezien rapportpagina's dynamisch geladen worden op basis van een `currentPage` variabele, is het niet mogelijk om hier een @ViewChild aan te koppelen. Wat er dan zou gebeuren is dat een nieuwe component weergegeven zal worden op alle pagina's in plaats van de huidige pagina. Vandaar dat er gekozen is voor bovenstaande oplossing.

```
<div *ngFor="let page of pages; index as i">
    <div [hidden]="(currentPage-1)!==i" class="rapport-page" id="{{'rapport-page-'+i}}" (click)="onSelect($event, Onderdelen.rapport, this.styling)">
        <div class="inner-page" id="{{'inner-page-'+i}}">
            <p>pagina {{i + 1}}</p>
            <app-rubriek></app-rubriek>
        </div>
    </div>
</div>
```

Figuur 49: Code voorbeeld rapport html

DYNAMISCHE FORMULIEREN

De gehele applicatie bevat meerdere formulieren. Voor elk van de verschillende rapportonderdelen wordt er een ander formulier weergegeven in het EditMenuComponent op basis van de desbetreffende styling-opties. Om voor elk van deze formulieren een aparte template te maken is redelijk onnodig en om deze reden is er bedacht om een systeem te implementeren die ervoor zorgt dat formulieren op een dynamische wijze opgesteld kunnen worden.



Figuur 50: Classdiagram input entiteiten

De InputLayout staat voor het gehele formulier. Een formulier kan uit meerdere categorieën bestaan met elk een eigen titel; de InputCategory. ook heeft elke InputCategory een grid variabele, welke aangeeft hoeveel invoervelden er naast elkaar komen te staan. Tot slot kan een InputCategory meerdere InputBases hebben. De InputBase dient als een soort basis class voor een set besturingselementen die omgezet zal worden in een invoerveld.

```
layouts: InputLayout[] = [
    new InputLayout(Onderdelen.rubriekMenu, form: [
        new InputCategory( title: 'Rubriek', key: 'header', grid: '1fr', options: [
            new InputTextbox( options: {
                key          : 'text',
                customization: Customization.text,
                label        : 'Naam',
            }),
            new InputCheckbox( options: {
                key          : 'show',
                customization: Customization.show,
                label        : 'Show',
                options      : [{ key: 1, value: 'actief' }],
            }),
        ])
    ]),
```

Figuur 51: Code voorbeeld InputLayout

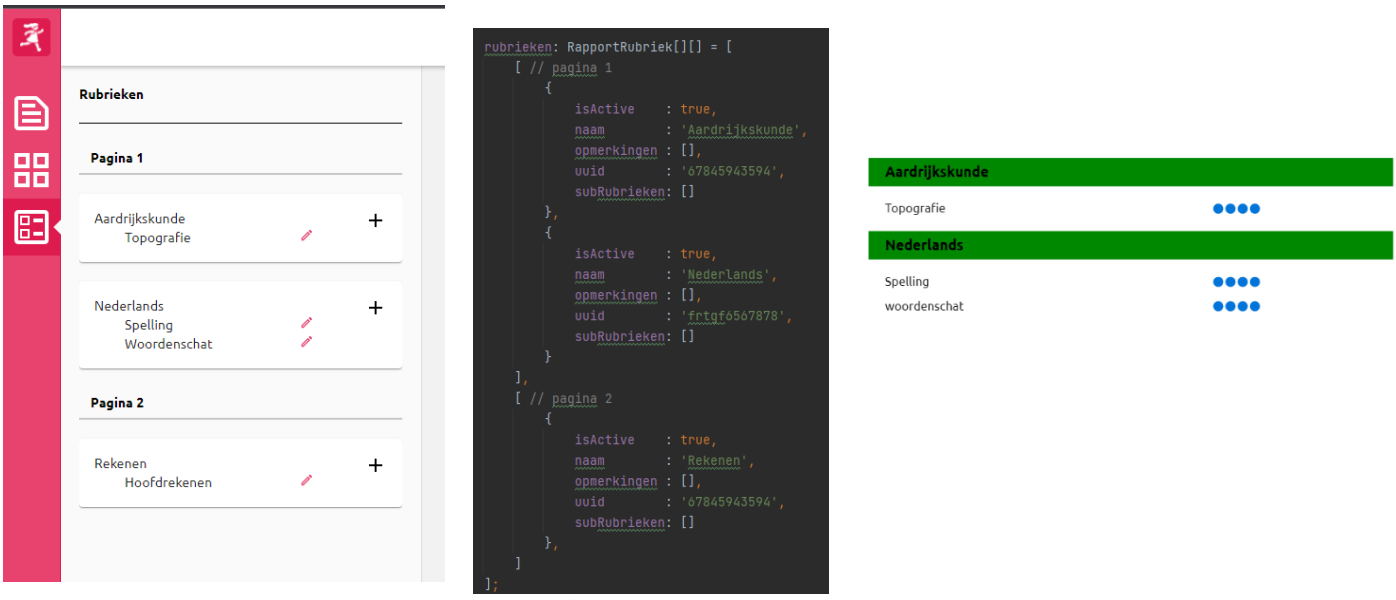
Figuur 52: EditMenuComponent

Met behulp van bovenstaande classes kunnen er in de FormService meerdere InputLayouts aangemaakt worden. Door middel van een Observer in de EditMenuComponent kan er een layout naar keuze opgehaald worden welke vervolgens in de HTML van dit component omgezet wordt naar een formulier, aan de hand van deze layout en zijn waardes. Voor elke InputBase wordt er een invoerveld weergegeven. Ook hiervan zijn er verschillende soorten beschikbaar (textbox, dropdown, checkbox)

HET BEHEREN VAN RUBRIEKEN

Aan een rapport kunnen rubrieken toegevoegd worden. Dit zijn eigenlijk de vakken die weergegeven zullen worden op een rapport. Het is de bedoeling dat een leerkracht deze zelf kan aanmaken, met daaronder subrubrieken. Hiervoor is bedacht om dit door middel van een lijst met cards te doen die vervolgens versleept kunnen worden om zo de volgorde te veranderen.

Een RapportDefinitie heeft een Array (rubrieken) met daarin meerdere RapportRubrieken. Voor alle pagina's van een rapport wordt er een nieuwe entry in de rubrieken array gemaakt. Hierdoor zijn de rubrieken per pagina duidelijk van elkaar afgesloten. Wanneer een gebruiker één van de rubrieken versleept, zal er door middel van een moveItemInArray() methode de rubriek naar een andere index verplaatst worden. Al deze logica gebeurt in het RubriekenMenuComponent.



Figuur 53: De weergave van rubrieken op verschillende plekken (in het menu, in de code, op het rapport)

Dit idee is tot stand gekomen door een soortgelijke functionaliteit in het huidige Digitale Rapport. Het aanpassen van de volgorde van de rubrieken wordt hier gedaan door op pijltjes te drukken waarna de desbetreffende rubriek naar boven of naar onder in de lijst verschuift. Dit ziet er vrij onoverzichtelijk uit doordat er bij elke rubriek en subrubriek dezelfde icoontjes staan. Ook wordt er geen ondersteuning geboden voor het verplaatsen van rubrieken tussen verschillenden pagina's.

Figuur 54: Rubrieken weergave huidige Digitale Rapport

WEBCOMPONENTS

Een van de gevraagde requirements was dat rapporten ingesloten konden worden in andere applicaties van ParnasSys. Hiervoor moest er onderzoek gedaan worden naar een juiste techniek zodat dit zo goed mogelijk gerealiseerd kan worden.
Uit dit onderzoek is voortgekomen dat Webcomponents van Angular hier het meeste geschikt voor zijn. Met gebruik van de angular elements package @angular/elements.
Het was niet de bedoeling om de gehele applicatie te embedden maar alleen het rapport zelf. Echter is er voor gekozen om de bottom-toolbar hierbij te doen zodat gebruikers nog steeds kunnen wisselen tussen pagina's.

```
export class AppModule implements DoBootstrap {

    constructor(private injector: Injector) {
        const webComponent = createCustomElement(RapportComponent, config: { injector });
        customElements.define( name: 'rapport-view', webComponent);
    }
```

Figuur 55: Voorbeeld code Webcomponents

Het omzetten van een Component, in dit geval het RapportComponent, naar een webcomponent is niet heel ingewikkeld. Het enige wat er eigenlijk hoefde te gebeuren was invoegen van bovenstaande code in de AppModule om vervolgens een build command uit te voeren in de terminal:

```
$ ng build --prod --output-hashing none
```

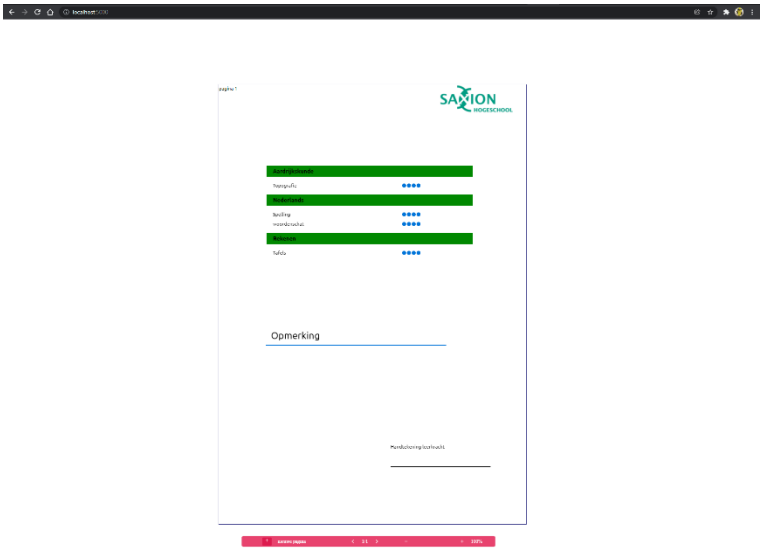
Dit command maakt een nieuwe map aan met een aantal gegenereerde built bestanden. Deze bestanden zijn noodzakelijk om de webcomponent te kunnen gebruiken in een andere applicatie. Vervolgens kan de tag die gedeclareerd is in de AppModule (in dit geval rapport-view) geplaatst worden in een index.html.

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, ini
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <base href="/">
</head>
<body>
    <rapport-view></rapport-view>

    <script type="text/javascript" src="rapport-element.js
```

Figuur 56: Ingesloten webcomponent in index.html

De uitdaging zat hem dan ook niet zozeer in het bouwen van een webcomponent maar meer in het weergeven van de juiste rapport-data. Het rapport moet namelijk de data van één specifieke leerling kunnen weergeven en dus zou die op een of andere manier een invoer moeten krijgen van de applicatie waarin die ingesloten is. Na veel onderzoek gedaan te hebben bleek dat de Angular @Input() decorator ook werkt bij Webcomponents, waardoor dit probleem op deze manier opgelost kon worden: `<rapport-view id=2></rapport-view>`. Door het id van een rapport als invoer te gebruiken kan er vanuit de rapport editor op basis daarvan het juiste leerling-rapport weergegeven worden.



Figuur 57: Voorbeeld rapport in andere applicatie

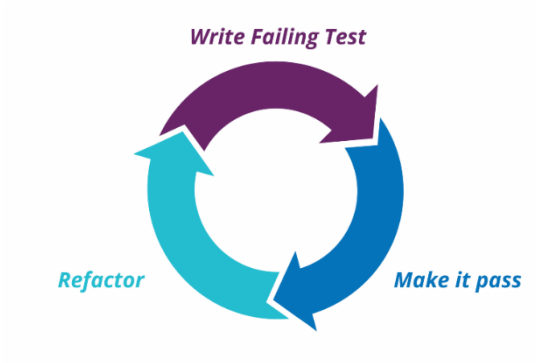
10. VALIDATIE

Gedurende het gehele project is er geregeld gevalideerd. Dit is gedaan door onder andere het testen en laten reviewen van de code. In dit hoofdstuk staat beschreven wat er allemaal gedaan is qua validatie.

Test Driven Development

Het testen van de applicatie is gedaan volgens de Test-driven development methode. Bij deze ontwikkelmethode worden er eerst tests geschreven en daarna pas de code. De cyclus bestaat uit de volgende stappen:

1. Test maken
2. Controleer of de test faalt
3. Code schrijven
4. Controleer of de test slaagt
5. Code herschrijven



Figuur 58: Test Driven Development

Voordelen van het werken volgens deze methode is dat de testen voornamelijk worden geschreven op basis van de user stories. Hierdoor kunnen bepaalde problemen eerder gevonden worden. Ook zal de ontwikkeltijd korter worden aangezien fouten in een zeer vroeg stadium gevonden zullen worden. Doordat er eerst testen geschreven worden alvorens er gecodeerd wordt, is er altijd zekerheid dat alle code getest is.

