Software ontwikkeling
en professionele vaardigheden

2de bachelorjaar informatica

Visual Programming IDE: Analyseverslag

Groep 5:
Pim Goffinghs (1849269),
Wout Oben (1745605),
Joep Stevens (1848586)

Begeleider
Joris Herbots

Coördinerend verantwoordelijke
prof. dr. Wim Lamotte

Jaar 2019-2020

Inhoudsopgave

1 Diepgaande beschrijving van het onderwerp

De CoderDojo is een non-profit beweging die evenementen organiseren waarbij kinderen aangezet worden om te leren programmeren. Anaïs Ools, één van de Hasselt CoderDojo begeleiders, heeft ons de opdracht gegeven om een 'Visual Programming IDE' te creëren. Meer specifiek, een IDE waarbij kinderen van 8 tot 12 jaar op een interactieve manier de denkwijzen en de logica achter het programmeren kunnen aanleren. Dit doen ze dan door middel van blokken te slepen en te verbinden op een semi-oneindig canvas. Door de jonge leeftijd van het doelpubliek is een simpel, maar aantrekkelijke en uitnodigende lay-out dus zeker een must voor het programma. Dit proberen wij te creëren door de plaatsen waar blokken kunnen connecteren voor te stellen als puzzelstukken die in elkaar passen. Zo zal visueel ook duidelijk zijn welke blokken connecteerbaar zijn met elkaar en welke niet. Iedere blok heeft zijn eigen betekenis in de programmeer-syntax. De functionaliteiten die wij zullen voorzien, zijn:

1. Lussen: for- en while-loops
2. Conditionele operaties: if en else-blokken
3. Logica operatoren: && (AND), || (OR), ! (NOT)
4. Wiskundige operatoren: +, -, *, /, ==, ¡, ¿
5. String operaties: Lengte van een string berekenen, strings samenvoegen...
6. Variabelen: Een blok waaraan men een waarde van type string, boolean of integer aan kan koppelen.

Ook zal er een start-blok aangemaakt worden waaraan de gebruiker de code kan connecteren die uitgevoerd moet worden bij de start van het programma. Het starten en stoppen van het programma gebeurt met behulp van 2 extra knoppen. Bij het drukken op de start-knop zal de code geconnecteerd met de start-blok uitgevoerd worden. Bij het drukken op de stop-knop zal het lopende programma (als er een programma loopt) stoppen met uitvoeren. De output van het programma zal gebeuren aan de hand van een apart venster waar de output op afgebeeld zal worden in de vorm van tekst, afbeeldingen en geluiden. Doordat er maar een gelimiteerd aantal laptops beschikbaar zijn op de evenementen van de CoderDojo, en omdat de kinderen niet altijd dezelfde laptop terug krijgen, zal dit project web-based worden. Zo kunnen kinderen vanop elke laptop de applicatie gebruiken. Het opslaan en inladen van projecten zal gebeuren via een systeem met accounts, het aanmaken van een account zal niet verplicht zijn, enkel als men nieuwe projecten wilt opslaan of wilt verder werken aan oudere projecten. Dit hele systeem zal draaien op een web-server die de CoderDojo zelf voorziet op hun evenement, hierdoor zal de applicatie enkel beschikbaar zijn op het evenement van de CoderDojo. Om de kinderen van kleine opdrachten te kunnen voorzien zou een API-ondersteuning zeer handig zijn voor de CoderDojo. Dit gaan we implementeren door verschillende blokken te voorzien voor elke HTTP request method (denk aan GET, POST, DELETE, etc...). Deze blokken kunnen dan gemakkelijk gebruikt worden in samenwerking met een RESTful API die kan opgezet worden door CoderDojo. Omdat er vaak kinderen zonder enige programmeerervaring zijn, zullen we ook tutorials implementeren. Deze zullen duidelijk aanduiden wat de basisblokken doen en hoe ze van deze een programma kunnen maken. Deze tutorials zullen gemaakt worden in een apart venster. Zo zal men een step-by-step guide kunnen volgen op eigen tempo en tegelijkertijd alles uittesten in de applicatie. Om het vinden van de correcte blokken makkelijker te maken, gaan we de meest gebruikte (if-else, for, while...) en basis-blokken apart onderscheiden van de rest van de blokken. Op de user-interface zal zich links een menu bevinden, waar de gebruikers blokken uit kunnen kiezen om te plaatsen op het canvas. De opdrachtgever liet blijken dat een menu waar eerst de categorie van een blok geopend moest worden, vooraleer een blok gekozen kan worden, vaak onduidelijk was voor kinderen. Om toch een duidelijke structuur in de blokken te houden, zullen wij de meest voorkomende blokken opsommen vooraleer we blokken opsommen in categorieën. Zo kunnen gebruikers zonder ervaring snel de basisblokken terugvinden, terwijl gebruikers met meer ervaring meer geavanceerde blokken per categorie kunnen zoeken. Zo zal het maken van een klein programma zeer simpel worden, zelfs voor degene zonder enige ervaring. Doordat blokken op een oneindig canvas wel eens verloren kunnen geraken, zal er een

klein venster voorzien worden waar een overzicht van het canvas te zien zal zijn. Verloren blokken terugvinden zal op deze manier eenvoudiger zijn. Ten slotte zal het delen van projecten met andere gebruikers ook een mogelijkheid zijn. Wanneer men een project deelt, wordt een tijdelijke kopie van het project gemaakt. Deze kopie wordt opgeslagen in een aparte database, samen met een 5-cijferige code. Een andere gebruiker kan de kopie dan inladen met behulp van deze code. Deze andere gebruiker kan dan werken op zijn eigen kopie van dit project en eventueel opslaan onder zijn eigen account.
Eens dit allemaal werkend is, zal er worden gewerkt aan een aantal extra features. Deze features worden als extra gezien, doordat het suggesties van ons waren en de opdrachtgever er dus minder belang aan hecht. Zo zal er dan een functie-blok gemaakt worden, waarmee men stukken code kan groeperen, deze zal men dan meerdere malen kunnen oproepen in de code. Ook de implementatie van lijsten en dictionaries zal als extra feature beschouwd worden. Als laatste extra zouden we ook de ondersteuning van de Engelse taal kunnen implementeren, want de user-interface zal standaard in het Nederlands staan.

2 Gerelateerd werk: software, libraries en literatuur

2.1 Software

2.1.1 Scratch[1]

Scratch is waarschijnlijk de software die het dichtst aansluit bij ons project. Het is een browser-gebaseerde programmeertaal waarmee men programma's maakt door blokken op een semi-oneindig canvas te verbinden met elkaar. Opslaan en inladen van projecten gebeurt aan de hand van accounts. Ook ondersteunt het het delen van projecten doordat gemaakte projecten kopieerbaar zijn door iedere andere gebruiker. Scratch heeft een aantal ingebouwde extensies (bijvoorbeeld een extensie die je micro:bit-programma's laat aanmaken), ook dit is een feature die wij gaan implementeren. Output van deze programma's wordt altijd voorgesteld door afbeeldingen die iets doen. Dit nemen we niet mee in ons project, aangezien wij een meer console-based output zullen hebben. De indeling van de blokken gaan wij ook anders doen, doordat de opdrachtgever heeft laten blijken dat zij vond dat de manier waarop Scratch dit deed redelijk onduidelijk was voor de doelgroep. Ten slotte voorziet Scratch ook tutorials voor gebruikers zonder ervaring. Het opent een klein nieuw venster waarin foto's van iedere stap zichtbaar zijn, zo kan men tegelijkertijd de tutorial volgen en de stappen zelf uitvoeren in zijn/haar eigen project. Aangezien de doelgroep voor ons project kinderen zijn, is dit ook zeer belangrijk in ons programma.

2.1.2 MIT App inventor for android[2]

App inventor for android is een browser-gebaseerd programma waarmee apps (voor een android-device) op een zeer simpele manier aangemaakt kunnen worden. Ook hier wordt de code geschreven door blokken tegen elkaar te plaatsen. De output kan men testen door op een android-device de 'MIT Companion'-app te downloaden en deze te linken met het webproject. Zo kan men de app rechtstreeks op het device testen. Doordat onze output totaal verschillend is, zullen wij dit niet meenemen naar ons project. Een leuke feature van app inventor is dat men beschikt over een "backpack". In deze backpack kan men delen van code opslaan. De backpack is bruikbaar in ieder project van de user, zo kan men delen code van project tot project kopiëren. In ons project zullen we ook code moeten kunnen delen, maar dit ook van één user naar een andere. Hierdoor kunnen wij dit niet op deze manier doen.

