Technisch ontwerp

Djimaro Talahatu

Djimaro Talahatu - 563631
Hogeschool van Arnhem en Nijmegen, faculteit ICA, HBO-ICT Software Development
Docentbegeleider: Dennis Breuker
Bedrijfsbegeleider: Alex Bijsterveld
Opdrachtgever: Cito
3 mei 2020, Apeldoorn
Versie 1.0

Inhoudsopgave

1 Definitions of Terms and Acronyms

Term	Definition
Onderzoeker	Het personeel van Cito dat helpt bij het ontwikkelen van toetsen en examens.
Niveau	Een categorie voor een verzameling variabelen. Bijvoorbeeld: De categorie 'school' heeft de variabelen HAN en Radboud. De categorie 'response' heeft antwoorden van leerlingen als variabelen.
Variabele	Een element met informatie. Bijvoorbeeld: "15", "Apeldoorn", "HBO" of "De hoofdstad van Amerika is Washington D.C.".
Eigenschap	De eigenschappen van variabelen. Bij een antwoord (response) van een leerling kan dit de lengte van de tekst zijn. Voor een set van voorbeelden zie (Zie Bijlage D)
Bestand	Een bestand met kolommen en rijen. Dit kan een Excel bestand zijn. In de eerste rij van een bestand staan de niveaus. Elke rij daaronder bevat variabelen.
Script	Een script dat de eigenschappen van variabelen achterhaalt en omzet naar een plaatje.
Open vraag	Een vraag waarop een leerling op zijn manier een antwoord (response) kan geven.
MoSCoW	Methode om requirements te prioriteren. Hierin staat de M voor Must haves, de S voor Should haves, de C voor Could haves, en de W voor Would haves.
Response	Is hetzelfde als een antwoord van een leerling.
Variable	Is hetzelfde als variabele.
Levels	Is hetzelfde als niveaus.
Property	Is hetzelfde als eigenschap.

Tabel 1.1: Definities

2 Introduction

Cito maakt het onderwijs krachtiger door te laten zien wat leerlingen in hun mars hebben. Ze zetten hun unieke expertise in om de ontwikkeling van kennis, vaardigheden en competenties te volgen en inzichtelijk te maken. Daarmee kunnen zij leraren, ouders en werkgevers ondersteunen. Op deze manier kunnen leerlingen het beste uit zichzelf halen en richting geven aan hun toekomst.

Cito laat zien wat leerlingen in hun mars hebben door het ontwikkelen van toetsen en examens en de resultaten vervolgens te meten met hulpmiddelen zoals meetinstrumenten. Om te laten zien wat leerlingen kunnen, moet de kwaliteit van toetsen en examens goed zijn. De kwaliteit van toetsen en examens onder ander wordt bepaald door onder ander de antwoorden die leerlingen geven op een onderdeel van een toets of examen.

Het doel van Cito is het ervoor zorgen dat iedere leerling gelijke kansen krijgt. Dit is, sinds de oprichting, de drijfveer voor medewerkers van Cito. De primaire focus van 'goed en eerlijk toetsen' is verbreed naar het objectief meten van mogelijkheden en talenten.

Toetsen en examens bestaan uit open en gesloten vragen. Bij een open vraag kan de leerling op zijn manier antwoord geven. Bij een gesloten vraag kan de leerling kiezen uit een set van ant-

woorden.

Er zijn meetinstrumenten die de kwaliteit van gesloten vragen kunnen meten. Bij een gesloten vraag is er een beperkte set van antwoorden beschikbaar. Deze set van antwoorden ondersteunen de interpretatie van de vraag. Hierdoor is het gemakkelijk om de antwoorden van leerlingen te analyseren met behulp van meetinstrumenten.

Het is moeilijker om de kwaliteit van open vragen te beoordelen. Dit geldt nog meer voor Nederlandse open vragen. Hiervoor zijn er nog minder hulpmiddelen beschikbaar.

De kwaliteit van Nederlandse open vragen worden op dit moment minder goed gewaarborgd, omdat het niet duidelijk is op welke eigenschappen er gelet moet worden. Voor dit project heeft Cito een open vraag(Zie Bijlage A) met daarbij antwoorden(Zie Bijlage B) gegeven. Deze open vraag gaat over het maken van een samenvatting in minder dan 150 woorden. Om te bepalen welke eigenschappen er nodig zijn om de kwaliteit van de open vraag te waarborgen is er onderzoek gedaan.(Zie Bijlage C) Hiervoor is een taalcoach van de HAN geïnterviewd.(Zie Bijlage D) Uit het interview bleek dat er geen theorie is over welke eigenschappen er nodig zijn. Wel is er advies gegeven om te kijken naar een set van eigenschappen. (Zie Bijlage D)

De kwaliteit van Nederlandse open vragen worden op dit moment minder goed gewaarborgd, omdat het meten van de kwaliteit zonder meetinstrumenten teveel tijd kost. Iemand moet een grote hoeveelheid antwoorden verzamelen en deze antwoorden en de eigenschappen ervan analyseren. Dit handmatig proces moet iedere keer gebeuren als er een nieuwe vraag is of als de vraag wordt gewijzigd.

Als de kwaliteit minder goed wordt gewaarborgd, kunnen de vragen op verschillende manieren worden geïnterpreteerd. Bij een open vraag kan de leerling op zijn manier antwoord geven. In vergelijking met gesloten vragen zijn er minder hulpmiddelen voor de leerling beschikbaar. Hierdoor is het lastiger voor de leerling om de vraag op de juiste manier te interpreteren.

Als de vragen op verschillende manieren kunnen worden geïnterpreteerd, hebben de leerlingen dus geen gelijke kansen. Elke leerling is anders en begrijpt de vraag op zijn eigen manier. Hierdoor kunnen leerlingen verkeerde antwoorden geven en misschien zelfs een toets niet halen. Dit beïnvloedt de toekomst van de leerling. Als de leerlingen geen gelijke kansen hebben is dit dus in strijd met het doel van Cito.

Het probleem is dus dat de kwaliteit van Nederlandse open vragen minder goed wordt gewaarborgd, omdat het proces teveel tijd kost en er geen meetinstrument voor beschikbaar is.De opdrachtgever van dit project is Eva de Schipper. Zij is een onderzoeker van Cito. Dit document is opgesteld aan de hand van gesprekken met de opdrachtgever. De uitvoerder van dit project is Djimaro Talahatu. Hij is een afstudeerder van de HAN. Dit document beschrijft het gedrag van de applicatie op code niveau die bijdraagt aan het oplossen van het probleem. Bij het lezen van dit document wordt er dan ook verwacht dat de lezer de termen zoals design class diagram, sequence diagram en deployment diagram kent. De opdrachtgever van dit project is Eva de Schipper. Zij is een onderzoeker van Cito. De uitvoerder van dit project is Djimaro Talahatu. Hij is een afstudeerder van de HAN.

