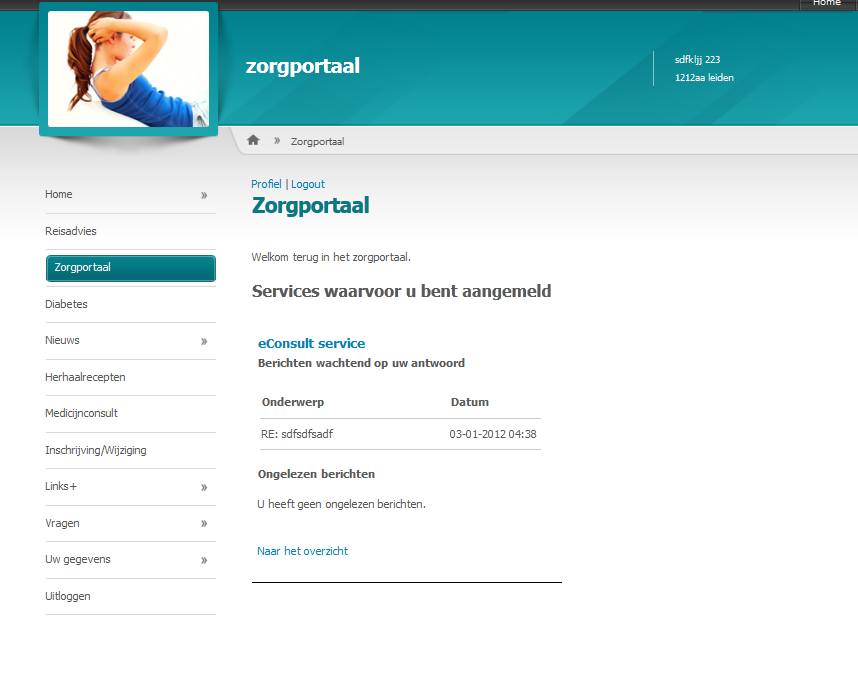
**Bijlages**



## Samenbrengen van zorgservices in het Zorgportaal

*Bob Brinks  
25 mei 2011  
Amsterdam*

# Referaat

Deze bijlages behoren bij het afstudeerdossier: “Samenbrengen van zorgservices in het zorgportaal” door Bob Brinks.

Deze bijlages zijn primair geschreven als documentatie over het zorgportaal binnen Pharmeon. Secondair zijn deze bijlages zijn bijgevoegd voor lezers die extra informatie willen over het afstudeerproject.

# Inhoud

|  |
| --- |
| **Document** |
| Afstudeerplan |
| Plan van aanpak |
| Architectural Description |
| Herbruikbare formulieren documentatie |
| User Stories |

**Afstudeerplan**

**Informatie afstudeerder en gastbedrijf (***structuur niet wijzigen***)**

**Afstudeerblok**: 2012-1.1

**Startdatum uitvoering afstudeeropdracht**: 6 feb 2012

**Einddatum uitvoering afstudeeropdracht**: 1 jun 2012

**Inleverdatum afstudeerdossier volgens jaarrooster**: 01 jun 2012

**Studentnummer**: 20060249

**Achternaam**: dhr Brinks (\*) *weghalen niet van toepassing*

**Voorletters**: B.

**Roepnaam**: Bob

**Adres**: Breestraat 72

**Postcode**: 2311CS

**Woonplaats**: Leiden

**Telefoonnummer**:

**Mobiel nummer**: 06 384 371 10

**Privé emailadres: bob brinks <brinks\_bob@hotmail.com>**

**Opleiding**: INF

**Locatie**: Den Haag

**Variant**: voltijd

**Naam studieloopbaanbegeleider**: W. van Vliet

**Naam begeleider/examinator**: G.A. Mijnarends

**Naam expert/examinator**: E.M. van Doorn

**Naam bedrijf**: Pharmeon

**Afdeling bedrijf**: IT

**Bezoekadres bedrijf**: Hettenheuvelweg 14

**Postcode bezoekadres**: 1101 BN

**Postbusnummer**:

**Postcode postbusnummer**:

**Plaats**: Amsterdam Zuid-Oost:

**Telefoon bedrijf**: +31(0)20 6500100

**Telefax bedrijf**:

**Internetsite bedrijf**: pharmeon.nl/

**Achternaam opdrachtgever**: dhr Jody de Vries

**Voorletters opdrachtgever***:* J.

**Titulatuur opdrachtgever**: ......................

**Functie opdrachtgever**: ......................

**Doorkiesnummer opdrachtgever**: ......................

**Email opdrachtgever:**  [jody@pharmeon.nl](mailto:jody@pharmeon.nl)

**Achternaam bedrijfsmentor**: dhr Jody de Vries

**Voorletters bedrijfsmentor**: J

**Titulatuur bedrijfsmentor**: ......................

**Functie bedrijfsmentor**: ......................

**Doorkiesnummer bedrijfsmentor**: ......................

**Email bedrijfsmentor**: [jody@pharmeon.nl](mailto:jody@pharmeon.nl)

*NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever*

**Doorkiesnummer afstudeerder***: ......................*

**Functie afstudeerder (deeltijd/duaal)**:

**Titel afstudeeropdracht:** Ontwikkeling van een zorgportaal voor de verschillende services van Pharmeon

**Opdrachtomschrijving**

1. **Bedrijf**

Pharmeon is een full service internetonderneming gespecialiseerd in het aanbieden van Internetdiensten in de gezondheidszorg. Pharmeon is gevestigd in Amsterdam en werd in januari 2000 opgericht door een aantal specialisten uit de farmaceutische industrie en de ICT-branche.

De internetoplossingen van Pharmeon hebben tot doel zorgverleners, zoals artsen, apothekers, tandartsen en mondhygiënisten te ondersteunen in het uitoefenen van hun zorgtaak. Door zorgverleners online te brengen voorziet Pharmeon in een groeiende behoefte van consumenten aan continue, deskundige en betrouwbare informatie over gezondheid en behandelingsmogelijkheden.  
  
Pharmeon biedt de zorgsector de gelegenheid een eigen website te openen, waarmee hun klantenkring eenvoudig toegang krijgt tot een breed pakket online gezondheidsservices. Hiermee wordt het gemak van het internet gecombineerd met de betrouwbaarheid en de kwaliteit van de eigen zorgvoorzieningpraktijk.

Voorbeelden van services die Pharmeon levert zijn: de herhaalrecept service, de pil service, de diabeten service en de e-consult services. De services van Pharmeon worden gebruikt door verschillenden zorgverleners, waaronder apothekers, huisartsen en tandartsen.

1. **Probleemstelling**

In de huidige situatie zijn de services losse webapplicaties. Iedere service heeft zijn eigen database en programmatuur. Onderlinge communicatie tussen de verschillende services en hun databases bestaat niet. De integratie tussen de services is hierdoor slecht.  
  
In de onderstaande diagram is deze situatie weergegeven.

  
Afbeelding 1: De huidige situatie binnen Pharmeon.

Pharmeon wil graag een nieuwe service ontwikkelen, het zorgportaal. De bedoeling van dit zorgportaal is het aanbieden van een totaaloverzicht van een patiënt. Dit houdt in dat het zorgportaal data uit alle services moet ophalen.   
Naast het aanbieden van de informatie moet het zorgportaal ook de mogelijk bieden om de data aan te passen zoals dat in de huidige situatie met de verschillende services kan.  
  
In de huidige situatie staat data van één patiënt verspreid over meerdere databases, hierdoor is het niet mogelijk om een totaaloverzicht van een patiënt te krijgen.

Door het ontbreken van samenhang tussen de datamodellen van de verschillende services weet het systeem niet dat data uit de databases van de verschillende services mogelijk data van dezelfde patiënt zijn.

De huidige situatie is ontstaan, doordat de services die Pharmeon levert over een tijdspanne van 10 jaar als losse systemen ontwikkeld zijn. De services zijn ontwikkeld in verschillende talen; ASP (classic) en ASPX. De verschillende services zijn niet ontworpen en geïmplementeerd volgens best practices en design patterns, zo ontbreekt de scheiding tussen de model, view en controller (MVC).

1. **Doelstelling van de afstudeeropdracht**

De doelstelling voor de opdracht is het ontwikkelen van het zorgportaal waarmee een totaaloverzicht van een patiënt weergegeven en aangepast kan worden.

Bij het ontwikkelen van het zorgportaal ligt de nadruk op de backend. Deze backend is een middlewaresysteem dat de integratie tussen de verschillende services verbetert.

1. **Resultaat**

Na het succesvol uitvoeren van de opdracht heeft Pharmeon een zorgportaal dat een totaaloverzicht van een patiënt kan weergeven.