2.1.3 Hopscotch[3]

Hopscotch is een app voor iOS-devices waarmee men zelf apps kan maken. Het is vergelijkbaar met 'app inventor for android', maar dan voor iOS en in mobile-app vorm. Het doelpubliek voor Hopscotch zijn kinderen van 7 tot 13 jaar, dit is zeer vergelijkbaar met ons doelpubliek. Hierdoor ziet de app er ook zeer simpel en kindvriendelijk uit, iets wat wij met ons project ook willen bereiken. Hopscotch werkt echter eerder object-georiënteerd, er moet eerst een voorwerp aangemaakt worden en dan wordt er beschreven wanneer dit object iets moet doen en wat het

precies moet doen. Doordat wij een console-based output hebben, gaan wij dit niet volledig zo kunnen doen. Wij gaan wel op aanvraag van de opdrachtgever dit implementeren, zodat de mogelijkheid er wel is.

2.1.4 Micro:bit[4]

De BBC micro:bit is een kleine programmeerbare microcomputer. Door de simpele makeCode-editor is het programmeren van deze microcomputer zeer eenvoudig, zelfs voor degene zonder programmeerervaring. De online editor laat je programma's maken door blokken te verbinden op een semi-oneindig canvas, maar geeft je ook de optie om in javascript een programma te typen. Iets wat we niet meenemen naar ons project is het feit dat alle blokken onderverdeeld staan per categorie. Je moet eerst de juiste categorie aanduiden als je een bepaalde blok wilt gebruiken, dit was iets dat onze opdrachtgever had laten blijken dat zij niet goed vond. Iets wat we wel meenemen naar ons project is dat de meest gebruikte en simpelste blokken vanboven staan en er een onderverdeling 'Advanced' is. Hier kan je op klikken als je al wat meer gevorderd bent en meer geavanceerde programma's wilt maken. De output is zeer specifiek, doordat het bedoelt is om een micro:bit microcomputer aan te sturen. Deze gaan we dus ook niet meenemen naar ons project.

2.1.5 Open Roberta[5][6]

Open Roberta is een Duits intuïtief leerprogramma met als doel programma's te leren programmeren om robots te besturen. Het is zeer fel te vergelijken met micro:bit, het enige verschil is dat men als output nu verschillende robots kan besturen in plaats van 1 microcomputer. Dit is opnieuw niet bruikbaar in ons project, aangezien wij niet met robots werken.

2.2 Libraries

2.2.1 Blockly[7]

Blockly is een door Google gemaakte library die een visuele code editor aan een website toevoegt. Het voegt een semi-oneindig canvas in met een voorgedefinieerde 'toolbox' die bepaalde invoegbare blokken presenteert. De volgorde van de blokken en welke blokken precies in de toolbox staan, kan de developer zelf kiezen. De developer kan zelf ook nieuwe blokken creëren. Het aaneensluiten van de blokken in het canvas en deze code vertalen naar javascript gebeurt allemaal automatisch door de library. Importeren en exporteren van projecten gaat simpel van en naar een XML-bestand. Bij gebruik van de Google app engine (betalend) kunnen projecten gewoon op de cloud van Google opgeslagen worden en van daaruit ook ingeladen. Een voordeel hieraan is dat men projecten simpelweg kan delen met behulp van links. Projecten kunnen ook simpelweg op de local storage opgeslagen worden. De Blockly-library heeft ook een goede documentatie waar duidelijk in beschreven staat hoe de library juist geïmplementeerd moet worden. Doordat de Blockly-library zelf al veel van het project implementeert, is het gebruik ervan niet toegestaan. Het delen van een project via een link gaan we echter op een gelijkaardige manier proberen te doen. Blockly creëerde een nieuw kopie van een project iedere keer men het wou delen, zo kan men projecten delen zonder dat anderen iets kunnen wijzigen aan het project.

2.2.2 Scratch blocks[8]

Scratch blocks is gebaseerd op Blockly en is gemaakt door het team achter Scratch. Het grootste voordeel aan Scratch blocks in vergelijking met Blockly is het feit dat het blokken ook horizontaal laat verbinden. Zij, het team achter Scratch, beweren dat dit makkelijker zou zijn voor gebruikers zonder programmeerervaring en voor mobile-users. Doordat Scratch blocks gebaseerd is op Blockly, is het gebruik ervan niet toegestaan bij dit project. Het gebruik van horizontale blokken vinden wij persoonlijk minder overzichtelijk en zal dus niet worden geïmplementeerd in ons project.

2.2.3 Microsoft MakeCode/PXT[9]

Microsoft MakeCode is gebaseerd op de Microsoft PXT-library. Microsoft MakeCode is de naam voor de user-face editors (de eerder aangehaalde Micro:bit-editor is bijvoorbeeld een Microsoft

MakeCode-editor) en Microsoft PXT is de naam van de library gebruikt in al de GitHub sources van deze editors. De main features van de PXT-library zijn: een op Blockly gebaseerde code editor met converter naar een tekstformaat, een Monaco code editor die Visual Studio Code ondersteunt, extensibility support om nieuwe blokken te definiëren in TypeScript, een machine code emitter voor een ARM Thumb machine en een package manager voor de command-line. Deze features zijn niet direct een meerwaarde voor ons project, waardoor Microsoft PXT een beetje 'overkill' gaat zijn voor ons project.

2.2.4 Droplet[10]

Droplet probeert een tussenweg te creëren tussen het programmeren met blokken en tekst. Het laat toe om gemaakte programma's in te laden en deze om te zetten naar code met blokken, maar ook om rechtstreeks code met blokken om te zetten in een gewenste taal. Om dit duidelijk te maken, zorgt Droplet er voor dat de blokkencode meer op tekstcode lijkt. Hiermee bedoelen we dat de blokken niet op een semi-oneindig canvas geplaatst kunnen worden, maar enkel in een editor met lijnen en hun lijnnummer (denk aan een standaard text-editor). Doordat de opdrachtgever gevraagd heeft voor een visual programming IDE waar men blokken moet plaatsen op een semi-oneindig canvas kunnen we Droplet niet gebruiken voor ons project.

2.2.5 Snap![11]

Snap! is eerder een volledige applicatie dan een library. Het is een programma dat gebaseerd is op de werking van Scratch. We noteren het toch onder libraries doordat de source-code volledig open-source is en het dus even goed gebruikt mag worden als library voor een project. Buiten het front-end 'block-editor'-gedeelte heeft het al een hele backend-source voor het opslaan en inladen van projecten en het aanmaken van accounts. Deze maakt het ook mogelijk om projecten simpel te delen via een link, wat al één van onze vereisten is. Een nadeel van Snap! is dat het werkt met dezelfde output als Scratch (Afbeeldingen die men laat bewegen) en deze output is niet geschikt voor ons project.

2.2.6 Infinite canvas[12]

Voor het aanmaken van een semi-oneindig canvas bestaan er reeds een aantal API's. Eén van de simpelere manieren blijft echter gewoon het gebruik van HTML-canvas. Als men via JavaScript op het canvas objecten aanmaakt met een positie relatief ten opzichte van een offset, is dit implementeerbaar met een paar simpele JavaScript functies. In ons project gaan wij werken met een div voor ons canvas waar we een translatie aan gaan geven met JavaScript en CSS.

2.2.7 Drag and drop[13]

Voor drag and drop features op het canvas kan men de events in Vue gebruiken. HTML heeft tegenwoordig ook drag and drop features, deze zijn ook handig, maar bij deze features is het moeilijk om te limiteren op welke plaats blokken nu juist wel of niet gedropt mogen worden. Met Vue events en event handlers is dit zeer simpel te implementeren en te limiteren.

2.2.8 Web Workers[14]

Bij het uitvoeren van scripts in een HTML pagina, zal de webpagina niet meer reageren totdat het script klaar is met uitvoeren. Een Web Worker is een script dat op de achtergrond uitgevoerd wordt, waardoor de main pagina bruikbaar blijft. De output van ons project zal pas berekend worden vanaf het moment dat de start-knop ingeduwd wordt. Het berekenen van de output kan soms lang duren, zo kan men in een zeer lange loop terecht komen. Ondanks de handige functionaliteit van web workers, zou dit een beetje 'overkill' zijn voor de relatief kleine programma's die gemaakt zullen worden. Daardoor hebben we besloten om er geen gebruik van te maken.

2.3 Literatuur

Bij het zoeken naar gerelateerde literatuur hebben we geen vermeldenswaardige boeken of papers gevonden.

3 Evaluatiecriteria

In de volgende sectie zullen de verschillende features van de applicatie besproken worden. Deze worden opgedeeld in functionele en niet-functionele vereisten. Achteraf worden ook de extra features besproken die geïmplementeerd zullen worden eens alle functionele vereisten geïmplementeerd zijn. Een afgewerkt project zou al deze features moeten bevatten.