```
it( expectation: 'should update the styling of newly added pages', fakeAsync( fn: () => {
    const styling = [{ key: Customization.padding, value: '20', elementId: 'rapport-page' }];

    fixture.detectChanges();

    component.styling = styling;
    component.appendStylingOnEveryPage(styling)
    let pages = findElementsByClass(fixture, selector: '.rapport-page');
    expect(pages.length).toBe( expected: 1);
    expect(pages[0].styles['padding']).toBe( expected: '20px');

    component.addPage();
    fixture.detectChanges();
    tick( millis: 100);

    pages = findElementsByClass(fixture, selector: '.rapport-page');
    expect(pages.length).toBe( expected: 2);
    expect(pages[1].styles['padding']).toBe( expected: '20px');
  }));
});
```

```
it( expectation: 'should show different content based on the currentTab', assertion: () => {
    component.currentTab = component.tabs[1];
    fixture.detectChanges();

    let definitieForm = findElementById(fixture, id: '#definitie-form');
    let onderdelenMenu = findElementById(fixture, id: '#onderdelen-menu');
    let rubriekenMenu = findElementById(fixture, id: '#rubrieken-menu');

    expect(definitieForm).toBeNull();
    expect(onderdelenMenu).not.toBeNull();
    expect(rubriekenMenu).toBeNull();
});
```

Figuur 59: Voorbeeld van een aantal testen

Pull Requests

Een pull request is een mechanisme in GitHub waarbij een ontwikkelaar een verzoek kan indienen bij teamleden om te laten weten dat bepaalde code gereed is. Dit laat alle betrokkenen weten dat ze de code kunnen bekijken en hier feedback op kunnen geven.

Gedurende de realisatiefase is er met de bedrijfsbegeleider afgesproken om na iedere sprint een pull request te maken op GitHub met alle nieuwe code zodat hij hier feedback op kon geven waar nodig. Deze zou dan in de volgende sprint verwerkt worden. Deze manier van werken gaf een goede validatie dat de gemaakte code van een zekere kwaliteit was en dat er geen gekke dingen in stonden.

src/app/components/dynamic-form/checkbox-input/checkbox-input.compone

```
...  ...   @@ -0,0 +1,5 @@
1    +  <section class="example-section" [formGroup]="group">
2    +      <p *ngFor="let option of inputField.options">
```

tedroeloffzen on 8 Dec 2021

Is er een specifieke reden dat je de mat-checkbox binnen een

hebt staan?
Lijkt mij niet direct nodig of wel?

src/app/components/results/result-display/result-display.component.html

