Relate.ch - the social addressbook

Bachelorarbeit

Dario Bosshard & Luzius Tiefenauer

Frühlingssemester 2009
11. Juni 2009

Betreuer: Josef Joller
Experte: Matthias Lips

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Inhaltsverzeichnis

Teil I
Übersicht

1 Aufgabenstellung

1.1 Einführung und Aufgabe

Im Rahmen einer Studienarbeit [1] wurde bereits ein Prototyp von Relate.ch erstellt. Dabei handelt es sich um eine Webplattform, welche das Verwalten von Kontaktdaten ermöglicht und dies mithilfe von Social Networking Techniken grundlegend vereinfacht. Das Resultat der bisherigen Arbeit bietet eine solide Grundlage für den Aufbau einer solchen Plattform.

1.2 Erwartete Resultate

Ziel der Bachelorarbeit ist es, die Webplattform Relate.ch zur Marktreife zu führen. Dabei sollen folgende Punkte abgehandelt werden:

- Gewährleistung der Sicherheit von sensiblen Kontaktdaten
- Verbesserung der Stabilität und Performance der Applikation
- Erstellen eines Web-API für die Entwicklung von unabhängigen Clients
- Implementierung von Synchronisationswerkzeugen (Outlook Add-In, PDA-Client)
- Erweiterung der bestehenden Funktionalität (Gruppierung, Messaging, Suchfilter)

Das Ziel ist eine stabil funktionierende Webplattform, welche durch ihre Funktionalität überzeugt und dadurch als nützliches Werkzeug im Alltag Verwendung findet.

2 Abstract

2.1 Projektidee

Die hohe Anzahl an Kommunikationsmitteln erschwert zunehmend das Verwalten von Kontaktdaten , da sie auf verschiedenen Geräten und Plattformen aktuell gehalten werden müssen.

Ziel dieser Bachelorarbeit ist es, eine Webplattform zu entwickeln, welche das Verwalten von Kontaktdaten ermöglicht und Synchronisationswerkzeuge für Endgeräte zur Verfügung stellt.

Das Erfassen neuer Kontakte wird durch einen automatischen Telefonbuchabgleich sowie eine Visualisierung der gefundenen Treffer erleichtert.

Sobald die erfassten Personen das System ebenfalls nutzen, verwalten diese ihre Kontaktdaten selber. Dies soll langfristig ein stets aktuelles Kontaktbuch ermöglichen.

2.2 Ergebnisse

Der Prototyp der Studienarbeit wurde komplett überarbeitet, um eine stabile Basis für den Ausbau zu bieten. Dabei ist eine Webplattform entstanden, welche bereits von ersten aktive Usern genutzt wird.

Ein umfassendes Sicherheitssystem ermöglicht es, für jede erfasste Kontaktinformation genau zu definieren, welche Personen diese einsehen dürfen.

Zeitversetztes Senden von Emails und SMS-Nachrichten an Einzelpersonen oder ganze Gruppen ist mit dem integrierten Messaging Dienst möglich.

Das Design der Webseite ist auf eine einfache und intuitive Bedienung ausgelegt.

Eine Webservice-Schnittstelle bietet zudem die Möglichkeit, von beliebigen Programmen auf die Kontaktdaten zuzugreifen. Als Beispielsapplikation wurde ein Synchronisationstool für Microsofts Outlook erstellt.

2.3 Ausblick

Viele geplante Funktionen konnten wegen des engen Zeitrahmens noch nicht implementiert werden. Deshalb wird die Webplattform nach dieser Arbeit weiterentwickelt und ausgebaut.

Zudem wird die Wirtschaftlichkeit einer langfristigen Betreibung der Plattform geprüft, da eine einfache, zentrale und vernetzte Kontaktverwaltung für viele Personen ein Bedürfnis darstellt.

3 Management Summary

3.1 Ausgangslage

Die stetig steigende Anzahl von Kommunikationsmitteln erfordert einen zunehmend grösseren Aufwand, um Kontaktdaten auf allen Geräten und Plattformen aktuell zu halten. Das Ziel dieser Arbeit ist es, diesen Prozess zu vereinfachen, indem eine zentrale Verwaltung der Kontaktdaten ermöglicht wird. Diese zentrale Kontaktverwaltung soll automatisch mit Emailprogrammen, Mobiltelefonen und anderen Kommunikatiosgeräten synchronisiert werden können.

Im Rahmen einer Studienarbeit wurde bereits ein lauffähiger Prototyp einer Webplattform entwickelt. Diese ermöglicht das Erfassen von Kontaktdaten und unterstützt dies durch einen automatischen Telefonbuchabgleich sowie eine Visualisierung der gefundenen Treffer. Sobald die erfassten Personen das System ebenfalls nutzen, werden sie mit den bereits erfassten Daten verknüpft und verwalten diese fortan selber. Ändert nun ein Mitglied seine Wohn- oder Emailadresse, erfahren sämtliche Bekannten, die das System ebenfalls nutzen, sofort davon. Dies soll langfristig ein stets aktuelles Kontaktbuch ermöglichen.

Im Rahmen dieser Bachelorarbeit soll nun dieser Prototyp weiterentwickelt werden, mit dem Ziel eine marktreife Plattform zu erschaffen. Dabei sollen die jetzige Plattform verbessert und sinnvolle neue Funktionalitäten integriert werden.

Einzelpersonen sollen Gruppen erstellen und ihnen beitreten können, um eine einfache Kommunikation von Personengruppen zu ermöglichen. Dabei kann für jede Gruppe eine Emailadresse erstellt werden, welche eintreffende Emailnachrichten an alle Gruppenmitglieder weiterleitet. Neben Email sollen auch SMS direkt versendet werden können.

Zusätzlich sollen die sensiblen Kontaktinformationen durch ein Sicherheitssystem geschützt werden, welches jedem Benutzer ermöglicht, für jede seiner Kontaktinformation genau zu definieren, wer diese einsehen darf.

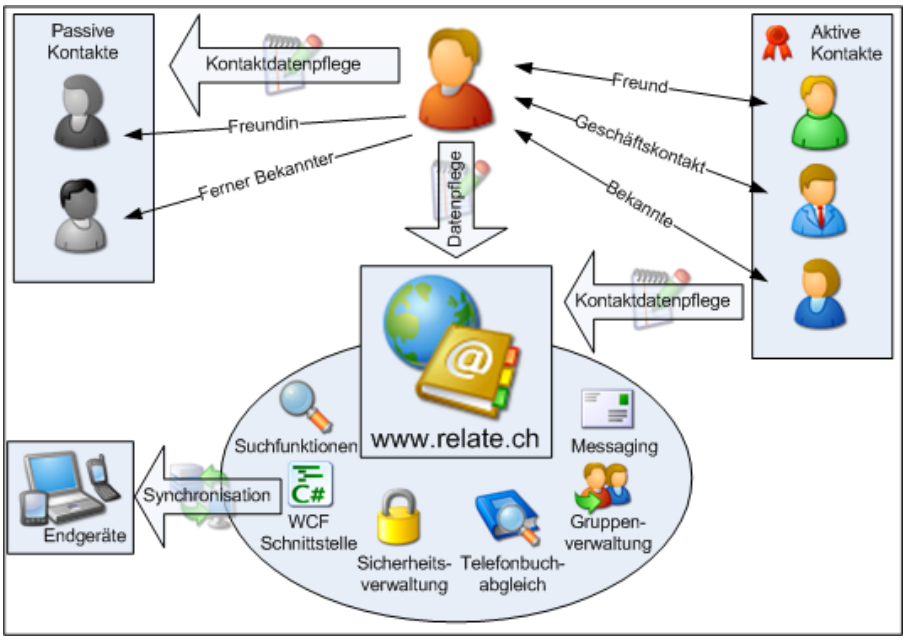

Abbildung 1: Schematische Darstellung der Webplattform www.relate.ch

3.2 Vorgehen

Da diese Arbeit durch einen relativ knappen Zeitrahmen beschränkt ist, wurden zuerst die geplanten Aufgaben priorisiert und mittels einem Gantt-Diagramm ein Projektplan erstellt. Dadurch war es möglich, die gegebene Zeit optimal zu nutzen.

In einem ersten Schritt wurde der vorhandene Prototyp analysiert und umfassend überarbeitet. Die zuvor mittels LINQ to SQL erstellten Datenbankabfragen wurde durch Stored Procedures ersetzt, was eine grössere Kontrolle über die Abfragen ermöglicht. Ausserdem wurde es dadurch möglich, einen Grossteil der Businesslogik direkt in der Datenbank zu verankern. Die Datenbankresultate wiederum werden nun viel effektiver direkt auf Data Transfer Objects gemappt. Gesamthaft wurde ein sehr grosser Aufwand betrieben, um die Qualität der bestehenden Codebasis zu verbessern, da diese für die langfristige Wartung und Weiterentwicklung sehr wichtig ist.

Als nächstes wurden die geplanten Sicherheitsfunktionen implementiert. Die neue Codestruktur ermöglichte es, die Authentisierung sowie die Autorisation tief im System zu verankern. Der Benutzer muss sich immer zuerst beim System anmelden, um ein Sicherheits-Token zu erhalten. Nur mit diesem kann er auf die Datenbank zugreifen, die anhand dieses Tokens die Zugriffsberechtigung auf die einzelnen Kontaktinformationen prüft.

Nachdem die Sicherheit der Daten gewährleistet war, konnten neue Funktionen wie das Messaging oder die Kontaktgruppen implementiert werden. Dabei wurden auch neue Ideen wie der zeitversetzte Versand laufend geprüft und direkt integriert.

Während der Implementation neuer Funktionen musste auch die Benutzeroberfläche der Webseite laufend überarbeitet werden. Mithilfe von Usabilitytests wurde das Bedienkonzept in mehreren Schritten kontinuierlich verbessert.

Als letztes wurde eine WCF-Schnittstelle entwickelt, die es ermöglicht, von beliebigen Programmen auf die Daten der Kontaktverwaltung zuzugreifen. Als Beispielsimplementation wurde ein Synchronisationsprogramm für Microsoft Outlook erstellt.

Die erforderliche Stundenzahl wurde mit über 760 geleisteten Stunden überschritten, was neben der umfangreichen Arbeit vor allem auch auf eine hohe Selbstmotivation zurückzuführen ist.

3.3 Ergebnisse

Das Ergebniss dieser Bachelorarbeit ist unter www.relate.ch einsehbar und wird bereits von ersten aktiven Usern genutzt. Wie schon im Prototyp werden die Kontaktdaten automatisch mit dem Telefonbuch abgeglichen. Zusätzlich können die Daten nun mittels Luftaufnahmen von Virtual Earth visualisiert und sogar in 3D erkundet werden.

Jeder Benutzer kann nun genau bestimmen, welche seiner Kontaktinformationen er wem anzeigen will. So kann er seine private Emailadresse vor seinen Geschäftskontakten verbergen oder seine Mobiltelefonnummer nur den besten Freunden preisgeben.

Durch das neue Webdesign kann der Platz moderner Breitbild-Monitore besser genutzt werden. Beim Aufbau der Webseite wurde mehr auf Konsistenz geachtet. Die Karte befindet sich nun beispielsweise stets am selben Ort. Auch die verbesserte Suchfunktion steht nun auf jeder Seite zur Verfügung und macht mittels AJAX-Technologie bereits während dem Tippen Vorschläge für mögliche Treffer.

Neben dem Erfassen von Personen und der Definition der Beziehung zu diesen, können nun auch Gruppen erstellt werden, um die Kommunikation innerhalb der Gruppe zu verbessern. Mithilfe der neuen Messagingfunktionen können beliebig viele Personen auf einfache Weise angeschrieben werden. Das Versenden von Emails und SMS kann dabei auch zeitversetzt stattfinden. Dies ermöglicht es beispielsweise, Erinnerungsnachrichten oder Geburtstagsglückwünsche bereits im Voraus zu planen.

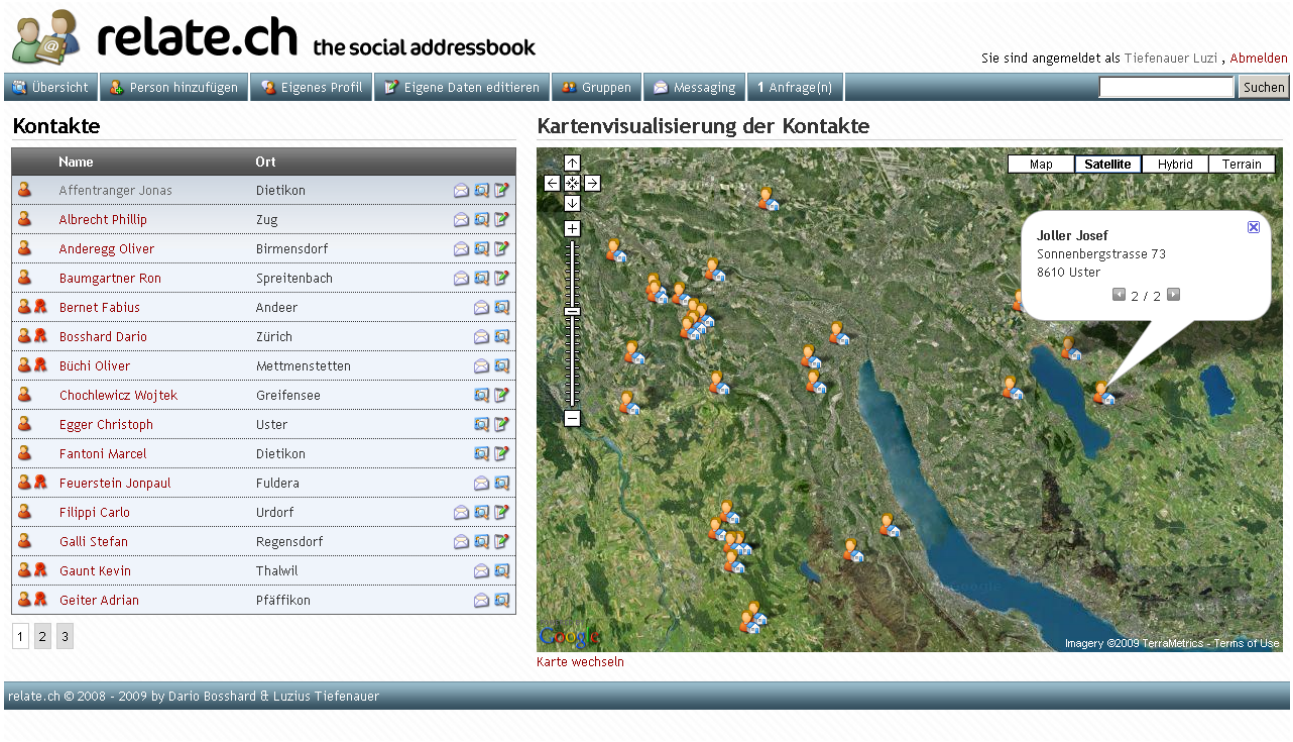

Abbildung 2: Webplattform www.relate.ch

Ein Add-In für Micosoft Outlook ermöglicht das Synchronisieren der online geführten Kontakte mit einem Knopfdruck. Neben diesem einfachen Synchronisationsprogramm können dank der Webservice-Schnittstelle beliebige andere Programme auf die Daten von Relate.ch zugreifen.

3.4 Ausblick

Beide Teammitglieder sind weiterhin vom Potenzial der Kontaktverwaltungsplattform überzeugt und motiviert, auch nach dieser Arbeit Zeit in dieses Projekt zu investieren. Wegen des engen Zeitrahmens konnten viele geplante Funktionen noch nicht implementiert werden. Trotzdem haben wir schon sehr viel positives Feedback erhalten und sind deshalb überzeugt, dass eine einfache, zentrale und vernetzte Kontaktverwaltung für viele Personen ein Bedürfnis darstellt.

Um eine langfristige Betreibung zu ermöglichen, soll nun auch die Wirschaftlichkeit geprüft werden. Bannerwerbung bringt heutzutage längst nicht mehr ausreichend Ertrag, um die Kosten für den Unterhalt und vor allem auch für die Weiterentwicklung zu decken.

Für die Analyse der wirtschaftlichen Situation konnte erfreulicherweise ein Wirtschaftsstudent der Hochschule St.Gallen gewonnen werden, welcher seinerseits eine Bachelorarbeit zu diesem Thema erarbeiten wird.

Das Ziel dieser Arbeit war es, die Kontaktverwaltung grundlegend zu vereinfachen. Nun sind wir gespannt, ob unsere hierfür entwickelte Webplattform www.relate.ch bei den Benutzern Anklang findet.

Teil II
Technischer Bericht

4 Analyse

4.1 Allgemeine Beschreibung

4.1.1 Perspektiven

Mit dieser Bachelorarbeit soll die Basis der Studienarbeit weiter ausgebaut und um neue Funktionen angereichert werden. Dabei soll eine funktionsfähige und sinnvoll nutzbare Plattform entstehen, welche auch über diese Arbeit hinaus verwendet und weiterentwickelt wird.

4.1.2 Funktionen

Folgende neuen Funktionen sollen implementiert werden:

- Gruppierung der Kontakte
- Austausch von Nachrichten via Email und SMS
- Schnittstelle zur Synchronisation mit anderen Programmen auf Basis von WCF Services
- Beispiel Implementation einer Synchronisation via Outlook Add-In
- Refactoring der bestehenden Funktionen, der Codebasis und Datenbankstruktur

4.1.3 Benutzer Charakteristik

Das Endprodukt soll von einer möglichst breiten Zielgruppe bedienbar sein, da ein grosser Benutzerkreis essentiell für den Erfolg des Systems ist. Deshalb kann der Benutzerkreis nicht gezielt definiert werden. Es soll aber darauf geachtet werden, dass möglichst keine Benutzer ausgeschlossen werden, sei dies aus technischen, demografischen oder physischen Gründen.

4.1.4 Einschränkungen

Da die zur Verfügung stehende Zeit für diese Arbeit beschränkt ist, soll in regelmässigen Abständen der Fortschritt analysiert und der Funktionsumfang bei bedarf eingeschränkt werden.

Das System soll vorerst nur für die Schweiz entwickelt und erst zu einem späteren Zeitpunkt internationalisiert werden. Trotzdem soll darauf geachtet werden, dass die Internationalisierung in Zukunft möglichst einfach gemacht werden kann.

Wichtig ist vor allem, dass die Grundfunktionen zuverlässig und fehlerfrei funktionieren.

4.1.5 Bestehende Lösungen

In der Studienarbeit [1] wurden die bestehenden Lösungen, wie Facebook, XING und StudiVZ, bereits umfassend analysiert. Aus den gefundenen Schwachstellen wurden entsprechende Anforderungen an diese Plattform definiert.

Aus der Schlussfolgerung sind vor allem zwei wichtige Punkte hervorzuheben:

- Die Plattform soll die bisherigen Kommunikationsmittel einbinden, statt einfach nur neue zu schaffen.
- Es soll möglich sein, Personen zu kontaktieren, welche sich nicht selbst bei der Plattform registriert haben.

4.1.6 Vision

Die Idee eines interaktiven Kontaktbuches soll mithilfe einer stabilen und stetig leistungsfähiger werdenden Plattform einem breiteren Publikum näher gebracht werden. Das Hauptziel ist es, dass das Resultat dieser Arbeit stabil läuft und als nützliches Werkzeug im Alltag Verwendung findet.

Der Applikationscode soll übersichtlich und sauber gehalten werden, um die Weiterentwicklung der Plattform möglichst zu vereinfachen.

4.2 Spezifische Anforderungen

4.2.1 Funktionale Anforderungen

Sicherheit

Da die Benutzer viele persönliche Kontaktdaten angeben, ist die Sicherheit dieser Daten ein wichtiger und zentraler Punkt dieser Arbeit. Es muss jederzeit sichergestellt werden, dass die Daten jeweils nur von dazu berechtigten Personen eingesehen und verändert werden können.

Eine genauere Spezifikation der sicherheitsrelevanten Anforderungen findet sich im Kapitel 4.5 auf Seite 24.

Standardisierte Datenhaltung

Die einzelnen Daten von Personen sollen von Beginn an nach einem standardisierten Muster gespeichert werden. Dies soll das Erstellen von Schnittstellen zu anderen Systemen (Import & Export) vereinfachen. Zusätzlich wird damit eine spätere Ausdehnung auf andere Länder ermöglicht.

4.2.2 Bedienbarkeit

Um den Benutzerkreis möglichst nicht einzuschränken, muss das System sehr einfach zu bedienen sein. Die heutige Webtechnologien bieten mit Javascript und Ajax viel Spielraum bei der Vereinfachung der Bedienung von Webseiten. Allerdings soll auch jederzeit sichergestellt werden, dass Benutzer mit deaktiviertem Javascript nicht vom System ausgeschlossen werden.

Die Lernzeit muss möglichst tief gehalten werden, da Benutzer einer Webseite nicht bereit sind, viel Zeit für die Einarbeitung zu verwenden.

4.2.3 Zuverlässigkeit

Um eine breite Akzeptanz bei den Benutzern zu erreichen, müssen lange Ladezeiten und Ausfälle verhindert werden. Bei einem Ausfall des Systems dürfen die Daten nicht verloren gehen.

4.2.4 Leistung

Da die Anwenderzahl theoretisch nach oben nicht begrenzt ist, muss die Applikation sehr gut skalieren. Daher ist auch die Wahl der richtigen Technologie ein wichtiger Punkt. In der Studienarbeit [1] wurde deshalb umfassend analysiert, welche Technologie die Anforderungen am besten erfüllt.

Der in der Studienarbeit eingesetzte virtuelle Server der HSR genügt den gestiegenen Anforderungen nicht mehr. Zudem würde dieser nach der Bachelorarbeit gelöscht werden, was eine Weiterführung des Projektes erschweren würde. Deshalb sollen Alternativen dazu geprüft und ein leistungsfähiger Ersatz gefunden werden.

4.2.5 Schnittstellen

Benutzerschnittstellen

Die Benutzer sollen über ein Webinterface Zugriff auf alle zur Verfügung stehenden Funktionen haben. Dies ermöglicht eine schnelle Verbreitung und eine kurze Einarbeitungszeit, da keine Installation nötig ist.

Das bestehende Webinterface des Prototypen der Studienarbeit soll bezüglich Benutzerfreundlichkeit und Aussehen umfassend verbessert werden.

Softwareschnittstellen

Innerhalb des Programms soll auf eine möglichst schwache Kopplung zwischen den einzelnen Komponenten geachtet werden. Um dies zu erreichen, sollen die einzelnen Komponenten hauptsächlich über Interfaces miteinander kommunizieren.

Kommunikationsschnittstellen

Die Kontaktdaten sollen auch über einen Webservice abrufbar sein. Dadurch soll das Entwickeln von Synchronisationswerkzeugen ermöglicht werden. Hierzu soll ein WCF-Service entwickelt werden, welcher nach einer erfolgreichen Authorisierungsphase die benötigte Funktionalität zur Verfügung stellt.

4.3 Use Cases

4.3.1 Use Case Diagramm

Die möglichen Benutzeraktionen wurden nicht in einem klassischen Use Case Diagramm visualisiert, sondern mithilfe eines Mindmaps gruppiert. Dies ermöglicht einen groben Überblick über die Funktionen, als auch eine relativ fein granulierte Liste der benötigten Interaktionsmöglichkeiten.

Anhand dieses Diagramms kann jederzeit überprüft werden, was in den einzelnen Bereichen noch beachtet und implementiert werden muss. Einige komplexere Use Cases werden nachfolgend detaillierter erläutert.

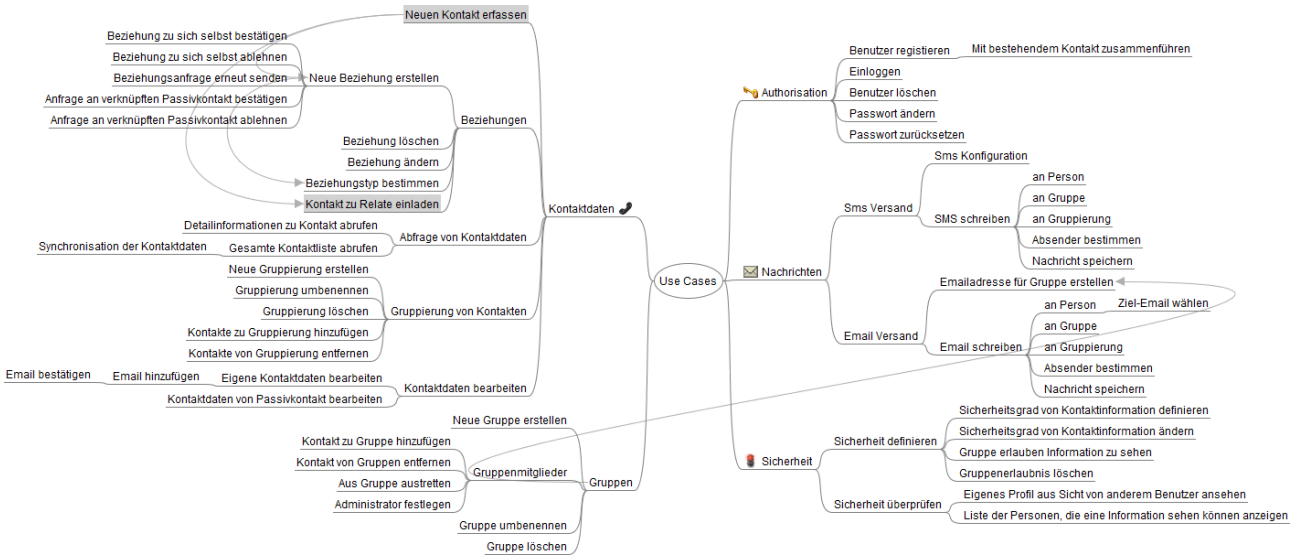

Abbildung 3: Use Case Diagramm

4.3.2 Aktoren und Stakeholders

Aktor ist stets der aktuelle Benutzer, welcher die Kontaktverwaltung benutzt. Als Stakeholder treten dabei jeweils seine verknüpften Kontakte auf, welche Interesse daran haben, dass der Benutzer immer die aktuellen, aber auch nur die zur Einsicht freigegebenen Kontaktinformationen sehen kann.

4.3.3 Use Case "Benutzer registrieren"

1. Der Benutzer füllt das Registrierungsformular mit den wichtigsten persönlichen Daten aus

   (a) (*Alternativszenario*) Benutzer gerät über eine Einladungsemail auf die Seite

      i. System speichert Identität und Aktivierungscode, welche im Link der Email gespeichert sind.

2. Das System überprüft, ob die Emailadresse bereits in Gebrauch ist.

   (a) (*Alternativszenario*) Person mit derselben Emailadresse ist bereits erfasst, jedoch noch ohne aktiven Account.

      i. System präsentiert dem Benutzer die bereits existierende Person

      ii. Benutzer willigt dem Übernehmen der bestehenden Person zu

A. (*Alternativszenario*) Benutzer lehnt Übernahme ab und muss andere Email wählen. Zurück zu 1.

iii. System erstellt Benutzer und verknüpft ihn mit bestehender Person. Zusätzlich wird ein Aktivierungsemail versendet.

A. (*Alternativszenario*) Benutzer ist über Einladungsemail auf die Seite gelangt und System kann Benutzer direkt aktivieren.

(b) (*Alternativszenario*) Person mit derselben Emailadresse ist bereits mit aktivem Account verknüpft

i. System gibt Fehlermeldung an Benutzer aus und gibt ihm die Möglichkeit, das Passwort des bestehenden Accounts neu zu setzten.

3. Der Benutzer gibt seine Adressenangaben ein und klickt auf `Suchen`
4. Das System sucht nach einem Eintrag im Telefonbuch und zeigt die Resultate an

(a) (*Alternativszenario*) Das System findet keine Übereinstimmung

5. Der Benutzer wählt einen Telefonbucheintrag aus der Liste aus und vervollständigt die Kontaktdaten
6. Das System erstellt die Person, den Benutzer und sämtliche Kontaktdaten. Zusätzlich wird eine Aktivierungsemail versendet.

(a) (*Alternativszenario*) Person, Benutzer oder Kontaktdaten konnten nicht gespeichert werden. Fehlermeldung wird ausgegeben.

7. Das System fordert den Benutzer auf, einen Bestätigungscode einzugeben
8. Der Benutzer gibt den erhaltenen Bestätigungscode ein, um seine Emailadresse zu verifizieren
9. Das System überprüft den Bestätigungscode und aktiviert das Benutzerkonto

(a) (*Alternativszenario*) Der Bestätigungscode ist nicht korrekt

i. System gibt eine Fehlermeldung aus und bittet um erneute Eingabe

Ablaufdiagramm

Da der Ablauf bei einer Registrierung relativ komplex ist, wurde er zum besseren Überblick zusätzlich als Ablaufdiagramm visualisiert.

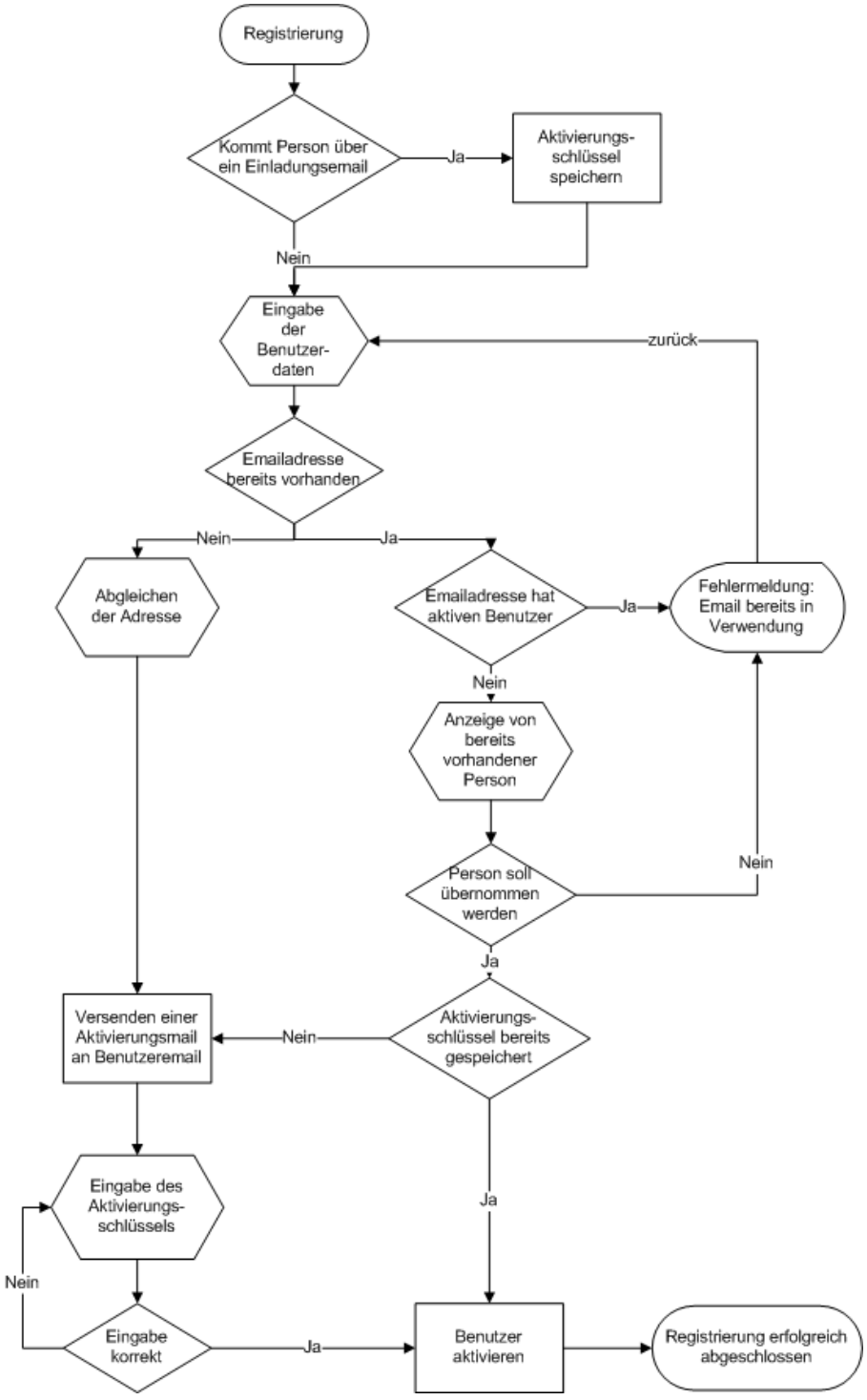

Abbildung 4: Ablaufdiagramm der Benutzerregistrierung

4.3.4 Use Case "Neuen Kontakt erfassen"

1. Benutzer gibt Name und Adresse des Kontaktes ein
2. Das System sucht diese Person in den bereits bei Relate erfassten Personen sowie im Telefonbuch. Die Resultate werden dem Benutzer präsentiert und auf einer Karte visualisiert.
3. Benutzer klickt auf eine Person in den Telefonbuchresultaten
   (a) (*Alternativszenario*) Benutzer klickt auf eine Person bei den Relate-Kontakten
      i. Sprung zu Punkt 6.
   (b) (*Alternativszenario*) Gesuchte Person nicht gefunden, Benutzer fährt ohne Datenübernahme fort
4. System übernimmt Daten aus dem Telefonbuch und präsentiert Erfassungsmaske für weitere Kontaktdaten
5. Benutzer erfasst beliebige weitere Kontaktdaten der Person und klickt auf "Weiter"
6. System präsentiert Maske zur Definition der Beziehung
   (a) (*Alternativszenario*) Erfasste Emailadresse wird bereits von einer anderen Person verwendet.
      i. Zurück zu Punkt 5.
7. Benutzer definiert die Art wie er zu dieser Person in Beziehung steht und klickt auf speichern.
   (a) (*Alternativszenario*) Benutzer wünscht, dass Person eine Einladungsemail zu Relate erhält
      i. System versendet Einladungsemail an Person
8. System speichert die neue Person und die Beziehung

4.3.5 Use Case "Neue Gruppe erstellen"

1. Der Benutzer erfasst Namen und Zweck einer Gruppe
2. System erstellt Gruppe
3. Benutzer fügt Personen der Gruppe hinzu
4. System speichert Verknüpfung der Personen mit der Gruppe und verschickt jeweils eine Einladungsemail.
5. Benutzer kann Gruppe administrieren (Gruppenemail erstellen, Personen hinzufügen/entfernen, Personen zum Administrator machen)

4.4 Domain Analyse

4.4.1 Domainmodell

Aus Gründen der Übersichtlichkeit wird das erarbeitete Domainmodell in mehreren thematisch geordneten Teilebereichen erläutert.

Kontaktdaten

Im Zentrum des Domainmodells steht die Person und ihre Kontaktdaten. Die Kontaktdaten werden auf zwei unterschiedliche Arten auf die Person abgebildet. Die SimplePerson stellt eine Person mit abgeflachten Informationen dar. In ihr ist immer nur eine Kontaktinformation pro Informationsart untergebracht. Die DetailedPerson kann mehrere Kontaktinformationen wie Adressen, Telefonnummern, Emailadressen und Instant-Messenger-Adressen besitzen. Diese werden durch einen Informationstyp (InfoType) kategorisiert. Einer Person kann ein Benutzer (User) angefügt werden.

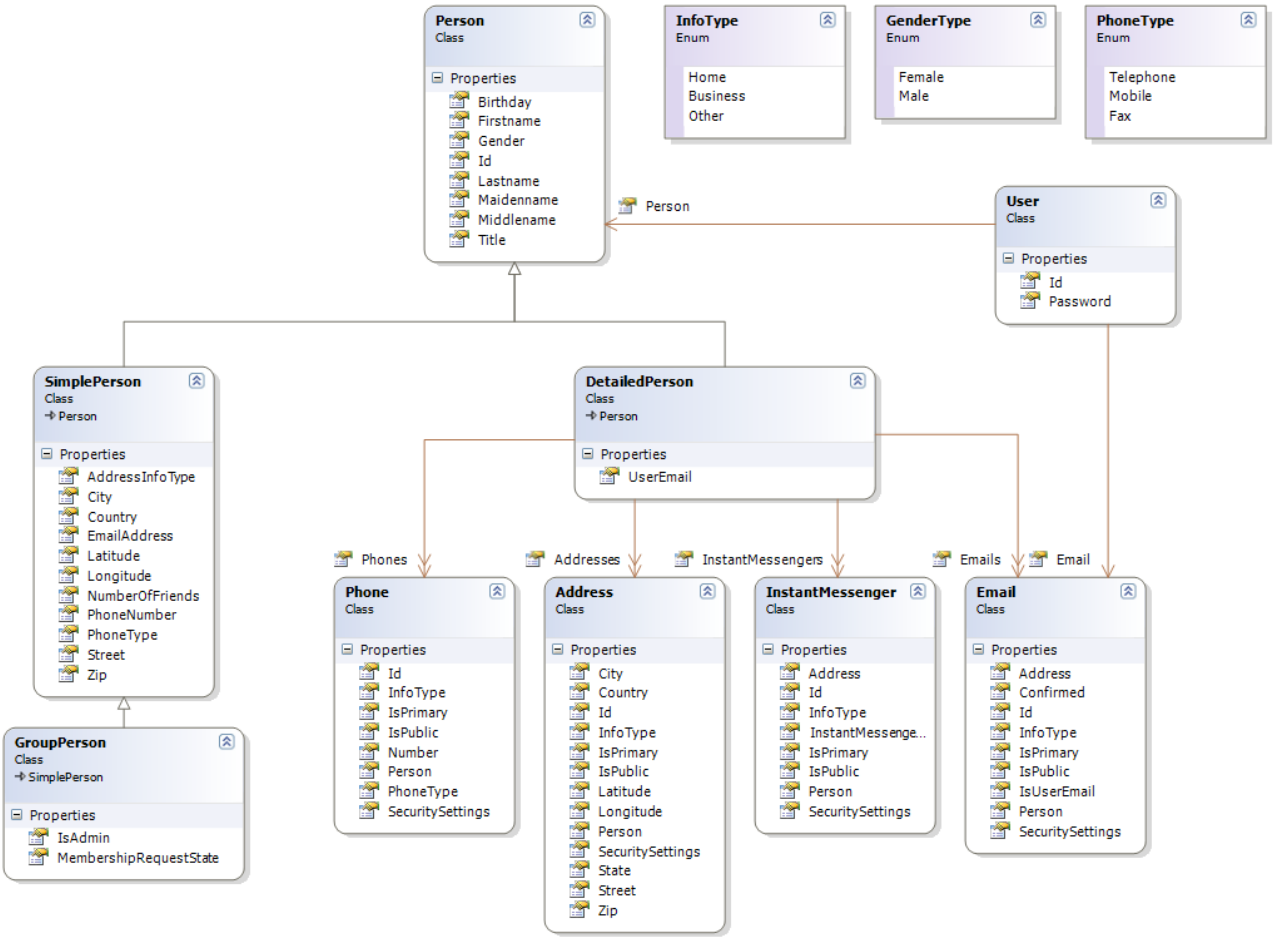

Abbildung 5: Domainmodell - Kontaktdaten

Beziehungen

Personen können untereinander in Beziehung stehen. Eine Verbindung zwischen zwei Personen wird durch einen Beziehungstyp unterschieden. Die Beziehung kann verschiedene Status einnehmen. Wird eine Beziehung aufgebaut, nimmt diese den Status Angefragt an. Ist die angefragte Person einverstanden mit der Anfrage, wird der Status in Akzeptiert, ist sie dagegen, in Abgelehnt geändert. Erlischt die Beziehung, wird der Status auf Gelöscht geändert.

