5-6-2014

Versie 1.0

(Tussen)producten

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Inleiding

Dit document bevat alle (tussen)product documenten. Elk (tussen)product heeft een eigen voorblad, inhoudsopgave etc. Dit hoofddocument is dus puur als inleidend stuk toegevoegd.

De volgende documenten komen in deze volgorde aan bod:

DOCUMENT	VERSIE	AANTAL PAGINA'S
PLAN VAN AANPAK	1.6	13
ONDERZOEK SOCIAL MEDIA	1.3	42
REQUIREMENTS	1.6	17
ONTWERP	1.7	45
TEST RAPPORTAGE	1.0	27

5-6-2014

Versie 1.6

Plan van aanpak

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Wijzigingen

- Document opmaak aangepast

Inhoudsopgave

1 Inleiding

Dit document is het plan van aanpak voor de afstudeeropdracht van Stijn van der Meer, en heeft direct betrekking op het project “Social media plug-in module”.

1.1 Tekstopmaak uitleg

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.

Dit document houdt verschillende termen aan. Betekenissen en/of uitleg van deze termen staan vermeld in het document **Afstudeerverslag**, bijlage 2: Begrippenlijst.

2 Opdracht

2.1 Stakeholders

De volgende stakeholders hebben belangen bij dit project:

- Geert Merkelbach, opdrachtgever & directeur Webbeat
- Eddie Horde, begeleider en verantwoordelijk voor Datumprikker
- Stijn van der Meer, afstudeerder
- Social media:
    - Facebook
    - LinkedIn
    - Google
    - Microsoft

Hoewel al deze stakeholders (in wisselende zwaarte) belangen hebben bij dit project, hebben alleen de eerste 3 daadwerkelijk directe invloed op het project. De social media hebben ook directe invloed op het project, maar zullen niet direct communiceren met/binnen de projectgroep.

2.2 Probleemstelling

Er komt steeds meer de vraag van gebruikers voor de mogelijkheid om sociale media met Datumprikker te kunnen gebruiken. Daarnaast zijn concurrerende partijen reeds bezig om deze sociale media integratie toe te passen.

Er is op dit moment bij Datumprikker dus nog geen mogelijkheid voor dit soort integratie, en om er voor te zorgen dat klanten niet voor de concurrent zullen kiezen, zal er dus moeten worden voldaan aan de wensen van de klanten.

Omdat er echter geen tijd is voor Webbeat om deze integratie te implementeren zal dit via de afstudeeropdracht gedaan worden.

2.3 Doelstelling

De doelstelling van dit project is om een plug-in module te ontwerpen en te bouwen die het mogelijk maakt om sociale media te integreren met Datumprikker. Deze module zal generieke functies bevatten zodat het bedrijf makkelijk externe social media API's (SmA's)[1] kan integreren. Deze API's zullen dan gebruik maken van deze functies, zodat een gebruiker bijvoorbeeld kan inloggen met een extern product (bijvoorbeeld Facebook), of zijn/haar agenda kan synchroniseren (Google Calender).

Voordat de plug-in module gebouwd kan worden, zal er eerst een vooronderzoek gedaan worden. In dit onderzoek zal er gekeken worden naar de functionele eisen van verschillende SmA's. Aan het einde van het onderzoek zal er een schema zijn met alle onderzochten API's en de functionele en kwaliteits- eisen die hier aan verbonden zijn. Op basis van dit schema kunnen er dan keuzes gemaakt worden voor de requirements.

Op basis van de bevindingen van dit vooronderzoek, zal er, tezamen met de opdrachtgever, gekeken worden aan welke eisen de plug-in module moet voldoen. Ook is het mogelijk dat de opdrachtgever nog extra wensen/eisen heeft. Al deze eisen/wensen zullen dan verwerkt worden doormiddel van het opstellen van de requirements.

Naast het bouwen van de plug-in module, zullen er ook ten minste 3 verschillende implementaties[2] worden gecreëerd om te testen of de module aan de functionele en kwalitatieve requirements voldoen. Tevens kunnen deze 3 integraties gebruikt worden als blauwdruk voor latere integraties.

2.4 Eisen afstudeeropdracht

Er zijn een aantal eisen voor de afstudeeropdracht:

- De student kan, met behulp van vakliteratuur, zelfstandig (een deel van) een project uitvoeren, dat aansluit bij het beroeps specifieke deel van de major van de opleiding.[3]
- De student toont een selectie van een realistisch aantal van zijn majorcompetenties/beroepstaken aan op hbo-eindniveau, i.e. minimaal niveau 3, in de betekenis waar in de AGO deze heeft beschreven in het Competentiemodel.[3]
- De student laat tijdens zijn afstuderen zien over het vermogen te beschikken tot zelfregie en transfer (het geleerde kunnen toepassen in een andere context).[3]
- Het afstudeerverslag zal uiterlijk 6 juni in drievoud opgeleverd worden.

[1] De externe API's komen van externe bedrijven als Facebook, LinkedIn, Google, etc.
[2] Er zullen ten minste de volgende typen geïmplementeerd worden: login, contacten synchronisatie, afspraak synchronisatie.
[3] Bron: Uitvoeringsreglement afstuderen definitieve versie 03 (dat.20111108), Haagse Hogeschool – Academie ICT & Media

2.5 Scope

De scope binnen dit project betreft het ontwerpen en bouwen van de plug-in module zelf, en de eventuele daarbij betrokken methode aanroep naar deze module vanuit Datumprikker webpagina's. Tevens dienen er implementaties (de zogeheten plug-ins) van deze module gemaakt te worden.

De ontwerpen kunnen buiten de scope onderdelen bevatten, mits deze onderdelen invloed hebben op onderdelen binnen de scope. Tevens moeten deze buiten de scope onderdelen zo globaal mogelijk gehouden worden.

Daarnaast zal er documentatie gemaakt worden over de plug-in module, en zal er een eindverslag over het gehele project gemaakt worden voor de Haagse Hogeschool.

Alles wat hier buiten valt is, maar niet gelimiteerd tot:

- Database aanpassingen (zal in overleg door Webbeat zelf gedaan worden).
- Datumprikker aanpassingen buiten de plug-in module, implementaties en methode aanroeping naar de plug-in methode.
- Webpagina aanpassingen (behalve de methode aanroeping naar de plug-in methode of proof-of-concept pagina's).
- Andere projecten van Webbeat.

Het project zal van 10 februari 2014 tot uiterlijk 6 juni 2014 duren, waarvan ten minste 2 weken gereserveerd zullen worden voor het opstellen van het afstudeerverslag. Het (deels) live gaan van het eindproduct mag binnen het project plaatsvinden, maar is geen verplichting.

2.6 (tussen)producten

De volgende (tussen)producten zullen in de loop van dit project aangemaakt en geüpdatet worden:

- Plan van Aanpak
- Vooronderzoek verslag
- Functionele en niet-functionele requirements
- Ontwerp
- Test rapportage
- Plug-in module
- SmA implementaties
- Afstudeerverslag

2.7 Resultaat

Na het voltooien (dus inclusief testen) van de plug-in module, zal het bedrijf allerlei externe API's makkelijk kunnen integreren met Datumprikker. Omdat sociale media platformen relatief vaak van populariteit verandert, maakt het feit dat de externe API's makkelijk toegevoegd en verwijderd kunnen worden, essentieel.

3 Uitvoering

Om de in het vorige hoofdstuk omschreven opdracht uit te kunnen voeren, zal er gebruik gemaakt worden van een aantal methoden/hulpmiddelen. Deze worden in dit hoofdstuk beschreven.

3.1 Software ontwikkelmethode

Als ontwikkelmethode zal in grote lijnen Scrum aangehouden worden. Dit is een methode waarbij er op een flexibele manier in korte sprints (1-4 weken) software ontwikkeld wordt. Niet alle onderdelen van deze methode passen echter bij dit project. Deze aanpassingen staan vermeld in **Aanpassingen op Scrum**.

3.1.1 Agile

Scrum valt onder de Agile softwareontwikkeling. Dit concept heeft een aantal principes en richtlijnen. Deze staan opgesteld in het Agile Manifesto[4]:

- Richtlijnen
  - Personen en interacties boven processen en tools.
  - Software die werkt boven lijvige documentatie.
  - Samenwerking met de klant boven onderhandeling over het contract.
  - Omgaan met verandering boven het volgen van een plan.
- Principes
  - Klanttevredenheid, door snelle, continue levering van bruikbare software.
  - Zelfs late veranderingen in de requirements zijn welkom.
  - Werkende software wordt regelmatig geleverd (weken eerder dan maanden).
  - De ontwikkelaars werken nauw en dagelijks samen met de mensen die de business kennen.
  - Projecten steunen op gemotiveerde en betrouwbare personen.
  - Een gesprek in levenden lijve is de beste manier van communicatie, wat betekent dat men zich best op dezelfde plek bevindt.
  - Werkende software is de eerste maatstaf van vooruitgang.
  - De ontwikkeling kan te allen tijde worden voortgezet.
  - Er is voortdurende aandacht voor technische uitmuntendheid en goed ontwerp.
  - Eenvoud is belangrijk: hoe meer er niet gedaan wordt, hoe beter.
  - De teams organiseren zichzelf.
  - Men past zich aan aan de omstandigheden.

[4] Bron: “Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process”, ISBN: 978-0-471-20282-0

3.1.2 Scrum[5] (overeenkomend met dit project)

De belangrijkste kenmerken van een Scrum project zijn als volgt:

- Lichtgewicht
- Makkelijk te begrijpen
- Moeilijk te beheersen
- Pilaren:
  - Transparantie
  - Inspectie
  - Adaptatie
- Werken in (korte) sprints (iteraties)
- Dagelijkse meeting
- Elke sprint levert een (werkend) product op
- De teams zijn:
  - Kleinschalig
  - Flexibel
  - Creatief
  - Productief
- Communicatie is belangrijker dan documentatie

Daarnaast wordt de Agile Manifesto aangehouden.

3.1.2.1 Rollen

Scrum kent 3 (hoofd)rollen. Deze worden hieronder uitgelegd.

3.1.2.1.1 Product-owner

De product-owner (product eigenaar) is de opdrachtgever / klant, en heeft dus het meeste belang bij het (software)product dat gemaakt wordt. Hij / zij beheert ook de productbacklog, en bepaalt wat er moet gebeuren en in welke volgorde. In principe wordt begonnen met het belangrijkste, waar het meeste voordeel mee te behalen is. Dit staat boven aan de productbacklog. De product-owner zorgt er tevens voor dat de rekening betaald wordt.

Deze rol zal ingevuld worden door de stakeholder Geert Merkelbach.

3.1.2.1.2 Ontwikkelteam

Het ontwikkelteam is multidisciplinair samengesteld en is verantwoordelijk voor het afleveren van het (software)product aan het einde van elke sprint. Het team bestaat meestal uit 3 tot 9 personen. Het team organiseert zichzelf. Zij doen de analyse, ontwerp, ontwikkeling, test en documentatie en zorgen dat er aan het eind van de sprint een kant-en-klaar product is, dat in principe in productie genomen kan worden.

Deze rol zal ingevuld worden door de stakeholder Stijn van der Meer.

[5] Bron: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf

3.1.2.1.3 Scrum-master

De Scrum-master begeleidt / helpt het team door te zorgen dat het juiste scrumproces gevolgd wordt. Hij verzorgt ook eventuele trainingen. De Scrum-master regelt alle vergaderingen. Ook regelt hij de voorzieningen zoals een werkruimte, hardware en software. De Scrum-master zorgt ervoor dat het team niet lastig gevallen wordt door derden die met extra eisen tussendoor komen of die bijvoorbeeld tijdelijk mensen nodig hebben uit het team. De Scrum-master is geen projectmanager. Hij regelt bijvoorbeeld niet de personele zaken zoals selectie, beoordeling en beloning van de mensen. Dit bevordert de openheid en samenwerking.

Deze rol zal ingevuld worden door de stakeholder Eddie Horde.

3.1.2.2 Werkwijze

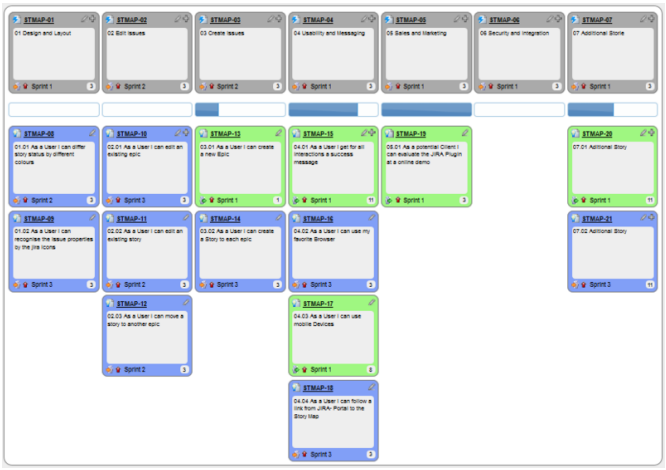
Bij Scrum wordt er zo veel mogelijk op een gezamenlijke ruimte gewerkt, waardoor er zo veel en gemakkelijk mogelijk gewerkt en overlegt kan worden binnen het Scrum team (bestaande uit de rollen beschreven in het vorige paragraaf). Per sprint levert dit team een werkend product/onderdeel op. Deze oplevering heeft betrekking op de door de product-owner gekozen eisen uit de productbacklog. Deze eisen worden gezet in een story map.

3.1.2.2.1 Story map

Een story map is de grafische weergave van de productbacklog. Per sprint is er 1 hoofdeis (epic), met daaronder de bijbehorende andere eisen, in volgorde van belangrijkheid.

Hieronder staat een voorbeeld van een story map. Deze is opgebouwd uit (van boven naar beneden):

- Epics
- Statusbalk (hoe ver de bouw/implementatie is)
- Requirements (het verschil tussen blauw en groen is hier onduidelijk, maar bij dit project zal er een legenda opgesteld worden)



Figuur 1, voorbeeld van een story map[6]

[6] Bron: http://upload.wikimedia.org/wikipedia/commons/thumb/b/bd/User_Story_Map_in_Action.png/800px-User_Story_Map_in_Action.png

3.1.3 Aanpassingen op Scrum

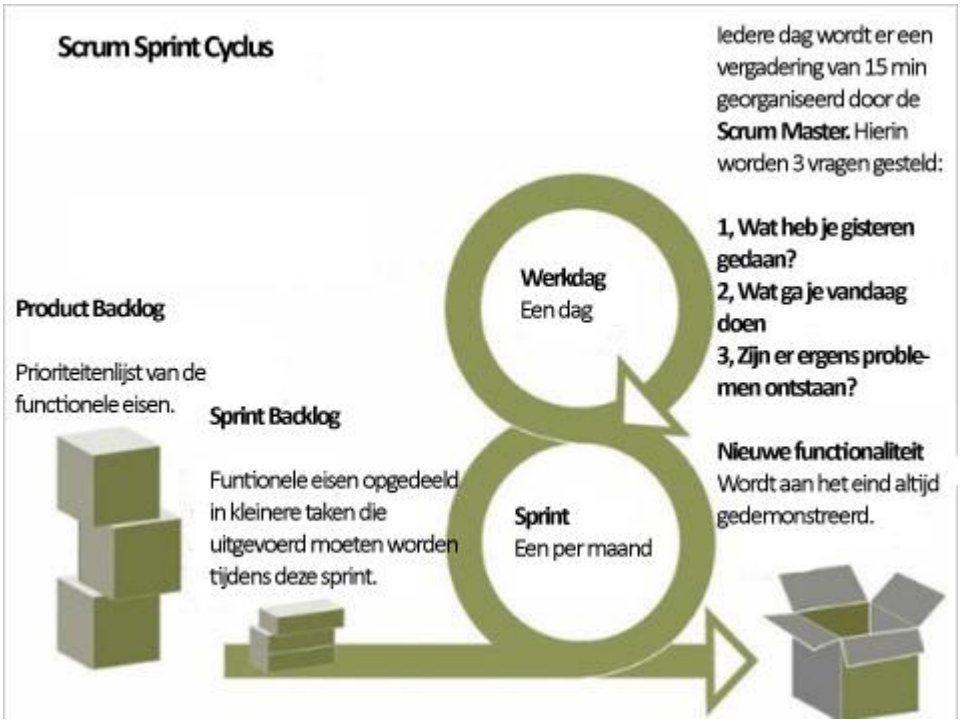
Zoals reeds gemeld in het begin van dit hoofdstuk, wordt niet alles van Scrum gebruikt. De onderdelen die anders zijn, worden in deze paragraaf omschreven.

3.1.3.1 Ontwikkelteam

In dit project zal het "team", in tegenstelling tot de standaard, bestaan uit alleen de afstudeerder. Dit is namelijk 1 van de eisen voor het afstudeerproject (**Eisen afstudeeropdracht**). Wel zijn de medewerkers van Webbeat beschikbaar voor vragen en diens expertise.

3.1.3.2 Sprint cyclus

De standaard sprint cyclus staat hieronder afgebeeld:



Figuur 2, standaard sprint cyclus[7]

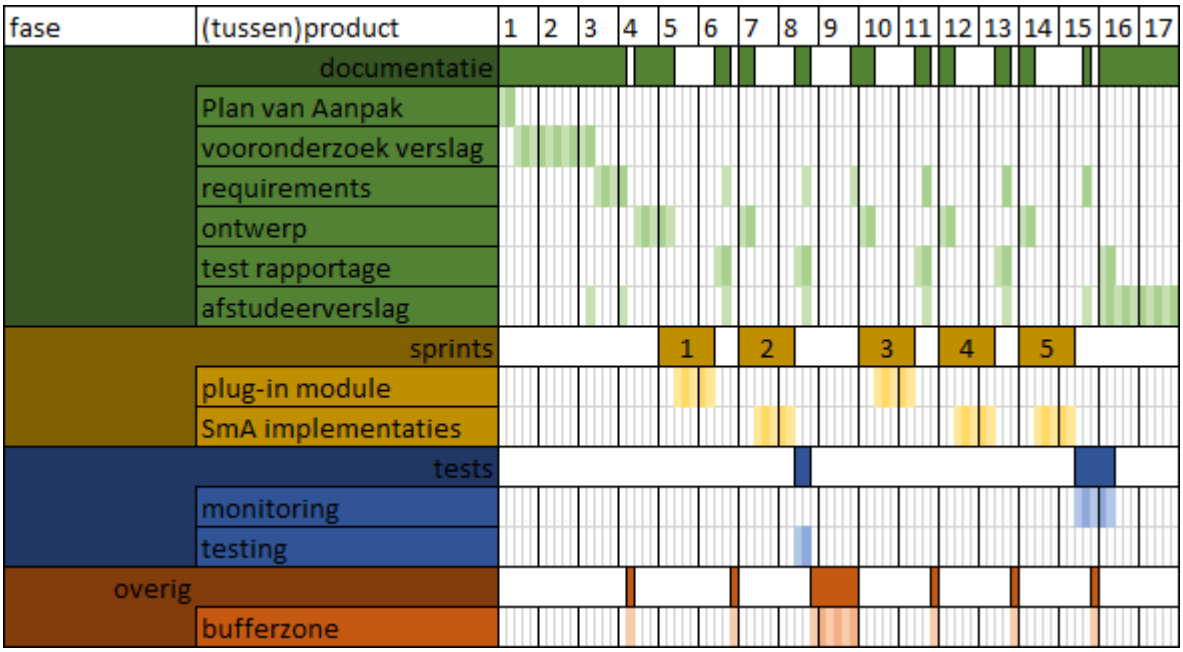
Bij dit project zal er echter niet elke werkdag een meeting plaatsvinden. Dit heeft vooral te maken met het feit dat de Scrum master meerdere projecten/werkzaamheden heeft, waardoor hiervoor simpelweg geen tijd is. Er zal dan ook maar eens per week een vergadering georganiseerd worden.

Dit houdt in dat de afstudeerder zelfstandiger te werk zal moeten gaan, en met behulp van zelf opgestelde planning de doelen moet bereiken. Dit is positief gezien dit 1 van de eisen van de afstudeeropdracht bevorderd (**Eisen afstudeeropdracht**).

[7] Bron: http://upload.wikimedia.org/wikipedia/commons/b/bb/NL-scrum.jpg

3.2 Planning

De hier onderstaande planning zal aangehouden worden voor dit project:



Figuur 3, planning

Er zullen dus 5 sprints gedaan worden nadat het vooronderzoek is uitgevoerd, en de (eerste) requirements zijn opgesteld. Van deze sprints zullen er 3 gebruikt worden om de plug-in module te bouwen, en 2 om de implementaties te bouwen en implementeren.

Daarnaast zal er (indien hier tijd voor is) wat monitoring gedaan worden. Dit zal alleen gedaan worden wanneer Webbeat besluit om de plug-in module op de live omgeving te zetten. Door de module live te gebruiken met Datumprikker, kan er goed gekeken worden naar wat de gebruiker van de nieuwe functionaliteit vindt. Hierdoor kunnen er extra requirements naar voren komen. Het zal dus een (live) acceptatie test zijn. Voordat dit gedaan kan worden, zal de module dus goed genoeg aan de geselecteerde (tot dan) opgestelde requirements moeten voldoen.

Aan het einde van elke sprint (en na het afronden van het vooronderzoek en de requirements), zal er een stuk geschreven worden in het afstudeerverslag. Tevens zal er nogmaals gekeken worden naar de requirements om te controleren of er hier (al) aan voldaan is. Ook kunnen er op deze momenten toevoegingen en/of wijzigingen gemaakt worden.

Omdat het altijd lastig is om bij een project in te schatten hoeveel tijd er exact nodig is voor alle onderdelen, zijn er ook bufferzones ingepland. Deze zones moeten ervoor zorgen dat er geen tijdgebrek optreedt wanneer onderdelen langer blijken te duren dan verwacht.

Aan het einde van week 17 moet het afstudeerverslag af zijn en ingeleverd worden bij de examinatoren. Vandaar ook dat er voor deze deadline 2 weken zijn ingepland voor dit verslag.

5-6-2014

Versie 1.0

Onderzoek Social Media

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Wijzigingen

- Pagina opmaak aangepast

Inhoudsopgave

1 Inleiding

Dit document bevat alle onderzoeken die gedaan zijn ten behoeve van dit project. De resultaten van deze onderzoeken dienen als advies en input voor het te maken project, en voor de applicatie Datumprikker.

De volgende onderzoeken zijn uitgevoerd:

- Eisen en beperkingen van de API's bij web gebruik
- Eisen en beperkingen van de API's bij mobile gebruik
- Aanpassingen Datumprikker voor gebruik van module

1.1 Tekstopmaak uitleg

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.
- **Variabele**; dit is een verwijzing naar een variabele die gebruikt wordt binnen een bepaalde context (bijvoorbeeld binnen een **afbeelding**).
- **Package**; dit is een **variabele** dat een package voorstelt. De reden dat er hier onderscheid gemaakt wordt, is omdat er in dit document vaak een package variabele wordt gebruikt. Daarnaast bevat een package (vaak) andere variabelen.

1.2 Leeswijzer

De onderzoeken zijn in 3 hoofdstukken verdeeld:

- **Social media API's web-gebruik**
- **Mobile (app) gebruik**
- **Aanpassingen Datumprikker**

Dit document houdt verschillende termen aan. Betekenissen en/of uitleg van deze termen staan vermeld in het document **Afstudeerverslag**, bijlage 2: Begrippenlijst.

2 Social media API's web-gebruik

Om een gewogen beslissing te kunnen maken over welke eisen er aan de social media plug-in module gesteld zullen worden, moet er eerst onderzoek gedaan worden naar de Social media API's (SmA). Het belangrijkste hierbij is het achterhalen van de eisen, mogelijkheden, en beperkingen die hierbij horen.

De hieronder staande SmA's zijn onderzocht:

- **Facebook**
- **LinkedIn**
- **Google** (**Plus** & **Calendar**)
- **Microsoft** (**Office 365** & **Live/Hotmail**)

Per paragraaf zullen de onderzochte Social Media API's (SmA's) behandeld worden. Daarbij zal de volgende structuur aangehouden worden:

1) Algemene informatie over bedrijf/product.
2) Welke talen en methodes de SmA ondersteund.
3) Welke eisen er worden gesteld voor het gebruik van de SmA.
4) Wat er gedaan kan worden met de SmA.
5) Wat de (mogelijke) nadelen zijn voor het gebruik van de SmA.
6) Bronnen/verdere informatie over de SmA.

Tijdens het onderzoek zijn er een aantal methodes naar voren gekomen. Deze zullen ook behandeld worden. Het gaat hierbij om de volgende methodes:

- **OAuth**
- **JSON**
- **REST**

2.1 Onderzochte social media

2.1.1 Facebook

'Facebook is een gratis sociaalnetwerksite om online contact te maken of houden. Gebruikers kunnen er een persoonlijk profiel samenstellen en informatie en interesses delen op het zogeheten "Prikbord" (in het Engels "Wall").

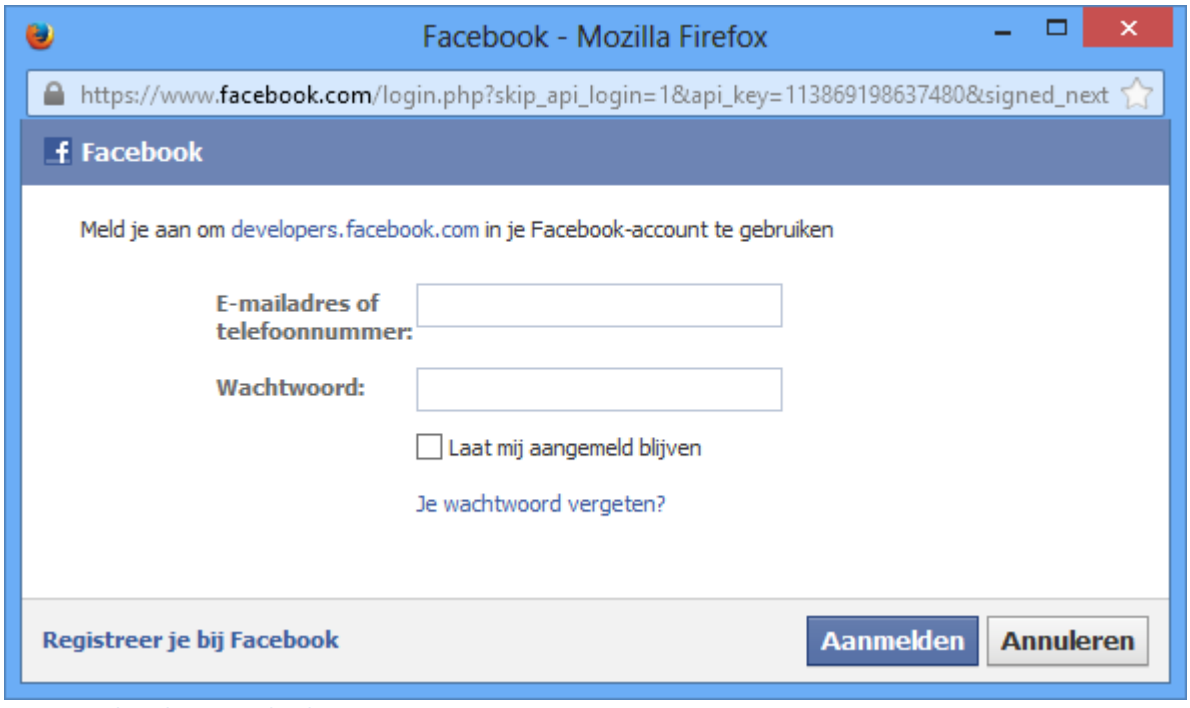
Sinds mei 2008 bestaat er een Nederlandstalige versie. In december 2008 groeide de site met 600.000 nieuwe aanmeldingen per dag. In april 2009 had Facebook meer dan 200 miljoen actieve gebruikers, vijf maanden later waren dat er 50 miljoen meer. In juli 2010 bediende Facebook een half miljard gebruikers, circa zeven procent van het aantal aardbewoners. In 2012 had Facebook ongeveer 1 miljard gebruikers. Het is te gebruiken vanaf 13 jaar.'

- Bron: http://nl.wikipedia.org/wiki/Facebook

Indien de SmA geïmplementeerd wordt, dan komt de volgende button op het login scherm:

Figuur 4, login knop Facebook[8]

Deze knop linkt door naar de webpagina van Facebook, die het hieronder staande scherm toont:



Figuur 5, inlogscherm Facebook

De gebruiker is na het invullen van de gegevens (en het drukken op de knop 'Aanmelden') ingelogd, waarna de applicatie gebruik kan maken van het account.

[8] Bron: https://developers.facebook.com/docs/plugins/login-button

2.1.1.1 Welke talen en methodes worden er ondersteund?

Facebook heeft een (eigen) standaard implementatie met de volgende talen[9]:

- Javascript
- PHP

Daarnaast zijn er de volgende 3rd party implementaties[9]:

- .NET
- HTML5
- Javascript
- Python
- Ruby
- Wordpress

Ook ondersteund Facebook de volgende methodes:

- JSON
- OAuth V1a
- OAuth V2

2.1.1.2 Welke eisen worden er gesteld?

Om de Facebook SmA te kunnen implementeren, moet er een developer account bij Facebook aangemaakt worden.

2.1.1.3 Wat zijn de mogelijkheden?

Facebook is niet duidelijk over wat er exact gedaan kan worden nadat iemand inlogt via de SmA. Wel staan de volgende variabelen op de site vermeld als toegankelijk (er staat niet bij of er voorwaarden aan verbonden zijn):

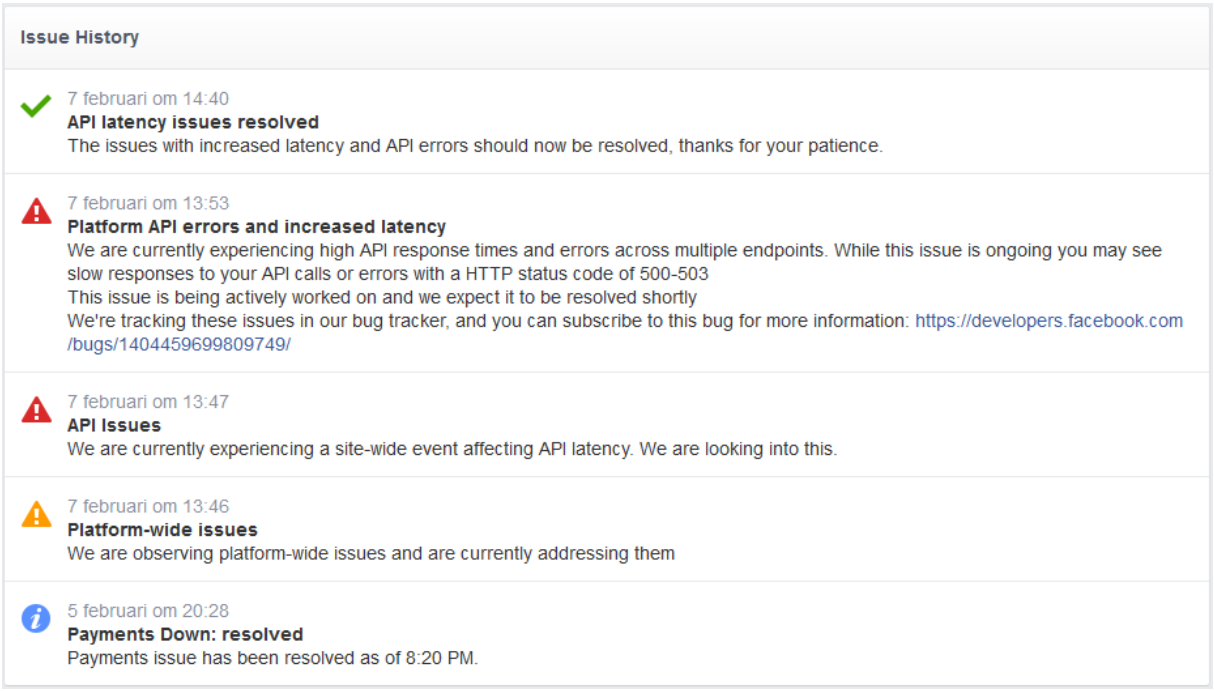
- Id
- Naam
- Geslacht
- Link (pagina link?)
- Access-token (hiermee zou (tijdelijk) meer functionaliteit gebruikt kunnen worden)

Dit houdt concreet in dat er dus ingelogd kan worden in Datumprikker met behulp van Facebook, en dat er dan data opgevraagd kan worden van de persoon die inlogt. Wat deze data precies is, is alleen niet geheel duidelijk.

[9] De talen die niet van belang zijn voor een web-applicatie zijn weggelaten.

2.1.1.4 *Wat zijn de (mogelijke) nadelen?*

Facebook geeft niet aan of er (bijvoorbeeld bij bepaalde talen) bekende problemen/nadelen zijn. Wel is er een status pagina waar onder andere de geschiedenis van problemen vermeld staat. In deze geschiedenis stond toevallig iets over de API's:



Figuur 6, Facebook issue history[10]

Het komt dus wel eens voor dat de SmA eruit ligt. Hierdoor kan er dan op dat moment geen gebruik van gemaakt worden.

2.1.1.5 *Bronnen/verdere informatie*

De volgende URL's bieden verdere informatie (en zijn daarnaast gebruikt als bron):

- https://developers.facebook.com/docs/other-sdks
- https://developers.facebook.com/docs/facebook-login/login-flow-for-web/
- https://developers.facebook.com/docs/graph-api/using-graph-api

[10] Bron: https://developers.facebook.com/status/ (11-02-2014)

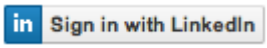
2.1.2 LinkedIn

'LinkedIn is een online sociaal netwerk, actief sinds 5 mei 2003, dat gericht is op vakmensen. Sinds 13 juni 2013 zijn er wereldwijd ongeveer 225 miljoen geregistreerden. LinkedIn groeit de laatste jaren zeer sterk. In 2007 was het het snelst groeiende sociale netwerk in de VS met 189% groei. Nederland staat met 4 miljoen registraties in de top tien. Het hoofdkantoor staat in Mountain View (Santa Clara County), Californië met filialen in Amsterdam, Bangalore, Chicago, Delhi, Dublin, Londen, Los Angeles, Madrid, Melbourne, Milaan, Mumbai, München, New York, Omaha, Parijs, San Francisco, São Paulo, Singapore, Stockholm, Sydney, Tokio en Toronto.

Het belangrijkste doel van de website is geregistreerden gebruik te laten maken van elkaars (zakelijke) netwerk. Dat gebeurt door contacten te leggen met anderen die je vertrouwt.'

- Bron: http://nl.wikipedia.org/wiki/LinkedIn

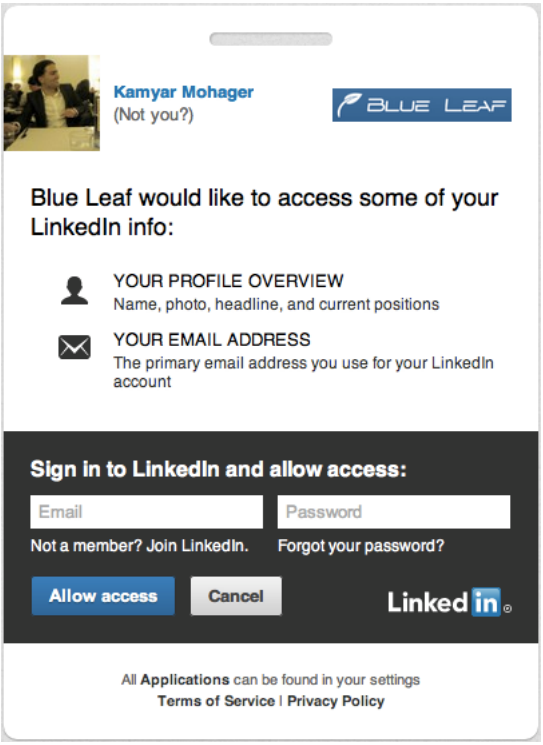
Indien de SmA geïmplementeerd wordt, dan komt de volgende button op het login scherm:



Figuur 7, login knop LinkedIn(Engelse versie)[11]

Als de gebruiker hierop klikt, dan wordt er doorverwezen naar de webpagina van LinkedIn, met daarop een scherm waarmee de gebruiker kan inloggen, en accepteren dat de applicatie gebruik gaat maken van de account gegevens.