```
11  +      </div>
12  +
13  +      <div class="item" *ngSwitchCase="WeergaveType.BLOKJE">
14  +          <div *ngIf="van >= 1 && tot <= 2" class="block" [style
```

tedroeloffzen on 8 Dec 2021

hetzelfde hier. Dit zou ik ook oplossen door een ngClass en maar 1 element. Dan kun je de background color opnemen in een specifieke class

Figuur 60: Voorbeeld van feedback

11. CONCLUSIES EN AANBEVELINGEN

Gedurende de afstudeerperiode is er gezocht naar een antwoord op de vraag: Hoe kan het Digitale Rapport verbeterd worden zodat leerkrachten meer vrijheid krijgen in het uiterlijk van het rapport?
Hiervoor zijn een aantal mogelijk technieken onderzocht, maar het grootste gedeelte bestond uit het implementeren van een prototype van deze rapport editor.

Voor het beantwoorden van de onderzoeksvraag is het project begonnen met een aantal onderzoeken naar onder andere de huidige staat van het Digitale Rapport en mogelijke technieken die gebruikt kunnen worden. Uit dit onderzoek zijn onder andere een aantal handige Angular Material componenten en functionaliteiten gekomen.
Ook is er uit het onderzoek voortgekomen dat de Angular Elements library de beste oplossing biedt voor het embedden van een rapport in een andere applicatie. Doordat de applicatie zelf ook gebouwd is in Angular is een component binnen enkele stappen omgezet naar een webcomponent. Wel heeft Angular Elements het nadeel dat bij het omzetten van grotere delen van een applicatie naar een webcomponent de navigatie zelf geregeld moet worden. Gelukkig is dat niet van belang in deze situatie aangezien het rapport uit één enkele component bestaat.

Het prototype voldoet aan alle vooraf opgestelde requirements. Een leerkracht kan nieuwe rapportdefinities aanmaken, bestaande definities inladen, rapporten embedden in andere applicaties en de indeling van een rapport bepalen. De koppeling met de applicatie van ParnasSys is niet geïmplementeerd, waardoor er nog geen echte data weergegeven kan worden in de editor. Echter is er wel een koppeling met een Node.js backend en een Postgres database gerealiseerd waardoor de data zoveel mogelijk overeenkomt met de gegevens uit ParnasSys die uiteindelijk weergegeven zal moeten worden.

Het prototype is getest door unit testen te schrijven alvorens er code gemaakt was. Hierdoor is er een zekerheid dat alle geschreven code getest is, en dus ook allemaal zal werken. Op dit moment ontbreken nog de integration tests. Deze zij nodig om de samenhang tussen verschillende onderdelen te testen.

Met de huidige staat van het prototype zou een leerkracht in staat moeten zijn om een eigen indeling te maken voor een rapport. Echter zijn er nog een aantal verbeterpunten die het bewerken van rapporten gemakkelijker zouden kunnen maken.

Meer styling opties

Op het moment is het mogelijk om de styling van een rapportpagina en rapportonderdelen aan te passen, door middel van invoervelden. Dit gaat dan voornamelijk over de basis styling, zoals bijvoorbeeld achtergrondkleur, tekstgrootte, hoogte en breedte. Echter zijn er nog genoeg andere styling opties die geïmplementeerd zouden kunnen worden.

Voorblad invoegen

Momenteel is het niet mogelijk om een apart voorblad aan een rapport toe te voegen. Dit stond als een could-requirement op de planning maar hier is helaas niks van gekomen.

Meer verschillende rapportonderdelen

Momenteel is het mogelijk om bepaalde onderdelen in een rapport te plaatsen (logo, grafiek, legenda, handtekening, definitie). Echter is er altijd ruimte voor extra andere onderdelen zoals bijvoorbeeld afbeeldingen of tekstvakken.

Andere database

Voor het opslaan van de data is er gekozen voor een PostgreSQL database, omdat dit ook gebruikt wordt in de huidige ParnasSys applicatie. Echter zou een NoSQL database beter passen in deze situatie, aangezien een rapportdefinitie een dynamisch aantal onderdelen en styling kan bevatten. Het zou dus praktischer zijn om een rapportdefinitie op te slaan als een grote JSON, in plaats van kolommen te maken voor elk van de verschillende waardes

