HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ins institute for networked solutions

Guidelines.NET

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Herbstsemester 2012

Autor(en):	Lukas Kretschmar, Lukas Wegmann, Quentin Willimann
Betreuer:	Prof. Hansjörg Huser, Clemens Meier
Projektpartner:	Kantonsspital St. Gallen, St. Gallen

AUFGABENSTELLUNG

GUIDELINES.NET - WEBAPP MIT ASP.NET MVC & HTML5

Aufgabenstellung für Lukas Kretschmar, Lukas Wegmann und Quentin Willimann

EINFÜHRUNG

Guidelines.ch wurde in den letzten Jahren am Kantonsspital St.Gallen entwickelt. Es handelt sich um eine Webapplikation, welche in erster Linie als Nachschlagewerk und Verwaltungswerkzeug für Behandlungsrichtilinien dient. In Zukunft soll Guidelines aber auch als öffentliche Informationsplattform für Gesundheit und intelligenter Ratgeber für Ärzte weiter wachsen. Guidelines.ch wird bereits im Kantonsspital St.Gallen verwendet und soll im Spitalverbund Ostschweiz eingeführt werden. Es sind auch andere Spitalregionen wie z.B. Basel auf das Projekt aufmerksam geworden.

Die bestehende Lösung basiert auf den Technologien PHP 5.2/Zend Framework und MySql 5 und ist im Moment nicht stark skalierbar. Die verwendeten Technologien erschweren ausserdem eine Integration von Microsoft Office Dokumenten oder eine Zusammenarbeit mit anderen Microsoft Technologien wie Microsoft Sharepoint. Das System sollte später auch mit mobilen Clients und Tablets verwendet werden können.

Mit .NET und Windows Azure bietet Microsoft Technologien, welche den Ansprüchen einiges besser gerecht werden. Das Ziel dieser Arbeit ist ein Architekturprototyp in asp.Net MVC mit einem REST-Webservice. Eine Anforderungsanalyse und ein erster Prototyp wurde bereits während eines SE2 Projektes erstellt. Die Software stellt hohe Anforderungen an das User Interface, und mit einem Webbrowser ohne Plugin bedienbar sein. Ausserdem soll die neue Lösung in den angesprochenen Punkten ausbaubar, stark skalierbar und „Cloud ready“ sein.

AUFGABENSTELLUNG

- Analyse der Anforderungen
  - Analyse des bestehenden Systems
  - Erfassen der Anforderungen
  - Dokumentation des Validierungsprozesses
  - Resultat: Use Case-Spezifikation, Domain-Modell
- Umsetzung der Businesslogik
  - Umsetzung des Rechtesystems
  - Umsetzung der Historisierung
- Definition und Implementation der Serverseite
  - asp.Net MVC 4
  - REST-Service mit asp.NET WebAPI
  - MS-SQL-Express
  - Windows Azure ready
- Design und Implementation der Client Seite
  - UI-Design
  - HTML5
  - Ajax mit Json
  - evtl. MVVM in Javascript mit KnockoutJS

- Systemtests
- „Remoteentwicklung“ (Rapperswil-Shanghai)

Es müssen nicht alle oben erwähnten Punkte bis zum Schluss der Arbeit umgesetzt werden.

RESULTATE

- Analyse-und Design-Dokumente
- Ausführbare Software mit Dokumentation

AUFTRAGGEBER

Kantonsspital St.Gallen
Ansprechpartner: Christian Kahlert (Anforderungen),

INS
Ansprechpartner: Clemens Meier (Anforderungen, Architektur)

PROJEKTTEAM

Lukas Kretschmar (lkretsch@hsr.ch)
Quentin Willimann (qwillima@hsr.ch)
Lukas Wegmann (l1wegman@hsr.ch)

BETREUUNG HSR

Hansjörg Huser (hhuser@hsr.ch), Tel: 055 222 49 12 (HSR Raum 6.010)
Clemens Meier (cmeier@hsr.ch), Tel: 055 222 47 38 (HSR Raum 6.010)

PROJEKTABWICKLUNG

TERMINE:

- Beginn der Arbeit: 20.9.2012
- Abgabetermin Kurzfassung/Mgmt-Summary zum Review: 17.12.2012
- Abgabetermin : 21.12.2012, 17.00
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

ARBEITSAUFWAND

Für die erfolgreich abgeschlossene Arbeit werden 8 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 240 Stunden pro Student.

HINWEISE FÜR DIE GLIEDERUNG UND ABWICKLUNG DES PROJEKTES:

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:

- Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen)

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Letzter Meilenstein: Systemtest abgeschlossen
  Termin: ca. eine Woche vor Abgabe
- Entwickeln Sie Ihre SW in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Setzen Sie konsequent Unit-Tests ein! Verwalten Sie ihre Software und Dokumente auf einem TFS. Stellen Sie sicher, dass der/die Betreuer jederzeit Zugriff auf das Repository haben und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.
- Achten Sie auf die Einhaltung guter Programmier- und Designprinzipien
  - DRY, high cohesion, loose coupling, etc.
  - Clean Code!
- Halten Sie sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.

PROJEKTADMINISTRATION

- Führen Sie ein individuelles Projekttagebuch aus dem ersichtlich wird, welche Arbeiten Sie durchgeführt haben (inkl. Zeitaufwand). Diese Angaben sollten u.a. eine individuelle Beurteilung ermöglichen.
- Dokumentieren Sie Ihre Arbeiten laufend. Legen Sie Ihre Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch in einem Projektordner ab. Dieser Projektordner sollte jederzeit einsehbar sein (z.B svn-Server oder File-Share).