3.1 Functionele Vereisten

De functionele vereisten van een programma geven het gewenst gedrag weer. In deze subsectie gaan we alle functionele vereisten bespreken op volgorde volgens hoogste prioriteit.

High priority:

1. Oneindig canvas
2. Blokken lijst
3. Basis blokken
4. Start- en stopknop
5. Output via console

Medium priority:

1. Opslaan en laden op webserver met accounts
2. Api ondersteuning (GET)
3. Tutorials
4. Output en input via afbeeldingen

Low priority:

1. Delen van projecten
2. String operaties
3. Variabelen
4. Api ondersteuning (POST, PUT , DELETE)
5. Undo voor acties

3.1.1 Oneindig canvas

Onze IDE heeft een oneindig canvas waarop de gebruiker zijn programma kan maken. De gebruiker moet de volgende dingen kunnen met dit canvas:

1. Leeg oneindig canvas kunnen aanmaken.
   De gebruiker moet een canvas kunnen aanmaken. Na het aanmaken moet er een leeg canvas zichtbaar zijn voor de gebruiker.
2. Blokken kunnen zetten op het canvas.
   De gebruiker moet nieuwe blokken kunnen zetten op het canvas.
3. Blokken kunnen slepen op het canvas.
   Geplaatste blokken moeten met de muis bewogen kunnen worden op het canvas.

De blokken die tijdens het implementeren van het canvas gebruikt worden, zullen simpele rechthoeken zijn totdat de basis blokken zijn geïmplementeerd (zie 3.1.3).

3.1.2 Blokken lijst

We zetten de blokken die gebruikt kunnen worden in een lijst naast het canvas. De blokken in deze lijst kunnen door de gebruiker op het canvas worden gezet. Deze lijst zal meer blokken bevatten dan de grootte van het scherm. Daarom zal er ook een scrollbar moeten zijn. De blokken staan niet in een lange lijst maar eerder sublijsten op categorie. Bij het implementeren van deze lijst zullen simpele rechthoeken gebruikt worden totdat de basis blokken zijn geïmplementeerd (zie 3.1.3).

3.1.3 Basis blokken

Deze blokken worden het vaakst gebruikt en zijn nodig om programma's te maken. De volgende blokken noemen we basis blokken:

1. Start blok
   Dit is het blok die de start van het programma zal aanduiden.
2. If-else blokken
   Het if en else blok voor meerdere branches in het programma.
3. Vergelijkings blokken
   Deze blokken zullen integers vergelijken met $>, <, ==$,... of booleans met `&&`, `||`, ...
4. For en while blokken
   Blokken zodat stukken code meerdere keren uitgevoerd kunnen worden.
5. Booleans
   Deze blokken geven true of false terug en kunnen in verschillende andere blokken gezet worden.

3.1.4 Start- en stopknop

De IDE moet een start en stopknop hebben zodat we het zelfgemaakt programma kunnen uitvoeren. Zodra de start-knop is geïmplementeerd, kan een zelfgemaakt programma worden uitgevoerd. De stop-knop moet het programma tijdens het uitvoeren kunnen onderbreken en stopzetten. De stop-knop kan alleen ingedrukt worden als het programma bezig is en de start-knop enkel als het programma niet bezig is. Voor meer informatie over de werking van de start- en stop-knop zie 4.1.1 en 4.1.2.

3.1.5 Output via console

De output van het programma zal worden afgebeeld op een apart venster, het output-venster. Dit venster zal werken als een soort console zichtbaar voor de gebruiker. Met print-blokken zal het mogelijk zijn om booleans, integers en strings op de console uit te printen.

3.1.6 Opslaan en laden op webserver met accounts

We zullen de gebruiker de mogelijkheid geven om een account aan te maken. Dit account is bedoeld voor het opslaan en inladen van projecten. De gebruiker zal eerst een account moeten aanmaken voordat hij/zij een project kan opslaan. Dit project wordt opgeslagen in onze database (zie 4.4.1). Wanneer de gebruiker ingelogd is, zal er een opslaan-knop verschijnen. Wanneer er dan op de opslaan knop gedrukt wordt, krijgt de gebruiker de optie om een naam in de geven en zal het project opgeslagen worden onder die naam. Het project zal dan achterliggend in de database worden gezet (zie 4.4.2). Voor het inladen van projecten, zal een ingelogde gebruiker een lijst van zijn projecten te zien krijgen. Er zal op gelet worden dat bij het inladen van een project het huidige project niet meteen overschreven wordt, maar eerst een waarschuwing gegeven wordt.

3.1.7 API ondersteuning

We gaan verschillende HTTP-method blokken voorzien zoals GET, POST, PUT en DELETE. De GET blok gaat een return blok zijn met een input veld voor de URL van de API endpoint, bij uitvoering gaat dan dit endpoint aangeroepen worden en gaat de waarde die gereturned wordt worden teruggeven aan het programma. Voor POST, PUT en DELETE gaat er niet enkel een URL input voorzien moeten worden, maar ook key => value input. Dit kunnen we doen door aan deze blokken een EnclosureComponent toe te voegen en dan een key => value blok te maken die enkel in deze blokken hun EnclosureComponent past. Aangezien deze POST, PUT en DELETE blokken redelijk complex zijn, staan deze blokken in de low priority. Het GET blok proberen we wel al eerder te implementeren.

3.1.8 Tutorials

Voor beginners kan ons programma best ingewikkeld zijn, daarbovenop werken we ook met jonge kinderen. Het is dus belangrijk om tutorials te voorzien voor ons programma, zodat de gebruiker zonder hulp van anderen ons programma kan gebruiken. Zoals eerder vermeld, zullen tutorials plaatsvinden in een klein, apart venster. Dit venster zal zichtbaar zijn bij het eerste gebruik. Daarna zal het altijd beschikbaar zijn in een hulp-knop (zie mockup, figuur 5). De tutorials gaan stap voor stap door het creëren van een voorbeeld programma. Tegelijkertijd kan de gebruiker dit op zijn eigen canvas volgen. De tutorials zullen veel afbeeldingen hebben zodat het duidelijker is welke blokken er nodig zijn en waar ze plaatsen. De tutorial kan op elk moment gesloten worden.

3.1.9 Output en input via afbeeldingen

Er zal een mogelijkheid zijn om een afbeelding naar onze output console te sturen. Dit zal gebeuren met een afbeelding-print blok. Dit blok zal om een afbeelding vragen. Wanneer dit blok in de code bereikt wordt, zal de afbeelding verschijnen in de console.

3.1.10 Delen van projecten

Wij voorzien de mogelijkheid om projecten te delen. Hiervoor hoeft de gebruiker geen account te hebben. Op de project pagina zal er een knop zijn waarmee men projecten kan delen. Deze knop zal een kopie van het huidige project opslaan in de database (zie 4.4.3). De gebruiker krijgt een 5-cijferige code waarmee hij/zij en andere gebruikers het project kunnen openen. Bij het delen wordt een kopie van het project op de database gezet, aanpassingen zullen dus niet gedeeld worden met andere gebruikers. Er kan altijd een nieuwe code gemaakt worden voor het project met de laatste aanpassingen. Om de database niet te overbelasten, worden alle deelbare projecten tijdelijk bewaard. Dit betekent dat de projecten verwijderd worden na 24 uur. Deze optie is dus niet bedoelt voor het permanent opslaan van projecten, dat gebeurt met de eerder vermelde accounts.

3.1.11 String operaties

Blokken voor string operaties kunnen handig zijn omdat we met een console output werken. We zullen met zekerheid het samenvoegen van 2 strings ondersteunen. Daarna kunnen nog operaties zoals lengte van een string berekenen, een string omzetten naar hoofdletters, naar kleine letters of splitsen van een string toegevoegd worden. Al deze blokken bevatten Ghostblokken waarin de gebruiker zijn eigen text kan typen.

3.1.12 Variabelen

Om de mogelijkheid om waardes op te slaan te geven, kan de gebruiker gebruik maken van variabelen. Alle aangemaakte variabelen zullen globaal zijn, dus moeten ze eerst aangemaakt worden door de gebruiker, voordat ze dit blok kunnen gebruiken. Het variabele blok zal dan toegevoegd worden aan de blokken lijst. De gebruiker kan dit blok dan uit de blokken lijst slepen of verwijderen. Alle variabelen worden opgeslagen als string. Als een variabele in een blok zit waar een int verwacht wordt, gebeurt er een cast naar een int. Als dit faalt zal het blok worden genegeerd. Om een

variabele een waarde te geven, voorzien we een set-blok. Dit blok verwacht een variabele en een waarde.