3 Architectural overview

De kwaliteit van een open vraag wordt bepaald door het analyseren van de antwoorden van leerlingen. Bij bijvoorbeeld de vraag "Wat is de hoofdstad van China?"wordt er Beijing of Peking als antwoord verwacht. Als het grootste deel van de leerlingen een totaal ander antwoord geven dan is er waarschijnlijk iets mis met de vraag.

Om te achterhalen waar er op gelet moet worden bij deze antwoorden is er een expert op het gebied van schrijven geïnterviewd. Binnen Cito is er niemand beschikbaar voor een interview. Hierom is Wout Waanders een taalcoach van de HAN geïnterviewd. (Zie Bijlage D)

De taalcoach geeft aan dat er geen theorie is die bevestigt dat deze teksteigenschappen de enige of de juiste teksteigenschappen zijn, omdat iedereen op zijn eigen manier schrijft. Hij geeft aan dat een goede tekst kan bestaan uit veel of weinig woorden met korte of lange zinnen. Hierom is het lastig om te bepalen of een tekst goed is.

Hij raadt aan dat voor het bepalen van de kwaliteit van een open vraag het handig is om de teksteigenschappen (Zie Bijlage D) van antwoorden te analyseren.

Als het niet duidelijk is welke teksteigenschappen de kwaliteit van open vragen inzichtelijk maken dan moet er nog onderzoek gedaan worden. De aanbevolen teksteigenschappen zijn niet onderbouwd met onderzoeken. Hierom moet er één of meerdere onderzoeken gedaan worden om te bepalen wat antwoord goed maakt en welke teksteigenschappen dat bepalen.

Het doen van onderzoek naar teksteigenschappen is niet haalbaar of relevant voor dit afstudeerproject, hierom kan de applicatie worden uitgebreid met andere teksteigenschappen. Het onderzoeken van alle teksteigenschappen valt buiten het project, omdat het niet bijdraagt aan het halen van dit afstudeerproject. Doordat de onderzoekers eigenschappen kunnen toevoegen, wijzigen en verwijderen zijn ze niet alleen beperkt tot de teksteigenschappen die tijdens dit project zijn achterhaald.

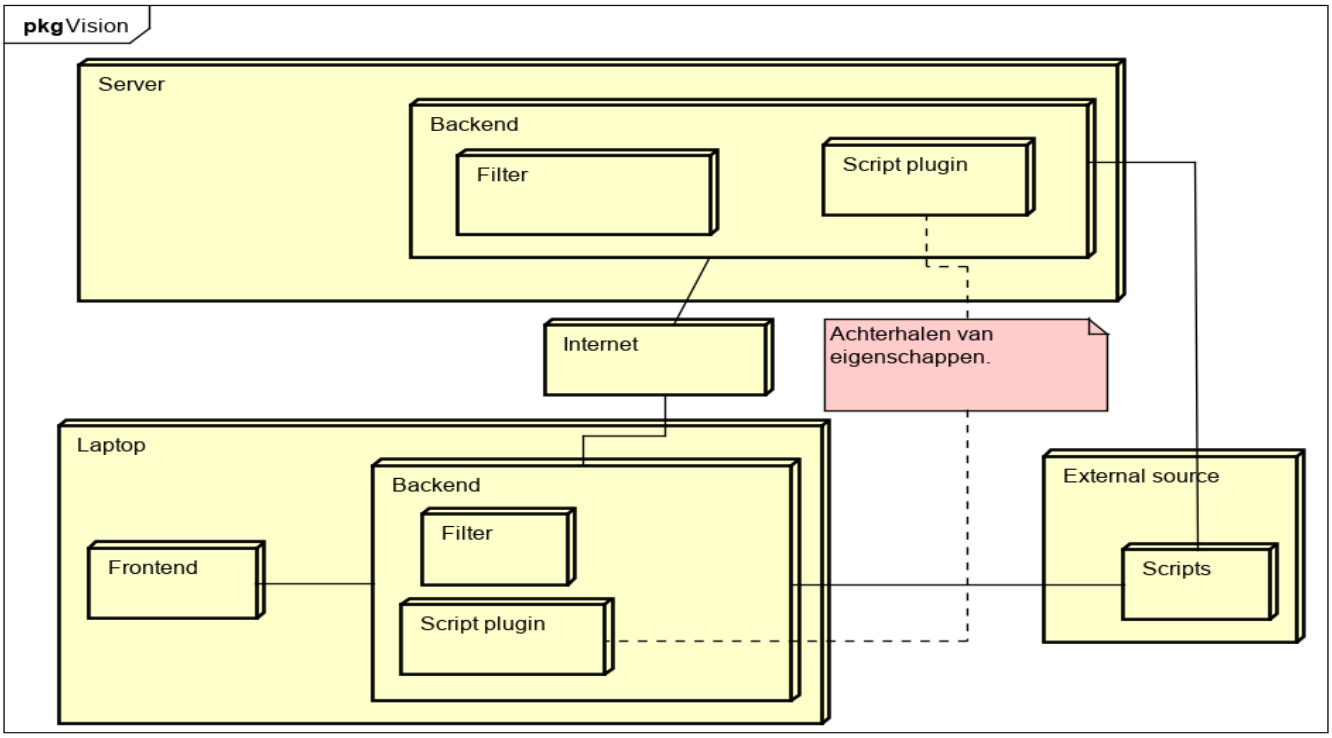
Door de eigenschappen te visualiseren kunnen de onderzoekers op basis van een grote hoeveelheid antwoorden de kwaliteit bepalen. Doordat de applicatie uitgebreid kan worden en de teksteigenschappen gevisualiseerd worden, worden er twee processen versnelt: het onderzoeken welke teksteigenschappen er nodig zijn en het proces om de kwaliteit te bepalen.

De applicatie uitbreiden met andere teksteigenschappen en het visualiseren ervan zijn één van de factoren die de structuur van de applicatie bepalen. Er zijn een aantal factoren die de structuur van de applicatie bepalen. In dit hoofdstuk worden alleen de drivers behandeld. De rest van deze factoren zijn onderdeel van hoofdstuk Architectural views. (Zie Bijlage E)

Drivers zijn één van de factoren die invloed hebben op de structuur van de applicatie. Deze drivers zijn verkregen door de requirements en concerns uit het software architecture document (Zie Bijlage E)te analyseren. Hieronder staan de drivers die de structuur van de applicatie bepalen.