INHALT DER DOKUMENTATION

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

- Dokumente gemäss Vorgabe: https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
- Aufgabenstellung
- Technischer Bericht
- Projektdokumentation
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.

FORTSCHRITTSBESPRECHUNG

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt via Skype.
Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
Termin: einmal wöchentlich, Raum 6.010 + Skype (Wird noch festgelegt)

TRAKTANDEN

Was wurde erreicht, was ist geplant, welche Probleme stehen an

Review von Code/Dokumentation (Abgabe jeweils einen Tag vor dem Meeting)

Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.
Sie erstellen zu jeder Besprechung ein Kurzprotokoll, welches Sie spätestens 2 Tage nach der Sitzung per E-Mail an den Betreuer senden.

Rapperswil, 20.09.2012
Hansjörg Huser
Clemens Meier

EIGENSTÄNDIGKEITSERKLÄRUNG

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, 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.
- das wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Ort, Datum: Rapperswil, 20.12.2012

Kretschmar, Lukas

Wegmann, Lukas

Ort, Datum: Shanghai (上海), 13.12.2012

Willimann, Quentin

ABSTRACT

Guidelines.ch ist eine Plattform der Infektiologie des Kantonsspitals St. Gallen, welche das Wissen der Ärzte und Abteilung verwaltet und anderen in Form von einheitlich strukturierten Dokumenten zugänglich machen kann. Im Rahmen der Studienarbeit wurde die Problemdomain, wofür es bereits eine in-house PHP-Lösung gibt, als RESTful Webservice mit den aktuellsten Microsoft-Technologien im Backendbereich (ASP.NET WebAPI, EntityFramework) und gängigsten Web-Technologien (ASP.NET MVC 4, HTML 5, KnockoutJS) fürs Frontend umgesetzt. Das Ziel der Arbeit war im Rahmen einer Technologiestudie zu zeigen welche Technologien in Frage kämen, wie weit es möglich ist eine Lösung mit diesen verschiedenen Technologien bereitzustellen und wie weit diese miteinander harmonieren.

Das Hauptaugenmerk wurde dabei auf eine flexible Architektur, einfache Erweiterung, Qualitätssicherung in Form von Tests und die zwei wichtigsten Use Cases (Suche, Einträge bearbeiten) gelegt. Als Grundlage für die Umsetzung dieser Use Cases mussten zuvor noch weitere Aspekte bezüglich der Kommunikation und Sicherheit umgesetzt werden (HMAC-Authentifizierung, ACL-basierte Zugriffsverwaltung).

Als Resultat der Arbeit entstand eine mögliche Lösung wie man die Plattform neu gestalten kann. Und wie eine Architektur aussehen könnte, die den Anforderungen an Performanz, Flexibilität und Erweiterbarkeit gerecht wird.

MANAGEMENT SUMMARY

AUSGANGSLAGE

Guidelines.NET ist eine Parallelentwicklung zu www.guidelines.ch, welches im Kantonsspital St. Gallen entwickelt wird. Dabei handelt es sich um eine Plattform, welche dem Spitalpersonal, aber auch Aussenstehenden, erlauben soll, Zugriff auf Informationen über Krankheiten und Richtlinien des Spitals zu gewähren und zu verwalten. Dabei soll die Aktualität und Richtigkeit der Daten gewährleistet sein, des Weiteren ist eine schnelle und benutzerfreundliche Suche erwünscht. Als Basis dieser Entwicklung galt immer die Version des Kantonsspitals.

VORGEHEN, TECHNOLOGIEN

Guidelines.NET ist im Rahmen einer Studienarbeit an der HSR Hochschule für Technik Rapperswil entstanden. Dabei war das Ziel, gewisse Funktionen von www.guidelines.ch in eine Microsoft-Umgebung zu portieren. Mit Technologien wie ASP.NET MVC, MSSQL, C#, HTML 5 und KnockoutJS sollte eine robuste Kommunikation zwischen den Komponenten gewährleistet werden. Beginnend mit anfänglichen Recherchen, gefolgt von der Analyse des bestehenden Systems, der Erfassung der Anforderungen und der gleichzeitigen Entwicklung auf allen Ebenen der Software ist ein Produkt entstanden, welches den Benutzer bei seinen Aufgaben unterstützt, einfach zu verstehen und grafisch ansprechend ist.

Zusätzlich zur Aufgabenstellung des Kantonsspitals konnte während des Projekts die einmalige Gelegenheit genutzt werden, sich mit der Thematik der Remoteentwicklung zu beschäftigen. Da das Team aus Rapperswil, Schweiz, aber auch Shanghai, China, an der Studienarbeit entwickeln konnte, mussten Probleme gehandhabt werden, welche nicht alltäglich bei anderen Teams und Arbeiten waren.