Afbeelding 2: Gewenste situatie binnen Pharmeon

De bovenstaande afbeelding laat de structuur van het zorgportaal zien.   
Met de uitvoering van de opdracht zullen de volgende systeemdelen ontwikkeld zijn.

* De middleware die de integratie van de verschillende services verbetert.
* Het zorgportaal dat een totaaloverzicht van een patiënt weergeeft en de mogelijkheid biedt om dit overzicht aan te passen.
* Twee van de API’s die de bestaande services uitbreiden voor communicatie met de middleware. Voor de andere twee services zal een ontwerp van de API gemaakt zijn. In verband met de doorlooptijd van de opdracht valt de implementatie van deze twee API’s buiten de scope van de opdracht.

De ontwikkelde systeemdelen zullen getest zijn met unittests. Het testteam van Pharmeon zal het overige testwerk op zich nemen.

1. **Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten**

|  |  |  |
| --- | --- | --- |
| **uit te voeren activiteiten / gebruikte technieken** | **naam van het resulterend product** | **Aantal dagen** |
| Orientatie:  Opstellen van plan van aanpak  Huidige situatie analyseren  Opstellen product backlog, userstories  Opstellen architectural description (AD) | Plan van aanpak  Scrum artifacts   * Product backlog * Eerste versie userstories   Eerste versie AD met de nadruk op:   * Functional viewpoint * Information viewpoint * Security perspective | 10 |
| Sprint 1:  Ontwerpen en ontwikkelen van de middleware | Ontwerp van de message bus (technisch ontwerp)  Werkende versie van de middleware met dummy API’s.  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 2:  Service uitbreiden met de API | Ontwerp van de uitbreidingen van de services (technisch ontwerp)  Werkende API die kan communiceren met de middleware  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 3:  Service uitbreiden met de API | Nieuwe versie ontwerp van de uitbreidingen van de services (technisch ontwerp)  Werkende API die kan communiceren met de middleware  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 4:  Ontwerpen en ontwikkelen van het zorgportaal | Ontwerp van het zorgportaal (technisch ontwerp)  Werkende versie van het zorgportaal  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 5:  Ontwikkelen rest producten uit de product backlog | GUI voor het zorgportaal.  Werkende versie van overige producten in de product backlog  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 6:  Project afronding | Werkende versie van het zorgportaal met realistische data, GUI en werkend met twee van de API’s  Unit tests  Scrum artifacts:   * Sprint backlog   Sprint burndown | 5 |

1. **Op te leveren (tussen)producten**
2. Scrum artifacts
   1. Product backlog
      * User stories
   2. Product burndown
   3. Sprint backlogs
   4. Sprint burndowns
3. Achitectural description met de nadruk op volgende viewpoints en perspectives
   1. Function Viewpoint
   2. Information viewpoint
   3. Secutiry perspective
4. Zorgportaal dat voldoet aan de volgende eisen
   1. Zorgverleners en patiënten kunnen met het zorgportaal alle informatie over een patiënt weergeven en/of aanpassen.
   2. Het zorgportaal draait in de ontwikkelomgeving.
5. Middleware
   1. De middleware faciliteert de communicatie tussen de verschillende services en het zorgportaal.
   2. De middleware zal metadata over de data in de verschillende services bijhouden. Hierdoor weet de middleware waar welke data over een patiënt zijn opgeslagen. Tevens bevat de middleware domeinkennis.
   3. De middleware verbetert de datavoorziening binnen Pharmeon door de data in de verschillende services voor elkaar beschikbaar te maken.
6. **Te demonstreren competenties en wijze waarop**

|  |  |  |  |
| --- | --- | --- | --- |
| Nr beroepstaak | niveau | Tekst van de beroepstaak | Deze beroepstaak wordt in het bijzonder aangetoond met het dit product (genoemd onder punt 6) |
| **2.1** | **3** | Analyseren en beschrijven van gegevens over objecten in hun onderlinge samenhang. | Information viewpoint in het AD |
| **3.1** | **3** | Bepalen en beschrijven van de architectuur van een applicatie. | Ontwerp voor de nieuwe architectuur |
| **3.2** | **3** | Ontwerpen systeemdeel | Functional viewpoint in het AD |
| **3.3** | **3** | Bouwen applicatie | Het zorgportaal, middleware en API’s. |

**Verklarende woordenlijst**

Service:

Binnen dit document wordt het woord service gebruikt om een webapplicatie van Phameon te beschrijven. Voorbeelden van services die Pharmeon levert zijn: de herhaalrecept service, de pil service, de diabeten service en de e-consult services.

Middleware:

“In the computer industry, middleware is a general term for any programming that serves to "glue together" or mediate between two separate and often already existing programs. A common application of middleware is to allow programs written for access to a particular[database](http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci211895,00.html) to access other databases.”  
Bron: <http://searchsoa.techtarget.com/definition/middleware>

API:

“Een application programming interface (API) is een verzameling definities op basis waarvan een [computerprogramma](http://nl.wikipedia.org/wiki/Computerprogramma) kan communiceren met een ander programma of onderdeel (meestal in de vorm van [bibliotheken](http://nl.wikipedia.org/wiki/Bibliotheek_(informatica))).”  
Bron: <http://nl.wikipedia.org/wiki/Application_programming_interface>

**Plan van aanpak**

Ontwikkelen zorgportaal

Pharmeon  
Bob Brinks

# Inleiding

In dit plan van aanpak staat beschreven hoe de opdracht “Ontwikkeling van een zorgportaal voor de verschillende services van Pharmeon” uitgevoerd zal worden. Er zal een systeem ontwikkeld worden waarmee een totaaloverzicht van een patiënt weergegeven kan worden.

Deze opdracht zal worden uitgevoerd bij het bedrijf Pharmeon. Pharmeon is een full service Internet onderneming gespecialiseerd in het verlenen van internetdiensten binnen de gezondheidszorg. De internetoplossingen van Pharmeon ondersteunen medische professionals bij het bieden van online zorg aan de consument.

Voorbeelden van services die Pharmeon levert zijn: de herhaalrecept service, de pil service, de diabeten service en de e-consult services. De services van Pharmeon worden gebruikt door verschillenden zorgverleners, waaronder apothekers, huisartsen en tandartsen.

# Aanleiding

Gebruikers van de services die Pharmeon levert moeten om verschillende informatie op te vragen steeds veranderen van services. Onderlinge communicatie tussen de verschillende services en hun databases bestaat niet. Hier door kan geen totaaloverzicht van een patiënt weergegeven worden.

Het ontbreken van de onderlinge communicatie zorgt er ook voor dat de gebruiker onnodige stappen moet ondernemen bij het gebruiken van de verschillende services. Voorbeelden hiervan zijn:

* Bij het eerste gebruik van elk van de services moet de gebruiker zich opnieuw registreren en daarbij opnieuw zijn NAW gegevens invoeren. Als de gebruiker meerdere services gebruikt zijn deze gegevens al bekend binnen het systeem en zouden deze gegevens met behulp van onderlinge communicatie tussen de services opgevraagt kunnen worden.
* Als de gebruiker, na het gebruik van de ene service een andere service wilt gebruiken moet de gebruiker opnieuw inloggen. Dit is het geval doordat de gebruikersessies (met daarin de inlog-status van de gebruiker) niet tussen de services gedeeld worden.

# Probleemstelling

In de huidige situatie zijn de services losse webapplicaties. Iedere service heeft zijn eigen database en programmatuur. Onderlinge communicatie tussen de verschillende services en hun databases bestaat niet. De integratie tussen de services is hierdoor slecht.  
  
In de onderstaande diagram is deze situatie weergegeven.

  
Afbeelding 1: De huidige situatie binnen Pharmeon.

Pharmeon wil graag een nieuwe service ontwikkelen, het zorgportaal. De bedoeling van dit zorgportaal is het aanbieden van een totaaloverzicht van een patiënt. Dit houdt in dat het zorgportaal data uit alle services moet ophalen.   
Naast het aanbieden van de informatie moet het zorgportaal ook de mogelijk bieden om de data aan te passen zoals dat in de huidige situatie met de verschillende services kan.  
  
In de huidige situatie staat data van één patiënt verspreid over meerdere databases, hierdoor is het niet mogelijk om een totaaloverzicht van een patiënt te krijgen.

Door het ontbreken van samenhang tussen de datamodellen van de verschillende services weet het systeem niet dat data uit de databases van de verschillende services mogelijk data van dezelfde patiënt zijn.