Scripts	De onderzoekers willen bij het gebruik van deze applicatie hun eigen scripts gebruiken. Hierbij willen niet alle onderzoekers een nieuwe taal leren. Hier kunnen zij vanwege de plugin architectuur de applicatie uitbreiden met nieuwe programmeertalen.
External resources	Onderzoekers wisselen scripts onderling uit. Hierbij is het probleem dat de script niet altijd werkt. Om dit te garanderen worden de script centraal opgeslagen.
Bestand	De onderzoekers zijn gewend om in een bestand te werken met kolommen en tabellen. Door de plugin architectuur kunnen de onderzoeker de applicatie uitbreiden met andere soorten bestanden waarin zij werken met kolommen en rijen. Hierdoor is de applicatie meer toekomst bestendig.
Filteren	De onderzoekers willen niet telkens een nieuwe bestand inladen. Hierom zou het gemakkelijk zijn om een volledige bestand in te laden en dat het daarna mogelijk is om te filteren.

Door de bovenstaande drivers ziet de applicatie er op architectuur niveau zo uit:



Figuur 3.1: Initial vision

Figuur 4.2: Deployment diagram

Frontend	Graphical user interface van de software.
Filter plugin	Is verantwoordelijk voor het inladen en filteren van een bestand.
Script plugin	Is verantwoordelijk voor het uitvoeren van de script, installeren van benodigde libraries, waar de script wordt uitgevoerd en het updaten van scripts.
External Resource	Is verantwoordelijk voor het inladen en filteren van een bestand.

4 Detailed Design Description

Dit diagram is overgenomen uit het software architecture document. (Zie Bijlage E)
De laptop is een laptop van de gebruiker zelf. De server wordt opgezet door Cito.

4.1 Design decisions related to deployment

Voor de communicatie tussen de frontend, backend en de server wordt er gebruikt gemaakt van JSON, omdat dit leesbaar is voor mensen. Hierdoor kan Cito de applicatie nog gemakkelijker onderhouden.

Het valt op dat de backend van de laptop en server er hetzelfde eruit zien. Ze zijn ook bijna hetzelfde. Er is één verschil qua gedrag: Als de script op de server wordt uitgevoerd wordt deze uitgevoerd zonder te kijken naar hoeveel rijen aan variabelen er zijn. Doordat beide backenden hetzelfde zijn kan de applicatie op meerdere manieren gedeployed worden. Als Cito geen server wil gebruiken, zetten ze de applicatie op de laptop en blijft het werken. Als Cito geen goede laptop heeft, kan de gehele applicatie op de server.

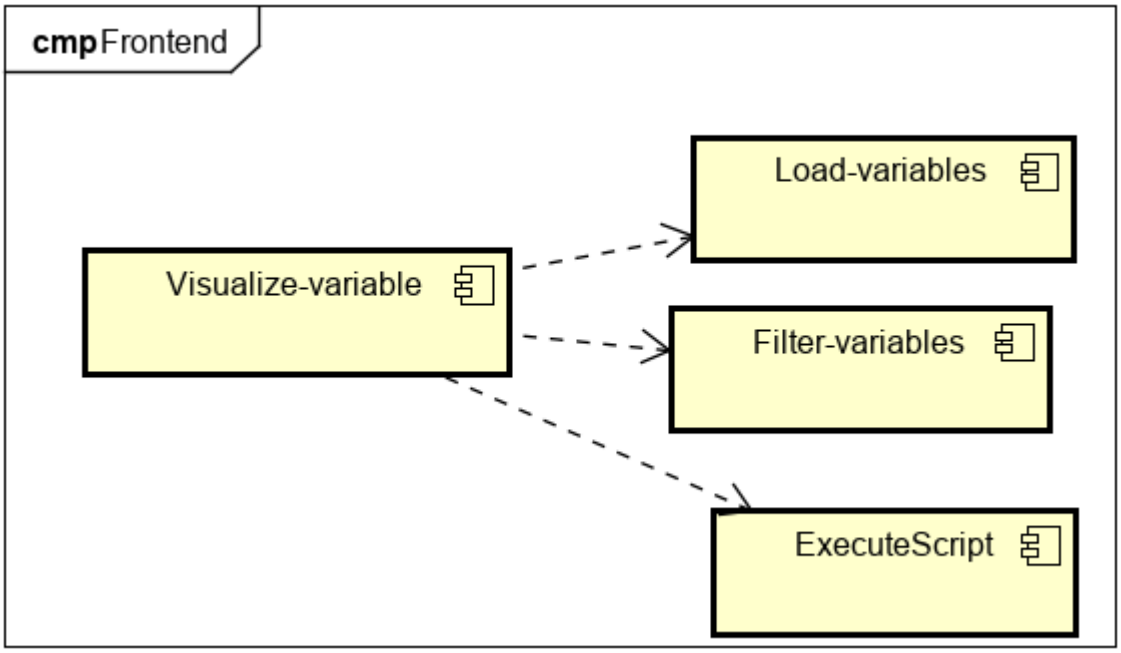
Bij de sequence diagrammen wordt per diagram getoond wat de url is naar de functie en hoe de request eruit ziet. Deze applicatie bestaat uit een frontend en een backend. Dit hoofdstuk is ook daarop verdeeld. Eerst wordt de frontend besproken en daarna de backend.

Voor alle documentatie is er gebruik gemaakt van UML.

4.2 Frontend

Tijdens het maken van het prototype van de frontend (Zie Bijlage I) is gebleken dat het lastig is om hiervoor een duidelijk ontwerp te maken. Dit komt doordat er gebruik wordt gemaakt van Angular.

Een module in Angular is een bouwblok van de applicatie. Bij het genereren van een module wordt een verdeling van code gemaakt, waarbij elk onderdeel een andere verantwoordelijkheid heeft.(Angular, z.d.-a)



Figuur 4.3: Front end modules

Als één module voldoet aan de SOLID principe, is de kwaliteit voor het onderhouden en uitbreiden van de applicatie genoeg. Er kan niet volledig rekening gehouden worden met dit principe, omdat er geen interfaces zijn en het niet object georiënteerd is.

Het ontwerpen op code niveau is met angular complex, omdat de structuur van angular complex is. De frontend is voor onderzoekers lastig uit te breiden, omdat zij Angular niet kennen en Angular een hoge leercurve heeft (Rajput, 2016). Voor de onderzoekers wordt er aangeraden om een andere frontend te gebruiken, omdat Angular een hoge leercurve heeft en zij geen uitgebreide ervaring hebben met programmeren.

