**Projekt: VirtualCalendar**

**Software Architektur Spezifikation**

**(Software Architecture Document)**

[Dokumentstruktur basiert auf RUP „Software Architecture Document“]

# **1.Dokumentinformationen**

## **1.1 Änderungsgeschichte**

|  |  |  |  |
| --- | --- | --- | --- |
| *Datum* | *Version* | *Änderung* | *Autor* |
| 24.04.2017 | 1.0 | Erstellung | Kuczera |
| 04.05.2017 | 2.0 | Bearbeitung | Kuczera |
| 08.05.2017 | 3.0 | Bearbeitung | Kuczera |
|  |  |  |  |

## **1.2 Inhalt**

[1 Dokumentinformationen 2](l%20)

[1.1 Änderungsgeschichte 2](l%20)

[1.2 Inhalt 3](l%20)

[2 Einführung (Introduction) 4](l%20)

[2.1 Definitionen und Abkürzungen (Definitions, Acronyms, Abbreviations) 4](l%20)

[2.2 Referenzen (References) 4](l%20)

[2.3 Übersicht (Overview) 4](l%20)

[3 Architektonische Darstellung (Architectural Representation) 4](l%20)

[4 Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints) 4](l%20)

[5 logische Architektur (Logical View) 4](l%20)

[5.1 Übersicht (Overview) 4](l%20)

[5.2 Design Pakete (Architecturally Significant Design Packages) 4](l%20)

[5.2.1 Package GUI 4](l%20)

[5.2.1.1 Beschreibung des Package 4](l%20)

[5.2.1.2 Diagramme 4](l%20)

[5.2.1.3 Schnittstellen 4](l%20)

[5.2.1.4 Operationen 4](l%20)

[5.2.1.4.1 Interne Operation 1 4](l%20)

[5.2.1.4.2 Interne Operation 2 4](l%20)

[5.2.2 Package Problem-Domain 5](l%20)

[5.2.2.1 Beschreibung des Package 5](l%20)

[5.2.2.2 Diagramme 5](l%20)

[5.2.2.3 Schnittstellen 5](l%20)

[5.2.2.4 Operationen 5](l%20)

[5.2.2.4.1 Interne Operation 1 5](l%20)

[5.2.2.4.2 Interne Operation 2 5](l%20)

[5.2.3 Package Datenbanksystem 5](l%20)

[5.2.3.1 Beschreibung des Package 5](l%20)

[5.2.3.2 Diagramme 5](l%20)

[5.2.3.3 Schnittstellen 5](l%20)

[5.2.3.4 Operationen 5](l%20)

[5.2.3.4.1 Interne Operation 1 5](l%20)

[5.2.3.4.2 Interne Operation 2 5](l%20)

[5.2.4 Package Email-Benachrichtigungssytem 5](l%20)

[5.2.4.1 Beschreibung des Package 5](l%20)

[5.2.4.2 Diagramme 5](l%20)

[5.2.4.3 Schnittstellen 5](l%20)

[5.2.4.4 Operationen 5](l%20)

[5.2.4.4.1 Interne Operation 1 5](l%20)

[5.2.4.4.2 Interne Operation 2 5](l%20)

6 [Prozesse und Threads (Process View) 5](l%20)

7 [Datenspeicherung (Data View) 5](l%20)

8 [Größen und Leistung (Size and Performance) 6](l%20)

# **2 Einführung (Introduction)**

## **2.2 Referenzen (References)**

*<Liste aller verwendeten Dokumente, Bücher, etc…>*

## **2.3 Übersicht (Overview)**

**Im Nachfolgenden Dokument wird die Architektur und deren Komponenten beschrieben.**

*<Übersicht über den restlichen Teil dieses Dokuments geben und dessen Aufbau erläutern>*

# **3 Architektonische Darstellung (Architectural Representation)**

Wir haben uns für die Schichtenarchitektur und gegen die anderen Architekturen entschieden, da unsere Komponenten voneinander abhängig sind.