Hieronder staat een voorbeeld van zo'n scherm:



Figuur 8, inlogscherm LinkedIn

[11] Bron: https://developer.linkedin.com/sites/default/files/js-signin.png

2.1.2.1 Welke talen en methodes worden er ondersteund?

LinkedIn ondersteund daarnaast de volgende talen[12]:

- Javascript (geprefereerd omdat er hiervoor een standaard implementatie is)
- .NET
- PHP
- Python
- Ruby

Daarnaast worden de volgende methoden ook ondersteund:

- JSON
- REST
- OAuth V1a
- OAuth V2 (wordt geprefereerd)

2.1.2.2 Welke eisen worden er gesteld?

Om de LinkedIn SmA te kunnen implementeren, dient de applicatie geregistreerd te worden.

Opmerking
Sinds april 2014 heeft LinkedIn de beveiliging opgeschroefd[13]. Dit betekend dat er nu (net zoals bij de andere SmA's) een "redirect URL" moet aangegeven worden. Voorheen hoefde dit niet.

2.1.2.3 Wat zijn de mogelijkheden?

De hieronder staande data kan van een gebruiker afgehaald worden nadat deze is ingelogd met LinkedIn. Elk item van de lijst valt onder een andere permissie waarvoor de gebruiker (in zijn geheel) toestemming moet geven

- Profiel overzicht; naam, foto, headline, huidige banen
- Volledig overzicht; werkervaring, opleiding, skills, recommandaties, en de hierboven genoemde data
- Email adres; het primaire emailadres dat bij LinkedIn gebruikt wordt
- Connecties; 1e- en 2e-graad connecties
- Contact info; adres, telefoonnummer, verbonden accounts
- Netwerk updates; ontvangen en verzenden van LinkedIn namens de gebruiker
- Bedrijf pagina en analytics; bedrijf pagina's van gebruiker aanpassen en statusupdates versturen namens die bedrijven
- Groep discussies; ontvangen en verzenden van groep discussies namens de gebruiker
- Uitnodigingen en berichten; berichten en uitnodigingen verzenden namens de gebruiker

2.1.2.4 Wat zijn de (mogelijke) nadelen?

LinkedIn geeft geen mogelijke problemen of nadelen aan. Wel is het verstandig om van te voren goed in te schatten welke permissies er nodig zullen zijn voor het gebruik van de applicatie. Hoe minder permissies, des te minder ingrijpend het is voor de gebruiker.

[12] De talen die niet van belang zijn voor een web-applicatie zijn weggelaten.
[13] Bron: https://developer.linkedin.com/blog/register-your-OAuth-2-redirect-urls

2.1.2.5 Bronnen/verdere informatie

De volgende URL's bieden verdere informatie (en zijn daarnaast gebruikt als bron):

- http://developer.linkedin.com/rest
- http://developer.linkedin.com/documents/libraries-and-tools
- http://developer.linkedin.com/documents/authentication
- http://developer.linkedin.com/documents/sign-linkedin
- Voorbeelden/tutorials voor implementatie:
    - PHP/Ruby: http://developer.linkedin.com/documents/code-samples
    - Javascript: http://developer.linkedin.com/documents/tutorials

2.1.3 Google

Google heeft veel online producten. Al deze producten worden gekoppeld aan één account. Dit maakt het gemakkelijk voor zowel de gebruikers als voor bedrijven die SmA's willen implementeren. Elk product van Google heeft echter een andere API, die elk andere dingen kan. Het koppelen blijft echter volgens hetzelfde principe gaan.

In dit paragraaf zal daarom zoveel mogelijk globaal blijven, maar waar nodig zal er specifiek naar 2 producten van Google gekeken worden (Plus en Calendar). Deze producten worden hieronder kort beschreven.

3.2.1.1.1 Plus

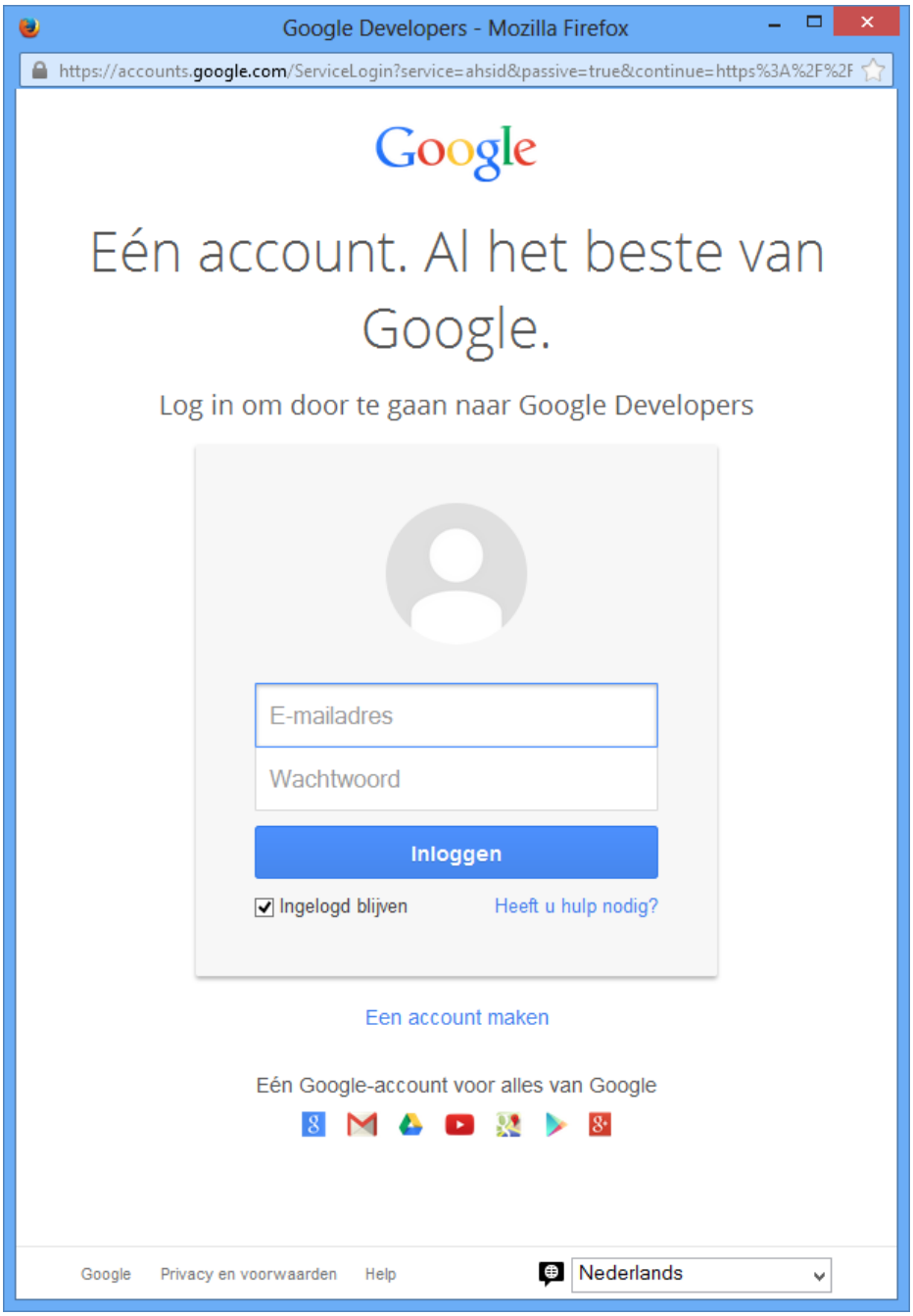
'Google+ (ook bekend als Google Plus of G+) is een sociaalnetwerksite die draait op Google-technologie en die door Google wordt beheerd en uitgebaat. Google lanceerde Google+ in 2011. De lancering werd door verschillende experts gezien als een poging om de dominante positie van Facebook aan te vallenhttp://nl.wikipedia.org/wiki/Google%2B - cite_note-1, maar dat heeft Google nooit bevestigd. Er zijn meer dan 500 miljoen mensen aangemeld op Google+, waarmee Google+ het op één na grootste sociale medium is.

In Google+ kunnen geregistreerde gebruikers zogenoemde *kringen* aanmaken. Binnen deze kringen kunnen *vrienden* toegevoegd worden om berichten per vriendengroep mee te delen. Vrienden kunnen een *hangout* beginnen om na installatie van een plug-in te kunnen 'videobellen' en samen video's op Google-dochter YouTube te bekijken.'

- Bron: http://nl.wikipedia.org/wiki/Google+

Figuur 9, login knop van Google(+) [14]

[14] Bron: https://developers.google.com/accounts/images/sign-in-with-google.png



Figuur 10, inlog scherm van Google(+)

3.2.1.1.2 Calendar

'Google Agenda (Engels: *Google Calendar*) is een online agenda. Het is een gratis service van Google en kan worden gekregen met een Google-account.

Per account kunnen meerdere agenda's gemaakt worden, die allemaal in één scherm worden getoond. Met Google Agenda kunnen online afspraken gemaakt worden met andere mensen die een Google-account hebben. Ook kan de agenda gedeeld worden met anderen of door anderen worden bijhouden.'

- Bron: http://nl.wikipedia.org/wiki/Google_Agenda

2.1.3.1 Welke talen en methodes worden er ondersteund?

De volgende talen worden door Google geaccepteerd:

- Javascript (wordt geprefereerd)
- .NET
- PHP
- Javascript
- Python
- Ruby

Daarnaast ondersteund Google de volgende methodes:

- REST
- JSON
- OAuth V2

2.1.3.2 Welke eisen worden er gesteld?

Om Google SmA's te kunnen implementeren moet er een (standaard) Google account aangemaakt worden. In dit account kan de applicatie geregistreerd worden binnen de developers console.

2.1.3.3 Wat zijn de mogelijkheden?

3.2.1.1.3 Plus

Er kunnen contacten van de gebruiker ingezien worden. Daarnaast wordt (op dit moment) Plus gezien als de noemer voor het inloggen bij Google.

3.2.1.1.4 Calendar

De datums van de gebruiker kan ingezien worden, en er kunnen datums toegevoegd, gewijzigd en verwijderd worden.

2.1.3.4 Wat zijn de (mogelijke) nadelen?

Ook Google geeft nergens (duidelijk) aangegeven wat de mogelijke nadelen zijn bij het gebruik van de SmA's.

Het grootste nadeel voor Datumprikker is waarschijnlijk het feit dat ieder product van Google een aparte API heeft. Hierdoor moet er iedere keer een nieuwe implementatie gebouwd worden.

2.1.3.5 Bronnen/verdere informatie

De volgende URL's bieden verdere informatie (en zijn daarnaast gebruikt als bron):

- https://developers.google.com/+/policies
- https://developers.google.com/google-apps/calendar/?hl=nl
- https://developers.google.com/+/api/
- Tutorials:
    - https://developers.google.com/+/web/api/javascript
    - http://www.asp.net/mvc/tutorials/mvc-5/create-an-aspnet-mvc-5-app-with-facebook-and-google-OAuth2-and-openid-sign-on

2.1.4 Microsoft

In tegenstelling tot Google, heeft Microsoft meerdere accounts voor de producten. Deze accounts vallen onder 2 typen:

- Office 365
- Live (vroeger Hotmail)

2.1.4.1 Office 365

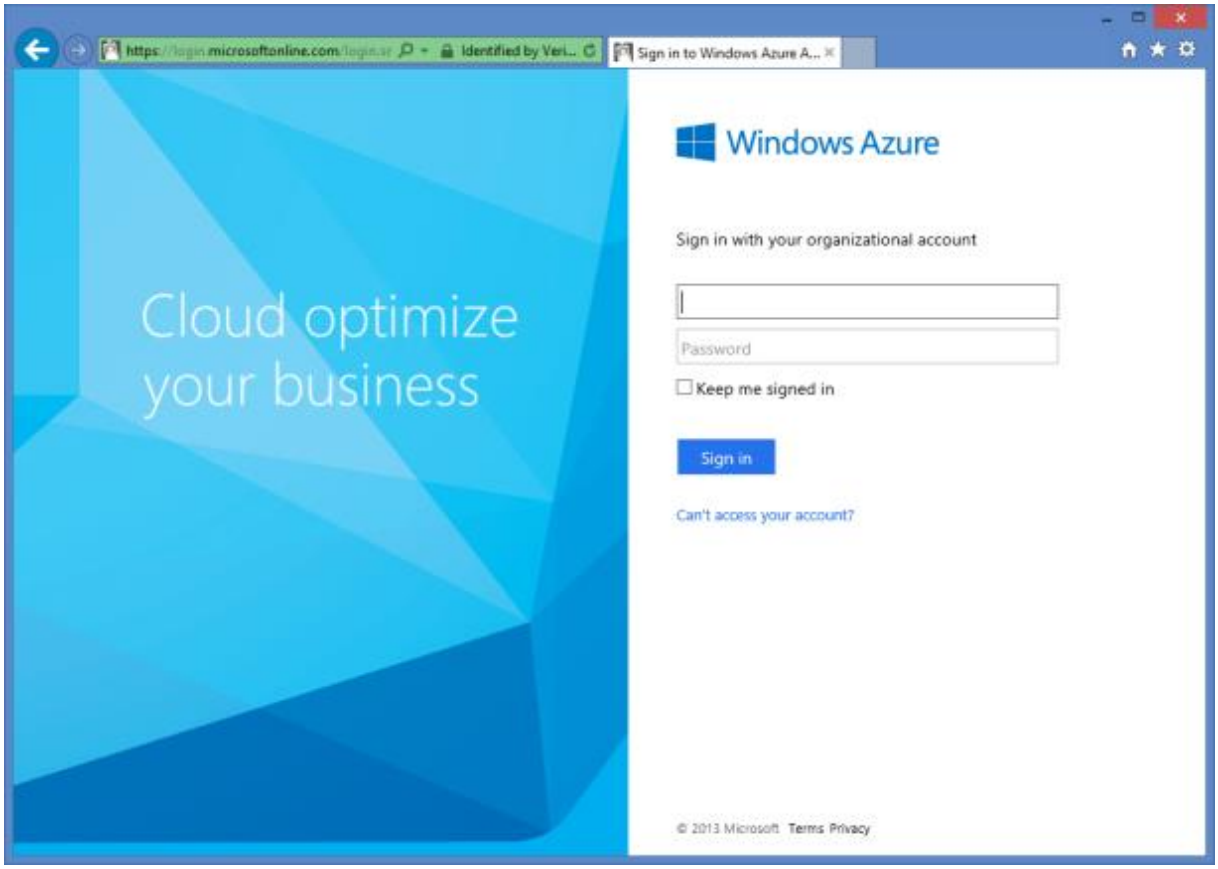
'Microsoft Office 365 is een verzameling van betaalde internetdiensten, bedoeld voor de zakelijke markt en het onderwijs. Het is vergelijkbaar met het kantoorpakket Microsoft Office en wordt eveneens door Microsoft ontwikkeld. De bètaversie van Office 365 werd gelanceerd in oktober 2010, gevolgd door de definitieve versie op 28 juni 2011.

Microsoft Office 365 is de opvolger van BPOS.'

- Bron: http://nl.wikipedia.org/wiki/Microsoft_Office_365

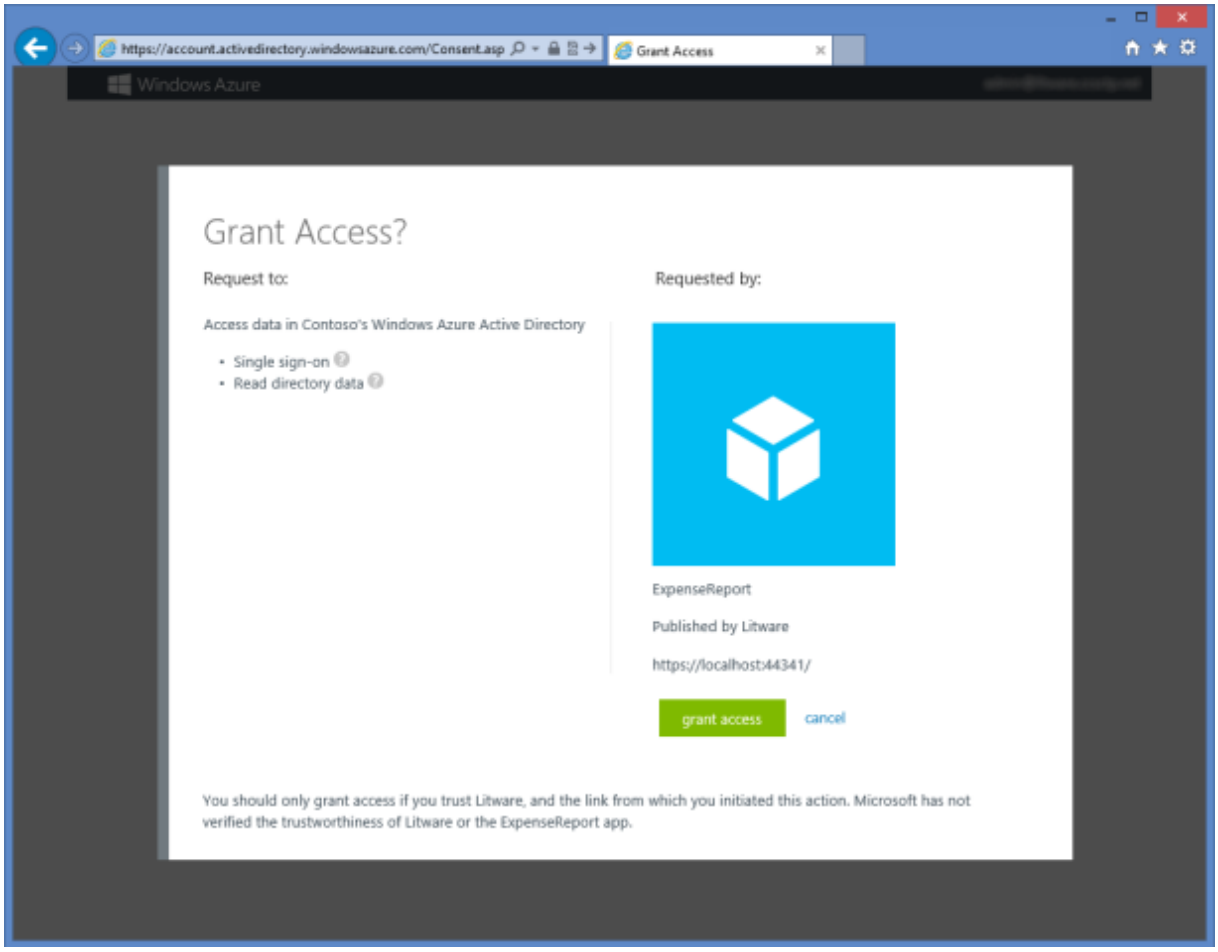
Indien er gebruik gemaakt wil worden van de Microsoft 365 omgeving doormiddel van een API, dan moet er gebruik gemaakt worden van de "Windows Azure AD Graph" API.

Een groot verschil van Azure AD ten opzichte van andere SmA's, is dat de gebruiker geen melding krijgt over wat de rechten zijn voor de applicatie. Dit wordt geregeld door de admin van de gebruiker.



Figuur 11, inlogscherm van de SmA

Voordat een gebruiker gebruik kan maken van de applicatie, moet de admin inloggen op de pagina van de applicatie (doormiddel van de SmA). Er verschijnt na inloggen een pop-up scherm met de rechten die de applicatie wil hebben. De toestemming voor het geven van deze rechten, gelden dan voor alle gebruikers binnen het bedrijf van de admin.



Figuur 12, toestemming pop-up voor de admin

Vanaf het moment dat de admin toestemming heeft gegeven, kunnen alle medewerkers (van dat betreffende bedrijf) inloggen en gebruik maken van de applicatie.

Indien de admin de rechten wil opheffen, dan moet hij/zij dit aangeven bij het account van het bedrijf. Dit gaat dus via Microsoft zelf, niet via de applicatie.

Belangrijk om te weten, is dat de SmA alleen werkt als de applicatie geregistreerd is bij Windows. Dit betekend echter nog niet dat Microsoft de applicatie geverifieerd heeft (zoals het geval is bij **Figuur 12**). De admin moet zich dus goed beseffen dat het accepteren ernstige consequenties kan hebben (zeker indien de applicatie lees en schrijf rechten krijgt).

2.1.4.1.1 Welke talen en methodes worden er ondersteund?

Microsoft is zeer onduidelijk in wat wel en niet ondersteund wordt. Door zeer gericht te zoeken, zijn de volgende ondersteunde talen/methoden echter wel naar voren gekomen:

- Talen:
  - .NET
  - PHP
- Methoden:
  - JSON
  - REST
  - OAuth V2

2.1.4.1.2 Welke eisen worden er gesteld?

Om de Microsoft Office 365 SmA te kunnen implementeren, moet er aan de volgende eisen voldaan worden:

- Een actieve Windows Azure subscriptie
- Visual Studio 2012 Professional/Ultimate voor het bouwen van de implementatie
- Identity and Access Tools (voor Visual Studio 2012) dient geïnstalleerd te zijn.
- De applicatie dient gekoppeld te worden aan het account van het bedrijf van de applicatie

2.1.4.1.3 Wat zijn de mogelijkheden?

De volgende concrete dingen zijn mogelijk met behulp van de SmA:

- Inloggen in Datumprikker met behulp van een Microsoft Office 365 account
- Inlezen van (mogelijk) ALLE data van de gebruiker, dus inclusief:
  - Naam
  - Adres
  - Contacten (binnen het bedrijf)
  - Applicaties (dus bijvoorbeeld outlook of iets dergelijks)
- Aanpassen (inclusief verwijderen) van deze data (bijvoorbeeld om een datum in de agenda te zetten)

2.1.4.1.4 Wat zijn de (mogelijke) nadelen?

Microsoft is ook hier zeer onduidelijk in. Wat wel sterk naar voren kwam, is dat een admin voor het hele bedrijf moet bepalen of er gebruik gemaakt kan worden van de applicatie. Aan de ene kant maakt dit het veiliger omdat er zo niet zo maar toegang verleend wordt. Maar aan de andere kant betekend dit een extra (eenmalige) stap voor gebruikers. Ook zijn de rechten die gegeven worden zeer uitgebreid, waardoor er belangrijke/gevoelige data ingezien/gewijzigd/verwijderd kan worden.

Een ander belangrijk punt om hier op te merken, is dat alleen Office365 en/of Azure accounts worden toegelaten voor deze SmA. Voor live accounts moet er gebruik gemaakt worden van een andere API: http://techjoomla.com/invitex/how-to-use-hotmailrest-api-to-import-contacts.html.

2.1.4.1.5 Bronnen/verdere informatie

De volgende URL's bieden verdere informatie (en zijn daarnaast gebruikt als bron):

- http://community.office365.com/en-us/blogs/office_365_community_blog/archive/2013/05/22/using-the-graph-explorer-to-query-the-graph-api.aspx
- http://msdn.microsoft.com/en-us/library/hh974476.aspx
- http://code.msdn.microsoft.com/Write-Sample-App-for-79e55502
- Tutorials:
    - http://msdn.microsoft.com/en-US/library/windowsazure/dn151790.aspx
    - http://msdn.microsoft.com/en-us/library/windowsazure/dn151789.aspx

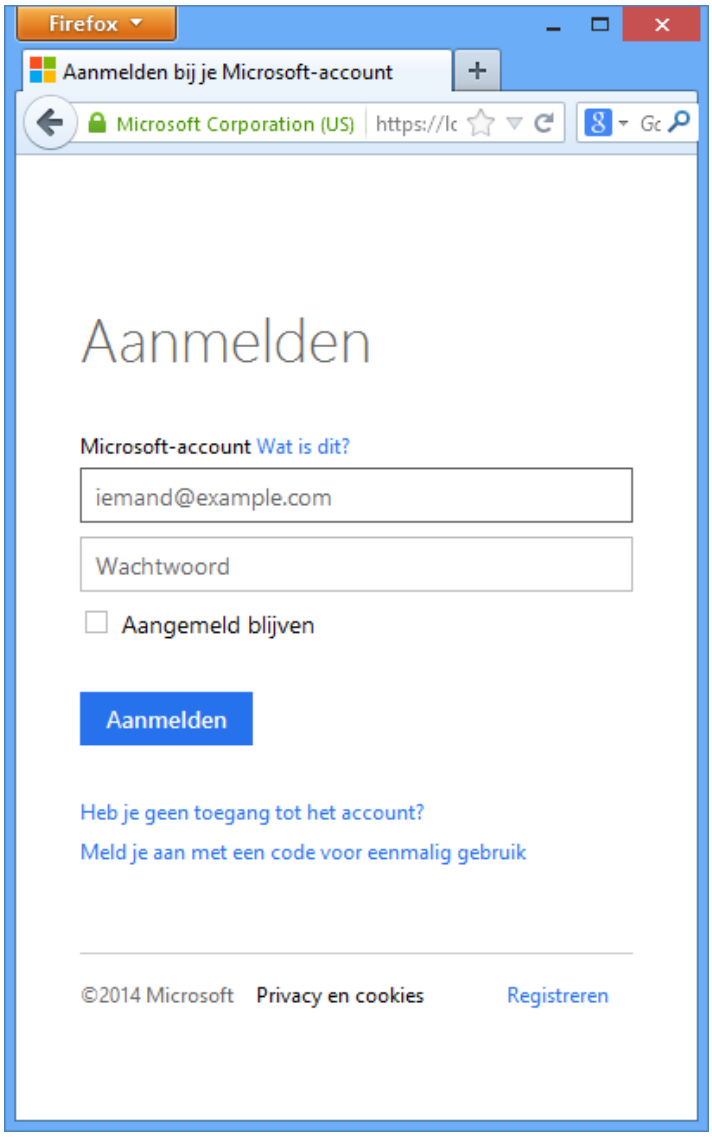
2.1.4.2 Live/Hotmail

'Windows Live is de naam voor een serie gratis internet- en desktopdiensten van softwarebedrijf Microsoft. De bedoeling van Windows Live is dat alles wat men nodig heeft voor het beheer van bestanden op z'n computer en op het internet op één plek samenkomt. Windows Live moet het centrale punt voor alle bekende diensten zijn en ervoor zorgen dat ze elkaar goed ondersteunen.'

- Bron: http://nl.wikipedia.org/wiki/Windows_Live

Figuur 13, voorbeeld van een login knop voor Live[15]

[15] Bron: http://www.belshe.com/wp-content/uploads/2010/05/auth.png



Figuur 14, login scherm van Live

2.1.4.2.1 Welke talen en methodes worden er ondersteund?

Microsoft is ook zeer gebrekkig qua informatie voor deze API. De volgende talen schijnen echter ondersteund te worden[16]:

- Javascript
- .NET

Daarnaast worden de volgende methodes ondersteund:

- JSON
- REST
- OAuth V2

[16] De talen die niet van belang zijn voor een web-applicatie zijn weggelaten.

2.1.4.2.2 Welke eisen worden er gesteld?

De applicatie moet geregistreerd worden bij Microsoft.

2.1.4.2.3 Wat zijn de mogelijkheden?

Met behulp van deze SmA kunnen de volgende applicaties/data benaderd worden:

- Identity (voor contact gegevens gebruiker)
- Outlook (voor email adressen)
- SkyDrive (voor allerlei bestanden van de gebruiker), dit is in principe (voor het korte termijn) niet interessant voor Datumprikker

2.1.4.2.4 Wat zijn de (mogelijke) nadelen?

Zoals gewoonlijk is er nagenoeg niets te vinden over mogelijke nadelen. Zolang deze SmA gebruikt wordt voor het ophalen van data en verifiëren van een gebruiker, zou er dan ook niet (veel) fout moeten kunnen gaan.

Mocht er echter gebruik gemaakt willen worden van SkyDrive, dan moet hier veel dieper naar onderzocht worden.

2.1.4.2.5 Bronnen/verdere informatie

De volgende URL's bieden verdere informatie (en zijn daarnaast gebruikt als bron):

- http://msdn.microsoft.com/en-US/library/live/hh826538
- http://msdn.microsoft.com/en-US/library/live/hh826538#csharp
- http://msdn.microsoft.com/en-us/library/live/hh243647.aspx
- http://techjoomla.com/invitex/how-to-use-hotmailrest-api-to-import-contacts.html

2.2 Andere methodes

2.2.1 OAuth

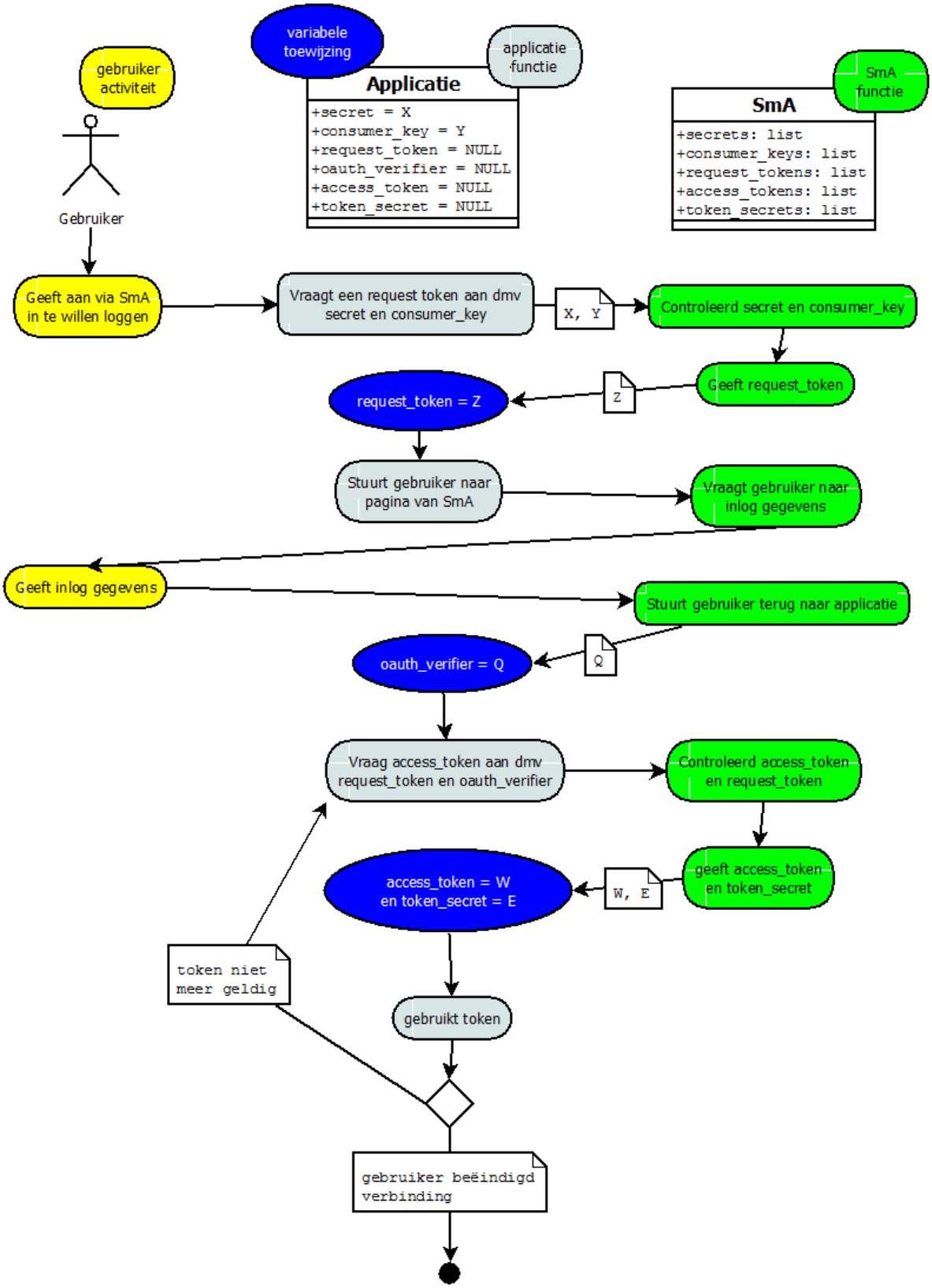
OAuth is een methode om op een veilige manier third party's (tijdelijk) gebruik te laten maken van een bestaande gebruiker van een bepaalde applicatie/omgeving. In dit geval is de third party Datumprikker, en de applicatie/omgeving de SmA.

In de afbeelding hieronder staat een globaal diagram van hoe OAuth werkt. Het is gebaseerd op een diagram van Yahoo[17]. Op de site staat echter niet vermeld om welke versie van OAuth het gaat. Gebaseerd op de punten van de paragraaf **Het verschil tussen V1a en V2** zou het moeten gaan om V1a.

Wat belangrijk is om op te merken bij het onderstaande diagram, is dat de variabelen **consumer_key** en **secret** reeds bekend zijn bij de **applicatie** voordat de communicatie begint. Deze 2 waardes zijn namelijk gegeven aan de **applicatie** door het verantwoordelijke social media bedrijf dat bij de **SmA** hoort. Het is het account en "wachtwoord" dat, in het geval van dit project, bij Datumprikker hoort.

Het voordeel van deze vorm van communicatie is dat het (praktisch) onmogelijk is voor niet-geautoriseerde personen/computers om de **consumer_key** te achterhalen. Dit komt doordat **secret** een hash is, die gebruikt wordt om de communicatie te van encryptie te voorzien.

[17] Bron: http://developer.yahoo.com/OAuth/guide/OAuth-auth-flow.html



Figuur 15, flow OAuth

2.2.1.1 Het verschil tussen V1a en V2

Er zijn een aantal verschillen tussen de 2 (standaard) versies van OAuth. De verschillen die hieronder staan, zijn gebaseerd op een blog geplaatst op Apigee[18].

V1(a)	V2
Geen SSL (verplicht).	SSL verplicht bij alle communicatie omtrent het genereren van een token.
Veel complexer vanwege complexe signatures bij genereren token.	Geen complexe signatures nodig bij genereren token.
Bij aanroepen van SmA moet de signature ook gebruikt worden.	SmA aanroepen (na de gegenereerde token) kan zonder signature (gebruik SSL aangeraden).
Na genereren token, moet er bij elke aanroep naar de SmA 2 beveiligingstokens meegestuurd worden, die de signature genereert.	Er wordt maar 1 beveiligingstoken verstuurd worden bij de aanroep, en er wordt geen signature gegenereerd.
Er is geen verschil te zien tussen de onderdelen van het protocol die geïmplementeerd worden door de SmA, en welke door een aparte autorisatie server.	Het is duidelijk welke onderdelen van het protocol geïmplementeerd worden door de SmA, en welke door een aparte autorisatie server.
Minder requirements richting de aanroeper.	Meer requirements richting de aanroeper.
Goed gedefinieerde standaard die niet snel meer zal veranderen.	Nog geen stabiele IETF draft, dus er kunnen nog veel dingen veranderen.
Veel libraries beschikbaar voor de implementatie.	Nog niet veel libraries beschikbaar, dus er moet veel zelf gebouwd worden.
Vanwege complexiteit lastiger te implementeren.	Makkelijker te implementeren op zowel de client als server kant.
Sneller vanwege het niet gebruiken van SSL.	Langzamer vanwege het SSL gebruik.