ERGEBNISSE

Dabei wurden folgende Ziele erreicht:

- Ein Rechtesystem, welches den Zugriff auf kritische Dokumente regelt
- Eine Historisierung, welche die Änderungen eines Dokumentes aufzeigt
- REST-Service, welcher von anderen Programmen leicht angesprochen werden kann
- Eine Softwarearchitektur, welche leicht erweiterbar ist
- Eine schnelle Suche, die selbst bei 26‘000 Einträgen ein gesuchtes Dokument innerhalb einer Sekunde findet (sofern nicht zu viele Einträge gefunden werden)
- Eine Autokorrekturfunktion bei Suchbegriffen
- Eine Suche, die leicht konfigurierbar ist
- Eine Software, die gut mit Tests abgedeckt wurde
- Ein UserInterface, das einfach zu verstehen ist und den Benutzer zum Mitmachen animiert

AUSBLICK

Dadurch, dass diese Arbeit im nächsten Semester nicht weitergeführt wird, liegt es auf Seiten der HSR vorerst auf Eis. Sollte das Kantonsspital St. Gallen jedoch Interesse an einer Weiterführung haben, wäre dies sicherlich möglich in einer späteren Arbeit auszubauen und am Kantonsspital einzuführen. Eine Erweiterung der jetzigen Version von Guidelines.NET besteht sicher im Bereich der Suche, wo eine Implementation einer dynamischen Zusammenfassung dem Benutzer noch hilfreich wäre. Zusätzlich müssen noch weitere Funktionen eingebaut werden, wie die Benutzerverwaltung und Dateienverwaltung, welche aber zum Teil bereits von der Idee her in den bestehenden Dokumenten beschrieben sind oder ansatzweise implementiert wurden.

INHALTSVERZEICHNIS

A TECHNISCHER BERICHT

1 EINLEITUNG

1.1 BESTEHENDES SYSTEM

Aus der Aufgabenstellung:

> *Guidelines.ch wurde in den letzten Jahren am Kantonsspital St.Gallen entwickelt. Es handelt sich um eine Webapplikation, welche in erster Linie als Nachschlagewerk und Verwaltungswerkzeug für Behandlungsrichtilinien dient. In Zukunft soll Guidelines aber auch als öffentliche Informationsplatform für Gesundheit und intelligenter Ratgeber für Ärzte weiter wachsen. Guidelines.ch wird bereits im Kantonsspital St.Gallen verwendet und soll im Spitalverbund Ostschweiz eingeführt werden. Es sind auch andere Spitalregionen wie z.B. Basel auf das Projekt aufmerksam geworden.*
>
> *Die bestehende Lösung basiert auf den Technologien PHP 5.2/Zend Framework und MySql 5 und ist im Moment nicht stark skalierbar. Die verwendeten Technologien erschweren ausserdem eine Integration von Microsoft Office Dokumenten oder eine Zusammenarbeit mit anderen Microsoft Technologien wie Microsoft Sharepoint. Das System sollte später auch mit mobilen Clients und Tablets verwendet werden können.*

1.2 AUFGABENSTELLUNG

Aus der Aufgabenstellung:

> *Mit .NET und Windows Azure bietet Microsoft Technologien, welche den Ansprüchen einiges besser gerecht werden. Das Ziel dieser Arbeit ist ein Architekturprototyp in asp.Net MVC mit einem REST-Webservice. Eine Anforderungsanalyse und ein erster Prototyp wurde bereits während eines SE2 Projektes erstellt. Die Software stellt hohe Anforderungen an das User Interface, und mit einem Webbrowser ohne Plugin bedienbar sein. Ausserdem soll die neue Lösung in den angesprochenen Punkten ausbaubar, stark skalierbar und „Cloud ready“ sein.*

1.3 ZIELSETZUNG

Das Kernziel der Studienarbeit ist das Ausarbeiten der Architektur die die Anforderungen bezüglich Skalierbarkeit und Erweiterbarkeit erfüllen kann. Darauf aufbauend sollen die Grundfunktionalitäten wie Guidelines lesen, suchen und bearbeiten auf allen Layern umgesetzt werden.

1.4 SYSTEMÜBERSICHT

Das in der Studienarbeit umgesetzte System lässt sich grob in zwei Komponente unterteilen. Zum einen ein Webservice, der die Business Logik der Guidelines Applikation abbildet und die Persistierung der Dokumente auf der Datenbank übernimmt und zum anderen eine Web Applikation, die das Arbeiten mit der Guidelines API im Browser ermöglicht.

Die Guidelines API wurde als technologieunabhängige REST Schnittstelle umgesetzt und ist somit offen für unterschiedlichste Client Plattformen. Teile der API könnten auch für Drittanbieter zugänglich gemacht werden.