12. INHOUDELIJKE REFLECTIE

De eerste paar weken van het afstuderen bestonden voornamelijk uit het installeren van de benodigde programma's, het aanmaken van accounts en het opstellen van het Plan van Aanpak. Ook werd er meer uitleg gegeven over het huidige Digitale Rapport. Dit was fijn want hierdoor kreeg ik een beter idee van wat er al stond en kon dit goed verwerkt worden in het Plan van aanpak.
Aangezien het nog niet helemaal duidelijk was wat er allemaal gevraagd zou worden van mij, heb ik een aantal meetings gehad waaruit naar voren kwam wat er precies gerealiseerd zou moeten worden en welke requirements daar bij hoorden.
Nadat het plan van aanpak afgerond was kon er begonnen worden aan het onderzoek. De deelvragen die hierbij onderzocht gingen worden waren van tevoren al opgesteld.

Met het uitvoeren van het onderzoek ben ik drie weken bezig geweest. Tijdens dit onderzoek is er gekeken naar de huidige situatie van de applicatie en naar technieken die gebruikt konden worden. Van tevoren was er al bepaald dat de applicatie in Angular gebouwd zou worden. Hier was ik blij mee aangezien ik hier al veel ervaring mee had opgedaan.
Op een gegeven moment is er onderzoek gedaan naar een library voor het weergeven van grafieken op een rapport. Doordat er veel verschillende opties beschikbaar zijn hiervoor moest er op een bepaalde manier een keuze gemaakt worden. Aan de hand van een aantal criteria werden de libraries met elkaar vergeleken waarna uiteindelijk de beste keuze naar voren kwam. Gedurende de specialisatie: Advanced App Development is er aangeleerd om via deze manier een library of framework te kiezen, vandaar dat ik via deze manier gehandeld had.

De realisatie van het prototype kwam vrij spoedig op gang, mede doordat ik al enige ervaring had met Angular. Binnen een aantal dagen stond het begin van de applicatie er al. Over het algemeen vond ik de realisatie van de applicatie goed verlopen, zonder dat ik echt grote problemen tegenkwam. Het meest ingewikkelde vond ik toch het creëren van de samenhang tussen verschillende componenten. Zo moesten vrijwel alle componenten communiceren met een bepaalde service om zo de juiste styling binnen te krijgen. Doordat dit op meerdere plekken hetzelfde moest zijn wou ik dit zo consistent mogelijk implementeren, en naarmate de applicatie groeide werd dit steeds meer een uitdaging.
Ook het leggen van de connectie met een database en backend was een echte uitdaging. Er moest namelijk rekening gehouden worden met het klassensysteem in de ParnasSys applicatie en het opslaan van de opmaak van een rapportdefinitie. Over dit laatste heb ik dan ook veel gebrainstormd met mijn bedrijfsbegeleider.

Door te werken volgens Test Driven Development denk je automatisch beter na over de verschillende situaties die kunnen optreden. Doordat ik de testen van tevoren al moest opstellen, werd het testen ook beter gehandhaafd. Normaal gesproken ben ik daar nooit zo strikt in, vandaar dat ik gekozen had om via deze methode te werken.

Zoals in de aanbevelingen van het verslag te lezen is, zijn er nog een aantal dingen die verder uitgebreid zouden kunnen worden om de applicatie gebruiksvriendelijker te maken. Al met al kan ik zeggen dat ik tevreden ben, en ik denk dat er een goede eerste versie is opgeleverd van het prototype.

13. BIBLIOGRAFIE

8 ADVANTAGES OF ANGULAR FOR BUSINESSES AND DEVELOPERS. (n.d.). Retrieved from Light-IT: https://light-it.net/blog/8-advantages-of-angular-for-businesses-and-developers/

A developer's guide to the most common open-source licenses (MIT, Apache 2.0, BSD). (2020). Retrieved from ghinda.com: https://ghinda.com/blog/opensource/2020/open-source-licenses-apache-mit-bsd.html

About Wicket. (n.d.). Retrieved from cwiki.apache.org: https://cwiki.apache.org/confluence/display/WICKET#Index-AboutWicket