Personen haben zudem die Möglichkeit, sich in Gruppen zusammen zu schliessen.

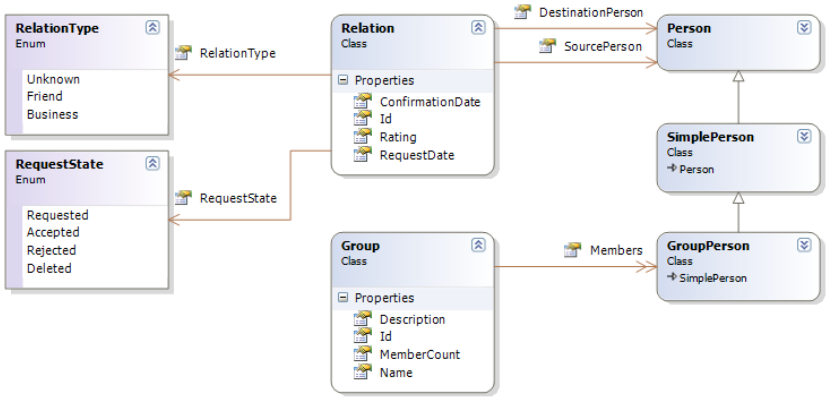

Abbildung 6: Domainmodell - Beziehungen

Sicherheit

Zu jeder Kontaktinformation und Beziehungsart kann eine minimal erforderliche Beziehungsbewertung festgelegt werden. Diese wird als Beziehung zwischen der Beziehungsart und der Kontaktinformation abgebildet.

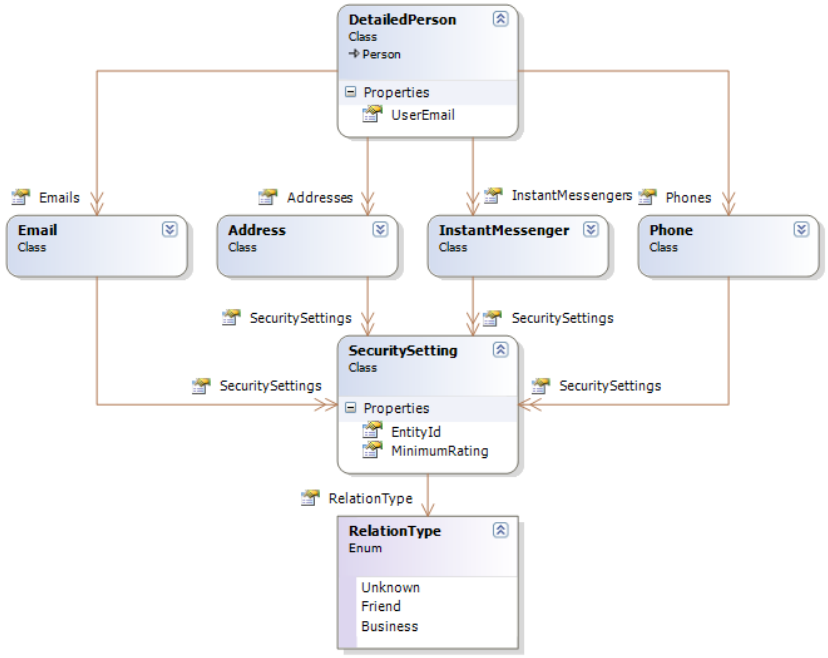

Abbildung 7: Domainmodell - Sicherheit

Messaging

Eine Person kann einer oder mehreren Personen eine Nachricht zukommen lassen. Dabei wird zwischen einer SMS Nachricht und einer Email Nachricht unerschieden. Damit eine SMS Nachricht versendet werden kann, muss eine Konfiguration (SMSMessage) für einen Dienstanbieter (SMSProvider) existieren.

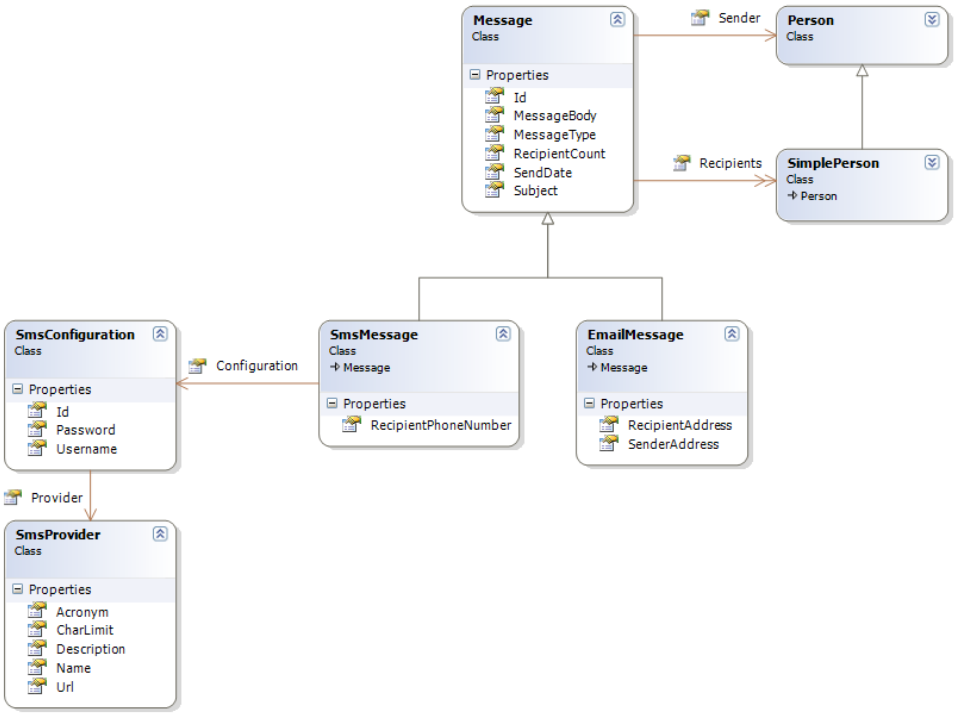

Abbildung 8: Domainmodell - Messaging

4.4.2 System Sequenzdiagramme

Benutzer registrieren

Beim Registrieren eines neuen Benutzers wird als erstes geprüft, ob die übergeben Person bereits existiert. Ist dies nicht der Fall, wird über den ContactController eine neue Person erstellt. Dieser Person werden anschliessend die einzelnen Kontakdaten angefügt. Danach wird die erstellte, bzw. die bereits existierende Person, als Benutzer registriert.

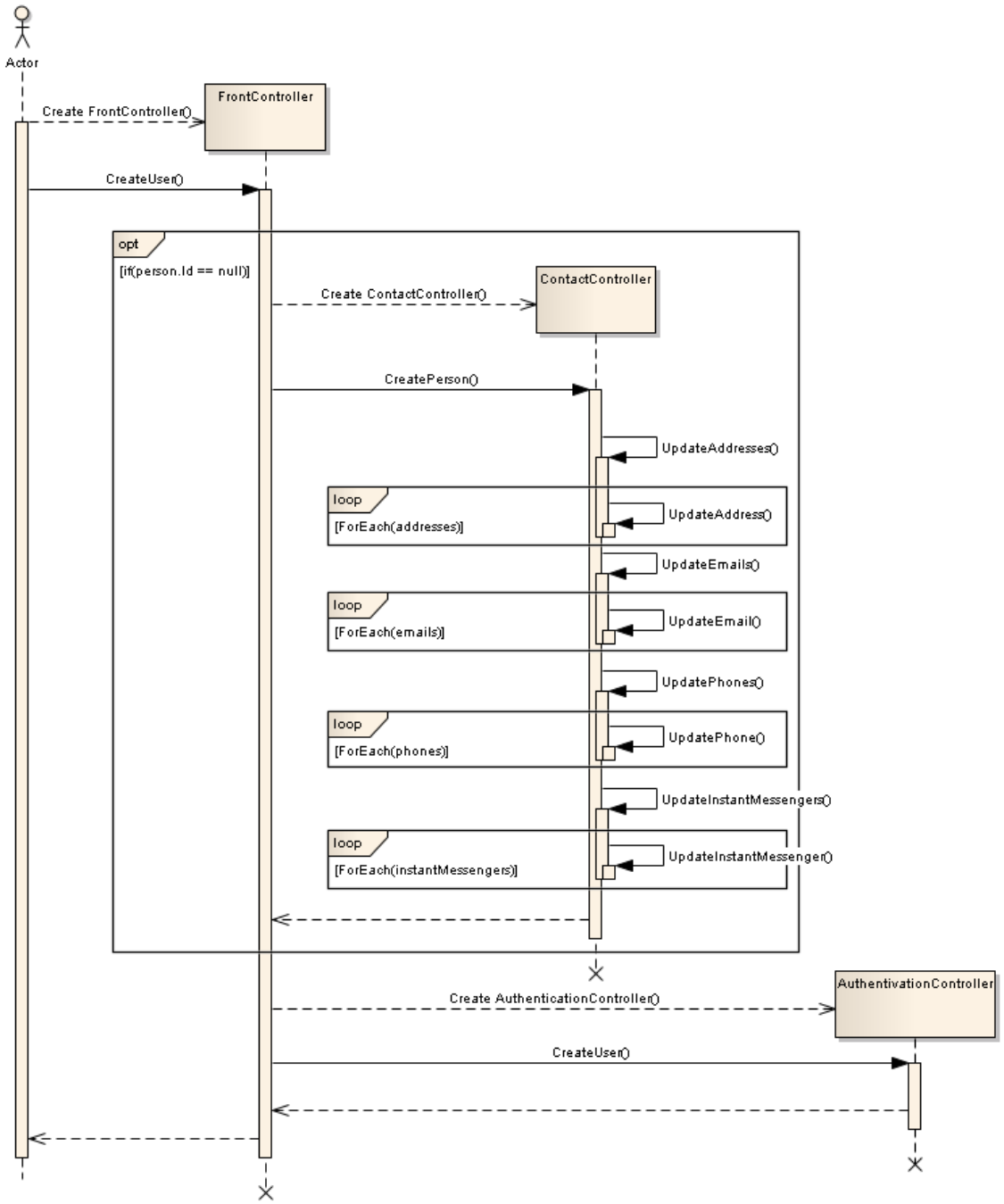

Abbildung 9: System Sequenzdiagramme - Benutzer registrieren

Neue SMS Nachricht erstellen

Das Erstellen einer neuen SMS Nachricht bedingt, dass die zu verwendende SMS Konfiguration gültig ist. Dabei wird geprüft, ob ein Login noch möglich und noch genügend Guthaben für den Versand vorhanden ist.

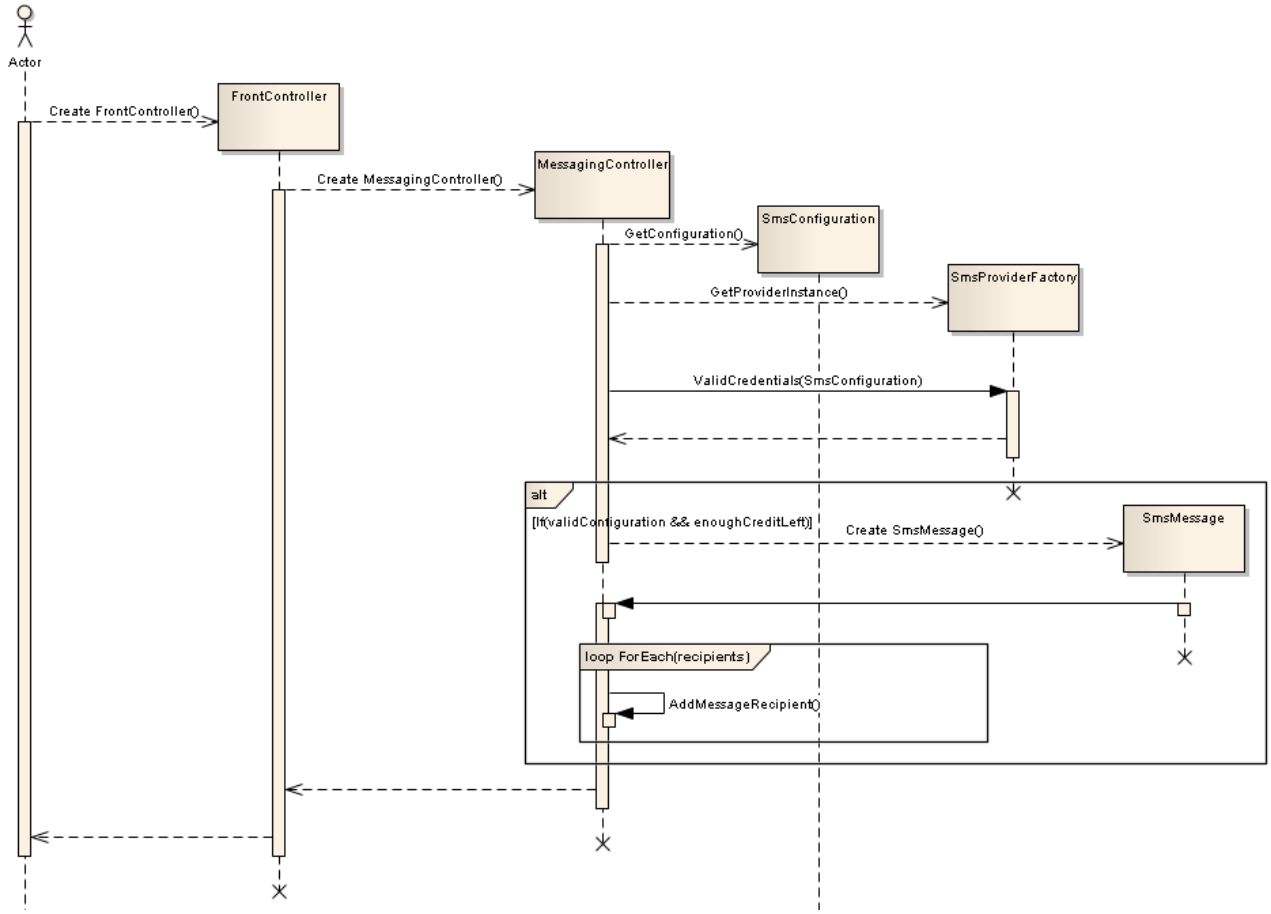

Abbildung 10: System Sequenzdiagramme - Neue SMS Nachricht erstellen

4.4.3 System Contracts

Benutzer registrieren

Cross References Use Case im Kapitel 4.3.3 auf Seite 14

Postconditions

- Personen Objekt wurde erstellt
- Mindestens ein Email Objekt wurde erstellt und mit Person verknüpft
- User Objekt wurde erstellt und mit Person und Email verknüpft
- Aktivierungs Objekt wurde erstellt und mit Email verknüpft

Neuen Kontakt erfassen

Cross References User Case "Neuen Kontakt erfassen" im Kapitel 4.3.4 auf Seite 17

Preconditions

- Benutzer wurde aktiviert

Postconditions

- Personen Objekt wurde erstellt
- Verbindung mit neuer Person und angemeldeter Person wurde erstellt
- Eventuell wurden Adress Objekte erstellt und mit Person verknüpft
- Eventuell wurden Email Objekte erstellt und mit Person verknüpft
- Eventuell wurden Telefon Objekte erstellt und mit Person verknüpft

Neue Gruppe erstellen

Cross References User Case "Neue Gruppe erstellen" im Kapitel 4.3.5 auf Seite 17

Preconditions

- Benutzer wurde aktiviert

Postconditions

- Gruppen Objekt wurde erstellt
- Verbindung zwischen Person und Gruppe wurde erstellt
- Administratoren wurden korrekt gekennzeichnet

4.5 Sicherheit

4.5.1 Grundanforderungen

Die Sicherheit in einem IT System ist ein zentraler Punkt für die Akzeptanz bei den Benutzern. Ist es für einen Angreifer einfach, in das System einzudringen, Daten zu manipulieren oder diese zu missbrauchen, sind die Benutzer auch nicht gewollt, ihre Daten freizugeben. Eine Schwachstelle im System führt zu einem Imageverlust und im Extremfall gar zur Stilllegung des Systems.

Persönliche Daten von Personen stellen ein beliebtes Ziel von Kriminellen dar, da diese Daten für die Werbebranche sehr wertvoll sind und dafür hohe Beträge gezahlt werden.

Gemäss den drei Sicherheitsgrundsätzen Confidentiality, Integrity und Availability, können wir folgende Grundziele der Sicherheit definieren:

- Daten des Benutzers dürfen nur vom Benutzer selbst manipuliert werden können. (Integrity)
- Nur vom Benutzer festgelegte Personen dürfen Daten einsehen (Confidentiality)
- Die Daten müssen zu jeder Zeit vollständig abrufbar sein (Availability)

4.5.2 Zugriffsschutzmodell

Zusätzlich zu den Grundanforderungen soll es für den Benutzer möglich sein, für jede seiner erfassten Kontaktinformationen genau zu definieren, wer diese abrufen darf.

Nach einigem Studium in diesem Bereich [7], stellte sich heraus, dass die Wahl sowie die effektive Erstellung eines solchen Sicherheitsmechanismus ein bekanntes Problem darstellt. Leider stellte sich auch heraus, dass keines der gefundenen Zugriffskontrollparadigmen die hier gestellten Anforderungen komplett erfüllt.

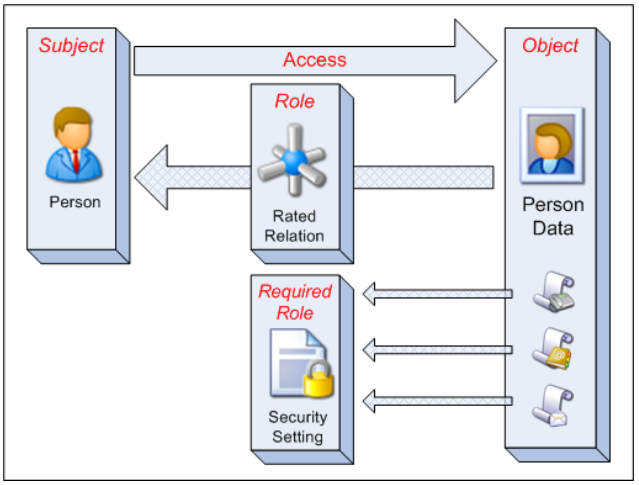

Abbildung 11: Schema der Zugriffsschutz Strategie

Im Gegensatz zu herkömmlichen Szenarien sind die Rollen im gegebenen Fall nicht so klar definiert. Eine Person kann sowohl als Subject wie auch als Object auftreten. Wobei jedes Object wiederum aus mehreren Subobjects (Kontaktinformationen einer Person) besteht, für die jeweils einzeln die Anforderungen für einen Zugriff definiert sind.

Die Definition der erforderlichen Rechte für den Zugriff auf ein Object spalten sich entsprechend auf diese Subobjects auf. Die Rechte eines Subject beim Zugriff auf ein Object sind wiederum durch die Beziehung zwischen Object und Subject definiert und damit bei jedem Zugriff unterschiedlich.

Einfache Modelle wie Mandatory Access Control und Access Control Lists können diesen mehrdimensionalen Anforderungen bereits nicht mehr gerecht werden. Eine Kombination dieser Strategien, wie es bei RBAC [8] durchgeführt wird, ist schon näher am gesuchten Resultat. Die Beziehung zwischen zwei Personen kann man sich als Rolle des Subjects beim Zugriff vorstellen. Dabei ist es wichtig zu beachten, dass die Bewertung der Beziehung vom Object zum Subject zählt und nicht umgekehrt. Nun kann für jedes Subobject definiert werden, welche Rollen darauf zugreifen dürfen.

Bei aktiven Personen ist der einfache Fall gegeben, dass das Object und nur dieses die erforderlichen Rechte auf eine Kontaktinformation definiert. Handelt es sich beim Object aber um eine passive Person, so hat diese nicht die Möglichkeit, die Zugriffsbeschränkungen auf die Information zu definieren. Daher wird in diesem Fall für den Zugriff eine vorhandene Beziehung zwischen Object und Subject vorausgesetzt. Diese Beziehung wiederum muss von mindestens einem anderen Subject, welches bereits eine Beziehung zum Object hat, bestätigt werden.

Um die Komplexität des Modells zu reduzieren, können die Zugriffsbeschränkungen nur auf Beziehungsarten gesetzt werden und nicht direkt auf Beziehungen zu einzelnen Personen. Dies führt dazu, dass man nicht einer Person gezielt den Zugriff auf eine Information verweigern kann. Dazu muss stattdessen die Beziehungsbewertung zu dieser Person entsprechend geändert werden.

4.5.3 Abuse Cases

Um auf mögliche Attacken gegen das System vorbereitet zu sein, sollen nachfolgend mögliche Angriffe analysiert werden. Dadurch soll es möglich sein, bereits beim Design der Plattform den möglichen Angriffsversuchen entgegen zu wirken, um das Risiko eines Schadens zu minimieren.

1. Ausnützen von Softwarelücken
Ist die verwendete Software auf dem Server nicht aktuell oder eine Sicherheitslücke noch nicht geschlossen, droht die Gefahr, dass über Sicherheitslücken Angreifer ins System eindringen können. Je nach Software und Lücke kann das ganze System fremdgesteuert und auf alle Daten auf dem Server zugegriffen werden.

Geplante Gegenmassnahme: Es muss darauf geachtet werden, dass sämtliche Software auf dem Server stets auf dem neusten Stand sind.

2. Zugang als andere Person
Ein Angreifer beschafft sich Zugang zum System, indem er sich als anderen Benutzer anmeldet. Hierzu muss er die Logindaten dieses Benutzers in Erfahrung bringen. Der Angreifer hat anschliessend Zugriff auf sämtliche Daten der Person, wie auch auf die Daten von dessen Kontakten.

Geplante Gegenmassnahmen: Den Benutzern muss die Bedeutung der Logindaten bewusst gemacht werden und sie müssen vor Phisingattacken gewarnt werden. Zusätzlich wäre ein Login über SSL wünschenswert, um das Auslesen der Logindaten auf Transportebene zu verhindern.

3. Direktzugriff auf die Datenbank
Ein Angreifer kann sich direkten Zugriff auf die Datenbank beschaffen. Sämtliche Daten sind les-, veränder- und löschbar.

Geplante Gegenmassnahmen: Der Zugriff auf die Datenbank soll möglichst gut geschützt sein. Die Webapplikation soll nicht mit einem Administratorbenutzer auf die Datenbank zugreifen, sondern mit einem speziellen Benutzer, der nur beschränkte Rechte besitzt. Idealerweise kann dieser Benutzer nur Stored Procedures ausführen.

Als weitere Schutzmassnahme sollen besonders sensible Daten, wie beispielsweise die Logindaten für externe SMS-Dienstleister, nur verschlüsselt abgelegt werden.

4. Session Hijacking
Eine laufende Benutzersitzung wird von einem Angreifer übernommen. Der Angreifer hat Zugriff auf sämtliche Daten der Person, sowie auf die Daten von dessen Kontakten.

Geplante Gegenmassnahmen: Einerseits soll auf technischem Weg dafür gesorgt werden, dass ein Session Hijacking erschwert wird. Andererseits sollen die Benutzer, welche sich zum Ende einer Benutzersitzung nicht abmelden, jeweils darauf hingewiesen werden, dass sie sich ausloggen sollen.

5. Fehlverhalten im System erzwingen
Durch Fehleingaben wird ein Fehler im System erwirkt. Je nach Art des Fehlers ist es dem Angreifer möglich, Informationen über das Innenleben des Systemes zu erhalten.

Geplante Gegenmassnahmen: Wenn das System nicht im Debug Modus betrieben wird, sollen keinerlei interne Fehlermeldungen sichtbar sein. Stattdessen sollen diese jeweils in neue Fehlermeldungen gekapselt werden, welche keine Informationen über das System preisgeben.

6. Querystring Manipulation
Der Angreifer erhält Informationen einer Person, obwohl er keine Rechte für diese Informationen hat, indem er Werte, die via URL übergeben werden, verändert.

Geplante Gegenmassnahmen: Um solche Attacken zu verhindern, sollen die Sicherheitsmechanismen zum Schutz der Kontaktdaten möglichst in den unteren Schichten des System implementiert werden.

Zusätzlich sollen kritische Querystrings verschlüsselt werden, um Attacken über Querystring Manipulationen erheblich zu erschweren.

Risikomatrix

Nachfolgend sind die analysierten Abuse Cases in einer Risikomatrix nach ihrem Gefahrenpotenzial und der Eintrittswahrscheinlichkeit angeordnet. Je dunkler das Feld, auf dem ein Abuse Case eingeordnet ist, umso dringender sind die Massnahmen gegen diese Attacken.

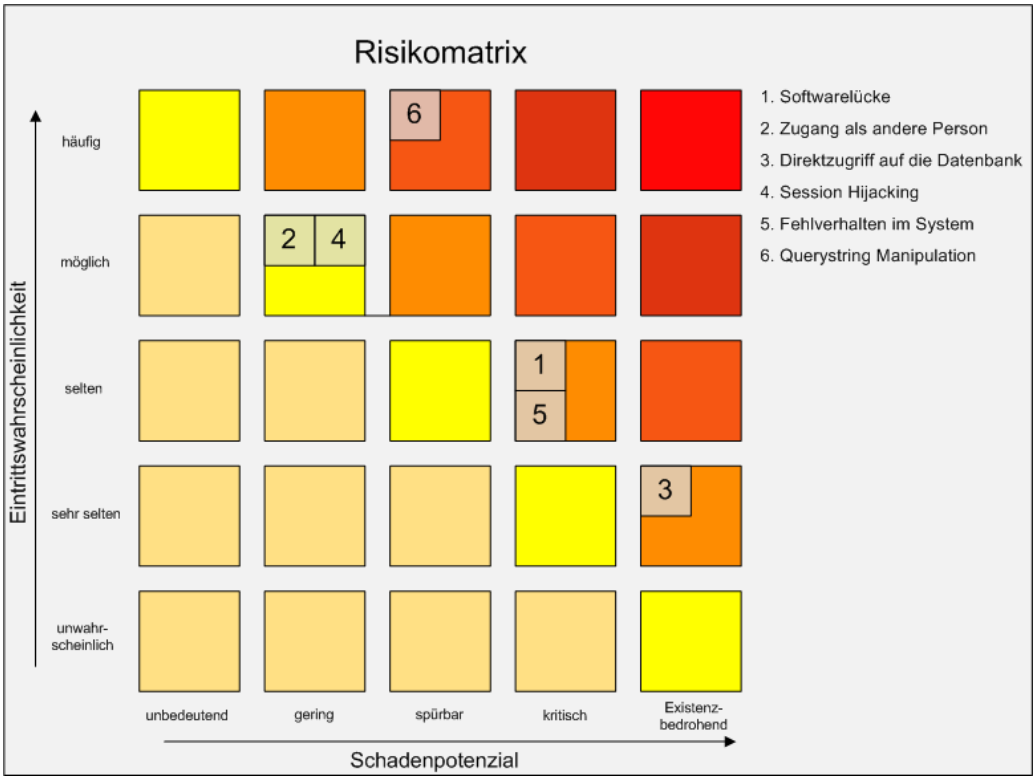

Abbildung 12: Risikomatrix der Abuse Cases

4.6 Messaging

Die Kommunikation zwischen Personen wird immer wieder dadurch erschwert, dass keine aktuelle Emailadresse eines Kontaktes verfügbar ist. Emailadressen ändern sich mit der Zeit oder werden nicht mehr regelmässig vom Besitzer abgerufen. Es soll deshalb möglich sein, einer anderen Person eine Nachricht zukommen zu lassen, ohne dass deren aktuelle Emailadresse bekannt ist.

Vorteile

- Emailadresse des Empfängers muss nicht bekannt sein

Nachteile

- Versand nur über die Plattform möglich
- Nicht ohne weiteres in externen Email Programmen nutzbar

4.6.1 Gruppen Emailadresse

Will man allen Mitgliedern einer Gruppe von Personen eine Email schreiben, steht man schnell vor dem Problem, dass man nicht von jeder Person eine aktuelle Emailadresse besitzts. Es soll daher

möglich sein, pro Gruppe eine Verteiler-Emailadresse zu erstellen. Sendet nun ein Mitglied dieser Gruppe eine Email an diese Verteiler-Adresse, wird das Email automatisch an alle Mitglieder der Gruppe weitergeleitet. Treten neue Personen der Gruppe bei, werden diese automatisch auch über diese Adresse erreichbar.

4.6.2 Personalisierte Nachricht

Schreibt man eine Nachricht an mehrere Benutzer, ist es meist nicht möglich, diese persönlich anzusprechen. Der Benutzer soll die Möglichkeit erhalten, Platzhalter für Namen und Vornamen in seiner Nachricht einzufügen, die vom System beim Versenden automatisch mit den korrekten Werten abgefüllt werden.

4.6.3 SMS Kommunikation

Viele Mobilfunkanbieter stellen ihren Kunden ein Kontingent an SMS Nachrichten gratis zur Verfügung. Dabei ist es dem Kunden möglich, über eine Webseite SMS unter seiner Nummer zu versenden.

Meist muss man auf diesen Seiten die Nummer des Empfängers eingeben oder kann diese in einem internen Adressbuch speichern. Folglich müssen diese Daten aber auch auf jeder dieser Plattformen verwaltet und aktuell gehalten werden.

Das Ziel soll nun sein, die Kontaktdaten von Relate direkt mit den SMS-Versand-Anbietern zu verknüpfen um den Adressverwaltungsaufwand auch hier zu vereinfachen. Idealerweise soll der SMS-Versand direkt über die Relateplattform und der eigentliche Versand über den jeweiligen Anbieter im Hintergrund stattfinden.

Zusätzlich zu den Angeboten der Mobilfunkanbieter gibt es noch eine Reihe unabhängiger Anbieter, welche den SMS-Versand gegen Entgelt erlauben. Um nicht Personen, deren Mobilfunkanbieter keinen Gratisversand anbieten, auszuschliessen, soll auch ein solcher Bezahldienst integriert werden.

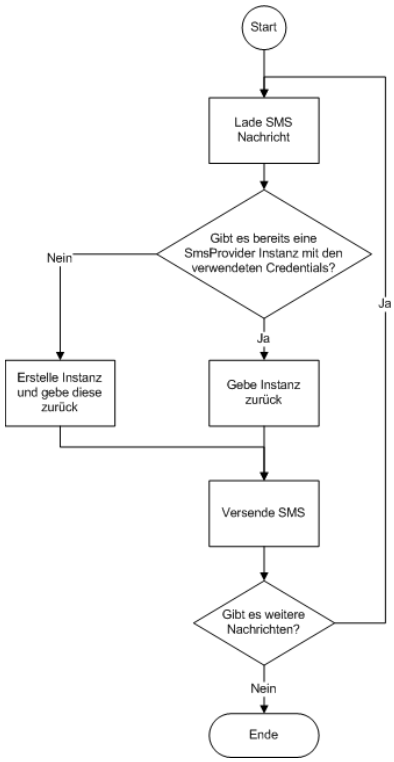

Abbildung 13: Ablauf des SMSDaemon beim Versand von SMS Nachrichten

Vorteile

- Nummer des Empfängers muss nicht bekannt sein
- Mehrere Empfänger gleichzeitig erreichbar
- Zentrale Verwaltung, kein zusätzliches Adressbuch

Nachteile

- Keine gemeinsame Schnittstelle zwischen den Anbietern
- Kostenloser Versand meist nur mit Mobilfunkvertrag möglich
- Kostenloser Versand meist nur an Nummern aus dem jeweiligen Land möglich

4.6.4 Zeitversetzter Versand

Beim klassischen Email- und SMS-Versand wird eine Nachricht stets sofort nach dem Erfassen an den Empfänger gesendet. In einigen Fällen wäre es aber von Vorteil, eine Nachricht erst zu einem definierten Zeitpunkt zu versenden. Dies ermöglicht das vorprogrammieren von Geburtstagsglückwünschen und Erinnerungsnachrichten, bevor man aus dem Haus geht und vieles mehr.

Diese Funktion ist ein klarer Vorteil gegenüber anderen Versandprogrammen, welche diese Funktion mangels zentralem Server nicht anbieten können. Um nämlich zeitversetzten Versand zu ermöglichen, müssen die Nachrichten zentral auf dem Server gespeichert werden, wo ein Scheduled Task regelmässig die versandbereiten Nachrichten verschickt. Dadurch ist der eigentliche Versand vom Erfassen der Nachricht entkoppelt, was dazu führt, dass der Benutzer seine Benutzerdaten für die Versandplattform zwingend im System speichern muss.

Vorteile

- Nachrichten können zu einem beliebigen Zeitpunkt versendet werden
- Erfassen und Senden der Nachricht sind nicht gekoppelt

Nachteil

- Für den zeitversetzten Versand von SMS Nachrichten müssen die Zugangsdaten auf dem Server gespeichert werden

4.7 WCF Schnittstelle

Es soll eine WCF-Schnittstelle erstellt werden, die externen Programmen den Zugriff auf die Kontaktdaten erlaubt. Dabei soll darauf geachtet werden, dass an einem zentralen Punkt definiert werden kann, welche Felder und Funktionen der Service zur Verfügung stellt.

Aus Sicherheitsgründen sollen vorerst nur lesende Zugriffe erlaubt sein. Für eine spätere bidirektionale Synchronisation soll der Service aber einfach erweitert werden können.

Die Authorisierung soll wie bei der Webseite über den Benutzernamen und ein Passwort ausgelöst und durch den Erhalt eines SecurityTokens abgeschlossen werden. Mehr Informationen zu diesem Thema findet sich im Kapitel 5.5 auf Seite 51.

Für den Endpoint soll ein BasicHttpBinding verwendet werden. Diese ermöglicht einfache Kommunikation mit dem Service über SOAP.

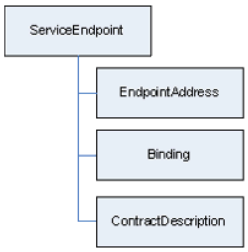

Abbildung 14: Ein WCF Endpoint definiert sich über seine Adresse, das Verwendete Binding und die DataContracts

4.8 Analyse der Studienarbeit

Im Umfang einer Studienarbeit wurde bereits ein lauffähiger Prototyp der Plattform erstellt [1]. Da diese Arbeit auf diesen aufbauen soll, wurde die Codebasis einer umfassenden Analyse unterworfen.

4.8.1 Sicherheit

Die Regelung der Sichtbarkeit von Kontaktdaten ist in der Studienarbeit im Business Layer, teilweise aber auch im Web Layer geregelt. Dies führt dazu, dass der Programmierer bei jedem Zugriff auf

sensible Kontaktdaten jeweils daran denken muss, die Rechtmässigkeit dieses Zugriffes zu überprüfen. Wird diese Regulierung vergessen, können sämtliche Daten abgerufen werden.

In der Codebasis ist keine Unterscheidung von verschieden stark bewerteten Beziehungen eingebaut. Besteht eine Beziehung zwischen zwei Personen, so sind alle Daten des anderen sichtbar. Es besteht keine Möglichkeit, den Zugriff zu regulieren. Nicht verknüpfte Personen sehen lediglich die Adresse einer anderen Person, die restlichen Daten bleiben verborgen.

Folgerung

Die Sicherheit soll zentral an einem Punkt definiert sein. Dazu muss die bestehende Basis grossflächig überarbeitet werden. Die Überprüfung für den Zugriff auf sensible Daten soll in der Datenbank direkt verankert werden, wodurch ein Umgehen der Sicherheitsmechanismen verhindert werden soll.

4.8.2 Codequalität

Obwohl die Qualität der Codebasis relativ gut ist, hat das agile Vorgehen sowie das Feature-Driven Design der Studienarbeit doch zu einigen Unschönheiten im Code geführt.

Die Gatewayschicht hat fast keine eigene Funktionalität, sondern dient nur als Indirektion zwischen dem Business-Layer und dem Frontend.

Zusätzlich fehlt ein klarer Einstiegspunkt für den Zugriff auf die Businesslogik. Jede Frontend-Komponente muss mehrere Gatewayklassen instanzieren, um die volle Funktionalität nutzen zu können. Dies führt zu einer sehr hohen Kopplung zwischen den einzelnen Schichten. Zudem wurde immer unklarer, bei welchem Controller eine gewisse Funktionalität zu finden ist.