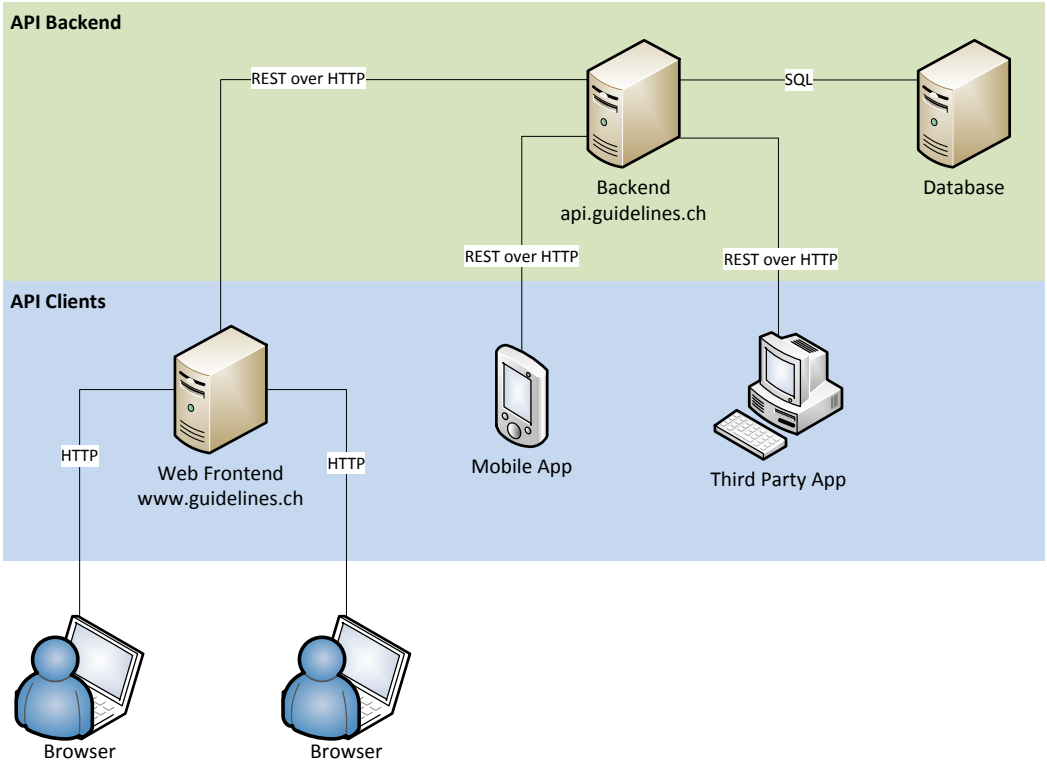


Abbildung 1 Guidelines Systemübersicht

2 BACKEND

Das Backend stellt die Guidelines API zur verfügung und ist für die Persistierung und die Abwicklung der Business Logik zuständig.

2.1 SCHNITTSTELLENARCHITEKTUR

Die Guidelines API ist die zentrale Schnittstelle zwischen Backend und Frontend und soll diese zwei Komponenten soweit als möglich voneinander entkoppeln. Neben der logischen Trennung wird auch die technologische Unabhängigkeit angestrebt. So soll es möglich sein, die Technologien des Backend oder eines API Clients auszuwechseln ohne das sich dies auf die übrigen Systeme auswirkt.

Aus diesem Grund wurde das Backend als RESTful Webservice umgesetzt. REST ist im Gegensatz zu SOAP nicht eine Erweiterung von HTTP, sondern setzt die Möglichkeiten von HTTP vollumfänglich um. Dadurch werden sehr geringe technologische Anforderungen an die API Clients gestellt, wobei Implementationen von HTTP wahrscheinlich in fast jeder Programmiersprache bereits vorhanden sind. Bei der Entwicklung der Guidelines API sollen die folgenden Prinzipien von REST besonders beachtet werden:

- **Ressourcenorientiert**: Entitäten des Domain Models werden als Ressourcen abgebildet, die über eine URL eindeutig identifizierbar sind. So kann auf die Guideline mit der Kennzeichnung „malaria-notfallbehandlung“ der Arbeitsgruppe „infektiologie“ kann über „GET /workgroups/infektiologie/guidelines/malaria-notfallbehandlung“ zugegriffen werden. Die Ressourcen können über die Standard HTTP Methoden wie GET, POST, PUT oder DELETE manipuliert werden, sofern die Berechtigungen dazu vorhanden sind.
- **Zustandslose Kommunikation**: Ein Request muss immer alle Informationen enthalten, die zu dessen Abarbeitung notwendig sind. Insbesondere müssen auch die Authentifizierungs-informationen bei jedem Request mitgesendet werden. Die zustandslose Kommunikation ist eine wichtige Voraussetzung für die Skalierung des Systems und ermöglicht das Cachen von Requests und das Betreiben mehrerer Instanzen des Webservers.
- **Konsequenter Einsatz der HTTP Methoden**: Die Mehtoden die auf eine Ressource angewendet werden können, sollen konsequent umgesetzt sein.
    - **GET** – Gibt die durch die URL angeforderte Ressource zurück, führt zu keinen Nebeneffekten
    - **POST** – Fügt eine neue Ressource zu einer Collection hinzu
    - **PUT** – Ersetzt eine bestehende Ressource durch eine neue Version
    - **DELETE** – Löscht eine Ressource

Für die Serialisierung der Daten wird in erster Linie das JSON Format eingesetzt.

2.1.1 AUTHENTIFIZIERUNG

Um die Identität des Clients zu überprüfen und die Requests zu verifizieren, wird eine Authentifizierung über das HMAC Verfahren angewendet. Dies bedeutet, das bei jedem Request die Authentifizierungsinformationen mitgesendet werden.