De huidige situatie is ontstaan, doordat de services die Pharmeon levert over een tijdspanne van 10 jaar als losse systemen ontwikkeld zijn. De services zijn ontwikkeld in verschillende talen; ASP (classic) en ASPX. De verschillende services zijn niet ontworpen en geïmplementeerd volgens best practices en design patterns, zo ontbreekt de scheiding tussen de model, view en controller (MVC).

# Doelstelling van de afstudeeropdracht

De doelstelling voor de opdracht is het ontwikkelen van het zorgportaal waarmee een totaaloverzicht van een patiënt weergegeven en aangepast kan worden.

Bij het ontwikkelen van het zorgportaal ligt de nadruk op de backend. Deze backend is een middlewaresysteem dat de integratie tussen de verschillende services verbetert.

# Uitgangsituatie

Zaken benodigd om het project uit te voeren.

## 5.1. Software

- Microsoft Visual Studio 2008

- Microsoft SQL server 2000

- Microsoft SQL server management studio

- Microsoft Office 2007

## 5.2. Hardware

- Werkplek

- Ontwikkelomgeving

- Testomgeving

## 5.3. Relevante documentatie

-Reverse engineered database modellen.

-Projectopzet – Online patiëntenkluis.doc (bijlage)

-Services overzichtspagine.docx (bijlage)

## 5.4. Betrokken personen

Hieronder de personen die betrokken zijn bij dit project.

|  |  |
| --- | --- |
| **Persoon** | **Rol** |
| Tom Meintser | Product owner |
| Erik-Jan de Keijzer | Begeleider |
| G.A. Mijnarends | Begeleider/examinator school |
| E.M. van Doorn | Expert/examinator school |
| Ashvin Jiawon | Scrum master |
| Allan Sonnenberg | Scrum Team |
| Erik-Jan de Keijzer | Scrum Team |
| Garry Ralm | Scrum Team |
| Mitchel van der Poll | Scrum Team |
| Reinder Glas | Scrum Team |

# Te hanteren methoden en technieken

## Methoden

### Scrum

Tijdens dit project zal er gebruik worden gemaakt van Scrum als projectfasering. Deze methode wordt binnen het bedrijf gebruik voor projecten en zal dus ook voor dit project gebruikt worden.

Binnen dit project zullen de volgende Scrum artifacts opgeleverd worden:

* Product backlog
  + User stories
* Product burndown
* Sprint backlogs
* Sprint burndowns

### UML

De ontwerp documentatie wordt geschreven in UML.

### Test driven developement

Voor dit project zal test driven developement gebruikt worden. Binnen Pharmeon wordt tijdens dit project in projecten die synchroon lopen test driven developement ingevoerd.   
Met test driven developement wordt bedoeld dat er eerst unit tests geschreven worden en daarna code geimplementeerd wordt die deze tests moet doen slagen.

### Technieken

* .NET framework
* C#
* ASP(X)
* SQL
* (X)HTML
* XML
* XSLT
* Javascript/JQuery

# Op te leveren producten

Dit project begint met een oriëntatie fase, in deze fase zullen de volgende producten opgeleverd worden:

* Plan van aanpak
* Scrum artifacts
  + Product backlog
  + Eerste versie userstories
* Eerste versie AD met de nadruk op:
  + Functional viewpoint
  + Information viewpoint
  + Security perspective

Na de oriëntatie fase zullen per sprint de volgende producten opgeleverd worden.

Sprint 1:Ontwerpen en ontwikkelen van de middleware

* Ontwerp van de message bus (technisch ontwerp)
* Werkende versie van de middleware met dummy API’s.
* Unit tests
* Scrum artifacts:
* Sprint backlog
* Sprint burndown

Sprint 2: Service uitbreiden met de API

* Ontwerp van de uitbreidingen van de services (technisch ontwerp)
* Werkende API die kan communiceren met de middleware
* Unit tests
* Scrum artifacts:
  + Sprint backlog
  + Sprint burndown

Sprint 3: Service uitbreiden met de API

* Nieuwe versie ontwerp van de uitbreidingen van de services (technisch ontwerp)
* Werkende API die kan communiceren met de middleware
* Unit tests
* Scrum artifacts:
  + Sprint backlog
  + Sprint burndown

Sprint 4: Ontwerpen en ontwikkelen van het zorgportaal

* Ontwerp van het zorgportaal (technisch ontwerp)
* Werkende versie van het zorgportaal
* Unit tests
* Scrum artifacts:
  + Sprint backlog
  + Sprint burndown

Sprint 5: Ontwikkelen rest producten uit de product backlog

* GUI voor het zorgportaal.
* Werkende versie van overige producten in de product backlog
* Unit tests
* Scrum artifacts:
  + Sprint backlog
  + Sprint burndown

Sprint 6: Project afronding

* Werkende versie van het zorgportaal met realistische data, GUI en werkend met twee van de API’s
* Unit tests
* Scrum artifacts:
  + Sprint backlog
  + Sprint burndown

# Planning

|  |  |  |
| --- | --- | --- |
| **uit te voeren activiteiten / gebruikte technieken** | **naam van het resulterend product** | **Aantal dagen** |
| Orientatie:  Opstellen van plan van aanpak  Huidige situatie analyseren  Opstellen product backlog, userstories  Opstellen architectural description (AD) | Plan van aanpak  Scrum artifacts   * Product backlog * Eerste versie userstories   Eerste versie AD met de nadruk op:   * Functional viewpoint * Information viewpoint * Security perspective | 10 |
| Sprint 1:  Ontwerpen en ontwikkelen van de middleware | Ontwerp van de message bus (technisch ontwerp)  Werkende versie van de middleware met dummy API’s.  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 2:  Service uitbreiden met de API | Ontwerp van de uitbreidingen van de services (technisch ontwerp)  Werkende API die kan communiceren met de middleware  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 3:  Service uitbreiden met de API | Nieuwe versie ontwerp van de uitbreidingen van de services (technisch ontwerp)  Werkende API die kan communiceren met de middleware  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 4:  Ontwerpen en ontwikkelen van het zorgportaal | Ontwerp van het zorgportaal (technisch ontwerp)  Werkende versie van het zorgportaal  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 5:  Ontwikkelen rest producten uit de product backlog | GUI voor het zorgportaal.  Werkende versie van overige producten in de product backlog  Unit tests  Scrum artifacts:   * Sprint backlog * Sprint burndown | 10 |
| Sprint 6:  Project afronding | Werkende versie van het zorgportaal met realistische data, GUI en werkend met twee van de API’s  Unit tests  Scrum artifacts:   * Sprint backlog   Sprint burndown | 5 |

Belangrijke data voor activiteiten school:

|  |  |
| --- | --- |
| **Week** | **Activiteit** |
| 6 | Bezoek stagebegeleider. |
| 9/ 10 | Bespreking concept afstudeerdossier (60%). |
| 14 / 15 | Tussentijds assessment afstudeerdossier (80%). |
| 17, 1 Juni, voor 13:00 | Afstudeerdossier inleveren. |

**Achitectural Description 1.3**

## Ontwikkeling van het zorgportaal

*Bob Brinks  
7 december 2011  
Amsterdam*

Inhoud

[Ontwikkeling van het zorgportaal 1](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453011)

[Change log 3](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453012)

[Inleiding 4](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453013)

[Functional viewpoint 5](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453014)

[Componenten en Interfaces van het zorgportaal 5](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453015)

[Middleware 7](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453016)

[Losse services 10](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453017)

[Het zorgportaal 10](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453018)

[Information viewpoint 11](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453019)

[Shared tables storage 12](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453020)

[Sites, gebruikers en services 12](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453021)

[Bestanden en bestandservers 13](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453022)

[SSIM omgeving storage 14](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453023)

[eConsult service storage 15](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453024)

[Herhaalrecept service storage 17](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453025)

[Pilservice storage 18](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453026)

[Webagenda storage 19](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453027)

[Information Flow 20](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453028)

[Information ownership 20](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453029)

[Deployment Viewpoint 21](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453030)

[Live servers infrastructuur 21](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453031)

[SSSIM omgeving infrastructuur 23](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453032)

[LIPU 24](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453033)

[Acceptatie en Developement servers 25](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453034)

[The availability and resilience perspective 26](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453035)

[Voor requests via de middleware 26](file:///C:\Users\Bob-thuis\Documents\Actitectural%20Description%201.3.docx#_Toc318453036)