*<Verbale Beschreibung wichtiger Architekturentscheidungen. Ggf. auch andere Architekturideen kurz beschreiben, die diskutiert und (warum?) verworfen wurden.>*

# **4 Architektonische Ziele & Einschränkungen (Architectural Goals and Constraints)**

Wir haben vier verschiedene Komponenten. Die Datenbank, das Email-Benachrichtigungssystem, die GUI und unser Programm an sich.

Unser Programm ist von den anderen Komponenten abhängig. Diese können auch die Schwachstellen unseres Programmes sein, denn ohne die Datenbank würden die Benutzer-Profile und Ihre Termine nicht eingespeichert werden. Das bedeutet, dass ohne die Datenbank unsere Software ihre Aufgabe nicht erfüllen würde. Außerdem kann für unser Programm das Email-Benachrichtigungssystem zum Problem werden, da wir die Einladung für unsere Software mit dem entsprechenden Gruppen-Namen und Gruppen-Passwort per Email versenden. Genauso sollen Benachrichtigungen durch das Email-Benachrichtigungssystem versendet werden, wenn Termine erstellt oder gelöscht werden. Das würde bedeuten, dass man den Gruppen-Namen, sowie das Gruppen-Passwort mündlich weitergeben müsste und dass der Benutzer über Änderungen im Kalender nicht informiert werden würde, wenn die Verbindung nicht vorhanden wäre. GUI ist für unsere Benutzerschnittstelle wichtig, denn sie ist die visuelle Darstellung in unserem Programm. Wenn diese nicht funktionieren würde, könnte man die Kalender und die dazugehörigen Termine nicht sehen. Außerdem könnte der Benutzer nichts in unserer Software machen, da er z.B. keine Buttons zur Termin-, Profil- und Gruppenerstellung sehen würde.

Wie bereits erwähnt, müssen auf den Computern sowie auf den Handys Windows10 installiert und eine Verbindung zum Internet vorhanden sein, damit unser Programm funktioniert und die Datenverbindung hergestellt werden kann.

Unsere Schichtenarchitektur ist aufgeteilt in GUI, Klassen, Email-Benachrichtigungssystem und Datenbank. Von der Datenbank ist alles abhängig, da ohne die Datenbank unser Programm nicht funktionieren würde. Das Email-Benachrichtigungssystem muss nur wissen, welche Termine zu welchem Zeitpunkt stattfinden und gibt diese Information entsprechend weiter. Die Klassen können nur in vollem Umfang funktionieren, wenn das Gruppenpasswort und der Gruppenname durch das Email-Benachrichtigungssystem weitergegeben und die Daten der Benutzer sowie der Termine in der Datenbank gespeichert werden. Alles andere muss funktionieren damit durch die GUI die Schnittstelle zum Benutzer hergestellt werden kann. Wir haben einen externen Server der immer online ist und welcher dadurch erlaubt, dass jeder Benutzer jederzeit Daten aufrufen oder sich anmelden kann.

# **5 logische Architektur (Logical View)**

*<Beschreibung der logischen Struktur des Projekts>*

## **5.1 Übersicht (Overview)**

Wir haben vier verschiedene Schichten bzw. Packages. Die GUI, die Datenbanksysteme und die Problem-Domain und das Email-Benachrichtigungssystem.

*<Aufteilung in Packages, Layer. Hier ein Übersichtsdiagramm, in den folgenden Abschnitten dann Detaildiagramme bis auf Klassenebene>*