Jeder API Client muss über einen öffentlichen und einen privaten Access Token verfügen. **Die Authentifizierung auf der API findet also nicht über Benutzername und Passwort statt.** Das Entkoppeln von Authentifizierung und Benutzer bietet folgende Vorteile:

- Es können Applikationen authentifiziert werden, die auf die REST Schnittstelle zugreifen dürfen. Z.B. eine Third-Party Anwendung, die alle Guidelines eines bestimmten Spitals lesen darf
- Die Authentifizierung basiert auf stärkeren Schlüsseln als das typische Passwort eines Benutzers. Z.B. ein zufällig generierter 64 Byte Schlüssel
- Das Passwort des Benutzers muss nicht bei jedem Request mitübertragen werden (auch nicht in stark verschlüsselter Form)

Damit das Web Frontend trotzdem in der Rolle eines Benutzers Anfragen an die API senden kann, muss nach erfolgtem Login auf der Website ein Access Token angefordert werden:

1. Das Web Frontend verfügt über einen Access Token, welcher die Anwendung authentifiziert. Alle Anfragen eines nicht eingelogten Benutzers werden mit diesem Token verschlüsselt.
2. Ein Benutzer logt sich auf der Webseite ein.
    a. Das Web Frontend fordert mithilfe von Benutzername und Passwort bei der API einen neuen Access Token an (POST access_token)
    b. Der Access Token wird der Session des authentifizierten Benutzers zugeordnet
    c. Alle weiteren Requests von dieser Session werden nun mit diesem Access Token signiert

Access Tokens können auch über eine beschränkte Gültigkeitsdauer verfügen, so dass sie z.B. nach 24 Stunden wieder verfallen.

2.2 TECHNOLOGIEN

Der Einsatz der .NET Plattform ist durch die Aufgabenstellung vorgegeben.

2.2.1 SERVICE

Durch den Entscheid für eine REST-Schnittstelle, konnten die möglichen Varianten mit welchen Technologien diese umgesetzt werden kann, auf deren zwei eingeschränkt werden. Zum einen würde WCF in Frage kommen, welches seit Version 3.5 (aktuell 4.5) auch REST unterstützt, und ASP.NET Web API. Nachfolgend gehen wir auf den Prozess ein, wieso ASP.NET Web API zum Einsatz kam.

2.2.1.1 WCF

WCF (Windows Communication Foundation) ist die Kommunikationstechnologie, welche im .NET-Framework standardmässig verwendet wird, wenn man Interprozess- oder Netzwerkkommunikation benötigt. Es ist eine generische Schnittstelle, die auf dem WSDL-Standard aufsetzt und als Transportprotokolle auf TCP, UDP, HTTP, NamedPipes und MSMQ setzt. Die Unterstützung für weitere Protokolle kann auch selbst zugefügt werden.

Seit der Version 3.5 unterstützt WCF auch Stateless-Services, was REST-Services entspricht. Da diese aber nicht unbedingt in die WCF-Architektur passen, wurde die bestehende Strukturen in WCF aufgebrochen oder umgangen. Für den Entwickler hat das aber keine grossen Änderungen mit sich gebracht, da das Hosting und Deklarieren der Schnittstelle immer noch gleich ist. Einzig neue Attribtue sind hinzugekommen. So muss anstelle des [OperationContract] ein [WebGet] (für HTTP GET) oder [WebInvoke] (für alle Aufrufe ausgenommen GET) verwendet werden.

Für das Binding werden zwei Möglichkeiten bereitgestellt. So kann das Binding einfach, wie es generell der Fall ist bei WCF, über die App.config respektive Web.config Datei erledigt werden. Das benötigte Binding für einen Stateless-Service wäre „WebHttpBinding“. Es besteht aber auch die Möglichkeit, wenn man Self-Hosting machen will, anstelle des ServiceHost() den WebServiceHost() zu verwenden. Dieser setzt das entsprechende Binding selbst.

Vorteile

WCF ist eine bekannte und weit verbreitete Technologie, mit .NET die Kommunikation zwischen mehreren Systemen aufzubauen. WCF ist darauf ausgelegt, die Schnittstelle allgemein zu definieren und dann die entsprechenden Protokolle und Medien, die für die Kommunikation verwendet werden sollen, per Konfiguration zuzufügen. Dies macht WCF zu einer sehr flexiblen Technologie, bei der Änderungen an Betriebsumgebungen meist nur eine Auswirkung auf die Konfiguration haben.

Nachteile

Da WCF eine Abstraktion für viele verschiedene Arten von Kommunikationsprotokollen und -technologien ist, leiden die technologiespezifischen Besonderheiten. Dies hat zur Folge, dass WCF nur Funktionalität zur Verfügung stellt, die von allen Technologien unterstütz werden können.

Zusammenfassung

WCF ist sehr flexibel und kann auf allen gängigen Kommunikationsprotokollen aufsetzen. Durch diese Flexibilität kann WCF aber nicht alles bis ins Detail unterstützen. Mit WCF ist man unabhängig, da man sehr systemnah die Kommunikation aufsetzten kann.

2.2.1.2 ASP.NET WEB API

ASP.NET Web API ist die Technologie, die aus den Bemühungen des WCF-Teams entstanden ist, die bestehende WCF Implementation RESTful zu gestallten.

Vorteile

ASP.NET Web API ist die Technologie, welche von Microsoft vorangetrieben wird, wenn man eine Schnittstelle im Web zur Verfügung stellen will. Web API baut auf ASP.NET MVC auf. Und durch die Vorgabe, dass als Kommunikationsprotokoll lediglich HTTP verwendet wird, wurde die Umsetzung im Code sehr detailreich gemacht. So gibt es bereits Wrapper für Request und Response.

Nachteile

Web API unterstützt ausschliesslich nur die Kommunikation über das HTTP-Protokoll.

Zusammenfassung

Web API kann nicht alles in Sachen Kommunikation, da nur HTTP als Protokoll unterstützt wird. Durch diese Limitation ist es aber perfekt an die Möglichkeiten des HTTP-Protokolls angepasst und für REST-Services ausgelegt.