Folgerung

Als klarer Einstiegspunkt soll auf Businesslayer Ebene ein Facade-Controller eingeführt werden, welcher den Zugriff auf die benötigte Funktionalität und die weiteren Controller zur Verfügung stellt. Auf der Ebene der Webseite soll eine ControllerFactory erstellt werden, welche pro Request jeweils denselben Controller zurück liefert.

Die Controller sollen klare Namen erhalten, welche automatisch suggerieren, welche Funktionalität darin zu finden ist.

4.8.3 Applikations-Performance

Die Codebasis hat einige Performance Schwachstellen. Wenn eine Kontaktinformation abgerufen wird, werden die Resultate aus der Datenbank mittels LINQ to SQL automatisch auf ein aus dem Datenbankschema generiertes Domainobjekt abgebildet. In der Businessschicht werden diese Objekte dann wiederum, mithilfe eines auf Reflection basierenden DTO-Converters, auf Data Transfer Objekte gemappt.

Eine weitere Performance-Bremse war der benutzte MSDTC Service. Um Transaktionen über mehrere Funktionen und deren Datenkontexte verwenden zu können, wurde der Microsoft Distributed Transaction Service verwendet. Dieser implementiert aber das sehr langsame Two-Phase Commit Protokoll, welches in unserem Fall gar nicht benötigt wird, da die Datenbankzugriffe zwar auf verschiedene Funktionen, nicht aber auf verschiedene Tiers verteilt sind.

Das Formulieren der Datenbankabfragen im LINQ Syntax führt zusätzlich dazu, dass oft mehr Informationen aus der Datenbank geholt werden als benötigt. Ein Kontakt kann nur mit allen oder keinen angehängten Kontaktinformationen abgerufen werden (DataLoadOptions). Zusätzlich sind für das Löschen oder Ändern eines Datensatzes stets mehrere Datenbankzugriffe nötig, da LINQ verlangt,

dass zuerst der aktuelle Datensatz aus der Datenbank geladen wird, um diesen dann zu verändern oder als gelöscht zu markieren.

Folgerung
Statt dem doppelten Mapping der Daten mittels LINQ und DTO-Converter sollen die Daten direkt aus der Datenbank auf die gewünschten DTO's gemappt werden. Auf relativ langsame Reflections soll ganz verzichtet werden.

Um das Verwenden des MSDTC Services überflüssig zu machen, sollen die Methoden auch mit einem mitgelieferten Datenkontext aufgerufen werden. Dadurch ist eine Transaktion über mehrere Methodenaufrufe ohne Probleme möglich.

Es soll möglich sein, einen Datensatz mit nur einem Datenbankzugriff zu löschen oder zu verändern. Das Lost-Update Problem kann in diesem speziellen Fall vernachlässigt werden, da es sehr unwahrscheinlich ist, dass zwei Personen dieselbe Kontaktinformation gleichzeitig bearbeiten.

4.8.4 Schlussfolgerung aus der Analyse der Codebasis

Aufgrund der Analyse hat sich gezeigt, dass die Codebasis ein umfassendes Refactoring benötigt, um beim weiteren Ausbau der Plattform nicht unübersichtlich und zu langsam zu werden. Da sich die Änderungen durch alle Schichten ziehen, soll ein Refactoring des bestehenden Codes schichtweise von unten nach oben stattfinden.

Nachfolgend findet sich eine Zusammenfassung der wichtigsten geplanten Änderungen an der bereits bestehenden Codebasis:

- Zentrale Regelung der Kontaktinformationssicherheit. Realisierung auf Datenbankebene mittels Stored Procedures.
- Schaffung eines Facade-Controllers für einen zentralen Einstiegspunkt
- Einführung einer Controller-Factory um Mehrfachinstanzierung des Facade-Controllers zu verhindern.
- Daten sollen direkt von der Datenbank auf DTO's gemappt werden
- Keine Verwenden des MSDTC
- Änderungen und Löschen von Datensätzen soll mithilfe von einem einzigen Datenbankzugriff möglich sein.

Ausführliche Beschreibungen zu den ausgeführten Refactorings finden sich im Kapitel 6 auf Seite 56.

5 Design

5.1 Physische Architektur

5.1.1 Nachteile der bestehenden Lösung

Der in der Studienarbeit verwendete Virtuelle Server der HSR erfüllte die Anforderungen für diese Bachelorarbeit nicht mehr. Durch die Sicherheitsbestimmungen der HSR ist kein Emailversand über einen SMTP Server aus der DMZ heraus möglich. Dadurch musste in der Studienarbeit ein unschöner Umweg über einen Webservice auf einen externen Server über den offenen Port 80 gemacht werden. Die Lebensdauer des HSR-Servers ist zudem nur bis zum Abschluss dieser Arbeit garantiert, danach wäre ein Umzug auf einen anderen Server zwingend notwendig geworden. Ein weiterer Nachteil der bestehenden Lösung ist das Fehlen eines zentralen Nameservers, auf dem der Domainnamen eingetragen werden könnte. Die HSR bietet hier leider keine vernünftige Lösung, weswegen bisher ein externer Nameserver eingesetzt wurde.

5.1.2 Anforderungen an das neue System

Die Anforderungen an das neue System sollten alle Negativpunkte der bestehenden Lösung ausmerzen und weitere Features enthalten, die zukünftig von Nutzen sein könnten. Die Anforderungen an das neue System wurden wie folgt definiert:

- Schweizer Hoster, um kurze Antwortzeiten sicherzustellen
- Aktuelle Technologie (.NET 3.5 SP1 und SQL Server 2008)
- WCF Service Hosting möglich
- Integrierter Mail Server für den Versand von Emails
- Einfache Konfiguration des Servers
- Schnelle Hilfeleistung durch den Support
- Zugriff auf Datenbank und Server Administration auch von ausserhalb der HSR möglich
- Cronjobs / Scheduled Tasks Unterstützung

5.1.3 Suche nach einem Hoster

Als Herausforderung stellte sich die Suche nach einem geeigneten Hoster heraus, der sämtliche Anforderungen erfüllt. Das Angebot an günstigen auf Windows basierenden Hostern ist im Vergleich zu Linux-Hostern sehr klein. Die Auswahl an privaten Linux Hostern ist sehr breit gefächert und die Preise dadurch gering. Die Windows Hoster konzentrieren sich mit ASP.NET mehr auf den professionellen Bereich und sind dadurch auch leicht teurer. Nach einer kurzen Recherche beschränkten wir uns auf zwei Hoster, die wir genauer betrachteten.

Hostfactory
Hostfactory bietet mit Plesk eine einfache Administration des Servers und reagierte auf unsere Anfragen sehr schnell und kompetent. Hostfactory erfüllt alle unsere Anforderungen, es bietet genügend Speicherplatz, die Wahl zwischen MS SQL und MySQL, die komfortable Administration des Servers und viele weitere Vorzüge.

Green.ch
Green.ch ist ein bekannter Anbieter von Hosting Angeboten sowie DSL Internetzugängen. Leider ist bei diesem Angebot kein MS SQL Server von Grund auf im Angebot integriert. Das Angebot wurde dadurch um einiges teurer, wodurch wir der Konkurrenz den Zuschlag gaben.

5.1.4 Budget

Da wir den Server selbst finanzieren, wollten wir einen Hoster, der ein gutes Preis/Leistungs-Verhältnis aufweist. Mit 6.95 pro Monat für das ASP-Basic Edition Angebot von Hostfactory erreichten wir dieses Ziel und haben nun einen günstigen Hoster, der all unsere Anforderungen erfüllt.

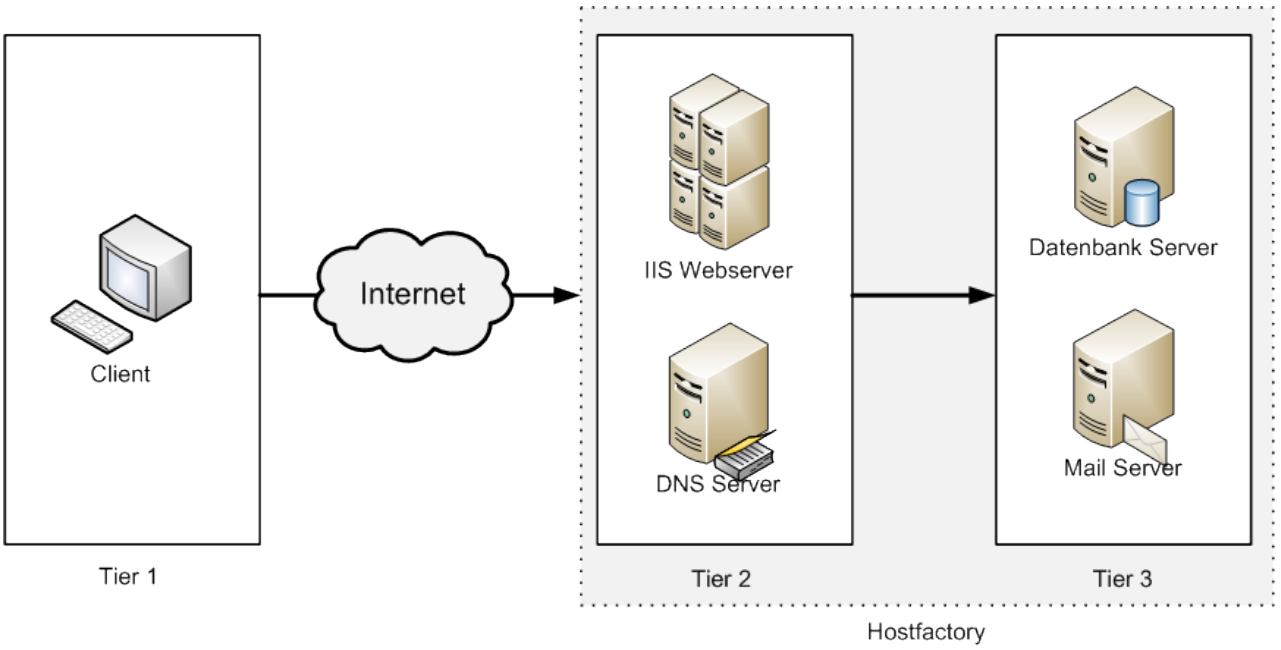

Abbildung 15: Physikalische Verteilung der Applikation

5.2 Logische Architektur

5.2.1 Architekturdiagramm

Die Codebasis hat wie unter Kapitel 4.8 auf Seite 29 beschrieben eine Gatewayschicht, welche neben dem Umwandeln der LINQ Objekt zu DTO's [3], keine eigentlich Funktionalität hat. Da diese Umwandlung mit dem direkten Mapping der Daten aus der Datenbank auf POCO's überflüssig wurde, konnte der Gatewaylayer aus der logischen Architektur entfernt werden.

Zusätzlich ist ein Paket für den MessagingDaemon dazugekommen. Dieses wird als ausführbare Datei auf den Server geladen, um dann mittels CronJob aktiviert zu werden und die versandbereiten Nachrichten zu verschicken.

Da ein Grossteil der Businesslogik nun in den Stored Procedures der Datenbank definiert ist, wurde diese nun ebenfalls ins logische Schichtenmodell aufgenommen.

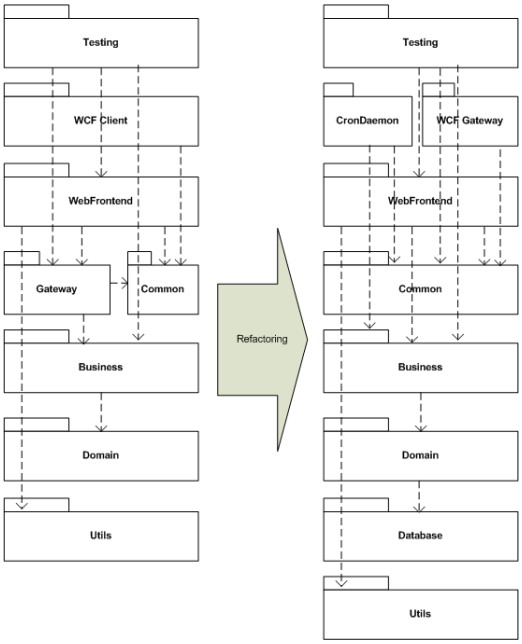

Abbildung 16: Veränderung der Logische Sicht Studienarbeit-Bachelorarbeit

5.2.2 Utils Layer

Diese Schicht stellt nützliche Hilfsklassen und Methoden zur Verfügung, wie beispielsweise eine Abstraktion des Querystring, Verschlüsselungsfunktionen, aber auch Webcrawler zum Versenden von SMS Nachrichten.

5.2.3 Database Layer

Die Datenbank speichert einerseits die Daten und stellt passende Views darauf zur Verfügung. Zudem stellt sie auch deren Konsistenz sicher und implementiert Mechanismen zur Überprüfung des Zugriffes auf die Kontaktinformationen.

5.2.4 Domain Layer

Der Domain Layer definiert das Mapping der POCO's auf das Datenbankmodell und stellt dem Business Layer Methoden für den Zugriff auf die Stored Procedures zur Verfügung.

5.2.5 Business Layer

Der Business Layer stellt Methoden für die oberen Schichten zur Verfügung. Er besitzt einen Facade-Controller, welcher den Zugriff auf die verschiedenen Methoden und Controller ermöglicht. Nähere Informationen zum Aufbau dieser Schicht findet sich unter 5.4.7 auf Seite 48.

In dieser Schicht wurde auch der Authentifizierungsmechanismus eingebaut, welcher die Identität eines Aufrufers überprüft und damit dann auf die Datenbank zugreift. Die Autorisierung der einzelnen Zugriffe übernimmt dann die Datenbank. Mehr zu dieser Aufteilung findet sich unter 5.6.4 auf Seite 54.

5.2.6 Common Layer

Der Common Layer stellt gemeinsame Komponenten zur Verfügung, welche die verschiedenen Schichten benötigen, um mit einer möglichst kleinen Kopplung miteinander kommunizieren zu können. Dazu gehören neben den Interfaces der POCO's auch Fehlermeldungen, Enums und Ressourcendateien mit Übersetzungen von Begriffen.

5.2.7 WebFrontend

Diese Schicht beinhaltet die ASP.NET Webseite, welche aufgeteilt ist in Webcontrols, Dateien für das Design (CSS, Bilder), Implementationen der Kommunikationsinterfaces, Erweiterungsfunktionen der Interfaces und natürlich die eigentlichen ASPX Seiten.

5.2.8 CronDaemon

Diese Schicht ist für das Versenden der SMS- und Emailnachrichten verantwortlich. Sie kann in eine ausführbare Datei kompiliert werden, welche auf den Webserver geladen und periodisch ausgeführt werden kann.

5.2.9 WCF Gateway

Über den WCF Gateway kann von externen Programmen aus auf Funktionen des Systems zugegriffen werden. Dies soll eine einfache Einbindung der Daten in andere Programme ermöglichen.

5.2.10 Testing

Die Testingschicht hat die Aufgabe alle Layer mittels möglichst automatisch ablaufender Tests zu testen und deren Funktionsfähigkeit laufend zu überprüfen.

5.3 Datenbankmodell

Das komplette Datenbank Modell ist durch die zusätzlichen Aufgaben enorm gewachsen und beinhaltet bereits 33 Tabellen. Der Übersicht halber wird deshalb das Modell in funktionalen Teilbereichen einzeln erläutert.

Die nachfolgenden Kapitel behandeln:

1. Die Speicherung von Personen und ihrer Kontaktdaten
2. Die für Messaging benötigten Tabellen
3. Die Abbildung von Beziehungen, Gruppen und Gruppierungen auf die Datenbank
4. Die für die Sicherheitseinstellungen der Kontaktdaten benötigten Tabellen

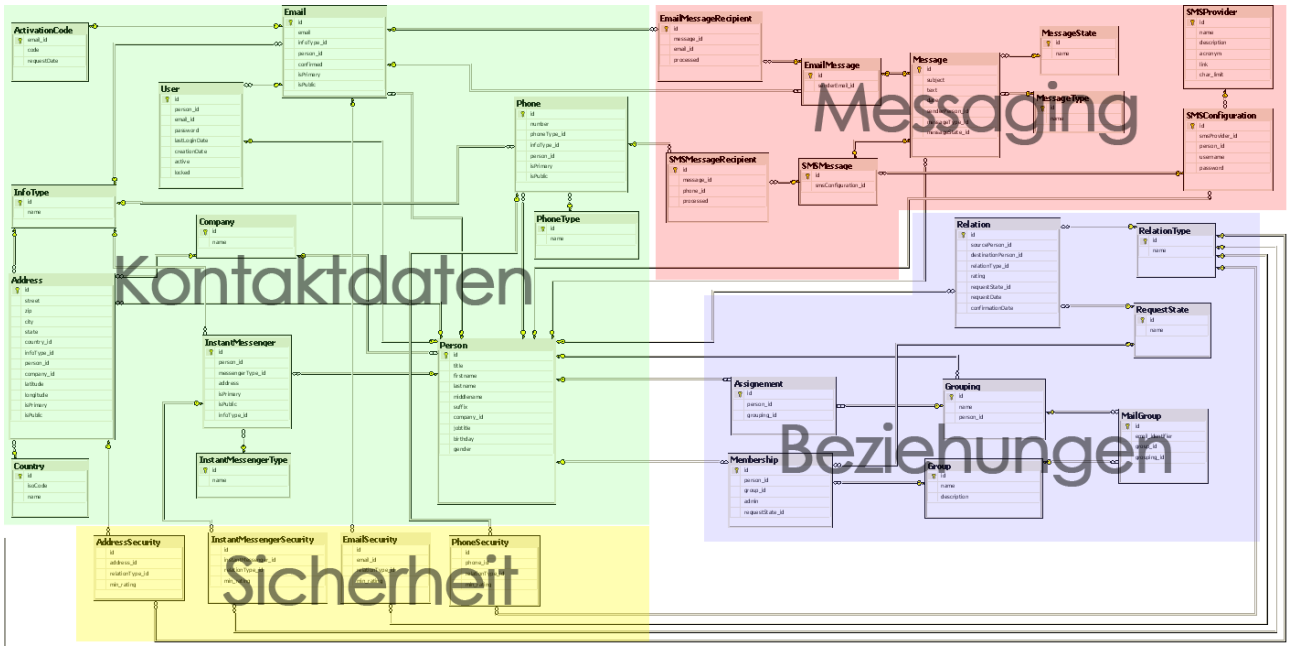

Abbildung 17: Die vier Bereiche der Datenbank im Überblick

5.3.1 Kontaktdaten

Das Herzstück der Datenbank ist die Speicherung der einzelnen Kontaktdaten. Es wurde darauf geachtet, dass die Datenstruktur die Kontaktdaten möglichst wenig einschränkt. Nachfolgend werden die einzelnen Tabellen und deren Funktion genauer beschrieben.

Person Die Tabelle speichert die Kerndaten zu einer Person. Dazu gehören Name, Geschlecht, Geburtstag und Ähnliches.

User Wenn eine Person aktiv bei Relate dabei ist, dann wird ein Benutzer für diese Person erstellt. Erst nach einer Aktivierung per Email wird der Benutzer aktiviert (active). Zusätzlich wird hier der Passwort-Hash, sowie die Zeitpunkte der Registrierung und des letzten Einloggens gespeichert. Als Benutzername wird eine Emailadresse der Person definiert.

InfoType Der Info-Typ bezeichnet die Art einer Kontaktinformation (Privat, Geschäftlich, Anderes)

Address Bei der Adresse werden zusätzlich zu den klassischen Adressenangaben wie Strasse und Ort auch noch die Geokoordinaten gespeichert. Dies ermöglicht sowohl ein einfacheres Visualisieren der Daten, als auch die Möglichkeit Distanzen zu berechnen. Pro Person und InfoType muss jeweils eine Primäradresse vorhanden sein. Dies wird mittels Triggers sichergestellt.

Country Die Landestabelle beinhaltet sämtliche Ländern mit ihren zugehörigen ISO Codes.

InstantMessenger Diese Tabelle speichert die Informationen von Instant Messenger Diensten. Später wird diese Tabelle eventuell erweitert, um eine automatische Autorisierung mit den jeweiligen Diensten zu ermöglichen.

InstantMessengerType Diese Tabelle speichert eine Auflistung der unterstützten Instant-Messaging-Dienste.

Phone Die Phone Tabelle speichert Telefonnummern, neben dem InfoType werden die Telefonnummern zusätzlich noch nach PhoneType unterschieden.

PhoneType Der Telefon-Typ definiert, ob es sich bei einer Nummer um eine Fax-, Festnetz-, oder Mobilnummer handelt.

Email Jede Emailadresse sollte bestätigt werden. Dies wird in der Tabelle vermerkt. Pro Person und InfoType muss jeweils eine Emailadresse als primär gekennzeichnet sein.

ActivationCode In dieser Tabelle werden generierte Codes gespeichert, welche zur Überprüfung der Emailadresse benutzt werden. Zusätzlich wird das Datum des Codes gespeichert, um diese beispielsweise nach einer gewissen Zeit löschen zu können.

Konsistenzbedinungen

Delete Constraints Wenn eine Person gelöscht wird, sollen alle zugehörigen Kontaktdaten (User, Address, InstantMessenger, Phone, Email) ebenfalls gelöscht werden.

IsPrimary Um einen einfachen Zugriff auf eine Hauptkontaktinformation einer Person zu ermöglichen, muss stets sichergestellt werden, dass es pro Person und InfoType nur eine primäre Adresse (resp. Phone, Email und InstantMessenger) gibt. Dies soll mithilfe eines AfterUpdate-/AfterInsert-Triggers sichergestellt werden. Genauere Informationen zur Implementation der Triggers finden sich im Kapitel 7.1.

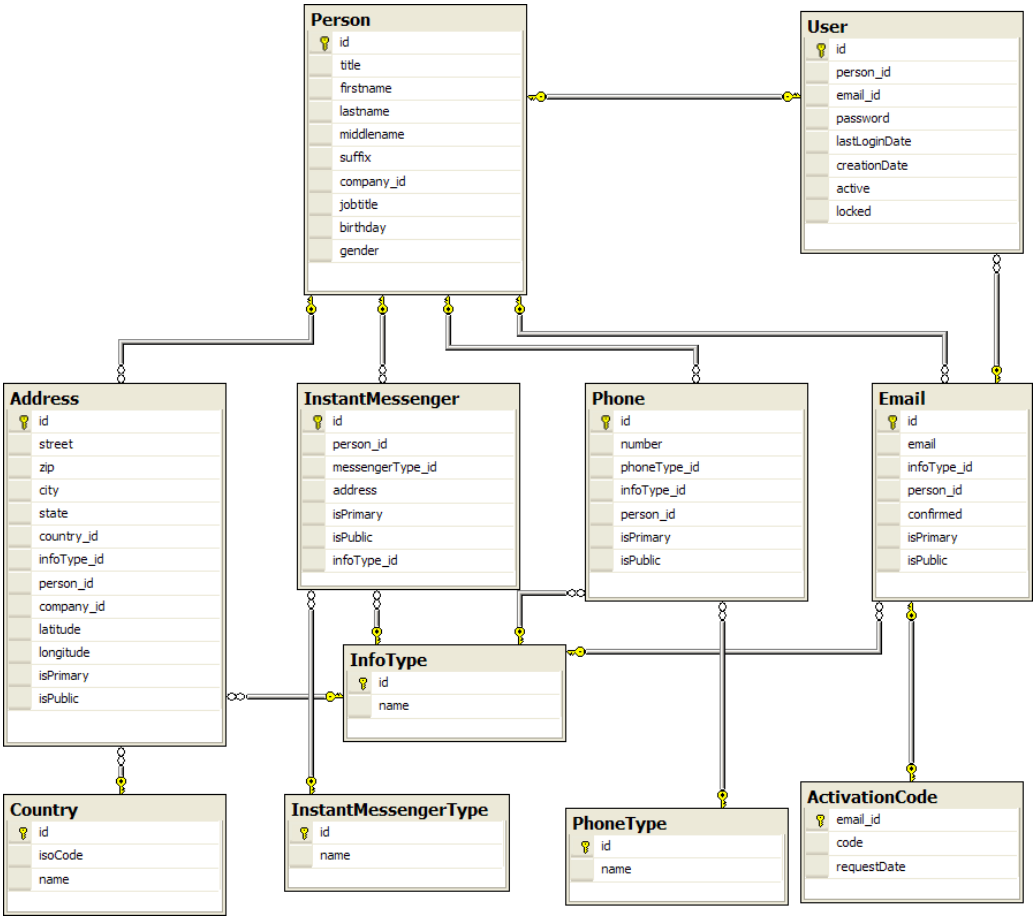

Abbildung 18: Entity Relationship Model der Kontaktdaten

5.3.2 Messaging

Um das Versenden, Abspeichern und später auch das Empfangen von SMS- und Email-Nachrichten zu ermöglichen, wird ebenfalls eine umfassende Datenstruktur benötigt. Nachfolgend werden die einzelnen Tabellen und ihre Funktion genauer erläutert. Auf die Tabellen Email, Phone und Person wird nicht nochmals eingegangen, da diese bereits in Kapitel 5.3.1 auf Seite 36 beschrieben wurden und hier nur zum besseren Verständnis aufgeführt sind.

Message Dient zur Speicherung von Nachrichten, egal ob SMS oder Email. Dies soll eine gemeinsame Behandlung aller Nachrichten vereinfachen. Gespeichert werden Titel, Text, Sendedatum, eine Verknüpfung zu der Empfangsperson sowie ein Message-Typ und Message-Status.

MessageTyp Dient zur Unterscheidung der verschiedenen Nachrichtenarten (SMS, Email)

MessageState Je nach Status wird die Nachricht verschieden behandelt.

Inbox Empfangene Nachricht

Draft Gespeicherte Nachricht, die noch nicht versendet werden soll

Outbox Versandbereite Nachricht, welche nun verschickt werden soll

Sent Versendete Nachricht

EmailMessage Dies ist eine Kindertabelle von Message, welche emailspezifische Informationen abspeichert. Der Primärschlüssel ist zugleich Fremdschlüssel auf die Tabelle Message. Jede Emailnachricht hat einen Verweis auf die Absender-Emailadresse.

EmailMessageRecipient Pro Empfänger wird ein EmailMessageRecipient erstellt. Diese Tabelle enthält ein Flag (processed), ob die Nachricht bereits verarbeitet wurde.

SMSMessage Dies ist eine weitere Kindertabelle von Message, welche SMS spezifische Informationen abspeichert. Auch hier ist der Primärschlüssel zugleich Fremdschlüssel auf die Tabelle Message. Zusätzlich enthält jede SMSMessage einen Verweis auf die verwendete SMSConfiguration, über die sie versendet werden soll.

SMSMessageRecipient Pro Empfänger einer SMS wird ein SMSMessageRecipient erstellt. Ein Flag (processed) zeigt wiederum den Verarbeitungsstatus der Nachricht.

SMSConfiguration Um fremde SMS-Dienstleister verwenden zu können, muss eine Person jeweils seine Credentials für den jeweiligen SMSProvider speichern.

SMSProvider Hier werden Informationen zu den verschiedenen zur Verfügung stehenden SMS-Dienstleister gespeichert. Neben Namen und Beschreibung findet sich auch ein Link zur Anmeldung sowie die Anzahl versendbarer Zeichen.

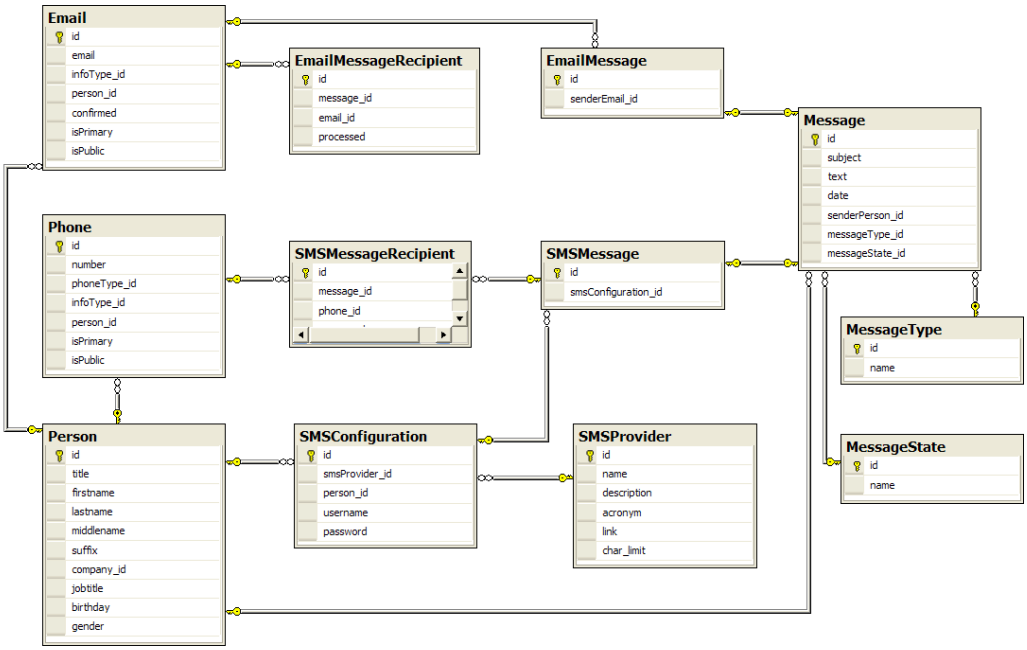

Abbildung 19: Entity Relationship Model der Nachrichtendienste

5.3.3 Beziehungen

Neben einfachen Beziehungen zwischen den Personen sollen auch Gruppen gespeichert werden können. Zusätzlich soll jeder Benutzer die Möglichkeit haben, seine Kontakte selbst zu gruppieren. Nachfolgend werden die einzelnen Tabellen zur Organisation der Beziehungen zwischen Personen erläutert.

Relation Diese Tabelle speichert eine Beziehung zwischen zwei Personen. Jede Beziehung hat eine Wertung, ein Erstellungs- und ein Bestätigungsdatum. Zu beachten ist zusätzlich, dass pro Beziehung jeweils zwei Einträge gemacht werden. Jede Person tritt einmal als sourcePerson und einmal als destinationPerson auf. Dies ermöglicht asymmetrische Beziehungen, welche andere Typen und Bewertungen haben.

RelationType Bezeichnet den Typ der Beziehung. Mögliche Werte sind: Friend, Business und Unkown

RequestState Definiert den Status einer Beziehung.

Requested Bestätigung der Beziehung ist noch ausstehend

Accepted Beziehung ist bestätigt

Rejected Beziehung wurde abgelehnt

Deleted Beziehung wurde gelöscht

Grouping Diese Tabelle speichert Gruppierungen von Kontakten, welche immer nur für die erstellende Person sichtbar sind. Deshalb hat Grouping stets einen Verweis auf die Person zu der sie gehört.

Assignement Jeder Kontakt in einem Grouping wird in einem Assignement gespeichert.

Group Diese Tabelle speichert Informationen wie Namen und Beschreibung von Gruppen.

Membership Jedes Mitglied einer Gruppe wird hier vermerkt. Zusätzlich wird gespeichert, ob das Mitglied über Adminrechte für diese Gruppe verfügt. Der RelationState gibt an, ob die Person dem Gruppenbeitritt bereits zugestimmt hat oder nicht.

MailGroup Mithilfe dieser Tabelle kann pro Gruppe oder Gruppierung eine Emailadresse erstellt werden, über welche dann sämtliche Gruppenmitglieder erreichbar sind.

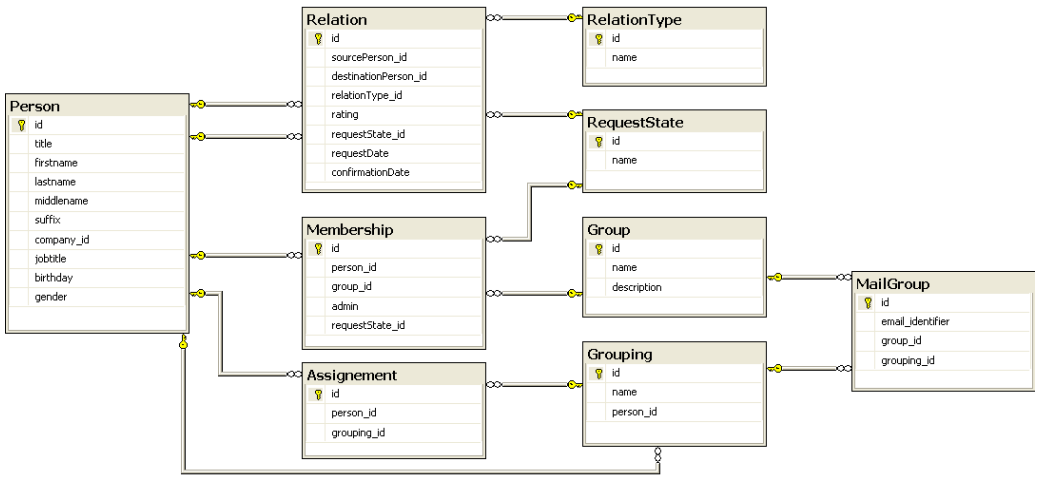

Abbildung 20: Entity Relationship Model der Beziehungstabellen

5.3.4 Sicherheit

Mithilfe der Sicherheitstabellen kann jeder Benutzer genau definieren, welche Informationen für wen sichtbar sind.

SecurityTable Für jede Kontakttabelle gibt es eine passende SecurityTable. Will man für eine Kontaktinformation den Zugriff beschränken, so erstellt man einen Eintrag in der passenden Security Tabelle.

relationType_id Definiert für welchen Informationstyp dieser Sicherheitseintrag gilt.

min_rating Definiert wie gut die Bewertung einer Beziehung mindestens sein muss, um diese Information sehen zu dürfen.

Kontaktinformationstabellen Diese Tabellen speichern wie unter 5.3.1 auf Seite 36 beschrieben Kontaktinformationen einer Person.

isPublic Dieses Feld ist auf true gesetzt, wenn keine Sicherheitseinstellungen gespeichert sind.

Konsistenzbedinungen

IsPublic Ein Trigger muss bei der Änderung der Sicherheitseinstellungen jeweils überprüfen, ob für eine bestimmte Kontaktinformation solche definiert sind. Wenn nicht, muss die entsprechende Information auf isPublic gesetzt werden und umgekehrt.

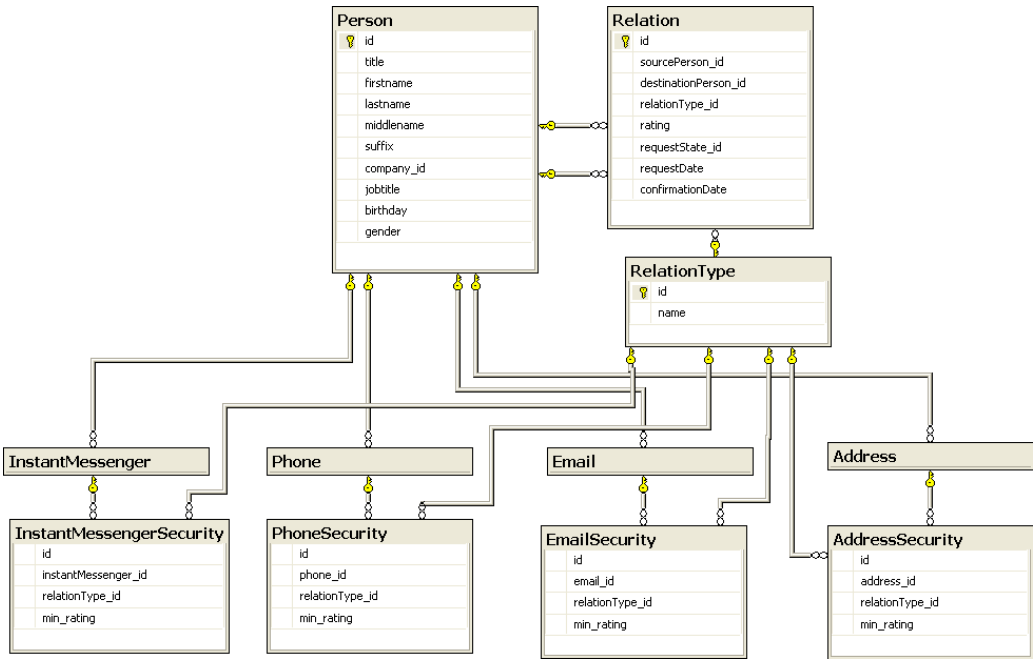

Abbildung 21: Entity Relationship Model der Sicherheitstabellen

5.4 Domainmodelle

5.4.1 Allgemein

Damit eine einfache Kommunikation zwischen den Schichten möglichst entkoppelt stattfinden kann, findet der Austausch von Daten ausschliesslich mit Interfaces statt. Jede Schicht arbeitet mit einer eigenen Implementation dieser Interfaces. Dadurch ist es möglich, die Implementation der unteren Schichten zu verbergen und nur die Signaturen der POCO's bekannt zu geben. Mehr zu diesem Thema findet sich im Kapitel 6.2 auf Seite 61.

5.4.2 Person

Eine Person und deren Repräsentation als Kontakt stellt den zentralen Punkt im Domainmodell dar. Der Zugriff auf die Person und deren Daten kann dabei auf zwei verschiedene Arten erfolgen.

Zum einen sind dies Listen von Personen, meist in Form von Suchresultaten. Dabei sind in der Regel nicht alle Daten notwendig, da eine Resultatliste nur beschränkt Platz zur Visualisierung der Kontaktdaten bietet. Die variable Anzahl an Kontaktdaten erschwert eine übersichtliche Darstellung der Daten zusätzlich. Ausserdem würde es einiges mehr Performance beanspruchen, auf jede Suchanfrage komplexe Objektbäume zurück zu liefern (siehe Abbildung 23 auf Seite 45). Daher wird in solchen Fällen ein abgeflachtes Objekt (ISimplePerson) geliefert.