# Change log

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Versie | Change | Reden |
| 25-11-2011 | 1.0 | Eerste versie | - |
| 07-12-2011 | 1.1 | De services registreren hoeven zichzelf niet langer te registreren bij de Middleware. Het registreren zal gedaan worden met behulp van een configuration tool. | Omdat services in runtime niet van zichzelf weten of ze geregisteerd staan bij de middleware zouden ze dit met elke “run” (page request) moeten checken en zichzelf opnieuw registreren. Dit zorgt voor veel onnodig dataverkeer in de servers van Pharmeon. |
| 08-12-2011 | 1.1 | Beschrijving van LIPU toegevoegd en beschreven hoe de zorgportaal samenvalt met LIPU | De zorgportaal gebruikt LIPU omdat het anders niet past binnen andere services en websites van Pharmeon. |
| 13-12-2011 | 1.2 | Beschrijving van het zorgAPI protocol gedefinieerd | Was hiervoor nog niet genoeg informatie om deze definitie te schrijven (niet genoeg kennis van C#). |
| 16-12-2011 | 1.2 | Required attribute toegevoegd aan de parameters in functie definities in het commondatamodel | Om het mogelijk te maken om functies te overloaden. |
| 20-12-2011 | 1.2 | Inleidingen en contect bij de bestaande informatie geschreven | Om de leesbaarheid van het document te verbeteren. |
| 27-12-2011 | 1.3 | RequiresUser attribute toegevoegd aan functie definities in het commondatamodel | Dit zorgt ervoor dat userId niet iedere keer meegegeven hoeft te worden bij het aanroepen van functies waarvoor de gebruiker ingelogged moet zijn. Ook maakt het het afhandelen van exceptions waar de gebruiker niet is ingelogt generieker en gecentraliseert. |
| 27-12-2011 | 1.3 | Listed, humanReadableName, id en url toegevoegd aan functie definities in het commondatamodel | Om het mogelijk te maken naar een service te linken, aan te geven welke services in het overzicht van het zorgportaal moeten verschijnen en een leesbare naam te kunnen tonen. |

# Inleiding

In dit document staat de architectuur van het zorgportaal systeem en de ICT-omgeving waarin het zorgportaal zal draaien beschreven.

De architectuur wordt beschreven aan de hand van enkele viewpoints en perspectives, deze zijn gebaseerd op Addison-Wesley “Software Systems Architecture”. Om de hoeveelheid documentatie tot het nodige te beperken heb ik ervoor gekozen niet alle viewpoints en perspectives op te nemen in de beschrijving van het zorgportaal systeem.

Viewpoints:

* Functional Viewpoint
* Information Viewpoint
* Deployment Viewpoint

Perspectives

* Availability and resilience perspective

# Functional viewpoint

Dit hoofdstuk beschrijft de functionele elementen in de runtime van het zorgportaal systeem. Van deze elementen worden de verantwoordelijkheden en interfaces beschreven.

## Componenten en Interfaces van het zorgportaal

**Probleem**  
Het *zorgportaal* moet voor het uitvoeren van zijn verantwoordelijkheid, het aanbieden van data uit verschillende services, data ophalen en wegschrijven in de *verschillende services*. De services zijn niet ontwikkeld met de mogelijkheid om te integreren met andere applicaties (bijvoorbeeld door een API aan te bieden).

Bij het ophalen en wegschrijven van deze data moet het zorgportaal rekening houden met de geïmplementeerde bussiness logica van de services.  
De services kunnen data op verschillende manieren aanbieden, enkele voorbeelden hiervan zijn:

* Een service die draait binnen de zelfde .NET runtime als het zorgportaal zelf, de methodes van deze service zouden direct aangeroepen kunnen worden.
* Een service die draait als webservice en data via SOAP (XML) aanbied.
* Een service die draait in ASP classic en data via HTTP requests aanbied.

**Oplossing**  
Om de data uit de verschillende services op een gestandaardiseerde manier aan te bieden is er voor gekozen tussen de services (inclusief de Zorgportaal service) een *middleware systeem* te ontwikkelen. Deze middleware regelt de infrastructuur waarover de data uit de services verstuurd wordt.

## 

Figuur 1: Interne Componenten en interfaces van het zorgportaal systeem

In de bovenstaande diagram zijn de verschillende services vervangen door ‘Services’ en de bijbehorende API’s zijn vervangen door ‘Service API’ om de diagram leesbaar te houden. Elk van de services bied een API, de middleware communiceert met deze API om informatie op te halen of weg te schrijven.

De verantwoordelijkheden en werking van de verschillende componenten wordt in de volgende paragraven toegelicht.

## Middleware

Het belangrijkste component van het zorgportaal systeem is de Middleware. Dit component ligt tussen de verschillende services en het zorgportaal in.

**Verantwoordelijkheden**De verantwoordelijkheden van de Middleware:

* De middleware houdt een lijst bij van beschikbare services
* De middleware houdt bij hoe de verschillende services aangeroepen kunnen worden (infrastructuur).
  + Draait de service in de zelfde runtime of als webservice of bied de services zijn data op een andere manier aan?
  + Welk communicatie protocol wordt gebruikt voor het ophalen van de data?
* De middleware houdt bij welke gebruiker is ingelogd (authenticatie en autorisatie).
  + Welke data mag de ingelogde gebruiker inzien/aanpassen?
  + Welke functies mag de ingelogde gebruik gebruiken?
* De middleware houdt bij welke data en functionaliteiten beschikbaar zijn.

**Interne werking**  
De interne werking van de middleware is in het onderstaande diagram weergegeven, dit diagram wordt nog verder toegelicht.



Figuur 2: Interne werking middleware

* Services worden geregistreerd bij de middleware met behulp van een Configuration tool (*Brooker pattern*)
  + Bij het registreren vraagt de middleware welke data en functionaliteit een service bied, deze informatie wordt opgeslagen in de Common datamodel manager.
  + In de service pool wordt een proxy van de geregistreerde service bijgehouden.  
    Met deze proxy kan de service worden aangeroepen waar deze ook draait.
* De services kunnen aan de middleware vragen om functionaliteit en data die door de geregistreerde services geboden wordt. Als de gevraagde functionaliteit of data niet beschikbaar is geeft de middleware een exception terug.
  + De service vraagt data of functionaliteit aan bij de Zorg API, deze request wordt afgehandeld door de request processor.
  + De request processor vraagt aan het Common datamodel management in welke services (of waar) de data of functionaliteit wordt aangeboden.
  + In het Common datamodel is per functionaliteit of data ook bekend welke ‘permissions’ (toestemmingen) er voor nodig zijn.
* De Sessionmanger verteld de Request processor of een gebruiker ingelogd is en welke permissions hij/zij heeft.
* De communicatie tussen de services en de middleware verloopt *synchroon*

**Externe werking**  
Zoals in de Figuur 1 weergegeven wordt de middleware door de verschillende services als een blackbox behandeld. Deze blackbox biedt de services twee interfaces, namelijk:

**Register server**

|  |  |
| --- | --- |
| Type | Webservice |
| Inputtype | Methode Call |
| Input | String serviceName IServiceAPIProxy serviceProxy |
| Inputtype  Service definition | XML |
| Input  Service definition | <service id='\*ServiceDBID\*' name='\*ServiceName\*' version='\*ServiceVersion\*' listed='\*boolean\*'  humanReadableName='\*Service name\*' url='\*ServiceUrl\*' >  <function name='\*FunctionName\*' requiresUser='\*boolean\*' >  <parameter name='\*ParameterName\*' type='\*ParameterTypeFullName\*' />  /\* More parameter definitions may be passed \*/  <return type='\*ReturnTypeFullName\*' name='\*ReturnName\*' />  </function>  /\* More function definitions may be passed \*/  </service> |
| Output | Boolean succes |
| Opmerkingen | Service definition is fetched via the serviceProxy passed to registerService. The ServiceName is then set by the commondatamodel. |

**Zorg API**

|  |  |
| --- | --- |
| Type | Webservice |
| Inputtype | XML |
| Input | <request>  <function service='\*ServiceName\*' name='\*MethodName\*' requiresUser='\*boolean\*'>  <parameter required='\*BOOLRequired\*' name='\*ParameterName\*' type='\*ParameterTypeFullName\*'>  \*SerializedValueObject\*  </parameter>  /\*Parameters are passed in order to the method\*/  <return type='\*ReturnTypeFullName\*' name='\*ReturnName\*' />  </function>  /\*More functionCalls can be made\*/  </request> |
| Output | <result>  <function name='\*MethodName\*' requiresUser='\*boolean\*' >  <return type='\*ReturnTypeFullName\*' name='\*ReturnName\*'>  \*SerializedValueObject\*  </return>  </function>  /\*More function return can be available\*/  </result> |
| Opmerkingen | Generic API heeft de zelfde input en output. De XML wordt samengesteld door de middleware. |

## Losse services

* De services die informatie aanbieden (*information supplier*) en functionaliteiten bieden moeten een definitie van die informatie en functionaliteiten aanbieden via hun API.