Ambulkar, S. (2016). *Angular JS.* India: Department of CSE. Retrieved from https://www.ijser.org/researchpaper/Angular-JS.pdf

Angular. (n.d.). Retrieved from angular.io: https://angular.io/

Angular Chartist. (n.d.). Retrieved from npmjs.com: https://www.npmjs.com/package/ng-chartist

Angular elements overview. (n.d.). Retrieved from angular.io: https://angular.io/guide/elements

Angular Material. (n.d.). Retrieved from material.angular.io: https://material.angular.io

angular-billboard. (n.d.). Retrieved from github.com: https://github.com/akysil/angular-billboard

ApexCharts.js. (n.d.). Retrieved from apexcharts.com: https://apexcharts.com

Billboard.js. (n.d.). Retrieved from Billboard.js: https://naver.github.io/billboard.js/

Canvas vs SVG: Choosing the Right Tool for the Job. (2021). Retrieved from sitepoint.com: https://www.sitepoint.com/canvas-vs-svg/

Chart.js. (n.d.). Retrieved from chartjs.org: https://www.chartjs.org/

Chartist.js. (n.d.). Retrieved from chartist-js: https://gionkunz.github.io/chartist-js/

Comparison of JavaScript charting libraries. (2021). Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_libraries

jsPDF. (n.d.). Retrieved from github.com: https://github.com/parallax/jsPDF

ng2-charts. (n.d.). Retrieved from github.com: https://github.com/valor-software/ng2-charts

npm-stat. (n.d.). Retrieved from npm-stat.com: https://npm-stat.com/

Open Source Software Licenses 101: The BSD 3-Clause License. (2021). Retrieved from fossa.com: https://fossa.com/blog/open-source-software-licenses-101-bsd-3-clause-license/

Open Source Software Licenses 101: The MIT License. (2021). Retrieved from fossa.com: https://fossa.com/blog/open-source-licenses-101-mit-license/

Plotly JavaScript Open Source Graphing Library. (n.d.). Retrieved from plotly.com: https://plotly.com/javascript/

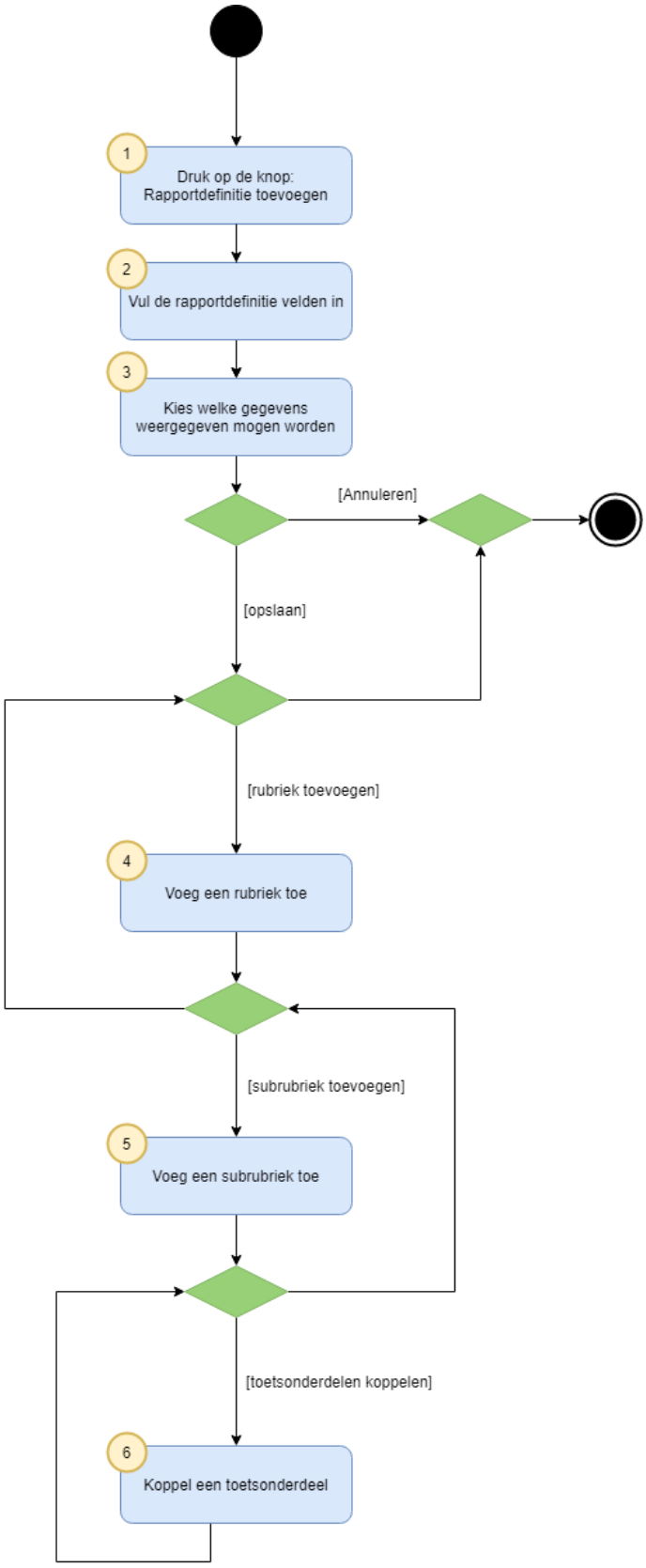
SVG vs canvas: how to choose. (2016). Retrieved from Microsoft docs: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/samples/gg193983(v=vs.85)?redirectedfrom=MSDN

Webcomponents. (n.d.). Retrieved from webcomponents.org: https://www.webcomponents.org

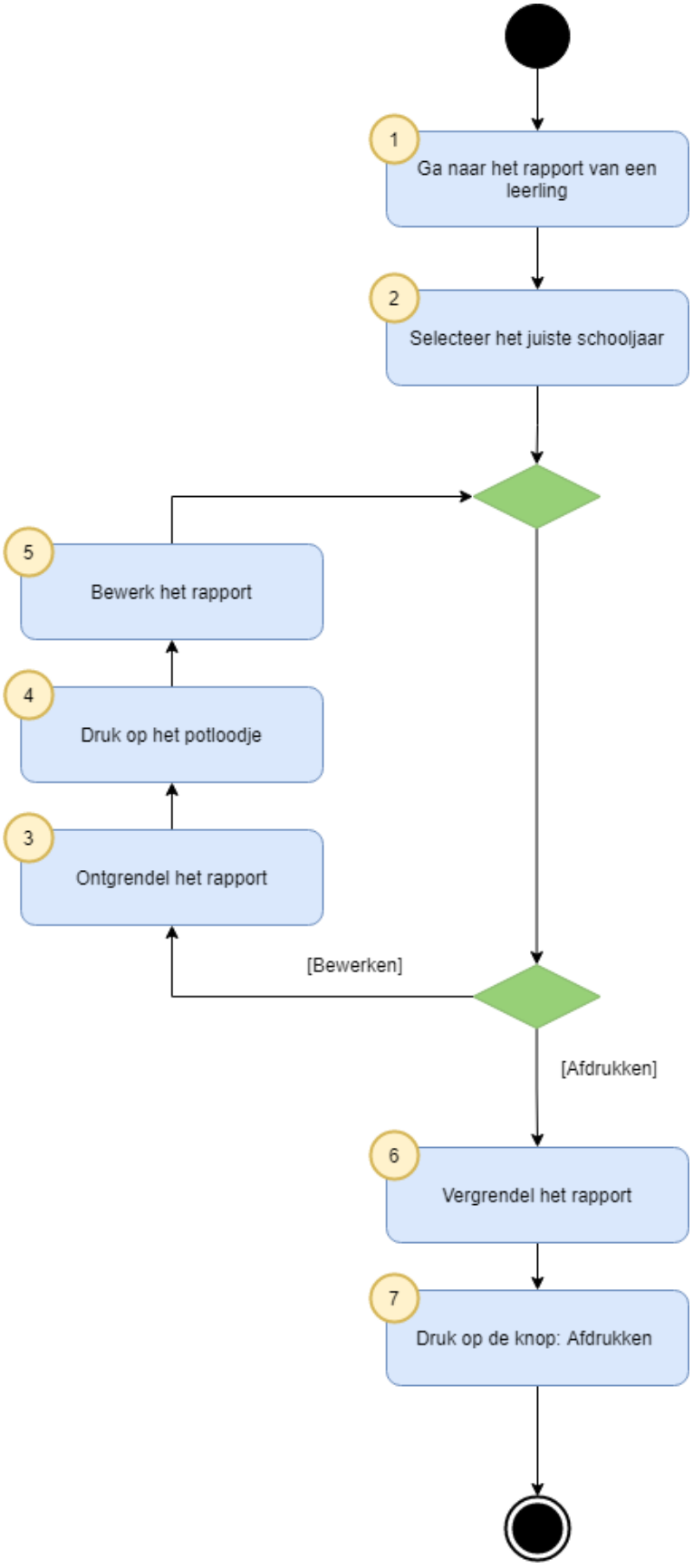
Wkhtmltopdf. (n.d.). Retrieved from wkhtmltopdf.org: https://wkhtmltopdf.org/

12. BIJLAGEN

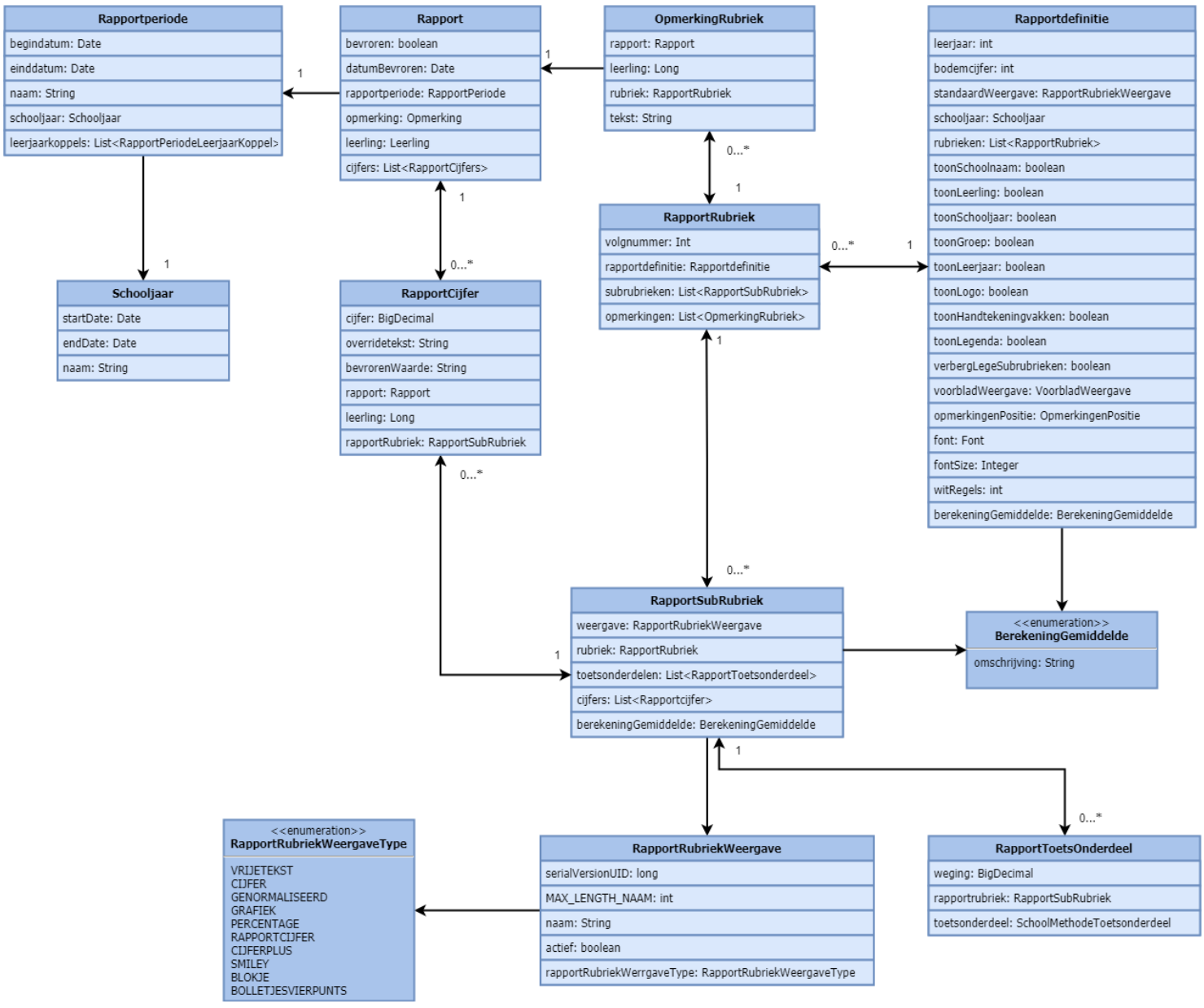
BIJLAGE A: ACTIVITY DIAGRAM: AANMAKEN VAN EEN RAPPORT



BIJLAGE B: ACTIVITY DIAGRAM: BEWERKEN EN EXPORTEREN VAN EEN RAPPORT



BIJLAGE C: CLASS DIAGRAM HUIDIGE APPLICATIE



BIJLAGE D: DE CONFIGURATIEOPTIES VAN HET DIGITALE RAPPORT

HET AANMAKEN VAN EEN RAPPORT

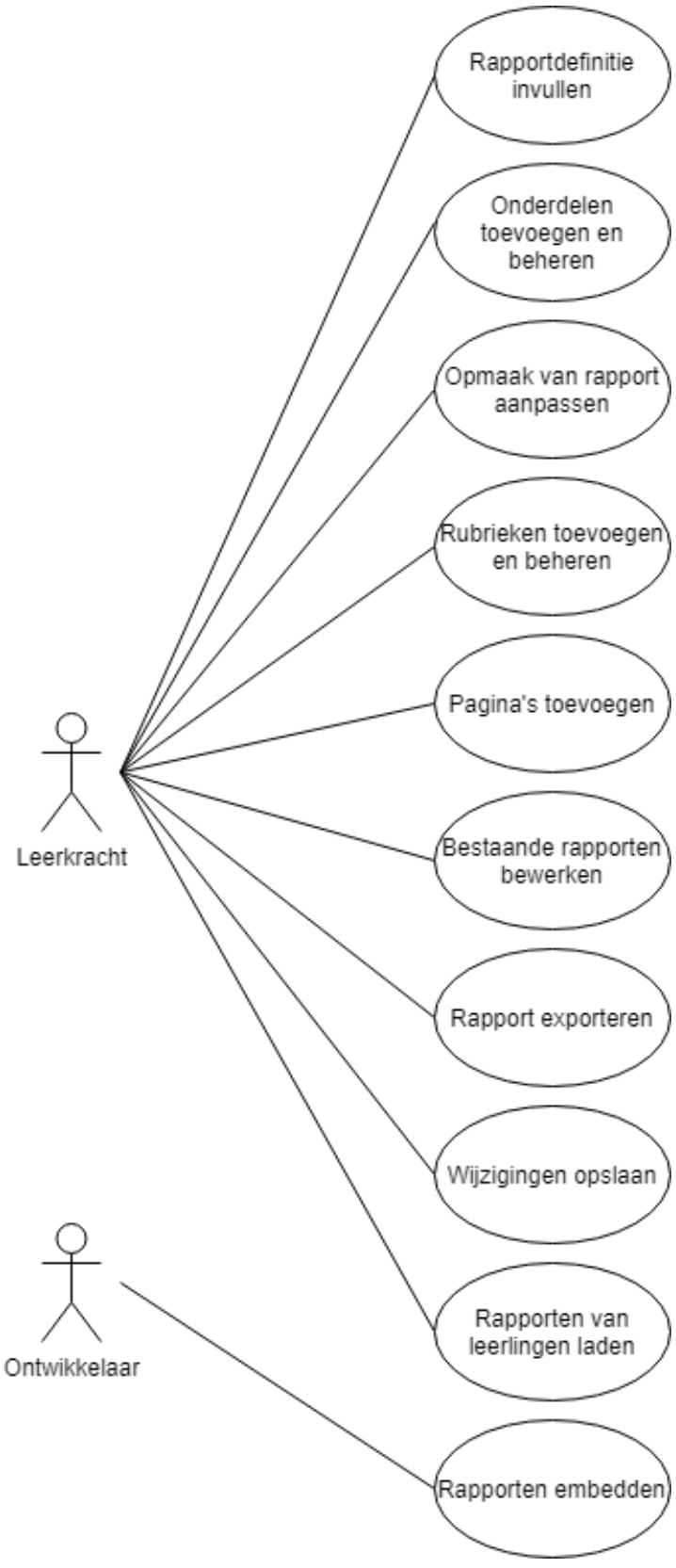
Optie	Element	Verklaring	Keuzemogelijkheden
Leerjaar	Tekstvak	Het leerjaar waar het rapport voor bedoeld is.	-
Bodemcijfer	Tekstvak voor getallen	Het cijfer dat minimaal gehaald dient te worden.	-
Standaard weergave	Selectielijst	De manier waarop de resultaten weergegeven worden.	*Cijfers – Tekst – Bolletjes – Progress bar – Smileys - Percentages*
Berekening rapportgemiddelde	Selectielijst	De manier waarop het gemiddelde berekend zal worden.	*Periode-gemiddelde – Voortschrijdend gemiddelde*
Schoolnaam	Checkbox	Bepaald of de schoolnaam weergegeven zal worden in het rapport.	-
Leerling	Checkbox	Bepaald of de leerling weergegeven zal worden in het rapport.	-
Schooljaar	Checkbox	Bepaald of de schoolnaam weergegeven zal worden in het rapport.	-
Groep	Checkbox	Bepaald of de groep weergegeven zal worden in het rapport.	-
Leerjaar	Checkbox	Bepaald of het leerjaar weergegeven zal worden in het rapport.	
Logo	Checkbox	Bepaald of het logo van de school weergegeven zal worden in het rapport.	-
Handtekeningvakken	Checkbox	Bepaald of er handtekeningvakken weergegeven zullen worden in het rapport.	-
Legenda	Checkbox	Bepaald of er een legenda weergegeven zal worden in het rapport.	-
Verberg lege subrubrieken	Checkbox	Bepaald of lege subrubrieken verborgen moeten worden.	-