2.2.1.3 ENTSCHLUSS

Zu Beginn des Projekts hatten wir uns für eine Umsetzung mit WCF entschieden, da hier bereits einiges an Know-How vorhanden war.

Bei genauerer Analyse des Problems wurde aber klar, dass ASP.NET Web API besser auf die Aufgabenstellung zugeschnitten war. Denn durch den Entscheid, dass nur eine REST-Schnittstelle angeboten werden soll, muss nur das HTTP-Protokoll unterstützt werden. Dies führte dazu, dass nach den Tests mit WCF, auf die Web API umgestiegen wurde. Web API nimmt viele Kleinigkeiten ab, bietet aber immer noch die Konfigurierbarkeit, welche benötigt wird.

2.2.2 DEPENDENCY INJECTION

Dependency Injection (auch Inversion of Control (IOC) genannt) ist ein Vorgehen wie Abhängigkeiten zwischen Komponenten aufgelöst und Klassen im Code instanziiert werden können. Verwendet man Dependency Injection übernimmt ein Framework die Instanziierung konkreter Klassen. Der Programmierer muss lediglich angeben, an welchen Stellen welche Instanzen verlangt werden. Besonders in Hinblick auf die testgetriebene Entwicklung, ist auch das einfachere In jizieren von Mock Objekten ein grosser Vorteil.

Vorteile

Keine starke Kopplung zwischen Klassendefinition und –deklaration. Dies führt dazu, dass man lediglich gegen ein Interface programmieren kann, ohne Wissen zu müssen, woher beim Ausführen die Instanz herkommen wird und was diese sonst noch alles kann. Die einzelnen Klassen und Komponenten können dabei aber sehr flexibel gehandhabt werden und sind leicht austauschbar.

Nachteile

Durch das verwenden eines Dependency Injection Frameworks kann ein geringer Overhead entstehen. Zudem wird je nach eingesetztem Framework ein Teil der Applikationslogik in die Konfigurationsdateien ausgelagert.

2.2.2.1 ENTSCHLUSS

Dependency Injection ist ein Entwurfsmuster, bei dem die Vorteile klar zu überwiegen scheinen und dessen Einsatz in ASP.NET durch die DependencyResolver Schnittstelle sehr einfach ist. Teile der Applikation können so auch leicht ausgetauscht werden, sobald die Anforderungen sich ändern.

Als Dependency Injection Framework wurde “Unity” verwendet.

2.3 DATEN

In der SA kamen zwei verschiedene Typen von Datenobjekten zum Einsatz, zum einen wurden POCO's und zum anderen DTO's verwendet. Nachfolgend werden die beiden Konzepte der Datenobjekte genauer erläutert.

2.3.1 POCO

POCO's (Plain Old CLR Object's) sind Klassen, die einen Datenbankeintrag repräsentieren, dabei aber nur CLR-Typen oder andere POCO's verwenden.

Vorteile

POCO's haben keine speziellen Framework-Abhängigkeiten und können auf den verschiedenen Layern, in denen sie Verwendung finden, ohne spezielle Einbindung weiterer Bibliotheken verwendet werden. Zudem können diese direkt durch das Entity Framework generiert und verwendet werden.

Nachteile

Wird ein Changetracking verlangt, muss dieses selbst implementiert werden.

2.3.2 DTO

DTO's (Data Transpfer Object's) sind Klassen, die dazu verwendet werden um Daten zwischen zwei oder mehreren Schnittstellen zu transportieren. Sie enthalten grundsätzlich keine oder nur sehr wenig Logik.

Vorteile

DTO's können sehr schlank gehalten werden und verarbeiten Daten nicht. Sie sind zudem nicht an eine bestimmte Technologie gebunden und können von vielen Schnittstellen verwendet werden, sofern diese die DTO's interpretieren können.

Nachteile

DTO's führen oft zu mehr Implementationsaufwand falls das Abbilden von Entities zu DTO's notwendig wird.

2.3.3 ENTSCHLUSS

Für den Transport von Daten innerhalb der Applikation wurde zunächst nur auf POCO's gesetzt. Im Verlaufe des Projekts entwickelten sich die Anforderungen des Frontends und Backends immer weiter auseinander, was dazu geführt hat, dass für die Kommunikation mit der API DTO's eingeführt wurden.

Die Trennung von POCO's und DTO's macht insofern auch Sinn, da in den POCO's .NET-spezifische Erweiterungen verwendet werden konnten. Die DTO's mussten hingegen so einfach gehalten werden, dass diese nicht nur in unserem Frontend verwendet werden konnten, sondern direkt auch im JavaScript. Die direkte Einbindung in Browser hat auch verlangt, dass weitere Informationen dem

DTO mitgegeben werden musste wie zum Beispiel der Pfad oder die http-Methoden, die auf die Ressource angewandt werden können. Diese Informationen werden hingegen bei den POCO's nicht benötigt. Durch die Aufteilung von POCO's und DTO's konnten so auch Informationen voneinander getrennt werden, die nicht an beiden Orten benötigt wurden.

Der einzige Nachteil, den diese Lösung mit sich zog, war die Notwendigkeit der Konvertierung zwischen POCO's und DTO's. Dadurch, dass an beiden Orten nicht oder in einer anderen Form dieselben Daten vorhanden sind, kann bei der Konvertierung von einem DTO zu einem POCO das ursprüngliche POCO nicht komplett wiederhergestellt werden. Dies führt dazu, dass man im Business-Layer aufpassen musste, das man nur mit Daten arbeitet, die auch tatsächlich im POCO vorhanden sind. Als einfachste Lösung des Problems lädt man das ursprüngliche POCO aus der Datenbank nach und fügt die fehlenden Daten hinzu oder arbeitet mit beiden POCO's.