Om deze reden is ervoor gekozen om de modules met behulp van SOLID en de use cases op te splitsen. Op deze manier is de code minimaal onderhoudbaar en uit te breiden als de onderzoekers er alsnog gebruik van willen maken.

Dit diagram is gemaakt op basis van use cases, SOLID en schermontwerpen. Voor de use cases (Zie Bijlage P) voor de schermontwerpen (Zie Bijlage P)

Visualize variable is de hoofdmodule die de usecase ”Visualiseren van eigenschappen”representeert. Deze module is verantwoordelijk voor het laden van de submodules:

- Load variables
- Filter variables
- Execute script

Load variables is verantwoordelijk voor het inladen van variabelen.

Filter variables is verantwoordelijk voor het filteren van variabelen.

Execute script is verantwoordelijk voor het uitvoeren van scripts.

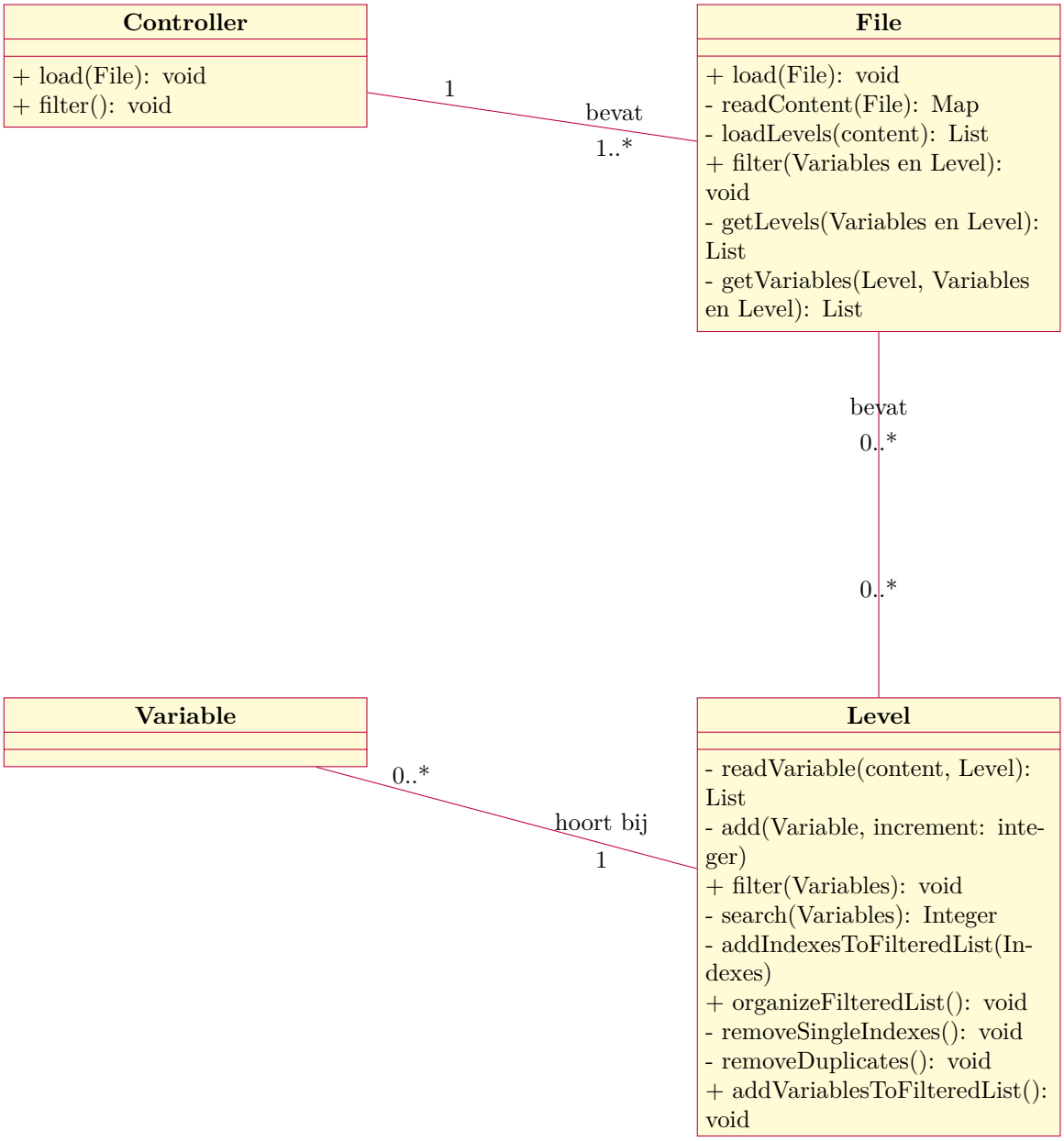
4.3 Backend

Hier worden de design class diagrammen besproken en de sequence diagrammen.