Ist man an einer detaillierten Ansicht einer Person interessiert, macht es Sinn, sämtliche Kontaktdaten einer Person in einem Schritt zu erhalten. Eine detaillierte Abfrage liefert ein umfangreiches Personenobjekt (IDetailedPerson)

IPerson Dies ist das Basisinterface jeder Person. Es umfasst Allgemeine Daten wie Name, Geschlecht und Geburtstag wie auch Informationen über die Beziehung, die zwischen dem Anfrager und der angefragten Person besteht. Die gemeinsame Basisklasse ermöglicht es ISimplePerson und IDetailedPerson Objekte abstrakt als IPerson anzusprechen und somit Code-Redundanzen zu verhindern.

RequestState Definiert den Status der Beziehung zu dieser Person.

Requested Signalisiert, das eine Anfrage an die Person gesendet, jedoch noch nicht bestätigt wurde.

Accepted Eine gültige Beziehung zwischen zwei Personen existiert.

Rejected Die Anfrage wurde abgelehnt.

Deleted Die Anfrage wurde vom Anfrager zurückgezogen.

GenderType Definiert das Geschlecht einer Person

Female Person ist weiblich.

Male Person ist männlich.

RelationType Definiert die Art einer Beziehung

Unknown Es wurde kein Beziehungstyp festgelegt.

Friend Zwischen den Personen besteht eine freundschaftliche Beziehung.

Business Zwischen den Personen besteht eine geschäftliche Beziehung.

RelationRating Das Rating ermöglicht es, eine Beziehung zu bewerten. Dadurch können Kontakte unterschiedlich behandelt werden. Mittels Sicherheitsfunktionen kann beispielsweise definiert werden, wie gross das Rating mindestens sein muss, um auf eine bestimmte Information zugreifen zu können.

NotSet Keine Bewertung
FarFriend Ferner Bekannter
Familiar Bekannter
Colleague Kollege
Friend Freund
Pal Bester Freund

IDetailedPerson Umfasst alle Personendaten und bietet die Kontaktdaten als Listen an. Die Liste beinhaltet nur Kontaktdaten, für die man auch eine Einsichtsberechtigung besitzt.

ISimplePerson Stellt eine abgeflachte Repräsentation von IDetailedPerson dar. Dabei werden die Personendaten nicht als Liste geliefert, sondern es liegt jeweils eine Kontaktinformation pro Typ direkt als Wert vor. Dabei wird aus den vorhandenen Kontaktdaten eine ausgewählt, die vom Besitzer als primär definiert wurde und auch vom Anfrager eingesehen werden kann.

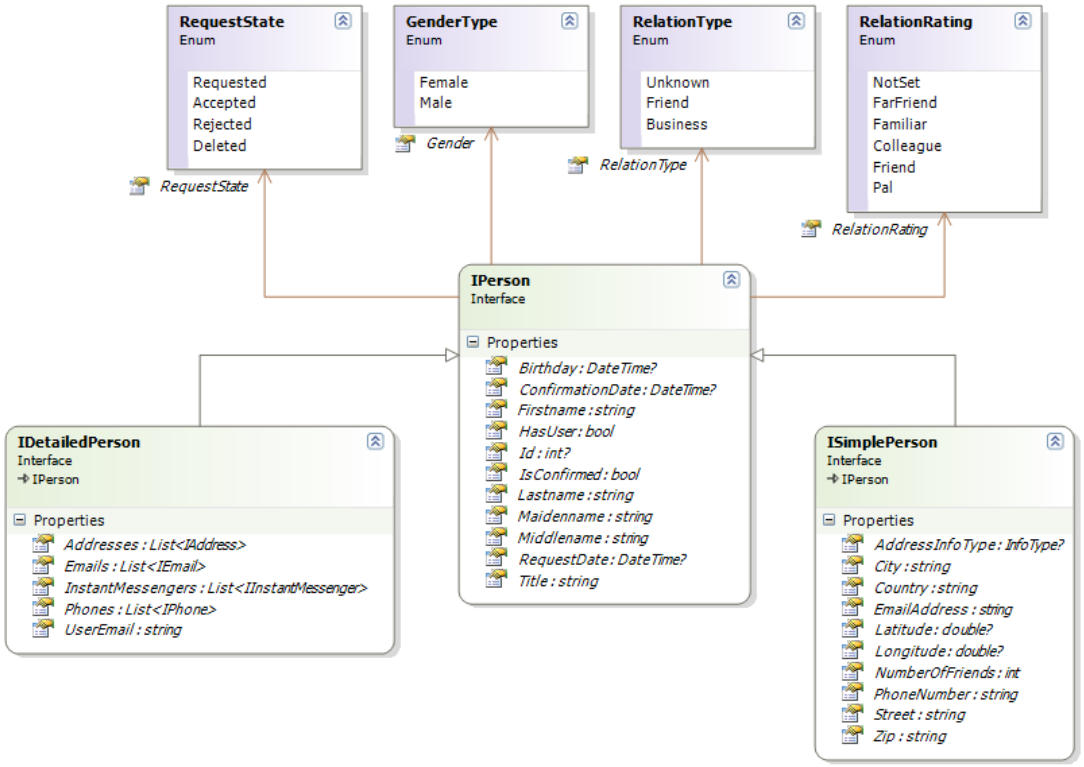

Abbildung 22: Personen-Interfaces und ihre Beziehung zueinander

5.4.3 Kontaktdaten

IContactDataObject Dient als Basisinterface für alle Kontaktdaten und fasst die übergreifenden Eigenschaften dieser zusammen. Dies ermöglicht die Entwicklung von generischen User Interface Controls zur Bearbeitung der Daten. Mehr Informationen dazu finden sich unter 7.7.1 auf Seite 73.

- **InfoType** Der InfoType definiert den Typ, dem die Kontaktdaten angehören können.
- **IsPrimary** Pro InfoType gibt es immer eine als primär definierte Kontaktinformation.
- **IsPublic** Definiert, ob eine Kontaktinformation für jeden in Beziehung stehenden Kontakt einsehbar ist. Ist IsPublic gesetzt, bedeutet dies gleichzeitig, dass keine SecuritySettings gesetzt sein dürfen.
- **SecuritySettings** Liste der definierten Sicherheitseinstellungen.

SecuritySetting Definiert die benötigten Beziehungseigenschaften für den Zugriff auf die Kontaktdaten. Mehr dazu im Kapitel 5.4.6 auf Seite 47

IInstantMessenger Repräsentiert Instant Messaging Adressen.

- **InstantMessengerType** Legt fest, um welchen Typ von Instant Messenger es sich handelt.

IEmail Die Emailadresse besitzt neben der eigentlichen Adresse noch folgende Eigenschaften:

- **Confirmed** Dieses Feld signalisiert, dass die Emailadresse vom Besitzer bestätigt wurde und es sich somit um eine gültige Emailadresse handelt.
- **IsUserEmail** Gibt an, ob diese Emailadresse von einer Person als Benutzername verwendet wird.

IAddress Hier werden die Details einer Adresse gespeichert. Besonders hervorzuheben sind dabei folgende Eigenschaften:

- **Latitude/Longitude** Speichern die Geografische Länge und Breite der Adresse. Dies ist für die Visualisierung, aber auch für die Berechnung von Distanzen wichtig.
- **Country** Länder Code nach ISO 3166 ALPHA-2 [11]

IPhone Repräsentiert eine Telefonnummer eines Kontaktes.

- **PhoneType** Es gibt verschiedene Arten von Telefonnummern (Telephone, Mobile, Fax), welche durch den PhoneType gekennzeichnet werden.

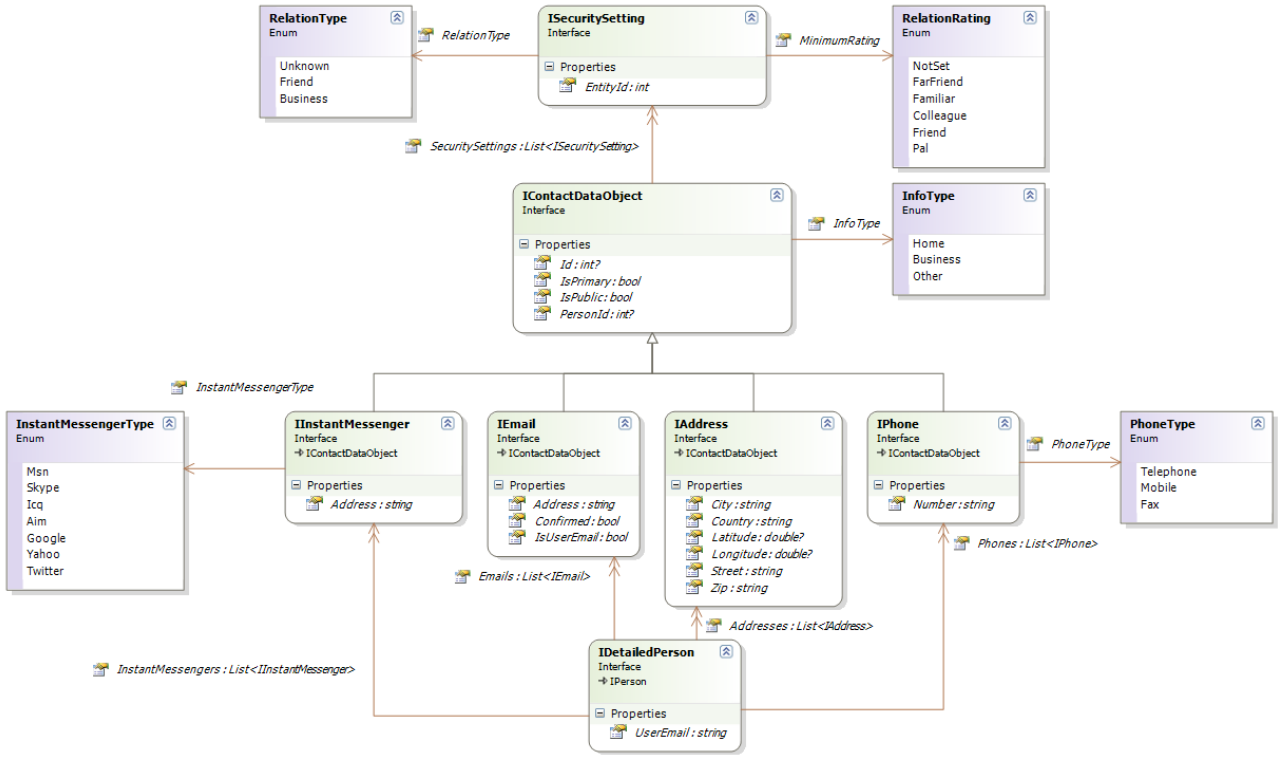

Abbildung 23: Objektbaum einer IDetailedPerson

5.4.4 Messaging

Das Versenden von SMS Nachrichten bedingt, dass der Benutzer seine Zugangsdaten preisgibt. Pro SMS Anbieter kann nur eine entsprechende Konfiguration gespeichert werden.

IMessage Repräsentiert eine Nachricht.

MessageBody Inhalt der Nachricht

MessageType Typ einer Nachricht

RecipientCount Anzahl der Empfänger

Recipients Liste aller Empfänger

SendDate Zeitpunkt, wann die Nachricht versendet werden soll

SenderId Id des Absenders

Subject Betreff der Nachricht

ISmsMessage Abgeleitet von IMessage. Repräsentiert eine SMS Nachricht.

SmsConfigurationId Verweis auf die verwendete SMS Konfiguration für den Versand der Nachricht.

IEmailMessage Abgeleitet von IMessage. Repräsentiert eine Emailnachricht.

EmailId Id der Email, die als Absender verwendet wird.

ISmsProvider Anbieter für den SMS Versand. Diese werden vom System vorgegeben und können durch die Benutzer nicht erweitert oder modifiziert werden.

Acronym Identifikation des Providers

CharLimit Anzahl der Zeichen, die für den Versand einer SMS zur Verfügung stehen.

Description Beschreibung des Anbieters und der Dienstleistung

Name Name des Anbieters / Dienstes

Url Url für weitere Informationen über den Anbieter / Dienst

ISmsConfiguration Konfiguration für den Versand von SMS Nachrichten über einen SMS Anbieter.

ProviderAcronym Verweis auf einen SMS Anbieter.

Username Benutzername für die Authentifizierung gegenüber dem SMS Anbieter.

Password Passwort für die Authentifizierung gegenüber dem SMS Anbieter.

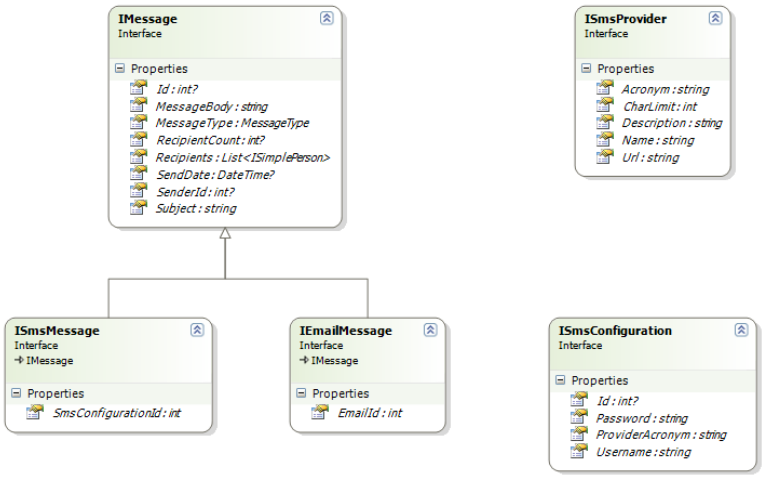

Abbildung 24: Messaging, SMS Provider und SMS Configuration Interfaces

5.4.5 Gruppen

IGroup Die Gruppe speichert neben dem Namen und einer Beschreibung der Gruppe eine Liste der Mitglieder.

IGroupMember Dieses von ISimplePerson ableitende Interface hat zwei zusätzliche Felder. Eines, um Gruppenadministratoren zu kennzeichnen und einen RequestState, um den Beitrittsstatus des Mitglieds zu speichern.

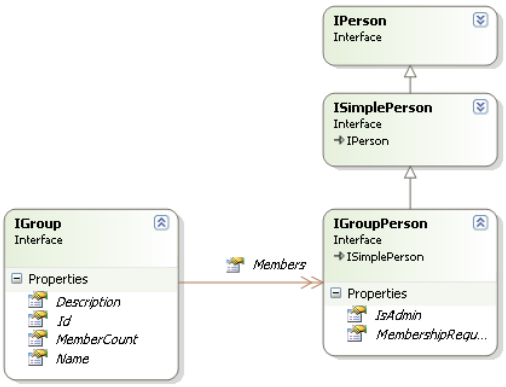

Abbildung 25: Eine Gruppe mit einer Liste von GroupMembers

5.4.6 Sicherheit

Der Zugriffsschutz auf die Kontaktdaten wird komplett auf Datenbankebene implementiert. Die Stored Procedures liefern jeweils nur die Daten, für welche man auch die entsprechenden Zugriffsrechte besitzt. Mehr dazu im Kapitel 5.3.4 auf Seite 41.

Um die Sicherheitseinstellungen zu konfigurieren, kann jedem Kontaktdatenobjekt eine Liste von SecuritySettings angehängt werden. Durch das Entfernen und Hinzufügen von SecuritySettings von und zu dieser Liste können die Sicherheitseinstellungen angepasst werden.

Das Abfragen der SecuritySettings ist aber nur für die eigenen Kontaktdaten möglich. Auch dies wird auf Datenbankebene sichergestellt.

Das Security Token wird zur Atorisierung gegenüber der Datenbank verwendet. Ohne gültiges Token ist kein Zugriff auf Personenbezogene Funktionen gestattet.

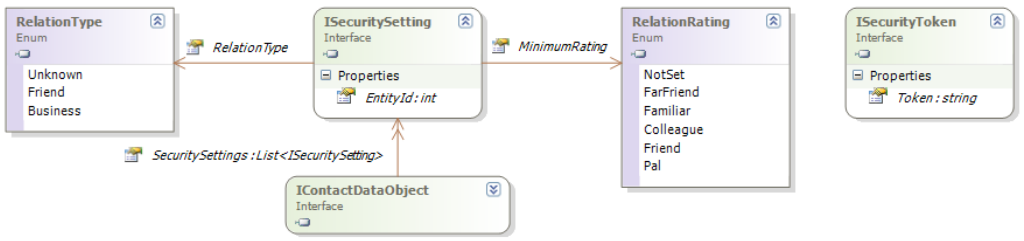

Abbildung 26: Eine Kontaktdatenobjekt hat eine Liste mit SecuritySettings angehängt.

5.4.7 Business Layer

Der Businesslayer besitzt mit dem FrontController einen Facade-Controller, welcher einen einzigen Einstiegspunkt zur Businesslogik liefert und die Komplexität hinter sich verbirgt. Dabei werden die restlichen Controller jeweils erst beim ersten Gebrauch auch tatsächlich initialisiert (Lazy Initialization).

FrontController Dient als Facade-Controller. Er besitzt Referenzen auf die restlichen Controller. Zusätzlich finden sich hier Use Case Methoden, welche komplexere Abläufe bündeln und teils auch statisch, d.h. ohne Benutzeranmeldung, aufgerufen werden können.

- **Authenticate** Diese Methode verlangt einen Benutzernamen und ein Passwort und liefert bei erfolgreicher Authentifizierung eine IDetailedPerson zurück.
- **ResetPassword** Diese Methode verlangt neben dem Benutzernamen auch noch einen Authentisierungscode, und setzt bei dessen Korrektheit das Passwort auf den mitgelieferten Wert.
- **ActivateUser** Stimmt der mitgelieferte Authentisierungscode, wird der mitgelieferte Benutzername aktiviert.
- **ActivatePassivePerson** Stimmt der mitgelieferte Authentisierungscode, wird die passive Person mit dem mitgelieferten Benutzer verknüpft und aktiviert.
- **CheckEmailAvailability** Wirft eine Fehlermeldung, wenn die mitgelieferte Emailadresse bereits verwendet wird.
- **GetSimplePerson** Liefert den Namen der Person mit der gegebenen Id zurück.
- **RegisterUser** Erstellt die Person und den Benutzer und verknüpft sie.
- **SendMergeRequest** Schickt eine Anfrageemail an die Adresse der zu übernehmenden Person.
- **RequestPasswordReset** Schickt eine Email mit Autorisierungscode für die Zurücksetzung an die gegebene Useremailadresse.

ContactController Dieser Controller kümmert sich um die Verwaltung der Kontaktinformationen. Pro Kontaktinformation stehen folgende Methoden zur Verfügung:

- **UpdateXXX** Speichert die mitgelieferte Kontaktinformation. Wenn diese bereits eine Id besitzt, so wird ein Update ausgeführt, ansonsten wird ein neuer Datensatz erstellt und die neue Id zurückgeliefert.
- **DeleteXXX** Löscht die Kontaktinformation mit der gegebenen Id.
- **GetXXXs** Liefert die Kontaktinformation, welche der Person mit der mitgelieferten Id gehören.

PhonebookController Dieser Controller dient zum Nachschlagen von Personen in Telefonbüchern im Internet.

AuthenticationController Neben der Authenticate Methode stellt dieser Controller noch weitere Methoden zur Verfügung, welche mit dem User und dessen Login zusammenhängen.

MessagingController Das Versenden und Empfangen von SMS- und Email-Nachrichten sowie die Konfiguration der nötigen Einstellungen übernimmt der MessagingController

RelationController Für das Erstellen, Ändern, Bestätigen und Löschen von Beziehungen ist der RelationController zuständig. Zusätzlich stellt er die Methoden zur Gruppierung von Kontakten zur Verfügung.

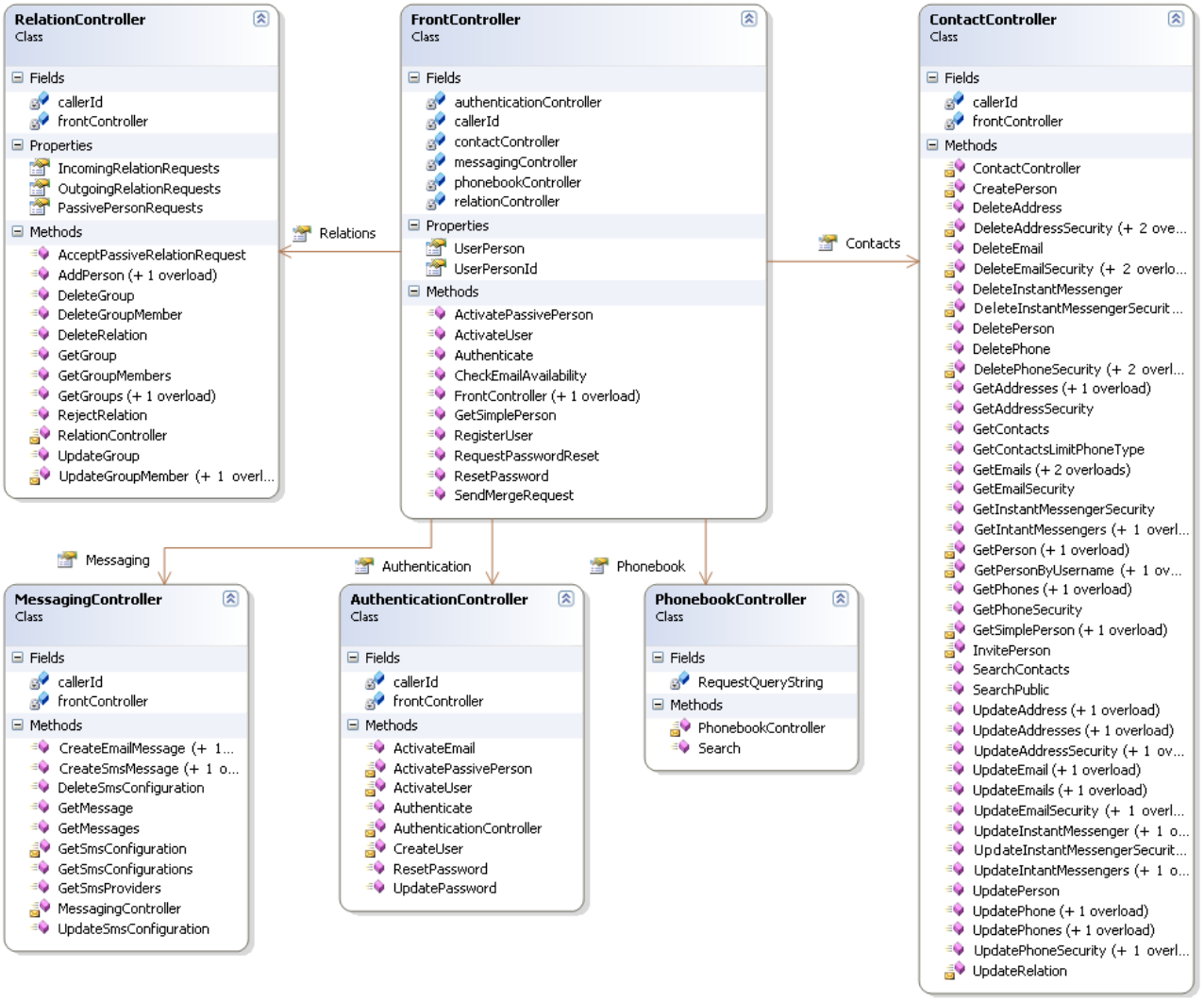

Abbildung 27: Der FrontController und die verschiedenen Controller der Businessschicht

5.4.8 Web Layer

Das Domainmodell der Webseite umfasst sehr viele Klassen, da jede Seite und jedes Control eine Klasse zur Steuerung besitzt. Deshalb wird hier nicht detailliert auf alle Klassen eingegangen. Stattdessen wird der Aufbau der Pakete innerhalb der Webseite erläutert.

services Hier befindet sich das Hosting des WCF Services.

controls Unter diesem Paket sind alle Webcontrols abgelegt. Sie sind ein wichtiger Bestandteil des Modularen Aufbaus der Webseite. Genauere Informationen zu den einzelnen Webcontrols finden sich im Kapitel 7.7.

design In diesem Ordner sind sämtliche Bilder für die Webseite abgelegt. Zusätzlich ist hier auch das Cascading Style Sheet untergebracht.

objects Hier finden sich die Implementationen der Domainobjekte. Zusätzlich sind hier nützliche Extender-Methoden untergebracht.

ControllerFactory Die ControllerFactory stellt sicher, dass pro Request jeweils nur ein FrontController instanziert werden muss.

web.config In dieser XML Datei sind sämtliche Konfigurationen für die Webseite gespeichert.

ASPX-Seiten Sämtliche im Browser aufrufbare Seiten sind im Hauptverzeichnis als .aspx-Seiten abgelegt.

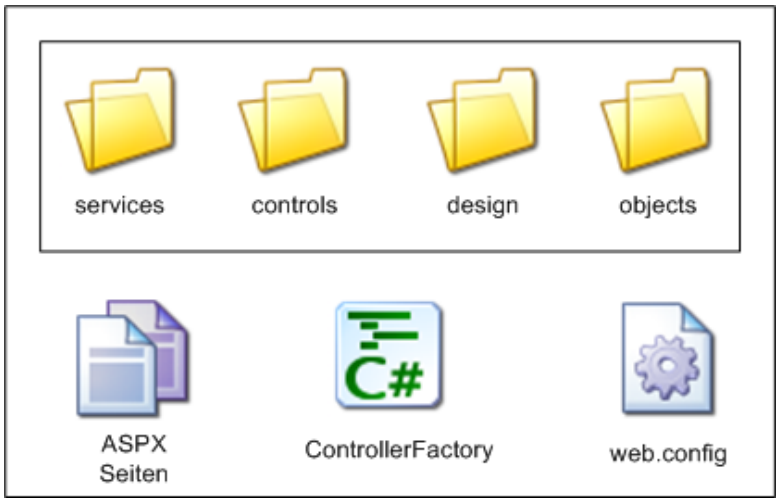

Abbildung 28: Ordnerstruktur der Webseite

5.5 Sicherheit

In der Analyse der Sicherheit in Kapitel 4.5 auf Seite 24 wurden die Anforderungen für den Zugriffsschutz der Kontaktinformationen folgendermassen definiert:

- Ausschlaggebend für die Zugriffsrechte ist die Beziehung zwischen der angefragten Person und dem Anfrager.
- Für jede Kontaktinformation kann die minimal benötigte Beziehungsstufe definiert werden.
- Der Zugriff kann nicht gezielt einzelnen Personen gewährt oder verweigert werden.
- Um Attacken zu erschweren, soll der Zugriffsschutz möglichst tief im System verankert sein.

Ausgehend von diesen Anforderungen wurde eine Lösung angestrebt, welche bereits auf Datenbankebene den Zugriff auf die Kontaktdaten absichert. Der Zugriff auf die Datenbank ist nicht mehr direkt, sondern nur noch über Stored Procedures möglich. Diese werden zusammen mit der Identität des Aufrufers ausgeführt und können damit die Rechtmässigkeit der Abfrage überprüfen.

Durch den Aufbau des Datenbankmodells (siehe Kapitel 5.3.4 auf Seite 41) ist es möglich, für den Zugriff ein einzelnes SQL Query zu verwenden, welches mithilfe von LEFT JOINS jeweils nur die erlaubten Informationen zurück liefert. Dadurch besitzt das zurückgelieferte Kontaktobjekt jeweils nur die Informationen, für welche der Aufrufer genügend Rechte besitzt. Somit ist die Autorisierung auf Datenbankebene realisiert. Genaueres zur Implementation der Stored Procedures findet sich im Kapitel 7.3 auf Seite 66.

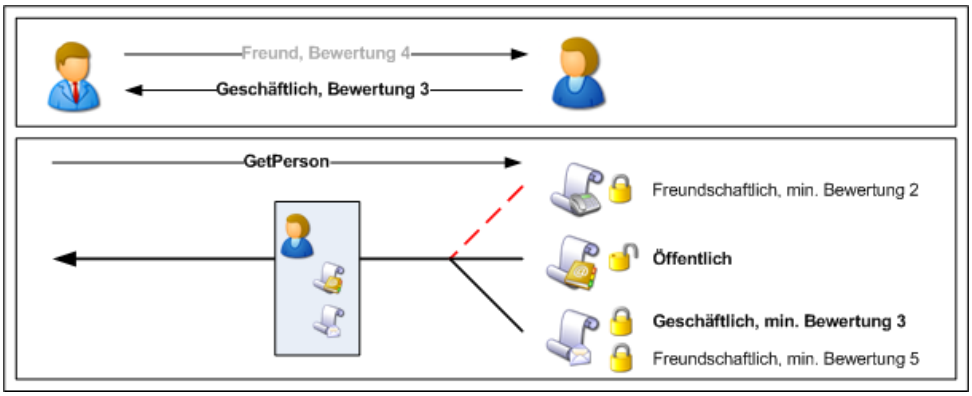

Abbildung 29: Zugriffsschutz beim Abfragen von Kontaktinformationen

Die Authentisierung ist im Businesslayer implementiert. Der FrontController kann nur mithilfe eines gültigen Benutzernamens und Passwortes instanziert werden. Der Controller speichert nach erfolgreicher Instanzierung die Id des Benutzers in einer privaten Variable. Diese gibt er jeweils den auszuführenden Stored Procedures als CallerId mit. Methoden, die auch ohne Login zur Verfügung stehen wie beispielsweise die Registrierungsmethode, sind statisch definiert. Mehr dazu im Kapitel 5.4.7 auf Seite 48.

5.6 Architekturkonzepte

5.6.1 Stored Procedures

In der Studienarbeit wurden sämtliche Abfragen auf die Datenbank mittels LINQ to SQL abgewickelt. Mit der Einführung neuer Funktionen und der Vergrösserung des Datenbankmodells auf über 33 Tabellen wurden die Abfragen über LINQ schnell sehr kompliziert, unperformant und umständlich. Gewisse komplexere Workflows waren mit LINQ schwierig auszuführen. Ausserdem führte das von LINQ bevorzugte Lazy Loading bei der eingesetzten Mehrfach-Tier-Architektur immer wieder wegen angehängten, aber nicht initialisierten Objekten zu Fehlern.

Stored Procedures stellen eine ideale Lösung für dieses Problem bereit. Mit ihrer Hilfe lassen sich komplexe Abläufe in der Datenbank kapseln und gezielt mittels Parameterübergabe ausführen. Mit LINQ to SQL können die Stored Procedures direkt angesprochen und die Resultate automatisch auf Objekte gemappt werden.

Ein weiterer Vorteil ist die Verlagerung eines Grossteils der Businesslogik in die Datenbank, wodurch diese Regeln zentral definiert und garantiert eingehalten werden. Der Programmcode kann dadurch übersichtlicher gehalten werden. Anstelle von mehreren aufeinander folgenden LINQ to SQL Queries reicht nun meist der Aufruf eines einzelnen Stored Procedures aus.

Zu den wichtigsten Businessregeln gehören die Zugriffskontrollen auf die sensiblen Kontaktdaten. Durch das Verwenden der Stored Procedures ist es möglich, dem Datenbanknutzer sämtliche Rechte zu entziehen und ihm nur die Ausführrechte der Stored Procedures zu geben. Dadurch muss jeder Zugriff über die definierten Prozeduren laufen, wodurch bereits ein Grossteil der möglichen Attacken zur Umgehung der Sicherheitsmechanismen ausgeschlossen werden kann. Mehr dazu im Kapitel 7.3 auf Seite 66.

Ein Nachteil dieser Lösung ist der grössere Aufwand bei der Implementation. Für jeden Use-Case muss mindestens ein Stored Procedure geschrieben werden. Zusammen mit dem Konfigurieren des Objektmapping ergibt sich ein erheblicher Mehraufwand gegenüber den LINQ to SQL Abfragen. Dieser Nachteil wurde aber zugunsten der erhöhten Sicherheit und Übersicht in Kauf genommen.

5.6.2 LINQ to SQL und POCO's

POCO ist ein Akronym für Plain Old CLR Object und bezeichnet "ganz normale" Objekte in einer .NET Programmiersprache ohne vielfältige externe Abhängigkeiten. Es ist die .NET Variante von POJO [12]. Der grosse Vorteil bei der Verwendung von POCO ist die einfache Serialisierung der Objekte, welche bei verteilten Architekturen wichtig ist.

LINQ to SQL bietet die Möglichkeit, Rückgabewerte einer Stored Procedure direkt auf diese POCO's zu mappen. Die Kontrolle über die Objekte und das Mapping der Daten kann so besser kontrolliert und gesteuert werden.

Mehr zur Implementation der POCO's findet sich im Kapitel 7.5 auf Seite 72

5.6.3 FrontController

Der FrontController bietet einen zentralen Zugriffspunkt auf alle Funktionen der Businessschicht. Dies ist eine Implementation des Facade-Pattern [13]. Dadurch wird vor allem der Zugriff vereinfacht, da die Komplexität des Businesslayer verborgen werden kann. Zusätzlich wird die Abhängigkeit zwischen den Schichten verringert, da obere Schichten nur den FrontController kennen müssen.

Die Sicherheit profitiert ebenfalls davon, wenn nur noch ein einziger Einstiegspunkt in das System existiert (Single Point of Access [2]). Die Sicherheit kann daher an einem zentralen Ort implementiert werden. Genaueres dazu findet sich im Kapitel 5.6.4.

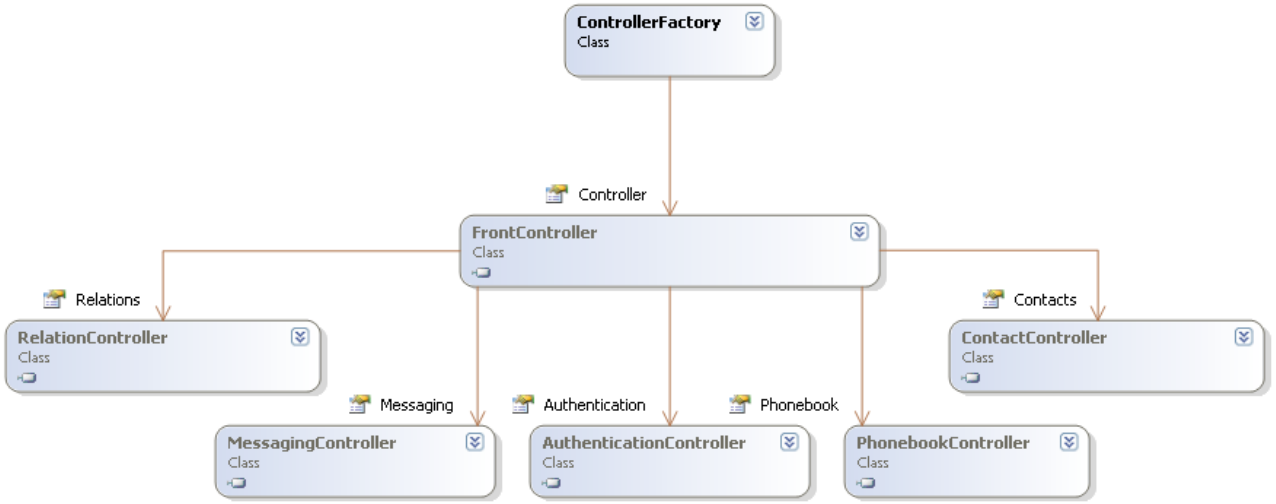

Abbildung 30: Die oberen Schichten (hier ControllerFactory) greifen über den FrontController auf die Businessschicht zu.

5.6.4 Authentisierung und Autorisierung

Bei der Sicherheit wurde darauf geachtet, dass die Mechanismen für die Authentisierung und die Autorisierung möglichst tief im System verankert sind. Daraus ist ein Konzept entstanden, welches diese Mechanismen fliessend in das Programm integriert.

Eine statische Methode des FrontControllers ermöglicht das Authentisieren. Wenn dies erfolgreich war, liefert die Methode ein SecurityToken zurück, welches die Benutzersitzung kennzeichnet. Dieses wird ebenfalls zusammen mit der Clientadresse in der Datenbank gespeichert. Nur mithilfe eines gültigen SecurityTokens ist es möglich, den FrontController zu instanzieren. Der FrontController prüft das Token und speichert die UserId. Die Authentisierung ist dadurch in der Businessschicht verankert.

Der FrontController ruft die Stored Procedures jeweils mit der UserId des aufrufenden Benutzers auf. Die Stored Procedures haben dadurch alle Informationen, die sie benötigen, um die Rechte des Benutzers abzuklären. Sie sind so implementiert, dass Sie nur die Kontaktinformationen zurück liefern, für welche auch genügend Rechte vorhanden sind. Die Autorisierung ist also auf Datenbankebene realisiert.