Subrubriek naast rubriek	Checkbox	Bepaald of subrubrieken naast rubrieken komen te staan.	-
Voorblad weergave	Selectielijst	Het uiterlijk van het voorblad.	*Geen – Standaard voorblad – Voorblad met pasfoto*
Positie opmerkingen	Selectielijst	De positie van opmerkingen in het rapport.	*Onder hoofdrubriek – In periodekolom – Onderaan rapport*
Lettertype	Selectielijst	Het lettertype dat gebruikt zal worden in het rapport.	*Comic Sans MS – Arial – Verdana – Times New Roman*
Lettertype grootte	Selectielijst	De grootte van het lettertype.	*8 t/m 14*
Aantal witregels tussen rubrieken	Selectielijst	Het aantal witregels tussen de rubrieken.	*0 t/m 10*

HET TOEVOEGEN VAN RUBRIEKEN AAN EEN RAPPORT

Optie	Element	Verklaring	
Naam	Tekstvak	De naam van de rubriek.	-
Actief	Checkbox	Bepaald of de rubriek wel of niet actief is.	-

Optie	Element	Verklaring	Keuzemogelijkheden
Naam	Tekstvak	De naam van de subrubriek.	-
Weergave	Selectielijst	De manier waarop de resultaten weergegeven worden.	*Cijfers – Tekst – Bolletjes – Progress bar – Smileys - Percentages*
Berekening gemiddelde	Selectielijst	De manier waarop het gemiddelde berekend zal worden.	*Periode-gemiddelde – Voortschrijdend gemiddelde*
Actief	Checkbox	Bepaald of de subrubriek wel of niet actief is.	-

BIJLAGE E: USECASEDIAGRAM

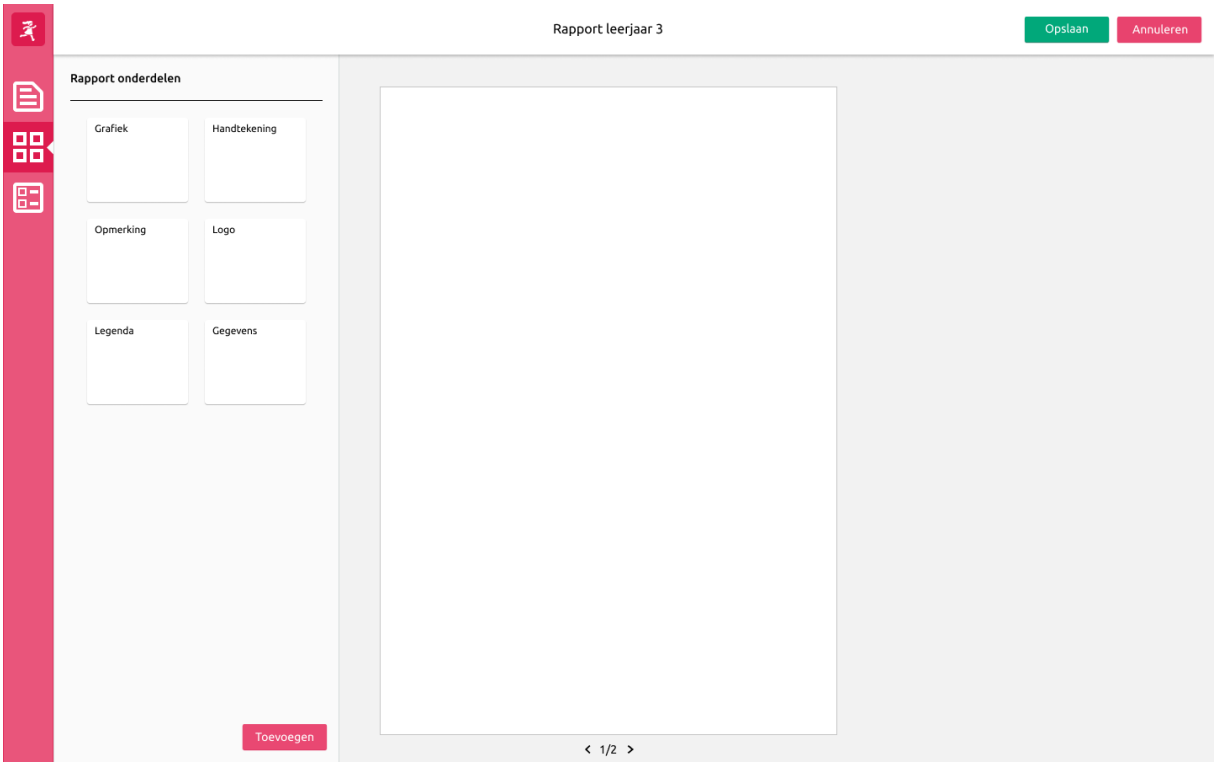


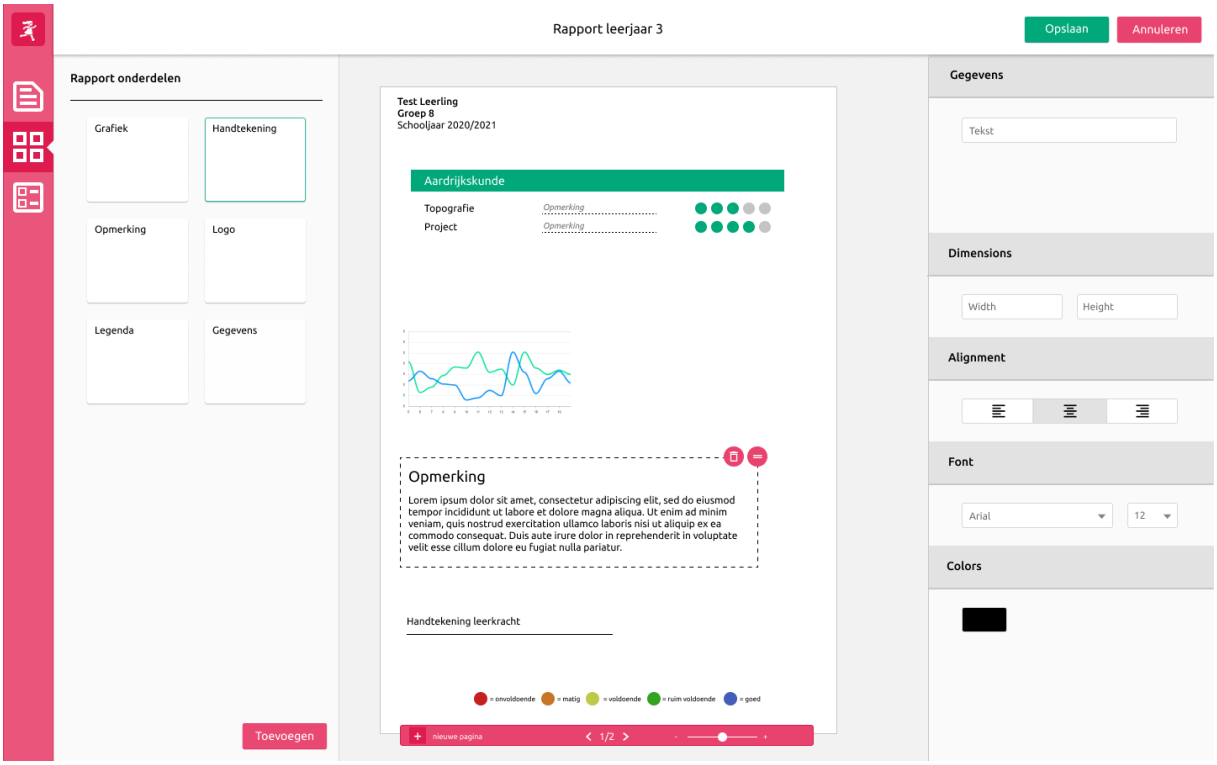
BIJLAGE F: USER STORIES

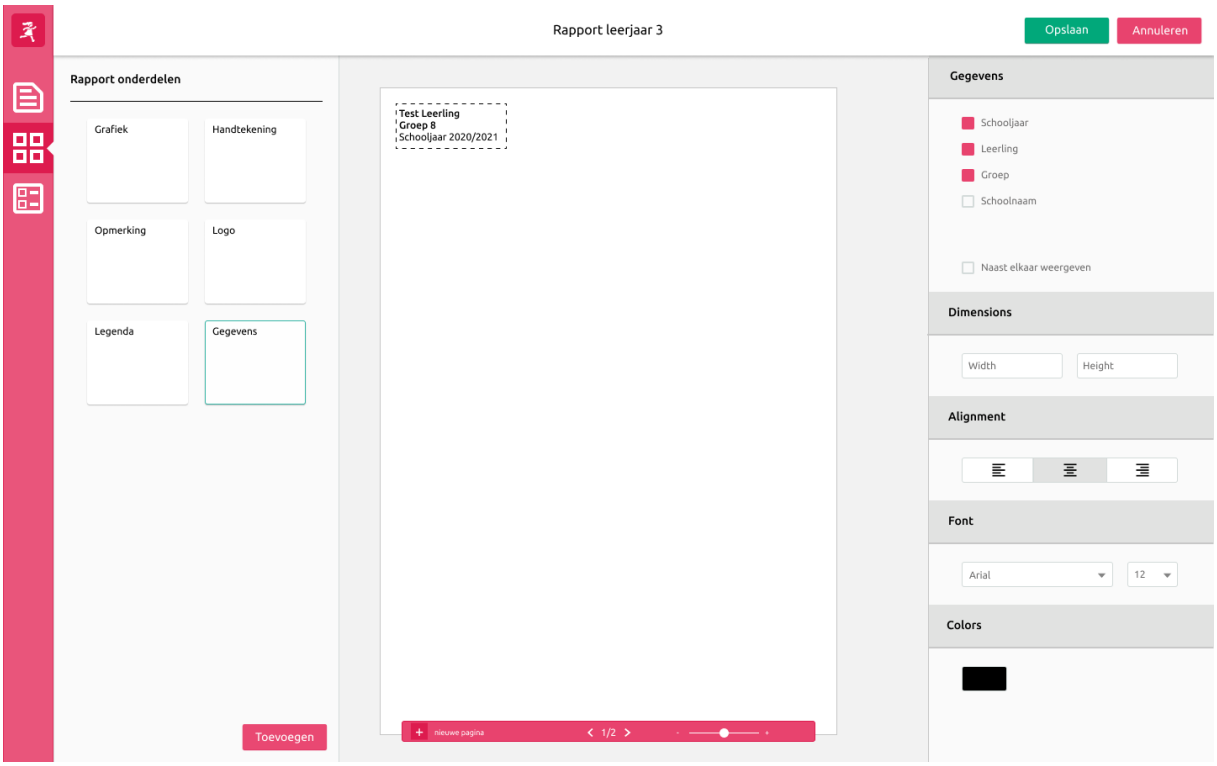
#	User story
US01	Als leerkracht wil ik de algemene opmaak van een pagina aanpassen, zodat ik zelf kan kiezen hoe deze eruit komt te zien.
US03	Als leerkracht wil ik handmatig de positie van een rapportonderdeel op het rapport veranderen, zodat ik zelf de indeling kan bepalen.
US04	Als leerkracht wil ik de rapportdefinitie invullen, zodat het duidelijk is om welke gegevens het gaat.
US05	Als leerkracht wil ik een bestaand rapport aanpassen, zodat ik niet steeds een nieuw rapport hoef aan te maken.
US06	Als leerkracht wil ik de wijzigingen aan een rapport opslaan, zodat ik deze kan gebruiken voor later.
US07	Als leerkracht wil ik de wijzigingen aan een rapport annuleren, zodat ik de keuze heb om een rapport niet aan te passen.
US08	Als leerkracht wil ik rapportonderdelen selecteren uit een menu, zodat ik deze op het rapport kan plaatsen.
US09	Als leerkracht wil ik een live weergave zien van het rapport, zodat ik weet hoe het eruit komt te zien.
US10	Als leerkracht wil ik rapportonderdelen verwijderen van een rapport, zodat ik het rapport een andere indeling kan geven.
US11	Als leerkracht wil ik pagina's kunnen toevoegen aan een rapport, zodat ik grotere rapporten kan maken.
US12	Als leerkracht wil ik resultaten weergeven in een grafiek of diagram, zodat ik een overzicht kan laten zien van de voortgang van een leerling.
US13	Als leerkracht wil ik de opmaak van een grafiek aanpassen, zodat deze overeenkomt met de rest van het rapport.
US14	Als leerkracht wil ik het type grafiek aanpassen, zodat ik de resultaten op een andere manier kan weergeven.
US15	Als leerkracht wil ik een logo toevoegen aan een rapport, zodat het duidelijk is om welke school het gaat.
US16	Als leerkracht wil ik de opmaak van een logo aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US17	Als leerkracht wil ik een legenda toevoegen aan een rapport, zodat ik kan laten weten wat de normering is.
US18	Als leerkracht wil ik de opmaak van een legenda aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US19	Als leerkracht wil ik een handtekeningregel toevoegen aan een rapport, zodat ik een handtekening kan zetten onder het rapport.
US20	Als leerkracht wil ik de opmaak van een handtekeningregel aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US21	Als leerkracht wil ik een opmerking aan het gehele rapport toevoegen, zodat ik kan laten weten wat ik over het gehele rapport te zeggen heb.
US22	Als leerkracht wil ik een opmerking kunnen wijzigen, zodat ik de opmerking kan aanpassen.