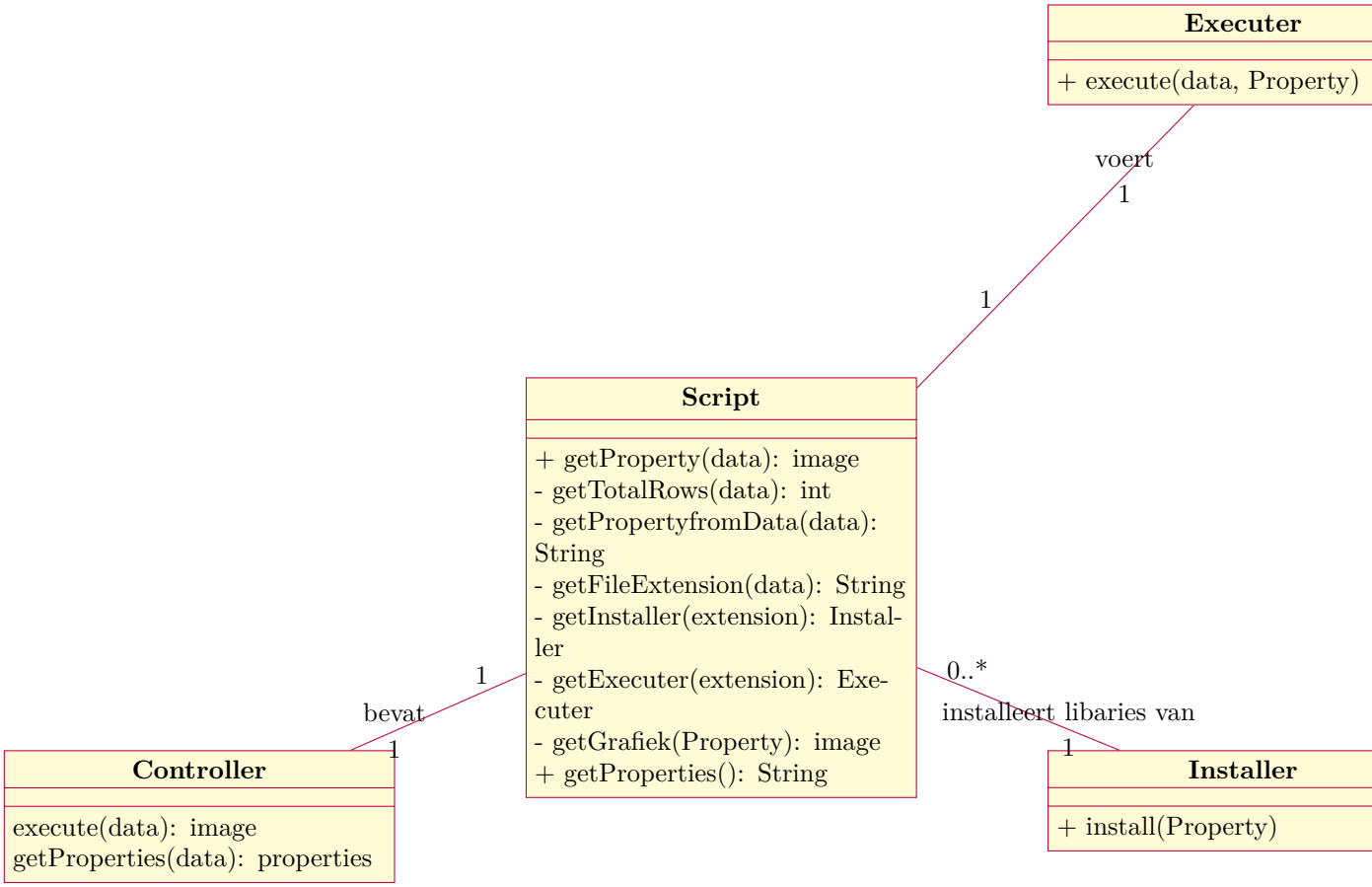
Bij het maken van de sequence diagrammen is er gebruik gemaakt van GRASP. Deze versies van de sequence diagrammen zijn de eerste versies van de applicatie. De design class diagrammen en sequence diagrammen zijn in combinatie met test driven development gebruikt om de applicatie te ontwikkelen. Tijdens het ontwikkelen is het mogelijk dat ik tot nieuwe inzichten ben gekomen. Deze diagrammen komen dus niet exact overeen met de code en zijn alleen bedoeld om de interactie tussen objecten weer te geven.

4.3.1 Design Class Diagram

Het onderstaand design class diagram is gemaakt op basis van de sequence diagrammen "filteren", ïnladenën het domain diagram. Voor het domain diagram (Zie Bijlage P)



Het onderstaand design class diagram is gemaakt op basis van de sequence diagram 'uitvoeren".



Python heeft geen interfaces. Dit effect kan wel worden gesimuleerd door het gebruik van bijvoorbeeld factories. Er worden instanties van executer en installer aangemaakt door middel van een factory. Hierdoor kan de applicatie worden uitgebreid met meerdere talen.

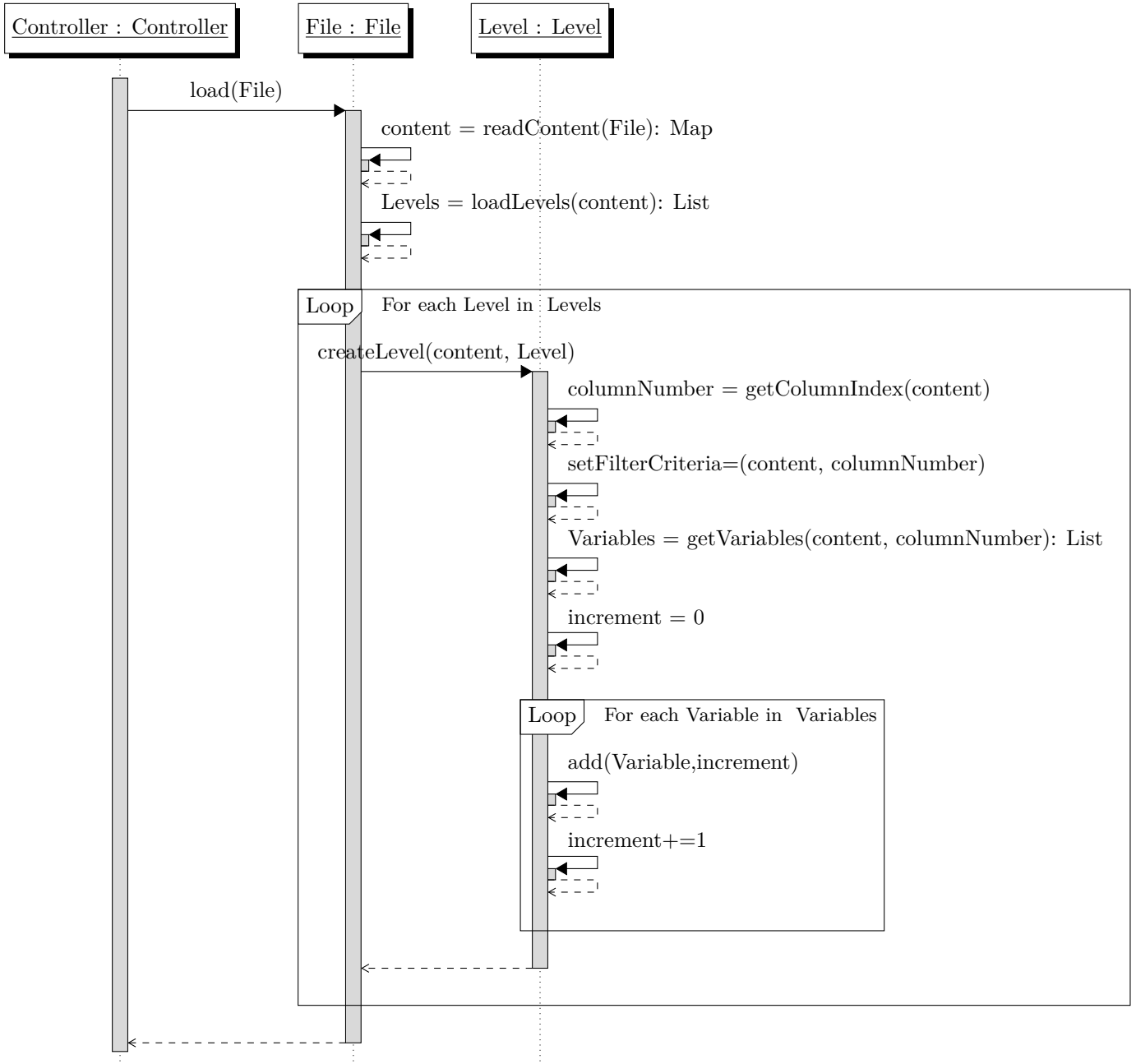
4.3.2 Sequence diagram: Inladen

Hier wordt een bestand ingeladen.