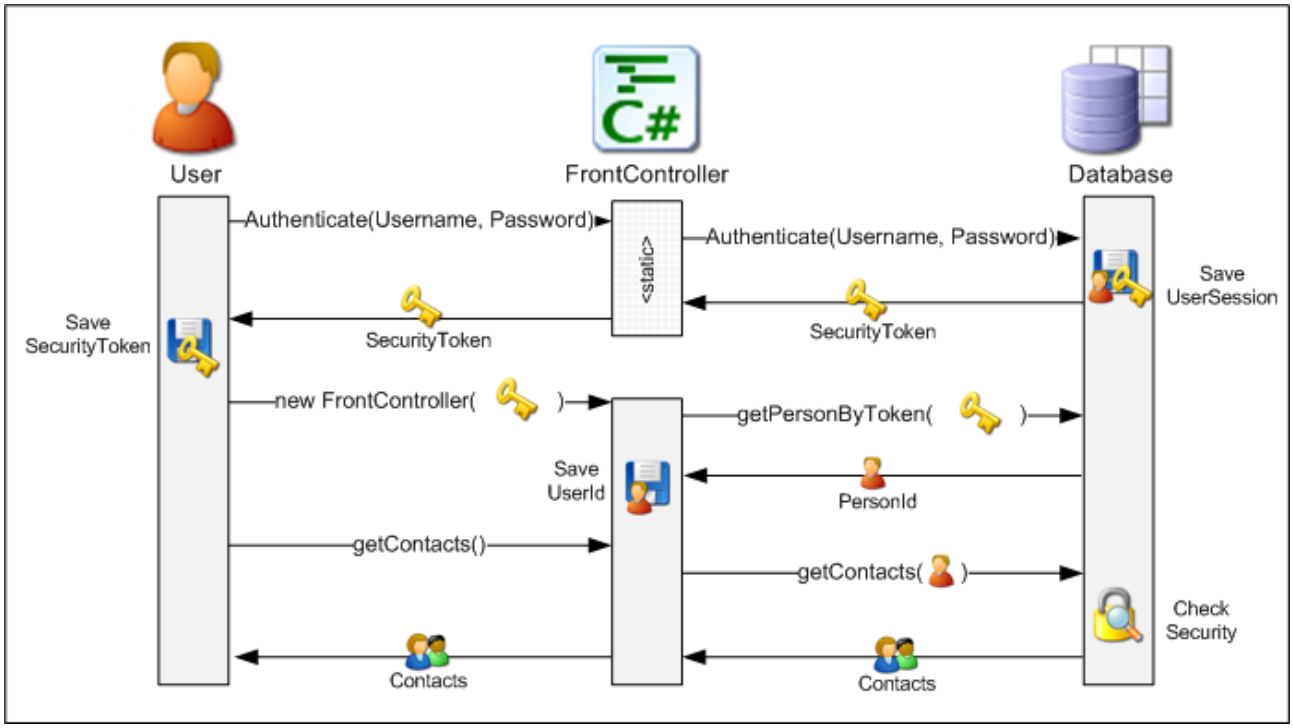

Abbildung 31: Ablauf der Authentisierung und Autorisierung

5.7 User Interface

Das externe Design für die Webseite wurde bereits in der Studienarbeit erarbeitet [1]. Dieses Design soll mit Hilfe von Usability Tests auf Benutzerfreundlichkeit getestet und umfassend überarbeitet werden. Mehr Informationen dazu finden sich in den Kapiteln 6.3 und 8.2.

In diesem Kapitel soll deshalb nur das Design der neuen Funktionen geplant werden.

5.7.1 Externes Design Messaging

Für die neue Messaging Funktionalität wurde ein Layout gewählt, das vielen Benutzer bereits bekannt sein sollte. Damit soll eine hohe Akzeptanz erreicht werden. Das Layout orientiert sich dabei an Outlook von Microsoft. Dabei werden auf der linken Seite die Nachrichten aufgelistet, während in der rechten Spalte eine neue Nachricht erfasst werden oder betrachtet werden kann.

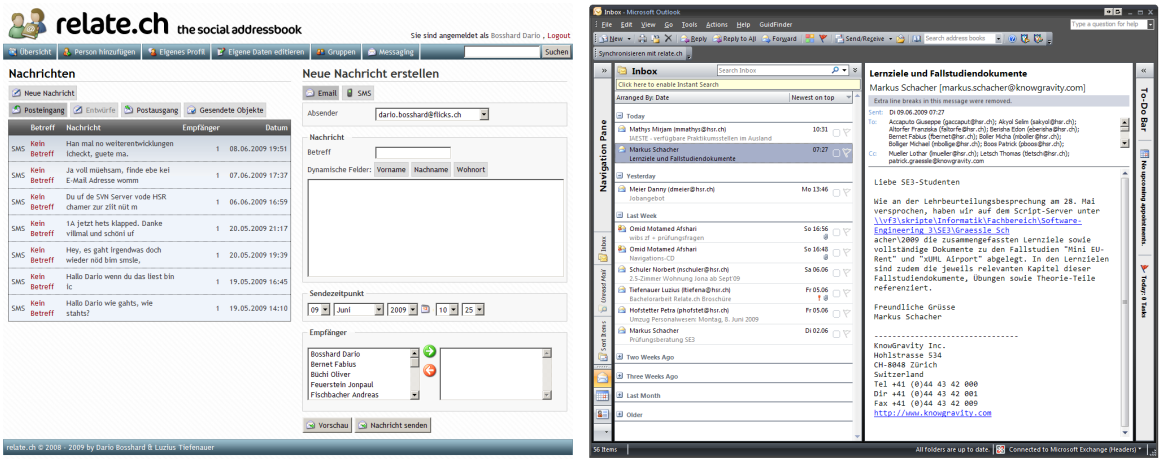

Abbildung 32: Messaging Ansicht im Vergleich zu Outlook

5.7.2 Externes Design Gruppen

Eine Gruppe mit ihren Mitgliedern wird ähnlich der Standard Übersicht dargestellt. In der Mitgliederliste ist zudem zu erkennen, ob ein Mitglied über Administratorenrechte für eine Gruppe verfügt und ob die Person bereits der Gruppe beigetreten ist. Als Administrator einer Gruppe hat man zusätzliche Einstellungsmöglichkeiten, die über ein Menü zugänglich sind.

Abbildung 33: Externes Design - Übersicht einer Gruppe

6 Refactoring

6.1 Datenbank

Das Datenbankmodell aus der Studienarbeit musste aufgrund der umfassenderen Funktionalität erweitert werden. Im Zuge des Ausbaus wurde zugleich das bestehende Modell analysiert und Schwachstellen überarbeitet. Im Nachfolgenden wird auf die einzelnen Änderungen am Datenbankmodell eingegangen.

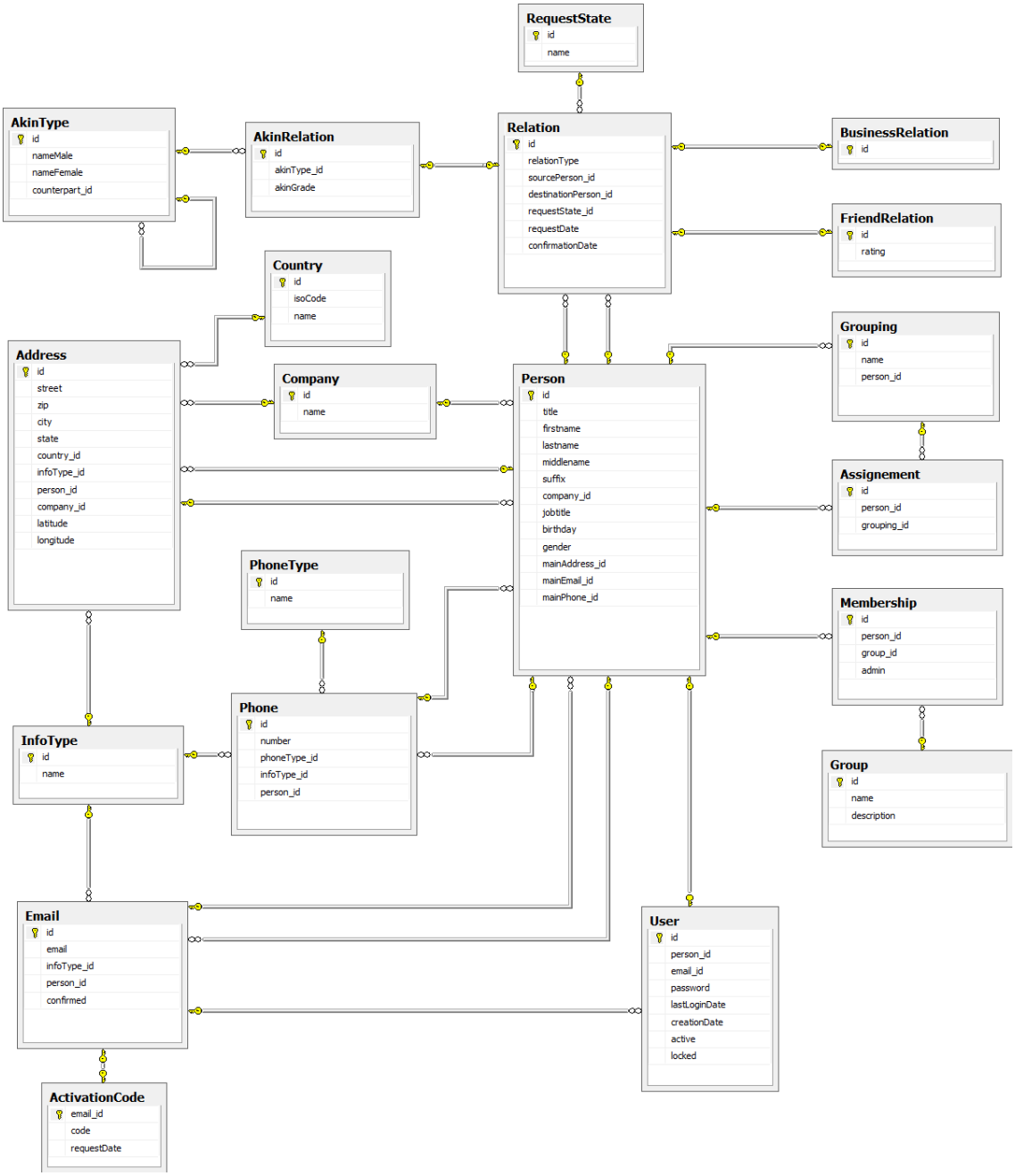

Abbildung 34: Datenbankmodell der Studienarbeit

6.1.1 Refactoring der Kontaktdaten

In dem bestehenden Modell wurde jeder Person jeweils eine Hauptadresse, -telefonnummer und -email zugewiesen. Dieses Design ermöglichte zwar einen schnellen Zugriff auf die Hauptkontaktdaten einer Person, stellte aber gleichzeitig eine unschöne Redundanz dar (sichtbar in 35 durch die doppelten Verbindungen zwischen der Person und den jeweiligen Kontaktdaten). Zusätzlich ist es durch diese Datenstruktur nicht möglich, dass eine Person für geschäftliche und private Kontakte eine andere Hauptadresse hat.

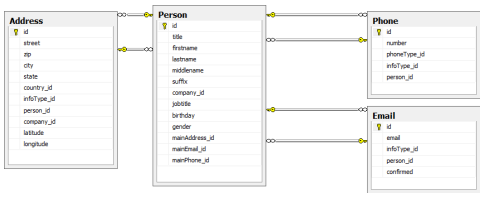

Abbildung 35: Ursprüngliches Kontaktdatenmodell

Durchgeführte Änderungen

Die Kontakttabellen erhalten eine neue Spalte, welche anzeigt, ob es sich um eine primäre Kontaktinformation handelt oder nicht. Über Triggers und Stored Procedures wird sichergestellt, dass jede Person jeweils pro Informationstyp (geschäftlich, privat, usw.) nur eine primäre Adresse, Telefonnummer und Email hat. Dadurch wurde die doppelte Beziehung zwischen der Person und den Kontaktdaten überflüssig.

Die erhöhte Komplexität beim Zugriff auf die Hauptkontaktinformationen ist eher als unproblematisch zu betrachten, da dieser zentral geschieht und daher nur einmal programmiert werden muss. Die schlechtere Performance kann durch die Verwendung von effektiveren Stored Procedures wieder wett gemacht werden.

Vorteile

- Vereinfachte Beziehung zwischen Person und Kontaktinformation
- Ermöglicht Hauptkontaktinformation pro Informationstyp

Nachteile

- Zugriff auf Hauptkontaktinformationen wird komplexer
- Integritätsprüfung über Triggers und Stored Procedures nötig

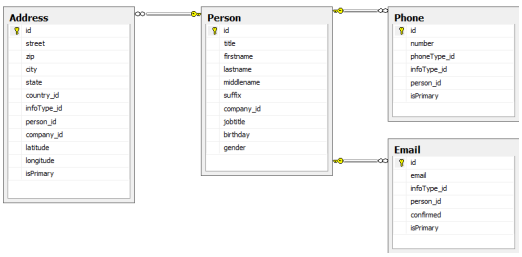

Abbildung 36: Neues Kontaktdatenmodell

6.1.2 Refactoring der Beziehungstabellen

Die Beziehungstabellen bestehen im alten Datenbankmodell aus einer Vererbungsstruktur auf Datenbankebene, welche einen sehr geringen Nutzen brachte, aber gleichzeitig die Komplexität erhöhte und die Performance verringerte. Die Verwandschaftsbeziehung war bisher nur im Datenbankmodell vorhanden und wurde in der darüber liegenden Applikation nicht berücksichtigt.

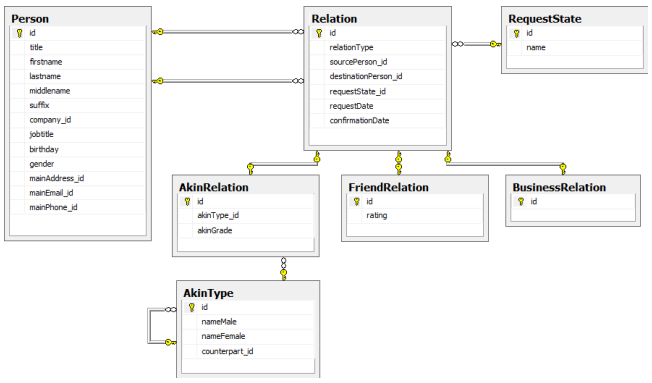

Abbildung 37: Ursprüngliche Beziehungstabellen

Durchgeführte Änderungen

Die Vererbungshierarchie der Beziehungen wurde denormalisiert und wieder in eine einzige Tabelle zusammengefasst. Es ist aber nach wie vor möglich, auf Applikationsebene mit einer Vererbungshierarchie zu arbeiten. Dazu wird das RelationType Feld als DiscriminatorValue verwendet, welcher bestimmt, auf welche Klasse die Beziehung gemappt werden soll.

Die Klassierung der Verwandschaftsbeziehung wurde weggelassen, da die Umsetzung eines Stammbaumes nicht geplant und der Nutzen dieser Beziehungsart deshalb zum jetzigen Zeitpunkt fragwürdig ist.

Vorteile

- Vereinfachter Zugriff auf Beziehungen
- Kompakteres Datenbankdesign

Nachteile

- Klassifizierung von Verwandschaftsbeziehungen wird nicht mehr unterstützt

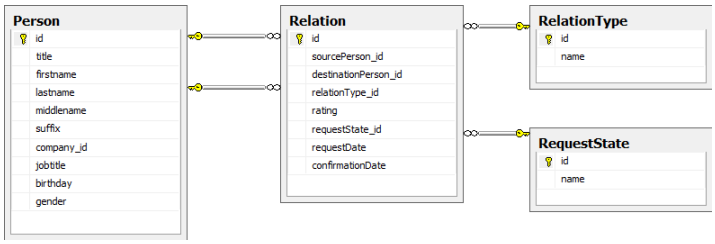

Abbildung 38: Neue Beziehungstabellen

6.1.3 Messaging

Um das Versenden von Nachrichten zu ermöglichen, mussten einige neue Tabellen erstellt werden. Vorerst werden zwei Nachrichtentypen unterstützt. Emailnachrichten sind mit einer Absenderemailadresse und SMS Nachrichten mit einer Absendertelefonnummer verknüpft.

Um SMS verschicken zu können, muss eine Person zuerst einen SMS Provider konfigurieren. Das heisst, sie muss beispielsweise ihre Logindaten für die Swisscom SMS-Box angeben. Diese Angaben werden dann in der SMSConfiguration Tabelle gespeichert. Die Tabelle MailGroup wird benötigt, um Gruppen und Gruppierungen eine eigene Emailadresse einzurichten.

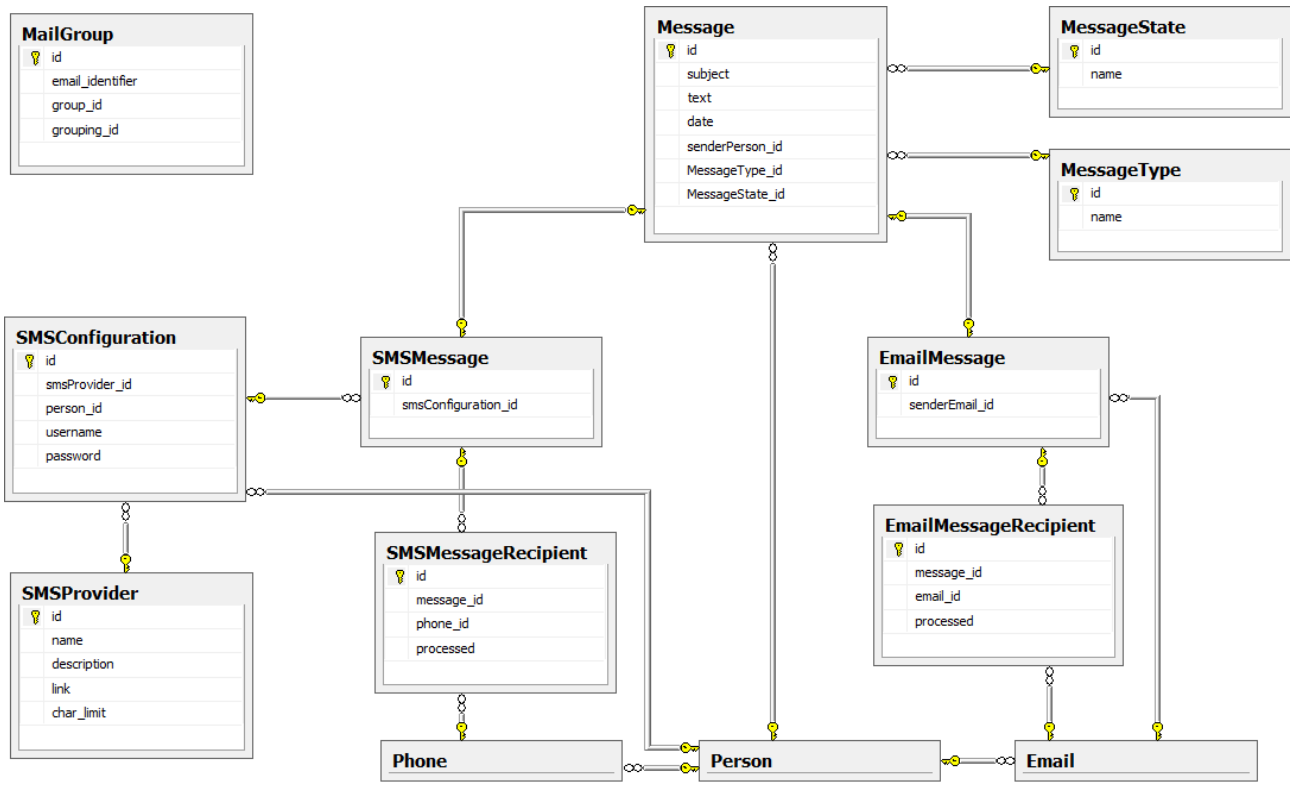

Abbildung 39: Tabellen für die Nachrichtenverarbeitung

6.1.4 Sicherheit

Um die Integrität der Daten sicherzustellen, ist ein umfassendes Sicherheitssystem nötig, das bereits auf Datenbankebene greift. Wichtig dabei ist, dass die Person einfach festlegen kann, wer bestimmte Daten sehen darf und wer nicht. Dabei wird zwischen zwei Einstellungen unterschieden. Zum einen können Daten als öffentlich markiert werden. Solche Daten sind für jede andere in Beziehung stehende Person ersichtlich. Die zweite Möglichkeit schränkt den Zugriff auf den Beziehungstyp und die Bewertung der Beziehung ein. So kann definiert werden, dass eine Emailadresse nur für private Kontakte ab einer Bewertung "Bekannter" einsehbar ist. Dabei spielt nur die Bewertung zwischen der Person, die die Daten zur Verfügung stellt, und der Person, die die Daten einsehen will, eine Rolle. Die Bewertung in die andere Richtung hat keinen Einfluss.

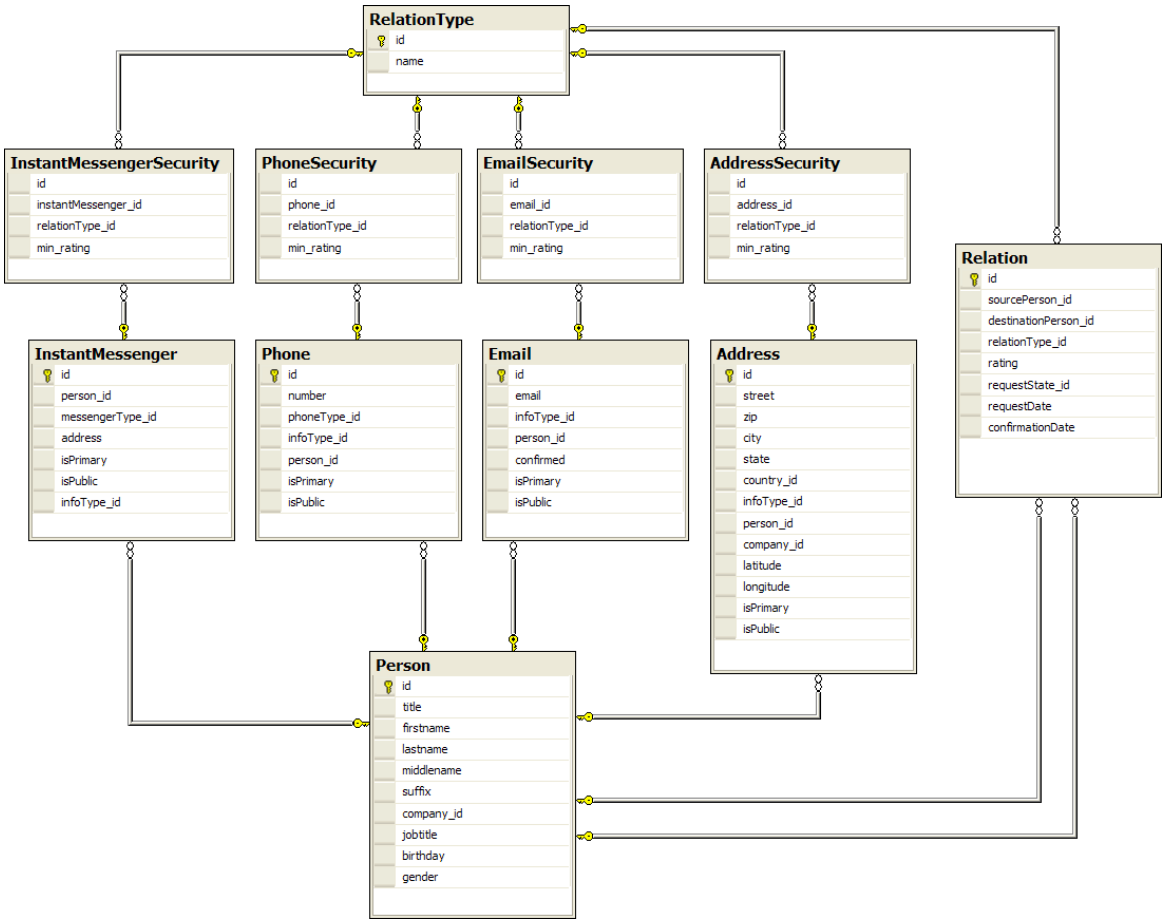

Abbildung 40: Tabellen für die Sicherheit

Vorteile

- Personen können den Zugriff auf ihre Daten selbst regulieren und jederzeit ändern
- Sicherheit besteht bereits auf Datenbankschicht

Nachteile

- Abfragen werden komplexer und aufwändiger

6.2 Interfaces

Im Prototyp der Studienarbeit werden durch LINQ to SQL Objekte direkt aus dem Datenbankmodell generiert. Dabei wird den Daten aus der Datenbank zusätzliche Funktionalität angefügt. So bieten diese Objekte die Möglichkeit, Änderungen nachzuverfolgen und diese zurück in die Datenbank zu schreiben. Um die Objekte für die Webseite und den WCF Service serialisieren zu können, mussten diese in DTO's konvertiert werden. Diese Konvertierung wurde durch einen DTO Converter erledigt, welcher mittels Reflection die Objekte automatisch zu DTO konvertierte. Hiermit wurden die Objekte zwischen Datenbank und Webseite zweimal konvertiert. Dieser Umstand sowie die Verwendung von Reflections hatten negativen Einfluss auf die Performance.

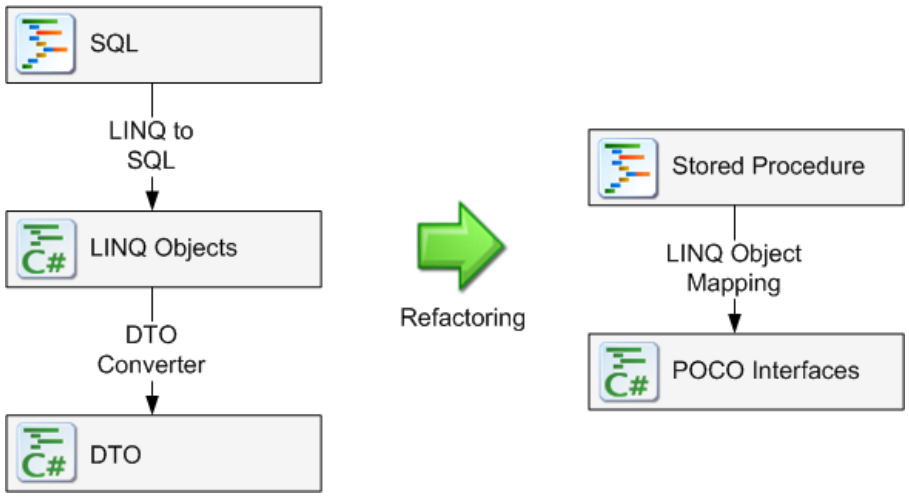

Abbildung 41: Statt doppelter Konvertierung werden die Datenbank Resultate direkt auf POCO's gemappt

Durchgeführte Änderungen

Statt einer doppelten Konvertierung werden die Resultate aus der Datenbank direkt auf POCO's gemappt. Abfragen können dadurch schneller durchgeführt werden.

Um die Kopplung zwischen den Schichten tief zu halten, kommunizieren die einzelnen Schichten jeweils nur über Interfaces. Diese sind relativ stabil und ermöglichen dadurch eine unabhängige Weiterentwicklung der einzelnen Schichten.

6.3 User Interface

6.3.1 Layout

Der Prototyp der Studienarbeit setzte auf ein klassisches Layout mit Header, Navigation, Informations- und Inhaltsbereich. Das Layout war auf eine feste Breite ausgelegt. Veränderte sich die Grösse des Browserfensters, änderte dies für den Benutzer nichts.

Die Grösse der Computermonitore und die damit verbundene Auflösung steigt kontinuierlich an. Bisher wurde das Potenzial dieser Monitore nicht optimal ausgenutzt. Um diesen Platz optimal zu nutzen, entschieden wir uns, das Layout komplett zu überdenken.

Das neue Layout ist so konzipiert, das auf der linken Seite ein fixer Bereich für Informationen reserviert ist. Der rechte Bereich der Seite passt sich der Grösse des Browserfensters an. Vorzugsweise wird dieser Bereich für die Visualisierung der Personendaten eingesetzt.

Der bisherige Kopfbereich der Seite verbrauchte sehr viel Platz, ohne dem Benutzer einen entsprechenden Mehrwert zu bieten. Mit der Reduzierung des Headerbereichs wurde auch gleichzeitig ein neues Logo eingeführt. Dieses Logo soll professioneller und schlichter wirken, es besteht aus einem schlichten Schriftzug.

Die Seite ist in matten Farbtönen gehalten, was ihr einen professionelleren Look verschaffen soll.

Abbildung 42: Layout der Studienarbeit und der Bachelorarbeit

7 Implementation

7.1 Triggers

Einige Konsistenzbedingungen können nicht direkt als Constraints definiert werden, da sie entweder zu komplex sind oder technische Limitierungen deren Definition verhindern. Mehr dazu im Kapitel 7.14.3.

Damit diese Bedingungen trotzdem stets eingehalten werden, wurden entsprechende Triggers geschrieben.

7.1.1 Contact Data Trigger

Bei sämtlichen Kontaktdaten kann eine Hauptkontaktinformation pro Kategorie definiert werden. Wird eine Kontaktinformation neu als primär definiert oder bei einer bereits Primären der Informationstyp geändert, muss sichergestellt werden, dass es pro Person und InfoType nur eine primäre Adresse (resp. Phone, Email und InstantMessenger) gibt. 1 zeigt als Beispiel den AfterInsertTrigger, welcher nach jedem neuen Adresseintrag aufgerufen wird.

Wie komplex vor allem der AfterUpdateTrigger ist, lässt sich aufgrund des Ablaufdiagramms 43 abschätzen.

Quellcode 1 AfterInsertAddress Trigger

```
CREATE TRIGGER [dbo].[AfterInsertAddress]
    ON  [dbo].[Address]
AFTER INSERT AS  BEGIN

   DECLARE @Id INT, @InfoType INT, @PersonId INT, @IsPrimary BIT, @IsPublic
      BIT;
    SELECT @Id = id, @InfoType = infoType_id, @PersonId = person_id,
       @IsPrimary = isPrimary, @IsPublic = isPublic FROM inserted;

    IF (@IsPrimary = 0) AND NOT EXISTS (SELECT id FROM Address WHERE
       person_id = @PersonId AND infoType_id = @InfoType AND isPrimary = 1)
        UPDATE Address SET isPrimary = 1 WHERE id = @ID;
    ELSE IF (@IsPrimary = 1)
        UPDATE Address SET isPrimary = 0 WHERE person_id = @PersonId AND
           infoType_id = @InfoType AND id != @Id;

    IF (@IsPublic = 1)
        DELETE FROM AddressSecurity WHERE address_id = @Id
END
```

AfterInsert Wird beim Erzeugen einer neuen Kontaktinformation ausgelöst.

AfterUpdate Wird bei der Änderung einer Kontaktinformation aufgerufen.

AfterInsertSecurity Wird nach dem Einfügen einer Kontaktinformations-Sicherheitseinstellung aufgerufen. Die Kontaktinformation wird dabei automatisch auf Privat gesetzt.

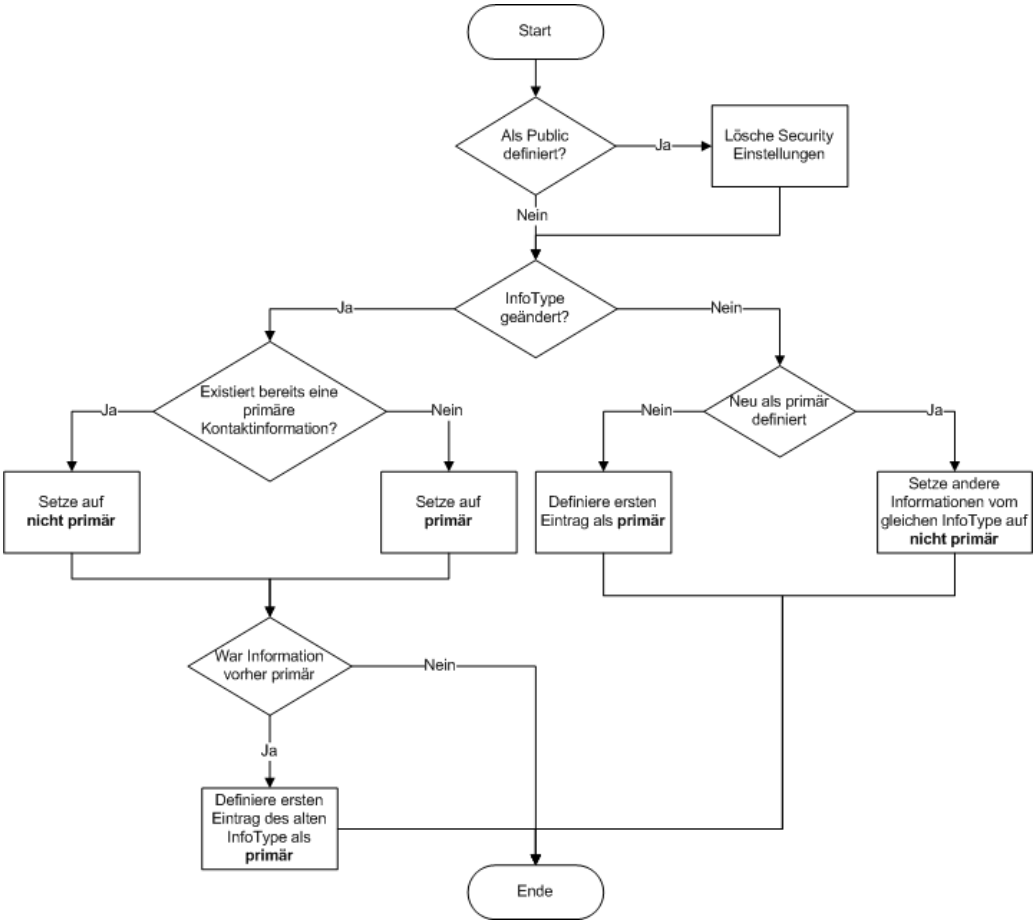

Abbildung 43: Ablauf des AfterUpdate Triggers

7.1.2 Messaging Triggers

AfterUpdateSMSRecipient Wird nach dem Update eines SMS Empfängers aufgerufen. Nach dem Versenden einer SMS Nachricht wird das Feld `processed` auf `True` gesetzt. Sind alle Empfänger einer Nachricht verarbeitet, wird die gesammte Nachricht auf versendet gesetzt.

AfterUpdateEmailRecipient Wird beim Update eines Email Empfängers aufgerufen. Gleicher Ablauf wie `AfterUpdateSMSRecipient` Trigger.

7.2 Datenbank Funktionen

Wiederkehrende Überprüfungen in Stored Procedures lassen sich komfortabel als Datenbankfunktionen speichern und überall wiederverwenden.

7.2.1 IsPassivePerson

Ob eine Person aktiv oder passiv ist, definiert sich durch die Existenz einer Beziehung zwischen Person und User Tabelle. Diese Information wird in vielen Stored Procedures benötigt und wurde deshalb als Datenbankfunktion umgesetzt.

7.2.2 CanEditPerson

Oft ist es in den Stored Procedures im Zusammenhang mit der Sicherheit wichtig zu wissen, ob man die vollen Zugriffsrechte auf eine Person besitzt oder nicht. Ein Benutzer kann nämlich nicht nur seine eigenen Daten editieren, sondern auch alle Daten seiner passiven Personen. Mit der Abfrage dieser Funktion kann sichergestellt werden, dass dem Benutzer dies erlaubt wird.

7.2.3 SplitValues

Ein Übergabewert eines Stored Procedures kann nicht mit mehreren Werten in Form einer Liste angesprochen werden. Um dieses Manko auszugleichen, wurde eine Funktion SplitValues definiert. Diese Funktion teilt eine mit Komma separierte Liste von Werten auf, und liefert diese in Form einer temporären Tabelle zurück. So ist es möglich, anstelle einer einzelnen Person, auch mehrere Personen an Hand ihrer Id zurück zu erhalten.

Quellcode 2 Aufruf einer Stored Procedure mit mehreren Werten

```
EXEC [dbo].[GetSimplePerson] @CallerId = 1, @PersonIds = N'1,2,3,4,5,6,7,8'
```

7.3 Stored Procedures

Der Zugriff auf die Datenbank findet komplett über Stored Procedures statt. Dabei enthalten sie bereits einen Grossteil der Businesslogik, wie beispielsweise Sicherheitsüberprüfungen. Somit sind sie ein wichtiger Punkt der Applikation und werden deshalb hier detailliert erläutert.

7.3.1 Sicherheit

Die Umsetzung des geplanten Sicherheitskonzepts, wie in Kapitel 5.5beschrieben, stellte eine Herausforderung dar. Einerseits muss die Sicherheit jederzeit gewährleistet werden, andererseits müssen die Abfragen performant ablaufen.

Das Design der Sicherheitstabellen ermöglicht es, mittels einfachen LEFT JOINS zu überprüfen, ob die Person über eine ausreichende Befugnis für einen Zugriff verfügt. Dabei wird die entgegengesetzte Beziehung zwischen dem Aufrufer und der Aufgerufenen Person herangezogen. Verfügt der Aufrufer über die nötige Einstufung (ReverseRelation.rating >= AddressSecurity.min_rating) oder ist die Kontaktinformation als öffentlich markiert (InnerAddress.isPublic = 1), wird der Zugriff auf die Daten gewährt.

Bei einer abgeflachten Person (ISimplePerson) ist der Vorgang noch ein wenig komplizierter, da immer nur eine Kontaktinformation pro Typ zurückgegeben wird. Dies führt zu einer zusätzlichen Unterabfrage, welche die Auswahl der Kontaktinformationen auf eine reduziert, die entweder öffentlich zugänglich oder durch den Aufrufer einsehbar ist.

Quellcode 3 Stored Procedure - Sicherheit der Kontaktdaten mittels LEFT JOIN

```
LEFT JOIN
    Address ON Address.person_id = Person.id AND Address.id IN (
        SELECT
            MIN(InnerAddress.id)
        FROM
            Address AS InnerAddress
        LEFT JOIN
            Relation AS ReverseRelation ON ReverseRelation.sourcePerson_id =
                Person.id AND
            ReverseRelation.destinationPerson_id = @CallerId
        LEFT JOIN
            AddressSecurity ON InnerAddress.id = AddressSecurity.address_id
               AND
            AddressSecurity.relationType_id = ReverseRelation.
               relationType_id
        WHERE
            InnerAddress.isPrimary = 1 AND
            (
                (AddressSecurity.id IS NOT NULL AND ReverseRelation.rating
                   >= AddressSecurity.min_rating) OR
                InnerAddress.isPublic = 1
            )
        GROUP BY
            person_id
    )
```

7.3.2 Personendaten und Kontaktinformationen

Um an Personendaten zu gelangen, wurden mehrere Stored Procedures erstellt, die an die verschiedenen Verwendungszwecke angepasste Resultate liefern. Im folgenden werden diese Stored Procedures aufgelistet und ihr Verwendungszweck kurz erläutert. Für die Kontaktinformationen wie Adresse, Emailadresse, Telefonnummer, Instant Messenger wird das Synonym ContactData verwendet.