US23	Als leerkracht wil ik rubrieken toevoegen aan een rapport, zodat ik hiervan de resultaten kan weergeven.
US24	Als leerkracht wil ik subrubrieken toevoegen aan een rubriek, zodat ik bepaalde resultaten van een rubriek kan weergeven.
US25	Als leerkracht wil ik toetsonderdelen koppelen aan een subrubriek, zodat ik kan bepalen welke toetsen er mee tellen voor het resultaat.
US26	Als leerkracht wil ik de opmaak van een rubriek aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US27	Als leerkracht wil ik het gemiddelde cijfer per subrubriek ophalen, zodat ik het gemiddelde cijfer op basis van de gemaakte toetsen kan tonen op het rapport.

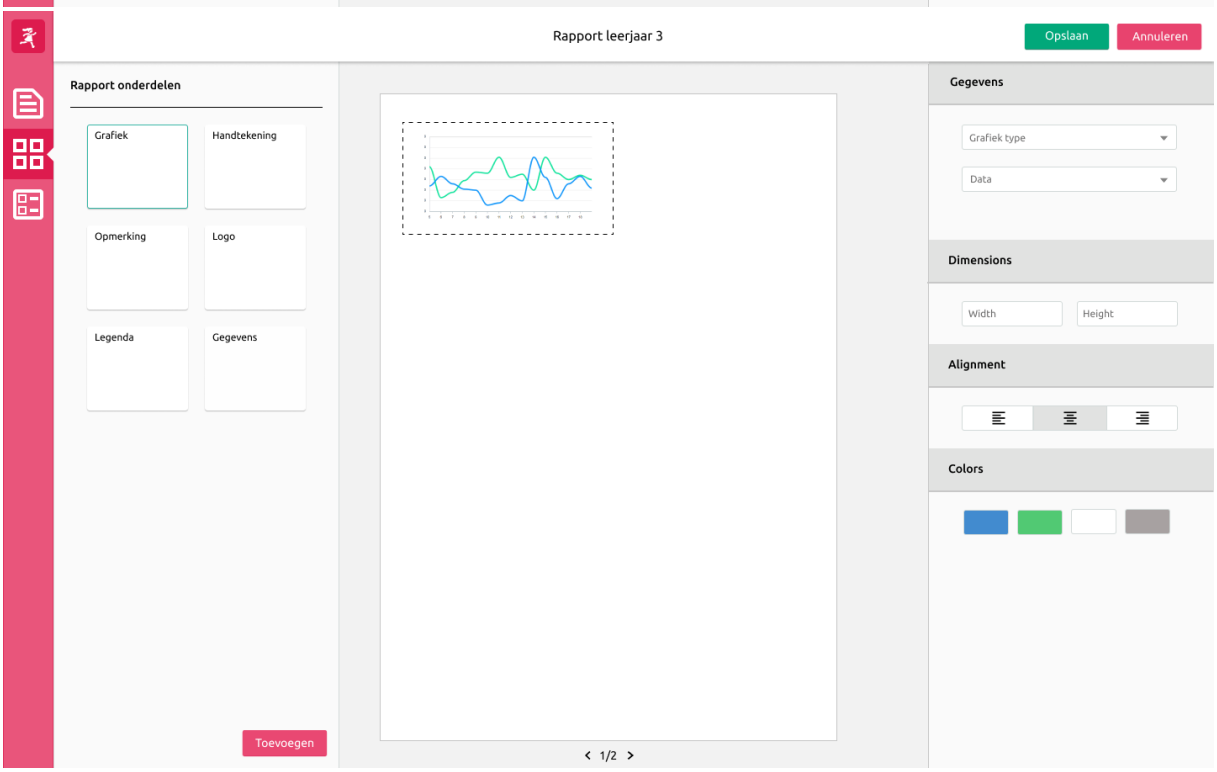
US28	Als leerkracht wil ik een opmerking plaatsen bij een subrubriek, zodat ik kan laten weten wat ik over het resultaat te zeggen heb.
US29	Als leerkracht wil ik de keuze hebben om het gemiddelde cijfer weer te geven op verschillende manieren, zodat ik de resultaten op verschillende manieren kan laten zien.
US30	Als leerkracht wil ik een rapport exporteren naar PDF, zodat ik deze naar ouders kan sturen.
US31	Als ontwikkelaar wil ik het rapport als component embedden in andere applicaties, zodat deze niet opnieuw gebouwd hoeft te worden.
US32	Als leerkracht wil ik het rapport van een specifieke leerling kunnen laden in de editor, zodat ik hier opmerkingen aan kan toevoegen.

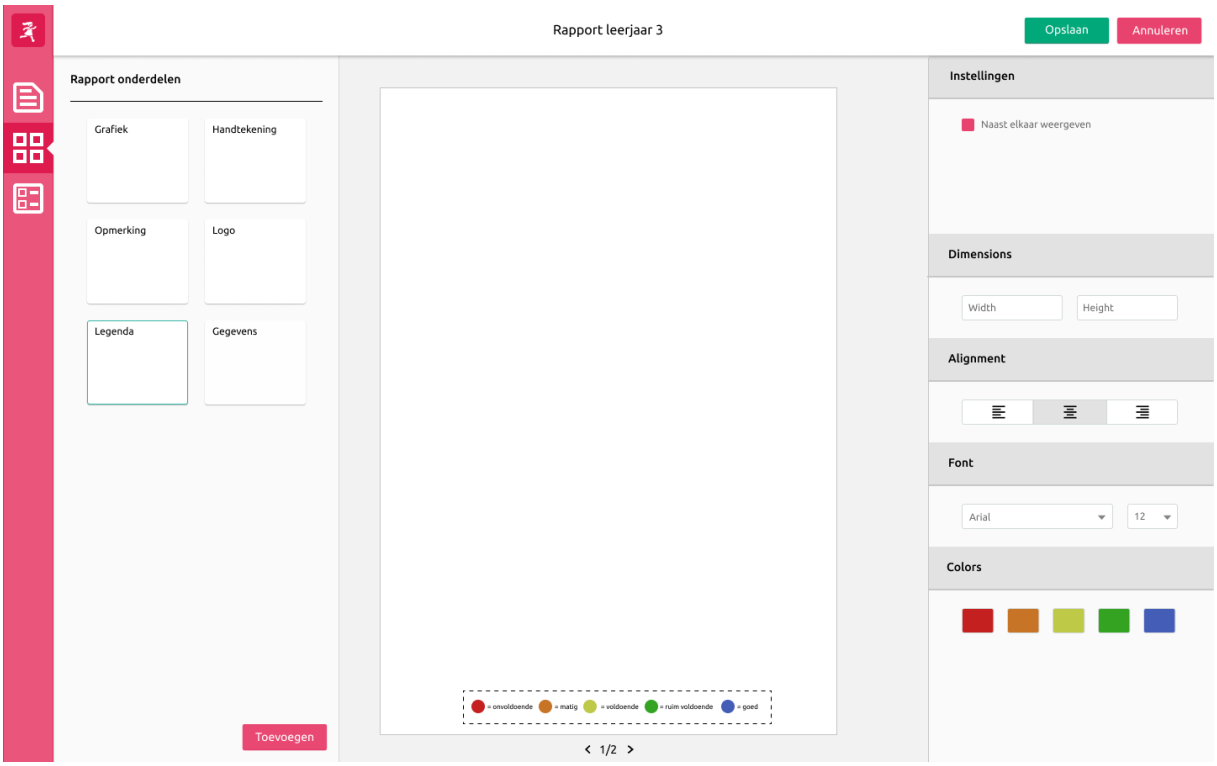
BIJLAGE G: SCHERMONTWERPEN

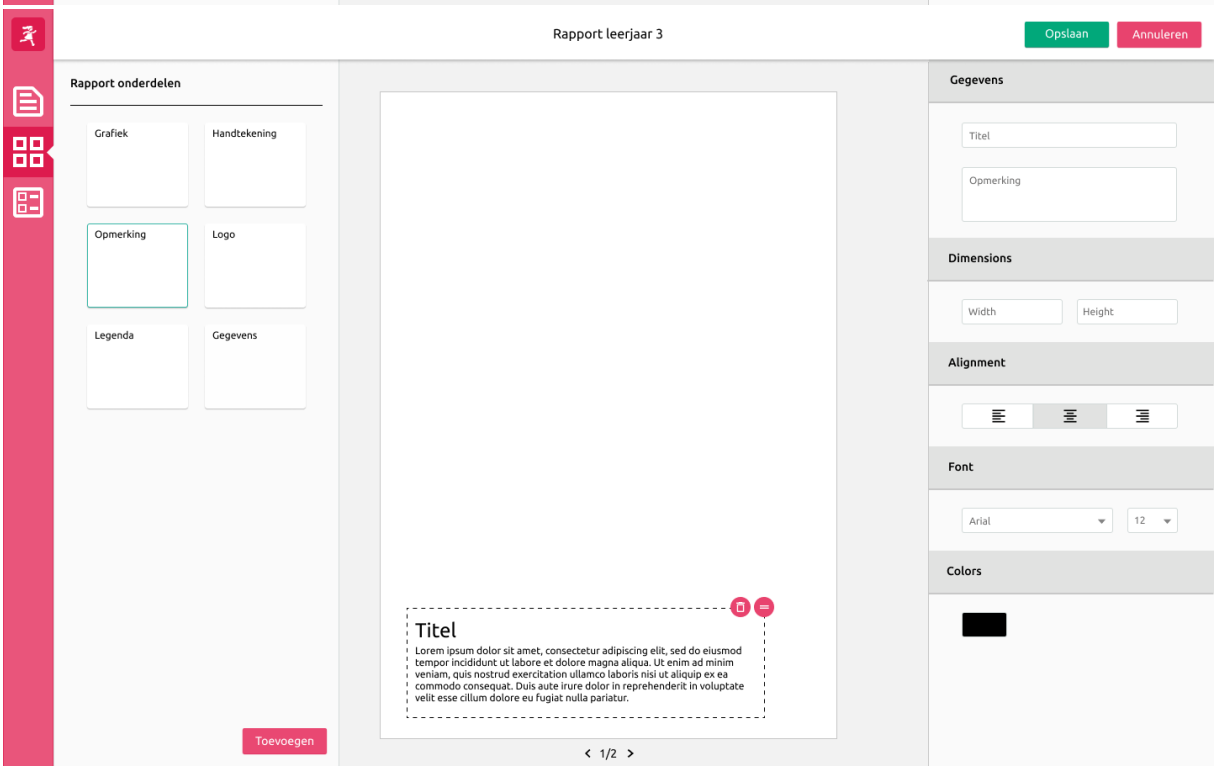


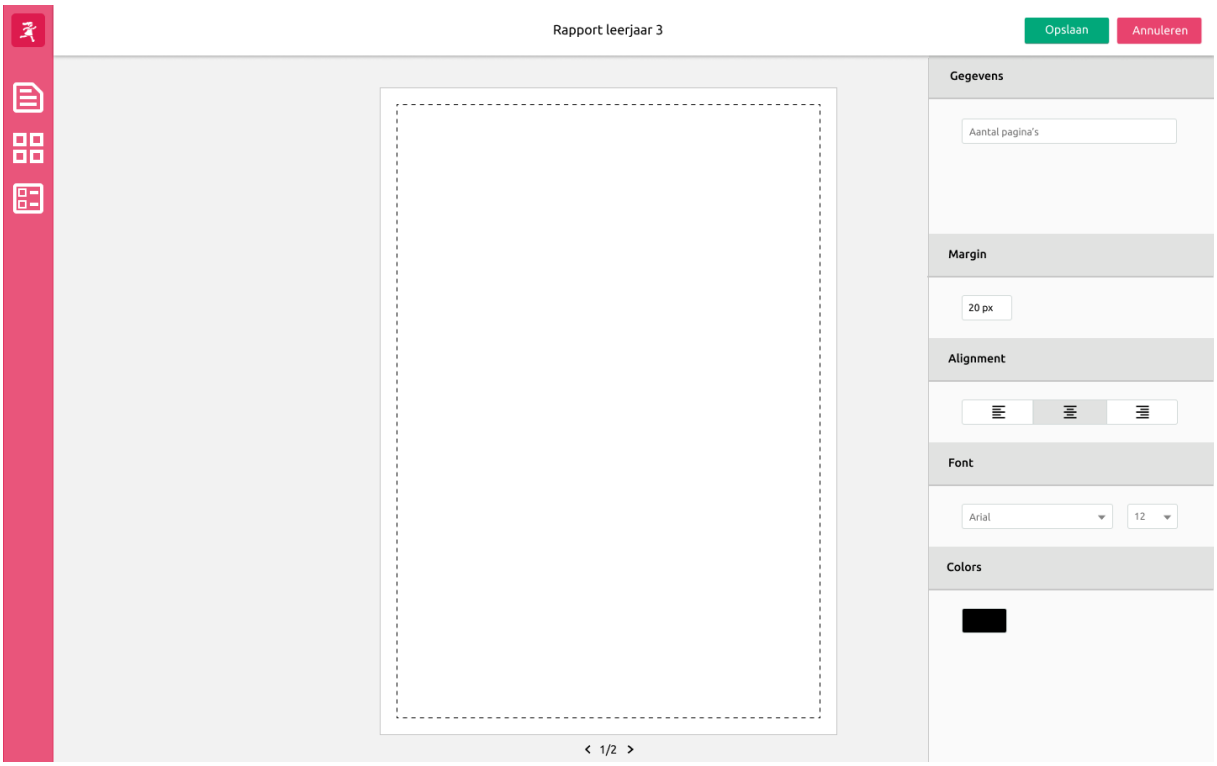


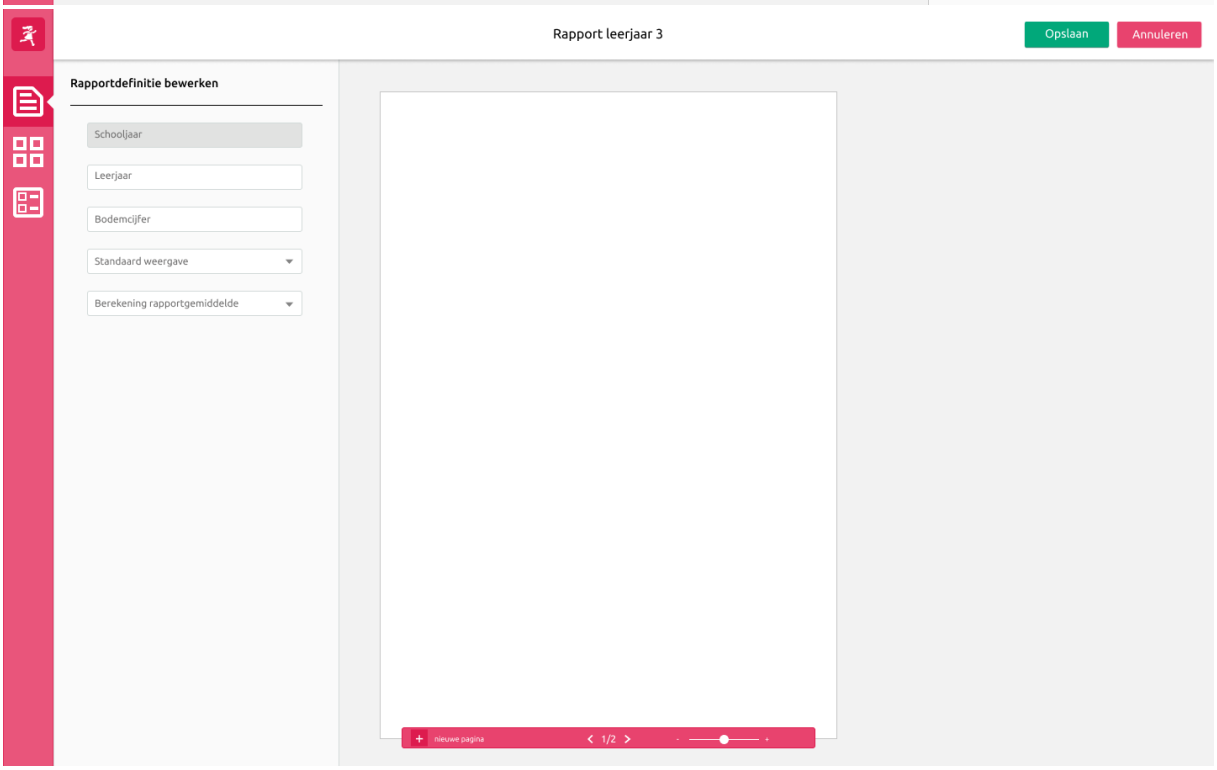

Rapport leerjaar 3
Opslaan
Annuleren
Rapport onderdelen
Grafiek
Handtekening
Opmerking
Logo
Legenda
Gegevens
Toevoegen
Test Leerling
Groep 8
Schooljaar 2020/2021
nieuwe pagina
1/2
Gegevens
Schooljaar
Leerling
Groep
Schoolnaam
Naast elkaar weergeven
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors

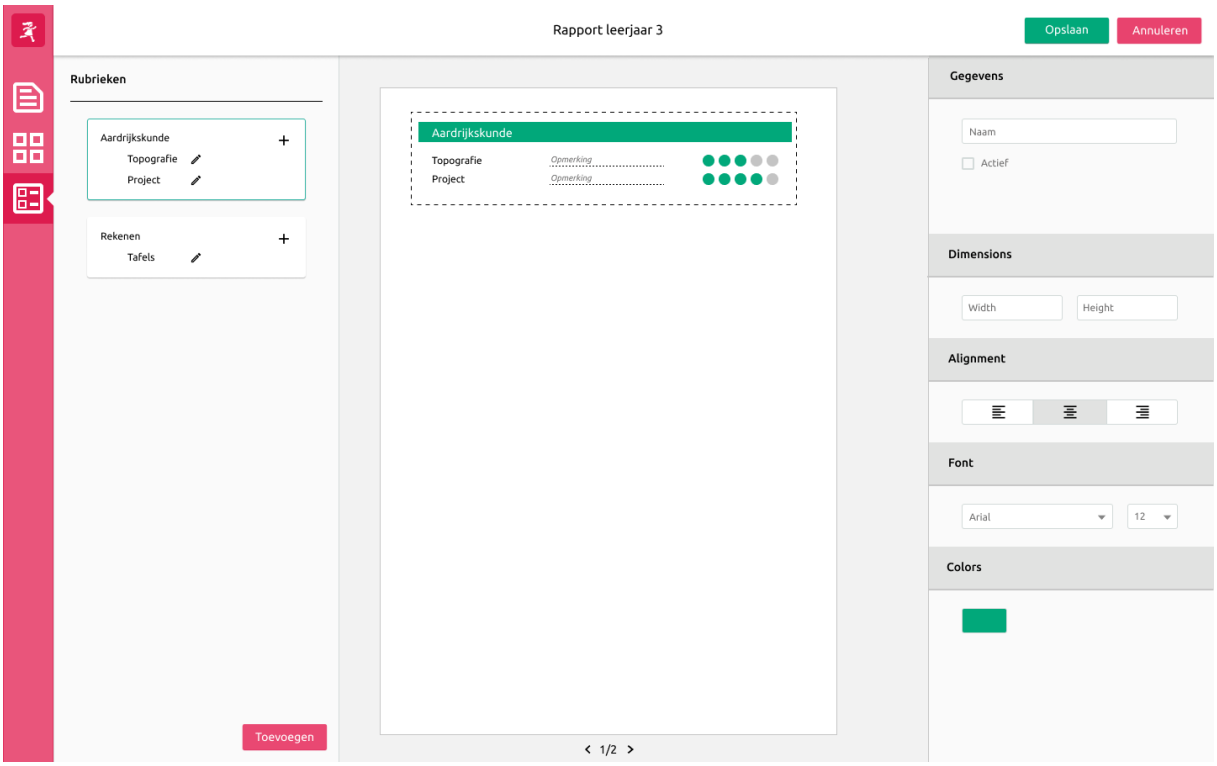

Rapport leerjaar 3
Opslaan
Annuleren
Rapport onderdelen
Grafiek
Handtekening
Opmerking
Logo
Legenda
Gegevens
Toevoegen
1/2
Gegevens
Grafiek type
Data
Dimensions
Width
Height
Alignment
Colors

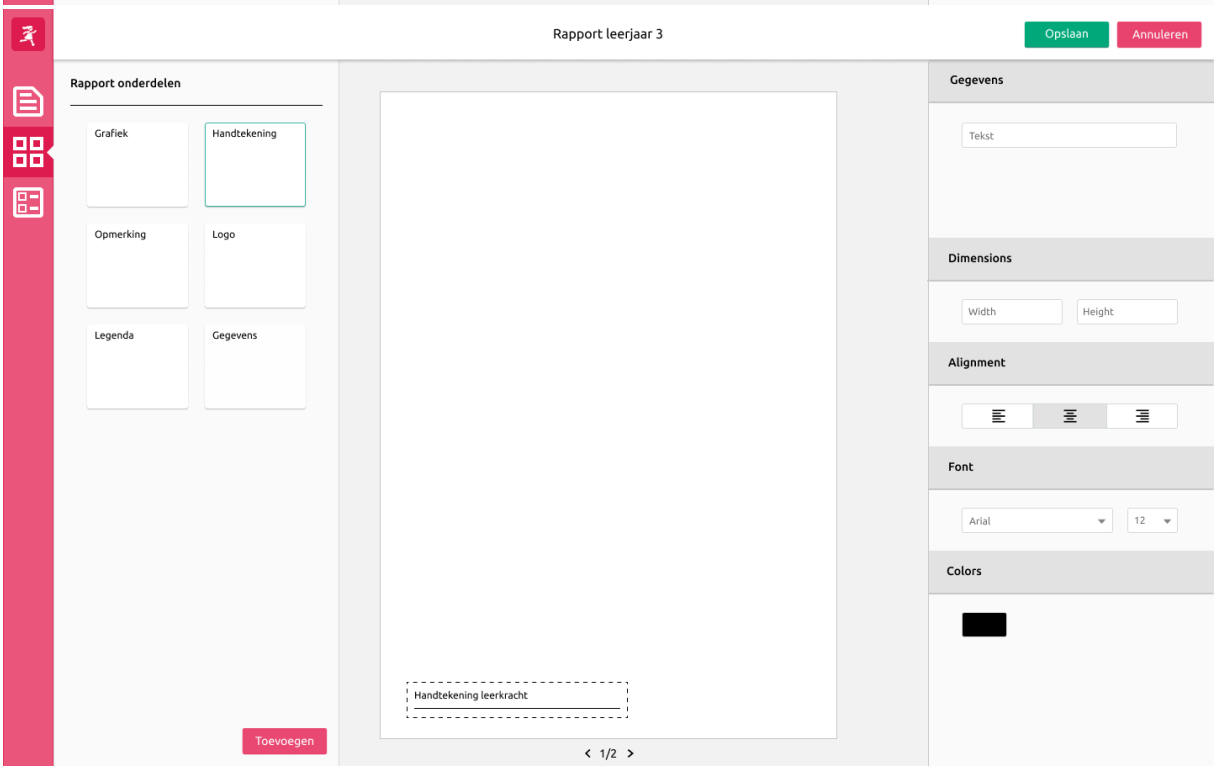

Rapport leerjaar 3
Opslaan
Annuleren
Rapport onderdelen
Grafiek
Handtekening
Opmerking
Logo
Legenda
Gegevens
Toevoegen
1/2
Instellingen
Naast elkaar weergeven
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors

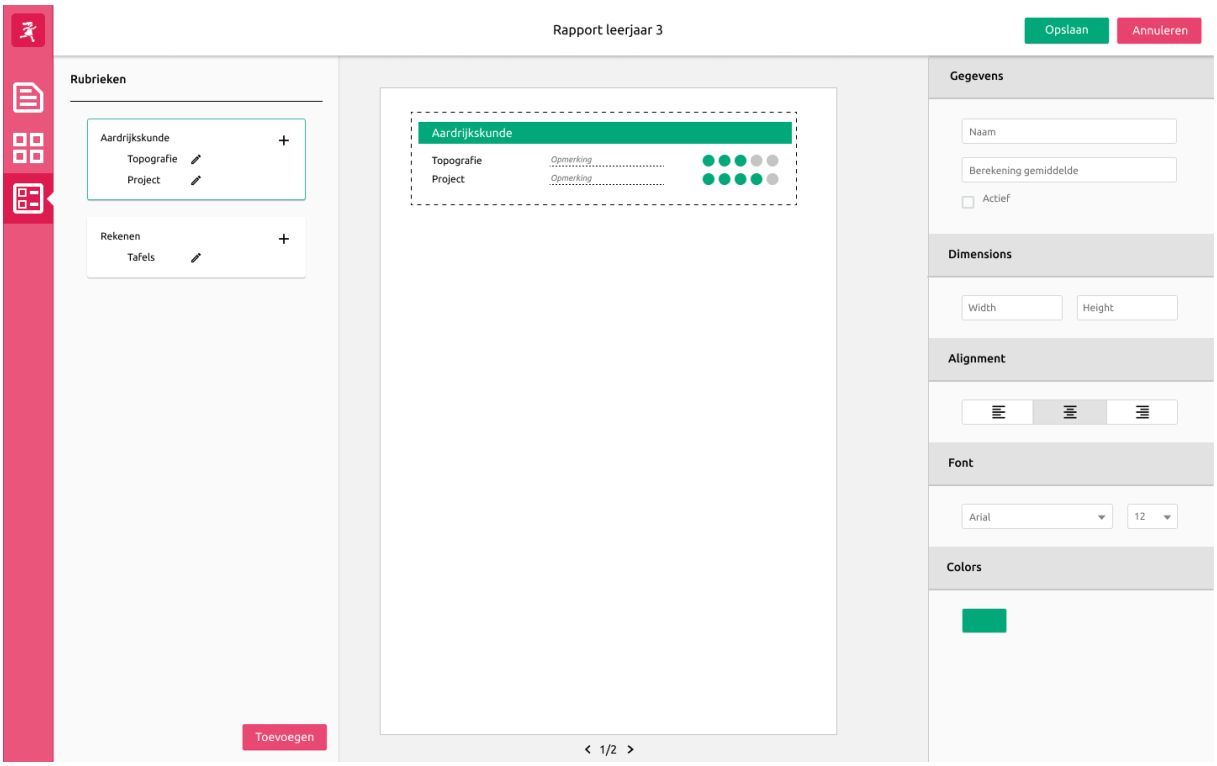

Rapport leerjaar 3
Opslaan
Annuleren
Rapport onderdelen
Grafiek
Handtekening
Opmerking
Logo
Legenda
Gegevens
Toevoegen
Titel
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
1/2
Gegevens
Titel
Opmerking
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors

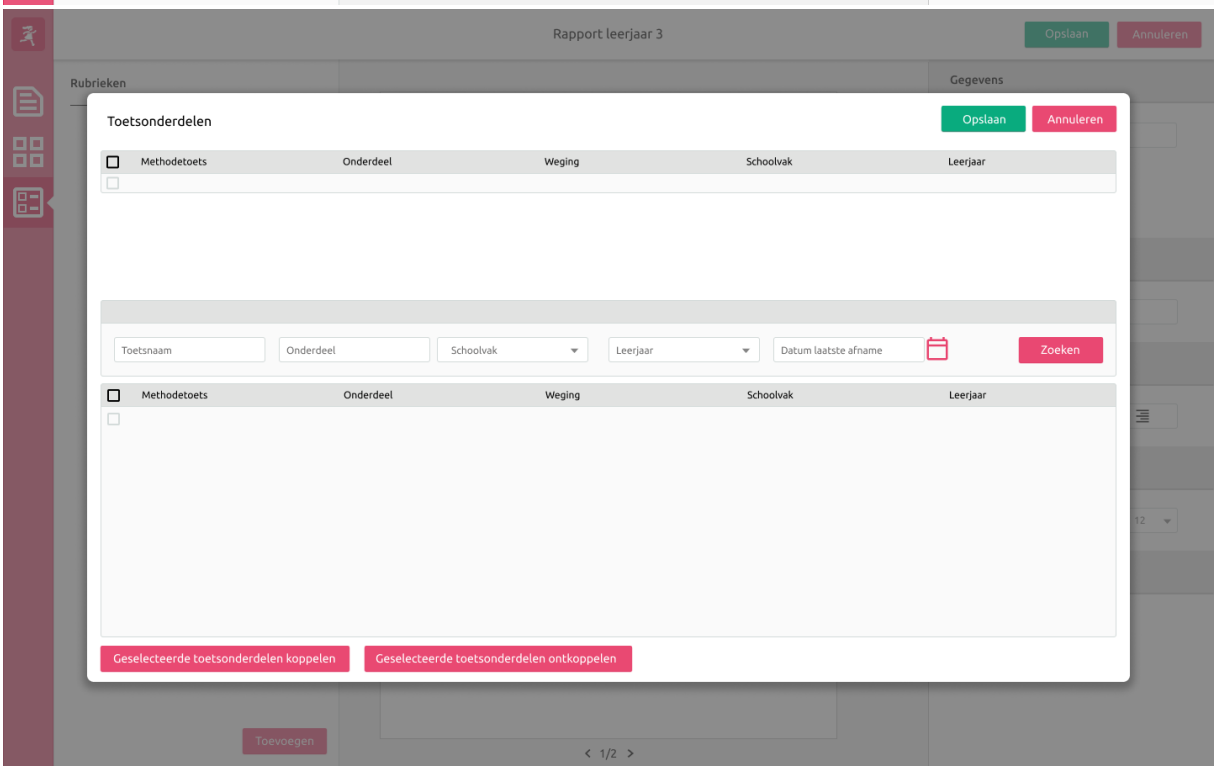

Rapport leerjaar 3
Opslaan
Annuleren
Gegevens
Aantal pagina's
Margin
20 px
Alignment
Font
Arial
12
Colors
1/2


Rapport leerjaar 3
Opslaan
Annuleren
Rapportdefinitie bewerken
Schooljaar
Leerjaar
Bodemcijfer
Standaard weergave
Berekening rapportgemiddelde
nieuwe pagina
1/2

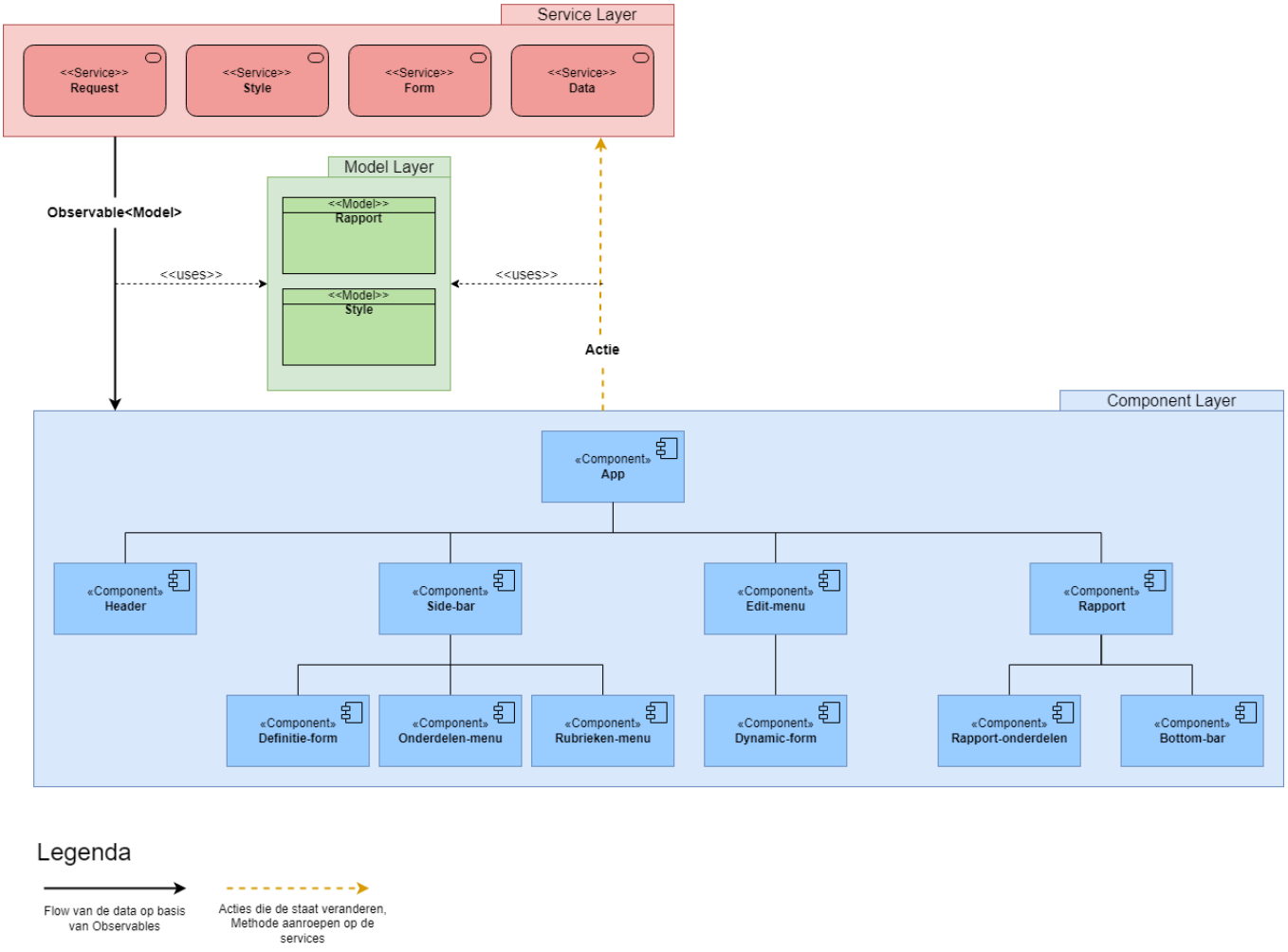

Rapport leerjaar 3
Opslaan
Annuleren
Rubrieken
Aardrijkskunde
Topografie
Project
Rekenen
Tafels
Toevoegen
Aardrijkskunde
Topografie
Project
Opmerking
Opmerking
1/2
Gegevens
Naam
Actief
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors


Rapport leerjaar 3
Opslaan
Annuleren
Rapport onderdelen
Grafiek
Handtekening
Opmerking
Logo
Legenda
Gegevens
Toevoegen
Handtekening leerkracht
1/2
Gegevens
Tekst
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors


Rapport leerjaar 3
Opslaan
Annuleren
Rubrieken
Aardrijkskunde
Topografie
Project
Rekenen
Tafels
Toevoegen
Aardrijkskunde
Topografie
Opmerking
Project
Opmerking
1/2
Gegevens
Naam
Berekening gemiddelde
Actief
Dimensions
Width
Height
Alignment
Font
Arial
12
Colors


Rapport leerjaar 3
Opslaan
Annuleren
Rubrieken
Gegevens
Toetsonderdelen
Opslaan
Annuleren
Methodetoets
Onderdeel
Weging
Schoolvak
Leerjaar
Toetsnaam
Onderdeel
Schoolvak
Leerjaar
Datum laatste afname