[18] Bron: https://blog.apigee.com/detail/OAuth_differences

2.2.2 JSON[19]

JSON Staat voor Javascript Object Notation, en is een variant op XML. Oorspronkelijk komt het van de Javascript taal af, maar tegenwoordig is het (programmeer)taalonafhankelijk data formaat. Wel lijkt de opbouw nog steeds op de oorspronkelijke taal. Ook wordt hetzelfde principe voor variabelen aangehouden, namelijk dat variabelen een dynamische type[20] hebben.

JSON Wordt gebruikt om variabelenwaarden uit te wisselen. Hiervoor worden er een aantal regels aangehouden:

- Het bestand begint met een '{', en eindigt met een '}'.
- Variabele-namen worden met dubbele quotes ingesloten (voorbeeld: "variabele").
- Een waarde toewijzing naar een variabele gebeurd doormiddel van een dubbele punt (en een spatie), met daarachter 1 van de volgende type toewijzingen:
    - Getal (double-precision floating-point format[21], oftewel een getal met mogelijkheid tot decimalen);
    - Tekst, ingesloten met dubbele quotes;
    - Boolean ('true' of 'false');
    - Array van variabelen, beginnend met een '[', en eindigend met een ']';
    - Object gevuld met variabelen, beginnend met een '{', en eindigend met een '}' (het bestand zelf wordt dus ook als een object gezien);
    - Leeg, aangegeven met 'null';
- Indien er meerdere variabelen voorkomen binnen een bestand/array/object, dan worden deze gescheiden van elkaar met behulp van een komma (en een enter).

Als een SmA JSON gebruikt/ondersteund, dan wordt er bedoeld dat des SmA een JSON bestand teruggestuurd naar de aanvrager (voor dit project Datumprikker).

[19] Bron: http://en.wikipedia.org/wiki/JSON
[20] Voorbeeld: variabele X is eerst een integer(getal), en wordt later een string(tekst)
[21] http://en.wikipedia.org/wiki/Double-precision_floating-point_format

2.2.2.1 *Voorbeeld*

Hieronder staat een voorbeeld van hoe een JSON bestand eruit ziet:

```
{
    "voornaam": "Stijn",
    "achternaam": "van der Meer",
    "leeftijd": 23,
    "adres": {
        "straat": "Afstudeerlaan",
        "nr": 8,
        "stad": "Den Haag"
    },
    "telefoon": [
        {
            "type": "thuis",
            "nummer": null
        },
        {
            "type": "mobiel",
            "nummer": 1234567890
        }
    ]
}
```

Figuur 16, JSON voorbeeld

Het betreft bij **Figuur 16** dus om 5 (hoofd)variabelen, waarbij **adres** een object is, en **telefoon** een array van objecten.

2.2.3 REST[22]

REST Werkt doormiddel van de volgende 4 HTTP begrippen:

- GET; hiermee kan data opgevraagd worden.
- PUT; hiermee kan (bestaande) data geüpdatet worden.
- DELETE; hiermee kan data verwijderd worden.
- POST; hiermee kan nieuwe data geplaatst worden

Deze begrippen worden dan gevolgd met een URI, waardoor deze data geplaatst kan worden. Voorbeelden hiervan zijn:

- *POST* *http://www.example.com/customers*; het aanmaken van een nieuwe klant
- *GET* *http://www.example.com/customers/33245*; het ophalen van klant nr.33245

De data die gebruikt wordt, staat opgeslagen in een form.

[22] Bron: http://www.restapitutorial.com/

2.3 Conclusie

product/bedrijf	talen/methodes			benodigdheden	mogelijkheden
	invoer	uitvoer	overig		
Datumprikker	.NET	JSON	REST		
	HTML5		Oauth 1a		
	Javascript		Oauth 2		
	PHP				
	Python				
	Ruby				
	Wordpress				
Facebook	.NET	JSON	REST	developer account	Inloggen
	HTML5		Oauth 1a		Contacten ophalen
	Javascript		Oauth 2		
	PHP				
	Python				
	Ruby				
	Wordpress				
LinkedIn	.NET	JSON	REST	registratie app	Inloggen
	HTML5		Oauth 1a		Contacten ophalen
	Javascript		Oauth 2		
	PHP				
	Python				
	Ruby				
	Wordpress				
Google	.NET	JSON	REST	Google account	Inloggen
	HTML5		Oauth 1a		Contacten ophalen
	Javascript		Oauth 2		Agenda CRUD
	PHP				
	Python				
	Ruby				
	Wordpress				
Microsoft Office365	.NET	JSON	REST	Actieve Windows Azure subscriptie	Inloggen
	HTML5		Oauth 1a	Visual Studio 2012 Proffesional/Ultimate	Contacten ophalen
	Javascript		Oauth 2	Identity and Access Tools voor Visual Studio 2012	Agenda CRUD
	PHP			Registratie app door admin	
	Python				
	Ruby				
	Wordpress				
Microsoft Live	.NET	JSON	REST	registratie app	Inloggen
	HTML5		Oauth 1a		Contacten ophalen
	Javascript		Oauth 2		
	PHP				
	Python				
	Ruby				
	Wordpress				

Mogelijk
Niet mogelijk
Onduidelijk of dit mogelijk is
SmA specifiek

Figuur 17, SmA's met elkaar vergeleken

Zoals te zien in **Figuur 17**, zijn er veel overeenkomsten tussen de verschillende SmA's. De enige echte verschillen zitten in de invoer taal. Daar wordt in ieder geval bij alle SmA's .NET ondersteund.

Een andere conclusie die opvalt, is dat als er gebruik gemaakt wil worden van Microsoft Office365, er zo goed als zeker met .NET gewerkt moet worden.

	Facebook	LinkedIn	Google			Microsoft	
			Plus	Contacts	Calendar	Office 365	Windows Live
Social login	J	J	J	N	N	J	J
Deelnemers uitnodigen							
Contacten opvragen	J	J	J	J	N	J	J
Contacten mailen	J[23]	J[24]	J	J	N	J	J
Contacten "intern" bericht sturen	J	J	J	N	N	N	N
Beschikbaarheid opvragen	N	N	N	N	J	J	J[25]
Afspraak datum(s) opslaan/publiceren	J[26]	N	N	N	J	J	J[25]

2.4 Aanbevelingen

De volgende aanbevelingen voor dit project kunnen gemaakt worden op basis van de onderzochte SmA's:

- .NET Dient gebruikt te worden om de SmA's aan te spreken.
- De module dient JSON te kunnen ontvangen en gebruiken.
- Er moet met REST, OAuth V1a en OAuth V2 gewerkt worden.
- De 3 (hoofd)typen functionaliteit moeten mogelijk later uitgebreid kunnen worden
- Per SmA moet er rekening gehouden worden met extra eisen, sommige hebben invloed op de omgeving/code

[23] Hetzelfde als bij LinkedIn
[24] Via LinkedIn mail systeem (dus mail wordt vanuit LinkedIn gestuurd). Direct mailen kan (volgens mij) niet, omdat daar specifieke permissies aan zitten.
[25] Indien het live account gekoppeld is aan outlook. Dit is niet altijd het geval (zoals bij mijn account)
[26] Doormiddel van events (https://developers.facebook.com/docs/graph-api/reference/event/)

3 Mobile (app) gebruik

Omdat de communicatie tussen een app en de SmA's meestal anders verloopt dan via een (standaard) browser, is er per (uit het vorige onderzoek) onderzochte SmA gekeken wat de extra eisen, mogelijkheden en/of beperkingen zijn indien deze ook voor een app beschikbaar moet zijn.

De 2 omgevingen waar rekening mee gehouden moet worden zijn:

- IOS
- Android

3.1 Social media

Er zijn 5 verschillende SmA's onderzocht:

- **Facebook**
- **LinkedIn**
- **Google**
- **Microsoft Office365**
- **Microsoft Live**

3.1.1 Facebook

Hoewel niet goed gedocumenteerd, lijkt het er op dat de app exact hetzelfde werkt qua proces als bij de web-omgeving. De app zal dus binnen zijn eigen omgeving naar een URL moeten gaan om te kunnen autoriseren.

Wel dient er gebruik gemaakt te worden van een zogenaamde "Appsecret_proof" [27]. Dit is een extra variabele die er moet voor zorgen dat er niet zo maar gebruik gemaakt kan worden van de tokens.

Bron/verdere informatie:
https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow

3.1.1.1 SDK

Facebook heeft ook SDK's beschikbaar voor IPhone en Android. Deze bieden dezelfde functionaliteit als de plug-in module, maar dan alleen voor Facebook.

- IOS: https://developers.facebook.com/docs/ios
- Android: https://developers.facebook.com/docs/android

[27] https://developers.facebook.com/docs/graph-api/securing-requests

3.1.2 LinkedIn

Indien er gebruik gemaakt wordt van OAuth2, dan lijkt het er op dat de app aan hetzelfde proces en beperkingen zit als bij de web-omgeving. De app zal dus in zijn eigen omgeving naar een URL moeten gaan om te kunnen autoriseren. Omdat de token 'maar' (maximaal) 60 dagen geldig is, zal er dus ofwel automatisch de token vernieuwd moeten worden, ofwel zal de gebruiker elke 60 dagen opnieuw moeten inloggen.

Indien er gebruik gemaakt wordt van OAuth1, dan kan het hierboven beschreven proces omzeild worden. OAuth1 wordt echter afgeraden, en de documentatie over dit proces is niet duidelijk.

Bron/verdere informatie:
https://developer.linkedin.com/documents/authentication
https://developer.linkedin.com/documents/OAuth-10a

3.1.2.1 Libraries

LinkedIn heeft geen SDK beschikbaar (voor app's), maar wel verschillende libraries[28] voor uiteenlopende omgevingen. Deze libraries zouden gebruikt kunnen worden in plaats van de plug-in module. Bij de meest gewaardeerde Android library[29] lijkt het er echter op dat deze ook niet het refresh probleem op lost.

3.1.3 Google

Aangezien Google de OS Android heeft, is het uiteraard mogelijk om de Google SmA's te koppelen met een Android apparaat. In die documentatie[30] hierover gaat het echter over de eigen SDK.

Indien er niet gebruik gemaakt zal worden van de SDK, dan gelden nagenoeg de zelfde dingen als bij de web-omgeving. Er zal dus binnen de eigen omgeving naar een URL gegaan moeten worden om te kunnen autoriseren.

Bron/verdere informatie:
http://developer.android.com/google/play-services/auth.html

3.1.3.1 SDK

Google heeft voor zowel Android als IOS SDK's beschikbaar. Deze bieden dezelfde functionaliteit als de plug-in module, maar dan alleen voor Google.

- Android: https://developers.google.com/+/mobile/android/getting-started
- IOS: https://developers.google.com/+/mobile/ios/getting-started

3.1.4 Microsoft Office365

Er is nog minder informatie te vinden over de app variant als voor de web-omgeving zelf. Er kan hier dus weinig over gemeld worden.

Wel zou het logisch zijn dat het proces hetzelfde zal gaan als bij de web-omgeving, aangezien er net zoals bij alle vorige SmA's er gebruik gemaakt wordt van dezelfde methodes.

[28] https://developer.linkedin.com/documents/libraries-and-tools
[29] Bron: http://code.google.com/p/linkedin-j/
[30] https://developers.google.com/+/mobile/android/sign-in

3.1.5 Microsoft Live

Volgens de site[31] gaat het proces nagenoeg hetzelfde als bij de web-omgeving. Het enige verschil is dat er bij de autorisatie de volgende stuk tekst meegegeven moet worden aan de URL: “&display=touch”.

Bron/verdere informatie:
http://msdn.microsoft.com/en-us/hh826540#consent_java

3.1.5.1 Libraries

Microsoft heeft libraries beschikbaar voor de mobiele omgevingen. Deze bieden dezelfde functionaliteit als bij de plug-in module, maar dan alleen voor Live.

- IOS: http://msdn.microsoft.com/en-us/library/hh847068.aspx
- Android: http://msdn.microsoft.com/en-us/library/hh846809.aspx

3.2 Conclusie

Na alle SmA’s onderzocht te hebben, kan er met vrij grote zekerheid gesteld worden dat voor het gebruik van deze SmA’s, hetzelfde proces aangehouden kan worden. Alleen bij Office365 is dit niet duidelijk. Ook is het niet duidelijk hoe het inloggen van een gebruiker precies werkt. Dat wil zeggen: bij de web-omgeving komt er een pop-up window die informatie terugspeelt naar het (hoofd)scherm. Bij de app kan dit niet met een window, maar zal dit via een (interne) browser moeten gebeuren.

Daarnaast moet er bij sommige SmA’s een aantal aanpassingen gemaakt worden qua instellingen/variabelen.

Concrete punten waar rekening mee gehouden moet worden:

- **Facebook**:
  - “Appsecret_proof” variabele gebruiken.
- **LinkedIn**:
  - Token 60 dagen geldig.
  - Mogelijk gebruik maken van OAuth1.
- **Microsoft Office365**:
  - Onduidelijk hoe dit exact werkt.
- **Microsoft Live**:
  - “&display=touch” toevoegen aan de URL.
- Algemeen:
  - Er zal met OAuth2 (of OAuth1) en REST gewerkt moeten worden.

Tevens zijn er voor alle (op Office365 na) SmA’s een SDK/library beschikbaar voor zowel Android als IOS. Deze omzeilen dan wel het concept van de plug-in module.

[31] http://msdn.microsoft.com/en-us/hh826540

3.3 Aanbevelingen

- Voordat er gebruik gemaakt gaat worden van Office365 voor de app, moet er eerst nagegaan worden hoe deze SmA precies werkt voor de web-omgeving.
- Er moet dieper onderzoek gedaan worden naar hoe Android en IOS omgaan met redirects binnen een app. Belangrijk om te achterhalen is of data die binnen de omgeving van deze redirect (interne/externe brower) teruggespeeld kan worden naar de app zelf.

4 Aanpassingen Datumprikker

Om volledig gebruik te kunnen maken van de plug-in module, zijn er een aantal dingen waar rekening mee gehouden moet worden. Bij dit onderzoek worden alle benodigdheden en/of restricties opgenoemd.

Hoewel deze dingen vooral voor Datumprikker bedoelt zijn, zal er bij een ander product wat gebruik gaat maken van deze module, ook rekening gehouden moeten worden met deze punten.

De uitkomsten van het onderzoek zijn verdeeld onder 3 onderdelen:

- **Database**
- **(G)UI**
- **Social media accounts**
- **Gebruik van module**

4.1 Database

Indien er een koppeling gemaakt dient te worden tussen data van de SmA's en data van Datumprikker, dan moet de database de volgende dingen kunnen opslaan/linken:

- **Token**; een string ((hoofd)letters, cijfers, en (een aantal) speciale tekens worden gebruikt) van een variabele lengte (houdt rekening met (minstens) 256 tekens), mogelijk een hash.
- Id van gebruiker bij SmA; een string ((hoofd)letters, cijfers, en (een aantal) speciale tekens worden gebruikt) van een variabele lengte (houdt rekening met (minstens) 32 tekens).
- SmA-naam; aangezien er meerdere SmA's gebruikt (kunnen) worden, dient er aangegeven te worden bij welke SmA de data hoort. Hoe dit aangegeven worden staat volledig vrij, zolang de bij de plug-in module bekende naam maar teruggekoppeld kan worden. Indien de namen van de plug-in module direct overgenomen worden, dan betreft het (op dit moment) een string van maximaal 8 tekens ((hoofd)letters) lang.

Deze 3 variabelen hoeven dus alleen opgeslagen te worden indien er een koppeling gemaakt dient te worden tussen de Datumprikker gebruiker en een SmA. Dit is bijvoorbeeld het geval bij het inloggen van een gebruiker met behulp van een SmA.

Als er alleen (tijdelijk) gebruik gemaakt moet worden van de SmA's, zoals bijvoorbeeld bij het opvragen van contacten of kalenderdata, dan is er geen koppeling nodig.

belangrijk

Andere data die van de SmA's binnenkomt, kan uiteraard ook opgeslagen worden. Dit is echter redelijk omslachtig voor de meeste gevallen aangezien de data bij de SmA ook al opgeslagen is (en nog belangrijker: up-to-date gehouden wordt).

4.1.1 Contacten

Een van de uitzonderingen hierop is het opslaan van (een selectie van) contacten. Deze contact-data kan dan gebruikt worden om berichten te sturen met behulp van de berichten functionaliteit. De belangrijke data die hierbij opgeslagen kan worden, zijn:

- naam; een string ((hoofd)letters en spaties) van een variabele lengte. Houdt hiervoor de standaard (zelf afgesproken) lengte voor een naam aan.
- Verzend-id; een string ((hoofd)letters, cijfers, en (een aantal) speciale tekens worden gebruikt) van een variabele lengte. Dit is in de meeste gevallen gewoon een emailadres, maar bij sommige SmA's is dit een code. Voor de code is een lengte van 16 voldoende. Dus als de lengte van een email wordt aangehouden (die zeer waarschijnlijk langer is), zal hier geen probleem ontstaan.

4.1.2 Kalender-data

Een andere uitzondering op het niet opslaan, is het opslaan van kalenderafspraken die door een gebruiker via Datumprikker heeft aangemaakt. Aangezien de afspraak wordt geretourneerd indien de afspraak ingepland wordt, kan de afspraak-id opgeslagen worden. De totale opsomming van data rondom kalenders kan als volgt zijn:

- Afspraak-id; een string ((hoofd)letters, cijfers, en (een aantal) speciale tekens worden gebruikt) van een variabele lengte. Houdt rekening met ten minste 32 tekens.
- Begintijd; een datum/tijd. Het liefst in ISO 8601 formaat, aangezien die door de module aangehouden wordt.
- Eindtijd; een datum/tijd. Het liefst in ISO 8601 formaat, aangezien die door de module aangehouden wordt.

Later kan het id dan gebruikt worden (mits er uiteraard nog toegang is, en de afspraak nog bestaat), om de afspraak aan te passen of verwijderen.

Indien een gebruiker de afspraak van naam veranderd, of de datum/tijd aanpast, dan heeft dit geen invloed op het ophalen van de afspraak. Hier moet echter wel rekening mee gehouden worden, aangezien de afspraak dan ook aan de kant van Datumprikker (mogelijk) aangepast moet worden.

Dit houdt in dat er periodiek (of anders in ieder geval iedere keer wanneer de afspraak binnen Datumprikker bekeken wordt) de afspraken afgegaan moet worden om te controleren of de afspraken nog correct (en niet verwijderd) zijn.

Dit zorgt voor een hoop dataverkeer wat zeer nadelig kan zijn voor de server/database.

4.1.3 Token

De token is een speciale variabele (in de zin van opslag), omdat deze regelmatig vernieuwd zal moeten worden. Indien een gebruiker niet direct voor alle rechten toestemming hoeft te geven, zal er bijgehouden moeten worden voor welke rechten er toestemming is. Dit maakt het controle proces een stuk ingewikkelder, maar wel eerlijker/overzichtelijker voor de gebruiker.

4.1.3.1 Overschrijven

Aangezien Google slechts de enige SmA is die rechten cumulatief aan een token kan koppelen, zal de token iedere keer bij een rechten-update de token overschreven moeten worden. Om dit te kunnen doen, moet er (in de database) bijgehouden worden welke rechten er al gegeven zijn, en hier dan de nieuwe rechten aan toe voegen. Deze groep rechten moeten dan meegegeven worden aan de SmA bij het inlog proces van de gebruiker. Bij een wijziging van rechten moet er namelijk (automatisch) opnieuw ingelogd worden door de gebruiker, zodat deze de rechten kan verlenen.

4.1.3.2 Rechten preventief koppelen

Om dit lastige proces te omzeilen, kan er ook gekozen worden om van te voren een aantal groepen rechten te koppelen. Hierdoor hoeft er minder bijgehouden te worden. De huidige plug-in module ondersteund een aantal vooropgestelde rechten-groepen:

- LoginContacts; bevat alle rechten voor inloggen en contacten.
- LoginContactsMessage; bevat alle rechten voor inloggen, contacten en berichten.
- LoginCalendar; bevat alle rechten voor inloggen en kalender.
- ALL; bevat alle rechten voor alle functionaliteiten.

Het aller makkelijkste voor het opslaan van tokens, is om ‘ALL’ aan te houden. Dit voorkomt namelijk volledig het hiervoor gestelde probleem. Dit betekend echter wel dat de gebruiker in een keer voor alles toestemming moet geven.

4.1.3.3 Refresh-token

Indien er een nieuwe token opgevraagd wordt door de oude token te vernieuwen, dan dient de oude token altijd overschreven te worden. De oude tokens zijn soms nog wel (tijdelijk) geldig, maar hier kan niet vanuit gegaan worden.

4.2 (G)UI

4.2.1 SmA knoppen

Om gebruik te kunnen maken van de plug-in module moeten er knoppen geplaatst worden op plekken binnen Datumprikker waar de gebruiker de functionaliteit van de plug-in module dient te kunnen gebruiken.

Deze knoppen kunnen semiautomatisch geplaatst worden met behulp van de **GetSmaPlugins** en **ImplementsType** methodes. Uiteraard kunnen de knoppen ook handmatig geplaatst worden.

De knoppen dienen een SmA- en typenaam te bevatten, en daarnaast verwijzen naar een (Javascript) methode die op basis van die gegevens de plug-in module kan aanroepen.

4.2.2 Opvang-pagina

De opvang-pagina is een html document waar de SmA naar terug doorverwijst nadat de gebruiker is geauthentiseerd, en de gebruiker Datumprikker heeft geautoriseerd. Bij deze doorverwijzing stuurt de SmA een code mee terug. Deze code dient later gebruikt te worden om een token te ontvangen.

De opvang-pagina is zeer belangrijk voor het gebruik van de plug-in module, omdat het de logica moet bevatten die bepaald naar welke pagina er doorverwezen moet worden om bij de originele pagina te komen waar de gebruiker was voordat deze de SmA ging gebruiken.

4.2.2.1 Pop-up window

Om dit probleem op te lossen, kan er het beste gebruik gemaakt worden van een pop-up window. Hierdoor blijft de (hoofd)pagina gewoon waar de gebruiker was, en hoeft de pop-up pagina alleen maar naar zijn parent terug te verwijzen. De (hoofd)pagina kan daarna dan data opvragen aan/versturen naar de SmA.

4.3 Social media accounts

De plug-in module heeft (op dit moment) functionaliteit richting 4 SmA's:

- **Facebook**
- **LinkedIn**
- **Google**
- **Live/Hotmail**

Om hiervan gebruik te kunnen maken dient er voor elk van de SmA een account('app') gemaakt te worden. Hierbij horen 2 variabelen:

- companyKey; dit is het account(nummer) van de app.
- companySecret; dit is het geheim/wachtwoord/versleuteling van de app.

4.4 Gebruik van module

Om de plug-in module te kunnen gebruiken hoeft alleen de (laatste versie van de) plug-in module DLL gekoppeld te worden aan het project. Let er hierbij wel op dat de MVC versie (= 4) ondersteund wordt door het product.

Zodra de DLL gekoppeld is, kan de plug-in module met 5 (hoofd)methodes[32] aangeroepen worden, met bijbehorende in- en output[33]:

- **GetSmaPlugins**
- **ImplementsType**
- **Redirect**
- **GetSmaData**
- **PostSmaData**

Daarnaast zijn er een aantal façade methodes:

- **SmaLogin**
- **GetSmaContacts**
- **GetSmaCalendar**
- **PostSmaCalendar**
- **CheckSmaMessage**
- **PostSmaMessage**

[32] Het **grijze** gedeelte kan weggelaten worden indien de referentie is toegevoegd.
[33] De variabelen worden uitgelegd in H**4.4.3**.

4.4.1 Methodes

4.4.1.1 GetSmaPlugins

Webbeat.Plugin.Social.Models.ModuleController.GetSmaPlugins()

Deze methode geeft alle beschikbare SmA's.

4.4.1.1.1 Input

Geen input nodig.

4.4.1.1.2 Output

- **Dictionary<SocialType, Plugin>** pluginList; een lijst van alle plug-ins die gebruikt kunnen worden.

4.4.1.2 ImplementsType

Webbeat.Plugin.Social.Models.Plugin.ImplementsType()

Deze methode controleert of een plug-in een type ondersteund.

4.4.1.2.1 Input

- **SocialActionType** typeName; de type aanvraag (enum), bijv. **SocialActionType**.Login.

4.4.1.2.2 Output

- **bool** implements; geeft 'true' indien de SmA de typeName ondersteund. Anders wordt er 'false' geretourneerd.

4.4.1.3 Redirect

Webbeat.Plugin.Social.Models.ModuleController.Redirect()

Deze methode moet als eerste aangeroepen worden. Hij wordt gebruikt om de gebruiker te laten inloggen en authentiseren (van het product) bij de SmA. Het resultaat van de methode dient gebruikt te worden om de gebruiker terug te sturen naar een **Opvang-pagina**.

4.4.1.3.1 Input

- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Google.
- **SocialActionType** typeName; de type aanvraag (enum), bijv. **SocialActionType**.Calendar.
- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** returnUrl; de URL waar de plug-in module naar terug moet gaan.
- **bool** offline; indien de token opgeslagen zal worden (voor later gebruik), dient hier 'true' gebruikt te worden. Anders kan deze variabelen leeg gelaten worden (standaard 'false').

4.4.1.3.2 Output

- **string** URL; de URL (inclusief de correcte query's). Er dient een doorverwijzing gedaan te worden naar deze URL.

4.4.1.4 GetSmaData

Webbeat.Plugin.Social.Models.ModuleController.GetSmaData()

Deze methode wordt gebruikt om data op te vragen aan een SmA. Deze data wordt omgezet en teruggestuurd naar de aanroeper.

4.4.1.4.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **DatabaseSender** db; de variabelen van de database die nodig zijn. Bevat onder andere de token (indien beschikbaar).
- **string** code; de code die na de doorverwijzing na **Redirect** teruggestuurd wordt. Indien er een token is, moet dit 'null' zijn.
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Facebook.
- **SocialActionType** type; de type aanvraag (enum), bijv. **SocialActionType**.Contacts.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.success.
- out **DatabaseSender** dbResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').
- **Filter**[] filters; indien er gefilterd dient te worden, dan dienen deze hier gezet te worden. Anders kan deze variabele leeggelaten worden.

4.4.1.4.2 Output

- **ISmaResult** result; de data die geretourneerd wordt door de SmA.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_empty.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.1.5 PostSmaData

Webbeat.Plugin.Social.Models.ModuleController.PostSmaData()

Deze methode wordt gebruikt om data te sturen naar een SmA. De SmA stuurt hierop een resultaat terug, dat omgezet en doorgegeven zal worden naar de aanroeper.

4.4.1.5.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **DatabaseSender** db; de variabelen van de database die nodig zijn. Bevat onder andere de token (indien beschikbaar).
- **string** code; de code die na de doorverwijzing na **Redirect** teruggestuurd wordt. Indien er een token is, moet dit 'null' zijn.
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.LinkedIn.
- **SocialActionType** type; de type aanvraag (enum), bijv. **SocialActionType**.Login.
- **SmaPost** POST; de data die verstuurd moet worden naar de SmA.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_connection.
- out **DatabaseSender** dbResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.1.5.2 Output

- **ISmaResult** result; de data die geretourneerd wordt door de SmA.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_filter.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2 Façade methodes

Aangezien de GetSmaData en PostSmaData methodes nogal veel verwachten, zijn er 8 façade methodes[32] die specifieke functionaliteit bieden, met versimpelede input:

4.4.2.1 SmaLogin

Webbeat.Plugin.Social.Models.ModuleController.SmaLogin()

Met deze methode wordt een gebruiker ingelogd. Het resultaat is de accountgegevens die door de SmA teruggestuurd worden.

4.4.2.1.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** code; de verplichte code die na de doorverwijzing na **Redirect** teruggestuurd wordt.
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.LinkedIn.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.success.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.1.2 Output

- **ILoginResult**; de inlog data die de SmA retourneert.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_authorisation.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2.2 GetSmaContacts

Webbeat.Plugin.Social.Models.ModuleController.GetSmaContacts()

Met deze methode worden de contacten van een gebruiker opgehaald. Deze methode heeft 2 varianten qua input:

- **Input-A**; hierbij is er een code in plaats van een token nodig.
- **Input-B**; hierbij is er een token en id nodig, en geen code.

4.4.2.2.1 Input-A

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** code; de verplichte code die na de doorverwijzing na **Redirect** teruggestuurd wordt.
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.LinkedIn.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_wrongAccount.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.2.2 Input-B

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** id; de id van de gebruiker (van de SmA).
- **string** token; de token van de gebruiker (van de SmA).
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Facebook.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_empty.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.2.3 Output

- **IContactsJsonResult**; de contact data die de SmA retourneert.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.success.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2.3 GetSmaCalendar

Webbeat.Plugin.Social.Models.ModuleController.GetSmaCalendar()

Met deze methode worden de kalender-data van een gebruiker opgehaald. Deze methode heeft 2 varianten qua input:

- **Input-A**; hierbij is er een code in plaats van een token nodig.
- **Input-B**; hierbij is er een token en id nodig, en geen code.

4.4.2.3.1 Input-A

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** code; de verplichte code die na de doorverwijzing na **Redirect** teruggestuurd wordt.
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Google.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_connection.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.3.2 Input-B

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** id; de id van de gebruiker (van de SmA).
- **string** token; de token van de gebruiker (van de SmA).
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Live.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_wrongAccount.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.3.3 Output

- **ICalendarListResult**; de kalender data die de SmA retourneert.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_empty.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2.4 PostSmaCalendar

Webbeat.Plugin.Social.Models.ModuleController.PostSmaCalendar()

Met deze methode wordt er een datum naar een kalender van een gebruiker gestuurd. De afspraak data wordt teruggestuurd indien de methode succesvol was.

4.4.2.4.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** id; de id van de gebruiker (van de SmA).
- **string** token; de token van de gebruiker (van de SmA).
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Facebook.
- **SmaPost** POST; de afspraak-data die verstuurd gaat worden.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.success.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.4.2 Output

- **ICalendarListResult**; de afspraak data die de SmA retourneert.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_empty.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2.5 CheckSmaMessage

Webbeat.Plugin.Social.Models.ModuleController.CheckSmaMessage()

Met deze methode wordt er gecontroleerd of de token en id voor het verzenden van een bericht correct en geldig zijn. Dit is het geval indien 'out functionResult' de waarde **resultEnum**.success is.

4.4.2.5.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** id; de id van de gebruiker (van de SmA).
- **string** token; de token van de gebruiker (van de SmA).
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Facebook.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_authorisation.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.5.2 Output

- **IMessageResult**; er wordt altijd 'null' geretourneerd.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_connection.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.2.6 PostSmaMessage

Webbeat.Plugin.Social.Models.ModuleController.PostSmaMessage()

4.4.2.6.1 Input

- **string** companyKey; de bedrijfssleutel (**Social media accounts**).
- **string** companySecret; het bedrijfsgeheim (**Social media accounts**).
- **string** id; de id van de gebruiker (van de SmA).
- **string** token; de token van de gebruiker (van de SmA).
- **SocialType** sma; de naam van de SmA (enum), bijv. **SocialType**.Google.
- **SmaPost** POST; de bericht-data die verstuurd gaat worden.
- out **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_filter.
- out **DatabaseSender** connectionResult; de variabelen voor de database (dit mag hetzelfde zijn als 'db').

4.4.2.6.2 Output

- **IMessageResult**; indien succesvol, retourneert de methode 'true'. Anders is dit 'false'.
- **resultEnum** functionResult; het (functionele) resultaat van de methode (enum), bijv. **resultEnum**.fail_connection.
- **DatabaseSender** dbResult; de data die naar de database dient verstuurd te worden.

4.4.3 Variabelen

Bij de hiervoor genoemde methodes worden (speciale) variabelen gebruikt, deze staan hier uitgelegd:

variabele	uitleg
bool	Een (standaard C#) boolean.
DatabaseSender	Een object dat gebruikt wordt om gegevens van en naar de database te sturen.
Dictionary<x , y>	Een lijst-object dat gesorteerd/opgeslagen wordt op basis van de 'x'-waarde. De 'y'-waarde is de daadwerkelijke data.
Filter	Een object dat gebruikt wordt om te kunnen filteren.
ICalendarListResult	Een interface voor kalender-data.
IContactsJsonResult	Een interface voor contact-data.
ILoginResult	Een interface voor login-data.
IMessageResult	Een interface voor bericht-data.
Plugin	Een object dat de parent is van een SmA implementatie.
resultEnum	Een enum die aangeeft hoe de methode verliep. Mogelijkheden: • success (de methode is geslaagd) • fail_empty (de methode faalde omdat de inhoud of het resultaat van de aanroep leeg was) • fail_filter (de methode faalde omdat de (hiervoor verplichte) filter leeg was) • fail_connection (de methode faalde omdat er iets mis ging met de connectie) • fail_authorisation (de methode faalde omdat er iets mis ging met het autoriseren) • fail_wrongAccount (de methode faalde omdat er een ander account gebruik probeert te maken van de functionaliteit)
SmaPost	Een object waarin alle data staat die verstuurd moet worden naar de SmA.
SocialActionType	Een emum die aangeeft welke functionaliteit gebruikt dient te worden[34]. Mogelijkheden: • Calendar; opvragen van / inplannen in kalender • Contacts; opvragen van contacten • Login; inloggen van gebruiker • Message; versturen van een bericht naar contacten • LoginContacts[35]; alle rechten van Inloggen en Contacten • LoginContactsMessage[35]; alle rechten van Inloggen, Contacten en Berichten • LoginCalendar[35]; alle rechten van Inloggen en Kalender • ALL[35]; alle rechten van alle functionaliteit
SocialType	De SmA die gebruikt zal worden voor de functionaliteit[34]. Mogelijkheden: • Live • Google • LinkedIn • Facebook
string	Een (standard C#) String.

[34] Niet elke SmA ondersteund alle functionaliteit
[35] ALLEEN gebruikt bij **Redirect**.

5-6-2014

Versie 1.6

Requirements

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Wijzigingen

- Document opmaak aangepast

Inhoudsopgave

1 Inleiding

Dit document bevat alle requirements die voor dit project opgesteld zijn. Onder requirements wordt meer verstaan dan een lijst met eisen.

1.1 Tekstopmaak uitleg

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.
- **Variabele**; dit is een verwijzing naar een variabele die gebruikt wordt binnen een bepaalde context (bijvoorbeeld binnen een **afbeelding**). Bij de **Use case** wordt voor de variabelen de volgende tekstopmaak aangehouden:
    - Pagina; dit is een pagina dat de browser kan tonen. Een pagina kan van Datumprikker zelf zijn, of van een SmA.
    - Klasse; dit is een klasse die binnen de plug-in module aanwezig zal zijn.
    - Use case; dit is een andere use case.

1.2 Leeswijzer

Dit ontwerp bestaat uit de volgende onderdelen:

- **Eisstellers**; hier worden alle eisende partijen genoemd die invloed hebben op de requirements.
- **Requirements**; hier staan alle requirements die voor dit project gelden. De requirements zijn in 2 groepen te verdelen:
    - **Functioneel**; de requirements die aangeven wat de plug-in module moet kunen/doen.
    - **Niet-functioneel**; de requirements die aangeven waaraan de plug-in module zich moet voldoen (bijvoorbeeld op het gebied van veiligheid of onderhoud).
- **Use case**; hier staan alle use cases voor de plug-in module. Hierin staat omschreven hoe de plug-in functioneel zal werken. De use cases zijn opgesteld vanuit het perspectief van de gebruiker. Interne klassen en/of methoden zijn dus niet van belang.

Dit document houdt verschillende termen aan. Betekenissen en/of uitleg van deze termen staan vermeld in het document **Afstudeerverslag**, bijlage 2: Begrippenlijst.

2 Eisstellers

De partijen die hieronder staan hebben invloed op de requirements. Het betreft hier dus niet alle stakeholders van het hele project, maar de personen/groepen die eisen hebben over het product.

