

# Technisch ontwerp

---

**Project: Mobile versie Comtak optimaliseren**

**Naam: Randy Klaassen**

**Klas: ICIA-3E**

**Datum: 8-10-2013**

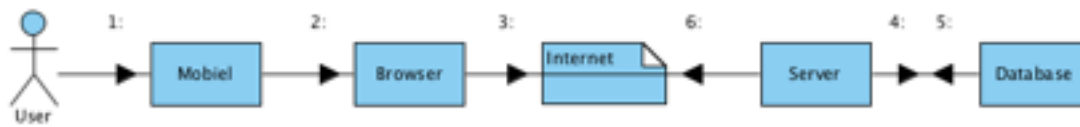
## Inhoudsopgave

Inleiding technisch ontwerp	3
Software architectuur	4
<i>C# ASPnet</i>	4
<i>MSSQL</i>	4
<i>CSS</i>	4
<i>Javascript</i>	4
<i>Prodata Framework</i>	4
Klassen analyse	5
Klassendiagram UML	6
Toelichting klassen	7
Toelichting methods/functions	8
<i>Main</i>	8
<i>CommonFunctions</i>	8
<i>Person</i>	8
<i>Address</i>	9
<i>PhoneNumber</i>	9
Presenteren ontwerp	10
Activitydiagrams (of Sequentie diagrams)	11

# **1. Inleiding technisch ontwerp**

Het hoofddoel van het project “Mobile versie Comtak optimaliseren” is het overzichtelijker maken van de mobiele versie van Comtak. Hierbij komt ook nog dat de applicatie daardoor direct sneller zal moeten worden. Dit wordt bijvoorbeeld behaald door bepaalde onderdelen van de navigatie in Comtak te verbergen voor de mobiele versie.

## 2. Software architectuur



Bovenstaand is kort de software architectuur van dit project uitgebeeld. De gebruiker opent op een mobiele omgeving een browser, die maakt via het Internet verbinding met de server, en haalt de benodigde gegevens op uit de database. Vervolgens worden deze gegevens via de server, door het Internet, weer terug gegeven en getoond op de browser.

Voor dit project zullen veel verschillende technieken toegepast moeten worden.

### C# ASP.net

Deze techniek zal voor het grootste gedeelte gebruikt worden voor dit project, vanuit deze techniek worden alle andere technieken toegepast. Hierin wordt de pagina gerenderd volgens OOP standaard.

### MSSQL

MSSQL wordt gebruikt voor de database waarin alle gegevens staan die in Comtak gebruikt worden. Ook in de mobiele versie wordt dit gebruikt om vanuit C# gegevens op te vragen om te tonen op de pagina.

### CSS

Deze techniek zal door C# aangeroepen worden om de styling van de pagina's die in Comtak weergegeven worden te verzorgen.

### Javascript

JavaScript wordt gebruikt om bepaalde animaties e.d. te verzorgen, in de desktop en in de mobiele versie van Comtak. Met name het menu dat in de mobiele versie wordt gebruikt maakt gebruik van JavaScript om het scherm op te "sliden".

### Prodata Framework

Programma's die gemaakt worden door Prodata zijn gebaseerd op het framework dat daarvoor al gebouwd is. Dit framework is gemaakt in C#, en is de basis voor Comtak, en dus ook de mobiele versie daarvan.

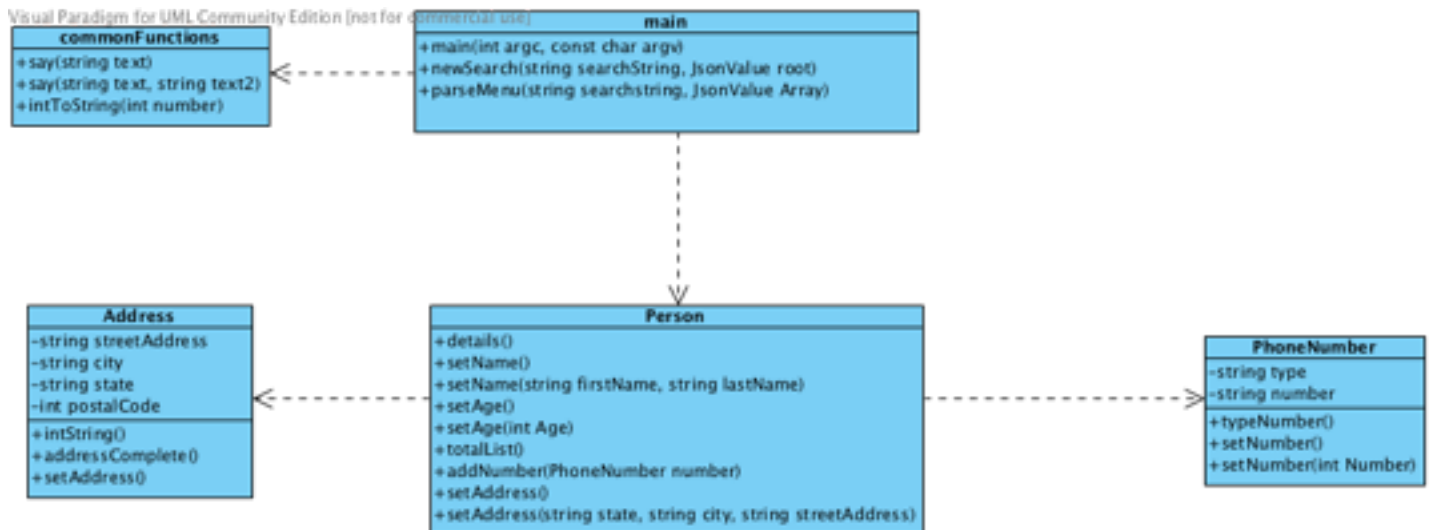
### **3. Klassen analyse**

Omdat dit project gebaseerd is op een al bestaand systeem, zijn de meeste klassen en dergelijke al bekend. Aangezien Comtak op een OO-gebaseerd systeem werkt zijn er ook een groot aantal klassen die in dit project niet gebruikt zullen worden.