Zoeken
Methodetoets
Onderdeel
Weging
Schoolvak
Leerjaar
Geselecteerde toetsonderdelen koppelen
Geselecteerde toetsonderdelen ontkoppelen
Toevoegen
1/2

BIJLAGE H: ARCHITECTURE DIAGRAM

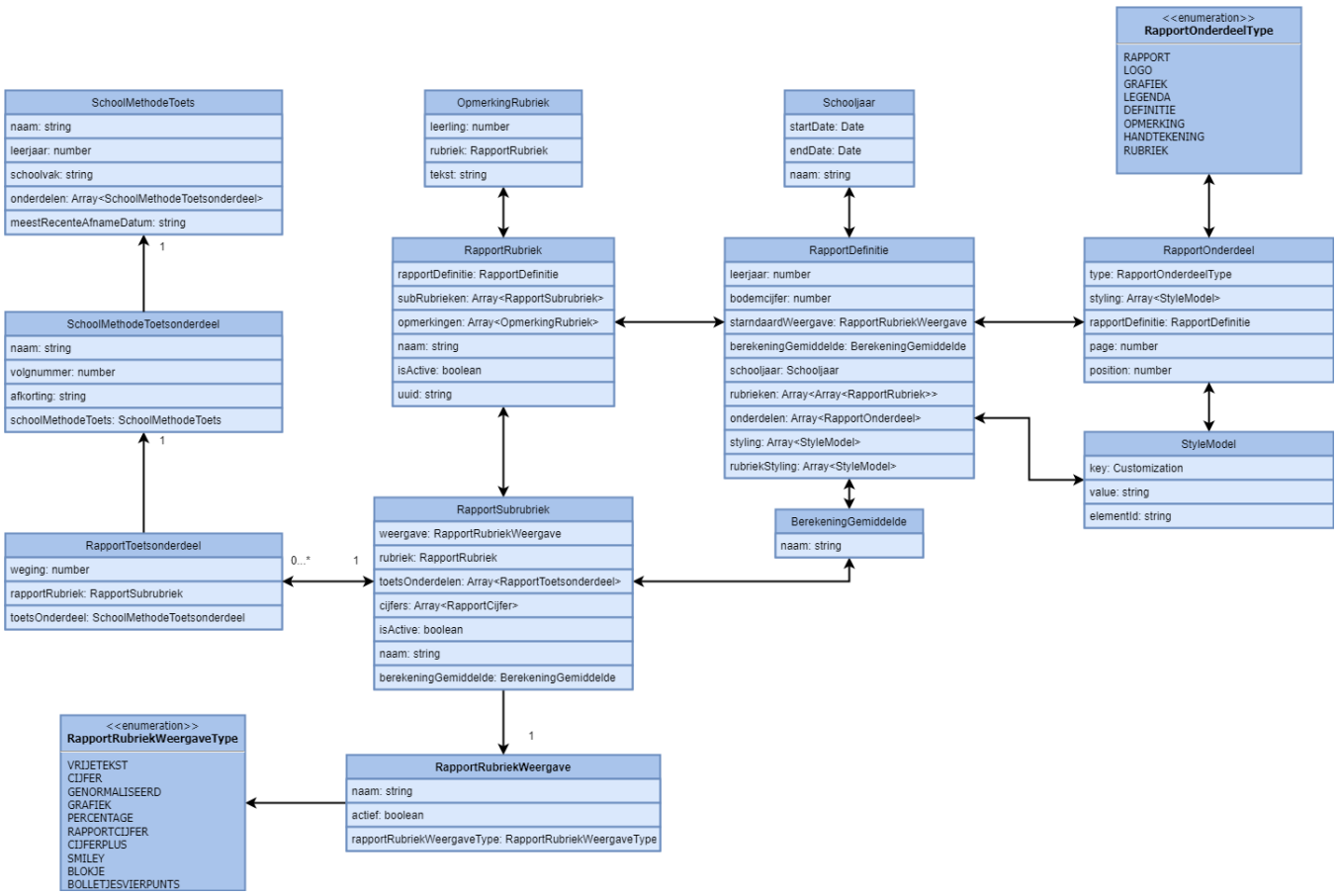


Legenda

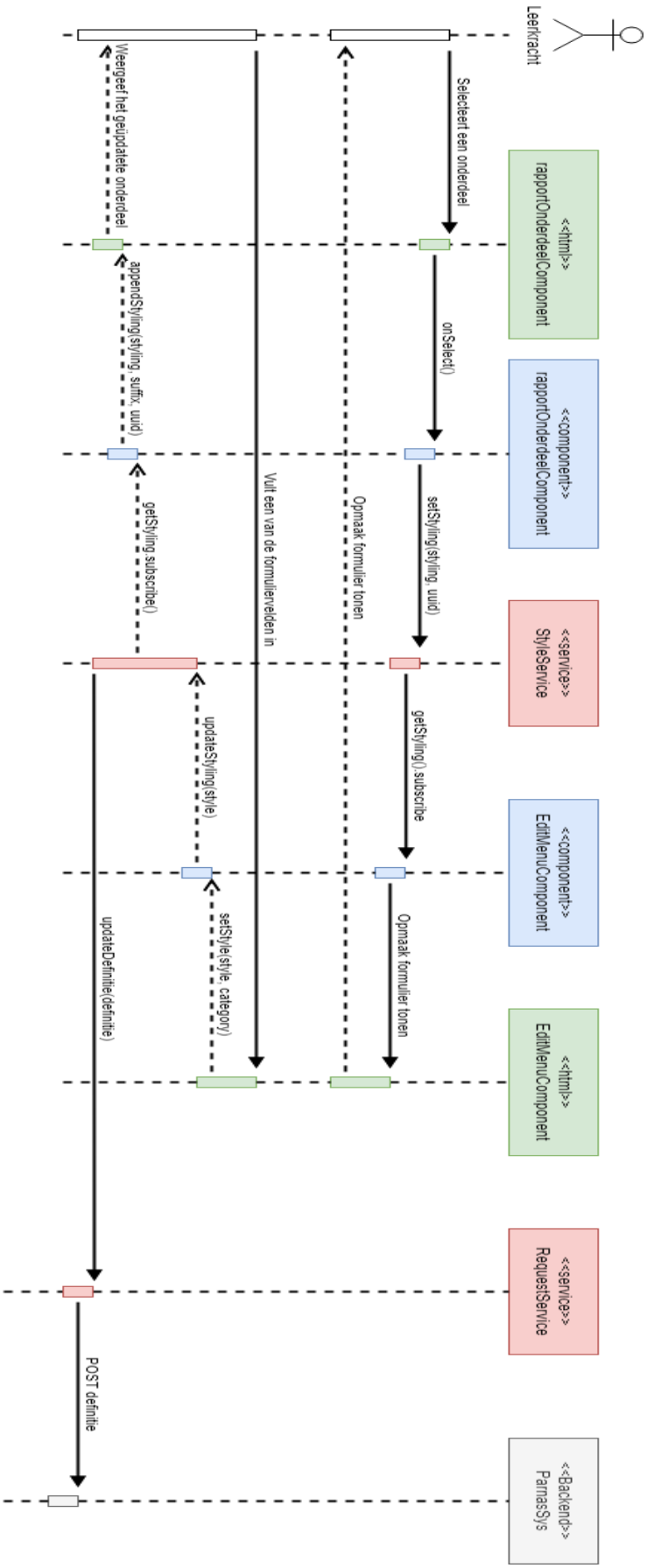
Flow van de data op basis van Observables

Acties die de staat veranderen, Methode aanroepen op de services

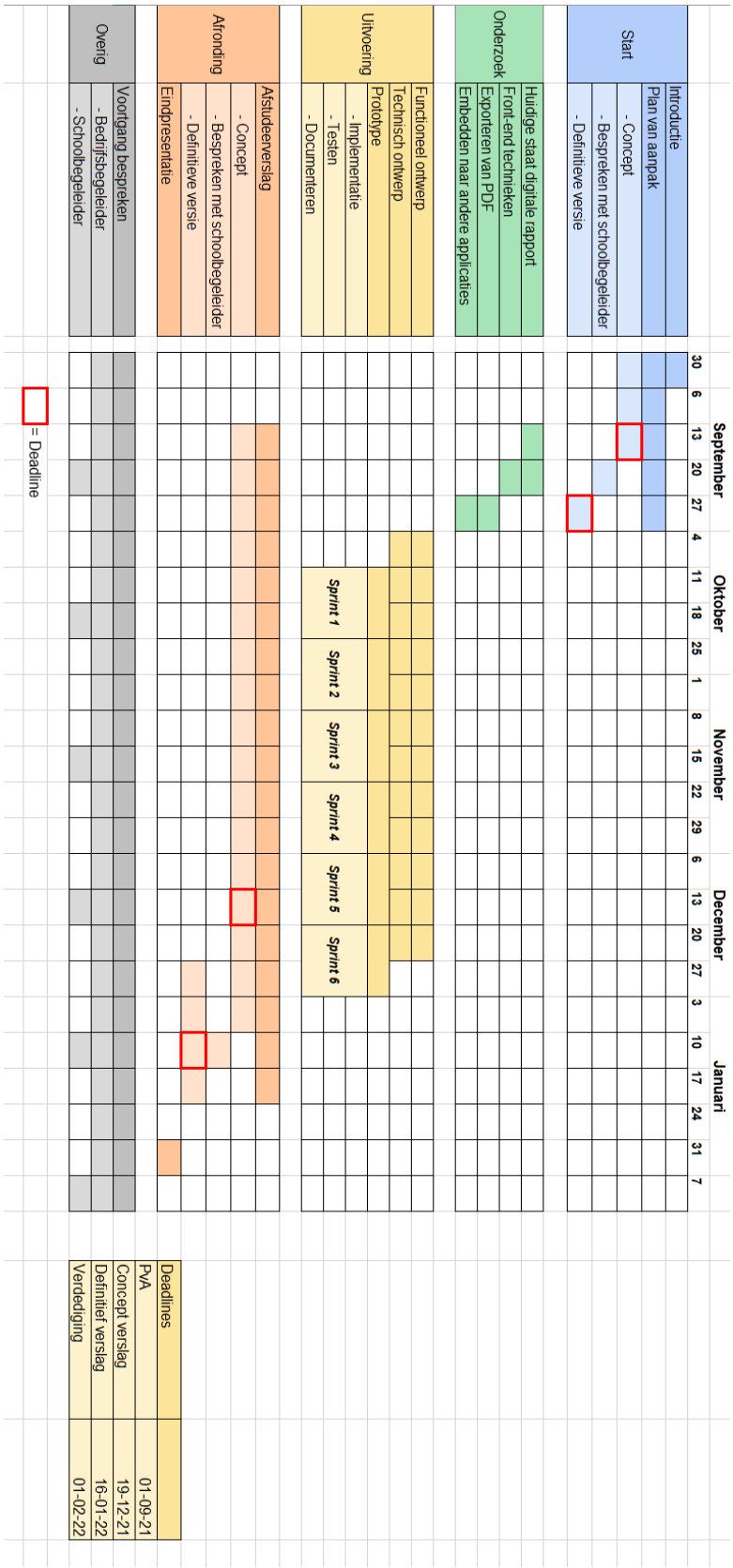
BIJLAGE I: CLASS DIAGRAM VAN DE APPLICATIE



BIJLAGE J: SEQUENCE DIAGRAM STYLEN VAN ONDERDELEN



BIJLAGE K: PLANNING



BIJLAGE L: AFGEDEKTE USER STORIES PER SCHERMONDERDEEL

Navigatiebalk
US03 - Als leerkracht wil ik de rapportdefinitie invullen, zodat het duidelijk is om welke gegevens het gaat.
US07 - Als leerkracht wil ik rapportonderdelen selecteren uit een menu, zodat ik deze op het rapport kan plaatsen.
US11 - Als leerkracht wil ik resultaten weergeven in een grafiek of diagram, zodat ik een overzicht kan laten zien van de voortgang van een leerling.
US14 - Als leerkracht wil ik een logo toevoegen aan een rapport, zodat het duidelijk is om welke school het gaat.
US16 - Als leerkracht wil ik een legenda toevoegen aan een rapport, zodat ik kan laten weten wat de normering is.
US18 - Als leerkracht wil ik een handtekeningregel toevoegen aan een rapport, zodat ik een handtekening kan zetten onder het rapport.
US20 - Als leerkracht wil ik een opmerking aan het gehele rapport toevoegen, zodat ik kan laten weten wat ik over het gehele rapport te zeggen heb.
US22 - Als leerkracht wil ik rubrieken toevoegen aan een rapport, zodat ik hiervan de resultaten kan weergeven.
US23 - Als leerkracht wil ik subrubrieken toevoegen aan een rubriek, zodat ik bepaalde resultaten van een rubriek kan weergeven.
US28 - Als leerkracht wil ik de keuze hebben om het gemiddelde cijfer weer te geven op verschillende manieren, zodat ik de resultaten op verschillende manieren kan laten zien.
US24 - Als leerkracht wil ik toetsonderdelen koppelen aan een subrubriek, zodat ik kan bepalen welke toetsen er mee tellen voor het resultaat.

Header
US05 - Als leerkracht wil ik de wijzigingen aan een rapport opslaan, zodat ik deze kan gebruiken voor later.
US06 - Als leerkracht wil ik de wijzigingen aan een rapport annuleren, zodat ik de keuze heb om een rapport niet aan te passen.

Live weergave van het rapport
US02 - Als leerkracht wil ik handmatig de positie van een rapportonderdeel op het rapport veranderen, zodat ik zelf de indeling kan bepalen.
US08 - Als leerkracht wil ik een live weergave zien van het rapport, zodat ik weet hoe het eruit komt te zien.
US09 - Als leerkracht wil ik rapportonderdelen verwijderen van een rapport, zodat ik het rapport een andere indeling kan geven.
US27 - Als leerkracht wil ik een opmerking plaatsen bij een subrubriek, zodat ik kan laten weten wat ik over het resultaat te zeggen heb.

Bottom-toolbar
US10 - Als leerkracht wil ik pagina's kunnen toevoegen aan een rapport, zodat ik grotere rapporten kan maken.

Opmaakmenu
US01 - Als leerkracht wil ik de algemene opmaak van een pagina aanpassen, zodat ik zelf kan kiezen hoe deze eruit komt te zien.
US12 - Als leerkracht wil ik de opmaak van een grafiek aanpassen, zodat deze overeenkomt met de rest van het rapport.
US13 - Als leerkracht wil ik het type grafiek aanpassen, zodat ik de resultaten op een andere manier kan weergeven.

US15 - Als leerkracht wil ik de opmaak van een logo aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US17 - Als leerkracht wil ik de opmaak van een legenda aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US19 - Als leerkracht wil ik de opmaak van een handtekeningregel aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.
US21 - Als leerkracht wil ik een opmerking kunnen wijzigen, zodat ik de opmerking kan aanpassen.
US25 - Als leerkracht wil ik de opmaak van een rubriek aanpassen, zodat ik meer vrijheid heb over het uiterlijk van het rapport.