Id	Naam	functie
S1	Geert Merkelbach	opdrachtgever
S2	Facebook	SmA verstrekker
S3	LinkedIn	SmA verstrekker
S4a	Google PLUS	SmA verstrekker
S4b	Google Calendar	SmA verstrekker
S5a	Microsoft Office 365	SmA verstrekker
S5b	Microsoft Live	SmA verstrekker
S6	Stijn van der Meer	afstudeerder

3 Requirements

In dit hoofdstuk worden alle requirements opgenoemd die betrekking hebben op dit project, en het eindproduct wat hier bij hoort. Elke requirement heeft een id, belanghebbende stakeholder en de waarde. De waarde geeft aan hoe er aan de requirement voldaan moet worden:

- Kritiek: zonder deze requirement kan het eindproduct niet bestaan.
- Hoog: aan deze requirement dient voldaan te worden indien er geen alternatief bestaat.
- Middel: aan deze requirement dient alleen voldaan te worden indien hier tijd voor is binnen dit project.
- Laag: aan deze requirement hoeft niet voldaan te worden. Bij een vervolg project zou deze echter wel naar voren kunnen komen.

3.1 Functioneel

De hieronder staande requirements hebben betrekking op de functionele eisen voor het eindproduct. Elk requirement heeft een id, bestaande uit de volgende elementen:

- ‘F’; voor functioneel requirement.
- [getal]; een oplopend getal dat een requirement(groep) onderscheid.
- [letter]; indien het om een requirement-groep gaat, dan onderscheid deze letter de requirements binnen deze groep.

Id	Functionele requirement	Stakeholders	waarde
F1a	Er moet ingelogd kunnen worden met behulp van de SmA’s.	S1; S2; S3; S4a; S5	Kritiek
F1b	Er moeten contacten van de gebruiker opgehaald kunnen worden met behulp van de SmA’s.	S1; S2; S3; S4a; S5	Kritiek
F1c	Er moet agenda data van de gebruiker opgehaald kunnen worden met behulp van de SmA’s.	S1; S2; S4b; S5b	Kritiek
F1d	Er moeten berichten naar bekende accounts van de gebruiker gestuurd kunnen worden met behulp van de SmA’s.	S1; S2; S3 S4; S5	Kritiek
F1e	Er moet een afspraak toegevoegd kunnen worden aan de kalender van de SmA’s.	S1; S4b; S5a	Middel
F2	Er moet een module zijn die plug-ins kan bevatten.	S1; S6	Kritiek
F3	De plug-ins dienen elk met 1 SmA te kunnen communiceren.	S1; S6	Kritiek
F4	De plug-ins moeten JSON accepteren van de SmA’s.	S2; S3; S4; S5	Hoog
F5a	De plug-ins dienen met REST om te kunnen gaan.	S3; S4; S5	Hoog
F5b	De plug-ins dienen met Oauth V1a om te kunnen gaan.	S2; S3; S4	Hoog
F5c	De plug-ins dienen met Oauth V2 om te kunnen gaan.	S1; S2; S3; S4; S5	Hoog
F6	De van de SmA’s ontvangen data moet omgezet worden naar een generiek object.	S6	Hoog
F7	Plug-ins moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.	S6	Middel
F8	Templates moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.	S6	Middel

3.2 Niet-functioneel

De hieronder staande requirements hebben betrekking op de niet-functionele eisen voor het eindproduct. Elk requirement heeft een id, bestaande uit de volgende elementen:

- [code]; dit geeft aan onder welk type het requirement past. De verschillende types zijn:
  - Np: performance
  - No: onderhoud
  - Nv: veiligheid
  - Nb: betrouwbaarheid
- [getal]; een oplopend getal dat een requirement(groep) onderscheid binnen het type.
- [letter]; indien het om een requirement-groep gaat, dan onderscheid deze letter de requirements binnen deze groep.

Id	Niet-functionele requirement	Stakeholders	waarde
No1	De module moet een opzichzelfstaande structuur hebben.	S1; S6	Kritiek
Nv1	De module moet gekoppeld zijn aan een bestaand developer account van de social media.	S2	Kritiek
Nv2	De module moet geregistreerd zijn bij de social media.	S3; S5	Kritiek
Nv3	De module moet in de developers console geregistreerd worden binnen een bestaand account van de social media.	S4	Kritiek
Nb1	De plug-in module moet volgens de use cases handelen.	S1; S6	Kritiek
No2	De requirements F2a, F2b, F2c, F2d, F2e moeten los van elkaar geïmplementeerd kunnen worden.	S1; S6	Hoog
No3	Het generieke object van No2 zal per implementatie moeten verschillen.	S6	Hoog
No4	De website(s) van Datumprikker moeten weten wat voor type object (van No2) er ontvangen zal worden.	S6	Hoog
Nv4	De authenticatie dient gedaan te worden met Oauth V1a.	S2; S3; S4a	Hoog
Nv5	De authenticatie dient gedaan te worden met Oauth V2.	S2; S3; S4a; S5	Hoog
No5	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om aan de data van de SmA's te komen.	S6	Middel
No6	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om data te versturen naar de SmA's.	S6	Middel

4 Use cases

4.1 Inleiding

De plug-in module wordt in verschillende scenario's gebruikt. Deze scenario's worden in dit hoofdstuk uitgewerkt. De scenario's zijn onderverdeeld in 4 groepen:

- **Login**
- **Contacten**
- **Datums**
- **Overig**

4.2 Structuur/opmaak

Er worden verschillende structuur/opmaak regels aangehouden voor de use cases. Deze zijn te groeperen onder 3 categorieën:

- **Metadata**
- **Use case**
- **Overig**

4.2.1 Metadata

- Code; de code voor de use case.
- Versie; de versie van de use case.
- Doel; het doel van de use case.
- Actoren; de personen die betrokken zijn bij de use case (dit zal voor dit project in principe altijd de 'gebruiker' zijn).
- Triggers; de variabelen en/of use cases die deze use case starten.
- Pre-condities; de voorwaarden waaraan voldaan moeten worden voordat deze use case kan plaatsvinden. Deze voorwaarden gelden ook indien een trigger de use case (probeert) te starten.
- Alternatieve wegen; indien er tijdens een use case naar een andere use case gegaan moet worden, dan worden deze use cases hier aan gegeven (inclusief de code). Binnen de use case zelf wordt de code van de andere use case aangehouden.
- Post-condities; Het uiteindelijke resultaat na voltooien van de use case.

4.2.2 Use case

- Nr.; De regelnummer van de use case.
- Activiteit; de activiteiten van de use case. Deze kunnen positief of negatief beantwoord worden.
- Volgende stap (true); Indien de activiteit positief is (standaard), dient er naar de hier ingevulde regel gegaan te worden. Indien er naar een andere use case verwijst, dan staat de regelnummer van die use case achter de code.
- Volgende stap (false); Indien de activiteit negatief is, dient er naar de hier ingevulde regel gegaan te worden. Indien er naar een andere use case verwijst, dan staat de regelnummer van die use case achter de code.

4.2.3 Overig

- Om variabelen typen uit elkaar te kunnen houden, zijn de volgende kleurcodes gebruikt:
  - Pagina; dit is een pagina dat de browser kan tonen. Een pagina kan van Datumprikker zelf zijn, of van een SmA.
  - Klasse; dit is een klasse die binnen de plug-in module aanwezig zal zijn.
  - Use case; dit is een andere use case.
- Tijdelijke variabelen[36] worden aangeduid door er '$' voor te zetten.
- Indien een tijdelijke variabele aangemaakt wordt binnen een use case, dan wordt dit als volgt aangegeven: "(= $var_naam)".
- Een variabele die uit iets anders gehaald wordt, wordt als volgt genoteerd: "$voorbeeld.variabele".
- Indien er bij een trigger of na het uitvoeren van een alternatieve weg er naar een specifieke regel gegaan moet worden, dan wordt dit aangegeven met een "→", gevolgd door de regelnummer.
- Indien er 1 type variabele binnen een vaste selectie mogelijk is, dan staat dit aangegeven door "[var_a, var_b, etc.]".

4.3 Scenario's

De paragrafen hieronder betreffen de scenario's waarin er gebruik gemaakt wordt van de plug-in module. Het grootste gedeelte van deze scenario's wordt echter (bijna) volledig door Datumprikker zelf afgehandeld. De gebruiker heeft dus nooit directe interactie met de plug-in module, maar de keuzes die de gebruiker maakt, hebben wel invloed op het resultaat van de module.

4.3.1 Login

4.3.1.1 Inloggen met SmA

Code	**LiS**
Versie	2
Doel	Inloggen met behulp van een SmA.
Actoren	Gebruiker
Triggers	geen
Pre-condities	1. Er zijn 1 of meer SmA's van type login. 2. Browser staat op een pagina (= $begin) van Datumprikker.
Alternatieve wegen	1. **Standaard flow**(OsF) → 2 2. Fout! Ongeldig resultaat voor tabel. (LaK) → 4
Post-condities	Gebruiker is ingelogd met een SmA.

Nr.	**activiteit**	**Volgende stap (true)**	**Volgende stap (false)**
1	Gebruiker kiest 1 van de SmA (= $sma) mogelijkheden.	OsF-1	-
2	Is $sma gekoppeld aan bestaand account (= $account)?	4	LaK-1
3	Is $begin gelijk aan home?	4	5
4	Browser toont account van gebruiker.	-	-
5	Browser toont $begin, met aangepaste data voor de gebruiker.	-	-

[36] Wordt gebruikt om kleine verschillen aan te geven die binnen de use case gebruikt worden. Hierdoor hoeven er niet meerdere use cases gemaakt te worden die praktisch hetzelfde zijn.

4.3.1.2 *SmA account koppelen*

Code	**LaK**
Versie	2
Doel	Inloggen met behulp van een SmA.
Actoren	Gebruiker
Triggers	**Inloggen met SmA** (LiS)
Pre-condities	1. Er is een SmA(= $sma) geselecteerd. 2. Browser staat op een pagina (= $begin) van Datumprikker. 3. Er is data(= $data) opgevraagd.
Alternatieve wegen	geen
Post-condities	Gebruiker heeft een (nieuw) account bij Datumprikker met SmA inlog mogelijkheden.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Bevat $data een emailadres (= $email)?	3	2
2	Browser vraagt gebruiker om (hoofd)emailadres (= $email)	3	-
3	Bestaat $email als account (= $account)?	7	4
4	Datumprikker controleert of gebruiker al een bestaand account heeft.	5	-
5	Heeft gebruiker al een bestaand account?	7	6
6	Datumprikker maakt een nieuw $account, gekoppeld aan de $sma.	8	-
7	$sma is gekoppeld aan $account.	8	-

4.3.2 Contacten

4.3.2.1 *Ophalen*

Code	**Co**
Versie	2
Doel	Contacten ophalen van een SmA.
Actoren	Gebruiker
Triggers	**Bericht sturen** (Cb)
Pre-condities	1. Er zijn 1 of meer SmA's van type contact. 2. Browser staat op [afspraak_maken / account] (= $begin)
Alternatieve wegen	**Standaard flow** (OsF) → 3
Post-condities	Contacten van gebruiker zijn opgehaald.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Gebruiker kiest 1 van de SmA (= $sma) mogelijkheden.	2	-
2	Datumprikker selecteert de pagina(= $pagina) van de $sma.	OsF-1	-
3	Bevat $data contacten (= $contacten)?	4	5
4	Browser toont $begin.	-	-
5	Browser toont $begin met $contacten.	-	-

4.3.2.2 *Bericht sturen*

Code	**Cb**
Versie	1
Doel	Bericht sturen naar geselecteerde contacten van gebruiker.
Actoren	Gebruiker
Triggers	geen
Pre-condities	1. Er zijn 1 of meer SmA's van type contact. 2. Browser staat op [afspraak_maken / afspraak_bekijken] (= $begin) 3. De gebruiker heeft voor alle bijbehorende SmA's, Datumprikker geautoriseerd. 4. Er is een $token voor alle bijbehorende SmA's.
Alternatieve wegen	1. **Authentiseren** (OaH) → 7 2. **SmA data versturen** (OsdV) → 8 3. **Ophalen** (Co) → 2

Post-condities

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Datumprikker toont gebruiker een invoerveld(= $veld).	2	-
2	Gebruiker vult $veld in met tekst(= $bericht).	3	-
3	Datumprikker haalt de contacten (= $contacten) op uit $begin.	4	-
4	Datumprikker stopt elk type SmA(= $sma) uit $contacten in een lijst (= $sma_lijst)	5	-
5	Bevat $sma_lijst een $sma die het type bericht ondersteunt?	6	9
6	Datumprikker authentiseert zich bij de $sma met behulp van de $token.	OaH-1	-
7	Datumprikker verstuurt $bericht naar de $contacten die bij de $sma horen.	OsDv-1	-
8	Verwijder $SmA uit $SmA_lijst.	5	-
9	Datumprikker verstuurd alle $contacten waar nog geen bericht naar gestuurd is, een email.	10	-
10	Browser bevestigd gebruiker dat de berichten verstuurd zijn.	-	-

4.3.3 Datums

4.3.3.1 controleren

Code	**Dc**
Versie	2
Doel	Datums controleren met een SmA
Actoren	Gebruiker
Triggers	geen
Pre-condities	1. Er zijn 1 of meer SmA's van type agenda. 2. Browser staat op beschikbaar_aangeven.
Alternatieve wegen	**SmA data opvragen** (SdO) → 3
Post-condities	Beschikbaarheid van gebruiker is gecontroleerd.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Gebruiker kiest 1 van de SmA (= $SmA) mogelijkheden.	2	-
2	Datumprikker selecteert de pagina van de $SmA (= $pagina)	SdO-1	-
3	Datumprikker controleert datums van $SmA.	4	-
4	Browser toont beschikbaar_aangeven met aangepaste data voor de gebruiker.	-	-

4.3.3.2 Inplannen

Code	**Di**
Versie	1
Doel	Geprikte afspraak in een SmA agenda plaatsen.
Actoren	Gebruiker
Triggers	geen
Pre-condities	1. Er zijn 1 of meer SmA's van type agenda. 2. Datum is geprikt. 3. Browser staat op afspraak_bekijken.
Alternatieve wegen	**SmA data opvragen** (SdO) → 3
Post-condities	Afspraak staat genoteerd in agenda SmA.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Gebruiker kiest 1 van de SmA (= $SmA) mogelijkheden.	2	-
2	Datumprikker selecteert de pagina van de $SmA (= $pagina)	SdO-1	-
3	Datumprikker plaatst de afspraak in de agenda van de SmA.	4	-
4	Browser bevestigd gebruiker dat de afspraak in de agenda gezet is.	-	-

4.3.4 Overig

4.3.4.1 Standaard flow

Code	**OsF**
Versie	1
Doel	Het autoriseren, authentiseren en data opvragen in de juiste volgorde uitvoeren.
Actoren	Gebruiker
Triggers	Een andere use case
Pre-condities	1. Er is een SmA(= $sma) geselecteerd
Alternatieve wegen	1. **Autoriseren** (AuO) → 3 2. **Authentiseren** (AuH) → 4 3. **SmA data opvragen** (SdO) → 5
Post-condities	Er is data(= $data) van de gebruiker beschikbaar.

Nr.	**activiteit**	**Volgende stap (true)**	**Volgende stap (false)**
1	Datumprikker selecteert de bijbehorende pagina(= $pagina) van de $sma.	2	-
2	Datumprikker opent een pop-up window(= $window).	OaO-1	-
3	Datumprikker authentiseert zich bij de $sma met behulp van de $token.	OaH-1	-
4	Datumprikker vraagt $data op aan $sma.	OsDo-1	-
5	Datumprikker verwerkt de $data.	-	-

4.3.4.2 Autoriseren

Code	**OaO**
Versie	1
Doel	Het autoriseren van Datumprikker voor het gebruik van data van de gebruiker bij een SmA.
Actoren	Gebruiker
Triggers	Een andere use case (= $uc)
Pre-condities	2. Er is een pop-up window(= $window) open gemaakt. 3. $uc Heeft een pagina(= $pagina) van een SmA(= $sma) geselecteerd.
Alternatieve wegen	geen
Post-condities	De gebruiker heeft Datumprikker geautoriseerd.

Nr.	**Activiteit**	**Volgende stap (true)**	**Volgende stap (false)**
1	$window Gaat naar de $pagina.	2	-
2	Is de gebruiker ingelogd bij de $sma?	4	3
3	De gebruiker vult de inlog gegevens in.	4	
4	Heeft de gebruiker Datumprikker geautoriseerd?	6	5
5	De gebruiker autoriseert Datumprikker voor het gebruik van de gevraagde toegang.	6	-
6	De $sma geeft Datumprikker een autorisatie-code(= $code).	7	-
7	Het $window wordt gesloten.	-	-

4.3.4.3 Authentiseren

Code	**OaH**
Versie	1
Doel	Het authentiseren van een gebruiker.
Actoren	geen
Triggers	Een andere use case (= $uc)
Pre-condities	1. Er is een SmA(= $sma) geselecteerd. 2. Er is een bedrijfsleutel(= $key) en geheim(= $secret) combinatie van de $sma. 3. $uc Heeft één van de volgende waarde-combinaties: a. $code, en geen/lege account gegevens(= $account) van de gebruiker b. $token, en bestaande account gegevens(= $account) van de gebruiker
Alternatieve wegen	**Foutmelding afhandeling** (OfA)
Post-condities	De gebruiker is geauthentiseerd.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Is er een $code?	3	2
2	Datumprikker authentiseert zichzelf met behulp van de $code, en de $key-$secret combinatie.	4	-
3	Datumprikker her-authentiseert zichzelf met behulp van de $token, en de $key-$secret combinatie.	4	-
4	Is de authenticatie gelukt?	5	OfA-1
5	De $sma retourneert een token(= $token), en eventueel een refresh-token(= $refresh)	6	-
6	Datumprikker vraagt account gegevens (= $check) op aan de $sma.	7	-
7	Komen $check en $account met elkaar overeen?	-	OfA-2

4.3.4.4 SmA data opvragen

Code	OsDo
Versie	2
Doel	Het opvragen van data.
Actoren	geen
Triggers	Een andere use case (= $uc)
Pre-condities	1. De gebruiker is geauthentiseerd. 2. Er is een SmA(= $sma) geselecteerd. 3. Er is een $token beschikbaar van de $sma.
Alternatieve wegen	**Foutmelding afhandeling** (OfA)
Post-condities	Er is data(= $data) van de gebruiker opgevraagd.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Verwacht de $sma een bedrijf sleutel(= $key) en geheim(= $secret) combinatie?	2	3
2	Datumprikker vraagt data op van gebruiker aan de $sma, met behulp van de $token en de $key-$secret combinatie.	4	-
3	Datumprikker vraagt data op van gebruiker aan de $sma, met behulp van de $token.	4	-
4	Ging het opvragen van de data succesvol?	5	OfA-3
5	De $sma retourneert de $data van de gebruiker naar Datumprikker.	-	-

4.3.4.5 SmA data versturen

Code	OsDv
Versie	1
Doel	Het versturen van data.
Actoren	geen
Triggers	Een andere use case (= $uc)
Pre-condities	1. De gebruiker is geauthentiseerd. 2. Er is een SmA(= $sma) geselecteerd. 3. Er is een $token beschikbaar van de $sma.
Alternatieve wegen	**Foutmelding afhandeling** (OfA)
Post-condities	Er is data(= $data) van de gebruiker verstuurd naar de $sma.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	Verwacht de $sma een bedrijf sleutel(= $key) en geheim(= $secret) combinatie?	2	3
2	Datumprikker verstuurt data van gebruiker naar de $sma, met behulp van de $token en de $key-$secret combinatie.	4	-
3	Datumprikker verstuurt data van gebruiker naar de $sma, met behulp van de $token.	4	-
4	Ging het opvragen van de data succesvol?	5	OfA-4
5	De $sma retourneert de $data van de gebruiker naar Datumprikker.	-	-

4.3.4.6 *Foutmelding afhandeling*

Code	**OfA**
Versie	1
Doel	Het afhandelen van (verwachte) foutmeldingen.
Actoren	Gebruiker
Triggers	Een andere use case (= $uc)
Pre-condities	geen
Alternatieve wegen	geen
Post-condities	Datumprikker toont een foutmelding(= $message) aan de gebruiker.

Nr.	Activiteit	Volgende stap (true)	Volgende stap (false)
1	$message Is: "Er is iets fout gegaan bij het inloggen.".	-	-
2	$message Is: "Je kan maar met één account van een social media ingelogd zijn.".	-	-
3	$message Is: "Er is een fout opgetreden bij het ophalen van de data.".	-	-
4	$message Is: "Er is een fout opgetreden bij het versturen van de data.".	-	-

5-6-2014

Versie 1.7

Ontwerp

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Wijzigingen

- Pagina opmaak aangepast

Inhoudsopgave

1 Inleiding

Dit document bevat het meest recente ontwerp voor de Social media plug-in module.

1.1 Tekstopmaak uitleg

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.
- **Variabele**; dit is een verwijzing naar een variabele die gebruikt wordt binnen een bepaalde context (bijvoorbeeld binnen een **afbeelding**).
- **Package**; dit is een **variabele** dat een package voorstelt binnen een context.
- **Methode**; dit is een **variabele** dat een methode voorstelt binnen een context.

1.2 Leeswijzer

Dit ontwerp bestaat uit de volgende onderdelen:

- **Diagram**; dit is een visuele weergave van hoe de plug-in module in elkaar zit.
  - **Totaal**; hier wordt het diagram in zijn totaliteit weergegeven.
  - **Packages**; hier worden de packages van het diagram uitvergroot en (diens klassen) uitgelegd.
- **Flow-chart**; dit is een visuele en tekstuele weergave van hoe de module globaal werkt.

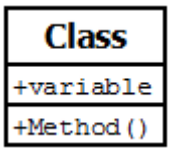
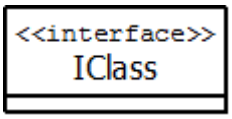
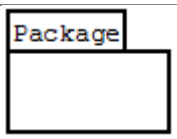
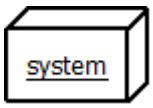
Dit document houdt verschillende termen aan. Betekenissen en/of uitleg van deze termen staan vermeld in het document **Afstudeerverslag**, bijlage 2: Begrippenlijst.

2 Diagram

Het ontwerp bestaat uit 5 packages:

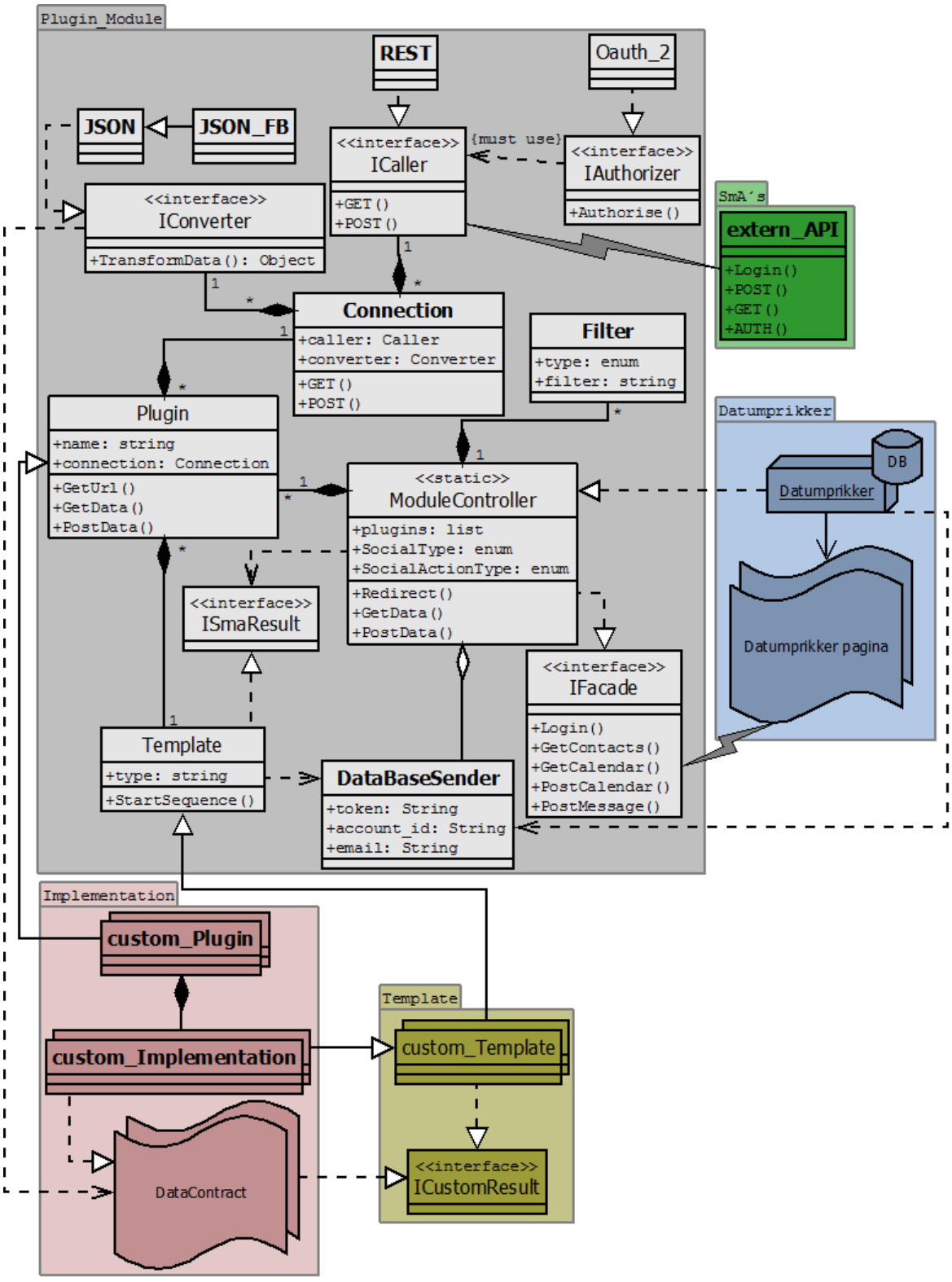
- **Plugin_Module**; deze package is de plug-in module zelf.
- **Datumprikker**; deze package is de reeds bestaande web applicatie van Datumprikker. In principe is dit dus een externe package. Er zullen hier echter wel wat wijzigingen in gemaakt worden om de communicatie met deze module te kunnen maken.
- **SmA's**; deze package is extern. Het zijn de API's van de social media die gebruikt worden om data op te vragen.
- **Template**; Deze package wordt gebruikt als blauwdruk voor de implementatie.
- **Implementatie**; Deze package wordt gebruikt om een SmA te integreren binnen de module.

Een (UML) diagram bestaat uit een aantal verschillende onderdelen:

Item	Betekenis
	De linkerkant van de pijl implementeert de interface die aan de rechterkant staat.
	De linkerkant van de pijl erft over van de rechterkant.
	De linker- en rechterkant van de lijn hebben een associatie met elkaar. Indien er een pijltje bijstaat (zoals bij dit icoon), dan is er sprake van een uni-directionele associatie. Anders is er sprake van een bi-directionele associatie.
	De linkerkant van de lijn is de aggregaat van de rechterkant.
	De linkerkant van de lijn is de compositie van de rechterkant.
	De linkerkant van de pijl heeft belangen bij de rechterkant, maar heeft geen (directe) associatie. Indien er tekst bij staat (dit staat tussen '{' en '}'), dan legt de linkerkant een restrictie/eis op aan de rechterkant.
	Binnen UML is het niet mogelijk om communicatie tussen 2 (externe) omgevingen weer te geven. Om dit toch weer te geven, is dit icoon gebruikt. De 2 uiteinden kunnen hierdoor wel met elkaar communiceren, maar hebben dus geen associatie met elkaar.
	Een klasse. Indien de titel niet **vetgedrukt** is, dan is er sprake van een abstracte of statische klasse. In het 2e vakje staan alle variabelen die voor het ontwerp van belang zijn. In het 3e vakje staan alle methodes die voor het ontwerp van belang zijn.
	Een interface. De titel begint altijd met een 'I'. Ook hier zijn de 2e en 3e vakjes voor variabelen en methodes.
	Een package. Alles wat hier binnen zit, hoort bij de package.
	Een bestand dat binnen het ontwerp gebruikt wordt, maar waarvan de inhoud niet van belang is. Dit icoon wordt niet binnen UML gebruikt, maar maakt het mogelijk om items die geen (concrete) klasse zijn toch weer te kunnen geven.
	Een (extern) systeem. Dit icoon wordt niet binnen UML gebruikt, maar maakt het mogelijk om een volledig systeem weer te geven, zonder de inhoud te weten/weergeven.
	Een database. Dit icoon wordt niet binnen UML gebruikt, maar maakt het mogelijk om aan te geven waar er een connectie is met de database.

2.1 Totaal

Hieronder staat het totale ontwerp van de module afgebeeld:



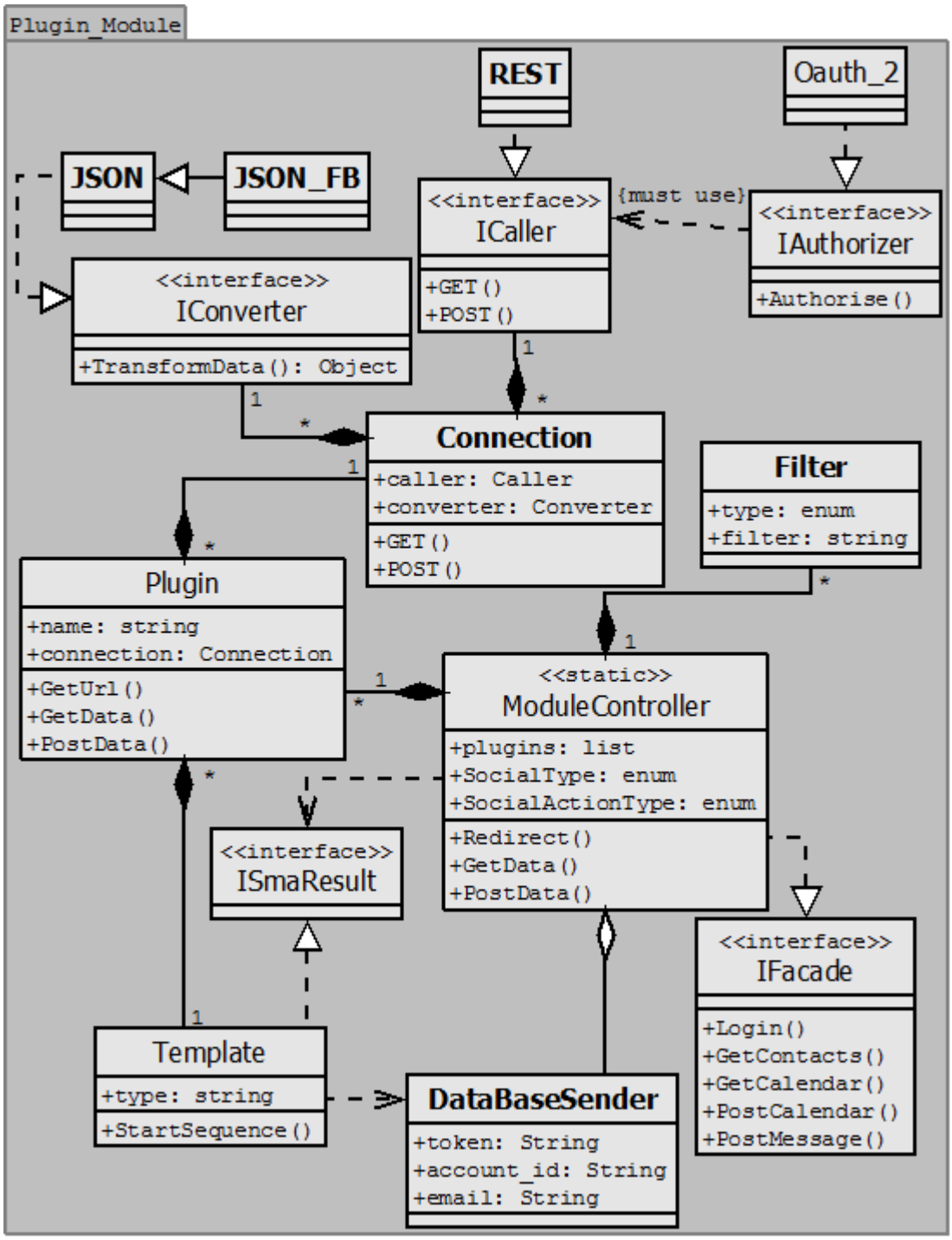
Figuur 18, Totaal ontwerp

2.2 Packages

In dit paragraaf worden alle packages los behandeld.

2.2.1 Plugin_Module

In **Figuur 19** staat het hoofd-package afgebeeld. In essentie is dit de volledige plug-in module zelf. Alle andere packages zijn koppelingen en/of extensies aan deze module.



Figuur 19, Plugin_Module ontwerp

De volgende items komen in deze package voor:

- **ModuleController**
- **Plugin**
- **Connection**
- **IConverter**
- **JSON**
- **JSON_FB**
- **ICaller**
- **IAuthorizer**
- **REST**
- **Oauth_2**
- **Template**
- **Filter**
- **ISmaResult**
- **DataBaseSender**
- **IFaçade**

Deze items zullen in de paragrafen **2.2.1.1** t/m **2.2.1.15** kort beschreven worden.

2.2.1.1 ModuleController

Deze statische klasse wordt aangeroepen door de **Datumprikker** package om data op te vragen en/of verzenden. Het type van data wordt bepaald door de **SocialActionType** enum. De SmA wordt bepaald door de **SocialType** enum. De klasse implementeert de IFaçade interface, en ontvangt objecten die de interface ISmaResult implementeren.

Om te kunnen communiceren met de **SmA's** zijn er inlog gegevens nodig. Deze worden door **Datumprikker** aangeleverd, en doorgegeven aan deze klasse.

2.2.1.2 Plugin

Deze abstracte klasse bevat alle variabelen en (abstracte) methodes die nodig zijn voor het opvragen en verzenden van data. De klasse communiceert met de klasse **Connection** om te kunnen communiceren met de SmA's.

Deze klasse wordt overgeërfd door de **custom_Template** klasse van **Implementatie**.

2.2.1.3 Connection

Deze klasse handelt alle communicatie met de SmA's af. Dit wordt mogelijk gemaakt door de interfaces **Converter**, **Caller** en **Authorizer**. Elke **Connection** heeft 1 implementatie van elk van deze interfaces. De (overgeërfde klasse van de) **Plugin** klasse bepaalt welke implementaties van de interfaces voor **Connection** nodig zijn om te communiceren met de desbetreffende SmA.

2.2.1.4 IConverter

Deze interface moet het mogelijk maken om ontvangen data van een SmA om te zetten naar een generiek object (**ISmaResult**). Per type data dat ontvangen moet kunnen worden, dient er een aparte overervende klasse te zijn. Op dit moment is dat alleen nog de **JSON** klasse.

Het generieke object dat deze klasse produceert, zal gebaseerd zijn op de **Datacontracten** van de **Implementatie**. Dit object zal dan via de **Connection** naar de **Plugin** gestuurd worden.

2.2.1.5 JSON

Dit is de overervende klasse van **Converter** die gebruikt zal worden voor dit project. Het zal de geretourneerde JSON data van de SmA's omzetten naar een generiek object (**ISmaResult**).

2.2.1.6 JSON_FB

Deze klasse erft over van de **JSON** klasse, en is puur bedoelt voor de Facebook SmA. Het is volledig hetzelfde als de **JSON** klasse, met als enige uitzondering dat het alle binnen gekomen data eerst controleert, en daarna (indien nodig) omgezet worden naar JSON. Dit heeft te maken met een fout van Facebook, dat in het document **Afstudeerverslag**, hoofdstuk 4.1.2.5 beschreven wordt.