3.1.13 Undo voor acties

Wij zullen undo voor de volgende acties ondersteunen:

1. Verwijderen van blokken
2. Verplaatsen van blokken
3. Nieuwe blokken op het canvas plaatsen
4. Blokken aan elkaar haken
5. Blokken in een input veld zetten

We gaan al deze data verzamelen in de controller. Als er een operatie gebeurt op blokken zal deze verandering opgeslagen worden in de controller. Deze wordt daarna doorgestuurd naar het model. Door deze data op te slaan, kunnen we bij undo de vorige staten makkelijk recreëren door de omgekeerde operatie toe te passen op de data die is opgeslagen.

3.2 Niet-functionele Vereisten

Dit zijn de kwaliteitseisen waaraan het programma moet voldoen. In de volgende subsecties gaan we het hebben over de layout, de leesbaarheid, het zoeken van blokken in de blokkenlijst, blokken terugvinden op het canvas, robuustheid en de mogelijkheid om thuis verder te werken.

3.2.1 Lay-out

Door ons jong doelpubliek, is onze lay-out heel belangrijk. De volgende dingen zijn zekere aandachtspunten voor de lay-out:

1. Makkelijk te begrijpen
   De lay-out moet makkelijk te begrijpen zijn voor de leeftijdsgroep van ons publiek. De werking van onderdelen moet onmiddellijk begrijpbaar zijn voor kinderen van 8 jaar.
2. Aantrekkelijk
   Het programma is bedoelt voor jonge kinderen. In het beste geval hebben deze kinderen plezier bij het gebruiken van onze IDE. De lay-out moet aantrekkelijk zijn zodat het kind het programma wil gebruiken en niet afgeschrikt wordt.
3. Simpel
   De gebruiker zou een beginner in het programmeren kunnen zijn. Het is daarom belangrijk dat de gebruiker zich volledig kan concentreren op het maken van zijn programma in onze IDE. Het zou daarom niet moeilijk mogen zijn om onze IDE te gebruiken.
4. Duidelijk
   Kinderen van 8-12 jaar zullen minder goed zijn in lezen, hierdoor zullen we zoveel mogelijk tekst vermijden. We proberen er ook voor te zorgen dat alle tekst duidelijk, makkelijk te begrijpen en groot genoeg is.

3.2.2 Leerbaarheid

De IDE moet makkelijk aan te leren zijn. De lay-out speelt hier een belangrijke rol in. Wanneer alles op een logische plaats geplaatst wordt en er dus niet teveel gezocht hoeft te worden, zal dit veel makkelijker aan te leren zijn voor de gebruikers. Ook de tutorials die voorzien worden, zorgen ervoor dat de gebruiker niet alles zelf hoeft uit te zoeken.

3.2.3 Zoeken in de blokken lijst

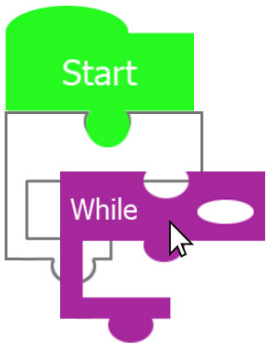
Wanneer de gebruiker een blok zoekt in de blokken lijst, moet de gebruiker dit blok zo snel mogelijk kunnen vinden. Hiervoor willen wij de blokken zichtbaar maken in categorieën en willen wij een veelgebruikte blokken categorie maken. Nieuwe gebruikers kunnen dan snel blokken terugvinden die vaak gebruikt worden, terwijl gebruikers met ervaring meer geavanceerde blokken kunnen zoeken per categorie.

3.2.4 Blokken vinden op het canvas

We werken met een oneindig canvas. Een beperkte hoeveelheid van het canvas kan maar zichtbaar zijn. We moeten er dus voor zorgen dat de gebruiker zijn blokken niet kwijtraakt. Dit kan met een klein overzicht van het canvas waarop meer blokken zichtbaar zijn, maar veel kleiner. Waar de gebruiker op het moment op het canvas zit, is dan zichtbaar in dit overzicht zodat de gebruiker weet naar welke kant hij/zij zich moet verplaatsen om zijn blokken te vinden.

3.2.5 Robuustheid

Om aan te geven of een blok op een bepaalde plaats geplaatst kan worden gaan we de silhouette van het blok al plaatsen, als dit is toegestaan, zodat de gebruiker weet waar zijn blok terecht zal komen als hij zijn linker muisknop loslaat. We gaan met onzichtbare placeholder-blokken werken zodat we kunnen detecteren wanneer de blok die de gebruiker aan het verslepen is binnen het bereik van dit blok komt. Om deze vervolgens dan op die plaats te bevestigen, als dit is toegelaten.



Figuur 1: Connecteren van 2 blokken

Ook zullen we foutmeldingen moeten voorzien voor wanneer de server offline is en de applicatie dus niet volledig gebruikt kan worden.

3.2.6 Thuis verder werken

Uit de meeting met de opdrachtgever blijkt dat thuis verder werken niet echt nodig is, de prioriteit ligt op het functioneel zijn tijdens de CoderDojo. Voor ons maakt dit weinig verschil tijdens het ontwerpen, want we moeten sowieso rekening houden met het feit dat de server niet beschikbaar kan zijn.

3.3 Extra 's

Eens al deze vereisten gehaald zijn, zouden we de volgende extra 's kunnen voorzien:

1. Zelf functies aanmaken
   Laat de gebruiker een groep blokken groeperen en meerdere keren aanroepen. Functies zorgen ervoor dat een grote groep blokken vervangen kan worden door 1 blok.

2. Output via geluid
   We hebben al output via afbeeldingen, misschien dat sommige gebruikers ook gebruik willen maken van geluid.

3. Gebruik van lijsten en dictionaries
   Een groot aantal waardes kunnen opslaan in de vorm van een lijst of een dictionary kan handig zijn in sommige programma's.

4. Ondersteuning van de Engelse taal
   Doordat kinderen meestal nog niet zo goed Engels kunnen, wordt de gehele applicatie in het Nederlands gemaakt. Engels is echter de belangrijkste taal voor programmeurs. Voor toekomstige programmeurs zou dit dus een leuke feature kunnen zijn.

4 Analyse

4.1 Algoritmen

4.1.1 Uitvoering van blokken

Als een programma gemaakt is in onze visual programming IDE en er op de startknop gedrukt wordt, zal de startblok op het canvas gevraagd worden om te beginnen. Elk blok met een AttachComponent heeft een pointer naar de volgende blok (zie sectie 4.2 Datastructuren) en zal na het uitvoeren, het volgende blok moeten aanroepen.
De logica gaan we uitvoeren binnen de klasse van het specifieke blok in de execute() functie. Deze functie is overridden van de basisklasse Block en gaat alle benodigde elementen uit de verschillende componenten halen en daar beslissen wat het volgende blok gaat zijn die uitgevoerd moet worden.

4.1.2 Stopknop

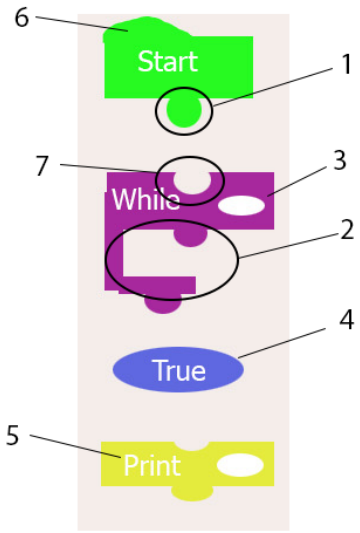
We zullen dit doen met een variabele. Deze variabele wordt dan veranderd als er op de stopknop wordt gedrukt. In elke execute() functie van de blokken zullen we eerst testen of deze veranderd is. Als dit het geval is zal de execute niks meer doen. Als het laatste blok bereikt wordt, zal er een waarde teruggegeven worden die het succes van het programma aanduid. Dit zal dan in de console zichtbaar worden, zodat de gebruiker weet dat er geen problemen waren met het uitvoeren van zijn programma.

4.2 Datastructuren

In deze sectie leggen we uit over de onderdelen die een blok kan hebben in ons programma. Deze onderdelen noemen we components. Elk blok kan een aantal components hebben die nodig zijn voor de werking van het blok. Op figuur 2 zijn een aantal van deze components aangeduid (niet alle components zijn aangeduid op het figuur). Voor meer informatie over welke components elk blok bevat zie 4.3.