## Het zorgportaal

Het zorgportaal wordt als LIPU plugin ontwikkeld, meer informatie over LIPU is terug te vinden in de Deployment view. Het zorgportaal zal de middleware als model gebruiken en zelf de controllers en views implementeren.

Figuur 3: MVC situatie LIPU plugin



# Information viewpoint

In de Information viewpoint staat beschreven met welke informatie het Zorgportaal systeem werkt. Binnen de systemen van Pharmeon is er meer informatie beschikbaar dan in deze viewpoint beschreven wordt, alleen wat binnen de scope van het Zorgportaal systeem valt wordt beschreven.

Om de diagrammen leesbaar te houden is bepaalde informatie weggelaten:

* Bij tabellen met meer dan 10 columns zijn de overige columns weggelaten. Belangrijke columns worden wel altijd weergegeven.
* Onbelangrijke referenties worden weggelaten.
* Tabellen met extra informatie die geen impact hebben op de globale informatiestructuur worden weggelaten.
* In de beschrijvingen bij de diagrammen wordt een object de eerste keer beschreven als objectnaam (Bij behorende tabel). Hierna zal de objectnaam gebruikt worden als referentie naar de tabel.

## Shared tables storage

In deze paragraaf wordt de informatiestructuur beschreven waar meerdere services gebruik van maken.

### Sites, gebruikers en services

Het onderstaande diagram beschrijft welke elementen er bij een site (tblSites) horen en belangrijk zijn binnen de scope van het Zorgportaal systeem. Deze informatie is vooral belangrijk voor het inloggen van gebruikers en bij houden van gebruikers bij verschillende services op verschillende sites.



Figuur 1: Shared tables

Bij een site horen de volgende elementen:

* Service (tblSer)
  + Een service draait op site(s)
* Gebruikersgroep (tblUserGroup)
  + Een gebruikersgroep hoort bij site(s)
* Gebruiker (tblUser)
  + Een gebruiker hoort bij een gebruikersgroep(en)
  + Een gebruiker maakt gebruik van service(s)

### Bestanden en bestandservers

De onderstaande diagram beschrijft de informatiestructuur die geuploade bestanden beheert.



Bij geuploade bestanden horen de volgende elementen:

* Een bestandserver (tblFilesServer)  
  Een server waar bestanden opgeslagen staan en naartoe geupload kunnen worden.
* Een bestand (tblFilesUpload)
  + Een bestand staat op een bestandserver.
  + Een bestand heeft een bestandstype (tblFileTypes).

## SSIM omgeving storage

Het volgende diagram beschrijft belangrijke informatie bij het authenticeren in de SSIM omgeving.

## 

De SSIM omgeving authenticeert met de volgende elementen:

* Een account (tblAccounts)
  + Een account wordt geauthenticeert op basis van een SMS (MobileNumber) of e-mail (EmailAddress).
  + Een geauthenticeerde account hoort bij een sessie (tblSessions).
* Een gebruiker (tblUsers)
  + Een gebruiker in de SSIM omgeving bevat de UserID van een gebruiker buiten de SSIM omgeving.
  + Een gebruiker hoort bij een account.

## eConsult service storage

Deze paragraaf beschrijft voor het Zorgportaal belangrijke informatiestructuren in de EConsult service.



Figuur 2: EConsult service

Bij de EConsult services horen de elementen:

* Een bericht (tblEC\_Message)
  + Een bericht is ingestuurd door een gebruiker (tblUser.
  + Een bericht wordt behandeld door een zorgverlener (tblEC\_Employee).
  + Een bericht behoort tot een reeks berichten (tblEC\_SOEP)
* Een zorgverlener (tblEC\_Employee).
  + Een zorgverlener hoort bij een instantie van de service.
  + Een zorgverlener behandelt een bericht.

Binnen de eConsult services kunnen bij berichten ook bijlagen (tblEC\_Attachment) hebben. In het onderstaande diagram staat de informatiestructuur voor deze bijlagen weergegeven.



Bij bijlagen horen de volgende elementen:

* Een bijlage (tlbEC\_Attachment)
  + Een bijlage hoort bij bericht(en)
* Een formulier (tblEC\_Form)
  + Een formulier hoort bij een bijlage.
  + Een bestand hoort bij een bijlage

## Herhaalrecept service storage

De data over patiënten voor de herhaalrecept service wordt opgeslagen in het AIS (Apotheek Informatie Systeem). Het AIS wordt aangeroepen via de SSIM omgeving, dit wordt verder toegelicht in de Deployment Viewpoint.

## Pilservice storage

Deze paragraaf beschrijft voor het Zorgportaal belangrijke informatiestructuren in de Pilservice.



Figuur 4: Pil service informatie structuur

Bij de Pilservices horen de elementen:

* Een gebruiker (tblUser)
* Een voorschrift (tblMed\_Ser\_Prescriptions)
  + Een voorschrift hoort bij gebruiker(s)
* Een voorhistorie (tblMed\_Ser\_PreHistory)
  + Een voorhistorie hoort bij gebruiker(s)
* Een product (tblMed\_Ser\_Prod)
  + Een product hoort bij voorhistorie(s)
  + Een product hoort bij voorschrift (s)

## Webagenda storage

Deze paragraaf beschrijft voor het Zorgportaal belangrijke informatiestructuren in de webagenda.

## 

Figuur 5: Webagenda informatie structuur

Bij de webagenda service horen de elementen:

* Een agenda (tblAgenda)
* Een afspraak (tblAppointment)
  + Bij een afspraak hoort een gebruiker geauthenticeert via de SSIM omgeving (met behulp van extUserId, een foreignkey naar een gebruiker in de SSIM omgeving)
  + Een afspraak kan verplaatst worden en verwijst in dit geval naar zichzelf.
  + Een afspraak valt in een timeslot.
* Een timeslot (tblTimeSlot)
  + Een timeslot heeft een tijd en datum (tblTimeSlotRange).

## Information Flow

Het aanmaken, verwijderen, aanvragen en wijzigen van de data binnen de verschillende services is de verantwoordelijkheid van de services die bij de data horen. De business logic die binnen deze services gedefinieerd is moet altijd gebruikt worden bij het ophalen van de data.  
De reden hiervoor is dat deze business logic getest is en er voor zorgt dat er alleen valide data in de databases te recht komt.



Figuur 6: Voorbeeld van een gewenste en ongewenste aanvraag van data

## Information ownership

De business logic binnen de verschillende services is verantwoordelijk voor het bepalen wie de data owner is.

“The data owner always has the correct value for that data and can act as the final arbiter where any data disputes occur.”

Bron: ‘SOFTWARE SYSTEMS ARCHITECTURE’   
- NICK ROZANSKI & EOIN WOODS

De zorgportaal beschouwt de business logic van de services als een black-box, voor de zorgportaal is de service zelf dus de data owner.

# Deployment Viewpoint

## Live servers infrastructuur

De systemen van Pharmeon zijn gedistribueerd uitgerold over een aantal servers. In het onderstaande diagram is deze situatie weergegeven. Het zorgportaal is onderdeel van de Pharmeon webapplication.



Figuur 7: Infrastructuur van de live servers

Elementen:

* Database server  
  Hier draaien databases voor de services, CMSsen en overige webapplicaties.  
  De DBMS die gebruikt wordt is Microsoft SQL Server 2008
* SSIM omgeving  
  Gevoelige patiënten gegevens worden opgeslagen binnen de SSIM omgeving. Deze omgeving heeft extra Authenticatie en Autorisatie nodig voordat het gegevens vrijgeeft. De SSIM omgeving wordt nog verder toegelicht.
* Webservers  
  De webapplicatie die verantwoordelijk is voor het draaien van website instanties draait op meerdere webservers in verband met Load Balancing.  
  De webservers draaien .NET, ASP(classic) en ASPX.
* Website instanties  
  De gebruiker maakt verbindingen met de webapplicatie van Pharmeon en deze webapplicatie presenteert de website instantie aan de gebruiker.

## SSSIM omgeving infrastructuur

Gevoelige patiënten gegevens worden opgeslagen binnen de SSSIM omgeving. Deze omgeving heeft extra Authenticatie en Autorisatie nodig voordat het gegevens vrijgeeft. Deze authenticatie en autorisatie wordt geregeld via de Authentication/authorisation server. Deze server stuur via SMS of email een sessiecode naar de gebruiker waarmee de gebruiker zich kan Authenticeren.

Figuur 8: SSIM omgeving architectuur

De gegevens binnen de SSSIM omgeving staan opgeslagen in Huisartsen Informatie Systemen(HIS), apotheken Apotheek Informatie Systemen (AIS). Deze services zijn apotheek en huisarts services die niet ontwikkeld zijn door Pharmeon en deze worden door veel apothekers en huisartsen gebruikt.