2.4 CONCURRENCY

Die Problematik der Nebenläufigkeit (Concurrency) wurde in der SA nicht ausführlich Behandelt werden, da nur in wenigen Fällen bestehende Datenbankeinträge bearbeitet werden müssen. Nimmt der Umfang von Guidelines zu, wird man zwangsläufig nicht darum herum kommen, die Thematik der Concurrency besser zu analysieren und einzubinden.

2.4.1 REVISIONEN SPEICHERN

Der einzige problematische Fall, der in der SA abgedeckt wurde, war das Setzten der Versionsnummer beim Speichern einer Revision.

Wird eine Guideline verändert, wird beim Speicher eine neue Revision angelegt. Diese sollte die nächstmögliche Versionsnummer erhalten, sodass eine fortlaufende Nummerierung entsteht. Beim Speichern könnte es aber vorkommen, dass zwei Revisionen gleichzeitig abgespeichert werden und die gleiche Versionsnummer erhalten könnten oder, dass in der Zwischenzeit eine weitere Version gespeichert wurde.

Beide Probleme verlangen nach derselben Lösung, nämlich, dass vor dem Speicher aus den vorhandenen Revisionen in der Datenbank die nächst mögliche Versionsnummer berechnet, anschliessend bei der neuen Revision gesetzt und gespeichert wird. Hierbei darf aber während des Setzens keine neue Revision erstellt werden, denn ansonsten könnten wieder oben genannte Probleme auftreten.

Um diese Problematik zu unterbinden, kämen drei mögliche Lösungsansätze in Frage, die nachfolgend erläutert werden. Grundsätzlich muss man sich festlegen, wo man die Problematik lösen will, denn sie kann sowohl auf der Datenbank als auch im Programm selbst behandelt werden.

2.4.1.1 TRANSAKTIONEN (IM CODE)

Eine Transaktion ist eine Möglichkeit in der Datenbank mehrere Abfragen so zu bündeln, dass diese nicht unterbrochen werden können durch Andere.

Vorteile

Die Problematik der Nebenläufigkeit wird von der Datenbank behandelt und muss nicht im Code nachprogrammiert werden. Zudem werden Transaktionen durch das Entity Framework unterstützt und können so einfach aus dem Code heraus erstellt werden.

Nachteile

Bei falscher Wahl des Isolationslevels könnten Abfragen blockiert werden, die eigentlich ausgeführt werden dürften. Zudem könnte ein Flaschenhals auf der Datenbank entstehen, wenn längere Transaktionen andere Abfragen oder Änderungen unterbinden.

2.4.1.2 STOREDPROCEDURE (IN DATENBANK)

Alternativ könnte man die gesamte Problematik auf die Datenbank auslagern. Hierfür müsste man das zu speichernde Revision einer StoredProcedure übergeben, die dann die letzten Schritte mit dem Berechnen der Versionsnummer, Setzten und Speichern übernehmen würde.

Vorteile

Letzt möglicher Ort, wo die Problematik behandelt werden kann. Zudem ist man frei, wo und wie die Transaktion zum Einsatz kommt.

Nachteile

Bearbeitende Logik auf Datenbankebene. Die zu speichernde Revision müsste in einzelnen Parametern übergeben werden. Die Logik, die eine Revision erstellt, wird aufgeteilt, da der grösste Teil im Code vorhanden ist und ein kleiner Teil direkt auf der Datenbank erledigt wird.

2.4.1.3 READWRITELOCK (IM CODE)

Die weitere alternative um die Concurrency-Problematik zu behandeln, wären ReadWriteLocks. Diese Locks können direkt im Code auf Objekte kapseln und den Zugriff darauf behandeln. So lässt ein ReadWriteLock mehrere Threads zu die nur Daten lesen. Möchte ein Thread das Objekt bearbeiten, lässt ein ReadWriteLock keine weiteren Zugriffe mehr auf das Objekt zu.

Vorteile

Direkt im Code auf Objekte anwendbar.

Nachteile

Verlangt Daten, die von mehreren Threads verwendet werden. Dies kann den Einsatz von mehreren Instanzen des Webservices stark erschweren und den Einsatz in der „Cloud“ verunmöglichen.

2.4.1.4 ENTSCHLUSS

Für das Bestimmen der Versionsnummern wurde auf Transaktionen gesetzt, da diese einfach in .NET aus dem Code heraus zu erstellen sind. Zudem hätte die Anwendung von ReadWriteLocks unsere Bestrebungen bezüglich Skalierbarkeit unterlaufen. Die Variante mit der StoredProcedure wurde

ignoriert, da dadurch die Business Logik zu sehr verstreut wird und dies der einzige Fall wäre, wo auf eine StoredProcedure gesetzt würde um Daten zu bearbeiten.

Hinter dem Web-Service ist erst wieder die Datenbank eine Ressource, die von allen Threads gemeinsam genutzt wird. Deshalb kamen nur Transkationen in Frage. Dies führte aber zu zwei kleinen Unschönheiten, die man sich noch genauer anschauen müsste. Da die Anzahl der Isolationslevel begrenzt ist und dennoch einer gewählt werden musste, der das Hinzufügen von neuen Einträgen unterbinden kann, wurde die stärkste Isolation gewählt. Das heisst aber, dass während der Transaktionen überhaupt keine neuen Einträge erstellt werden können, obwohl diese nicht unbedingt einen Einfluss auf die Revision hätten.