4.2.1 AttachComponent

Blokken waar je een ander component aan kunt hangen, zullen het AttachComponent hebben. Dit component zal een verwijzing hebben naar het volgende blok dat uitgevoerd moet worden. Dit volgende blok zal altijd een ConnectComponent hebben. De StartBlock zal bewaard worden door het programma. De andere blokken met het AttachComponent staan dus in een linked-list met het startblok als hoofd-element. We denken niet dat we double-linked lists nodig hebben, omdat een blok nooit het vorige blok nodig heeft. Het attachcomponent is herkenbaar op figuur 2: 1.



Figuur 2: Voorbeelden van de veschillende components die blokken kunnen bezitten.De volgende components zijn aangeduid: 1 AttachComponent, 2 EnclosureComponent, 3 InputComponent, 4 ReturnComponent, 5 LabelComponent, 6 StartComponent, 7 ConnectComponent.

4.2.2 EnclosureComponent

Dit component bevat een inwendig programma dat makkelijk meerdere keren kan uitgevoerd worden voordat er naar het volgende blok wordt gegaan. Alleen het eerste blok in het programma wordt opgeslagen. Dit component wordt gebruikt voor blokken zoals een while-lus en een for-lus. Het enclosurecomponent is te zien op figuur 2: 2.

4.2.3 InputComponent

Het InputComponent dient voor blokken waar de gebruiker een waarde moet voorzien. Dit component verwacht een bepaald input-type. Als men een blok wilt plaatsen in dit component, moet het type van de ReturnComponent van die blok overeenkomen met het input-type dat verwacht wordt. Zo verwacht een for-loop bijvoorbeeld een integer waarde om het aantal herhalingen van de loop te kunnen bepalen. Het succes van een if-statement wordt bepaald door een boolean-waarde, daardoor verwacht het InputComponent van een if-blok een boolean. InputComponents bevatten standaard een GhostBlock. Een GhostBlock dient als placeholder totdat er een nieuwe blok in het component geplaatst wordt. De vorm van het inputcomponent is te zien op figuur 2: 3.

4.2.4 ReturnComponent

Het ReturnComponent is gemaakt voor blokken die in een InputComponent gezet kunnen worden. Dit component laat de blokken een waarde teruggeven nadat hun functie gedaan is. Neem bijvoorbeeld het blok voor een vergelijking (==). Dit blok gaat 2 integer-waardes vergelijken, het ReturnComponent gaat op basis van deze vergelijking een boolean-waarde teruggeven. Blokken met dit component hebben een speciale vorm (Zie figuur 2: 4) gelijk aan de vorm van het inputcomponent.

4.2.5 LabelComponent

Het LabelComponent van een blok maakt duidelijk over welk soort blok het nu juist gaat. De naam van het blok zal hier zichtbaar in zijn. Zo zal een if-blok bijvoorbeeld de label 'als ...' bevatten. Zie figuur 2: 5.

4.2.6 StartComponent

Het startcomponent is bedoelt voor blokken waar het programma kan starten. Dit component is bedoelt voor makkelijk de blokken eruit de halen die bovenaan het programma te staan. We kunnen deze blokken dan ook wat extra uiterlijk geven zoals te zien op figuur 2: 6.

4.2.7 ConnectComponent

Dit component is voor blokken die je aan blokken met het attachcomponent kan hangen. Het component is te zien op figuur 2: 7.

4.3 Klassendiagram en ontwerp

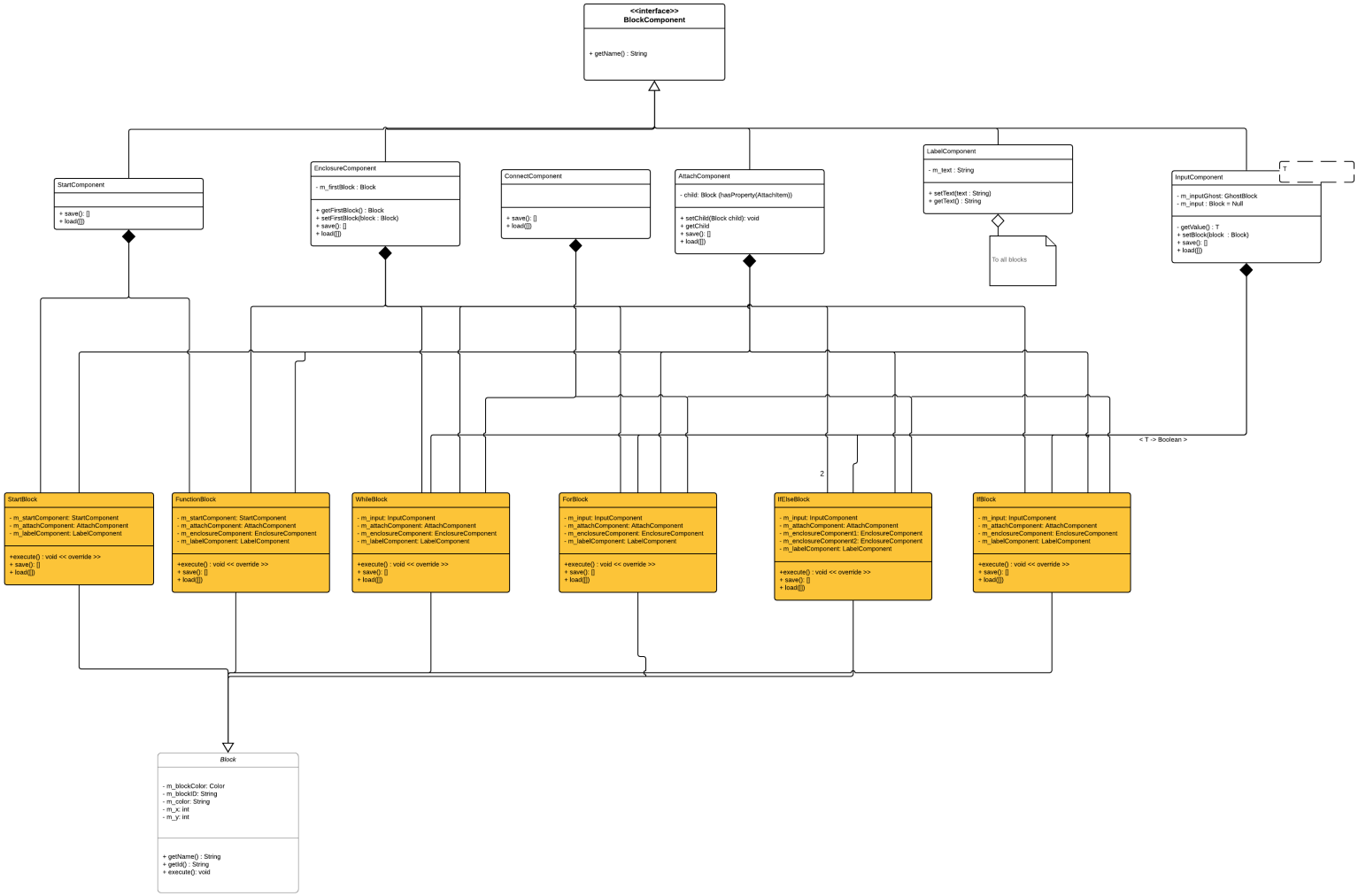
Vue is een zeer licht front-end framework, wat volgens meerdere bronnen beginners-vriendelijk is. Het is zeer makkelijk aan te leren doordat het enkel gebruik maakt van HTML en JavaScript. Daarbovenop wordt het aanbevolen voor lightweight projecten en relatief kleine programma's. Hierdoor verkiezen wij, als beginnende webdevelopers, om Vue te gebruiken als front-end framework voor ons project.[15][16][17][18][19][20]

Voor de back-end gaan we geen framework gebruiken, doordat de functionaliteit van ons project volledig op de front-end kan werken en er enkel een back-end nodig is voor communicatie met de database. Doordat er simpelweg accounts gecreëerd moeten worden en het inladen en opslaan van projecten allemaal relatief simpel gedaan kan worden, gaan we hiervoor PHP en AJAX gebruiken.