In dit project zullen er veel aanpassingen gemaakt worden in de navigatie menu's en filters van gegevens in de mobiele versie van Comtak, dus zullen daarvan een aantal klassen gebruikt worden.

## 4. Klassendiagram UML

N.B. Het klassendiagram dat hieronder is getoond is van een vervangend project (JSON + C++). De hoeveelheid klassen die in het Comtak project van toepassing zijn is astronomisch hoog. Een klassendiagram met al die klassen zou niet eens op deze pagina passen, laat staan enig overzicht te bieden.



## 5. Toelichting klassen

*N.B. DE KLASSEN DIE GEBRUIKT ZIJN, ZIJN VAN EEN VERVANGEND PROJECT (JSON + C++). DE HOEEVEELHEID KLASSEN DIE IN HET COMTAK PROJECT VAN TOEPASSING ZIJN IS ASTRONOMISCH HOOG. OM AL DIE KLASSEN TOE TE LICHTEN ZOU EEN VEEL TE GROTE HOEEVEELHEID INFORMATIE ZIJN.*

**Adres (Address):** In deze klasse worden alle gegevens van een adres van een persoon opgeslagen. De straat, stad en gemeente worden hierin opgeslagen.

**TelefoonNummer (PhoneNumber):** In deze klassen worden alle telefoonnummers van een persoon opgeslagen. Deze telefoonnummers hebben een type en een nummer.

**Persoon (Person):** Hierin staan alle details van een persoon. Hierbij is te denken aan de voornaam, achternaam, leeftijd, alle telefoonnummers, en het adres van een persoon. Dit maakt dus gebruik van de vorige klassen om deze klas zo beknopt mogelijk te houden.

**Main:** Vanuit hier wordt het programma opgestart. Hierin is het menu verwerkt, en wordt dus bepaald wat het programma uit moet voeren.

**CommonFunctions:** Hierin worden een aantal veel gebruikte functies gedefinieerd. Deze worden vervolgens in de code gebruikt, en hoeven dus niet vaker opnieuw op andere plekken worden aangemaakt.

## 6. Toelichting methods/functions

*N.B. DE FUNCTIES DIE GEBRUIKT ZIJN, ZIJN VAN EEN VERVANGEND PROJECT (JSON + C++). DE HOEVEELHEID KLASSEN DIE IN HET COMTAK PROJECT VAN TOEPASSING ZIJN IS ASTRONOMISCH HOOG. OM AL DIE FUNCTIES TOE TE LICHTEN ZOU EEN VEEL TE GROTE HOEVEELHEID INFORMATIE ZIJN.*

### **Main**

main: Start het programma, laadt het JSON bestand en geeft dat vervolgens door aan andere functies.

parseMenu: Hiermee wordt het menu getoond en wordt de optie die gekozen wordt door de gebruiker verwerkt tot een actie. Deze roept dan vervolgens weer de benodigde functies aan.

### **CommonFunctions**

say: Print de aangegeven tekst op het scherm met de nodige formatting.

intToString: Zet de meegegeven integer om in een string. De string kan vervolgens in teksten worden gebruikt.

### **Person**

De constructors van Person zullen er voor zorgen dat het hele object gevuld wordt zodra het gemaakt wordt. De constructor die geen waardes mee gegeven krijgt, vraagt de gebruiker voor alle nodige details. De constructor die wel waardes mee krijgt, zet deze automatisch bij de gegevens van een persoon.

Verder zijn er nog setters voor ieder detail van een persoon, voor het geval de gebruiker deze apart wil veranderen. Deze zijn er in de vorm van een setter zonder waardes mee te geven, die om de details vraagt, en een setter die de meegegeven waardes gebruikt om de details in te vullen.

De functie “addNumber” voegt een telefoonnummer toe op basis van een object dat mee wordt gegeven. Dit moet hiervoor dus gemaakt worden in een andere functie.

Er is ook een functie die de hoeveelheid telefoonnummers van een persoon toont, genaamd totalList.



## **Address**

De constructors van Address werken op een zelfde manier als die van Person. Als er gegevens mee worden gegeven, worden die gebruikt om de adresgegevens in te vullen. Zo niet, worden alle gegevens apart opgevraagd.

Verder heeft Address functies om het hele adres in string vorm terug te geven, met de nodige formatting, in addressComplete.

Verder heeft Address ook nog setters voor alle gegevens die er in verwerkt zijn, voor het geval dat de gebruiker deze apart wil aanpassen.

## **PhoneNumber**

PhoneNumber heeft constructors die ongeveer hetzelfde werken als die van Address en Person. Een PhoneNumber kan leeg aangemaakt worden en vervolgens ingevuld, of meteen ingevuld worden door middel van de meegegeven informatie. Het meegeven van gegevens op deze manier kan dus bijvoorbeeld gebruikt worden om objecten te maken van de JSON die ingeladen wordt.

## **7. Presenteren ontwerp**

Het ontwerp zal direct gepresenteerd worden aan de opdrachtgever door middel van een gesprek. Vanaf daar zal het direct mogelijk zijn om waar nodig aanpassingen te maken in het ontwerp.

## 8. Activity diagrams (of Sequence diagrams)

Visual Paradigm for UML Community Edition (not for commercial use)