De code kan worden aangeroepen door de url üploadFile"met POST als methode. voor een voorbeeld van de request (Zie Bijlage J).

Het onderstaand diagram is gemaakt op basis van de werkwijze van Cito, de eisen, een prototype en inzichten van de afstudeerder.

Bij het prototype blijkt dat python 64 bit nodig is om de applicatie uit te voeren. Dit is nodig zodat het zoveel mogelijk variabelen in te kan laden. (Powel-Morse, 2018)



Onderzoekers kunnen een Excelbestand met variabelen inladen. Dit bestand bestaat uit kolommen en rijen. In een bestand staan

rijen met variabelen. De eerste rij van een bestand bevat de niveaus. De rijen eronder zijn de variabelen. Deze rijen zijn niet gesorteerd. Omdat de rijen niet gesorteerd zijn, wordt er verwacht dat het filteren lang duurt. Dit blijkt ook uit het inladen van het prototype. (Zie Bijlage F)

Er kan worden gefilterd op variabele. Stel, er wordt gefilterd op "HAN"van het niveau school. In dit geval worden alle rijen variabelen getoond die ook "HANöp niveau school hebben. Het sorteren van de rijen lost het probleem dus niet op, omdat er meerdere rijen zijn met HAN. Als er meerdere rijen zijn, moet er per rij door de variabelen gezocht worden. Het opslaan van individuele variabelen kan niet, omdat het dan niet terug te halen is bij welke rij deze variabel hoort.

Een eis van de opdrachtgever is dat er snel gefilterd kan worden. (Zie Bijlage E) hoofdstuk Architecturally Significant Requirements"

Er moet dus een oplossing gezocht worden die rekening houdt met de eerder genoemde problemen. Hierbij moet er gekeken worden naar datastructuren, omdat er niet gesorteerd kan worden. Op basis van de eerdere problemen is een mogelijke oplossing een binary search tree. Deze garandeert echter niet dat er altijd snel gezocht kan worden. Hierom wordt er gebruik gemaakt van een AVL boom. Dit lost in elk geval het zoeken op, maar dit geeft nog niet een manier hoe de rijen of variabelen in de boom komen. Een rij kan niet in de boom worden ingeladen. Een rij kan niet worden ingeladen, omdat er op variabele kan worden gefilterd. In dit geval zou het zoekproces in een boom veel langzamer worden.

Losse variabelen kunnen niet worden ingeladen. Hierdoor is het niet meer te achterhalen waar de variabelen vandaan komen. Door de losse variabelen een nummer te geven kunnen alle bijbehorende variabelen worden achterhaald. Dit is vergelijkbaar met hoe het in Excel werkt: Een rij van variabelen heeft een rijnummer.

Het is mogelijk dat twee variabelen van verschillende niveaus overlappen. Als een antwoord van een leerling "HANïs en er is een niveau school met "HAN" als variabele. Dit soort situaties kunnen voor problemen zorgen. Om dit te voorkomen moet elk niveau een eigen boom krijgen. Daarnaast wordt het zoeken hierdoor ook sneller omdat er per boom minder variabelen zijn.

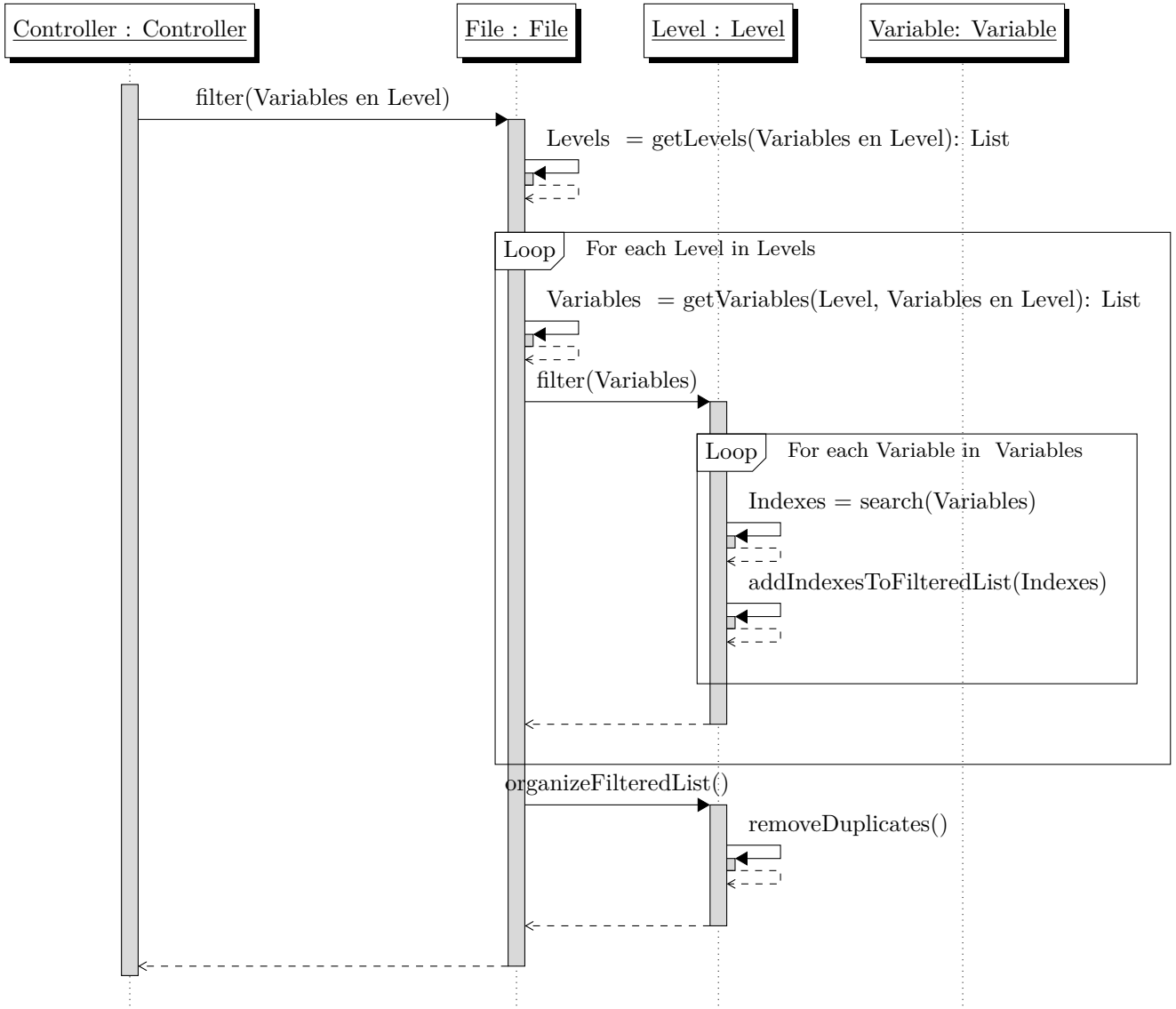
Stel, er wordt alleen gezocht op "HAN"met school als niveau. In dit geval moeten alle rijnummers gevonden worden waarvan het niveau school de variabele "HAN". In dit geval worden alleen de rijnummers gevonden en nog niet de bijbehorende variabelen. Dit is omdat elk niveau een andere boom met variabelen heeft. Nadat de gewenste rijnummers in de boom zijn gevonden worden deze dus gezocht door een map. Door een map met als key een rijnummer en als value een rij van variabelen, wordt het zoeken versnelt. Door deze structuur kan de gebruiker het bestand aanpassen met andere niveaus en variabelen.