2.2.1.7 ICaller

Deze interface moet het mogelijk maken om te communiceren met de SmA's. De overervende klassen zullen via een specifiek protocol deze communicatie uitvoeren. Alle ontvangen data van de aangeroepen SmA zullen door de **Converter** omgezet worden.

2.2.1.8 IAuthorizer

Om een beveiligde verbinding tot stand te brengen, dient er op een bepaalde manier geauthentiseerd te worden. Dit wordt gedaan doormiddel van deze interface. De reden dat dit geen volwaardige interface is, is omdat het authentiseren meestal op een manier gaat die de **Caller** ondersteund/integreert. De authenticatie zal dus geïnitieerd worden door de **Caller**.

2.2.1.9 REST

Dit is de overervende klasse van **Caller** die gebruikt zal worden voor dit project. Het zal doormiddel van REST communiceren met de **SmA's**.

2.2.1.10 Oauth_2

Dit is de autoriseer methode die gebruikt zal worden. Deze methode zal dus geïntegreerd worden met de **REST** klasse.

2.2.1.11 Template

Dit is de abstracte klasse waar alle **Templates** van zullen overerven. Het bevat tevens de **ISmaResult** interface. Hiermee verplicht de klasse dat alle **custom_Implementations** van **Implementatie** de interface implementeren.

2.2.1.12 Filter

Deze klasse bevat alle informatie die nodig is om (eventueel) te kunnen filteren op data.

2.2.1.13 ISmaResult

Deze interface wordt gebruikt om de van de SmA ontvangen data terug te sturen naar **Datumprikker**. **Template** (en diens overervende klassen) implementeren deze interface.

2.2.1.14 DataBaseSender

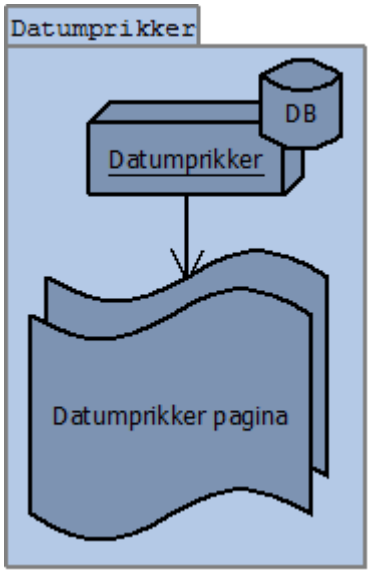
Deze klasse wordt gebruikt om de van de SmA ontvangen (database specifieke) data te versturen naar de **Datumprikker** klasse. Het versturen naar de database zelf gebeurd binnen **Datumprikker**.

2.2.1.15 IFaçade

In deze interface bevinden zich alle methodes die externe clients (Datumprikker) kunnen gebruiken om te kunnen communiceren met de plug-in module. Het versimpeld de aanroepen.

2.2.2 Datumprikker

Dit ontwerp is puur bedoeld om aan te geven welke onderdelen van **Datumprikker** er met de **Plug-in module** zullen praten. Zoals te zien in **Figuur 20**, bestaat de package uit 3 items:



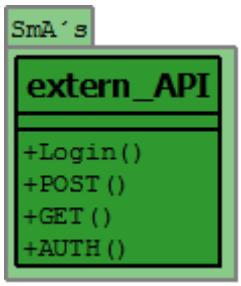
Figuur 20, Datumprikker ontwerp

- Datumprikker; deze klasse bevat alle logica voor de (hoofd)applicatie.
- DB; dit stelt de database voor. Alle communicatie hiermee wordt door Datumprikker gedaan.
- Datumprikker pagina; dit item stelt alle webpagina's voor die gebruik zullen maken van de **Plugin_Module**.

Hoe de structuur er voor de rest uit ziet binnen deze package doet er niet toe voor de andere packages.

2.2.3 SmA's

De **SmA's** package(**Figuur 21**) is er om aan te geven welk item van de **Plugin_Module** er met de SmA's communiceert. De functies die in de klasse **extern_API** staan, zijn er om te illustreren van welke functionaliteit er gebruik gemaakt zal worden.

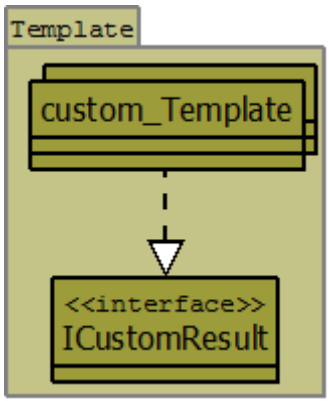


Figuur 21, SmA's ontwerp

De precieze structuur van deze package doet er niet toe voor de andere packages. Het enige wat van belang is voor de **Plugin_Module**, is de exacte methodes die aangeroepen moeten worden. De namen (URL's) hiervan staan opgeslagen in de **custom_Implementation**.

2.2.4 Template

De **Template** package (**Figuur 22**) is de blauwdruk voor de **Implementatie** package. De abstracte klasse **custom_Template** is bedoelt om generieke data op te vragen en verzenden. Elke instantie van de klasse heeft een eigen **ICustomResult** interface. Aan de hand hiervan wordt er SmA-onafhankelijk dezelfde data geretourneerd. De reden dat **custom_Template** een multi-object is, is omdat de klasse zelf helemaal niet voorkomt. In plaats daarvan is er voor elk type implementatie (Inloggen, Contacten, etc.) een aparte klasse. De klassen zijn echter voor wat het ontwerp betreft exact hetzelfde. Vandaar dat dit zo wordt weergegeven.



Figuur 22, template ontwerp

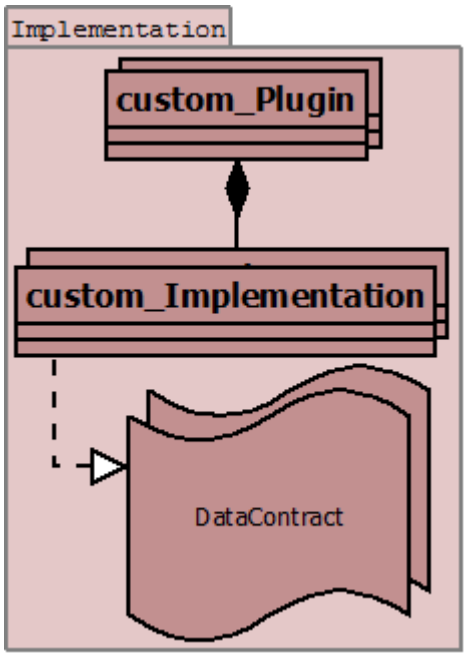
2.2.5 Implementatie

De **Implementatie** package die te zien is in **Figuur 23**, wordt gebruikt om alles wat specifiek is voor een SmA geldt, bij te houden.

Zo wordt de klasse **Plugin** van de **Plugin_Module** door **custom_Plugin** overgeërfd, en erft de klasse **custom_Implementation**, de klasse **custom_Template** van de **Template** package over.

Om de data van de SmA's om te kunnen zetten naar generieke objecten, worden er **Datacontracten** gebruikt. Omdat deze specifiek zijn voor een SmA, staan deze dan ook in deze package. De **Converter** klasse van de **Plugin_Module** zal deze **Datacontracten** gebruiken. Uiteindelijk wordt het resultaat van de SmA's via een custom_Implementation omgezet naar een **ISmaResult** implementerende instantie.

De klassen **custom_Plugin** en **custom_Implementation** komen net zoals bij **custom_Template** van **Template** niet echt voor in de applicatie. Elke klasse heeft een aantal andere daadwerkelijke klassen, die qua ontwerp hetzelfde zijn. Het ontwerp zou veel groter worden indien deze klassen allemaal getekend zouden worden. Dit zou het onoverzichtelijk en daarnaast volledig overbodig maken.



Figuur 23, Implementatie ontwerp

2.3 Design patterns

Bij het ontwerp voor de plug-in module, zijn de volgende design patterns[37] gebruikt:

- **Abstract factory**
- **Adapter**
- **Strategy**
- **Façade**
- **Proxy**

Deze patterns worden hier in de bijbehorende paragrafen uitgelegd, met daarbij 2 afbeeldingen:

- Globaal; hier wordt het pattern volgens de regels afgebeeld.
- Concreet; hier wordt het pattern met behulp van het ontwerp afgebeeld.

2.3.1 Abstract factory[38]

'De abstract factory pattern geeft de mogelijkheid om een groep van individuele factories met een overeenkomend thema in te kapselen, zonder de klassen te hoeven specificeren. Normaal gesproken maakt de client een implementatie van de AbstractFactory interface aan, en gebruikt dan de generieke methode van de interface om de bij het thema horende objecten aan te maken.

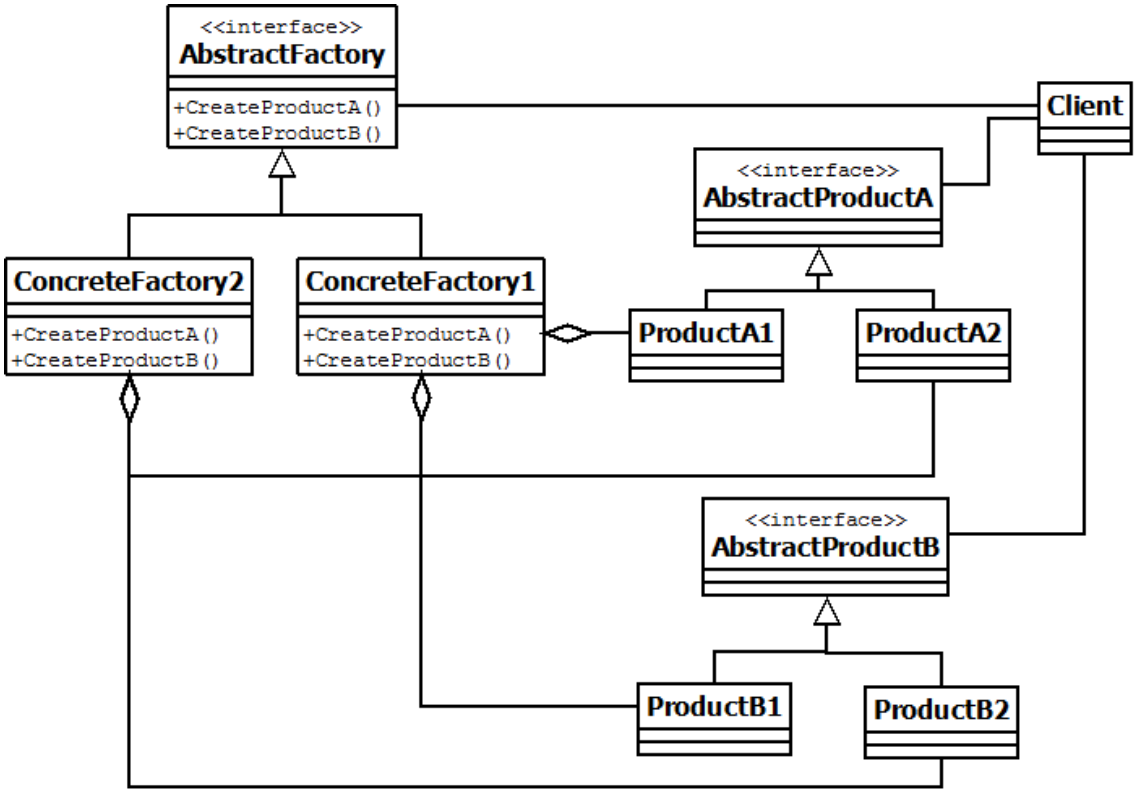
De client heeft geen kennis over/behoefte voor de concrete objecten die de interne factories aanmaken, omdat de bijbehorende interfaces aangeroepen zullen worden.

Door dit pattern te gebruiken, wordt het mogelijk gemaakt om concrete implementaties aan te maken en verwisselen, zonder dat de code hiervoor hoeft aangepast te worden. Gebruik van dit pattern kan het ontwerp en de code echter (onnodig) complex maken.' [39]

[37] Uitleg over hoe de (UML) ontwerpen opgebouwd worden, staat vermeld in **Ontwerp**, hoofdstuk 2.
[38] Bron: http://www.lepus.org.uk/ref/companion/AbstractFactory.xml
[39] Bron: http://en.wikipedia.org/wiki/Abstract_factory_pattern

2.3.1.1 Globaal ontwerp

In het onderstaande figuur staat een globale weergave van hoe een abstract factory er uit ziet:

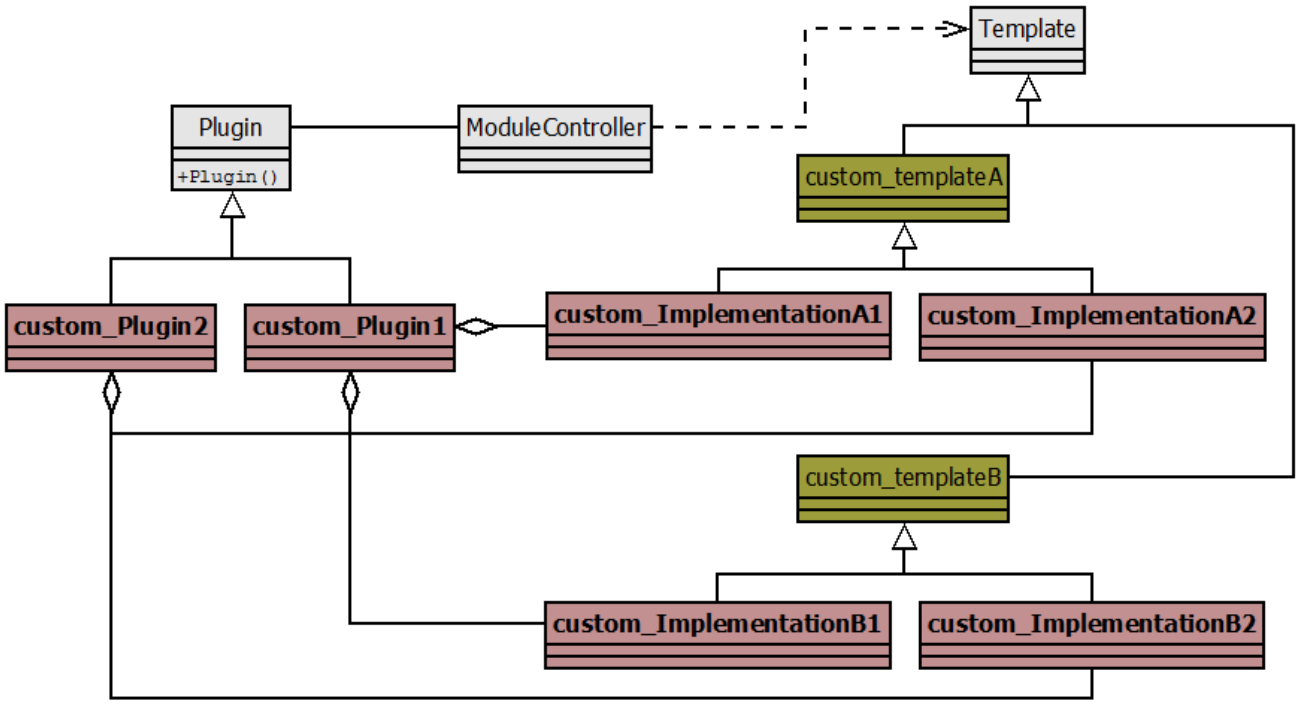


Figuur 24, Abstract factory (globaal) [38]

In Figuur 24 is te zien hoe een abstract factory eruit ziet. Hierbij is de Client de klasse die het proces start. Hij maakt een implementatie van de AbstractFactory interface aan. Dit is dus 1 van de ConcreteFactory's. Daarna kunnen de functies van de interface gebruikt worden om Producten aan te maken.

2.3.1.2 Concreet ontwerp

In het onderstaande figuur staat de concrete versie van de globale weergave. De klassen van het ontwerp zijn hiervoor gebruikt. Alle onnodige variabelen en functies zijn gestript, en er is waar nodig informatie toegevoegd om de overeenkomst met de globale weergave duidelijk te maken.



Figuur 25, Abstract factory (concreet)

Zoals te zien in **Figuur 25**, komt het ontwerp bijna 1 op 1 overeen met de globale variant. De uitzonderingen zijn:

- **Plugin**(AbstractFactory) is een abstracte klasse, geen interface. Een abstracte klasse lijkt veel op een interface, alleen dan met overervingsmogelijkheden en eigen variabelen en methodes. Dit is binnen het ontwerp nodig, omdat de klasse ook andere functionaliteiten aanbiedt binnen de module.
- **ModuleController**(Client) maakt niet direct gebruik van **Template** (of de overervende klassen). De klasse heeft **Template** wel nodig (vandaar de dependency lijn), maar communicatie gebeurt via een interface, die niet van belang is voor dit pattern.
- De methodes die gebruikt worden zijn de constructors van de klassen, niet losse methodes. Hierdoor worden de **custom_Implementation**s(Product) direct aangemaakt bij creatie van de **Implementation**(ConcreteFactory).

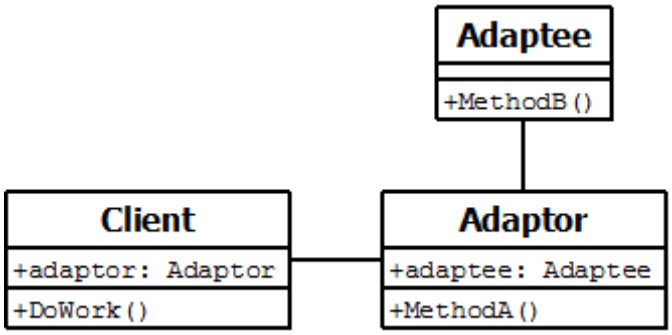
De reden dat deze abstract factory nodig is, komt doordat er voor elke SmA een aparte **Implementation** is. En deze hebben dan weer voor elke ondersteunde functie een aparte **custom_Implementation** nodig. Voor de **ModuleController** is dit allemaal niet van belang, zolang er uiteindelijk maar data ontvangen/verzonden wordt van/naar de SmA's.

2.3.2 Adapter[40]

'De adapter pattern maakt het mogelijk om 2 incompatibele interfaces met elkaar te laten werken. Het werkt dus net zoals een echte adapter.

De interfaces zijn dan misschien incompatibele, maar de interne functionaliteit moet wel overeen komen. Het pattern laat deze, anders incompatibele klassen, samenwerken door 1 van de klassen om te zetten naar een interface die de client-klasse verwacht.' [41]

2.3.2.1 Globaal ontwerp



Figuur 26, Adapter (globaal) [40]

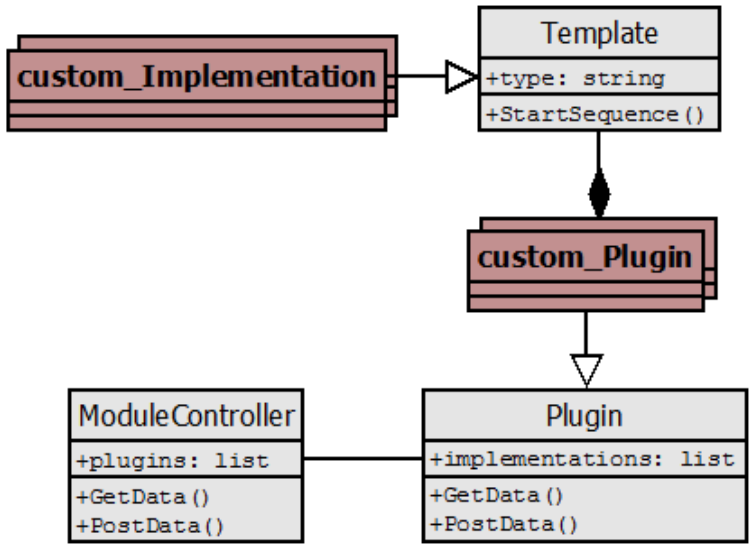
Zoals te zien in **Figuur 26**, heeft de **Adaptor** klasse niet dezelfde methode als **Adaptee**. Aangezien de **Client** klasse alleen bekend is met de **Adaptor** klasse, kan het nooit de methode van **Adaptee** uitvoeren. Om dit toch mogelijk te maken, is de **Adaptor** klasse wel bekend met de **Adaptee** klasse, en wordt de bijbehorende methode aangeroepen binnen de eigen methode.

Als **Client** de methode van **Adaptee** wil gebruiken, dan moet de klasse dus:

- **DoWork()** gebruiken;
- hierin wordt **Adaptor**.**MethodA()** aangeroepen;
- **Adaptor**.**MethodA()** roept **Adaptee**.**MethodB()** aan, en retourneert het resultaat volgens **MethodA()**;

[40] Bron: http://www.lepus.org.uk/ref/companion/Adapter(Object).xml
[41] Bron: http://en.wikipedia.org/wiki/Adapter_pattern

2.3.2.2 Concreet ontwerp



Figuur 27, Adapter (concreet)

Zoals te zien in Figuur 27, komt het (deel)ontwerp exact overeen met de globale variant op 2 onderdelen na:

- Overerving van Plugin(Adaptor) door custom_Plugin;
- Overerving van Template(Adaptee). door custom_Implementation;

Dit heeft echter geen enkele invloed op de werking van de pattern.

Wat opvalt, is dat er hier sprake is van een dubbele adapter. De methodes **GetData()** en **PostData()** maken namelijk allebei gebruik van de pattern.

De reden dat hier een adapter nodig is, komt door het feit dat, hoewel er uiteindelijk data verzonden of ontvangen wordt, er hier een hele hoop aanroepen/controles achter elkaar gedaan moeten worden doormiddel van de **startSequence()** methode. Zowel **GetData()** en **PostData()** roepen deze methode (indirect) aan, en het resultaat verschilt behoorlijk.

2.3.3 Strategy[42]

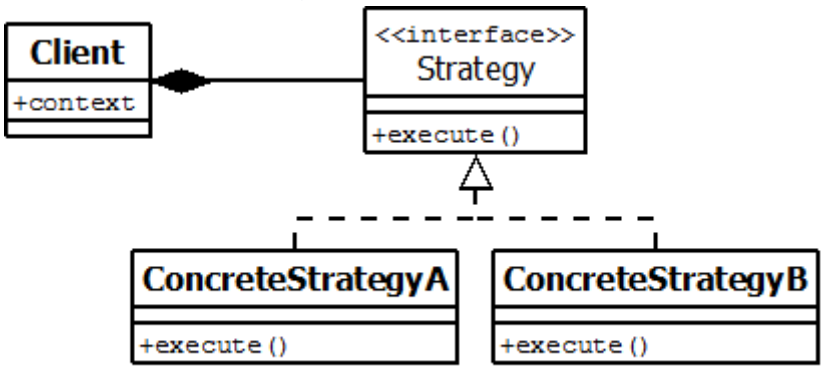
'De strategy pattern maakt het mogelijk om het gedrag van een algoritme te selecteren tijdens run-time. Dit wordt mogelijk gemaakt door:

- Definiëren van een familie van algoritmes;
- Inkapselen van elk algoritme;
- Er voor zorgen dat de algoritmes verwisselbaar zijn binnen de familie; '[43]

Hierdoor bepaald de client dus welk algoritme uitgevoerd zal worden. Deze keuze kan zowel van te voren door de client bepaald zijn (en dus altijd hetzelfde zijn), of veranderen op basis van (externe) variabelen.

[42] Bron: http://www.lepus.org.uk/ref/companion/Strategy.xml
[43] Bron: http://en.wikipedia.org/wiki/Strategy_pattern

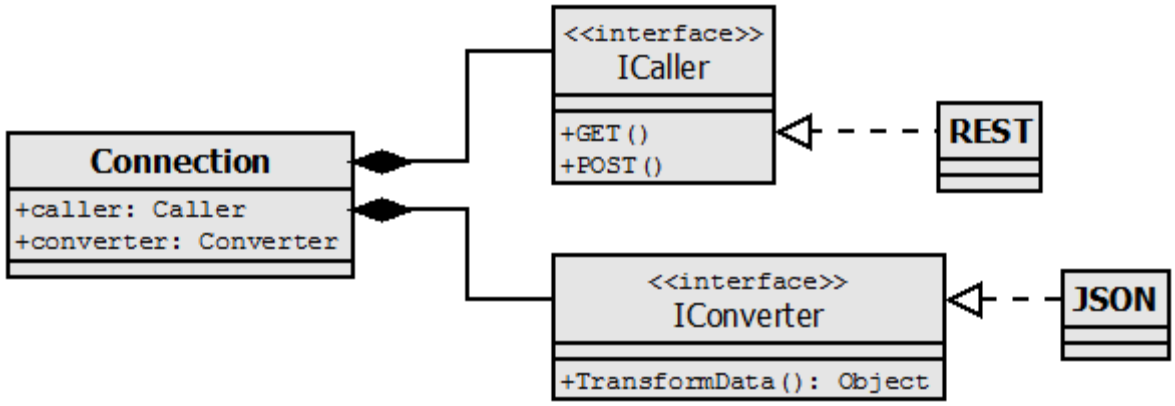
2.3.3.1 Globaal ontwerp



Figuur 28, Strategy (globaal) [42]

In Figuur 28 is te zien waaruit een strategy pattern is opgebouwd. De Client roept de methode van de Strategy interface aan. In run-time wordt dan, op basis van de eigen context, de methode van ConcreteStrategyA of ConcreteStrategyB uitgevoerd.

2.3.3.2 Concreet ontwerp



Figuur 29, Strategy (concreet)

In eerste instantie lijkt Figuur 29 totaal niet op de globale variant. Dit komt door 2 redenen:

- Er zijn 2 Strategy-interfaces (ICaller en IConverter);
- Er is (steeds) meer 1 ConcreteStrategy voor de Strategy;

Het betreft hier dus 2 strategy patterns in één. De reden dat er maar steeds 1 ConcreteStrategy is, komt omdat er op dit moment niet meer algoritmes nodig zijn. Maar met de toekomst in achtneming, is er hier dus ruimte open gehouden voor nieuwe algoritmes.

De reden dat de strategy pattern toegepast is, is omdat de verschillende implementaties voor de SmA's (mogelijk in de toekomst) op verschillende manieren communiceren. Door dit principe toe te passen, wordt er dus een zeer generiek en makkelijk te wijzigen structuur voor de communicatie gecreëerd.

2.3.4 Façade[44]

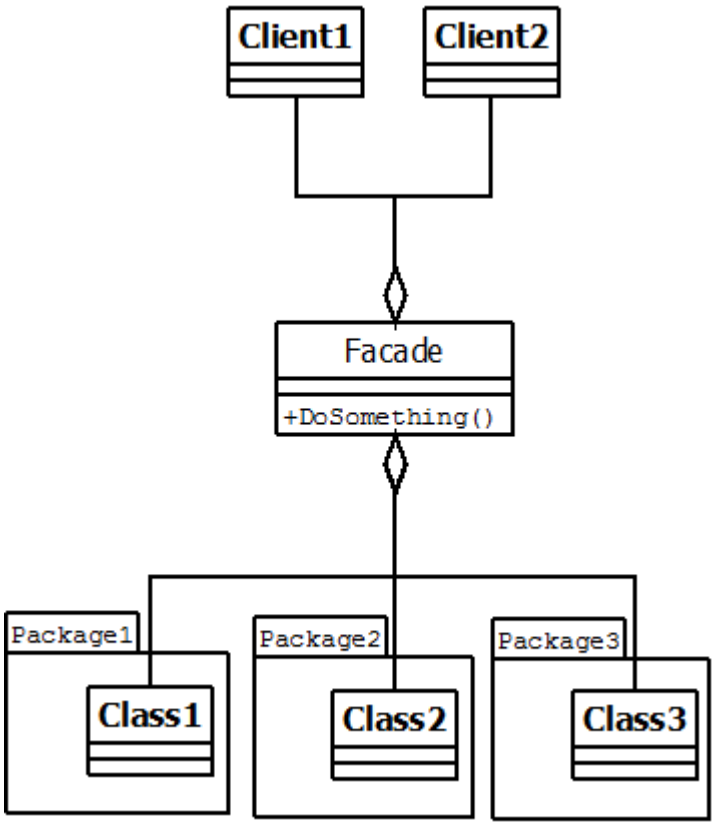
'Een façade is een object dat een versimpelde interface biedt voor een groter stuk code, zoals bijvoorbeeld een klasse library. Een façade kan:

- Een software library makkelijker in gebruik, leesbaarder, begrijpelijk en/of toetsbaar maken. Dit komt doordat het handige methodes voor veel voorkomende taken bevat;
- Externe afhankelijkheid verminderen op de interne werking op een library, aangezien de meeste code de façade gebruikt, waardoor er meer flexibiliteit ontstaat bij het ontwikkelen van het systeem;
- Een collectie slecht ontworpen API's met 1 goed-ontworpen API omhullen.

Een façade lijkt veel op de adapter (en decorator). Het verschil zit hem bij het gebruik:

- Adapter: wordt gebruikt indien de omhuller een specifieke interface moet aanhouden, en polymorfisme moet ondersteunen.
- Decorator: maakt het mogelijk om gedrag van de interface in run-time toe te voegen of aan te passen.
- Façade: is puur bedoeld om onderliggende implementatie te versimpelen voor de buitenkant. ' [44]

2.3.4.1 Globaal ontwerp

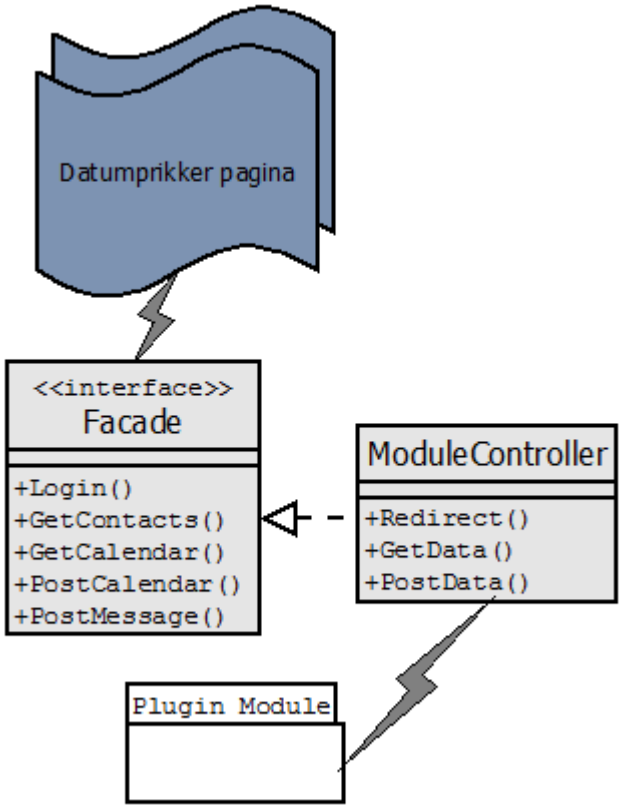


Figuur 30, Façade (globaal) [44]

[44] Bron: http://en.wikipedia.org/wiki/Facade_pattern

Zoals te zien in **Figuur 30**, gaat alle communicatie naar de **Package**s (en de onderliggende klassen) doormiddel van de **Façade** klasse. Hierdoor kan de logica binnen de **Package**s volledig veranderen, maar de **Client** klassen merken daar dan niets van.

2.3.4.2 Concreet ontwerp



Figuur 31, Façade (concreet)

Het ontwerp bij **Figuur 31** is iets globaler wat betreft de associaties. Dit komt doordat de **Datumprikker pagina**'s geen C# klassen zijn. De communicatie gaat hierbij dus niet direct, maar via protocollen.

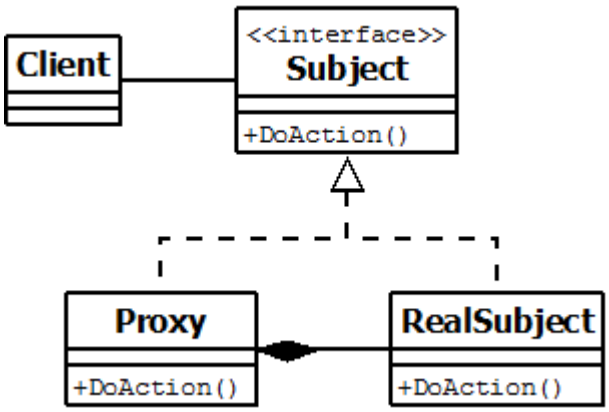
De reden dat de package **Plugin Module** bij dit ontwerp staat, is omdat **ModuleController**(Façade) verantwoordelijk is voor de gehele plug-in module. Vandaar ook dat hier de associaties zijn vervangen door het bliksem-icoon.

Om verwarring te voorkomen: de interface **Façade** wordt gebruikt door de **Datumprikker pagina**'s, maar de **ModuleController** klasse is de daadwerkelijke ('fysieke') façade. Vandaar ook dat **ModuleController** de **Façade** interface implementeert.

2.3.5 Proxy[45]

'Een proxy is, in zijn standaard vorm, een klasse die fungeert als een interface voor iets anders. Dit kan in principe van alles zijn: netwerk connectie, een groot object in het geheugen, een bestand, of een bron dat onmogelijk of te zwaar (qua kosten) is om te dupliceren.' [46]

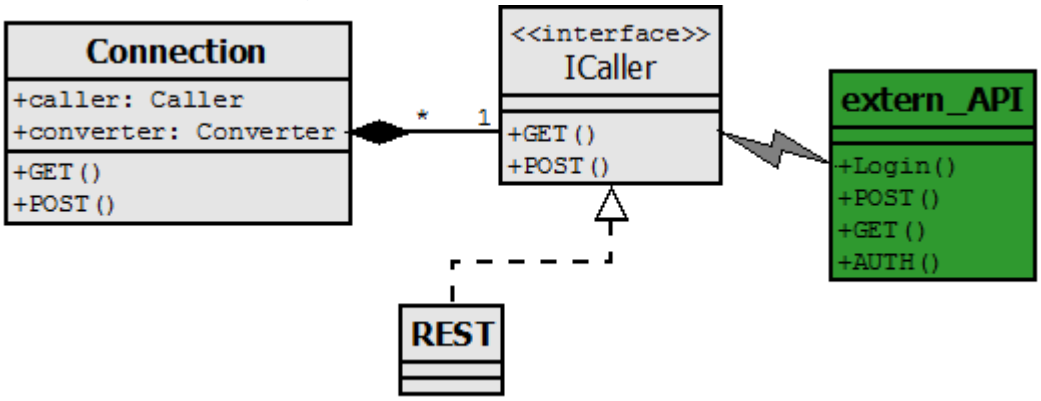
2.3.5.1 Globaal ontwerp



Figuur 32, Proxy (globaal) [45]

In **Figuur 32** is een globale proxy te zien. Dit ontwerp lijkt echter wel gericht op interne proxy's, aangezien een externe bron nooit de **Subject** interface kan implementeren.

2.3.5.2 Concreet ontwerp



Figuur 33, Proxy (concreet)

In **Figuur 33** is goed te zien dat de globale variant van **Figuur 32** niet goed werkt met externe bronnen. De interface **ICaller** wordt namelijk helemaal niet geïmplementeerd door de **extern_API**'s. Dit kan ook helemaal niet, want deze API's worden beheerd door de bedrijven van de SmA's.

Hoewel dit (en het hoofd-)ontwerp insinueert dat de interface **ICaller** met de **extern_API**'s communiceert, wordt dit uiteraard gedaan door de implementaties van deze interface. In dit geval communiceert alleen **REST** dus direct met **extern_API**.

[45] Bron: http://www.lepus.org.uk/ref/companion/Proxy.xml
[46] Bron: http://en.wikipedia.org/wiki/Proxy_pattern

2.3.6 Singleton[47]

'Een singleton pattern beperkt de instantiëring van een klasse tot één object. Dit is handig als er exact 1 object nodig is om acties te coördineren door het gehele systeem.' [48]

In tegenstelling tot de andere patterns, is het hier niet interessant om de ontwerpen te laten zien. Het betreft tenslotte 1 klasse, met wat variabelen.