## **5.2 Design Pakete (Architecturally Significant Design Packages)**

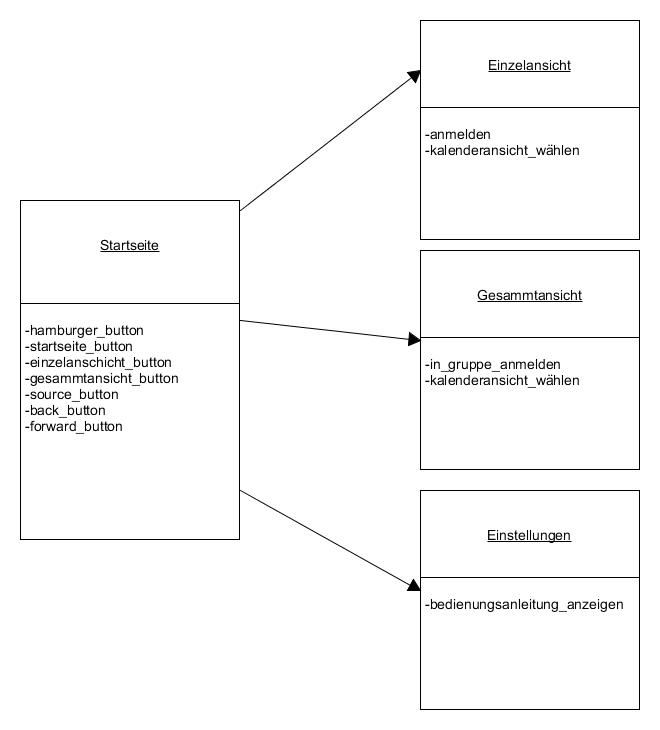
### **5.2.1 Package GUI**

#### ***5.2.1.1 Beschreibung des Package***

GUI ist die Schnittstelle zum Benutzer. Sie hat die Aufgabe die Software mit graphischen Symbolen, Steuerelementen und Widgets auszustatten und bedienbar zu machen. Dies würde für unsere Software bedeuten, dass man durch die GUI Gestaltung zwischen den einzelnen Seiten navigieren kann. Außerdem sieht man so die verschiedenen Kalenderansichten und kann so Termine eingeben die man ansonsten nicht sehen könnte.

<Beschreibung des Package. Aufgabe, etc…>

#### ***5.2.1.2 Diagramme***



#### ***5.2.1.3 Schnittstellen***

Der Benutzer öffnet die Software und sieht das Hauptfenster. Danach hat er die Möglichkeit durch den Hamburgerbutton die verschiedenen Funktionen der darunterliegenden Buttons angezeigt zu bekommen. Ein Button führt den Benutzer zur Einzelansicht. Dabei öffnet sich ein extra Fenster wo er seinen eigenen Kalender begutachten und zwischen verschiedenen Kalender-Ansichten wählen kann. Dieselbe Funktion bietet der Button für die Gesamtansicht, mit Ausnahme, dass der Benutzer den Kalender für die gesamte Gruppe sieht. Außerdem gibt es noch einen Button für Informationen. Dieser führt zu einer Seite wo der Benutzer eine Reihe von Auswahlmöglichkeiten hat. Hier findet er die Bedienungsanleitung, Hilfe, Profil\_bearbeiten und Exit als Funktion. Außerhalb des Hamburgerbuttons in der Taskleiste befindet sich noch ein Back- und Forwardbutton. Mit dem Backbutton kann er die zur letzt geöffnete Seite öffnen. Mit dem Forwardbutton navigiert er zu der Seite wo man vor dem betätigen des Backbuttons war. In der Taskleiste ist noch ein Sourcebutton vorhanden. Dieser beinhaltet ein Textfeld in dem der Benutzer einen Begriff eintippen kann und welcher dann als Termin gesucht wird. Auf jeder Seite sieht er ganz oben in einem Textfeld auf welcher Seite er sich befindet und kann sich dadurch zurechtfinden.