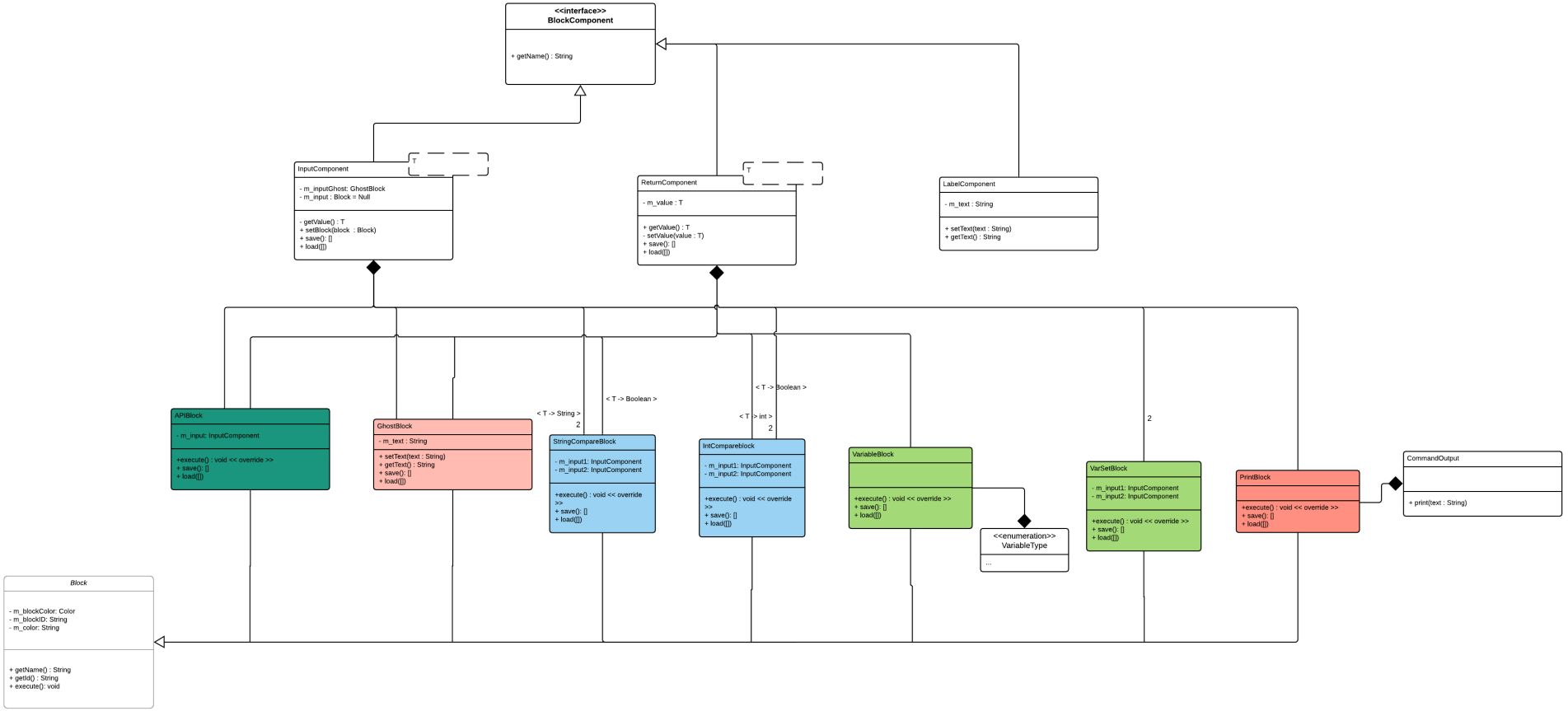
Tijdens het ontwerpen van het klassendiagram zijn we 2 mogelijkheden tegengekomen om blokcomponents aan een blok te geven. Als eerste kunnen we elke soort blok variabelen laten bevatten voor iedere component die deze bezit. Ten tweede kunnen we elke blok een lijst van blokcomponents laten bevatten. Als we voor de lijst kiezen, kunnen we makkelijk aan alle componenten geraken die een blok bevat. Als nadeel heeft dit wel dat intern, als we een bepaalde component nodig hebben, we telkens door de lijst moeten lopen op zoek naar het component van het type dat we nodig hebben. Als we voor de variabele manier kiezen, kunnen we de componenten intern makkelijk aanspreken. Als we dan alle componenten van een blok willen krijgen, zullen we echter wel voor elke soort blok een getAllComponents-functie moeten afleiden die specifiek voor die blok alle componenten teruggeeft. Het voordeel van de lijst is low coupling en dat het zeer open staat voor uitbreiding. Wij hebben er voor gekozen om de componenten als variabelen in de blok klasse te stoppen. Deze keuzen hebben we gemaakt omdat we vaker de componenten direct nodig hebben binnen de blok klasse dus is het efficiënter om niet eerst nog door een lijst te moeten loopen om het correcte component te vinden.

Bij de verschillende input types hebben we gekozen om met templates te werken. Dit heeft als voordeel dat we de gebruiker kunnen limiteren om enkel de toegestane types te gebruiken.

Voor de view gaan we werken met SVG-elementen om de blokken te tekenen. We hebben hiervoor gekozen in plaats van een canvas of SVG files te gebruiken. Dit doordat bij een canvas het niet mogelijk is SVG elementen, die nodig zijn om onze blokken een zelfgekozen vorm te kunnen geven, toe te voegen en als we SVG files gebruiken dan kunnen we de blokken niet van grootte veranderen. We gaan elke blok een uniek ID geven. De ID's van alle start blokken worden in een lijst bijgehouden in de klasse Canvas. Zo kunnen we gemakkelijk alle start blokken terugvinden voor het starten van de uitvoering van het programma. Ook het plaatsen van deze blokken in de overview wordt hierdoor vergemakkelijkt. Al de start blokken hebben hun eigen SVG-tag met een positie binnen de parent-div (het canvas). Deze parent SVG-elementen kunnen we dan een positie geven binnen het div. Door de functionaliteit van SVG gaan de children van een element automatisch mee opschuiven bij verplaatsing van de parent. Om het canvas zelf te verschuiven kunnen we een translatie via CSS aan deze div toevoegen.



Figuur 3: Klassendiagram voor blokken met het attachcomponent



Figuur 4: Klassendiagram voor blokken zonder het attachcomponent

4.4 Data en databaseschema

4.4.1 User table

In deze tabel worden de gebruiker-accounts opgeslagen. Het bevat de volgende velden:

- int id : uniek id per gebruiker (primary key value)
- string username: gebruikersnaam (unique)
- string password : wachtwoord

De gebruikersnaam moet uniek zijn zodat er mee ingelogd kan worden. Voor wachtwoorden zullen we de php password_hash() functie gebruiken. Deze functie maakt er een nieuw wachtwoord van met een sterk one-way hashing algoritme. Deze functie geeft de optie om zelf een algoritme te kiezen. Dit laten we op default staan omdat dit dan verandert als php een nieuw sterker algoritme vindt. Deze default is op het moment het CRYPT_BLOWFISH (bcrypt) algoritme.

4.4.2 File table

In deze tabel worden de projecten opgeslagen van de gebruikers. De tabel bevat de volgende velden:

- int fileid : uniek id per file (key value)
- int userid : id van bestandeigenaar (foreign key, references User table(id))
- string projectname : naam van het project, gekozen door de gebruiker
- json project : het project in JSON formaat

Voor het formaat van de JSON file zie 4.5.

4.4.3 TempFile table

Deze tabel dient voor het delen van stukken code en projecten. Bij het delen van projecten zal er een kopie gemaakt worden van het te delen project. Deze kopie wordt tijdelijk (24 uur) opgeslagen. De kopieën worden opgeslagen in deze tabel. Elke toevoeging wordt na 24 uur verwijderd. De gebruiker kan aanpassingen maken aan de kopie zonder het originele projecten aan te passen. Als de gebruiker dan zijn aanpassing wilt opslaan, kan deze dit doen onder zijn eigen account als nieuwe entry in de File table. Bij het aanmaken van een kopie wordt ook een 5-cijfercode aangemaakt. Deze code wordt berekend door de modulo van de index van de rij in de tabel en 10000 te nemen (code = index%10000). Codes die niet direct 5 cijfers lang zijn, worden aangevuld met nullen. De tabel heeft de volgende velden:

- int fileid : uniek per tijdelijke file
- json project : het tijdelijke project in JSON formaat
- timestamp time : de tijd en datum wanneer het project werd opgeslagen in de database
- int code : Een 5-cijferige code waarmee de kopie geopend kan worden

Voor het formaat van de JSON file zie 4.5. 4.4.2 .