## LIPU

Lipu is een webservice die Pharmeon gebruikt om de link tussen ASP en .NET te leggen.  
Omdat alle al uitgerolde sites draaien op ASP (classic) moeten nieuwe toevoegingen ook in ASP kunnen draaien. LIPU zorgt ervoor dat er geprogrammeerd kan worden in C# .NET, de html hier gegenereerd wordt kan via LIPU op een ASP pagina geprint worden.



Figuur 9: LIPU

Het zorgportaal wordt als een LIPU plugin ontwikkeld. Deze zorgportaal LIPU plugin gebruikt de middleware als model en implementeert zelf de controllers en views van de zorgportaal (MVC).



Figuur 10: LIPU en het zorgportaal

## Acceptatie en Developement servers

Pharmeon draait naast de live servers ook een Acceptatie en een Developement server. Deze servers draaien de zelfde webapplicaties en databases als de live server. In het geval van de Developement en Acceptatie servers draaien de databases en de webapplicaties op de zelfde fysieke systemen.

# The availability and resilience perspective

## Voor requests via de middleware

* Voor het ophalen van data geldt "at least once"  
  Dit houdt in dat bij een fout bij het ophalen van de gevraagde data, de data opnieuw opgehaald wordt.  
  De server (services) die de data aanbied kan in gevallen waar fouten zich voordoen de request meerdere malen behandelen.  
  Het maximaal aantaal keer dat de request opnieuw behandeld wordt moet nog vastgesteld worden door middel van strestests.  
  Als die maximum overschreden wordt krijgt de eindgebruiker een error message te zien.
* Voor het wegschrijven of updaten van data geldt "at most once"  
  Dit houdt in dat bij een fout bij het wegschrijven of updaten van data, de bewerking niet opnieuw wordt uitgevoerd.  
  Alle veranderingen die al zijn uitgevoerd voor de bewerking moeten in het geval van een fout terug gedraaid moeten worden.  
  De eindgebruiker krijgt in het geval van een fout een error message te zien.

**Formulieren**

## Herbruikbare formulieren voor Pharmeon

*Bob Brinks  
22 december 2011  
Amsterdam*

Inhoud

[Herbruikbare formulieren voor Pharmeon 1](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321511)

[Inleiding 2](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321512)

[Waarom herbruikbare formulieren? 2](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321513)

[De onderdelen van de herbruikbare formulieren module 4](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321514)

[Gebruikmaken van de herbruikbare form module 6](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321515)

[Het form definieren 6](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321516)

[Het form weergegeven 8](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321517)

[Validatie van formulieren 9](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321518)

[Formulier afhandelen 10](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321519)

[Standaard afhandelingen 12](file:///C:\Users\Bob-thuis\Documents\Formulieren.docx#_Toc312321520)

# Inleiding

Dit document is een handleiding voor het gebruik van de herberuikbare formulieren ‘module’. Deze handleiding is bedoeld voor programeurs die webformulieren gebruiken.

Als je alleen wilt weten hoe de module werkt kan je de rest van de inleiding overslaan.

## Waarom herbruikbare formulieren?

Webformulieren binnen Pharmeon worden in het algemeen per stuk ontwikkeld. Per form worden de volgende delen ontwikkeld:

* De html wordt in een XSL template geschreven. Dit gebeurd wel volgens standaarden maar er lopen meerdere standaarden door elkaar heen (in tables of in lists bijvoorbeeld). Bij de html wordt ook CSS ontwikkeld, deze kan vaak herbruikt worden van eerdere forms.  
  **Problemen**
  + Als zo’n ‘standaard’ veranderd zou worden moet per form (en dat zijn er veel 100+) de template aangepast worden.
  + Het schrijven van de templates is veel knip en plak werk en lijdt daarom tot dubbele code.
  + De geschreven template is natuurlijk nog niet getest en dus moet per template getest (integratie tests) worden. Dit kan tijd rovend zijn.
* De clientside validatie wordt gedaan met jQuery validation. De configuratie voor deze jQuery plugin wordt per form geschreven.  
  **Problemen**
  + De geschreven configuratie is niet getest en moet daarom per formulier getest (integratie tests) worden.
  + Als de jQuery validation plugin door een update veranderd is het voor Pharmeon moeilijk deze nieuwe versie ook te gaan gebruiken. Het is namelijk mogelijk de configuratie die per form geschreven is niet werkt met de nieuwere versie. In dat geval zal voor elke form de configuratie bewerkt en opnieuw getest moeten worden, dit is te tijdrovend en er zal gekozen worden om geen update naar de nieuwe versie te doen.
  + **Mogelijkheid**De berichten die door de jQuery validation worden weergegeven zijn geschreven in 1 taal (meestal nederlands soms engels). Door clientside validatie beter te regelen is het makkelijker om localisatie (sites in meerdere/andere talen) of de formaliteit (u vorm of jij vorm) van de berichten in te voeren.
* De serverside validatie wordt per form geschreven en is vaak onderdeel van de afhandeling van een form.  
  **Problemen**
  + Omdat serverside validatie per form schrijven tijdrovend is (bij het ontbreken van een gestandaardiseerde manier), wordt er voor gekozen om alleen de belangrijkste waardes serverside nog te controleren. Dit kan mogelijk zorgen voor security issues.
  + Er ligt waarschijnlijk te veel verantwoordelijkheid bij clientside validatie.
  + **Mogelijkheid**Clientside en serverside configuratie voor validatie op de zelfde plek configureren zodat er geen dubbel werk gedaan wordt.
* Afhandeling van formulieren wordt per formulier geschreven.  
  **Problemen**
  + Veel formulieren heb de zelfde afhandeling van de ingevoerde data, bijvoorbeeld het versturen van de mail of het vullen van een database met de ingevoerde data. Deze afhandelingen worden nu per form geschreven en getest. Hier zit ook veel knip en plak werk bij wat weer lijdt tot dubbele code (slecht voor de onderhoudbaarheid).
  + **Mogelijkheid**Door voor de veel voorkomende afhandelingen van formulieren 1 keer te schrijven (gebruikmakend van de herbruikbare formulieren module) kan veel tijd bespaard worden met implementatie en testen.
* **Samengevatte problemen**
  + Veel dubbele code en dus verloren tijd.
  + Veel opnieuw testen per formulier en dus verloren tijd.
  + Geen hergebruik van ge-unittestte herbruikbare code.
  + Geen standaardisatie waardoor veranderingen in de wensen of de omgeving een grote impact hebben (er moet op veel plaatsen code aangepast worden).

Met de herbruikbare formulieren wordt geprobeerd de bovengenoemde problemen op te lossen.

# De onderdelen van de herbruikbare formulieren module

De module is een c# dll die geinclude kan worden in een project (dsfkljsdf) in combinatie met een XSL template (sdklfsdlfjksdf).  
In deze dll zijn de volgende classes/interfaces beschikbaar:

* CForm  
  Een dataholder waar de elementen (<input>,<textarea> e.d.) en layout van 1 formulier in staan.  
  Deze classe genereerd ook de XML van het formulier die door de XSL template tot html gemaakt wordt.
* AFormElement (abstract)  
  Implementaties van deze abstracte class kunnen aan een object van CForm toegevoegd worden. Elke implementatie beschrijft een element van een formulier.  
  **Implementaties:**
  + CFieldset  
    Deze implementatie start een nieuwe groep van input elementen. Een CForm object moet altijd als eerste een CFieldset object binnen krijgen.
  + CInput  
    Deze implementatie voegt een input (<input />) aan een form toe. Zo’n input kan van de volgende types zijn.
    - Text
    - Password
    - Textarea
    - Submit
    - Button
    - Hidden
    - Checkbox
  + CInputSelect  
    Een subclass van CInput voor select elementen (<select />). Deze input kan van de volgende types zijn:
    - Dropdown
    - Radio (voor een groep van radiobuttons)
  + CInputComposed  
    Deze implementatie maakt het mogelijk meerdere inputs te combineren (voor deze implementatie betekend dit dat ze op de zelfde regel in het formulier verschijnen). Als je een samengestelde waarde uit meerdere inputs wilt hebben moet je een subclasse van deze implementatie aanmaken. Een voorbeeld van zo’n subclasse zou een datum met 3 dropdowns kunnen zijn waarbij de waardes van de 3 dropdowns tot 1 datum samen gevoegd worden.
  + CFormLayout  
    Deze implementatie maakt geen input aan maar voegt het meegegeven html element toe, voorbeelden: <p>, <h1>, <h2>, <span>. Het doel is om extra informatie in de forms te kunnen toevoegen.
* IValidateInput (Interface)  
  Implementaties van deze interface zijn verantwoordelijk voor de validatie van de gebruikers input. Elke implementatie zou (should) client en serverside validatie moeten uitvoeren. Een validatie hoort bij 1 CInput (of een subclass).  
  **Implementaties**
  + CValidateRequired  
    Maakt een veld verplicht om in te voeren.
  + CValidateEmail  
    Maakt een veld verplicht en checked of er een valide email adres wordt ingevoerd.
  + CValidateDate  
    Maakt een veld verplicht en checked of er een valide datum wordt ingevoerd.
* Meer implementaties en subclasses zullen waarschijnlijk nog volgen, heb jij er 1 geschreven of bedacht, voeg hem dan toe aan de module (solution) van de herbruikbare forms zodat andere mensen e rook gebruik van kunnen maken.  
  **Eisen**
  + Should be Unit tested.
  + Should be as generic as possible (for higher reusability)

# Gebruikmaken van de herbruikbare form module

Een beschrijving van de normal gebruik van de form module. Op andere manieren gebruik maken van de form module kan ook maar kan mogelijk de problemen beschreven in de inleiding weer terug brengen.