#### ***5.2.1.4 Operationen***

##### *5.2.1.4.1 Interne Operation 1*

<Beschreibung der Operation mit Sequenzdiagramm>

##### *5.2.1.4.2 Interne Operation 2*

<Beschreibung der Operation mit Sequenzdiagramm>

### **5.2.2 Package Problem-Domain**

#### ***5.2.2.1 Beschreibung des Package***

<Beschreibung des Package. Aufgabe, etc…>

#### ***5.2.2.2 Diagramme***

<Klassendiagramm>

#### ***5.2.2.3 Schnittstellen***

<Beschreibung der Schnittstellen>

#### ***5.2.2.4 Operationen***

##### *5.2.2.4.1 Interne Operation 1*

<Beschreibung der Operation mit Sequenzdiagramm>

##### *5.2.2.4.2 Interne Operation 2*

<Beschreibung der Operation mit Sequenzdiagramm>

### **5.2.3 Package Datenbanksysteme**

#### ***5.2.3.1 Beschreibung des Package***

<Beschreibung des Package. Aufgabe, etc…>

#### ***5.2.3.2 Diagramme***

<Klassendiagramm>

#### ***5.2.3.3 Schnittstellen***

<Beschreibung der Schnittstellen>

#### ***5.2.3.4 Operationen***

##### *5.2.3.4.1 Interne Operation 1*

<Beschreibung der Operation mit Sequenzdiagramm>

##### *5.2.3.4.2 Interne Operation 2*

<Beschreibung der Operation mit Sequenzdiagramm>

### **5.2.4 Package E-Mail-Benachrichtigungssystem**

#### ***5.2.4.1 Beschreibung des Package***

Das E-Mail-Benachrichtigungssystem wird gebraucht, um Erinnerungen und Einladungen zu verschicken.

Die Benutzer können für ihre Termine Erinnerungen einstellen und auch Wiederholungen für die Erinnerungen, die dann über E-Mail empfangen werden. Außerdem wird die Einladung zu einer Gruppe über E-Mail versendet.

#### ***5.2.4.2 Diagramme***

<Klassendiagramm>

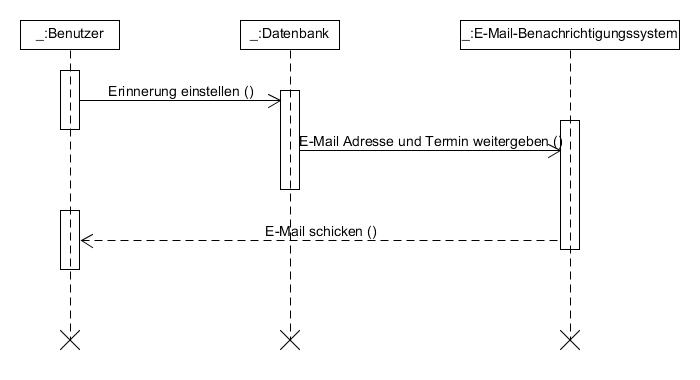
#### ***5.2.4.3 Schnittstellen***

Das E-Mail-Benachrichtigungssystem braucht eine Schnittstelle zu unserem Programm und zu der Datenbank, da dort die E-Mail Adressen der Nutzer gespeichert werden.

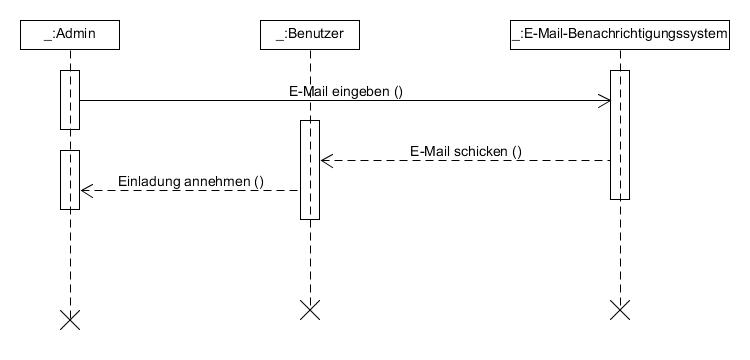
Die Schnittstelle zu unserem Programm ist Outlook. Durch den Outlook Client werden E-Mails mit unserer E-Mail Adresse an die Nutzer versendet.