4.5 Bestandsformaat

4.5.1 Opgeslagen projecten

We slaan de projecten op in JSON-files. In een JSON-file slaan we alle blokken op het canvas op in een lijst met elementen van de vorm {x,y,id,name,components}:

- x : het x-coördinaat van het blok op het canvas
- y : het y-coördinaat van het blok op het canvas

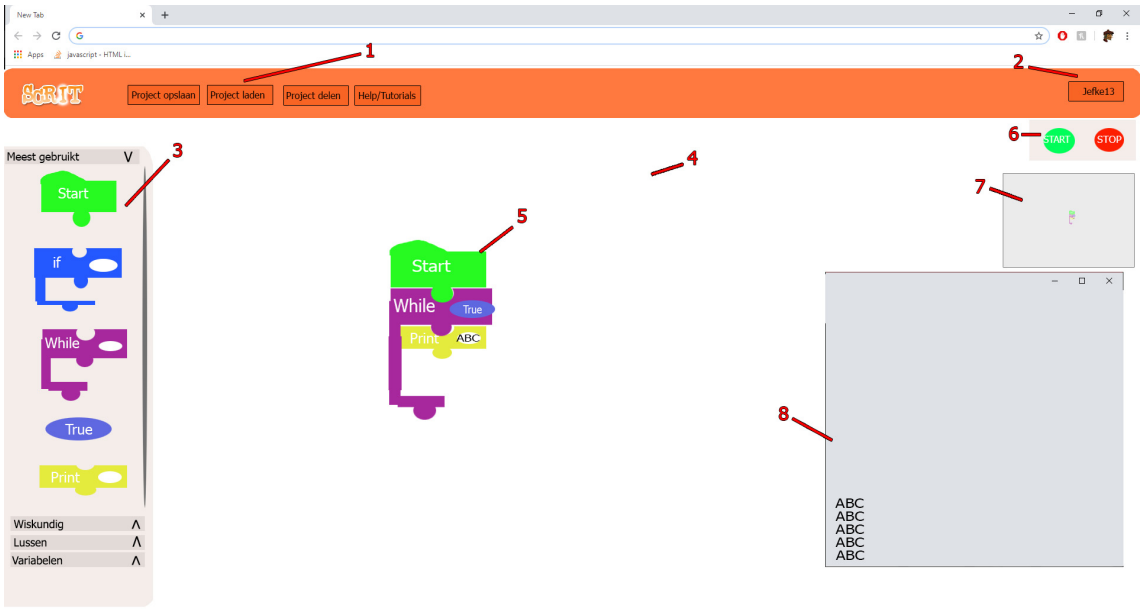
- id : het unieke id van het blok op het canvas
- name : de naam van het blok op het canvas
- components : een lijst van de info over de components in het blok

De coördinaten x en y zijn alleen belangrijk voor blokken zonder parentblok. Voor blokken met een parentblok kunnen deze coördinaten best herberekend worden bij het inladen zodat het blok correct vastzit in zijn parent. De naam van het blok is belangrijk om te weten welk blok het is. Het id is uniek per blok op het canvas zodat er naar verwezen kan worden in andere blokken. De lijst van components bevat elementen in de vorm {name, (componentinfo)}. Name geeft de naam van het component. Daarna komt informatie die het component nodig heeft. Dit is uniek per component.
Voorbeeld:

```
{
"blocks" : [
    {
        "x" : 10,
        "y" : 30,
        "id" : 1,
        "name" : "StartBlock",
        "components": [
            {
             "name" : "AttachComponent",
             "nextblockid" : 2
            }
        ]
    },
    {
        "x" : 10,
        "y" : 40,
        "id" : 2,
        "name" : "ForBlock",
        "components": [
            {
             "name" : "AttachComponent",
             "nextblockid" : null
            },
            {
             "name" : "EnclosureComponent",
             "firstblockid" : 5
            },
            {
             "name" : "InputComponent",
             "inputblockid" : null
            }
        ]
    },
    {
        "x" : 2,
        "y" : 4,
        "id" : 3,
        "name" : "VariableBlock",
        "components": [
            {
             "name" : "ReturnComponent",
             "type" : "int"
```

```
            }
        ]
    },
    {
        "x" : 10,
        "y" : 45,
        "id" : 5,
        "name" : "PrintBlock",
        "components": [
            {
             "name" : "AttachComponent",
             "nextblockid" : null
            },
            {
             "name" : "InputComponent",
             "inputblockid" : null
            }
        ]
    }]
}
```

5 Mockups



Figuur 5: Mockup voor de applicatie. Zie 7

Bovenstaande afbeelding is een eerste kijk op hoe de applicatie er ongeveer zal uitzien bij het maken en uitvoeren van een relatief simpel eerste programma. Op de afbeelding wordt er naar een aantal gebieden gewezen met een getal. Deze gebieden worden hieronder uitgelegd:

1. Links bovenaan in de header van de pagina bevinden zich een aantal knoppen. Een knop om projecten op te slaan, een knop om reeds opgeslagen projecten in te laden, een knop om een project te delen en een knop om het tutorial-menu, waar men tutorials kan starten, te openen.

2. In de rechterkant van de header kan men inloggen en registreren. Als men reeds ingelogd is, zal de gebruikersnaam van de gebruiker zichtbaar zijn.

3. Links op de pagina bevindt zich het blokken menu. Hier worden alle blokken die men op het canvas kan plaatsen opgesomd. Vooraleer men blokken in categorieën moet zoeken, zijn de meest voorkomende blokken zichtbaar. Dit vergemakkelijkt het maken van programma's voor gebruikers zonder enige ervaring.

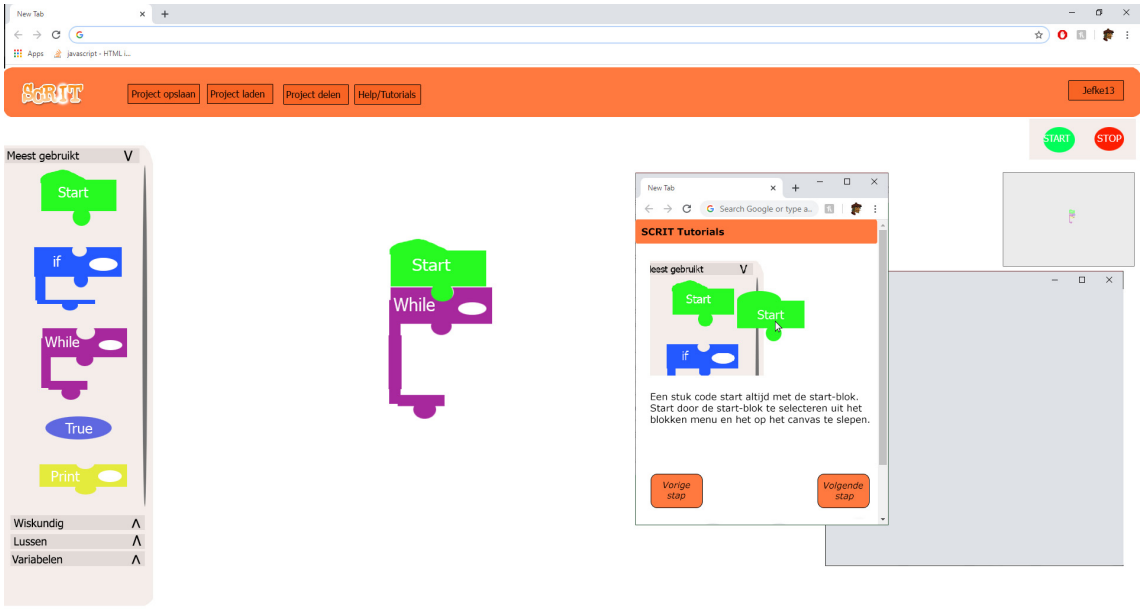
4. De witte achtergrond stelt het semi-oneindig canvas voor waarop men blokken kan plaatsen

5. Op het canvas kan men code creëren door blokken te plaatsen en te connecteren, zoals op de voorgestelde manier.

6. Rechts bovenaan op de pagina zijn 2 knoppen zichtbaar, een start en een stop-knop. De werking van deze knoppen spreekt voor zich. Als men op de start-knop duwt, begint het uitvoeren van de gemaakte code onder de start-blok. Op dezelfde manier stopt de applicatie met het uitvoeren van deze code als er op de stop-knop geduwd wordt.

7. Onder de start en stop-knop is een overzicht van het canvas te vinden. Zo is het makkelijker om verloren blokken terug te vinden.

8. Rechts beneden de applicatie zien we het output-venster. Dit is een apart venster waar de output van de gemaakte code zichtbaar zal zijn tijdens en na het uitvoeren.



Figuur 6: Mockup voor tutorials

In bovenstaande afbeelding is een extra venster te zien. Dit venster wordt gebruikt om een tutorial in te volgen. Tutorials, voor dit project, zijn step-by-step guides in een apart venster, zo kan de gebruiker een tutorial volgen op eigen tempo en tegelijkertijd zelf experimenteren met de blokken.