Het is zeer makkelijk om een singleton aan te maken met C#. Hiervoor hoeft de klasse namelijk alleen maar statisch gemaakt te worden. Hierdoor kan elke klasse (indien het toegangsniveau[49] dit toe laat) gebruik maken van de (functionaliteit van de) klasse, zonder dat er een instantie van gemaakt kan worden.

De reden dat er gekozen is voor een singleton pattern, is omdat er een controle-klasse (in dit geval de **ModuleController**) nodig is voor de plug-in module. Vanuit deze klasse kan dan alles geregeld worden. Meerdere instanties van de **ModuleController** en/of de plug-in module zijn onnodig, aangezien het puur een middel is om data te ontvangen/verzenden is.

[47] Bron: http://csharpindepth.com/Articles/General/Singleton.aspx
[48] Bron: http://en.wikipedia.org/wiki/Singleton_pattern
[49] http://msdn.microsoft.com/en-us/library/ba0a1yw2.aspx

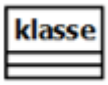
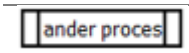
3 Flow-chart

In dit hoofdstuk wordt de flow-chart besproken die bij het ontwerp hoort. Omdat de totale flow-chart te groot is om in zijn geheel neer te zetten, is deze opgesplitst in processen.

De volgende processen worden besproken:

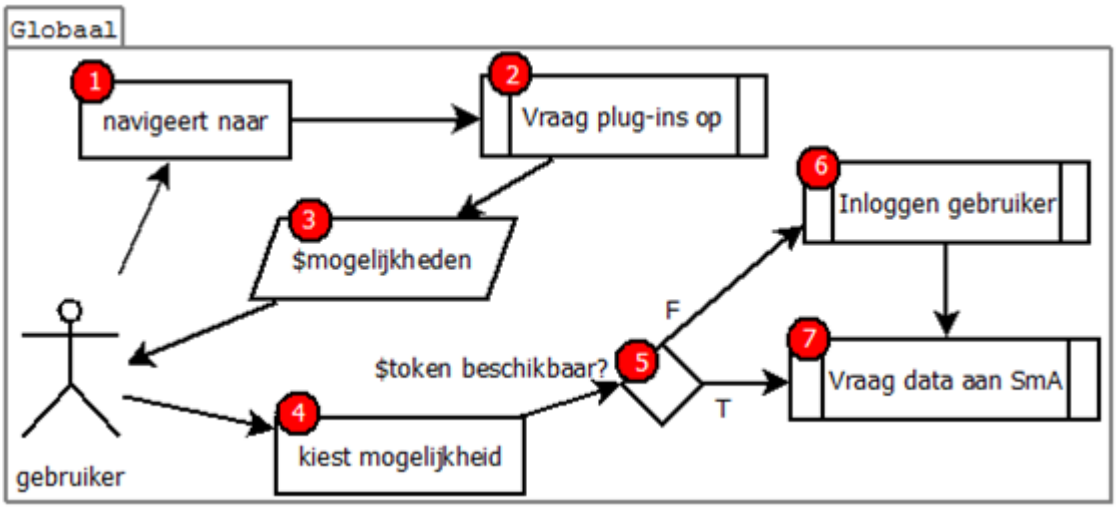
- **Globaal**
- **Vraag plug-ins op**
- **Inloggen gebruiker**
- **Vraag data aan SmA**
- **Authentiseer**
- **GET**
- **POST**

Elk proces bevat een aantal onderdelen:

Icoon	Betekenis
	Het gehele proces. Alles wat binnen dit icoon staat, hoort bij het proces.
	Het getal binnen dit icoon geeft aan welk nummer actie het betreft. Het icoon staat altijd in een hoek van de bijbehorende actie.
	Dit is de gebruiker van de applicatie. Bij dit icoon wordt er vanuit de applicatie verwacht dat de gebruiker een actie uitvoert.
	Dit icoon geeft de actie weer. De tekst binnen het icoon omschrijft de actie zelf.
	Dit icoon geeft het pad weer. Een pad koppelt (alle typen) acties, gebruikers en/of variabelen aan elkaar.
	Dit icoon geeft een controleactie weer. Een controleactie heeft altijd een controle tekst, en 2 tot 3 paden. Eén van deze paden gaat naar de controleactie toe. De andere paden worden gevolgd op basis van de uitkomst van de controleactie (true/false). De letter (T/F) die bij deze paden staat, geeft aan welke gevolgd moet worden.
	Dit icoon geeft een klasse weer. De klassen komen uit het ontwerp, en geven zodoende aan hoe het ontwerp gebruikt zal worden. Niet alle klassen van het ontwerp worden gebruikt, alleen degene die direct te maken hebben met de flow-chart.
	Dit icoon geeft een (set van) variabele(n) weer. De variabele(n) staan als tekst binnen het icoon. Indien het om een naam van de variabele gaat, dan staat er een '$' voor.
	Dit icoon geeft aan dat er verder gegaan wordt naar een ander proces. De tekst binnen dit icoon geeft aan welk proces dit betreft.

3.1 Globaal

Hieronder staat het globale proces afgebeeld.



Figuur 34, Globaal proces

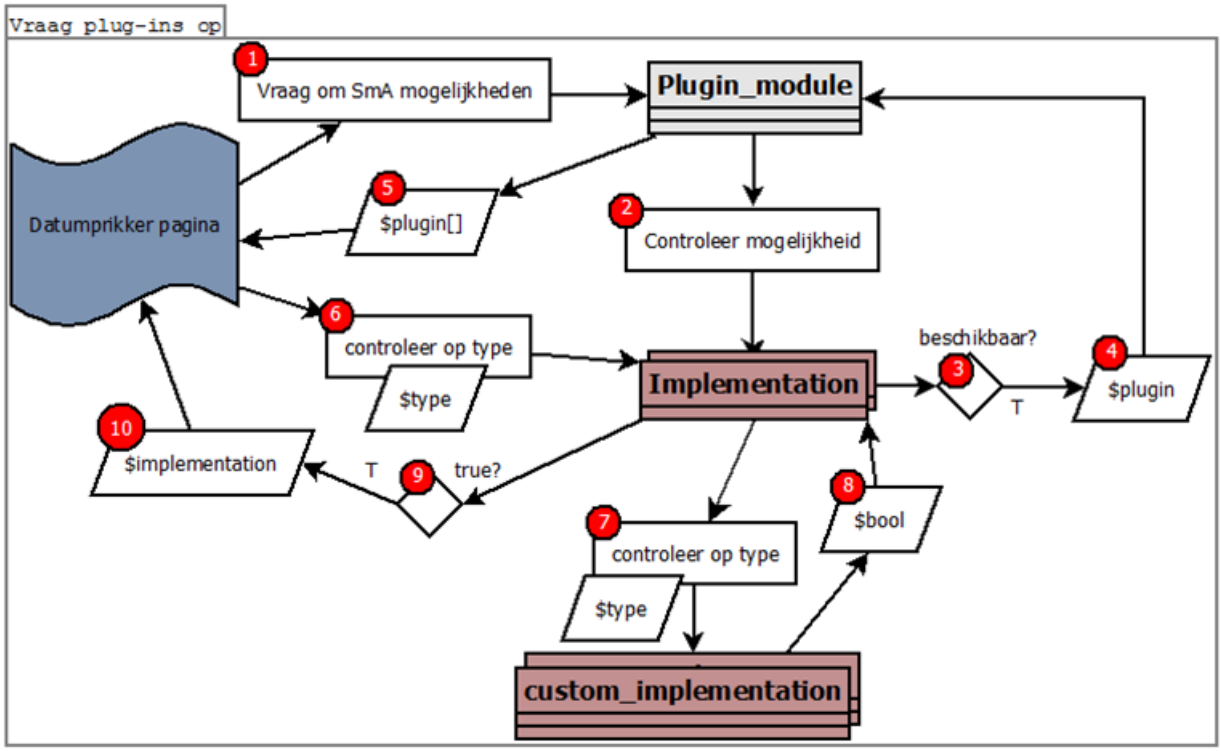
Zoals te zien in **Figuur 34**, zijn er 7 genummerde items. Elk van deze items is een actie. Sommige van deze acties zijn referenties naar andere processen, en nr.5 is een controleactie.

De acties houden als volgt in:

1. De **gebruiker** gaat naar een Datumprikker pagina met social media mogelijkheden (het volgende proces).
2. Het proces "**Vraag plug-ins op**" wordt uitgevoerd.
3. De **gebruiker** krijgt de mogelijke social media's te zien.
4. De **gebruiker** kiest één van de mogelijkheden.
5. Indien er een token beschikbaar is die hoort bij de gebruiker en de social media, dan wordt pad 'T' gevolgd. Indien er geen token is, dan wordt pad 'F' uitgevoerd.
6. Het proces "**Inloggen gebruiker**" wordt uitgevoerd.
7. Het proces "**Vraag data aan SmA**" wordt uitgevoerd.

3.2 Vraag plug-ins op

Hieronder staat het proces “Vraag plug-ins op” afgebeeld.



Figuur 35, Vraag plug-ins op proces

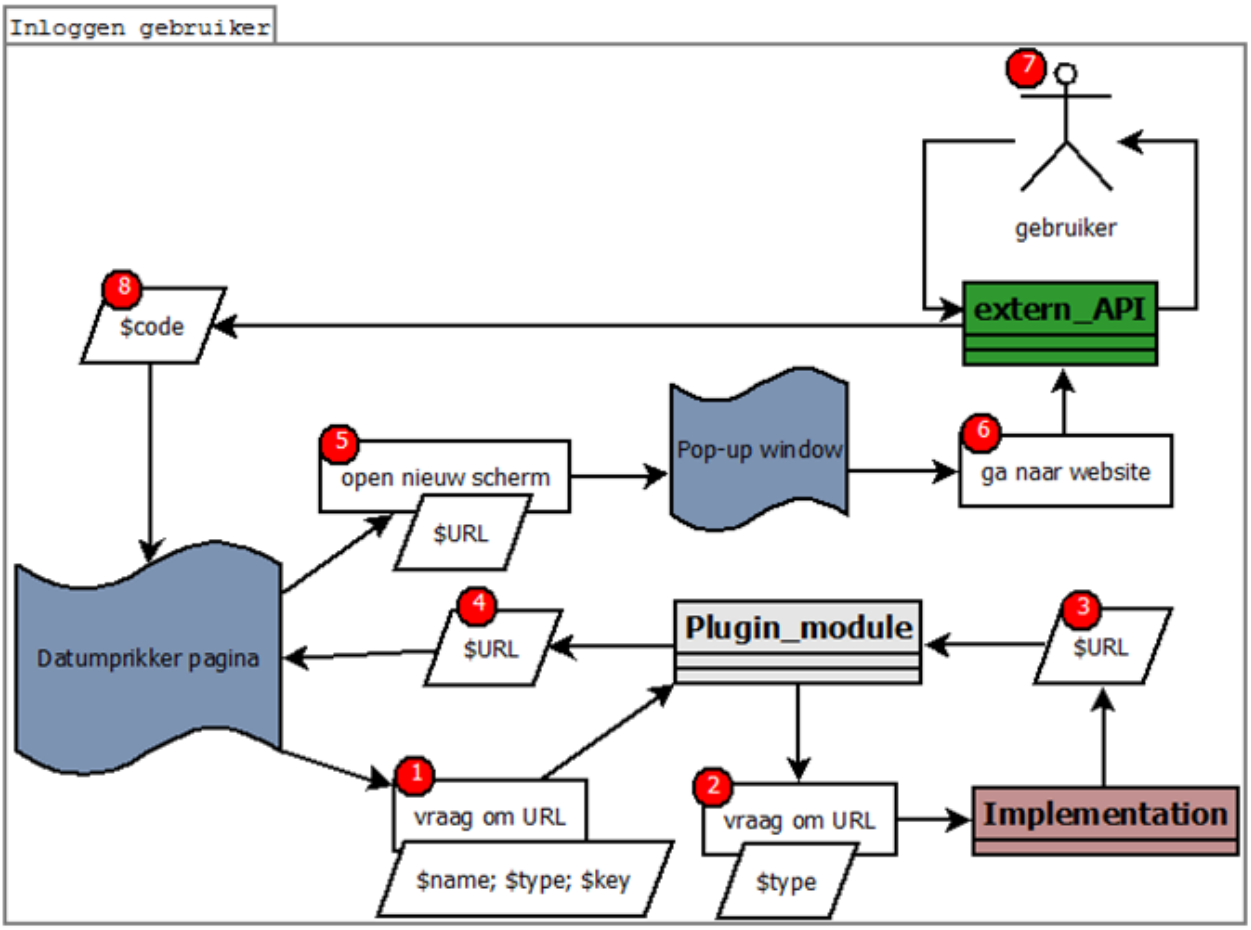
Zoals te zien in Figuur 35, zijn er 10 genummerde items. 7 Van deze items is een actie. Sommige van deze acties zijn controleacties. De andere genummerde items zijn retourneerwaardes.

De items houden als volgt in:

1. De Datumprikker pagina vraagt aan de Plugin_module om de plug-ins die gebruikt kunnen worden.
2. De Plugin_module gaat alle Implementation klassen af.
3. Elke Implementation klasse controleert of deze beschikbaar is. Indien dit het geval is, dan wordt pad ‘T’ gevolgd.
4. De Implementation klasse wordt geretourneerd.
5. Alle geretourneerde Implementation klassen worden aan de Datumprikker pagina gegeven.
6. Elke Implementation klasse wordt gevraagd om te controleren of deze $type ondersteund.
7. Elke Implementation klasse vraagt aan alle bijbehorende custom_implementation klassen of deze van type $type is.
8. Er wordt $bool geretourneerd (‘true’ als de klasse van type $type is).
9. Indien één van de geretourneerde $bool‘s ‘true’ is, dan wordt pad ‘T’ gevolgd.
10. De Implementation klasse wordt geretourneerd.

3.3 Inloggen gebruiker

Hieronder staat het proces “Inloggen gebruiker” afgebeeld.



Figuur 36, Inloggen gebruiker proces

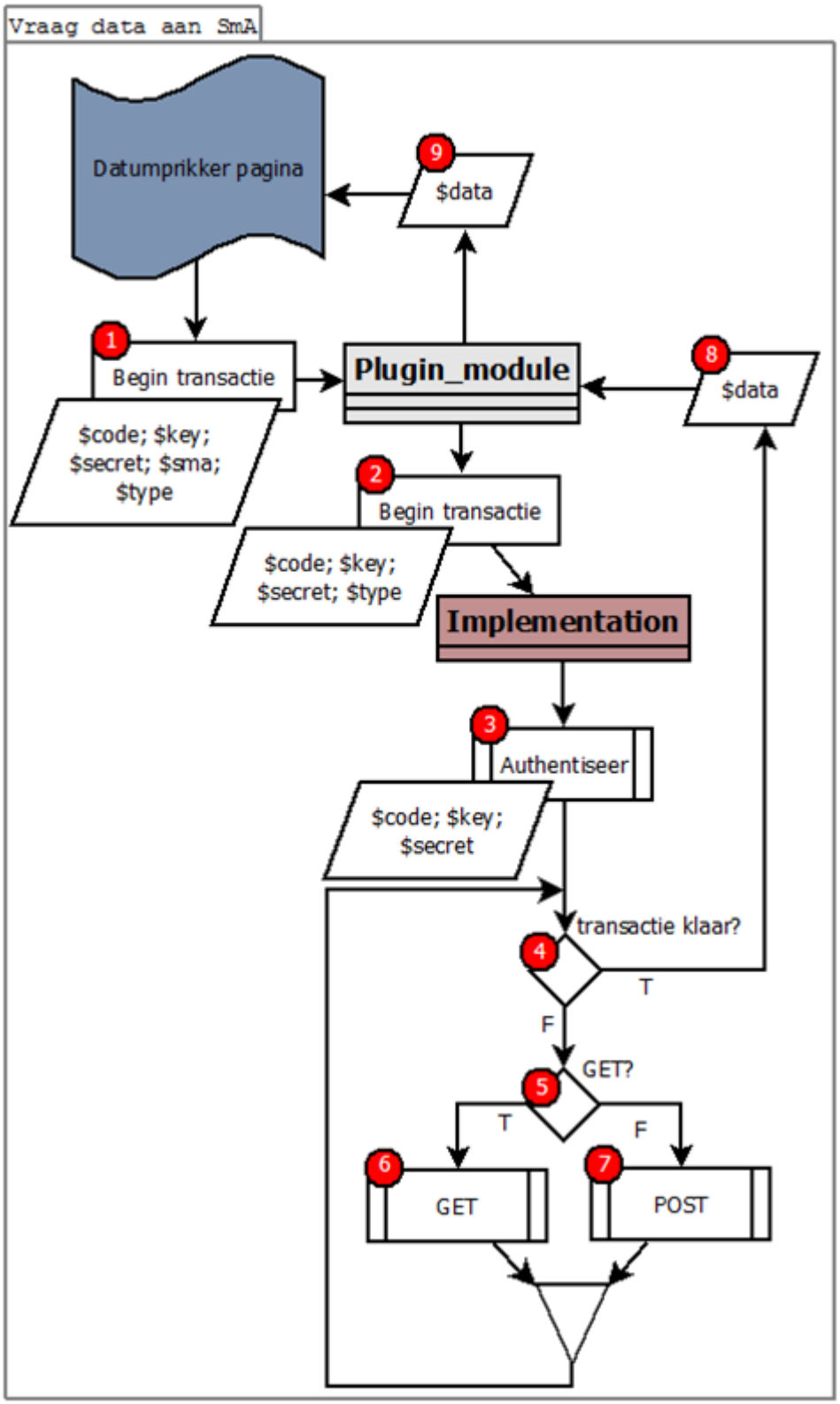
Zoals te zien in Figuur 36, zijn er 8 genummerde items. 5 Van deze items is een actie. De andere genummerde items zijn retourneerwaardes.

De items houden als volgt in:

1. De Datumprikker pagina vraagt aan de Plugin_module klasse om de URL voor het inloggen. Hierbij worden de variabelen $name, $type en $key meegegeven.
2. Op basis van $name, vraagt de Plugin_module klasse aan de bijbehorende Implementation klasse om de URL. Hierbij wordt de variabele $type meegegeven.
3. Op basis van onder andere de $type variabele, wordt er een URL($URL) geretourneerd.
4. De $URL wordt geretourneerd aan de Datumprikker pagina.
5. De Datumprikker pagina opent een nieuw scherm ($Pop-up window).
6. Het $Pop-up window gaat naar de $URL.
7. De $extern_API handelt tezamen met de $gebruiker het inloggen (en toestemming geven) af.
8. De $extern_API retourneert een $code waarmee geauthentiseerd kan worden.

3.4 Vraag data aan SmA

Hieronder staat het proces “Vraag data aan SmA” afgebeeld.



Figuur 37, Vraag data aan SmA proces

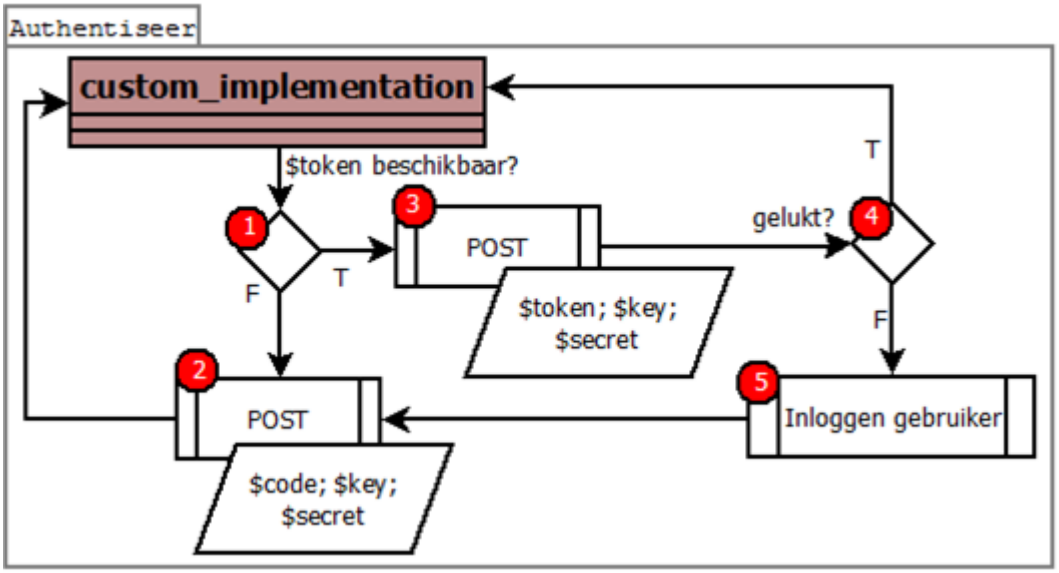
Zoals te zien in **Figuur 37**, zijn er 9 genummerde items. 7 Van deze items is een actie. Sommige van deze acties zijn controleacties. De andere genummerde items zijn retourneerwaardes.

De items houden als volgt in:

1. De **Datumprikker pagina** vraagt aan de **Plugin_module** klasse om een transactie te starten. Hierbij worden de volgende variabelen meegegeven: **$code**, **$key**, **$secret**, **$sma** en **$type**.
2. De **Plugin_module** klasse vraagt aan de **Implementation** klasse (gekoppeld aan de **$sma**) om een transactie te beginnen. Hierbij worden de volgende variabelen meegegeven: **$code**, **$key**, **$secret** en **$type**.
3. De **Implementation** klasse start het proces "**Authentiseer**", met de volgende variabelen: **$code**, **$key** en **$secret**.
4. Indien de transactie klaar is, dan wordt pad 'T' gevolgd. Anders wordt pad 'F' gevolgd.
5. Indien de volgende sub-transactie een 'GET' is, dan wordt pad 'T' gevolgd. Anders wordt pad 'F' gevolgd.
6. Het proces "**GET**" wordt uitgevoerd.
7. Het proces "POST" wordt uitgevoerd.
8. De **$data** wordt geretourneerd naar de **Plugin_module** klasse.
9. De **$data** wordt geretourneerd naar de **Datumprikker pagina**.

3.5 Authentiseer

Hieronder staat het proces "Authentiseer" afgebeeld.



Figuur 38, Authentiseer proces

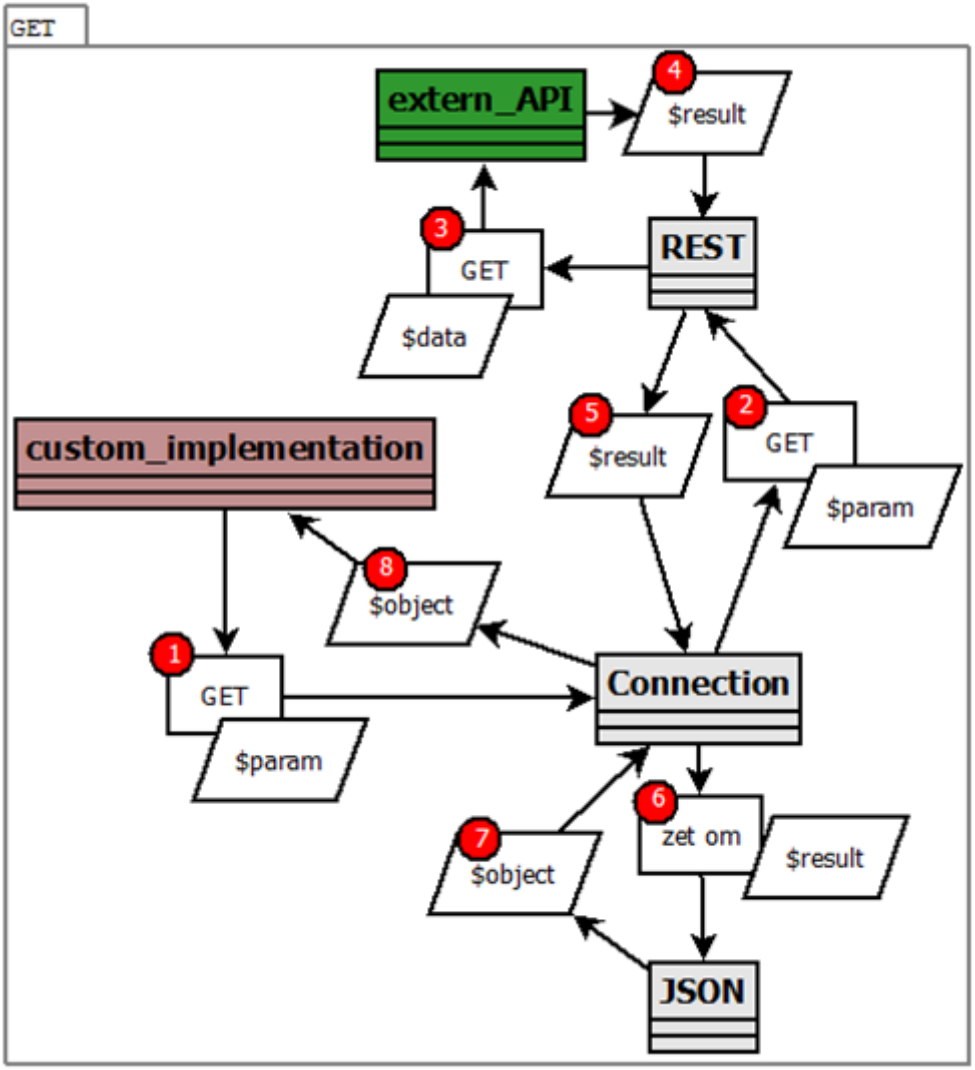
Zoals te zien in **Figuur 38**, zijn er 5 genummerde items. Elk van deze items is een actie. Sommige van deze acties zijn referenties naar andere processen, en nr.1 en 4 zijn controleacties.

De acties houden als volgt in:

1. Indien er een $token beschikbaar is, dan wordt pad ‘T’ gevolgd. Anders wordt pad ‘F’ gevolgd.
2. Het proces “POST” wordt uitgevoerd, met de volgende variabelen **$code**, **$key** en **$secret**.
3. Het proces “POST” wordt uitgevoerd, met de volgende variabelen **$token**, **$key** en **$secret**.
4. Indien het authentiseren gelukt is, dan wordt pad ‘T’ gevolgd. Anders wordt pad ‘F’ gevolgd.
5. Het proces “**Inloggen gebruiker**” wordt uitgevoerd.

3.6 GET

Hieronder staat het proces “GET” afgebeeld.



Figuur 39, GET proces

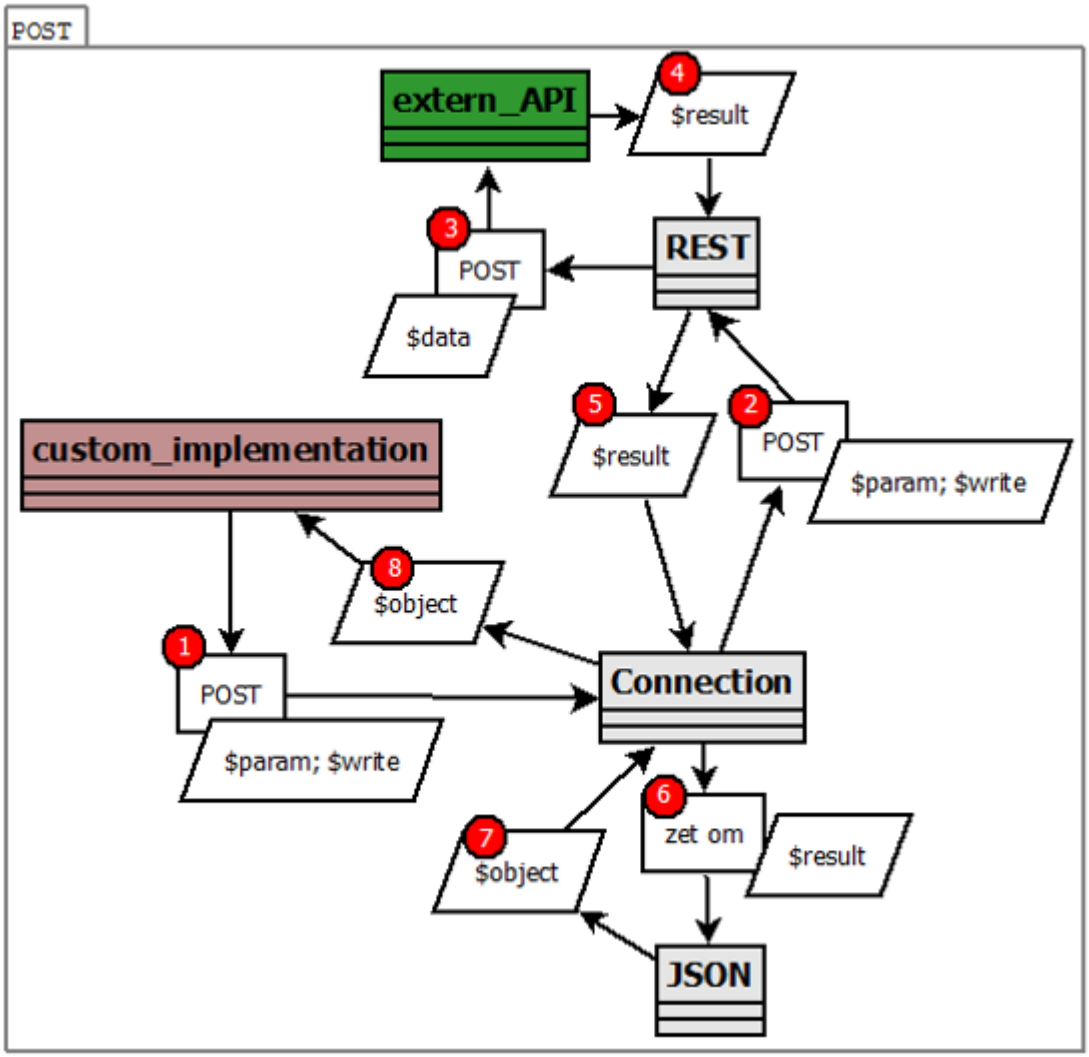
Zoals te zien in **Figuur 39,** GET proces, zijn er 8 genummerde items. 4 Van deze items is een actie. De andere genummerde items zijn retourneerwaardes.

De items houden als volgt in:

1. De **custom_implementatie** klasse vraagt aan de **Connection** klasse om een GET request uit te voeren. Hierbij worden de parameters **$param** meegegeven.
2. De **Connection** klasse vraagt aan de **REST** klasse om een GET request uit te voeren. Hierbij worden de parameters **$param** meegegeven.
3. De **REST** klasse voert een GET request uit bij de **extern_API**.
4. De **extern_API** retourneert **$result** terug.
5. De **REST** klasse geeft de **Connection** klasse **$result** terug.
6. De **Connection** klasse vraagt aan de **JSON** klasse om **$result** om te zetten. Hierbij wordt **$result** meegegeven.
7. De **JSON** klasse retourneert een **$object** naar de **Connection** klasse.
8. De **Connection** klasse retourneert een **$object** naar de **custom_implementatie** klasse.

3.7 POST

Hieronder staat het proces “POST” afgebeeld.



Figuur 40, POST proces

Zoals te zien in **Figuur 40**, zijn er 8 genummerde items. 4 Van deze items is een actie. Sommige van deze acties zijn controleacties. De andere genummerde items zijn retourneerwaardes.

De items houden als volgt in:

1. De **custom_implementatie** klasse vraagt aan de **Connection** klasse om een POST request uit te voeren. Hierbij worden de variabelen **$param** en **$write** meegegeven.
2. De **Connection** klasse vraagt aan de **REST** klasse om een POST request uit te voeren. Hierbij worden de variabelen **$param** en **$write** meegegeven.
3. De **REST** klasse voert een POST request uit bij de **extern_API**. Hierbij wordt er **$data** gebruikt.
4. De **extern_API** retourneert **$result** terug.
5. De **REST** klasse geeft de **Connection** klasse **$result** terug.
6. De **Connection** klasse vraagt aan de **JSON** klasse om **$result** om te zetten. Hierbij wordt **$result** meegegeven.
7. De **JSON** klasse retourneert een **$object** naar de **Connection** klasse.
8. De **Connection** klasse retourneert een **$object** naar de **custom_implementatie** klasse.

4 Oude versies

In dit hoofdstuk worden alle versies van het ontwerp en van de flow-chart besproken. Per versie (per onderdeel) zullen de volgende dingen behandeld:

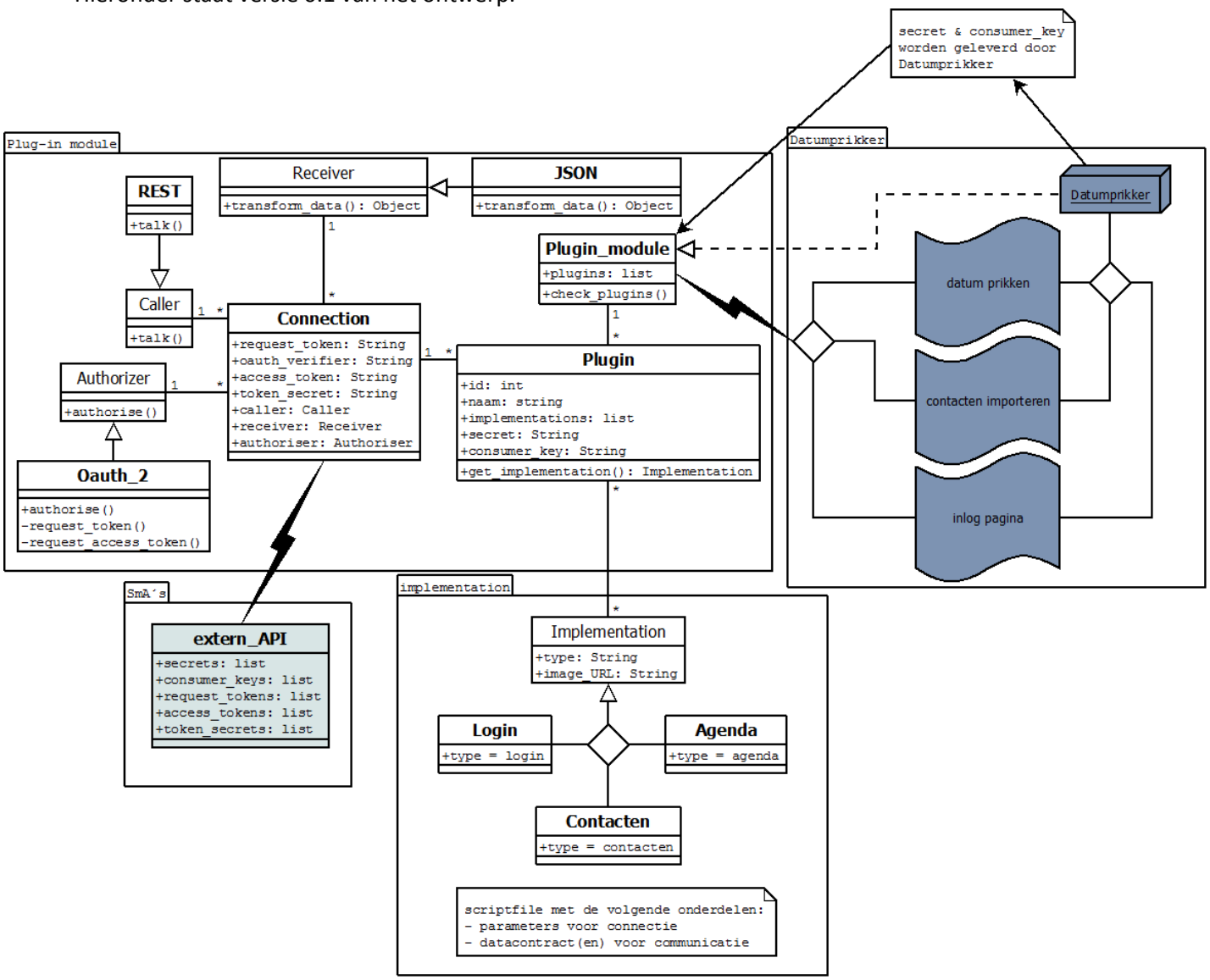
- Totaal-ontwerp; de afbeelding van het totale ontwerp.
    - Verschillen; hier worden de verschillen ten op zichten van het vorige ontwerp besproken, met daarbij de beargumentering. Bij de eerste versie zal het gehele ontwerp dus besproken worden.
    - Voordelen; hier worden de voordelen (ten opzichte van het vorige ontwerp) aangegeven.
    - Nadelen; hier worden de nadelen (ten opzichte van het vorige ontwerp) aangegeven.
- Flow-chart; de flow-chart die bij het ontwerp hoort.
    - Proces; een afbeelding van het (sub)proces.
    - Flow; uitleg over hoe de flow verloopt.
    - Voordelen; hier worden de voordelen (ten opzichte van het vorige ontwerp) aangegeven.
    - Nadelen; hier worden de nadelen (ten opzichte van het vorige ontwerp) aangegeven.