#### ***5.2.4.4 Operationen***

##### *5.2.3.4.1 Interne Operation ‚Erinnerung schicken‘*



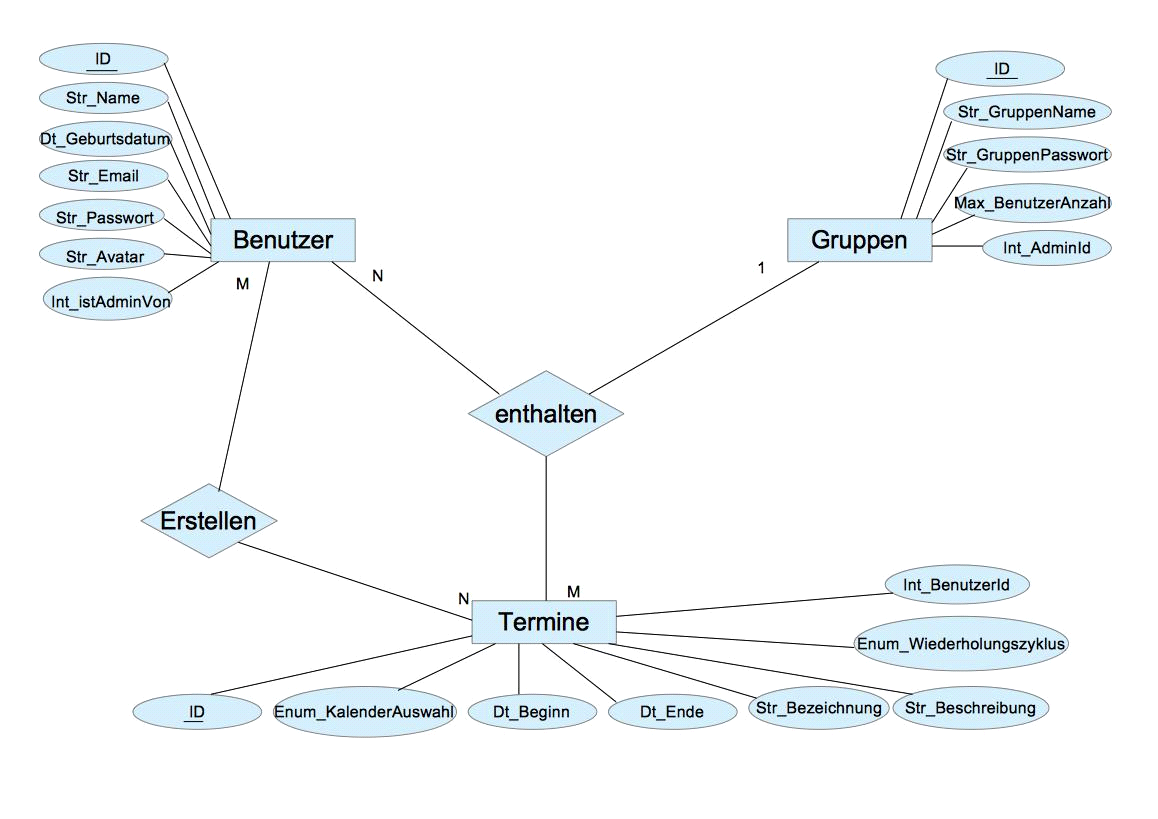
##### *5.2.3.4.2 Interne Operation ‚Einladung schicken‘*



# **6 Prozesse und Threads (Process View)**

*<Wenn mehrere Prozesse oder Threads eingesetzt werden, wird hier beschrieben, wie diese ablaufen, miteinander funktionieren, Daten austauschen, sich synchronisieren, etc...>*

# **7 Datenspeicherung (Data View)**



# **8 Größen und Leistung (Size and Performance)**

Die Verwaltung unterstützt pro Person maximal 20‘000 Einträge. Da sonst zu viel Speicher in Anspruch genommen wird durch die Einträge im Kalender. Damit aufgrund des vollen Speichers die Leistung nicht zu sehr in Anspruch genommen wird, müssen die Einträge regelmäßig gelöscht werden. So bleibt genügend Speicher frei für neue Einträge und die Leistung wird nicht zu stark behindert.

*<Einschränkungen der Applikation bezüglich Speicher, Leistung, etc…. (zum Beispiel: Verwaltung unterstützt maximal 20'000 Einträge)>*