SearchPublic	Liefert eine Liste von Personen, die auf ein Suchbegriff zutreffen. Die Liste enthält nur Personen, die noch nicht in der Kontaktliste des Aufrufers enthalten sind. Zudem enthält sie nur abgeflachte Personen in Form von ISimplePerson Objekten.
SearchContacts	Sucht innerhalb der Kontaktliste nach Personen, die auf den gegebenen Suchbegriff zutreffen. Werden die Suchbegriffe weggelassen, erhält man eine Liste sämtlicher Kontakte. Die Liste kann zudem auf den Status der Beziehung begrenzt werden. So erhält man Personen, die eine Beziehung angefragt haben oder solche, die bereits in Beziehung stehen. Die Liste enthält nur abgeflachte Personen in Form von ISimplePerson Objekten.
GetSimplePerson	Liefert eine abgeflachte Person in Form einer ISimplePerson aufgrund der gegebenen Id.
GetSimplePersonByEmail	Liefert eine abgeflachte Person in Form einer ISimplePerson aufgrund der gegebenen Emailadresse.
GetPerson	Liefert eine detaillierte Person mit allen zugänglichen Kontaktdaten in Form einer IDetailedPerson aufgrund der gegebenen Id.
GetPersonByUsername	Liefert eine detaillierte Person mit allen zugänglichen Kontaktdaten in Form einer IDetailedPerson aufgrund des gegebenen Usernamens.
GetPersonsByEmail	Liefert eine Liste von detaillierten Personen aufgrund der gegebenen Emailadresse.
UpdatePerson	Editiert Daten einer Person oder erstellt eine Neue.
GetContactData	Liefert eine Liste von Kontaktdaten zu einer bestimmten Person. Es werden nur Daten geliefert, für die der Aufrufer auch die entsprechenden Rechte besitzt.
UpdateContactData	Editiert eine bestehende Kontaktinformation oder fügt der Person eine neue hinzu. Kann nur von Personen aufgerufen werden, denen das Editieren von Daten erlaubt ist.
DeleteContactData	Löscht eine Kontaktinformation einer Person.

UpdateContactDataSecurity	Editiert eine bestehende Sicherheitseinstellung oder fügt eine neue hinzu.
DeleteContactDataSecurity	Löscht eine bestehende Sicherheitseinstellung.

7.3.3 Beziehungen

Diese Stored Prcedures werden für das Abrufen und Ändern von Beziehungen verwendet. Im folgenden werden diese Stored Procedures aufgelistet und ihr Verwendungszweck kurz erläutert.

UpdateRelation	Editiert eine bestehende Beziehung oder erstellt eine neue.
AcceptPassiveRelationRequest	Fügt ein Benutzer einen passive Kontakt seiner Kontaktliste hinzu, so wird der Ersteller dieses Kontaktes angefragt, ob der Benutzer Zugang zu den Daten erhalten darf.
GetIncomingPassivePersonRequests	Liefert die Anfragen anderer Personen für den Zugriff auf eine passive Person.
GetIncomingRelationRequests	Liefert die Anfragen anderer Personen für den Zugriff auf das Profil des Benutzers.
DeleteRelation	Löscht eine Beziehung zu einer Person.

7.3.4 Messaging

Für das Messaging wurden einige Stored Procedures erstellt, die das Erstellen und Abrufen von Nachrichten koordinieren. Im folgenden werden diese Stored Procedures aufgelistet und ihr Verwendungszweck kurz erläutert.

CreateEmailMessage	Erstellt eine neue Email Nachricht.
CreateSmsMessage	Erstellt eine neue SMS Nachricht.
AddMessageRecipient	Fügt einer bestehenden Nachricht einen Empfänger hinzu.
GetMessages	Liefert eine Liste aller Nachrichten, abhängig von ihrem Status, zurück.
GetSmsMessages	Liefert alle ausstehenden SMS Nachrichten. Dabei wird pro Empfänger eine Nachricht geliefert. Diese Liste von Nachrichten kann dann beim Versand der Reihe nach abgearbeitet werden.
SmsRecipientProcessed	Der Versand einer SMS Nachricht wird bestätigt.

GetEmailMessages	Liefert alle ausstehenden Email Nachrichten. Dabei wird pro Empfänger eine Nachricht geliefert. Diese Liste von Nachrichten kann dann beim Versand der Reihe nach abgearbeitet werden.
EmailRecipientProcessed	Der Versand einer Email Nachricht wird bestätigt.
GetSmsProviders	Liefert eine Liste aller zur Verfügung stehenden SMS Anbietern.
GetSmsConfiguration	Liefert eine bestehende SMS Konfiguration aufgrund der gegebenen Id.
UpdateSmsConfiguration	Editiert eine bestehende SMS Konfiguration oder erstellt eine neue.
DeleteSmsConfiguration	Löscht eine bestehende SMS Konfiguration.

7.3.5 Gruppen

DeleteGroup	Löscht eine Gruppe.
DeleteGroupMember	Entfernt Mitglieder aus einer Gruppe.
UpdateGroup	Ermöglicht das Editieren einer bestehenden Gruppe oder das Erstellen einer neuen Gruppe.
UpdateGroupMember	Fügt neue Gruppenmitglieder zu einer Gruppe hinzu.

7.3.6 Autorisierung

ActivateEmail	Prüft den gegebenen Aktivierungscode und aktiviert bei Übereinstimmung die gegebene Emailadresse.
ActivatePassivePerson	Prüft den gegebenen Aktivierungscode und aktiviert bei Übereinstimmung die passive Person.
ActivateUser	Prüft den gegebenen Aktivierungscode und aktiviert bei Übereinstimmung den Benutzer.
AuthenticateUser	Prüft den gegebenen Benutzernamen und das Passwort und liefert bei Erfolg ein SecurityToken zurück.
DeleteUser	Löscht den Benutzer mit gegebener Id.
InsertActivationCode	Speichert einen Aktivierungscode in der Datenbank.
RequestInvite	Überprüft, ob der Benutzer der gegebenen Person eine Einladungsemail senden darf.
ResetPassword	Prüft den gegebenen Aktivierungscode und setzt bei Übereinstimmung das Passwort neu.
UpdatePassword	Überprüft das alte Passwort und ersetzt dies bei Übereinstimmung mit dem neuen.

7.4 .NET 3.5

Das .NET Framework von Microsoft und C# im speziellen enthält mittlerweile viele nützliche Sprachfeatures. In diesem Kapitel wird aufgezeigt, wie diese Features bei der Programmierung dieser Arbeit eingesetzt wurden.

7.4.1 Extension Methods

Mit Extension Methods bietet das .NET Framework ein leistungsfähiges Werkzeug zur dynamischen Funktionalitätserweiterung von bestehenden Klassen an. Dabei wird das Objekt selbst nicht verändert, man muss nicht einmal die Rechte dazu haben.

Einsatz in dieser Arbeit
Extension Methods können nicht nur auf Klassen angewendet werden, sondern eignen sich auch für Interfaces. So kann auf einfache Art und Weise für alle auf einem Interface basierenden Implementationen neue Funktionen angefügt werden. Eine Person kann z.B. um eine Methode erweitert werden, die das Alter berechnet. Die vom Business Layer zurückgegebenen Objekte, Implementationen des Interface IPerson, müssen diese Funktion nicht enthalten und bleiben so schlank und übersichtlich.

Quellcode 4 Berechnung des Alters einer Person

```
public static int Age(this IPerson person)
{
    if (person.Birthday == null)
    {
        return 0;
    }

    int years = DateTime.Now.Year - person.Birthday.Value.Year;

    if (DateTime.Now.Month < person.Birthday.Value.Month || (DateTime.Now.
        Month == person.Birthday.Value.Month && DateTime.Now.Day < person.
        Birthday.Value.Day))
    {
        years--;
    }

    return years;
}
```

Ein weiteres Beispiel ist der vollständige Name einer Person. Diese Angabe wird nur vom Weblayer verwendet. Daher macht es Sinn, auch diese Angabe mit einer Extension Methode zur Verfügung zu stellen.

7.4.2 Lambda Expressions

Lambda Expressions entsprechen dem Konzept von Anonymen Methoden. In C# können dadurch Delegates direkt am Ort ihrer Verwendung definiert werden. Lambda Expression bieten somit einen flexibleren, schlankeren und meist auch verständlicheren Syntax an.

Lambda Expression	delegate
`x => x + 1`	`delegate (int x) {return x + 1}`
`(x, y) => {return x + y;}`	`delegate (int x, int y) {return x + y;}`

Abbildung 44: Unterschied zwischen anonymer Methode in Form von delegate und Lambda Expression

Vorteile

- Kompakter Code
- Funktionalität ist direkt beim Aufruf angesiedelt
- Einfacher und verständlicher Syntax

Nachteil

- Es ist nicht auf einen Blick ersichtlich, von welchem Typ die Parameter sind

Quellcode 5 Lambda Expression für die Zuweisung von Methoden zu Events

```
EmailEditControl.EntityRemove += id => ControllerFactory.Controller.Contacts
    .DeleteEmail(id);
PhoneEditControl.EntityRemove += id => ControllerFactory.Controller.Contacts
    .DeletePhone(id);
```

7.4.3 Objekt Initialisierung

In C# 3.0 wurde mit Object Initializers eine vereinfachte Art der Objekt-Initialisierung eingeführt. Dabei kann ein Objekt direkt nach dem Erstellen mit Werten initialisiert werden.

Quellcode 6 Klassische Objekt Initialisierung gegenüber dem Object Initializer

```
public class Person
{
    public string Firstname;
    public string Lastname;
}

Person p1 = new Person();
p1.Firstname = "Luzius";
p1.Lastname = "Tiefenauer";

Person p2 = new Person { Firstname = "Dario", Lastname = "Bosshard" };
```

Zudem ist es auch möglich, ganze Collections direkt zu initialisieren, was den Code sehr übersichtlich und schlank macht.

Quellcode 7 Initialisierung einer gefüllten Collection mit C# 3.0

```
new QueryString
{
    {"lastname", person.Lastname},
    {"firstname", person.Firstname},
    {"street", person.Street},
    {"zip", person.Zip},
    {"city", person.City},
    {"phoneNumber", person.PhoneNumber},
    {"latitude", person.Latitude.ToString()},
    {"longitude", person.Longitude.ToString()},
    {"IsPersonLink", "1"}
};
```

7.5 POCOs

Die Anbindung der Datenbank mit LINQ to SQL kann einerseits über den im Visual Studio eingebauten Designer, von Hand mittels C# Attributes oder mit einer XML Definition definiert werden. Der Designer erstellt dabei selbständig den benötigten Code und versieht die Objekte mit der zusätzlichen Funktionalität. Das manuelle Erstellen des Mappings bietet jedoch eine grössere Kontrolle. So ist es möglich, Resultate auf einfache POCO's zu mappen. Die resultierenden Objekte können ohne Umwege mit WCF verwendet werden.

Quellcode 8 Mapping von POCO's mittels C# Attributes

```
[Table(Name = "dbo.Email")]
public class Email : IEmail
{
    [Column(Name = "id", IsPrimaryKey = true, IsDbGenerated = true)]
    public int? Id { get; set; }

    [Column(Name = "person_id")]
    public int? PersonId { get; set; }

    [Column(Name = "email")]
    public string Address { get; set; }

    [Column(Name = "confirmed")]
    public bool Confirmed { get; set; }

    [Column(Name = "isPrimary")]
    public bool IsPrimary { get; set; }
}
```

7.6 Sicherheit

7.6.1 Verschlüsselung des Querystrings

Um die Sicherheit zu erhöhen, wird der Querystring verschlüsselt. Damit ist nicht mehr ersichtlich, welche Parameter mit welchen Werten übergeben werden und die Manipulation wird so erschwert. Der grössere Aufwand für die Verschlüsselung, ist durch den erhöhten Schutz vor Angriffen gerechtfertigt.

Die Logik für die Ver- und Entschlüsselung wurde in einer speziellen Querystring-Klasse untergebracht. Diese bietet einige zusätzliche nützliche Funktionen für das Erstellen und Auslesen von Querystrings.

Bei der Implementation der Verschlüsselung stiessen wir auf einige Probleme. Diese sind in Kapitel 7.14.4 auf Seite 89erläutert.

7.6.2 Sicherheit der Logindaten für externe SMS-Dienstleister

Problem

Der Versand von SMS Nachrichten bedingt, dass die Personen ihre Logindaten für den entsprechenden SMS Dienst preisgeben. Die Sicherheit dieser Daten soll jederzeit gewährleistet sein. Das zeitversetzte Senden von SMS Nachrichten setzt dabei voraus, dass die Logindaten für die Applikation im Klartext vorliegen, da sonst ein Login auf der entsprechenden Plattform nicht möglich ist.

Verschlüsselung der Daten

Um eine möglichst hohe Sicherheit der Daten zu garantieren und gleichzeitig den zeitversetzten Versand zu ermöglichen, werden die Logindaten in der Datenbank verschlüsselt abgelegt. Diese werden erst kurz vor dem Senden der Nachricht wieder entschlüsselt. Der Schlüssel ist dabei nicht in der Datenbank abgelegt. So soll sichergestellt werden, dass eine Person mit Einsicht in die Datenbank nicht ohne Probleme an die Logindaten gelangen kann.

7.7 UserControls

WebControls und UserControls sind ein zentrales Konzept von ASP.NET. Mit diesen Controls ist es dem Programmierer möglich, eigene Komponenten zu entwickeln, die er dann an beliebigen Orten einsetzen kann. Innerhalb dieser Controls stehen die gleichen Möglichkeiten zur Verfügung wie bei normalen ASP.NET Seiten.

Code und Layout werden dabei getrennt behandelt. Eine häufig verwendete Funktionalität kann so in einer Komponente gebündelt werden. Die Controls können mehrfach verschachtelt werden. Sie können als solches vom Projekt losgelöst und auch in anderen Projekten verwendet werden.

So gibt es für diverse Szenarien bereits Controls im Internet zu finden, die frei oder kommerziell angeboten werden. Meist ist es aber sinnvoll, eigene Controls zu schreiben, da sich die gewünschte Funktionalität meist stark von bestehenden Lösungen unterscheidet.

Nachfolgend sollen einige der entwickelten UserControls erläutert werden.

7.7.1 ContactDataEditControl

Die Kontaktdaten leiten alle vom Interface IContactDataObject ab, da viele Eigenschaften der Kontaktdaten gleich sind und sich gleich verhalten. Diese Gemeinsamkeiten flossen auch in die ASP.NET Controls zum Editieren der Kontaktdaten ein.

In der Studienarbeit wurden sämtliche Controls einzeln ausgebildet. Schon damals fielen Gemeinsamkeiten im Code auf, die wir als unnötige Redundanz empfanden. Um diesen Missstand zu bereinigen,

wurde ein Abstraktes Edit Control geschaffen, das alle Gemeinsamkeiten vereint. Zum Ausbilden des Controls sind nun nur noch wenige Zeilen Code notwendig, die sich auf die kleinen Unterschiede der einzelnen Kontaktdaten beschränken.

Abbildung 45: Sowohl AddressEditControl als auch PhoneEditControl leitet von ContactEditControl ab.

Properties Properties stellen einen einfachen Zugang zu Daten innerhalb des Controls dar.

Objects Über dieses Property werden bestehende Objekte an das Control übergeben. Beim Lesen des Properties werden die Daten, die der Benutzer in der Zwischenzeit eingegeben hat, mit den Originaldaten zusammengefasst.

Methoden

CreateNewObject Muss von einem abgeleiteten Control zwingend implementiert werden. Über diese Methode wird eine konkrete Implementation des Interfaces IContactDataObject instanziert.

Events Events dienen der Verminderung der Abhängigkeit zwischen dem Control und der implementierenden Seite.

OnEntityGetSecurity Event wird ausgelöst, sobald erstmals die Security Settings für Kontaktdaten angefordert werden.

OnEntityRemove Event wird ausgelöst, sobald ein Eintrag gelöscht werden soll.

OnEntitySaveSecurity Event wird ausgelöst, sobald die Sicherheitseinstellungen gespeichert werden sollen.

OnEntitySetPrimary Event wird ausgelöst, sobald Kontaktdaten als primär definiert werden sollen.

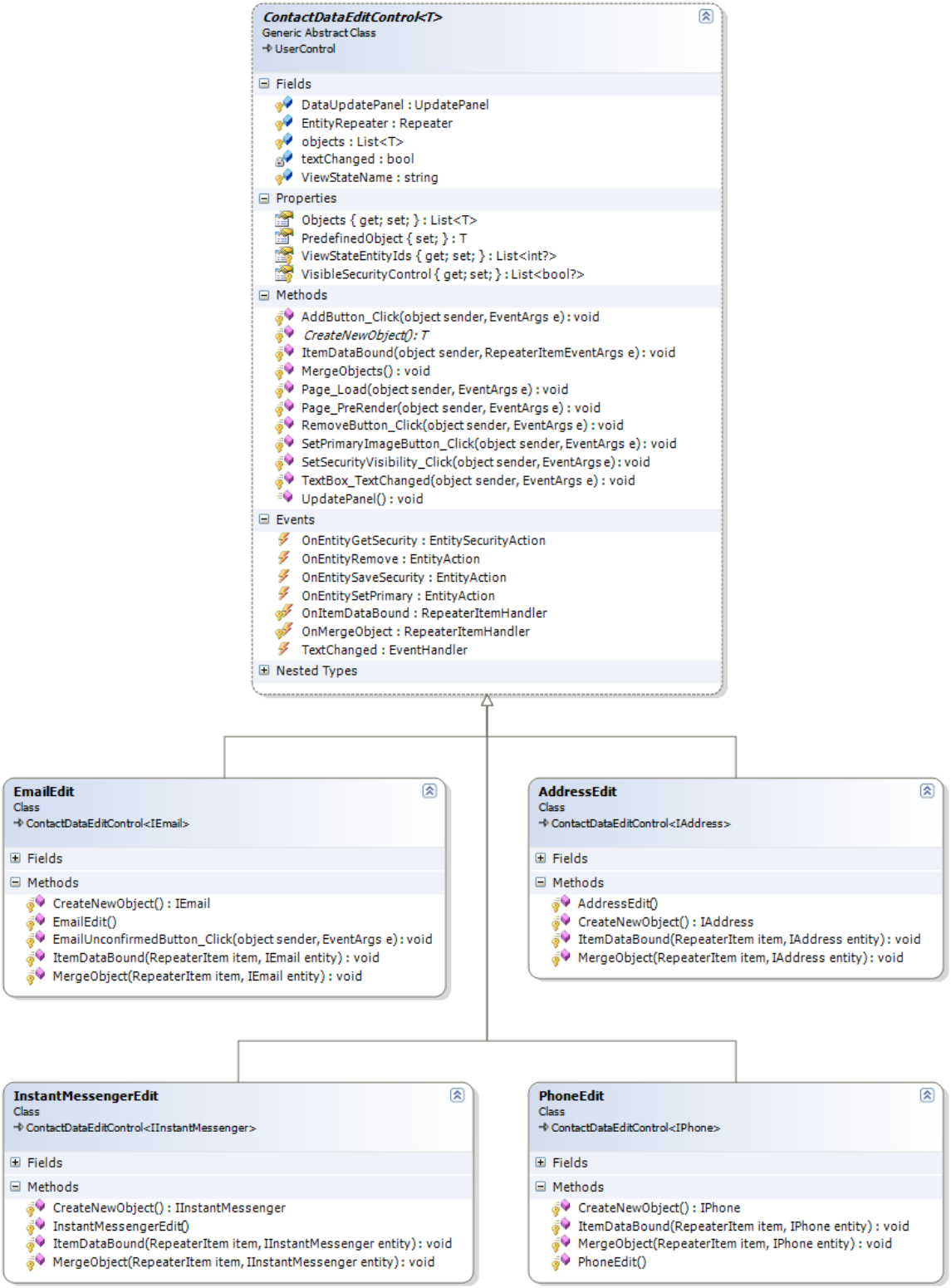

Abbildung 46: Klassendiagramm der ContactEditControls

7.7.2 GenericEnumDropDownList

Um aus den verschiedenen Werttypen jeweils eine DropDownListe zu generieren, wurde ebenfalls eine abstrakte Basisklasse erstellt. Mithilfe von Generics konnte auch hier viel redundanter Code eingespart werden. Den DropDownListen kann jeweils direkt ein Enum gesetzt werden und dieser kann ohne Umwandlung auch wieder direkt ausgelesen werden.

Der RessourceManager sorgt dafür, dass der Benutzer sinnvolle Namen in der Liste sieht.

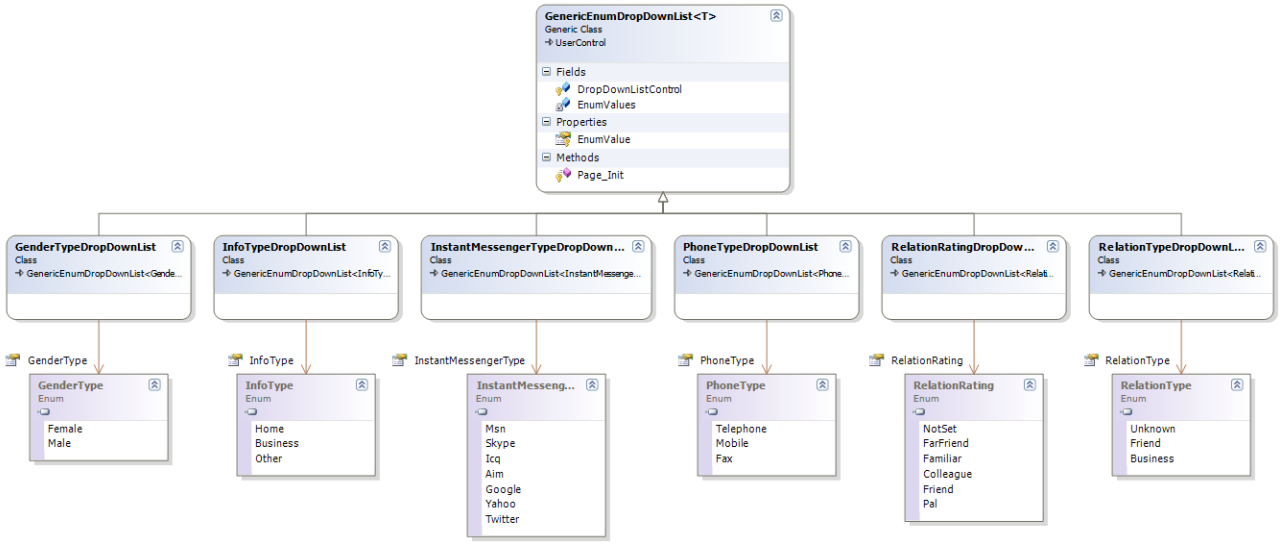

Abbildung 47: Klassendiagramm der verschiedenen DropDownListen

7.7.3 Menü

ASP.NET bietet von Haus aus eine eigene Komponente zum Erstellen von Menüs. Dieses bietet die Möglichkeit, eine aufwändige und in mehreren Hierarchien aufgebaute Menüstruktur abzubilden. Das Control bietet viel, war dadurch aber auch komplex bei der Konfiguration und erzeugte unschönen HTML-Code. Zusätzlich war dieses Menü nur mit Javascript nutzbar, wodurch es stark an Attraktivität verlor.

Aus diesen Gründen wurde ein eigenes Menü implementiert, das einfach in der Handhabung ist, schlanken Code generiert und sowohl mit als auch ohne Javascript funktioniert.

Abbildung 48: Das generierte Menü

Control für Menuknöpfe

Damit die Menüknöpfe auch ohne Javascript funktionieren, mussten wir ein Control wählen, welches bei einem Klick ein PostBack auf den Server ausführt. Aus HTML-technischen Gründen eignen sich hierfür nur wenige Controls. Zuerst wurde eine Lösung mit Hyperlinks gesucht. Diese bieten aber keine Serverseitigen OnClick Event Handler, da ein Klick auf einen Link keinen PostBack auslöst. Dadurch wäre es nicht möglich gewesen, das Menü in einem UpdatePanel für asynchrone Updates zu nutzen. In einer zweiten Lösung wurden die Knöpfe mittels ButtonControls, die dem HTML Submit Button entsprechen, implementiert, was zu dem gewünschten Resultat führte.

Das Control ermöglicht es sogar, für jeden Knopf zu definieren, ob er einen Event auslösen soll, auf den dann in einem EventHandler reagiert werden kann, oder ob direkt auf eine Url weitergeleitet werden soll.

Zusätzlich kann für jeden Knopf ein Bild definiert werden.

Quellcode 9 Konfiguration eines Menüs in ASP.NET

```
<uc:Menu ID="EditMenu" OnClick="MenuItem_Click" OnSelectedItemChanged="
    MenuItemSelectedItemChanged" runat="server">
     <Items>
         <uc:MenuItem runat="server" Selected="true" Image="~/design/groups.
            png" Text="Person" Value="0" />
         <uc:MenuItem runat="server" Image="~/design/address_book3.png" Text=
            "Adressen" Value="1" />
         <uc:MenuItem runat="server" Image="~/design/mail.png" Text="Email"
            Value="2" />
         <uc:MenuItem runat="server" Image="~/design/mobile.png" Text="
            Telefon" Value="3" />
         <uc:MenuItem runat="server" Image="~/design/message.png" Text="
            Instant Messengers" Value="4" />
     </Items>
</uc:Menu>
```

7.7.4 PersonList

Die Auflistung von Personen wird an vielen Stellen verwendet und deshalb wurde ein eigenes User-Control für diese Aufgabe programmiert. Dem Control kann eine Liste von Personen gesetzt werden. Zudem kann man die Anzahl angezeigter Personen pro Seite anpassen.

Das Control bildet die Basis für das spezialisierte GroupPersonListControl, das für die Auflistung von Gruppenmitgliedern konzipiert wurde. Dabei werden die meisten Eigenschaften des PersonList-Controls übernommen und mit zusätzlichen gruppenspezifischen Eigenschaften angereichert. So wird im GroupPersonListControl zusätzlich angezeigt, ob ein Mitglied der Gruppe bereits beigetreten ist und ob sie über Administrationsrechte verfügt.

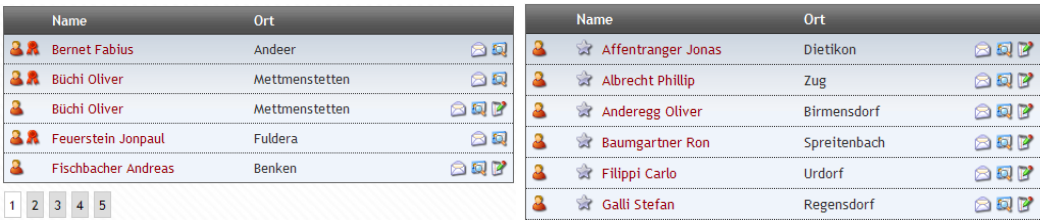

Abbildung 49: PersonListControl und das spezialisierte GroupPersonListControl

7.7.5 PersonSelect

Will man eine oder mehrere Personen auswählen, sei das für den Versand von Nachrichten oder um die Mitglieder einer Gruppe zu bestimmen, wird das PersonSelectControl verwendet. Das Control stellt alle verfügbaren Kontakte in einer Selectbox dar. Eine zweite Selectbox dient zur Anzeige der ausgewählten Personen. Um eine Person der Selektion hinzuzufügen, wählt man einen oder mehrere Kontakte aus und verschiebt sie mit Hilfe der entsprechenden Buttons in die andere Box.

Durch die selbst entwickelten Ajax Extender ist es zudem möglich, die Liste komplett mit der Tastatur zu bedienen. Bei einer Texteingabe werden die Elemente durchsucht und Treffer markiert. Beim Betätigen der Enter Taste werden die selektierten Personen in die gegenüberliegende Selectbox verschoben.

Abbildung 50: Automatische Auswahl von Personen bei der Eingabe von Namen

7.8 Messaging

7.8.1 Zeitversetztes Senden

Um ein zeitversetztes Senden von Nachrichten zu ermöglichen, wird ein Dienst benötigt, der im Hintergrund läuft und in regelmässigen Abständen überprüft, ob eine Nachricht versendet werden muss. Dabei wird für jede Versandart ein eigener Dienst erstellt. Diese verarbeiten bei jeder Ausführung alle anstehenden Nachrichten und versenden diese über die entsprechende Schnittstelle. Die Dienste werden als Daemon-Threads gestartet und laufen parallel zueinander ab.

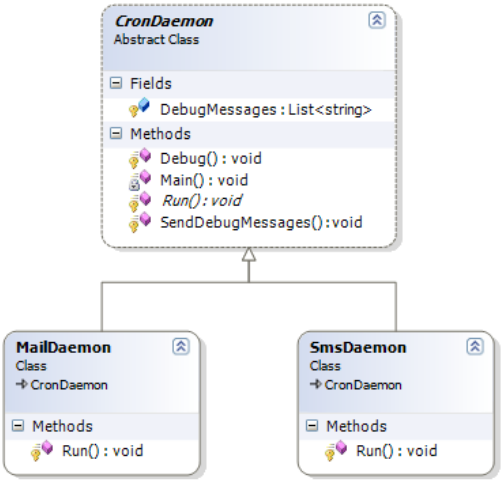

Abbildung 51: Aufbau der Messaging Cronjobs

7.8.2 SMS Konfiguration

Eine SMS Konfiguration ist ein Set von Einstellungen zu einem SMS-Anbieter, das sich je nach Anbieter unterscheiden kann. Einige verlangen die Mobiltelefonnummer als Benutzername während andere dazu die Emailadresse verwenden.

7.8.3 Versand von SMS Nachrichten

Wie in Kapitel 4.6.3 auf Seite 28 beschrieben, gibt es kaum gemeinsame Schnittstellen zwischen den einzelnen SMS-Anbietern. Deshalb soll unabhängig vom Anbieter eine allgemeine Schnittstelle erstellt werden. Für jeden einzelnen Anbieter wird dann eine konkrete Implementierung erstellt.

Der Aufwand dieser Implementationen unterscheidet sich je nach Art des Anbieters stark. Bei Anbietern, die keine Web-Schnittstelle anbieten, muss man sich zuerst anmelden, bevor man eine Nachricht versenden kann, während bei solchen mit einer eigenen Schnittstelle der Versand meist mit einem einzigen Aufruf möglich ist.

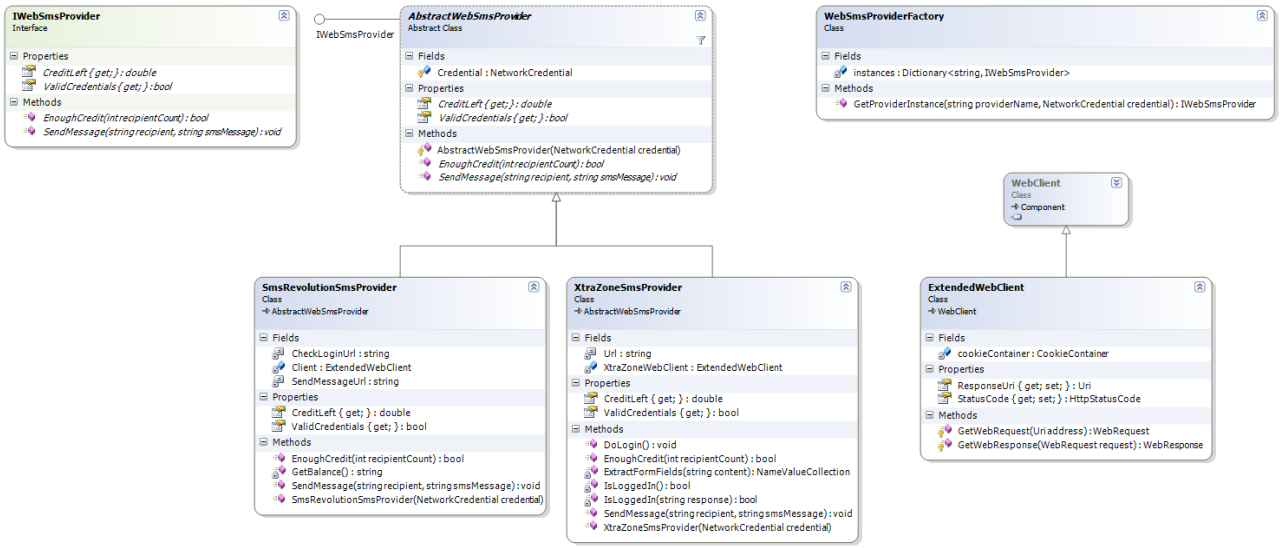

Abbildung 52: Klassendiagramm für SMS Provider

7.8.4 Anbieter: Xtra Zone

Xtra Zone ist ein Dienst des Mobilfunkanbieters Swisscom, der allen Kunden, die jünger als 26 Jahre sind, den kostenlosen Versand von 500 SMS Nachrichten pro Monat ermöglicht. Für diesen Dienst gibt es keine offizielle Schnittstelle. Das Versenden von Nachrichten wird direkt über die Webseite von Xtra Zone abgehandelt.

Für die Implementation dieses Dienstes wurde der HTML Parser HtmlAgilityPack (Kapitel 7.13.2 auf Seite 86) verwendet. Dabei wird der Inhalt der Seite ausgelesen, die Formulare ausgewertet, mit Daten abgefüllt und an die Seite zurückgesendet.

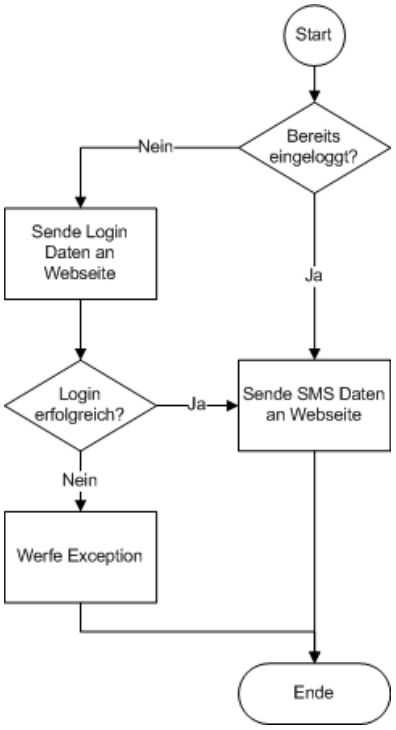

Abbildung 53: Ablauf des Versands von SMS über Xtra Zone

7.8.5 Anbieter: SMS-Revolution

SMS-Revolution ist ein kostenpflichtiger Anbieter von SMS Versand. Er wurde integriert, damit auch Nutzer von anderen Mobilfunkanbietern die Möglichkeit des SMS Versandes nutzen können.

Dieser Anbieter bietet verschiedene Schnittstellen an, um auf seinen Dienst zuzugreifen.

Schnittstellen

- HTTP-Schnittstelle (Get und Post)
- XML-Post Schnittstelle
- SOAP-Gateway
- mail2SMS

Neben dem Versenden von SMS Nachrichten können auch weitere Informationen abgerufen werden.

- Kontostandabfrage
- Adressbuch-Schnittstelle (Import/Export)
- Abfrage von Posteingang und Postausgang
- MassenSMS Gateway

7.9 Gruppen

Um Vereinen oder anderen Gruppierungen eine einfache Kommunikation untereinander zu ermöglichen, wurden die Gruppen eingeführt. Dabei hat eine Person die Möglichkeit, eine Gruppe zu erstellen und dieser eine oder mehrere Kontakte anzufügen. Die einzelnen Mitglieder müssen untereinander nicht in Beziehung stehen, um der selben Gruppe angehören zu können. Eine Gruppe wird dabei von einem oder mehreren Administratoren verwaltet. Diese haben die Möglichkeit, der Gruppe neue Mitglieder hinzuzufügen. Die Person muss einem Gruppenbeitritt jeweils zustimmen.

Der Beitritt zu einer Gruppe ist nur mit Einladung möglich. Damit soll dem Datenschutz Rechnung getragen werden. Die Möglichkeit eine Emailadresse für eine Gruppe zu erstellen wurde bisher noch nicht implementiert, da hierzu noch ein POP-Client erstellt werden muss. Auch lokale Gruppierungen wurde bisher noch nicht integriert, da deren Nutzen als eher klein eingestuft wurde.

Abbildung 54: Administration einer Gruppe

7.10 Kontaktvisualisierung

7.10.1 Einbindung von neuen Karten

Die Visualisierung der Kontakte wurde umfassend erweitert. Neben dem bereits integrierten GoogleMapsControl sollte nun auch das Kartenmaterial von Microsoft integriert werden. Für die Einbindung der Windows Live Karte in die .NET Umgebung stand leider erst ein Community Technology Preview zur Verfügung. Dieses hatte zwar einige kleine Bugs, lief im grossen und ganzen aber stabil.

Das Control von Mircosoft bietet sehr interessante neue Möglichkeiten zur Visualisierung. In einem 3D-Modus kann man zwischen seinen Kontakten hin- und herfliegen. In den grossen Ballungsgebieten stehen zusätzlich isometrische Luftaufnahmen zur Verfügung, die einen enormen Detailreichtum haben.