6 Taakverdeling en planning

Deadline	**Mijlpaal**	**Persoon**
26/03	Creëren van oneindig canvas op index	Pim
26/03	Blokkenklasses maken	Joep
26/03	BlokkenView maken	Wout
28/03	Blokken op canvas plaatsen	Pim
28/03	Blokken op canvas kunnen slepen	Pim
31/03	Blokken connecteren	Joep
31/03	Uiterlijk van blokken maken	Wout
8/04	Blokkenlijst creëren	Joep
8/04	Start en stop-knop aanmaken	Pim
8/04	Console voor output aanmaken	Wout
10/04	Blokkencode omzetten naar output	Wout
20/04	TESTEN VAN FUNCTIONELE VEREISTEN + DEBUG	Iedereen
24/04	Verwerking feedback	Iedereen
30/04	Databases aanmaken	Pim
3/05	Opslaan en inladen van projecten op de webserver	Joep
10/05	Uiterlijk website (Header)	Wout
10/05	Login voor accounts	Pim
10/05	API-blok aanmaken en achterliggende ondersteuning	Joep
17/05	Verbetering lay-out	Joep
17/05	Tutorials creëren	Pim
17/05	Overview van blokken op canvas	Wout
20/05	Variabelen blokken aanmaken	Joep
20/05	Ondersteuning van afbeeldingen	Wout
20/05	Delen van projecten via link	Pim
27/05	DEBUGGING	Iedereen

7 Bibliografie

Referenties

[1] https://scratch.mit.edu/info/faq.

[2] http://ai2.appinventor.mit.edu/.

[3] https://www.gethopscotch.com/faq.

[4] https://microbit.org/guide/.

[5] https://www.roberta-home.de/en/lab/.

[6] https://lab.open-roberta.org/.

[7] https://developers.google.com/blockly.

[8] https://scratch.mit.edu/developers.

[9] https://github.com/microsoft/pxt.

[10] https://github.com/pencilcode/droplet.

[11] https://github.com/jmoenig/snap.

[12] https://stackoverflow.com/questions/47371623/html-infinite-pan-able-canvas.

[13] https://javascript.info/mouse-drag-and-drop.

[14] https://www.w3schools.com/html/html5_webworkers.asp.

[15] https://www.monterail.com/blog/vue-vs-react-2019.

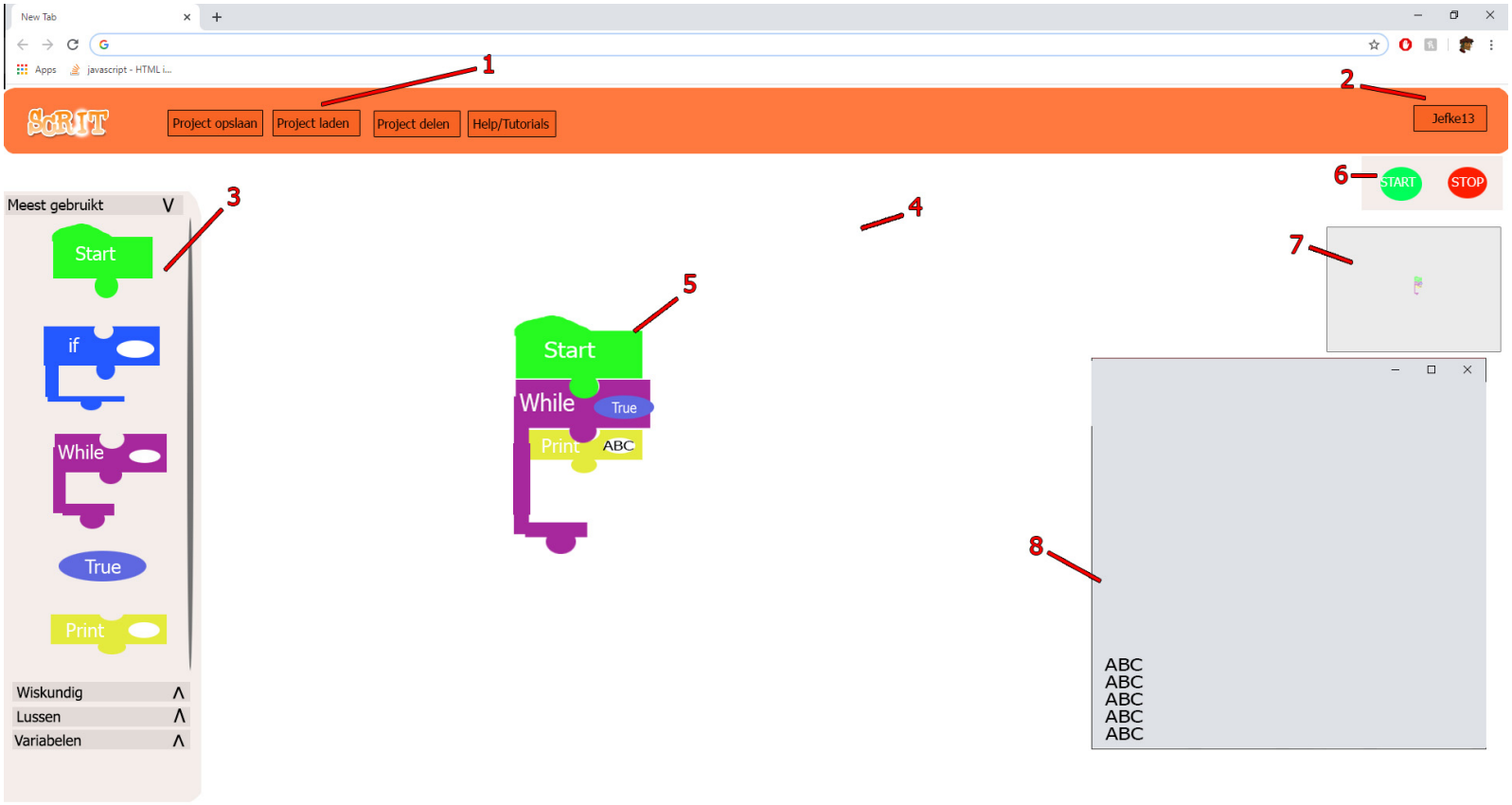
[16] https://vuejs.org/v2/guide/comparison.html.

[17] https://medium.com/javascript-in-plain-english/i-created-the-exact-same-app-in-react-and-vue-here-are-the-differences-e9a1ae8077fd.

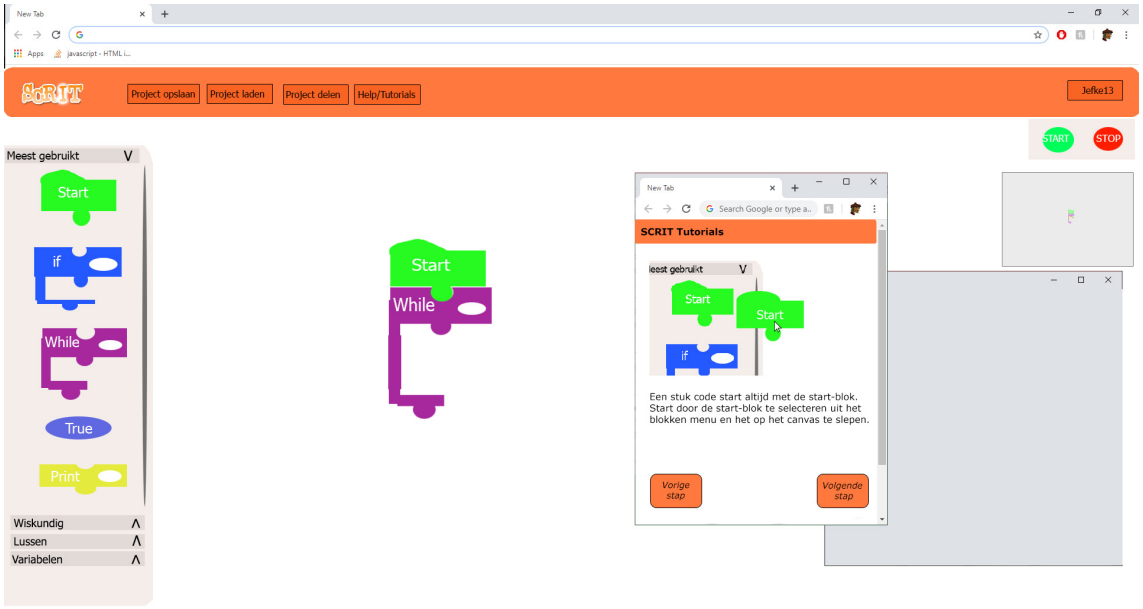
[18] https://www.decipherzone.com/blog-detail/top-javascript-frameworks-for-web-application-development.

[19] https://www.decipherzone.com/blog-detail/angular-vs-vue–which-is-better-for-web-app-development-.

[20] https://www.decipherzone.com/blog-detail/top-front-end-development-frameworks-in-2019.



Figuur 7: Mockup voor de applicatie



Figuur 8: Mockup voor tutorials