4.1 Ontwerp

4.1.1 Versie 0.1

4.1.1.1 Totaal ontwerp

Hieronder staat versie 0.1 van het ontwerp:



Figuur 41, ontwerp versie 0.1

Dit ontwerp laat goed zien dat er 4 scheidingen (packages) zijn binnen het project:

- **Plug-in module**; dit is de te bouwen module zelf. Dit zal dus alleen door de afstudeerder gemaakt worden.
- **Datumprikker**; dit is de reeds bestaande applicatie waar de module in geplaatst zal worden. De afstudeerder zal hier een paar aanpassingen moeten maken om te kunnen communiceren met de module.

- **SmA's**; dit zijn de API's van de social media die gebruikt zullen worden. De module zal hiermee moeten kunnen communiceren. De API's kunnen niet aangepast worden door de afstudeerder of Webbeat.
- **Implementatie**; deze package bevat alle SmA gerichte klassen, scripts en/of variabelen die nodig zijn om te kunnen communiceren met de SmA. Deze dingen zullen door zowel de afstudeerder als door Webbeat(na dit project) aangemaakt worden.

4.1.1.1.1 Verschillen

Aangezien er geen eerdere versies zijn, zal er hier kort uitgelegd worden hoe het ontwerp in elkaar steekt. Dit zal gedaan worden door de onderdelen in volgorde van gebruik te omschrijven.

1. [**Datumprikker**] 1 Van de pagina's wil data van een social medium ontvangen. Dit gebeurd door de klasse **Plugin_module** aan te spreken.
2. [**Plug-in module**] **Plugin_module** vraagt aan **Plugin** om te gaan communiceren met het social medium waar deze bij hoort. Dit wordt gebaseerd op de **Implementation**.
3. [**Implementatie**] Een **Implementation** geeft aan wat voor type data er gevraagd gaat worden (**Login**, **Contacten**, **Agenda**)
4. [**Plug-in module**] De **Plugin** vraagt aan **Connection** om de data op te vragen.
5. [**Plug-in module**] **Connection** communiceert met behulp van een **Caller** (bijvoorbeeld **REST**) met de **extern_API**.
6. [**Plug-in module**] Hiervoor moet er geautoriseerd worden, dit gebeurd doormiddel van een **Authorizer** (bijvoorbeeld **Oauth_2**).
7. [**Plug-in module**] De data die ontvangen wordt van de **extern_API**, wordt omgezet door een **Receiver** (bijvoorbeeld **JSON**).

4.1.1.1.2 Voordelen

De belangrijkste voordelen bij dit ontwerp zijn:

- Er kunnen gemakkelijk nieuwe implementaties toegevoegd worden.
- Er kan op verschillende manieren gecommuniceerd worden met de social media, zonder dat de aanroep vanuit **Datumprikker** veranderd.
- De plug-in module werkt bijna volledig autonoom. Hierdoor kan de module ook bij andere projecten gebruikt worden.

4.1.1.1.3 Nadelen

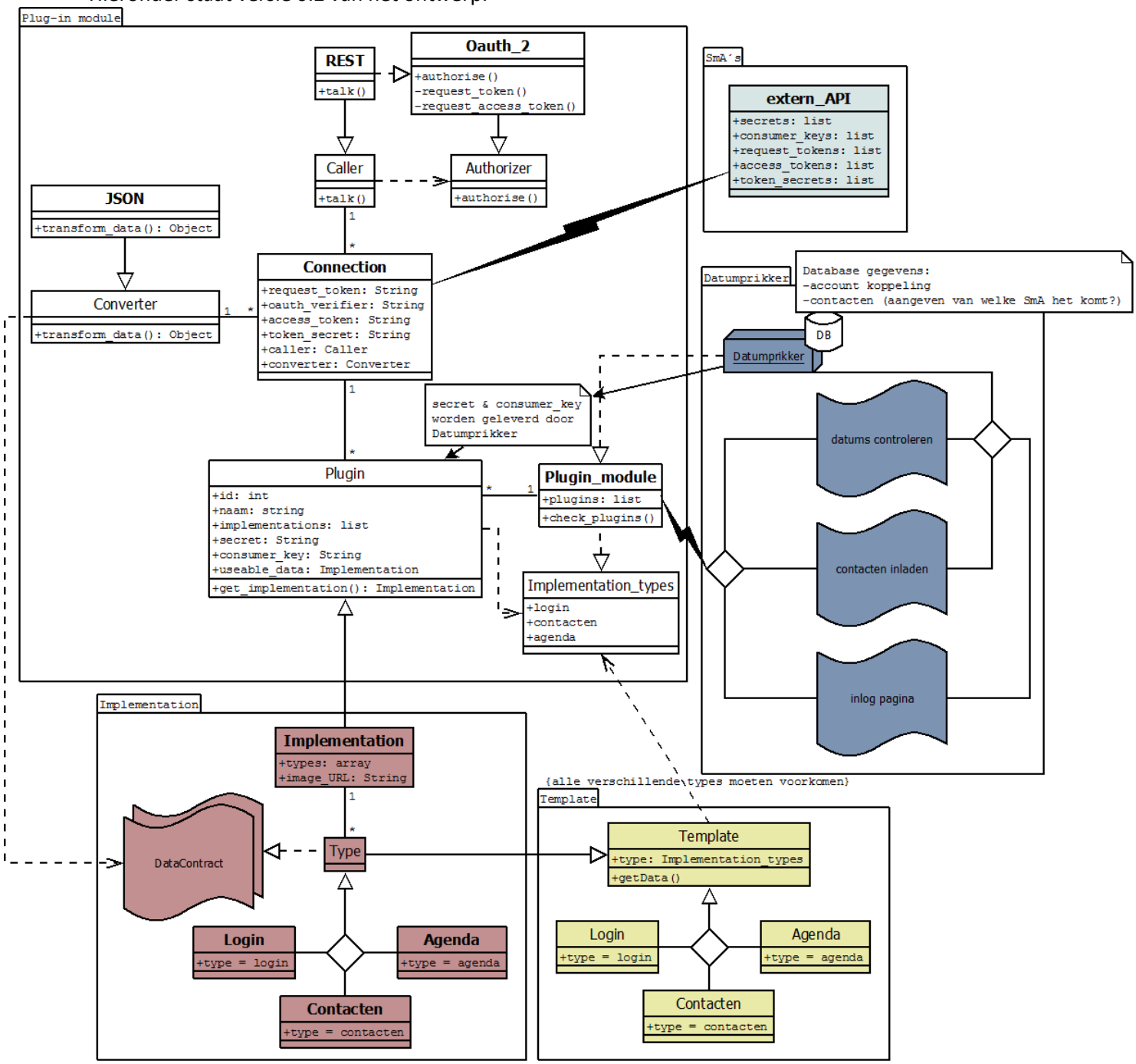
De belangrijkste nadelen bij dit ontwerp zijn:

- Het is niet duidelijk voor **Datumprikker** welke implementaties er beschikbaar zijn.
- Het is niet duidelijk naar wat voor type data de **Converter** de binnengekomen informatie moet omzetten.
- Het autoriseren ligt qua verantwoordelijkheid en functionaliteit zeer nauw verbonden tussen **Caller** en **Authorizer**. Het opsplitsen naar 2 klassen is daarom misschien een stap te ver.

4.1.2 Versie 0.2

4.1.2.1 Totaal ontwerp

Hieronder staat versie 0.2 van het ontwerp:



Figuur 42, ontwerp versie 0.2

4.1.2.1.1 Verschillen

De volgende onderdelen verschillen van de vorige versie:

- De klasse **Authorizer** dient nu door **Caller** gebruikt te worden. Dit betekend dat bijvoorbeeld **REST**, **Oauth_2** implementeert.
- De klasse **Implementation_types** is toegevoegd. Dit is een enumeratie die gebruikt wordt door **Plugin**, **Plugin_module**, **Template** en (indirect) door de **Datumprikker** pagina's.
- De **Template** package is toegevoegd. De implementaties zullen hierop gebaseerd zijn. Deze package bevat de volgende (tevens nieuwe) klassen:
    - **Template**; Deze klasse valt onder 1 van de enumeratiewaarden van **Implementation_types**. De **Datumprikker** pagina's verwachten deze klasse terug. Omdat deze klasse abstract is, dient hij overgeërfd te worden door andere klassen.
    - **Login**; Deze klasse heeft als enumeratiewaarden "login". Hij wordt dus gebruikt om login data op te vragen.
    - **Contacten**; Deze klasse heeft als enumeratiewaarden "contacten". Hij wordt dus gebruikt om contact data op te vragen.
    - **Agenda**; Deze klasse heeft als enumeratiewaarden "agenda". Hij wordt dus gebruikt om agenda data op te vragen.
- Het item **DataContract** is toegevoegd. In deze documenten staat aangegeven wat voor convertering de **Converter** klasse moet uitvoeren.
- De klasse **Type** is toegevoegd. Deze klasse is de superklasse die gebruikt wordt door **Implementation**. De volgende (tevens nieuwe) klassen erven hiervan over:
    - **Login**; Deze klasse is de implementatie van de gelijknamige **Template** variant.
    - **Contacten**; Deze klasse is de implementatie van de gelijknamige **Template** variant.
    - **Agenda**; Deze klasse is de implementatie van de gelijknamige **Template** variant.

4.1.2.1.2 Voordelen

De belangrijkste voordelen zijn:

- Er is onderscheid gemaakt tussen implementaties en templates. Hierdoor is er meer abstractie richting **Plugin** en **Plugin_module**.
- De klassen **Authorizer** en **Oauth_2** worden nu volledig geïntegreerd door **Caller** en de overervende klassen.

4.1.2.1.3 Nadelen

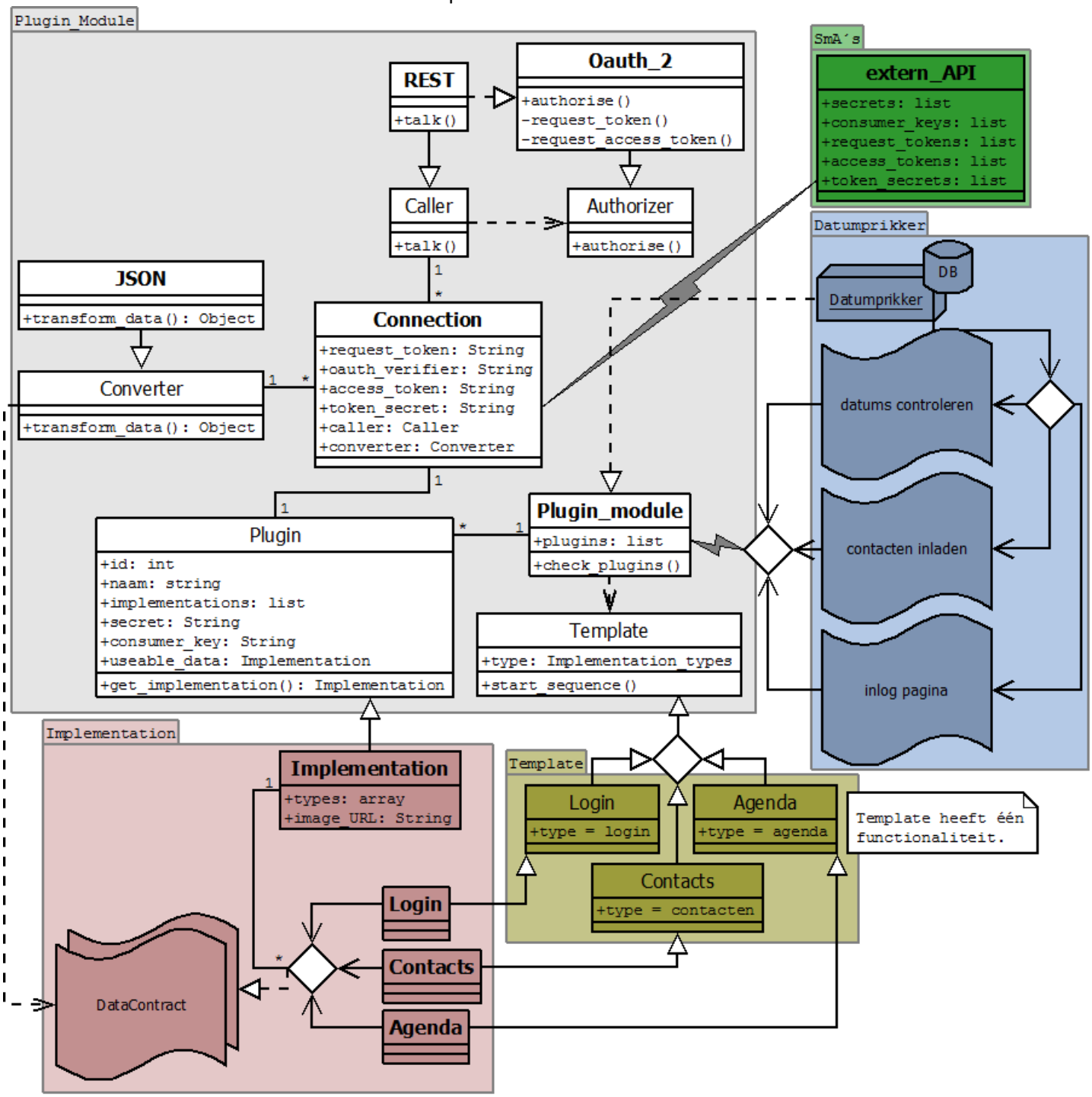
De belangrijkste nadelen zijn:

- De **Datacontract**en zijn niet direct verbonden aan de implementaties. Dit wordt waarschijnlijk moeilijk waar te maken in code.
- De overerving van de implementaties op de templates is niet duidelijk (genoeg).

4.1.3 Versie 0.3

4.1.3.1 Totaal ontwerp

Hieronder staat versie 0.3 van het ontwerp:



Figuur 43, ontwerp versie 0.3

4.1.3.1.1 Verschillen

De volgende onderdelen verschillen van de vorige versie:

- De packages hebben een duidelijkere scheiding door een toegevoegde achtergrondkleur.
- De klasse **Template** is verplaatst naar de **Plugin_Module** package.
- De klasse **Type** is weggehaald: deze had geen betekenis.
- De overerving van de implementaties op de templates is duidelijk(er) gemaakt.
- De implementaties hebben nu elk hun eigen **Datacontract**.

4.1.3.1.2 Voordelen

De belangrijkste voordelen zijn:

- Duidelijkere scheiding, zowel doormiddel van kleur als lijnen.
- Geen overbodige klassen meer.

4.1.3.1.3 Nadelen

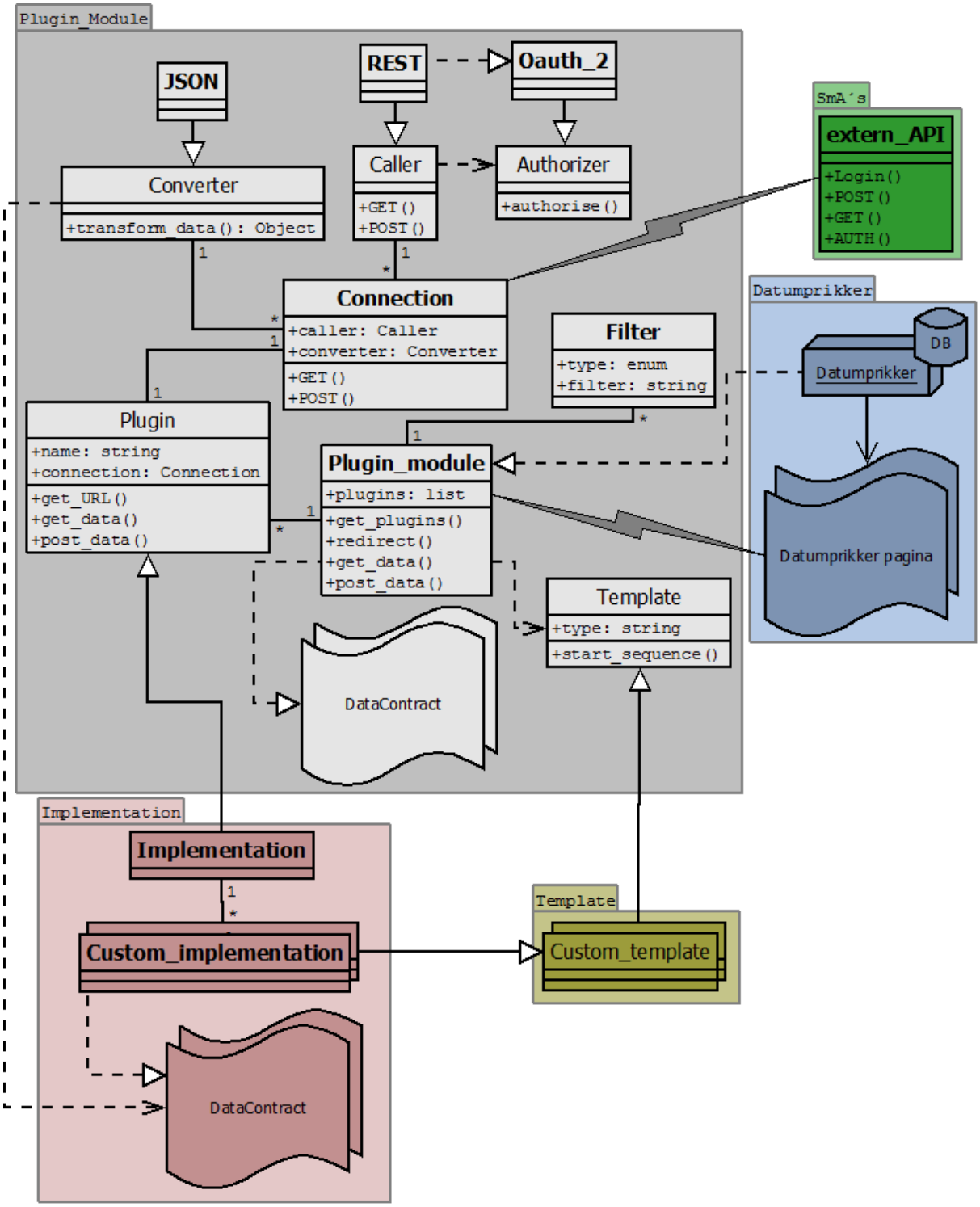
De belangrijkste nadelen zijn:

- Onoverzichtelijkheid doordat templates los staan, terwijl ze qua ontwerp 99% gelijk zijn.
- Onoverzichtelijkheid doordat implementaties los staan, terwijl ze qua ontwerp 99.9% gelijk zijn.
- De implementering van de **Datacontract**en van de implementaties is niet zeer duidelijk.
- Veel overbodige variabelen/methodes.

4.1.4 Versie 1.0

4.1.4.1 Totaal ontwerp

Hieronder staat versie 1.0 van het ontwerp:



Figuur 44, ontwerp versie 1.0

4.1.4.1.1 Verschillen

De volgende onderdelen verschillen van de vorige versie:

- Implementaties zijn samengevoegd tot 1 multi-object.
- Templates zijn samengevoegd tot 1 multi-object.
- Datumprikker pagina's zijn samengevoegd tot 1 multi-pagina.
- Klassen van **Plugin_Module** zijn nu ook gekleurd.
- De **Filter** klasse toegevoegd.
- Overbodige variabelen en methodes zijn verwijderd.

4.1.4.1.2 Voordelen

De belangrijkste voordelen zijn:

- Meer abstractie bij implementaties, templates en Datumprikker pagina's.
- Mogelijkheid tot filtering met nieuwe **Filter** klasse.

4.1.4.1.3 Nadelen

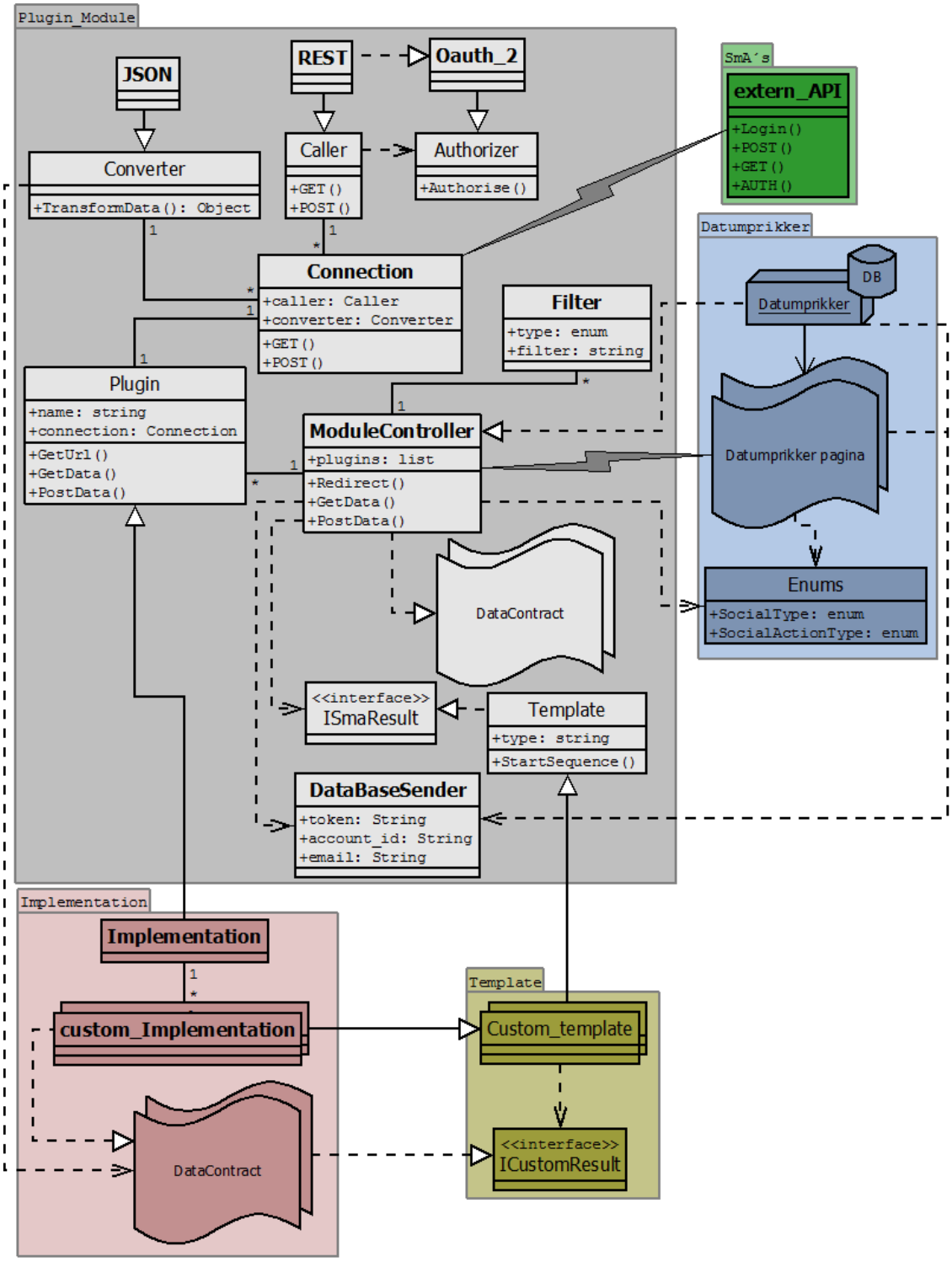
De belangrijkste nadelen zijn:

- Niet duidelijk wat **Plugin_module** moet ontvangen en retourneren naar de **Datumprikker pagina**'s.
- Niet duidelijk hoe de **Datumprikker pagina**'s de **Plugin_module** moeten aanspreken.
- Niet duidelijk hoe de scheiding tussen voor gebruiker zichtbare data en server data gegarandeerd wordt.

4.1.5 Versie 1.1

4.1.5.1 Totaal ontwerp

Hieronder staat het totale ontwerp van de module afgebeeld:



Figuur 45, ontwerp versie 1.1

4.1.5.1.1 Verschillen

De volgende onderdelen verschillen van de vorige versie:

- Interface **ISmaResult** toegevoegd.
- Klasse **DataBaseSender** toegevoegd.
- Klasse **Enums** toegevoegd.
- Interface **ICustomResult** toegevoegd.
- Klasse **DataContract** voor **ModuleController** toegevoegd.
- Namen van items aangepast om meer eenheid te creëren.

4.1.5.1.2 Voordelen

De belangrijkste voordelen zijn:

- Duidelijk hoe de **Datumprikker pagina**'s met ModuleController moeten praten (via **Enums** en **DataContract**).
- De klasse **ModuleController** ontvangt **ISmaResult** terug, en retourneert dit naar de **Datumprikker pagina**'s.
- De **DataContract**en van **custom_Implementation** implementeren nu allemaal dezelfde interface. Hierdoor is er meer abstractie richting de **ModuleController**.
- Er is een duidelijke scheiding tussen wat naar de database gestuurd wordt, en wat er naar de database gestuurd wordt.

4.1.5.1.3 Nadelen

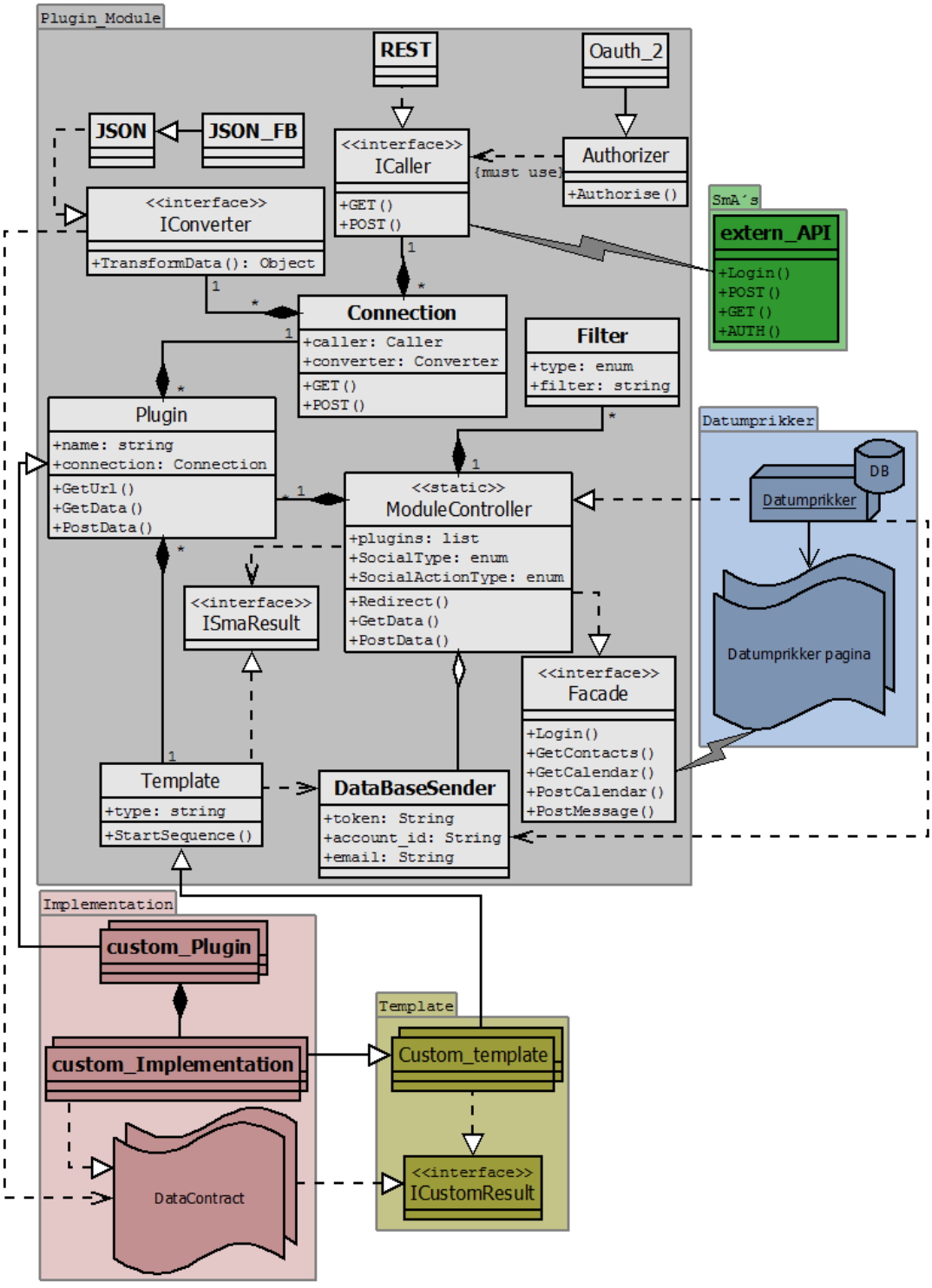
De belangrijkste nadelen zijn:

- De klasse **Authorizer** en **Oauth_2** zijn onduidelijk qua bedoeling.
- Geen koppeling tussen **Template** en **DataBaseSender**, terwijl die er wel zou moeten zijn.
- De klasse **Connection** communiceert in werkelijkheid niet direct met **extern_API**, alleen de **Caller** klasse doet dat.
- Sommige klassen zijn beter indien deze interfaces zijn, aangezien ze geen (concrete) inhoud hebben.
- DataBaseSender heeft geen associatie.
- Plugin heeft geen associatie met Template, terwijl dit wel zou moeten.

4.1.6 Versie 1.2

4.1.6.1 Totaal ontwerp

Hieronder staat het totale ontwerp van de module afgebeeld:



Figuur 46, ontwerp versie 1.2

4.1.6.1.1 Verschillen

De volgende onderdelen verschillen van de vorige versie:

- De items **IConverter** en **ICaller** zijn nu interfaces.
- De klasse **Template** maakt nu gebruik van **DataBaseSender**.
- De **Façade** interface toegevoegd om functionaliteiten van de plug-in module duidelijk(er) te maken.
- De klassen **Authorizer** en **Oauth_2** zijn leggen nu een restrictie op de **ICaller** interface.
- De klasse **Enums** is weggehaald, en diens inhoud staat nu bij **ModuleController**.
- De klasse **DataBaseSender** heeft nu een associatie met **ModuleController**.
- De klasse **Plugin** heeft nu een associatie met **Template**.

4.1.6.1.2 Voordelen

De belangrijkste voordelen zijn:

- De klasse **DataBaseSender** wordt nu ook daadwerkelijk (volgens het ontwerp) bijgehouden.
- Het is nu duidelijk wat de klassen **Authorizer** en **Oauth_2** nu betekenen binnen het ontwerp.
- Meer gebruik van interfaces.
- Meer duidelijkheid door verplaatsing van klassen etc.
- Het is proces van het ophalen/versturen van data is nu volledig uit te leggen aan de hand van het ontwerp (dus geen verborgen/ontbrekende items meer).

4.1.6.1.3 Nadelen

De belangrijkste nadelen zijn:

- De interface **Façade** begint de naam niet met ‘I’.
- De klasse **Authorizer** is geen interface, terwijl dat wel kan (aangezien er toch geen inhoud is).

5-6-2014

Versie 1.0

Test rapportage

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

Wijzigingen

- Document opmaak aangepast

Inhoudsopgave

1 Inleiding

Tijdens dit project zijn er op 2 momenten getest of de plug-in module werkt volgens de requirements. Dit document bevat alle tests en de uitslagen hiervan.

1.1 Tekstopmaak uitleg

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.
- **Variabele**; dit is een verwijzing naar een variabele die gebruikt wordt binnen een bepaalde context (bijvoorbeeld binnen een **afbeelding**).

2 Rapporten

2.1 Proof-of-concept

Het proof-of-concept is de uiteindelijke oplevering na sprint 3. Met dit proof-of-concept kunnen alle van te voren opgelegde functionaliteiten uitgevoerd worden aan de kant van de module. Dit houdt in dat de volgende dingen gedaan moeten kunnen worden:

- Inloggen; een gebruiker moet via een SmA kunnen inloggen. De data die geretourneerd wordt zal later gebruikt worden om de gebruiker te koppelen aan een Datumprikker account.
- Contacten ophalen; een gebruiker moet via een SmA contacten op kunnen halen. Later zullen deze contacten gebruikt worden binnen Datumprikker.
- Berichten versturen; een gebruiker met contacten van het hiervoor genoemde onderdeel een bericht kunnen sturen doormiddel van een SmA. Indien de contacten met een email adres benaderd kunnen worden, dan zal dit later met Datumprikker afgehandeld worden.
- Kalender controleren; een gebruiker moet via een SmA van te voren aangegeven datums automatisch laten controleren op beschikbaarheid. Later zal dit geïntegreerd worden met de pagina voor het aanmaken van een afspraak.

2.1.1 Testmethode

De testmethodiek die gebruikt zal worden om het proof-of-concept te testen is een (black-box) use-case acceptatie test.

2.1.1.1 Black-box

Bij black-box testen wordt de interne werking van de code genegeerd. Het gaat puur om de in- en uitvoer. Hierdoor maakt het zeer geschikt voor een acceptatie test, aangezien de tests ook door een externe partij (bijv. de opdrachtgever) uitgevoerd/gecontroleerd kunnen worden.

2.1.1.2 Use case stroomdiagram

Om de use cases te kunnen testen, zijn er stroomdiagrammen opgesteld. Deze stroomdiagrammen zijn gebaseerd op de use cases, met in achtneming van het proof-of-concept. Hiermee wordt bedoeld dat bepaalde onderdelen (knoppen, pagina's, etc.) weggelaten of anders zullen zijn.

2.1.1.3 Testsessie

Een testsessie is een aantal tests die bij elkaar voor 1 conclusie zorgen over 1 van de functionaliteiten. Elke testsessie is succesvol afgerond indien voor elke beschikbare SmA, alle paden van de bijbehorende stroomdiagrammen minimaal 1 keer gevolgd zijn.

2.1.1.3.1 Paden

Elk pad binnen een stroomdiagram heeft een id. Bij het start- en eind-pad zijn dit de letters 'S' en 'E'. Alle andere paden hebben een uniek (oplopend) nummer. Alle paden die gevolgd worden om van start naar eind te komen, vallen onder een (unieke) route.

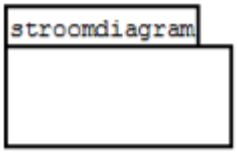
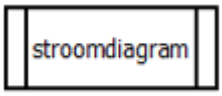
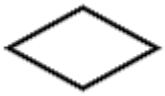
Sommige routes kunnen vrij lang worden doordat er veel stellingen voorbij komen die voor pad splitsingen (kunnen) zorgen. Daarom wordt er per stroomdiagram pad combinaties opgesteld.

2.1.1.3.2 Pad combinatie

Een pad combinatie is een sub-route die alle paden bevat tot een stelling (of het einde van het stroomdiagram). Dus alle paden tot aan de eerste stelling (dus afsplitsing) worden dan de eerste combinatie. En dan worden, per afsplitsing, alle daarop volgende paden tot de daaropvolgende stelling de volgende combinaties. Dit kan betekenen dat bepaalde combinaties terugkomen op de stelling waarmee de combinatie begon. Daarnaast is het mogelijk dat een pad combinatie maar uit 1 pad bestaat.