Abbildung 55: Verschiedene Möglichkeiten der Kontaktvisualisierung. Im Uhrzeigersinn von oben links: 1. VirtualEarth 3D Ansicht, 2. GoogleMaps Satelitenkarte, 3. Statische Karte ohne Javascript, 4. Terrainkarte von Google, 5. Mircosoft Bing Strassenkarte, 6. Microsoft BirdseyeView Nahansicht

7.10.2 Refactoring des MapControls

Um das Integrieren der zusätzlichen Karten zu vereinfachen, wurde das bisherige MapControl überarbeitet. Es wurde eine abstrakte Klasse erstellt, welche sämtliche benötigten Methoden definiert und implementationsunabhängige Methoden wie die Berechnung des Blickfeldes implementiert. Um eine neue Karte zu integrieren, musste danach nur noch von dieser abstrakten Klasse abgeleitet werden und in den abstrakten Methoden der entsprechende Befehl an die Karte weitergereicht werden.

Um dem Benutzer ein einfaches Wechseln zwischen den vierschiedenen Karten zu ermöglichen, wurde schliesslich noch eine Wrapperklasse (MapControl) erstellt. Diese ist dafür zuständig, dass die richtige Karte angezeigt wird und die anzuzeigenden Personen sowie der aktuelle Blickwinkel bei einem Wechsel an die neue Karte weitergereicht werden.

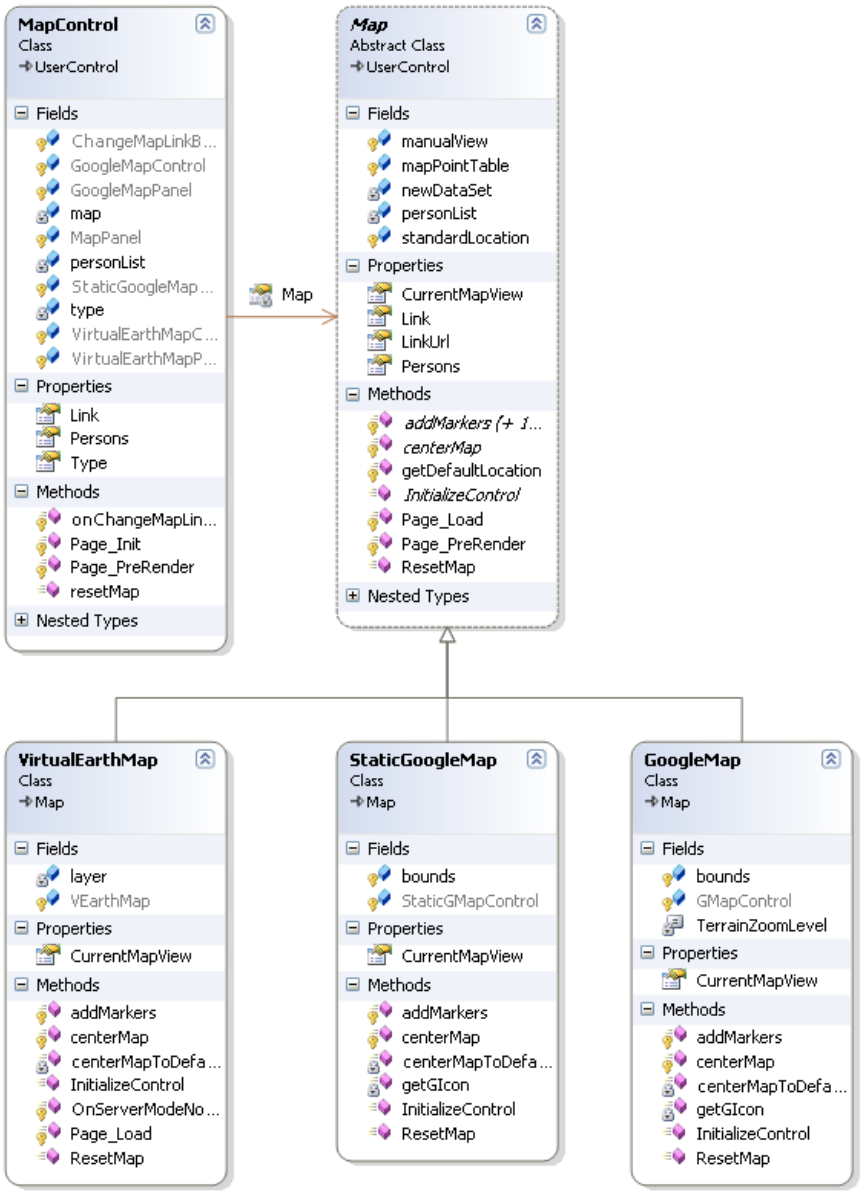

Abbildung 56: Aufbau der Visualisierungsklassen

7.11 WCF Service

Durch die umfassende Neugestaltung der Business- und Domain-Schicht konnte der alte WCF Service nicht weiter betrieben werden. Wurden die Objekte in der Studienarbeit für den WCF Service noch in DTO (Data Transfer Objects) umgewandelt, können neu direkt die von der Business Schicht zurückgegebenen Objekte verwendet werden. Dank dem Einsatz von POCO's (Kapitel 7.5) entfällt die lästige und langsame Konvertierung mittels Reflection.

Attributierung

WCF bietet mit der Attributierung der Datenobjekte eine flexible Möglichkeit, genau zu definieren, welche Daten übertragen werden sollen [6]. So können Felder, die für den Weblayer verwendet werden, für den Service ausgeblendet werden. Wie im unten stehenden Beispiel zu sehen, werden nur die Felder über den Service angeboten, die mit dem Attribut DataMember gekennzeichnet wurden.

Quellcode 10 Attributierung eines Daten Objekts für die Verwendung mit WCF

```
[DataContract]
public class Email : IEmail
{
    [DataMember]
    public int? Id { get; set; }

    [DataMember]
    public string Address { get; set; }

    [DataMember]
    public bool Confirmed { get; set; }

    public bool IsPublic { get; set; }
}
```

Der Service kann als Service Reference in ein Projekt eingebunden werden und steht unter der URL `http://www.relate.ch/services/Gateway.svc` zur Verfügung.

7.12 Outlook Add-In

Als Implementationsbeispiel wurde ein Outlook Add-In erstellt. Dieses erstellt für alle Kontakte einen neuen Eintrag in den Outlook Kontakten. Die Funktionalität des Add-In ist sehr begrenzt. Es werden bei jeder Synchronisation erneut neue Kontakte erstellt. Das Add-In soll derzeit nur zeigen, dass die Synchronisation über die Service Schnittstelle möglich ist. In einem späteren Schritt soll das Add-In so ausgebaut werden, dass eine bidirektionale Synchronisation möglich wird. Die Funktionalität würde den Rahmen dieser Arbeit aber überschreiten und wurde daher bewusst tief gehalten.

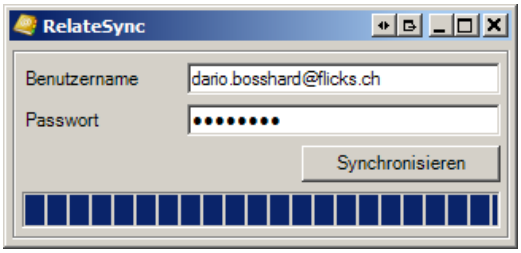

Abbildung 57: Outlook Add-In - RelateSync

7.13 Verwendete Drittbibliotheken

7.13.1 String Template

Problem

Will man Strings mit Platzhaltern verwenden, die je nach Aufruf einen anderen Inhalt aufweisen, stösst man schnell auf Probleme. So ist das Verketten von Strings mit Variablen sehr langsam und umständlich. Zusätzlich ergeben sich schnell riesige Codefragmente, die kaum noch lesbar oder gut zu verstehen sind. Will man nun Texte auslagern und übersetzen, wird rasch eine Komplexität erreicht, die kaum mehr zu bewältigen ist.

Quellcode 11 Ersetzen von Platzhaltern mit String Format

```
string helloConcat = "Hello " + person.Firstname + " " + person.Lastname;
string helloFormat = string.Format("Hello {0} {1}", person.Firstname, person
    .Lastname);
```

Idee zur Problembewältigung

Als Lösung solcher Probleme bieten sich Template Engines an. Diese ersetzen in einem String definierte Stellen automatisch mit Daten. Somit lassen sich die Texte einfach ausgliedern und bearbeiten und sind sauber vom Code getrennt. Das Ersetzen von einzelnen Strings ist relativ einfach umsetzbar, wie in Quellcode 13 zu sehen ist.

Quellcode 12 Ersetzen von Platzhaltern mit String Replace

```
string template = "Hello {firstname} {lastname}";
template.Replace("{firstname}", person.Firstname);
template.Replace("{lastname}", person.Lastname);
```

Die Bibliothek String Template bietet einen flexiblen Ansatz zur Bewältigung dieses Problems. Statt einzeln alle Platzhalter ersetzen zu müssen, können mit einem Template ganze Objekte assoziiert werden. In den Platzhaltern kann dann auf die Properties dieser Objekte zugegriffen werden. Wie dies möglich ist, zeigt Quellcode 13.

Quellcode 13 Ersetzen von Platzhaltern mit StringTemplate

```
StringTemplate template = new StringTemplate("Hello {person.Firstname} {
    person.Lastname}");
template.SetAttribute("person", person);
```

Vorteile dieser Drittbibliothek

- Einfache und flexible Template Engine
- Wenig Code notwendig
- Direktzugriff auf Properties senkt die Abhängigkeit

Weitere Informationen zu StringTemplate

`http://www.stringtemplate.org/`

7.13.2 HTMLAgilityPack

Problem

Das Versenden von SMS wird von den meisten Mobilfunkanbietern als Webdienst angeboten. Dabei meldet sich ein Kunde auf der Webseite an und kann dann eine begrenzte Anzahl SMS versenden. Damit SMS automatisiert versendet werden können, muss der gleiche Ablauf, der vom Benutzer auf der Seite ausgeführt wird, programmiertechnisch nachgestellt werden.

Realisierung

Die Requests auf die Seite werden mittels der Klasse WebClient realisiert, welche Bestandteil von .NET 3.5 ist. Um den Inhalt einer angefragten Seite zu verarbeiten, ist es nötig, den HTML Code zu parsen. Das Problem bei HTML Seiten ist aber, dass sie meist nicht standardkonform sind oder nicht den einfach maschinenlesbaren XHTML Standard verwenden. Somit ist es mit den gegebenen Funktionen der .NET Bibliotheken sehr umständlich, eine solche Seite vernünftig zu parsen.

Das HTMLAgilityPack schliesst diese Lücke. Im HTML Code kann dabei per XPath navigiert werden. So können Formulardaten ausgelesen, manipuliert und an die Seite gesendet werden.

Vorteile Drittbibliothek

- Nicht standardkonformes HTML ist verarbeitbar
- Einfacher Zugriff auf HTML Knoten

Nachteile

- Fehlerhaftes HTML wird unter Umständen falsch interpretiert

Weitere Informationen zu HTMLAgilityPack

`http://www.codeplex.com/htmlagilitypack`

7.13.3 GeoCoding

Um die Adresse auf einer Karte zu visualisieren, ist es notwendig, diese in Geokoordinaten umzuwandeln. Für diesen Zweck bieten Google, Microsoft, Yahoo und andere Unternehmen passende Dienste an. Diese Dienste lassen sich mithilfe der GeoCoding Library über eine API ansteuern. Die GeoCoding Daten helfen auch dabei, falsch geschriebene Adressen zu korrigieren. So kann z.B. auf die Eingabe einer Postleitzahl verzichtet werden. Kann die Adresse zugeordnet werden, werden von den Diensten die korrekte Postleitzahl sowie die Geokoordinaten zurückgeliefert. Ohne diese Dienste wäre eine Visualisierung der Daten nicht möglich. Leider funktionieren diese Dienste nicht immer wie gewünscht. Probleme, die wir mit den GeoCoding Diensten hatten, werden im Kapitel 7.14.2 auf der nächsten Seite beschrieben.

Weitere Informationen zum GeoCoding API

`http://code.google.com/p/geocoding-net/`

7.13.4 GMap

Das GoogleMapsControl von Subgurim ist das umfangreichste GoogleMaps-Control für ASP.NET 2.0. Mit dem vollen Funktionsumfang der offiziellen GoogleMaps API, ohne die Notwendigkeit einer einzigen Zeile JavaScript-Code.

Weitere Informationen zum GMapControl
`http://en.googlemaps.subgurim.net/`

7.13.5 VirtualMap

Das .NET Control für VirtualEarth von Microsoft steht leider bisher erst als Community Technical Preview zur Verfügung. Dieses hat bisher noch einige Fehler, überzeugt aber ansonsten durch seinen Funktionsumfang.

Weitere Informationen zum CTP des VirtualEarthControl
`http://dev.live.com/tools/`

7.14 Aufgetretene Probleme (Problem, Lösung, inklusive Tradeoff)

7.14.1 Installation

Leider wurde uns nach der Studienarbeit für diese Bachelorarbeit ein neuer Arbeitsplatz zugeteilt. Dabei mussten wir sämtliche Software erneut installieren. Bei dieser Installation gab es einiges zu beachten, damit man an beiden Arbeitsplätzen möglichst reibungslos zusammenarbeiten konnte.

SQL Server 2008
Da der Server von Hostfactory mit der neusten Version des SQL Servers betrieben wird, entschieden wir uns für eine Installation des SQL Server 2008. Der Installationsprozess wurde gegenüber der 2005er Version komplett überarbeitet. Dies führte zu einigen Verwirrungen und daraus folgenden Neuinstallationen. Der Server wurde analog zu demjenigen von Hostfactory konfiguriert.

Datenbank Instanz	SQLExpress
Datenbankname	CL45200_RELATE
Datenbank-Benutzer	CL45200_DBUSER

Tabelle 4: Konfigurations-Variablen des SQL Servers

Projekt Verzeichnis
Damit es beim Arbeiten an der Dokumentation und am Quelltext nicht zu Problemen, z.B. mit Pfadangaben kommt, wird bei allen Computern mit dem SVN Root C:\Relate gearbeitet.

7.14.2 GeoCoding mit falschen Resultaten

Das Auflösen von Adressdaten zu Geo-Koordinaten bringt einige Tücken mit sich.

Findet der GeoCoding Dienst von Google eine Adresse nicht, so versucht er eine möglichst ähnliche zu finden. So können auch Adressen aufgelöst werden, die falsch geschrieben wurden. Leider ist diese Korrektur nicht immer hilfreich. So führt Google die Oberdorfstrasse in Dietikon als Ober Dorfstrasse. Eine Suche nach einer Adresse an dieser Strasse wird von Google nicht erkannt und zu Oberdorfplatz in Oetwil an der Limmat korrigiert.

Abbildung 58: Google GeoCoding Korrektur

Die gleiche Anfrage beim Yahoo-Dienst liefert die korrekte Adresse. Jedoch ist auch das Resultat der Yahoosuche nicht perfekt. Die Postleitzahl wird im selben Feld wie die Ortschaft zurückgeliefert, während das Feld für die Postleitzahl leer bleibt. Um dieses Problem zu beheben wurde eine Extension Methode erstellt, die mittels Regular Expressions die Felder korrekt ausfüllt.

Quellcode 14 Korrektur des Yahoo Rückgabefehlers

```
public static void CorrectZipCode(this GeoCoding.Address address)
{
    if (string.IsNullOrEmpty(address.PostalCode) && Strings.Regex.Contains(
        address.City, @"^[0-9]{4}"))
    {
        address.PostalCode = address.City.Substring(0, 4);
        address.City = address.City.Substring(4);
    }
}
```

Da beide API's je nach Adresse unterschiedliche Resultate zurückliefern können, wird jeweils nur das bessere der beiden Resultate verwendet. Um zu entscheiden, welches Resultat besser ist, wurde ein einfaches Bewertungssystem programmiert. Stimmt ein Suchfeld mit einem Resultatfeld überein, wird ein Qualitätspunkt verteilt. Das Resultat mit der höheren Punktzahl wird bevorzugt.

Quellcode 15 Bewerten des GeoCoding Resultats

```
private static int CalculateGeocodeQuality(IAddress address, GeoCoding.
    Address geoCodeAddress)
{
    int quality = 0;
    if (address.Street.Equals(geoCodeAddress.Street))
    {
        quality++;
    }

    if (address.City.Equals(geoCodeAddress.City))
    {
        quality++;
    }

    if (address.Zip.Equals(geoCodeAddress.PostalCode))
    {
        quality++;
    }

    if (address.Country.Equals(geoCodeAddress.Country))
    {
        quality++;
    }
    return quality;
}
```

7.14.3 SQL Constraints

Bei der Umsetzung des Datenbankmodells auf den Microsoft SQL Server 2008 trat ein unerwartetes Problem auf. Im Gegensatz zu anderen DBMS unterstützt die Implementierung von Microsoft keine Cascading Constraints, welche eine Schleife bilden [18].

Das heisst, eine Fremdschlüsselbedingung kann nicht erstellt werden, wenn es dadurch ermöglicht würde, dass eine Tabelle in einer kaskadierenden Aktionsliste mehrmals vorkommt. Dadurch ist es beispielsweise nicht möglich, per Constraints zu definieren, dass beim Löschen einer Person alle Beziehung von und zu dieser gelöscht werden, weil sowohl das sourcePerson_id als auch das destinationPerson_id Feld auf die selbe Tabelle einen Fremdschlüssel haben und diese daher nicht beide einen Cascading Constraint haben dürfen.

Um dieses Problem zu umgehen und trotzdem alle Konsistenzbedingungen in der Datenbank definieren zu können, mussten als Workaround entsprechende Triggers geschrieben werden. Mehr Informationen zur Implementation dieser Triggers finden sich unter 7.1 auf Seite 63.

7.14.4 Querystring Verschlüsselung

Problembeschreibung

Zur Steigerung der Sicherheit kann der Querystring einer URL verschlüsselt werden. Bei der Verschlüsselung eines Strings wird ein Bytecode erzeugt, der den ASCII Zeichenraum überschreitet und in dieser Form nicht als Parameter in der URL übergeben werden kann.

Damit der Bytecode als Querystring in der URL mitgegeben werden kann, wird er in Base64 Encodiert. Base64 endet jedoch bei langen Bytecodes meist auf das Zeichen =, welches in der URL als Trennzeichen für Schlüssel und Wert verwendet wird. Damit dies korrekt unterschieden werden kann, werden Schlüssel und Wert URL-Encodiert. Dabei wird das Zeichen = als %3d repräsentiert.

Anfrage	`?personId=8`
Schlüssel	`personId`
Wert	`8`

Tabelle 5: Normale Anfrage

Anfrage	`?TCfuZ0NvFGy6ekWtd7AeEA%3d%3d=ahBiUcRln4a7J49b4xBw0A%3d%3d`
Schlüssel	`TCfuZ0NvFGy6ekWtd7AeEA==`
Wert	`ahBiUcRln4a7J49b4xBw0A==`

Tabelle 6: Verschlüsselte Anfrage

In Zusammenspiel mit Javascript kann es jedoch vorkommen, dass der Querystring zweifach URL-decodiert wird. Der Server erhält nun eine veränderte Anfrage. Der Server wertet diese Anfrage aus und teilt die Anfrage in Schlüssel und Wert auf, wie in 7 zu sehen ist.

Anfrage	`?TCfuZ0NvFGy6ekWtd7AeEA===ahBiUcRln4a7J49b4xBw0A==`
Schlüssel	`TCfuZ0NvFGy6ekWtd7AeEA`
Wert	`==ahBiUcRln4a7J49b4xBw0A==`

Tabelle 7: URL-decodierte Anfrage nach zweifachem Decodieren.

Da die Aufsplittung der Werte fehlschlägt, können die veränderten Werte so nicht mehr entschlüsselt werden. Der Quersystring ist unbrauchbar.

Problembehebung
Damit der Querystring korrekt entschlüsselt werden kann, muss die Aufspaltung in Schlüssel und Wert selbst realisiert werden. Anstelle des ersten Gleichheitszeichen soll jeweils das letzte für die Aufspaltung verwendet werden.

```
RegularExpressions.Regex.Split(keyValuePair, @"=([^=])")
```

Wurde das Problem damit wirklich gelöst?
Leider wurde damit das Problem noch immer nicht gelöst. Das Zusammenspiel von Base64-Encodierten Querystrings und dem Javascript basierten VirtualEarthControl führte immer wieder zu Fehlinterpretationen. Das Problem waren die Zeichen ausserhalb des Zahlen- und Buchstabenbereichs, die speziell codiert werden mussten. Base64 stellte sich nicht als die ideale Codierung heraus. Als Alternative wählten wir eine Hex-Codierung. Dies führt zu längeren Strings. Es hat jedoch den Vorteil, dass sie ohne Url-Encodierung auskommt und somit gut mit Javascript verwendet werden kann.

Anfrage	`?4c27ee67436f146cba7a45ad77b01e10=6a106251c4659f86bb278f5be31070d0`
Schlüssel	`4c27ee67436f146cba7a45ad77b01e10`
Wert	`6a106251c4659f86bb278f5be31070d0`

Tabelle 8: URL-decodierte Anfrage

7.14.5 Root Zertifikat von Externen Servern in Verbindung mit WebClient

Die in Kapitel 7.13.2 auf Seite 86beschriebene Verwendung der WebClient Klasse funktioniert bei Seiten mit verschlüsselter Verbindung nur dann, wenn der Client das Zertifikat des Servers verifizieren kann.

Im konkreten Fall funktionierte die Kommunikation mit der Swisscom Mobile Webseite nicht. Mit der folgenden Lösung kann die Validierung der Zertifikate standardmässig als gültig klassifiziert werden.

Quellcode 16 Verifizierung von Externen SSL Zertifikaten

```
ServicePointManager.ServerCertificateValidationCallback += new
    RemoteCertificateValidationCallback(OnCheckSSLCert);

private static bool OnCheckSSLCert(object sender, X509Certificate
    certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors)
{
    return true;
}
```

7.14.6 ASP.NET Page Lifecycle

Jeder Request an eine ASP.NET Seite durchläuft die HTTP-Pipeline, eine Kette von verwalteten Objekten, die den Request sequenziell abarbeiten. Ziel der ASP.NET-Pipeline ist es, die eingehende Anforderung in eine Instanz einer Klasse umzuwandeln, die die angeforderte ASP.NET-Seite repräsentiert.

ASP Seiten und UserControls bieten jeweils Events an, die an bestimmten Stellen der Verarbeitung ausgelöst werden. Die Kenntnis über die Funktionsweise und Reihenfolge dieser Events ist essentiell, da HTTP ein zustandsloses Protokoll ist und daher Werte von Objekten bei jedem neuen Request wieder verloren sind. Um trotzdem ein konsistentes Verhalten zu simulieren, müssen Techniken wie die SESSION oder der von Microsoft entwickelte VIEWSTATE eingesetzt werden.

Nach einem eingehenden Studium der Funktionsweise des Page Lifecycle und dem Fund einer praktischen Illustration, konnten schliesslich die jeweils richtigen Aufgaben am richtigen Ort gelöst werden.

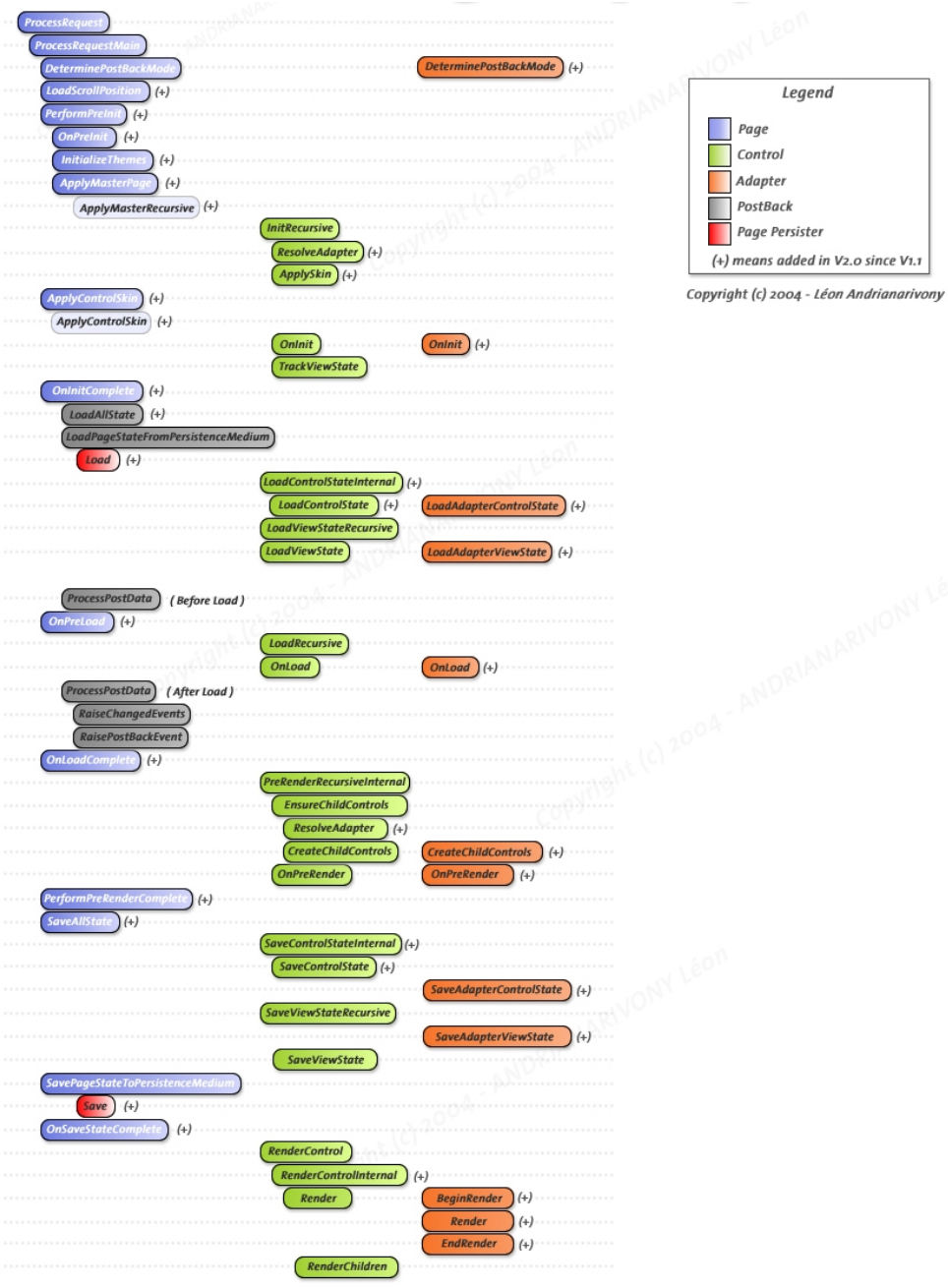

Abbildung 59: Ablauf des ASP.NET Page Lifecycle (ľ Leon Andrianarivony)

8 Testing

8.1 Unit Tests

Für eine zuverlässige Businessschicht ist es unerlässlich, dass die einzelnen Funktionen systematisch getestet werden. Es wurden daher Unit Tests für sämtliche Controller erstellt. Dies erwies sich vor allem bei den umfassenden Refactorings als sehr nützlich, da nach jedem Refactoring jeweils direkt getestet werden konnte, ob die Methoden weiterhin wunschgemäss funktionierten. Das Visual Studio bietet umfassende Unterstützung für das Testing. Über eine detaillierte Liste wird , wie in Abbildung 60 zu sehen, über die Testresultate informiert.

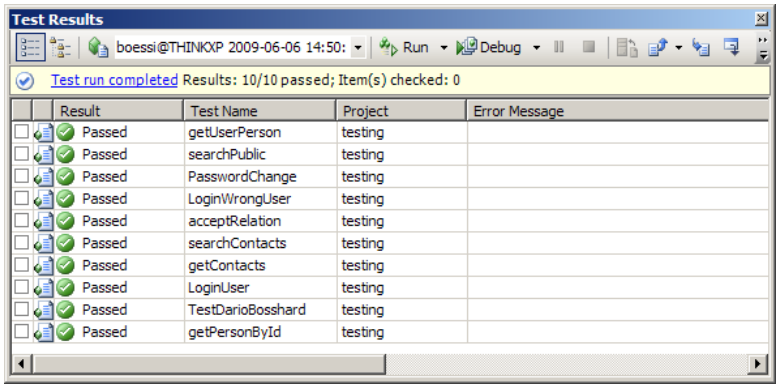

Abbildung 60: Resultate eines Testlaufes

8.2 Usability Tests

Nebst der reibungslosen Funktion des Backend ist auch die Bedienfreundlichkeit der Benutzeroberfläche wichtig. Daher wurden in verschiedenen Stadien der Entwicklung Usability Tests durchgeführt. Die Testpersonen wurden nur kurz instruiert und nicht aktiv bei ihrer Aufgabe unterstützt. Sämtliche Tests fanden als Feldversuche und nicht in speziell ausgerüsteten Labs statt. Genaue Beobachtungen und Gespräche nach den Tests haben dann jeweils zu interessanten Anregungen und einer Verbesserung der Benutzeroberfläche geführt.

Nachfolgend findet sich eine kurze Zusammenfassung der Resultate. Zur besseren Übersicht wurden diese nicht nach Testpersonen, sondern nach Use Cases geordnet.

8.2.1 Kontaktübersicht

Beobachtung

Die Kontaktübersicht war für viele Testpersonen überladen und unübersichtlich. Die Visualisierung der Kontakte am Ende der Kontaktliste fiel nicht direkt ins Auge, da dazu oft nach unten gescrollt werden musste.

Resultierende Änderung

Die Kontaktübersicht wurde entschlackt. Wurde bis anhin noch eine komplette Adresse angezeigt, wird nun nur noch der Ortsname erwähnt. Die Emailadresse wurde durch ein Icon erstetzt, das zur Messaging-Ansicht führt. Die Visualisierung auf der Karte wurde prominenter und grösser platziert.

Abbildung 61: Usability Tests - Überarbeitung der Kontaktliste

8.2.2 Daten Editieren

Beobachtung

Beim Editieren der Kontaktdaten musste man schon bei wenigen Daten nach unten scrollen, was zu einer verminderten Übersichtlichkeit führte.

Resultierende Änderung

Die einzelnen Kontaktinformationen werden nach ihren Gruppen aufgeteilt und mit einem Menü gegliedert. Damit sind die Informationen jederzeit übersichtlich erreichbar.

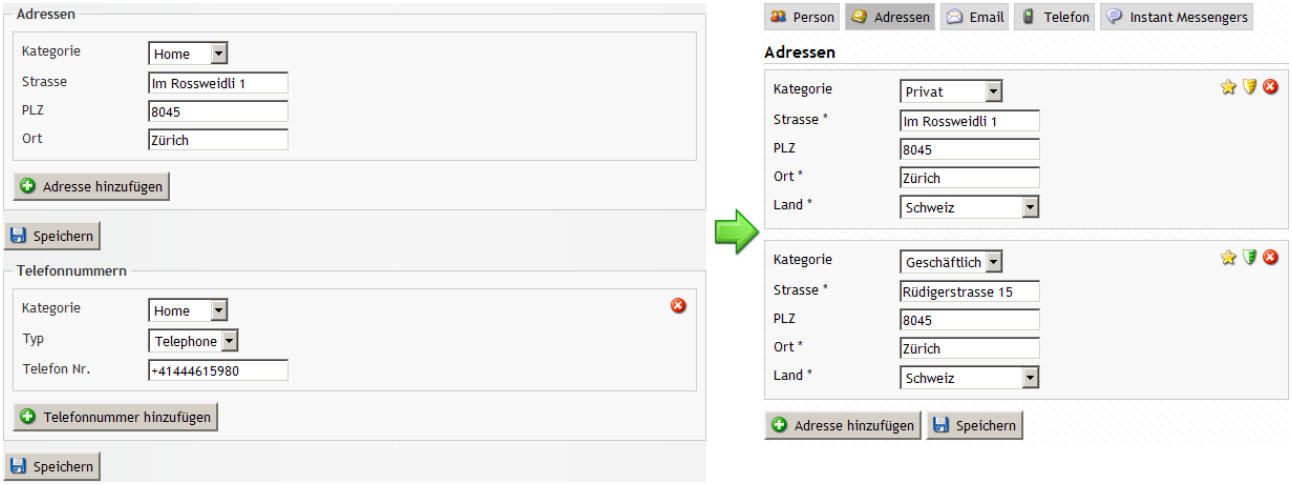

Abbildung 62: Usability Tests - Überarbeitung der Editiermaske für Kontaktdaten

8.2.3 Suche nach Personen

Beobachtung

Bewegte man sich ausserhalb der Startseite, war es nur über mehrere Klicks möglich, an Daten eines Kontaktes zu gelangen. Dazu musste zuerst auf die Startseite gewechselt werden und dann auf den ensprechenden Kontakt.

Resultierende Änderung

In der Navigationsliste wurde ein Suchfeld integriert, das auf jeder Seite ersichtlich ist. Gibt man die ersten Buchstaben eines Kontaktes ein, werden Kontakte, die auf die Buchstaben zutreffen, zur Auswahl vorgeschlagen. Werden mehrere zutreffende Kontakte gefunden, erhält man zuerst eine Übersicht über diese. Wird ein treffender Kontakt gefunden, erscheint direkt das Profil des Kontaktes.

8.2.4 Grosse Anzahl an Kontakten

Beobachtung

Besitzt man eine grosse Anzahl an Kontakten, geht die Übersicht schnell verloren, da jeweils alle Kontakte in einer Liste dargestellt wurden.

Resultierende Änderung

Ein Paging wurde eingebaut. Der Benutzer kann nun durch seine Kontakte blättern und erhält so immer eine übersichtliche Anzahl an Kontakten.

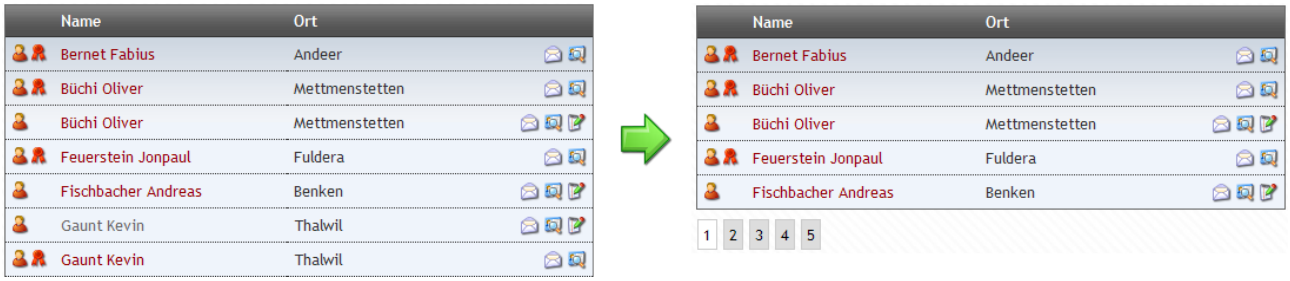

Abbildung 63: Usability Tests - Blättern in Kontakten

8.3 Browser Tests

Für eine Webapplikation ist die breite Unterstützung von verschiedenen Webbrowsern ein zentraler Punkt. Daher wurde die Seite von Beginn weg auf möglichst vielen Systemen und Konfigurationen getestet. Auch spezielle Konfigurationen haben einen grossen Einfluss auf die Webseite. Die Seite soll sowohl mit deaktiviertem Javascript als auch mit deaktivierten Cookies bedienbar bleiben.

Ein hilfreiches Werkzeug für die Kompatibilitätstests stellt die Webdeveloper Toolbar [17] für den Firefox dar. Diese ermöglicht, auf einfach Weise einzelne Browserfunktionen zu deaktivieren. Zudem stellt auch das Simulieren von anderen Bildschirmauflösungenkein Problem für das Tool dar.

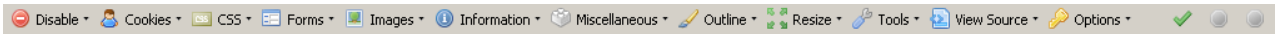

Abbildung 64: Die Webdeveloper Toolbar für den Mozilla Firefox

Um möglichst alle Browser zu unterstützen, wurde sowohl beim HTML als auch beim CSS darauf geachtet, die existierenden Standards des W3C [14] zu beachten. Die Seite wurde nach den CSS 2.1 und XHTML 1.0 Transitional Standards umgesetzt. Dadurch sollte sie auch für zukünftige Browsergenerationen gerüstet sein.

Abbildung 65: Webstandards

Das Testen auf unterschiedlichen Browserversionen ist kompliziert, da meist nur eine Browserversion gleichzeitig installiert sein kann. Für den Mozilla Firefox gibt es Versionen, welche ohne Installation lauffähig sind [15]. Glücklicherweise gibt es auch für den Microsoft Internet Explorer ein ähnliches Tool, welches aber eine Installation voraussetzt [16].

8.4 Integration Tests

Um Webkomponenten zu testen, bietet Visual Studio eine spezielle Art von Tests an. Webtests werden wahlweise direkt im Browser aufgezeichnet oder von Hand erstellt. Dabei wird ein Ablauf auf der Seite

definiert. Dieser Ablauf kann mit Validierungsfunktionen angereichert werden. So ist es möglich, das Vorhandensein eines bestimmten Textes oder eines Elements zu testen. Zudem kann überprüft werden, ob ein Request korrekt verarbeitet wird.