Naast een boom en map wordt er bij het inladen gebruik gemaakt van een reader factory. Als de onderzoekers geen gebruik meer willen maken van een Excelbestand of het bestand wordt niet meer ondersteund, kunnen ze relatief gemakkelijk wisselen naar een ander type bestand met kolommen en rijen.

4.3.3 Sequence diagram: Filteren

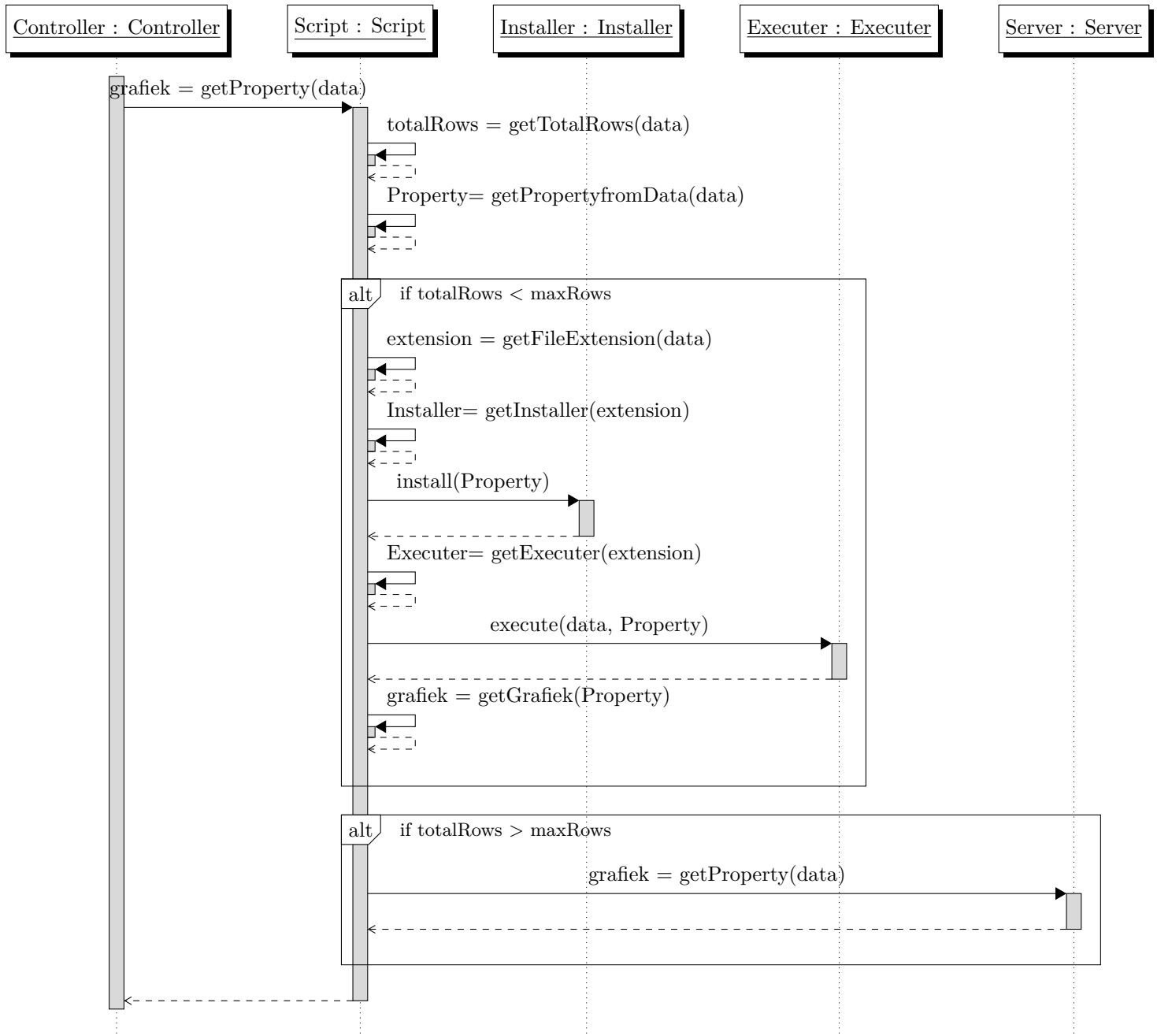
4.3.3.1 Filteren Hier wordt het filter toegepast zoals bij Inladen is uitgelegd. Zie onderstaand diagram voor uitleg.

De code kan wordt aangeroepen door de url ”filter”met POST als methode. Voor een voorbeeld van de request(Zie Bijlage M).