2.1.2 Voorbereiding: use case stroomdiagrammen

Elk stroomdiagram bevat de volgende onderdelen:

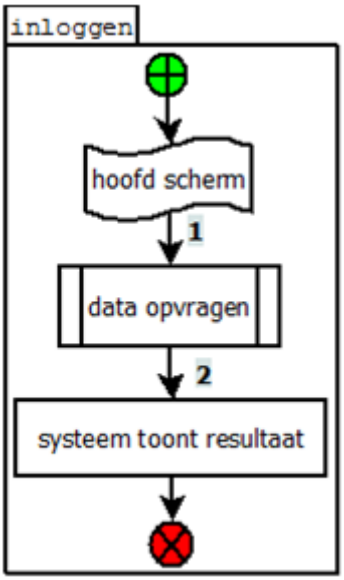
icoon	betekenis
	Alles wat binnen dit icoon staat, valt onder het diagram. De naam van het diagram staat link bovenin.
	Hier start het diagram.
	Hier eindigt het diagram.
	Dit is een pad dat gevolgd moet worden. De pijl geeft de richting aan.
	Dit icoon hoort bij een pad, en geeft aan welk nummer het is.
	Dit icoon stelt een bepaald scherm voor. Een scherm is een webbrowser pagina, al dan niet in een apart browser venster.
	Dit is een actie dat uitgevoerd wordt.
	Dit is een verwijzing naar een ander stroomdiagram. Zodra dat diagram klaar is, zal er verder gegaan worden met het huidige diagram.
	Hier scheiden paden zich op basis van een stelling. De stelling staat aan de kant van waaruit er naar dit icoon gewezen wordt. De uitkomst paden hebben een waarde bij zich staan die betrekking hebben op de stelling.

De volgende stroomdiagrammen zijn gemaakt op basis van de use cases:

- **Inloggen**
- **Contacten ophalen**
- **Kalender controleren**
- **Verstuur bericht**
- **Maak bericht**
- **Data opvragen**
- **SmA registratie**
- **SmA afhandeling**

2.1.2.1 Inloggen

Het uiteindelijke resultaat van dit stroomdiagram is dat er op het hoofdscherm de account gegevens staan van de gebruiker.



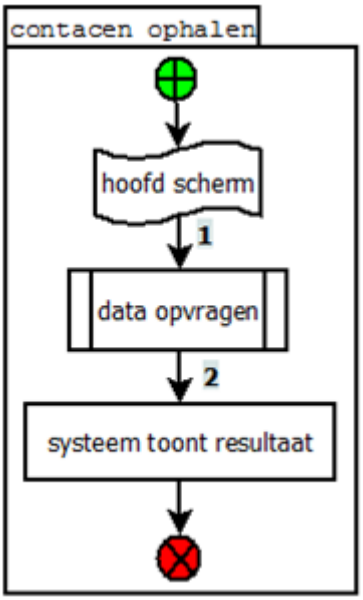
Figuur 47, inloggen stroomdiagram

De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2,E

2.1.2.2 Contacten ophalen

Het uiteindelijke resultaat van dit stroomdiagram is dat er op het hoofdscherm de contacten van de gebruiker (die bekend zijn bij de SmA) staan.



Figuur 48, contacten ophalen stroomdiagram

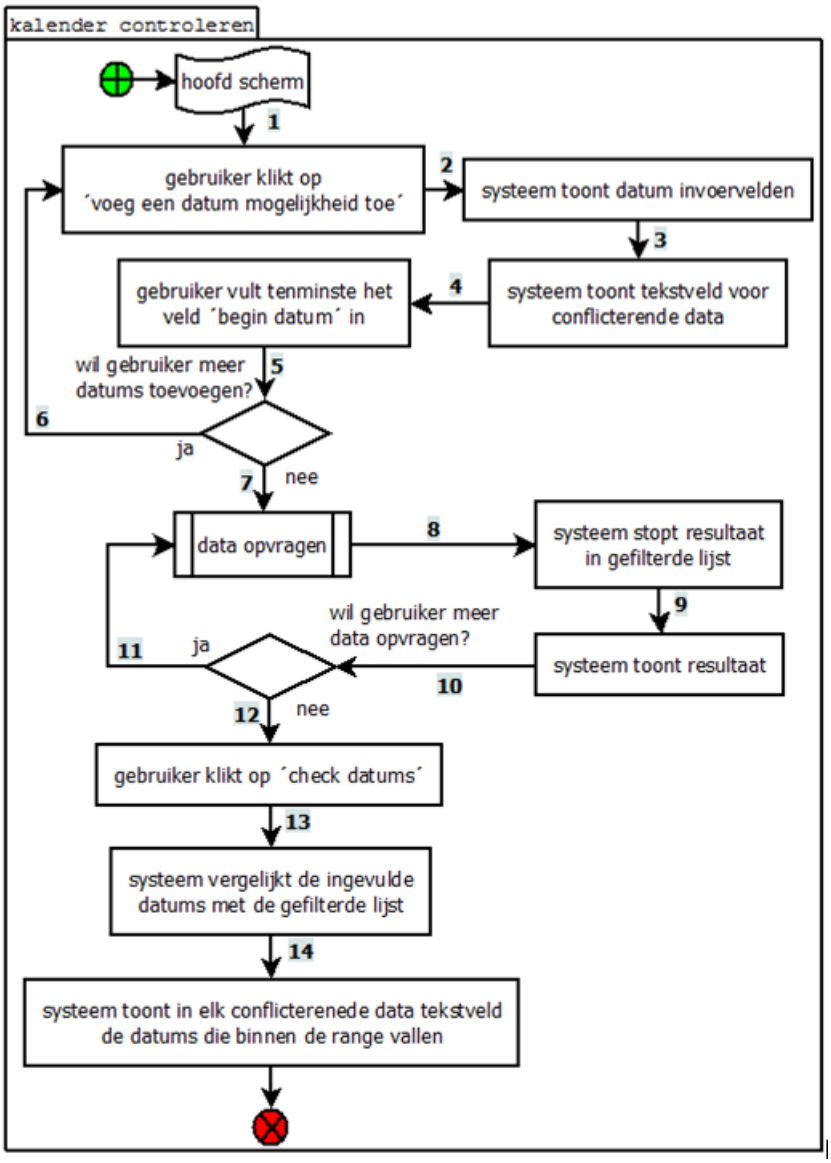
De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2,E

2.1.2.3 Kalender controleren

Het uiteindelijke resultaat van dit stroomdiagram is dat er op het hoofdscherm, per conflicterende data tekstveld, vermeld staat of er datums uit de agenda van de gebruiker binnen de range vallen van de (door de gebruiker) ingevulde datums.

Belangrijk om te op te melden bij dit stroomdiagram, is dat de actie “gebruiker vult ten minste het veld ‘begin datum’ in” dieper getest kan worden (wat gebeurd er bijvoorbeeld als een gebruiker onzin invult). Dit valt echter buiten de scope, omdat dit gedeelte door de Datumprikker pagina afgehandeld zal worden. Bij dit proof-of-concept wordt er dan ook vanuit gegaan dat alle ingevulde data correct ingevuld zal worden.



Figuur 49, kalender controleren stroomdiagram

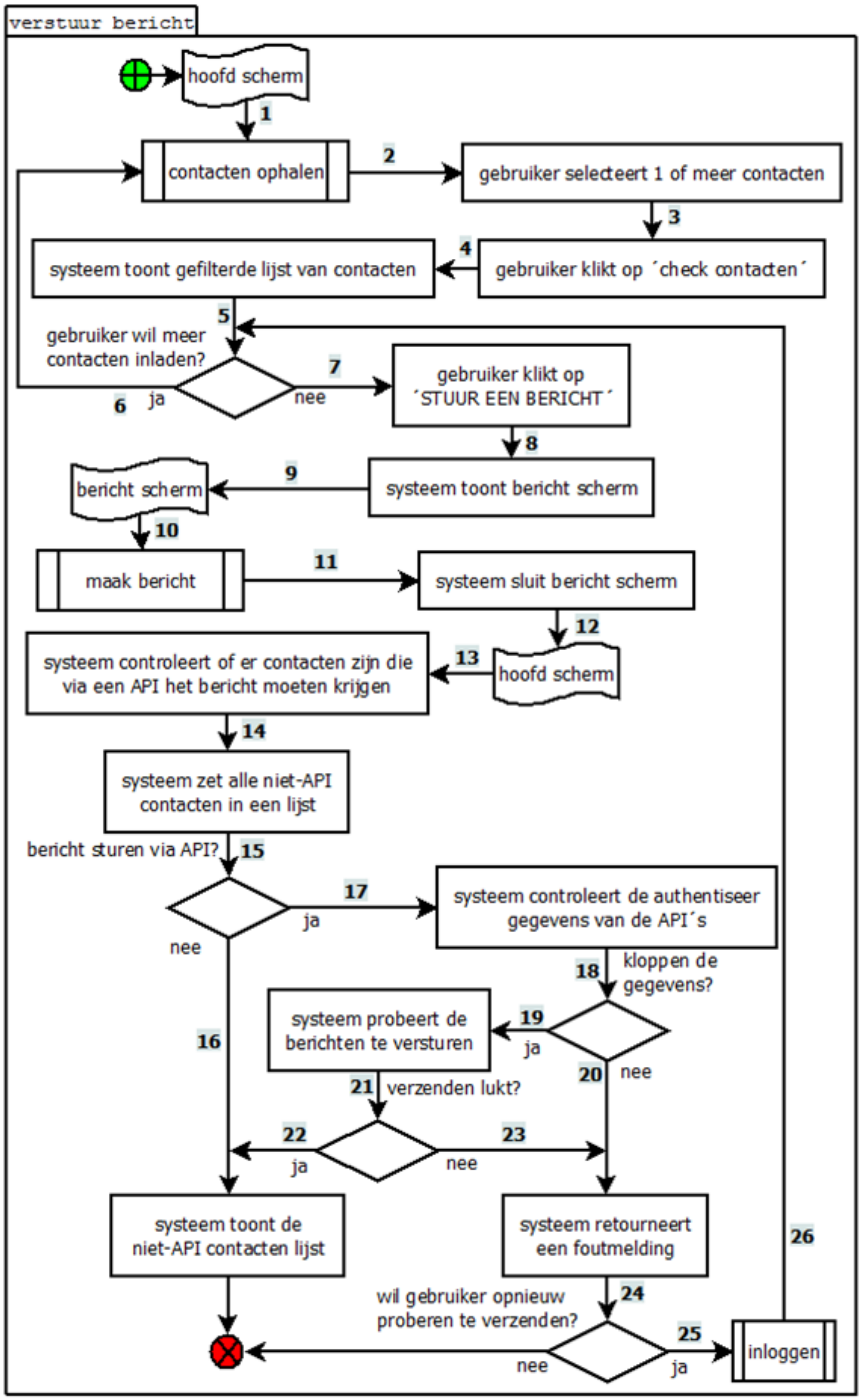
De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2,3,4,5
b	6,2,3,4,5
c	7,8,9,10
d	11,8,9,10
e	12,13,14,E

2.1.2.4 Verstuur bericht

Het uiteindelijke resultaat van dit stroomdiagram is dat 1 van de volgende stellingen waar zijn:

- De berichten naar de SmA contacten zijn niet verstuurd omdat de gebruiker verkeerd is ingelogd. Er zal hierover een foutmelding getoond worden aan de gebruiker.
- De berichten naar de SmA contacten zijn verstuurd, en er wordt een melding getoond waarin alle email-contacten staan.



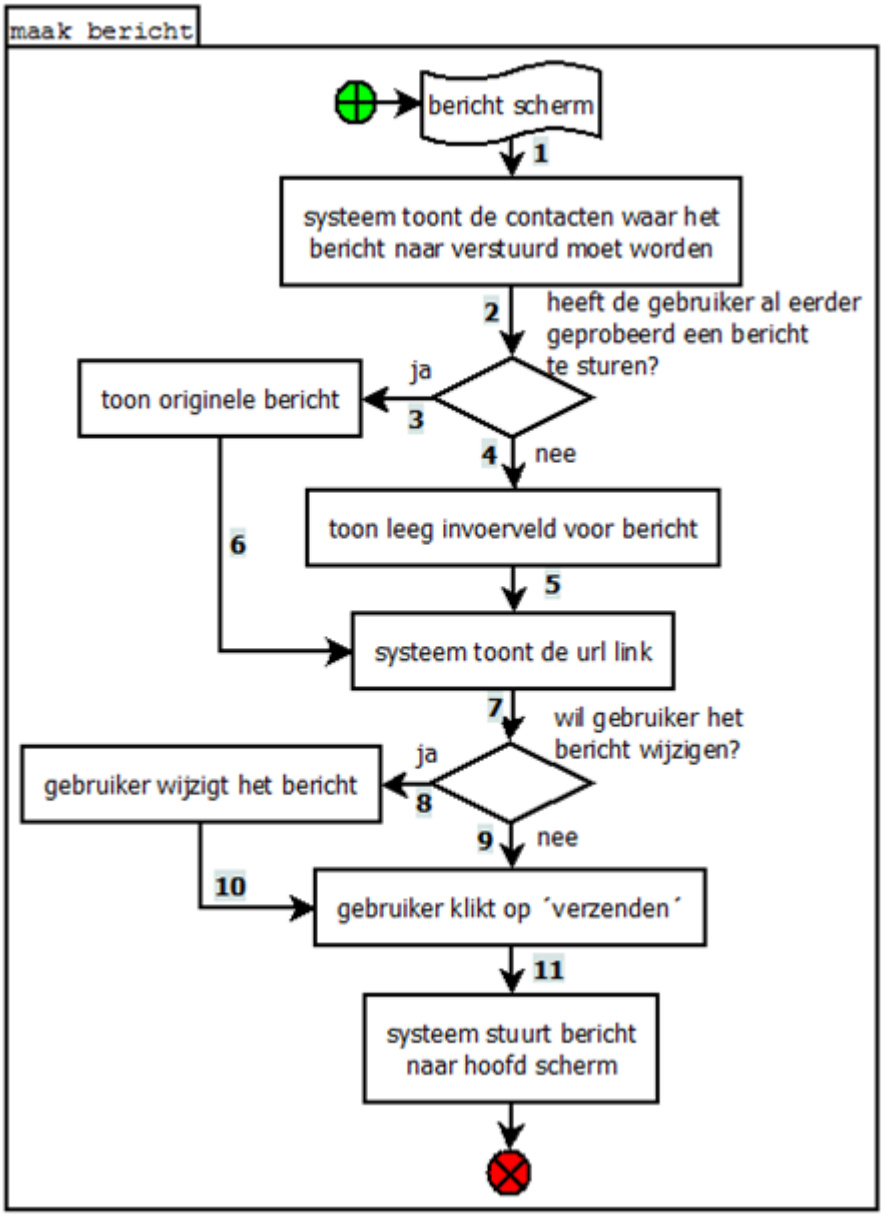
Figuur 50, verstuur bericht stroomdiagram

De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2,3,4,5
b	6,2,3,4,5
c	7,8,9,10,11,12,13,14,15
d	16,E
e	17,18
f	19,21
g	20,24
h	22,E
i	23,24
j	25,26
k	E

2.1.2.5 Maak bericht

Het uiteindelijke resultaat van dit stroomdiagram is dat er een bericht geschreven is door de gebruiker.



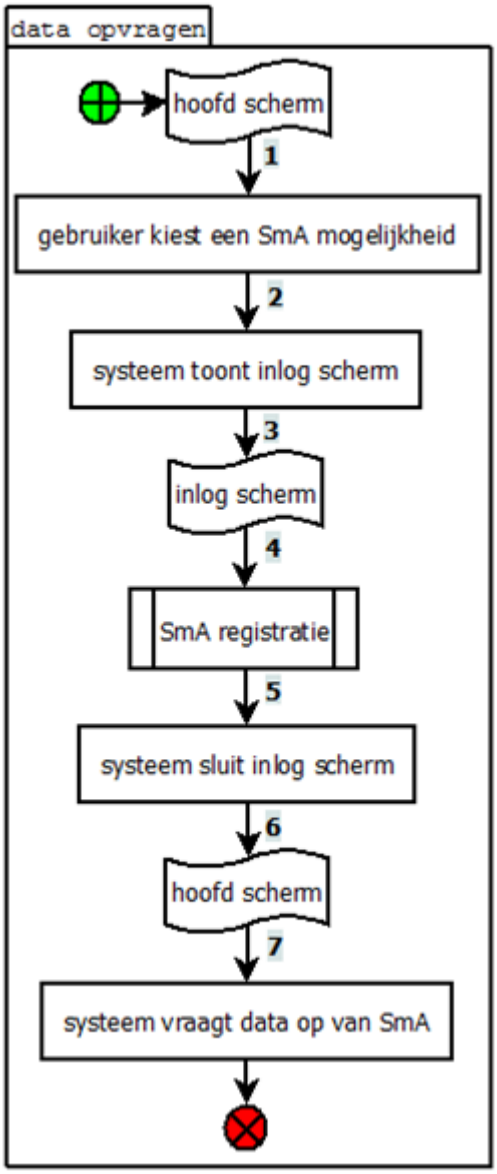
Figuur 51, maak bericht stroomdiagram

De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2
b	3,6,7
c	4,5,7
d	8,10,11,E
e	9,11,E

2.1.2.6 Data opvragen

Het uiteindelijke resultaat van dit stroomdiagram is dat er data ontvangen is van de SmA.



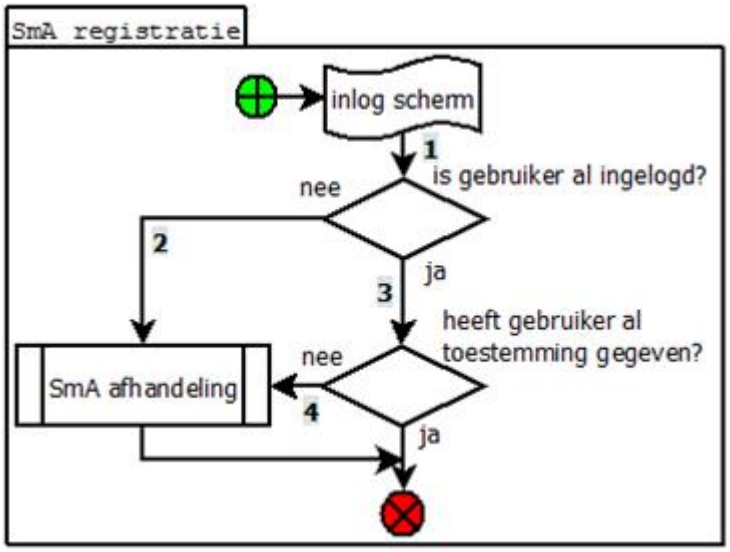
Figuur 52, data opvragen stroomdiagram

De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1,2,3,4,5,6,7,E

2.1.2.7 SmA registratie

Het uiteindelijke resultaat van dit stroomdiagram is dat de gebruiker ingelogd is bij de SmA, en dat de gebruiker toestemming heeft gegeven voor het gebruik van de data van de gebruiker, door Datumprikker.



Figuur 53, SmA registratie stroomdiagram

De volgende route combinaties zijn mogelijk:

combinatie	paden
a	S,1
b	2,E
c	3
d	4,E
e	E

2.1.2.8 SmA afhandeling

Deze use case is extern, en valt dus buiten de scope. Dit proces wordt namelijk door de SmA geregeld.

Verwacht wordt dat dit proces uitvoerig getest is, en dat dit dan ook altijd correct wordt afgerond.

2.1.3 Testsessies

Er worden voor het proof-of-concept 4 testsessies uitgevoerd:

- **Inloggen**; hier wordt het "**Inloggen**" (en alle hieraan gekoppelde) stroomdiagram(men) gevolgd. De volgende SmA's worden hiermee getest:
  - Live
  - Facebook
  - Google
  - LinkedIn
- **Contacten ophalen**; hier wordt het "**Contacten ophalen**" (en alle hieraan gekoppelde) stroomdiagram(men) gevolgd. De volgende SmA's worden hiermee getest:
  - Live
  - Facebook
  - Google
  - LinkedIn
- **Berichten versturen**; hier wordt het "**Verstuur bericht**" (en alle hieraan gekoppelde) stroomdiagram(men) gevolgd. De volgende SmA's worden hiermee getest:
  - Live (email)
  - Facebook (email)
  - Google (email)
  - LinkedIn (API)
- **Kalender controleren**; hier wordt het "**Kalender controleren**" (en alle hieraan gekoppelde) stroomdiagram(men) gevolgd. De volgende SmA's worden hiermee getest:
  - Live
  - Facebook
  - Google

2.1.3.1 Inloggen

2.1.3.1.1 Uitvoering

2.1.3.1.1.1 Inloggen

Inloggen	Live	Facebook	Google	LinkedIn
Route {a}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.1.1.2 Data opvragen

Data opvragen	Live	Facebook	Google	LinkedIn
Route {a}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.1.1.3 SmA registratie

SmA registratie	Live	Facebook	Google	LinkedIn
Route {a, b}	succesvol	succesvol	succesvol	succesvol
Route {a, c, d}	succesvol	succesvol	succesvol	succesvol
Route {a, c, e}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.1.2 conclusie

Bij alle SmA's zijn alle paden succesvol uitgevoerd, zonder dat er fouten optraden. Daarnaast waren alle resultaten zoals verwacht.

De conclusie die dus getrokken kan worden, is dat de functionaliteit "inloggen" correct werkt.

2.1.3.2 Contacten ophalen

2.1.3.2.1 Uitvoering

2.1.3.2.1.1 Contacten ophalen

Contacten ophalen	Live	Facebook	Google	LinkedIn
Route {a}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.2.1.2 Data opvragen

Data opvragen	Live	Facebook	Google	LinkedIn
Route {a}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.2.1.3 SmA registratie

SmA registratie	Live	Facebook	Google	LinkedIn
Route {a, b}	succesvol	succesvol	succesvol	succesvol
Route {a, c, d}	succesvol	succesvol	succesvol	succesvol
Route {a, c, e}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.2.2 conclusie

Bij alle SmA's zijn alle paden succesvol uitgevoerd, zonder dat er fouten optraden. Daarnaast waren alle resultaten zoals verwacht.

De conclusie die dus getrokken kan worden, is dat de functionaliteit "contacten ophalen" correct werkt.

2.1.3.3 Berichten versturen

Bij deze testsessie worden de SmA's niet los getest, maar in paren. Dit is gedaan omdat het mogelijk is om tegelijk berichten te sturen naar verschillende SmA's. De paren zijn zo gekozen dat er ten minste 1 keer geen API berichten worden verstuurd, en ten minste 1 keer wel minimaal 1 API bericht.

Omdat het wel of niet versturen via de API zware invloed heeft op de mogelijke te belopen paden, kunnen bij deze sessie niet alle routes belopen worden door alle paren.

2.1.3.3.1 Uitvoering

2.1.3.3.1.1 Verstuur bericht

Verstuur bericht	Live & Facebook	Google & LinkedIn
Route {a, b, c, d}	succesvol	succesvol
Route {a, c, e, g, j, b, c, e, f, h}	onmogelijk	succesvol
Route {a, c, e, g, j, b, c, e, f, i, k}	onmogelijk	succesvol
Eindresultaat zoals verwacht?	ja	ja
Fout bij pad:	-	-

2.1.3.3.1.2 Contacten ophalen

Aangezien het hier om exact dezelfde use case en data gaat als bij de test sessie **Contacten ophalen**, zal de uitslag hiervan ook exact hetzelfde zijn. Voor de uitslag wordt dan ook naar deze testsessie verwezen.

Wel belangrijk om op te merken is dat de **Contacten ophalen** testsessie ook bij deze testsessie alle SmA's los behandeld, en dus niet in koppels.

2.1.3.3.1.3 Maak bericht

Maak bericht	Live & Facebook	Google & LinkedIn
Route {a, c, e}	succesvol	onmogelijk
Route {a, b, d}	onmogelijk	succesvol
Eindresultaat zoals verwacht?	ja	ja
Fout bij pad:	-	-

2.1.3.3.1.4 Inloggen

Aangezien het hier om exact dezelfde use case en data gaat als bij de test sessie **Inloggen**, zal de uitslag hiervan ook exact hetzelfde zijn. Voor de uitslag wordt dan ook naar deze testsessie verwezen.

Wel belangrijk om op te merken is dat de **Contacten ophalen** testsessie ook bij deze testsessie alle SmA's los behandeld, en dus niet in koppels.

2.1.3.3.2 conclusie

Bij alle SmA's zijn (mogelijke) alle paden succesvol uitgevoerd, zonder dat er fouten optraden. Daarnaast waren alle resultaten zoals verwacht.

De conclusie die dus getrokken kan worden, is dat de functionaliteit "verstuur bericht" correct werkt.

2.1.3.4 Kalender controleren

Bij deze testsessie worden alle SmA's tegelijk getest doormiddel van 1 route. Dat wil zeggen: er is 1 lange route gevolgd waarin alle SmA's 1 keer aangesproken worden voor het ophalen van kalender data. Alle andere stellingen en bijbehorende afsplitsingen worden ook binnen deze route afgegaan.

Bij de sub-stroomdiagrammen wordt er wel weer per SmA gecontroleerd.

2.1.3.4.1.1 Kalender controleren

Maak bericht	Live & Facebook & Google & LinkedIn
Route {a, b, c, d, d, e}	succesvol
Eindresultaat zoals verwacht?	ja
Fout bij pad:	-

2.1.3.4.1.2 Data opvragen

Data opvragen	Live	Facebook	Google	LinkedIn
Route {a}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.4.1.3 SmA registratie

SmA registratie	Live	Facebook	Google	LinkedIn
Route {a, b}	succesvol	succesvol	succesvol	succesvol
Route {a, c, d}	succesvol	succesvol	succesvol	succesvol
Route {a, c, e}	succesvol	succesvol	succesvol	succesvol
Eindresultaat zoals verwacht?	ja	ja	ja	ja
Fout bij pad:	-	-	-	-

2.1.3.4.2 conclusie

Bij alle SmA's zijn (mogelijke) alle paden succesvol uitgevoerd, zonder dat er fouten optraden. Daarnaast waren alle resultaten zoals verwacht.

De conclusie die dus getrokken kan worden, is dat de functionaliteit "kalender controleren" correct werkt.

2.1.4 Eindconclusie

Alle testsessie zijn succesvol afgerond, en de eindresultaten waren zoals verwacht. Er kan dus geconcludeerd worden dat het proof-of-concept correct werkt volgens de use cases.

2.1.5 Aanbevelingen

Hoewel het product correct werkt, is het nog niet helemaal klaar voor de productie omgeving. Er moeten aan de kant van Datumprikker nog een aantal wijzigingen doorgevoerd worden. Ook moet er aan de front-end aanpassingen gemaakt worden aan het proof-of-concept, aangezien dit niet gebruiksvriendelijk (of zelfs bruikbaar voor een gebruiker) is.

Aan de kant van de plug-in module hoeft er echter nagenoeg niets meer veranderd te worden. Deze haalt alle data tenslotte correct op.

2.2 Monitoring

Nadat (een deel van de) plug-in module in de live omgeving verwerkt werd, konden de gebruikers van Datumprikker gebruik maken van de functionaliteit van het inladen van contacten (van Google en Live).

Al test je een product nog zo vaak en intens op de test-omgeving, zodra (externe) gebruikers erbij betrokken raken, komen er over het algemeen dingen naar voren die niet voorzien waren. Het gaat hier dan over het algemeen over zeer specifieke en uitzonderlijke fouten die optreden omdat de gebruiker net een andere configuratie, applicatie en/of manier van werken heeft dan wat verwacht werd.

Ook kan het voorkomen dat de gebruikers in een groter getale (tegelijk) gebruik maken van het product dan er getest is, waardoor er problemen kunnen ontstaan.

Om deze problemen zo snel mogelijk te kunnen traceren en verhelpen, is er rond de live-gang begonnen met het monitoren van de nieuwe functionaliteit.

2.2.1 Ophalen van contacten

De nieuwe functionaliteit[50] om contacten op te halen uit Google en Live is nu doormiddel van de plug-in module mogelijk gemaakt. Hierbij wordt een zeer groot gedeelte van de plug-in module (exclusief de niet gebruikte implementaties uiteraard) gebruikt, waardoor het zeer interessant is om te weten of het afstudeerproject gelukt is.

Bij het monitoren zijn de volgende dingen bijgehouden:

- Hoe vaak wordt er op ‘contacten inladen’ geklikt?
- Hoe vaak worden er contacten ook daadwerkelijk ingeladen?

Hierbij is er een scheiding tussen Google en Live gemaakt, en zijn er dus 4 meetpunten die voor de plug-in module gelden. De andere punten die gemeten worden zijn voor andere nieuwe onderdelen van Datumprikker, en zijn niet interessant voor dit afstudeerproject.

2.2.2 Resultaten

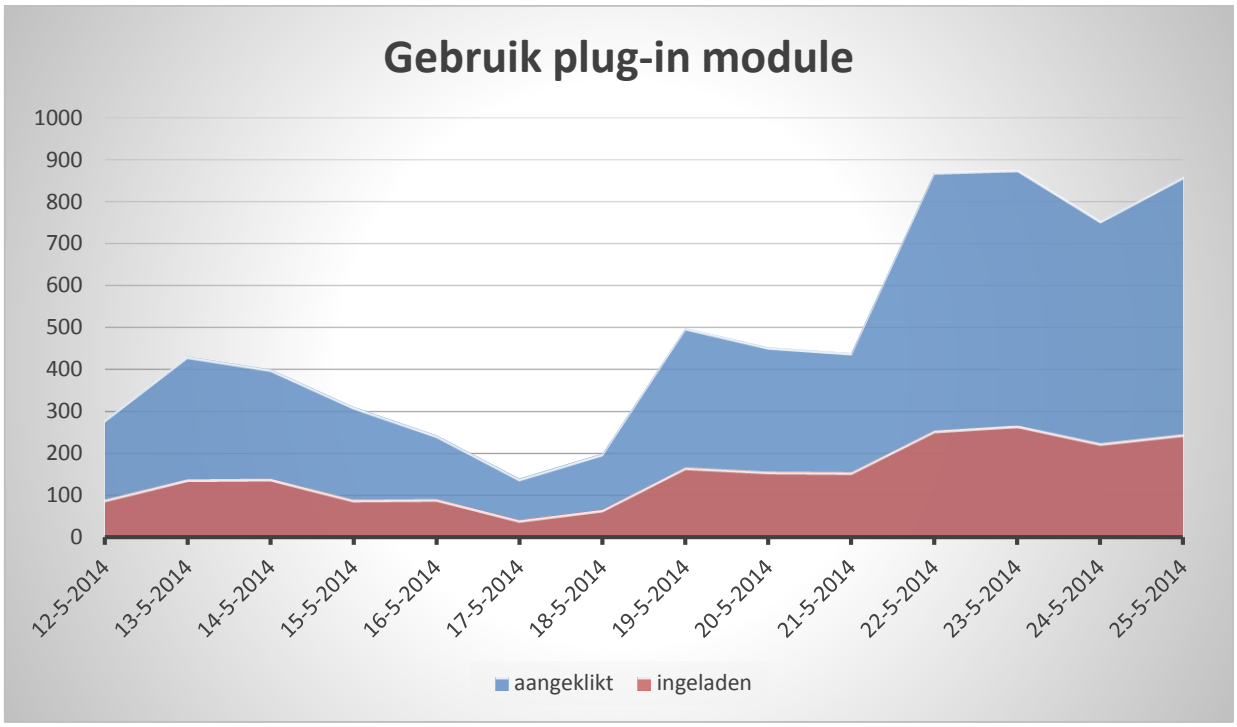
2.2.2.1 Bezoekers aantallen

In de grafieken **grafiek 1**, **grafiek 2** en **grafiek 3** zijn de resultaten van de monitoring te zien. Op het moment van schrijven zijn er (in 2 weken) **2119** keer succesvol contacten ingeladen.

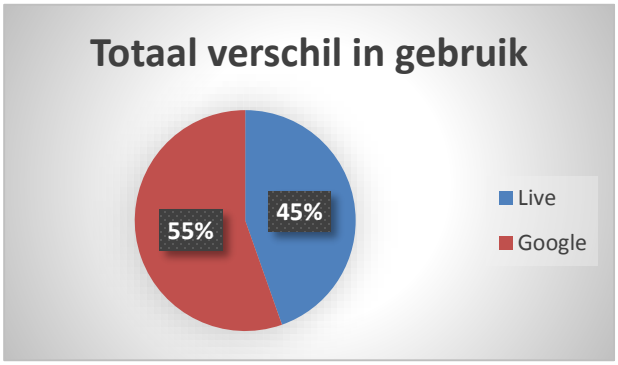
Aan de hand van de grafieken kunnen er een aantal conclusies getrokken worden:

- Er is een stijgende trend rondom het gebruik van de plug-in module, waarbij vooral het aantal mensen dat de optie bekijkt (maar uiteindelijk niet doorzet) behoorlijk stijgt.
- Er zit niet veel verschil in tussen de social media qua gebruik. Wel blijft Google iets populairder.
- Slechts een kwart van de mensen die de knop aanklikt om contacten in te laden, laadt ook daadwerkelijk contacten in. De rest bedacht zich bij het inloggen/rechten geven.

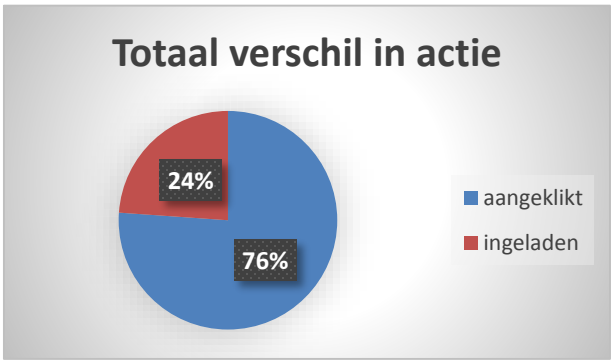
[50] De functionaliteit om contacten op te halen uit Google bestond al, maar is nu met de plug-in module vervangen.



grafiek 1, Gebruik plug-in module



grafiek 3, Totaal verschil in gebruik



grafiek 2, Totaal verschil in actie

2.2.2.2 Opgetreden fouten

Tijdens de monitoring was er slechts 1 fout opgetreden. Deze zal hieronder besproken worden:

2.2.2.2.1 Leeg aantal contacten

2.2.2.2.1.1 Probleem

Deze fout trad op toen een gebruiker contacten probeerde in te laden van diens Google-account, zonder dat er bij dit account contacten bekend waren.

2.2.2.2.1.2 Oorzaak

Hoewel er in de code wel werd gecontroleerd of het resultaat van Google leeg is, werd er niet (correct) gecontroleerd of de ontvangen contactenlijst ook leeg was.

Doordat er daarna vanuit gegaan werd dat de lijst gevuld is, trad er een interne fout op. Deze genereerde de correcte foutmelding bij het resultaat, die correct opgevangen werd door Datumprikker. De gebruiker kreeg hierdoor simpelweg een lege lijst te zien (zoals uiteindelijk ook te verwachten was).

Dit probleem werd gevalideerd door met een nieuw Google-account (zonder contacten) gebruik te maken van de plug-in module.

2.2.2.2.1.3 Oplossing

Bij de controle wordt er nu alsnog (correct) gecontroleerd of het ontvangen resultaat leeg is, inclusief (contact)lijsten etc.. De andere implementaties zijn hierop ook nog eens extra gecontroleerd.

2.2.2.2.1.4 Opgelost?

De fout kwam niet meer voor na de oplossing, ook niet met het hiervoor gemaakte test-account.