## Het form definieren

Het aanmaken van de form gebeurt in C#. De form zou aangemaakt (gedefinieerd) moeten worden (should) in zijn eigen method (voorbeeldForm).

public CForm voorbeeldForm()

{

CForm vbForm = new CForm(LIPU\_Handler, "vbForm");

return vbForm;

}

public CForm voorbeeldForm()

{

CForm vbForm = new CForm(LIPU\_Handler, "vbForm");

return vbForm;

}

In deze method moeten ook de elementen (fieldsets,input,layout) van het formulier worden toegevoegd. Als eerste moet een fieldset toegevoegd worden.

public CForm voorbeeldForm()

{

CForm vbForm = new CForm(LIPU\_Handler, "vbForm");

vbForm.addElement(new CFieldSet("Voorbeeld fieldset"));

vbForm.addElement(new CInput("text", "vbName", "vbLabel"), new CValidateRequired());

return vbForm;

}

Als laatste wordt in de form method de finalize() method aangeroepen deze method zorgt ervoor dat ‘geposte’ waardes in het CForm object worden geladen (van uit lipu).

public CForm voorbeeldForm()

{

CForm vbForm = new CForm(LIPU\_Handler, "vbForm");

vbForm.addElement(new CFieldSet("Voorbeeld fieldset"));

vbForm.addElement(new CInput("text", "vbName", "vbLabel"), new CValidateRequired());

…meer elementen toevoegen als dat nodig is…

regForm.finalize();

return vbForm;

}

De basis van de formulier defenitie is nu klaar.

Aan de verschillende elementen kunnen nog extra waardes meegegeven worden met verschillende doelen.

**CSS classes**

CInput voorbeeldInput = new CInput("text", "vbName", "vbLabel");

voorbeeldInput.addClass("voorbeeldCSSClass");

vbForm.addElement(voorbeeldInput, new CValidateRequired());

**Attributes**

CInput voorbeeldInput = new CInput("text", "vbName", "vbLabel");

voorbeeldInput.setAttribute("onclick", "vbJavaScriptFunctie()");

vbForm.addElement(voorbeeldInput, new CValidateRequired());

**Waardes voor select elementen**

CInputSelect geslachtDropdown = new CInputSelect("radio", "geslacht", "Geslacht (\*)");

geslachtDropdown.addValue("man", "Man");

geslachtDropdown.addValue("vrouw", "Vrouw");

vbForm.addElement(geslachtDropdown, new CValidateRequired());

Het definieren van het formulier is nu klaar.

## Het form weergegeven

Nadat het form is gedefinieerd kan het omgezet worden naar XML zodat het weergegeven kan worden via een XSL Template handler.

CForm voorbeeldForm = voorbeeldForm();

XmlNode voorbeeldFormXML = voorbeeldForm.getXML();

// De xml toevoegen aan de XML die aan de template handler wordt meegegeven.

pageXML.XmlDoc.DocumentElement.AppendChild(pageXML.XmlDoc.ImportNode(voorbeeldFormXML, true));

In de template (XSL) die voor het weergeven van de pagina gebruikt wordt (99% van de tijd is dat main.tpl.html). Moet de juiste forms template geinclude worden. Op het moment van schrijven zijn er twee forms templates beschikbaar: 1 met lists (ol) form-li.tpl.html en 1 met tables form-table.tpl.html.

De template voor forms moet dan nog aangeroepen worden met de form XML erbij.

<!--<LINKTEMPLATE id="form-table" template="\*"></LINKTEMPLATE>-->

…  
<xsl:call-template name="form">  
<xsl:with-param name="from" select="$lipu/PARAMS/PARAM[NAME='xmlparam']/VALUE/PAGE/FORM"/>  
</xsl:call-template>  
…

De form wordt nu weergegeven.

## Validatie van formulieren

In de defenitie van het formulier is het mogelijk validaties toe te voegen aan de verschillende inputs.  
Dit gebeurt door een implementatie van IValidateInput mee te geven aan de addElement method. In het onderstaande voorbeeld wordt een CValidateRequired meegegeven.

public CForm voorbeeldForm()

{

CForm vbForm = new CForm(LIPU\_Handler, "vbForm");

vbForm.addElement(new CFieldSet("Voorbeeld fieldset"));

vbForm.addElement(new CInput("text", "vbName", "vbLabel"), new CValidateRequired());

…meer elementen toevoegen als dat nodig is…

regForm.finalize();

return vbForm;

}

Door een validatie object mee te geven wordt bij het weergeven van het formulier automatisch clientside validatie code toegevoegd voor jQuery validate. In de meeste gevallen gebeurt dit door het toevoegen van een CSS class aan het element.

Het aanroepen van serverside validatie wordt besproken in het Formulier afhandelen hoofdstuk.

## Formulier afhandelen

Na dat een gebruiker het formulier heeft ingevuld en verstuurd kan de CForms class gebruikt worden voor het afhandelen van de ingevoerde data.

Hiervoor kan je aan het CForms object vragen of de gebruiker het formulier verstuurd heeft.

CForm voorbeeldForm = voorbeeldForm();

if (voorbeeldForm.submitted)

{

// Formulier is verstuurd

}

else

{

// Formulier is niet verstuurd  
XmlNode voorbeeldForm = voorbeeldForm.getXML();  
pageXML.XmlDoc.DocumentElement.AppendChild(pageXML.XmlDoc.ImportNode(voorbeeldForm, true));

}

**Serverside validatie**Als het formulier verstuurd is kan de serverside validatie aangeroepen worden. Deze validatie maakt een lijst van berichten (messages) van incorrect ingevulde velden. Net zoals de clientside validatie is de serverside validatie bij de form defenitie toegevoegd.

CForm voorbeeldForm = voorbeeldForm();

if (voorbeeldForm.submitted)

{

// Formulier is verstuurd

List<string> messages;

if (!regForm.validate(out messages))

{

// Formulier is niet valid

// Mogelijk messages weergeven aan de gebruiker

// Voor het weergeven van messages is nog geen standard template mogelijk wordt dit nog toegevoegd.

}

else

{

// Formulier is valid

// Formulier afhandelen

}

}

**Form afhandelen**

Uiteindelijk kunnen de gevalideerde gegevens uit het formulier gebruikt worden. Dit gaat op de volgende wijze.

CForm voorbeeldForm = voorbeeldForm();

if (voorbeeldForm.submitted)

{

// Formulier is verstuurd

List<string> messages;

if (!regForm.validate(out messages))

{

// Formulier is niet valid

}

else

{

// Formulier is valid

string vbWaarde = cform.getInput("vbInput").value;

stuurNaarDeDatabaseOID(vbWaarde);

}

}

## Standaard afhandelingen

Door het gebruik van de CForms class is het mogelijk gestandaardiseerde afhandeling van formulieren te implementeren.

In dit hoofdstuk wordt een voorbeeld van zo’n afhandeling in pseudo code beschreven. Dit voorbeeld mailt het resultaat van het formulier naar een email adres.

Class emailFormHandler  
{

Public void emailForm(string emailAdres, CForm form)

{

String waardes = “”;  
foreach(CInput waarde in form.inputList)

{

waardes += waarde.name + “ = “+ waarde.value

}

// mogelijk waardes nog door een xsl template halen om er een mooi mail van te maken.

mail(emailadres, waardes);

}

}

Standaard afhandelingen zouden in de forms module (de dll) opgenomen moeten worden zodat iedereen er gebruik van kan maken, dit voorkomt dat er dubbel werk gedaan wordt.