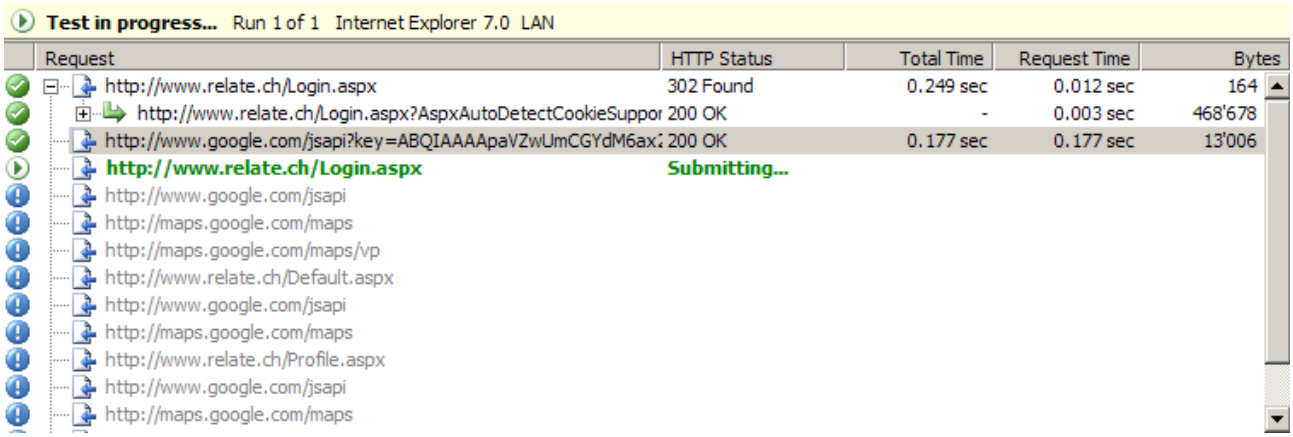

Abbildung 66: Laufender Web Test in Visual Studio 2008

9 Schlussfolgerung

9.1 Ergebnisse

Der Prototyp der Studienarbeit wurde erfolgreich weiterentwickelt, ist bereits unter www.relate.ch erreichbar und besitzt schon die ersten aktiven Nutzer.

Die Kontaktdaten werden bei der Erfassung automatisch mit dem Telefonbuch von tel.search.ch abgeglichen. Später sollen noch zusätzliche Telefonbücher angebunden werden. Die Softwarearchitektur ist bereits für diesen Fall ausgelegt.

Die Visualisierung der Kontakte wurde komplett überarbeitet. Neben den bekannten Karten von Google können die Kontakte nun mittels VirtualEarth auch in 3D betrachtet werden. Zusätzlich stehen mit der BirdsEye-Option hochauflösende Flugzeugaufnahmen von Ballungsgebieten zur Verfügung. Leider sind die Satelitenaufnahmen von ländlichen Gebieten qualitativ sehr schlecht. Der Benutzer hat aber jederzeit die Möglichkeit, zwischen den beiden Karten zu wechseln.

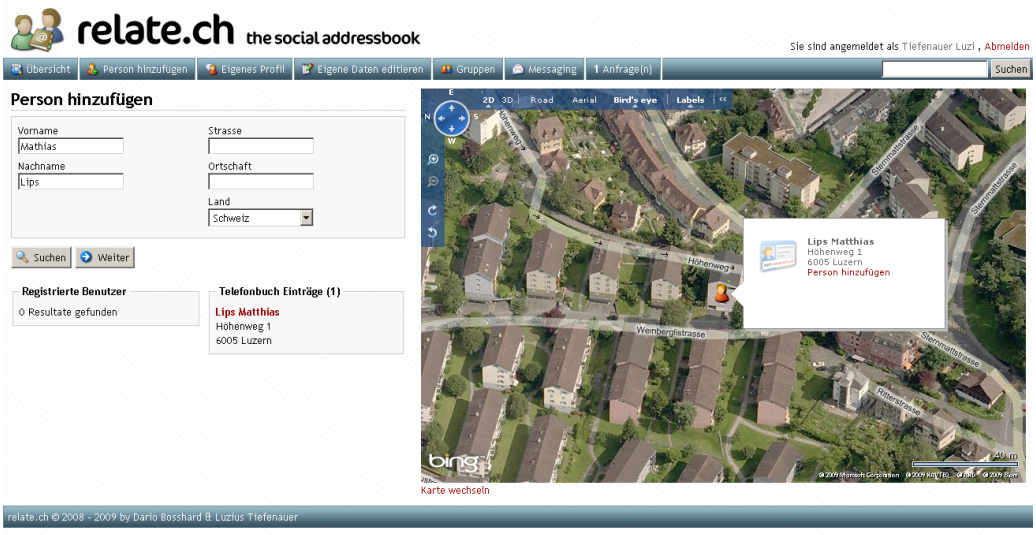

Abbildung 67: Webplattform www.relate.ch

Das neue Sicherheitssystem ermöglicht es, dass jeder Benutzer genau bestimmen kann, welche seiner Kontaktinformationen er wem anzeigen will. Dabei wurde aus Performance- und Verständlichkeitsgründen auf eine personenbezogene Definition der Zugriffsrechte verzichtet. Stattdessen werden die Bewertungen der Beziehungen verwendet, um die Rechte für den Zugriff zu definieren. Der Benutzer kann für jede Kontaktinformation bestimmen, wie gut eine geschäftliche oder eine private Beziehung sein muss, um die Information einsehen zu dürfen. So kann genau definiert werden, welche Kontaktdaten für geschäftliche Kontakte gedacht sind und welche für private.

Durch das neue flüssige Webdesign kann der Platz moderner Breitbild-Monitore besser genutzt werden. Wenn das Fenster vergrössert wird, nutzt die Karte jeweils den neuen Platz komplett aus. Beim Aufbau der Webseite wurde mehr auf Konsistenz geachtet, die Karte findet sich nun beispielsweise stets am selben Ort. Auch die verbesserte Suchfunktion steht nun auf jeder Seite zur Verfügung und macht mittels AJAX-Technologie bereits während dem Tippen Vorschläge für mögliche Treffer.

Mit dem AjaxControlToolKit für .NET konnten grosse Teile der Webseite mit Javascript optimiert werden. Meist muss nicht die ganze Seite neu geladen werden, sondern nur die neuen Teilbereiche. Dabei bleibt die Seite ohne Javascript weiterhin vollumfänglich bedienbar.

Neben dem Erfassen von Personen und der Definition der Beziehung zu diesen können nun auch Gruppen erstellt werden, um die Kommunikation innerhalb der realen Gruppe zu verbessern. Mithilfe der neuen Messagingfunktionen können beliebig viele Personen auf einfache Weise angeschrieben werden. Das Versenden von Emails und SMS kann dabei auch zeitversetzt stattfinden. Dies ermöglicht es beispielsweise, Erinnerungsnachrichten oder Geburtstagsglückwünsche bereits im Voraus zu planen. Leider konnte die geplante Sammelemailadresse für Gruppen noch nicht implementiert werden. Dies soll aber möglichst bald nach Abschluss dieser Arbeit geschehen.

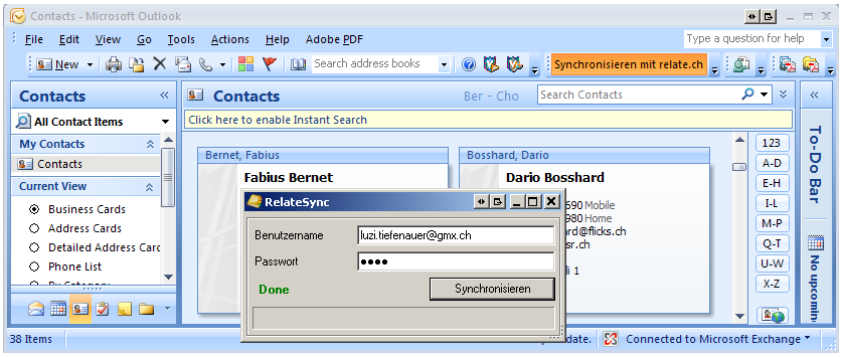

Abbildung 68: Synchronisations Add-In für Microsoft Outlook

Ein Add-In für Micosoft Outlook ermöglicht das Synchronisieren der online geführten Kontakte mit einem Knopfdruck. Neben diesem einfachen Synchronisationsprogramm können dank der Webservice-Schnittstelle beliebige andere Programme auf die Daten von Relate.ch zugreifen.

9.2 Ausblick

Beide Teammitglieder sind weiterhin vom Potenzial der Kontaktverwaltungsplattform überzeugt und motiviert, auch nach dieser Arbeit Zeit in dieses Projekt zu investieren. Dazu wurde bereits ein eigener SVN Server und ein Projektmanagementtool eingerichtet, da die verwendeten Programmen der HSR nicht mehr weiter benutzt werden können.

Wegen des engen Zeitrahmens konnten viele geplante Funktionen noch nicht implementiert werden. Trotzdem haben wir schon sehr viel positives Feedback erhalten und sind deshalb überzeugt, dass eine einfache, zentrale und vernetzte Kontaktverwaltung für viele Personen ein Bedürfnis darstellt. Im Kapitel 10 auf der nächsten Seite wurden Ideen für die Weiterentwicklung gesammelt und beschrieben.

Um eine langfristige Betreibung zu ermöglichen, soll nun auch die Wirschaftlichkeit geprüft werden. Bannerwerbung bringt heutzutage längst nicht mehr ausreichend Ertrag, um die Kosten für den Unterhalt und vor allem auch für die Weiterentwicklung zu decken. Für diese Analyse konnte erfreulicherweise ein Wirtschaftsstudent der Hochschule St.Gallen gewonnen werden, welcher seinerseits eine Bachelorarbeit zu diesem Thema erarbeiten wird.

Das Ziel dieser Arbeit war es, die Kontaktverwaltung grundlegend zu vereinfachen. Nun sind wir gespannt ob unsere hierfür entwickelte Webplattform www.relate.ch bei den Benutzern Anklang findet.

10 Weiterentwicklungen

Die entwickelte Plattform bietet viele Möglichkeiten zur Weiterentwicklung. Während dem Projekt wurden bereits viele Ideen gesammelt, die leider aus Zeitgründen nicht direkt implementiert werden konnten. Sie sollen nun hier aufgelistet und erläutert werden, damit sie zu einem späteren Zeitpunkt implementiert werden können.

10.1 Lokale Gruppierung von Kontakten

Neben den bereits implementierten Gruppen soll es dem Benutzer auch ermöglicht werden, seine Kontakte für sich selbst zu gruppieren. Hierfür sollen die bereits mehrfach angedachten lokalen Gruppierungen implementiert werden.

10.2 Hinzufügen von Kontaktdaten zu aktiven Personen

Sobald ein Kontakt aktiv am System teilnimmt, können seine Daten von den vorherigen Besitzern nicht mehr bearbeitet werden. Dies kann vor allem dann ein Nachteil sein, wenn sich eine Person einmalig anmeldet und danach die Plattform nicht weiter verwendet. Ein weiteres Problem ist die Möglichkeit, dass eine Person zwar die Mobilfunknummer einer Person kennt, diese aber aufgrund von dessen Sicherheitseinstellungen nicht einsehen darf.

Daher soll es ermöglicht werden, dass auch für aktive Personen neue Kontaktdaten erfasst werden können. Diese werden dann der jeweiligen Person zur Übernahme ins Profil vorgeschlagen.

10.3 Bidirektionale Synchronisation

Um Personen, die bereits eine gut gepflegte Kontaktdatenbank besitzen, den Einstieg zu erleichtern, soll es ermöglicht werden, dass ein Relate Account mit Daten aus einer anderen Kontaktverwaltung gefüttert wird. Das Outlook-Add-In würde dann nicht nur die neusten Kontakte von Relate.ch herunterladen, sondern könnte auch Relate.ch mit den Outlook Kontakten versorgen.

Hier stellt der Abgleich der Daten jedoch eine Herausforderung dar, müssen doch alle Kontakte jeweils wieder auf das selbe Kontaktobjekt abgebildet werden.

10.4 Kalenderfunktion

Auf Relate.ch soll auch ein Kalender eingebaut werden. Dies würde es ermöglichen, dass gemeinsame Termine immer direkt bei allen Beteiligten in der Agenda erscheinen. Mithilfe der Messaging Funktionen könnten die Teilnehmer automatisch an Sitzungen erinnert werden.

Eine weitere interessante Möglichkeit ist, automatisch freie Termine für einen Gruppenanlass zu finden. Hierfür müsste aber ein Grossteil der Gruppe die Kontaktfunktion benutzten.

10.5 Bannerwerbung

Damit die Seite bereits in der Startphase einen kleinen Ertrag abwirft, könnten an freien Stellen jeweils Werbebanner eingeblendet werden. Langfristig müssten aber weitere Modelle gefunden werden, wie mit der Plattform Geld verdient werden kann.

10.6 Automatische Gruppenerkennung

Neben einer Funktion, die einem möglicherweise bekannte Personen vorschlägt, soll mithilfe der vorhandenen Beziehungsinformationen versucht werden, automatisch Gruppierungen von Personen zu erkennen.

Wenn Personen einen realen Bezug, wie beispielsweise den Besuch der selben Schulklasse, zueinander haben, so ist meist auch die Verknüpfung unter diesen Personen relativ hoch. Mit einer Analyse dieser Daten soll versucht werden, diesen Personen automatisch das Erstellen einer Gruppe vorzuschlagen.

10.7 Zusammenführen zweier Personen

Es ist möglich, dass eine Person mehrmals von verschiedenen Benutzern erfasst wird. Wenn sich diese Person dann auch noch aktiv im System anmeldet, kann dies dazu führen, dass Benutzer diese Person mehrfach in ihrem Kontaktbuch haben, oder aber nur den passiven Kontakt, obwohl diese Person bereits aktiv am System teilnimmt.

Es soll daher möglich sein, dass man zwei Personen zusammenfügt. Dabei muss darauf geachtet werden, dass keine Informationen verloren gehen, die bestehenden Beziehungen auf die neue Person übernommen werden und vor allem auch, dass dieser Prozess nur von dazu berechtigten Benutzern durchgeführt wird.

10.8 Spitznamen einer Person

Eine Person besitzt abhängig von ihrem Umfeld verschiedene Spitznamen. Speziell auf mobilen Endgeräten werden Kontakte oft unter diesem Kurznamen abgelegt. Synchronisiert man nun seine Kontakte mit dem System, erscheinen diese unter ihrem normalen Namen. Es soll daher möglich sein, seinen Kontakten einen Spitznamen zuzuteilen. Somit kann bei einer Synchronisation mit einem Mobilgerät wahlweise der normale Name oder der Spitzname synchronisiert werden. Zudem können die bereits erfassten Spitznamen als Vorschläge angezeigt werden, wenn eine andere Person diesen Kontakt in seine Liste aufnimmt.

10.9 Adressbestimmung mittels Karte

Für den Fall, dass man die genaue Adresse einer Person nicht weiss, soll es möglich sein, auf der Karte die Wohnadresse zu bestimmen. Momentan ist es nicht möglich, eine Adresse zu erfassen, ohne die genaue Strasse zu kennen. Diese neue Variante würde es Personen ermöglichen, anhand der Karte das Haus eines Freundes erkennen und markieren zu können.

10.10 Ausblick

Neben den erwähnten Erweiterungen existieren noch viele Ideen, die hier nicht aufgelistet sind. Wir hoffen auf eine schnelle Umsetzung von vielen dieser Ideen in der Zukunft und sind nach wie vor überzeugt von den Möglichkeiten, die sich durch eine zentrale Kontaktverwaltung bieten.

Es lohnt sich sicherlich, in regelmässigen Abständen auf www.relate.ch vorbei zu schauen.

Teil III
Anhang

11 Persönliche Berichte

11.1 Dario Bosshard

11.1.1 Projektverlauf

Dass wir im Rahmen einer Bachelorarbeit weiter an diesem Projekt arbeiten konnten, legten wir bereits am Anfang der Studienarbeit fest. So konnten wir uns schon langfristig Ziele setzen und sie in dieser Arbeit angehen. Dabei mussten wir einige Aufgaben, die wir in der Studienarbeit bereits erledigt hatten, nochmals überarbeiten, um dem neuen Konzept zu entsprechen. Dabei nutzten wir die Erfahrungen, die wir während der Studienarbeit sammeln konnten. Der Sicherheitslayer, einer der wichtigsten und zentralsten Punkte, benötigte viel Denkarbeit. Es sollte keine Schnellschusslösung werden, die im effektiven Gebrauch nicht taugt. Eine Kombination aus Flexibilität und Zuverlässigkeit zu finden, war dabei nicht einfach. Auf Basis dieser Sicherheitsschicht konnten wir die übrigen Arbeiten schnell vorantreiben und die gesetzten Ziel beinahe erreichen.

Abbildung 69: Dario Bosshard

11.1.2 Arbeiten im Team

Durch die bereits dritte Arbeit, die wir zusammen bestritten, war der Einstieg und die Aufteilung der Arbeiten schnell und unkompliziert. Jeder arbeitete grundsätzlich für sich, traten Probleme auf, konnten diese zusammen aber immer gelöst werden. Dabei war es von Vorteil, dass wir nicht immer gleicher Meinung waren. So entstanden die besten Ideen meist in einer Diskussion. Nach Abgabe dieser Arbeit werden wir weiter an diesem Projekt arbeiten.

11.1.3 Arbeiten mit dem Betreuer

Das Arbeiten mit dem Betreuer war unkompliziert. Da wir die Arbeit selbst eingegeben haben, waren Besprechungen nicht so häufig wie bei anderen Teams. Sie beschränkten sich mehr auf das Ziel und das Format der Arbeit. Bei der Realisierung genossen wir viel Freiheit. Es ist für mich eine Bereicherung, selbst Erfahrungen machen zu können und nicht in ein Schema gedrückt zu werden. Auch wenn die Vorgehensweise nicht immer die beste sein mag, ist die Erfahrung, die man dadurch macht, umso wichtiger. Diese Erfahrung konnten wir dank Herrn Joller machen.

11.1.4 Fazit

Die weitere Vertiefung in das .NET Framework und der Ausbau der Plattform waren sehr lehrreich und spannend. Das Wissen, das wir in drei Jahren Studium an der HSR sammeln konnten, half uns dabei, die Zeile zu erreichen, die wir uns selbst gesteckt hatten. Eine eigene Idee einzubringen und diese von A bis Z selbst zu planen und durchzuführen, ist nicht immer einfach, da man viel Eigenverantwortung mitbringen muss. Das Potenzial, das wir in dieser Arbeit sehen, verleitete uns aber dazu, immer 100% und mehr zu geben.

11.2 Luzius Tiefenauer

11.2.1 Projektverlauf

Obwohl wir an demselben Projekt wie bereits in der Studienarbeit weiterarbeiteten, mussten wir in ein neues Labor umziehen. Dies bedeutete leider auch, dass wir unsere Arbeitsplätze erneut einrichten mussten. Diese unnötig verschwendete Zeit ärgerte mich sehr. Nachdem wir uns durch die Installationen gekämpft hatten, waren wir aber bereit, mit Hochdruck an der Weiterentwicklung unserer Idee zu arbeiten.

Da wir unsere Aufgabenstellung selbst definieren konnten, war es möglich, uns zuerst umfassend mit der Code-Qualität des Prototypen auseinanderzusetzten. Der Versuch, eine bestehende Problemlösung zu verbessern oder auch einmal komplett anders anzugehen, bereitete mir sehr viel Spass.

Ein weiterer Höhepunkt war für mich die Integration von neuen Funktionen. Es war unheimlich motivierend, plötzlich in 3D durch meine Kontakte fliegen zu können oder meine Freunde in den Ferien kurz nach der Landung mit zeitgesteuerten SMS Nachrichten zu begrüssen.

Abbildung 70: Luzius Tiefenauer

11.2.2 Arbeiten im Team

Wie bereits in vorherigen Arbeiten war ich äusserst zufrieden mit der Zusammenstellung unseres Zweierteams. Dario hat wie ich einen gewissen Ehrgeiz, wenn es um die Umsetzung von kleinen Finessen in einem Programm geht. Und so überraschten wir uns gegenseitig immer wieder mal mit neuen Verbesserungen, die wir in Eigenregie über Nacht eingebaut hatten.

Unsere jeweiligen Stärken und Schwächen ergänzen sich sehr gut und so freue ich mich bereits jetzt auf die gemeinsame Weiterarbeit mit Dario an diesem Projekt.

11.2.3 Arbeiten mit dem Betreuer

Das Arbeiten mit Herrn Joller war stets unkompliziert und angenehm. Die Kommunikation beschränkte sich meist auf ein Minimum. Diese Freiheit ermöglichte es uns, die Zeit in den Bereichen zu investieren, die uns am sinnvollsten erschienen. Durch unsere hohe Selbstmotivation für dieses Projekt waren wir immer sehr engagiert, ohne dass unser Betreuer dies forcieren musste.

11.2.4 Fazit

Ich persönlich bin mit dem Verlauf und dem Resultat der Arbeit sehr zufrieden. Wir haben immer wieder positives Feedback sowie Anregungen für Verbesserungen von den ersten Benutzern der Plattform erhalten. Ich freue mich auf die Weiterentwicklung der Plattform und bin sehr gespannt, was langfristig daraus wird.

12 Zeitplan

12.1 Projektplan

Nachdem die Ziele dieser Arbeit definiert waren, wurden die anstehenden Arbeiten in Pakete gegliedert. Zu diesen Paketen gehörten folgende Aufgaben:

- Installation
- Analyse
- Refactoring des bestehenden Prototyps
- Programming
- Webseite (GUI)
- Testing
- Dokumentation

Diese Pakete wurden wiederum in kleinere Pakete aufgeteilt und priorisiert. Danach wurde ein Gantt-Diagramm erstellt, um einen Projektplan zu erstellen. Mit der Definition der Abhängigkeiten zwischen diesen einzelnen Paketen konnte dann ein kritischer Pfad berechnet werden und die anstehenden Arbeiten ideal auf die zwei Projektmitglieder verteilt werden.

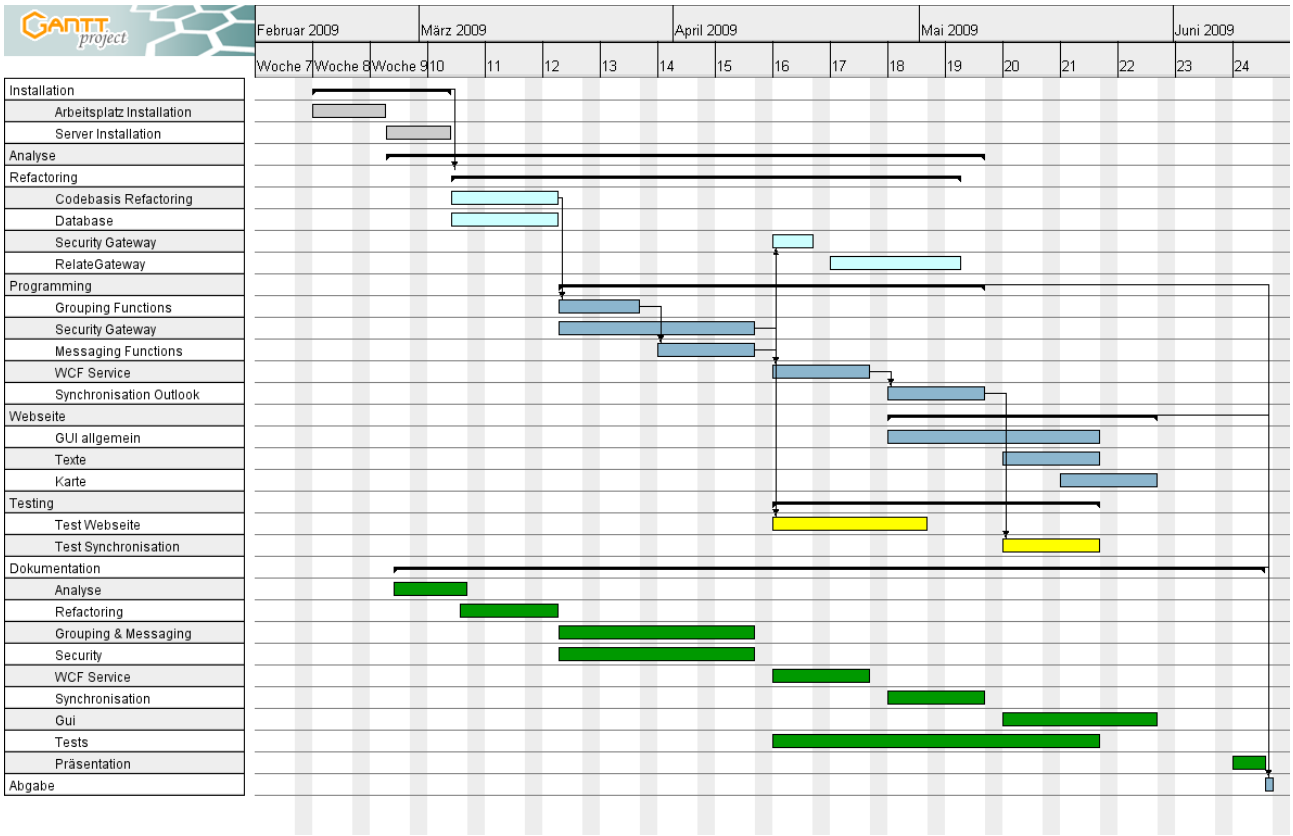

Abbildung 71: Gantt-Diagramm des geplanten Projektplanes

12.2 Meilensteine

Nachdem ein Projektplan erstellt war, wurden Meilensteine definiert. Mit diesen konnte der Fortschritt der Arbeit laufend kontrolliert werden.

12.2.1 Meilenstein 1 – Analyse Codebasis & Datenbank (17.03.2009)

Die Analyse des bestehenden Prototypen ist abgeschlossen. Zusätzlich ist genau definiert, was daran geändert und welche neuen Funktionen eingebaut werden sollen.

Artefakte

- Analyse (siehe Kapitel 4)
- Case-Cases (siehe Kapitel 4.3)

12.2.2 Meilenstein 2 - Security (10.04.2009)

Analyse, Design und Implementation der Sicherheit sind abgeschlossen und dokumentiert.

Artefakte

- Analyse Sicherheit (siehe Kapitel 4.5)
- Design Sicherheit (siehe Kapitel 5.3.4, 5.4.6 und 5.5)
- Implementation Sicherheit (siehe Kapitel 7.6)

12.2.3 Meilenstein 3 - Messaging (10.04.2009)

Analyse, Design und Implementation für das Versenden und Empfangen von Nachrichten sind abgeschlossen und dokumentiert.

Artefakte

- Analyse Messaging (siehe Kapitel 4.6)
- Design Messaging (siehe Kapitel 5.3.2 und 5.4.4)
- Implementation Sicherheit (siehe Kapitel 7.6)

12.2.4 Meilenstein 3 - Analyse & Programmierung WCF Service Schnittstelle (06.05.2009)

Die WCF Schnittstelle ist fertig analysiert und implementiert. Zusätzlich demonstriert ein Outlook Add-In die Funktionalität der Schnittstelle.

Artefakte

- Analyse WCF Schnittstelle (siehe Kapitel 4.7)
- Implementation WCF Schnittstelle (siehe Kapitel 7.11)
- Implementation Outlook Add-In (siehe Kapitel 7.12)

12.2.5 Meilenstein 4 - Umsetzung GUI (29.05.2009)

Das User Interface ist an die neuen Funktionen angepasst und ausgebaut.

Artefakte

- Design User Interface (siehe Kapitel 5.7)
- Refactoring User Interface (siehe Kapitel 6.3)

12.2.6 Meilenstein 5 - Abgabe Bachelorarbeit (11.06.2009)

Abgabe aller Dokumente.

12.3 Gesamtaufwand

Das Projekt wurde in mehrere Arbeitspakete gegliedert. Dabei wurde eine grobe Aufteilung der geforderten 720 Arbeitsstunden auf diese Pakete gemacht. Erfahrungsgemäss wurde dem Dokumentations- und Implementationsteil am meisten Zeit zugeteilt. Die Implementierung benötigte, wie schon bei der Studienarbeit, mehr Zeit als erwartet. Einige Bereich wie beispielsweise der komplexe Sicherheitsteil benötigten mehr Zeit als geplant.

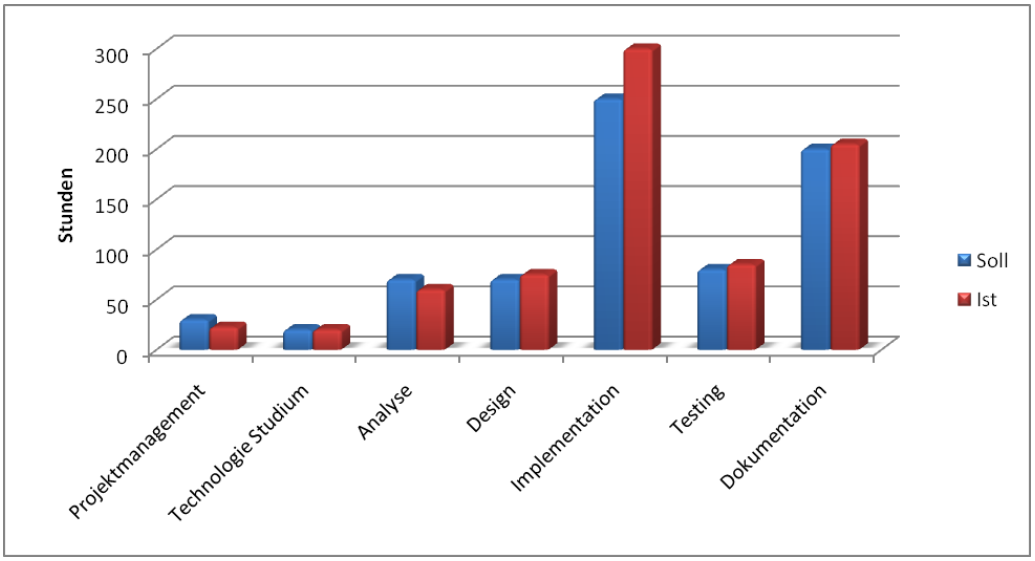

Abbildung 72: Vergleich der geplanten und effektiv geleisteten Arbeitsstunden pro Arbeitspaket

12.4 Arbeitsaufteilung

Wie die Abbildung 73 aufzeigt, wurde für die Implementation am meisten Zeit benötigt. Die Analyse und Designphasen waren relativ kurz, da ein Grossteil der Analyse bereits in der Studienarbeit gemacht werden konnte und diese nur noch entsprechend auf die neuen Anforderungen erweitert werden musste.

Daneben hat vor allem auch die sorgfältige Dokumentation der Arbeit viel Zeit benötigt.

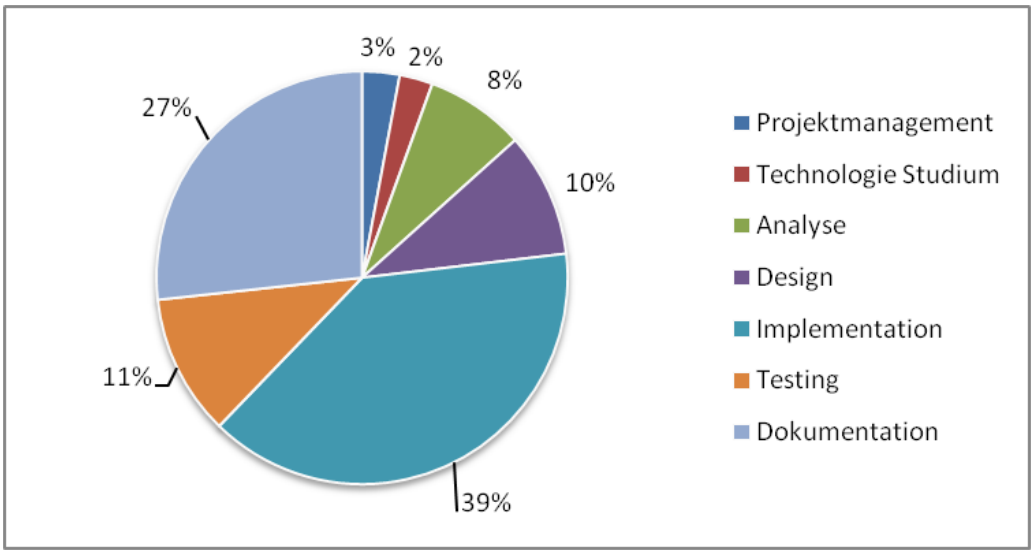

Abbildung 73: Verteilung der Stunden auf die Arbeitspakete

12.5 Zeitaufwand pro Person und Woche

Bei der aufgewendeten Zeit pro Woche erkennt man starke Schwankungen. Dies hängt damit zusammen, dass wir neben dieser Arbeit noch Vorlesungen besuchten und dadurch nicht in allen Wochen gleich viel Zeit zur Verfügung stand. Ausserdem war ein Teammitglied in den Frühlingsferien und musste so fast eine ganze Arbeitswoche nachholen, während das andere Teammitglied neben dem Studium noch einen Teilzeitjob hat, dessen Stundenaufwand auch je nach Woche unterschiedlich ausfällt.

Glücklicherweise waren die letzten beiden Wochen aber unterrichtsfrei und konnten dadurch intensiv für den Abschluss dieser Arbeit genutzt werden.

Abbildung 74: Zeitaufwand in Stunden pro Woche und Person

13 Glossar

Begriff	Erklärung
SOAP	SOAP (ursprünglich für Simple Object Access Protocol) ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht werden können (*Quelle: Wikipedia.org*)
POCO	POCO ist eine Abkürzung für Plain Old CLR Object, also ein „ganz normales“ Objekt in einer .NET Sprache. (*Quelle: Wikipedia.org*)
RBAC	Role Based Access Control (RBAC; deutsch: Rollenbasierte Zugriffskontrolle) ist in Mehrbenutzersystemen oder Rechnernetzen ein Verfahren zur Zugriffssteuerung und -kontrolle auf Dateien oder Dienste. (*Quelle: Wikipedia.org*)
Add-In	Mit Add in bezeichnet man meist eine Sammlung von zusätzlichen Funktionen oder Optionen, die in das Programm integriert werden. (*Quelle: Google Define*)
API	*engl. Application Programming Interface de. Schnittstelle zur Anwendungsprogrammierung.* Eine Programmierschnittstelle wird von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. *(Quelle: Wikipedia)*
Instant Messaging	Instant Messaging (kurz IM) (englisch für „sofortige Nachrichtenübermittlung“) ist eine Kommunikationsmethode, bei der sich zwei oder mehr Teilnehmer per Textnachrichten unterhalten. Dabei geschieht die Übertragung im Push-Verfahren, so dass die Nachrichten unmittelbar beim Empfänger ankommen.*(Quelle: Wikipedia)*
Social Networks	Social Networks sind Netzgemeinschaften, welche auf sozialen Strukturen zwischen den einzelnen Benutzern basieren.
AJAX	Ajax ist ein Apronym für die Wortfolge „Asynchronous JavaScript And XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Server und dem Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen. *(Quelle: Wikipedia)*
Lazy Loading	Lazy loading ist ein Design Pattern, welches eine Verzögerung der Initialisierung von Objekten, bis zu deren ersten Verwendung beschreibt. Dies kann zur Steigerung der Effizienz eines Programms führen. Das Gegenstück zu Lazy Loading ist Eager Loading. *(Quelle: Wikipedia)*
CLR	CLR steht für Common Language Runtime. Ein Sprachkonzept von .NET, welches das Schreiben von Code in verschiedenen Sprachen ermöglicht.
Object	Die passive Entität in einem Sicherheitsmodell. Zu schützende Daten. Im gegebenen Fall meist Kontaktinformationen einer Person
Subject	Die aktive Entität in einem Sicherheitsmodell. Auslöser eines Datenzugriffes, typischerweise der Benutzer.

Literatur

[1] HSR Studienarbeit Relate.ch - 2008 - Dario Bosshard & Luzius Tiefenauer

[2] Security Patterns - Integration Security and Systems Engineering

[3] Enterprise Solution Patterns Using Microsoft .NET

[4] Jesse Liberty, Donald Xie (2008) - Programmieren mit C# 3.0 - O'Reilly - ISBN 978-3-89721-859-8

[5] Joseph Albahari, Ben Albahari (2007) - C# in a Nutshell, Third Edition - O'Reilly - ISBN 978-0-596-52757-0

[6] Juval Löwy (2009) - Programming WCF Services, 2nd Edition - O'Reilly - ISBN 978-0-596-52130-1

[7] Florian Huonder (2008) - Analysis of Access Control Policies - Student Research Project

[8] Feinstein (1994) - Role-Based Access Control: A Multi-Dimensional View. (1994)

[9] Explained: Forms Authentication in ASP.NET 2.0 - November 2005 - http://msdn.microsoft.com/en-us/library/aa480476.aspx

[10] Asp.Net Custom Error Pages - 2004 - Milan Negovan - http://aspnetresources.com/articles/CustomErrorPages.aspx

[11] ISO 3166 - Standard für die Kodierung von geographischen Einheiten - Wikipedia - http://de.wikipedia.org/wiki/ISO_3166

[12] Plain Old Java Object - Wikipedia - http://de.wikipedia.org/wiki/Plain_Old_Java_Object

[13] Net Objectives - Pattern Repository - http://www.netobjectives.com/PatternRepository/

[14] World Wide Web Consortium - http://www.w3.org/

[15] Mozilla Firefox, Portable Edition - http://portableapps.com/apps/internet/firefox_portable

[16] Install multiple versions of IE on your PC - TredoSoft - http://tredosoft.com/Multiple_IE

[17] Web Developer extension - http://chrispederick.com/work/web-developer/

[18] Problem with multiple cascade paths - MSDN Artikel - http://support.microsoft.com/?scid=kb%3Ben-us%3B321843&x=12&y=8

[19] Julia Lerman (2009) - Programming Entity Framework - O'Reilly - ISBN 978-0-596-52028-1

[20] Der Lebenszyklus einer ASP.NET 2.0 Seite - http://www.visual-eye.de/visualeye/index.php?option=com_content&view=article&id=55:asp-page-lifecycle&catid=49:aspnet&Itemid=64

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum:	Rapperswil, 12. Juni 2009	
Name, Unterschrift	Dario Bosshard	Luzius Tiefenauer