Controller : Controller
File : File
Level : Level
Variable: Variable
filter(Variables en Level)
Levels = getLevels(Variables en Level): List
Loop
For each Level in Levels
Variables = getVariables(Level, Variables en Level): List
filter(Variables)
Loop
For each Variable in Variables
Indexes = search(Variables)
addIndexesToFilteredList(Indexes)
organizeFilteredList()
removeDuplicates()

4.3.4 Sequence diagram: Uitvoeren

Om de eigenschappen van variabelen te achterhalen worden de scripts uitgevoerd. Hiervoor is er een prototype gemaakt. Op basis van dit prototype is het sequence diagram gemaakt. De code kan wordt aangeroepen door de url ”Execute”met POST als methode. Voor een voorbeeld van de request (Zie Bijlage N)


Controller : Controller
Script : Script
Installer : Installer
Executer : Executer
Server : Server
grafiek = getProperty(data)
totalRows = getTotalRows(data)
Property= getPropertyfromData(data)
alt
if totalRows < maxRows
extension = getFileExtension(data)
Installer= getInstaller(extension)
install(Property)
Executer= getExecuter(extension)
execute(data, Property)
grafiek = getGrafiek(Property)
alt
if totalRows > maxRows
grafiek = getProperty(data)

De laptop beschikt soms over te weinig geheugen om de eigenschappen van variabelen te achterhalen. Hiervoor moet het achterhalen van eigenschappen op de server gebeuren. Het is mogelijk om uit te rekenen hoeveel geheugen de variabelen kosten. Dit verschilt echter wel per type variabele. Er zijn tekstuele variabele, audio variabele en video variabele. De hoeveelheid geheugen is afhankelijk van het type en de kwaliteit ervan. Het is mogelijk om automatisch in te schatten hoeveel geheugen het achterhalen van eigenschappen kost. Deze inschatting is alleen niet nauwkeurig, omdat het gebruik ervan afhangt van hoe er wordt geprogrammeerd. Het uitrekenen en inschatten maakt het voor de onderzoekers onnodig complex om te onderhouden en uit te breiden. Daarnaast levert het voor de onderzoekers weinig op. Hierom is ervoor gekozen om dit aanpasbaar te maken via een configuratie bestand. In dit bestand staat erbij hoeveel rijen aan variabelen de eigenschappen op de server worden achterhaald.

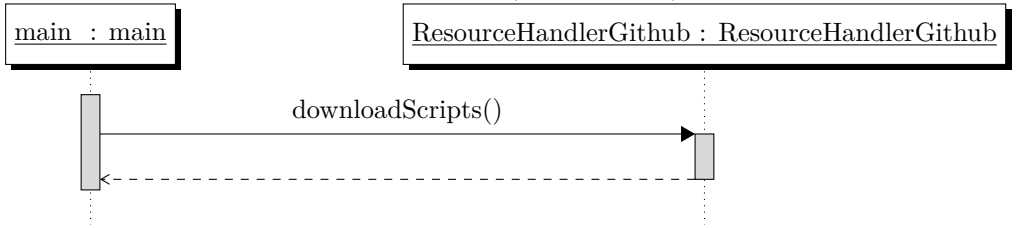
Voor het achterhalen van de eigenschappen moeten de zelf geschreven scripts worden uitgevoerd. Hierbij is één script één eigenschap en is het geschreven in een taal naar keuze. Hiervoor wordt er gebruik gemaakt van een factory.

Een zelf geschreven script maakt soms gebruik van libraries. In dit geval moeten de libraries eerst worden geïnstalleerd voordat het script wordt uitgevoerd. Het installeren hiervan is afhankelijk van de taal waar de script in is geschreven. Om deze reden wordt er gebruik gemaakt van een factory.

Zodra een script is uitgevoerd wordt er een plaatje gegenereerd. Dit is een onderdeel van de zelfgeschreven scripts. Hiervoor is er een vaste structuur nodig. Een map waar de scripts in staan is hierin noodzakelijk. Per script moet er een map zijn met libraries, een map voor het plaatje en het script zelf. (Zie Bijlage O)

4.3.5 Sequence diagram: Updaten

Hier wordt het sequence diagram besproken die verantwoordelijk is voor het updaten van scripts. Voor het maken van dit sequence diagram is er een prototype gemaakt. (Zie Bijlage H) Op basis hiervan is het diagram gemaakt.



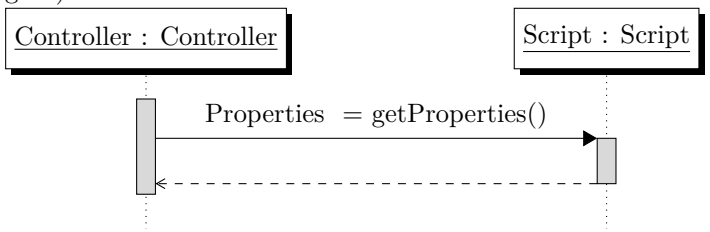
De code wordt uitgevoerd bij het opstarten van de applicatie.

4.3.6 Sequence diagram: Ophalen van scripts

De onderzoeker moet een script kunnen kiezen. Hierom worden de scripts die beschikbaar zijn in de applicatie gevonden en doorgestuurd naar de frontend.

Voor het uitvoeren van een script is er een vaste structuur gekozen. Op basis hiervan worden de scripts uitgevoerd en worden de bijbehorende libraries geïnstalleerd. Voor een voorbeeld van de structuur (Zie Bijlage O). Op basis hiervan is het sequence diagram gemaakt.

De code kan worden aangeroepen door de url "getProperties"met GET als methode. Voor een voorbeeld van de request(Zie Bijlage L).



5 Bronnenlijst

Angular. (z.d.). Introduction to modules. Geraadpleegd op 6 april 2020, van https://angular.io/guide/architecture-modules

Rajput, M. (2016, 18 maart). The pros and cons of choosing AngularJS. Geraadpleegd op 6 april 2020, van https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html

Powel-Morse, A. (2018, 24 januari). Python Exception Handling – MemoryError. Geraadpleegd op 6 april 2020, van https://airbrake.io/blog/python-exception-handling/memoryerror

6 Bijlagen

Bijlage A : Schrijftaak
Bijlage B : AntwoordenLeerlingen
Bijlage C : Onderzoek
Bijlage D : Interview
Bijlage E : Software architecture document
Bijlage F : Prototype Memory
Bijlage G : Prototype Script
Bijlage H : Prototype Github
Bijlage I : Prototype Frontend
Bijlage J : request Load variables
Bijlage K : request get criteria
Bijlage L : request get properties
Bijlage M : request filter
Bijlage N : request execute
Bijlage O : scripts
Bijlage P : functioneel ontwerp