Userstories

Ontwikkelen zorgportaal

Pharmeon  
Bob Brinks

1. **Overzicht van informatie uit verschillende services in het zorgportaal**

|  |  |
| --- | --- |
| Als een | Patient |
| Wil ik | Een overzicht van de services die een website aanbied kunnen zien |
| Zodat | Ik weet wat de website mij biedt op het gebied van services |
| Storyotype | New Functionality |
| Story points | 13 |

*Acceptation criteria’s:*

* Alleen services die geactiveerd zijn
* Alleen de volgende services: herhaalrecept, Pil, Diabetes, E-consult, Digitaal medisch dossier en webagenda
* Services worden toegevoegd via een XML editor

1. **Bij het inschrijven bij een zorgverlener de mogelijkheid bieden om een online account aan te maken (registreren).**

|  |  |
| --- | --- |
| Als een | Ongeregistreerde patient op een website van Pharmeon |
| Wil ik | Bij het inschrijven bij een zorgverlener meteen de mogelijkheid hebben om ook een online account aan te maken. |
| Zodat | Ik geen tijd verspil met het opnieuw invoeren van reeds bekende gegevens bij het registreren zoals dat nu gebeurt. |
| Storyotype | New Functionality |
| Story points | 20 |

*Acceptation criteria’s:*

* Bij het inschrijf formulier kan de patient kiezen om zich ook een online account aan te maken (registreren).
* De patient hoeft bij het registreren niet opnieuw bekende informatie in te voeren.

1. **E**é**nmalig registreren per site instantie**

|  |  |
| --- | --- |
| Als een | Registreerde patient die gebruik wilt maken van een website en de beschikbare services. |
| Wil ik | Eénmaal registreren op de website. |
| Zodat | Ik bij het switchen van services niet opnieuw hoef in te registreren. |
| Storyotype | New Functionality |
| Story points | 40 |

*Acceptation criteria’s:*

* Een patient kan na één keer registreren gebruik maken van alle services zonder opnieuw te registreren.
* De consumentengegevens van de patient (gebruiker) worden op één plek per site instantie bijgehouden.

1. **E**é**nmalig inloggen per site instantie via een inlogknop**

|  |  |
| --- | --- |
| Als een | Geregistreerde patient (gebruiker) gebruik wilt maken van een website en de beschikbare services. |
| Wil ik | Eénmaal inloggen op de website. |
| Zodat | Ik bij het switchen van services niet opnieuw hoef in te loggen. |
| Storyotype | New Functionality |
| Story points | 40 |

*Acceptation criteria’s:*

* Een patient kan na éénmalig inloggen gebruik maken van alle services zonder opnieuw in te loggen
* Het gebruik van sessiecodes valt buiten de scope van deze story

1. **E**é**nmalig inloggen per site instantie via een service**

|  |  |
| --- | --- |
| Als een | Geregistreerde patient (gebruiker) gebruik wilt maken van een website en de beschikbare services. |
| Wil ik | Eénmaal inloggen op de website. |
| Zodat | Ik bij het switchen van services niet opnieuw hoef in te loggen. |
| Storyotype | New Functionality |
| Story points | 20 |

*Acceptation criteria’s:*

* Een patient kan na éénmalig inloggen gebruik maken van alle services zonder opnieuw in te loggen
* Het gebruik van sessiecodes valt buiten de scope van deze story

1. **Aanmelden voor services via de zorgportaal**

|  |  |
| --- | --- |
| Als een | Geregistreerde en ingelogte patient in het zorgportaal. |
| Wil ik | Mij via de zorgportaal kunnen aanmelden bij services. |
| Zodat | Zodat ik niet op de website hoef te zoeken naar de verschillende services die de website aanbied. |
| Storyotype | New Functionality |
| Story points | 20 |

*Acceptation criteria’s:*

* Een patient kan zich in het overzicht dat het zorgportaal vertoont aanmelden voor services.
* Een patient ziet na het aanmelden bij een service de nieuwe status van de service in het overzicht dat het zorgportaal vertoont.

1. **Weergeven van een lijst van gebruikte services**

|  |  |
| --- | --- |
| Als een | Geregistreerde en ingelogte patient (gebruiker) in het zorgportaal. |
| Wil ik | Een overzicht kunnen zien van de services waar ik gebruik van maak. |
| Zodat | “ “ |
| Storyotype | New Functionality |
| Story points | 20 |

*Acceptation criteria’s:*

* Een patient (gebruiker) kan een lijst van de door hem gebruikte services inzien.
* Een patient kan vanuit de lijst navigeren naar de verschillende services.
* In de lijst staan: de naam, een link en status van de services.

1. **Overzicht van informatie uit verschillende services in het zorgportaal**

|  |  |
| --- | --- |
| Als een | Geregistreerde en ingelogte patient in het zorgportaal. |
| Wil ik | Een overzicht van de status van de services die ik gebruik. |
| Zodat | Ik in één oogopslag kan zien of ik actie moet ondernemen in één of meerdere van de services.  Ik een duidelijker overzicht heb welke services de website aanbied. |
| Storyotype | Complex Epic |
| Story points | 40 |

*Acceptation criteria’s:*

* Een patient kan een overzicht zien van meldingen (notifications) uit de verschillende services die hij gebruikt.

1. **Met sessiecode inloggen**

|  |  |
| --- | --- |
| Als een | Geregistreerde patient (gebruiker) gebruik wilt maken van een website en de beschikbare services. |
| Wil ik | Zo min mogelijk sessiecodes invoeren |
| Zodat | Ik bij het switchen van services minder tijd kwijt ben door het invoeren van sessiecodes |
| Storyotype | New Functionality |
| Story points | 40 |

*Acceptation criteria’s:*

* Alleen als er geen actieve sessiecode beschikbaar is wordt om er een nieuwe sessiecode gevraagt.

1. **Overige formulieren aanvullen met bekende informatie**

|  |  |
| --- | --- |
| Als een | Ingelogte Patient (gebruiker) op een website van Pharmeon |
| Wil ik | Gegevens die ik eerder in het system heb ingevoerd niet opnieuw invullen in webformulieren. |
| Zodat | Ik geen tijd verspil met het opnieuw invoeren van reeds bekende gegevens |
| Storyotype | New Functionality |
| Story points | 13 |

*Acceptation criteria’s:*

* Bij het laden van webformulieren worden bekende gegevens automatisch ingevoerd.
* Als de gebruiker de automatisch ingevoerde informatie aanpast krijgt hij hierover een melding bij het versturen van het formulier. Hij heeft dan de keuze om deze aanpassing in het system op te slaan of alleen voor de handeling van het formulier te gebruiken.

1. **Terug naar zorgportaal link**

|  |  |
| --- | --- |
| Als een | Patient (gebruiker) op een website van Pharmeon |
| Wil ik | Vanuit een webformulier of andere UI van een services terug kunnen naar het zorgportaal overzicht. |
| Zodat | Ik na het versturen van het webformulier of ander UI element niet ‘vast’ zit op een ‘versturen voltooit’ pagina. |
| Storyotype | User Interface Enhancement |
| Story points | 5 |

*Acceptation criteria’s:*

* De patient (gebruiker) kan altijd terug naar de overzicht pagina van het zorgportaal
* Als de patient (gebruiker) nog op een pagina is met onverstuurde/onopgeslagen gegevens wordt bij het verlaten van deze pagina een waarschuwing gegeven.

1. **Selecteren welke services het zorgportaal aanbied**

|  |  |
| --- | --- |
| Als een | Zorgverlener die een website bij Pharmeon heeft |
| Wil ik | Kunnen aanpassen welke services mijn website aanbied binnen de zorgportaal. |
| Zodat | Ik het toevoegen of verwijderen van een service niet via Pharmeon hoef te regelen, maar dit zelf kan regelen. |
| Storyotype | New Functionality |
| Story points | 20 |

*Acceptation criteria’s:*

* Een zorgverlener (beheerder) kan met een web UI services van zijn website verwijderen.
* Een zorgverlener (beheerder) kan met een web UI services aan zijn website toevoegen.

1. **Statistieken inzien**

|  |  |
| --- | --- |
| Als een | Beheerder op een website van Pharmeon |
| Wil ik | Statistieken over het gebruik van de verschillende services kunnen inzien |
| Zodat | Zodat ik een beeld kan krijgen welke services belangrijk zijn voor klanten |
| Storyotype | New functionality |
| Story points | 40 |

*Acceptation criteria’s:*

* Er worden statistieken over het gebruik van services opgeslagen (aantal requests, type requests, gefaalde en geslaagde requests).
* De statistieken die bijgehouden worden zijn op een manier in te zien.