2.5 SECURITY & ACCESS-MANAGEMENT

2.5.1 BENUTZERAUTHENTIFIZIERUNG

Da die Anforderung gegeben ist, dass Benutzer sich am System authentifizieren müssen um erweiterte Rechte zu erhalten, wurde die Thematik im Rahmen der SA auch behandelt. In den Nachfolgenden Abschnitten werden die Möglichkeiten aufgezeigt, die evaluiert wurden und welches Verfahren in der SA gewählt wurde.

2.5.1.1 WIF

WIF (Windows Identity Foundation) ist die von Microsoft zur Verfügung gestellte Lösung für die Authentifizierung an Webservern. Es wäre also naheliegend, sich an dieser Technologie zu orientieren.

Vorteile

WIF ist bereits im .NET-Framework enthalten und findet breite Akzeptanz. Zudem ist es sehr flexibel und kann auch mit anderen Authentisierungsschnittstellen gekoppelt werden.

Nachteile

Je nach Einbindungsgrad kann WIF auch einen Overhead mit sich bringen, da nur ein Subset der Technologie benötig wird.

2.5.1.2 EIGENE IMPLEMENTATION

Eine eigene Implementation der Authentifizierung macht dann Sinn, wenn man beabsichtigt, diese so schlank wie möglich umzusetzen. Im Rahmen der SA hat sich diese Möglichkeit angeboten, da der Schwerpunkt nicht auf der Authentisierung und Security gelegt wurde sondern auf die Funktionalität.

Als Rahmen würden aber die bereits vorhandenen Interfaces IPrincipal und IIdentity verwendet werden um dennoch die Kompatibilität zu den .NET Authentifizierungsmechanismen erreichen zu können. Diese würde dazu führen, dass für die Benutzerdaten von Guidelines ein Adapter geschrieben wird, der die zusätzliche Funktionalität enthält oder entsprechend behandeln kann. Da es sich beim Service um eine REST-Schnittstelle handelt und der Benutzer dadurch pro Request authentifiziert wird sowie einfach im Kontext zu finden sein muss, könnte man bei der

Implementation die Benutzerdaten direkt dem Request zuordnen und findet diese dementsprechend immer am selben Ort.

Vorteile

Komplette Kontrolle über die Implementation. Kann domainspezifische Optimierungen enthalten und man muss nur das zur Verfügung stellen, was benötigt wird.

Nachteile

Je nach Implementation nicht oder nur schwer wart- und erweiterbar. Zudem wird nicht auf bereits vorhandene Technologien gesetzt, sondern alles parallel dazu neu implementiert.

2.5.1.3 ENTSCHLUSS

Da die Authentication in der SA nicht das Hauptthema war, sondern lediglich rudimentär und einfach benötigt wurde, wurde eine eigene Implementation der Authentication bevorzug. Für das Gelingen eines Service-Zugriffs wird lediglich der Benutzer benötigt. Da dieser in der Guidelines Datenbank abgelegt ist und nicht aus einem anderen System gelesen wird, kam eine eigene Implementation gelegen, da diese sich sehr schlank realisieren lässt. Als weiteren Grund für die eigene Implementation sprach die Tatsache, dass WIF nicht leichte Materie ist. Hätte man im Rahmen der SA bereits WIF verwenden wollen, hätte man sich von Beginn an mit der Thematik und Technologie auseinandersetzen müssen.

Falls aber in Zukunft doch nicht WIF zum Einsatz käme, kann durch die Verwendung vom IPrincipal- und IIdentity-Interface einfach gewechselt werden. Diese sind die Basis von WIF respektive der Authentication in .NET. Somit kann die aktuelle Implementation einfach an andere Technologien gebunden werden.

2.6 EXCEPTIONHANDLING

Für das Behandeln von Fehler im Backend werden in Guidelines unterschiedliche Strategien verfolgt. Zum einen werden, entsprechend richtige HttpResponseExceptions in den jeweiligen Controller-Methoden geworfen, wo bekannt ist, dass diese auftreten können. Zusätzlich wurde aber noch auf eine globale Fehlerbehandlung gesetzt.

Die Ursache hierfür ist das Standardverhalten der WebAPI. Tritt das Szenario auf, dass innerhalb eines Controller-Aufrufs ein Fehler geworfen wird, wird dieser zwar dem Aufrufer zurückgeschickt, aber verpackt in einer „200 OK“-Response. Die Möglichkeit, dass der Statuscode im Response entsprechend gesetzt wird, wird also standardmässig nicht genutzt.

2.6.1 LOKALE BEHANDLUNG

Jeder Request wird auf eine entsprechende Methode in einem Controller gemappt, die den Request abhandeln kann. Tritt beim Abhandeln des Request ein Fehler auf, wird dieser in der Controller-Methode abgefangen und entsprechend seines Ursprungs in den korrekten HTTP-Fehler verpackt und dem Aufrufer zurückgeschickt.

Vorteile

Es ist transparent, was für Fehler auftreten und wie sie umgewandelt werden.

Nachteile

In der Controller-Methode muss bekannt sein, welche Fehler innerhalb des Backends alles auftreten können.

2.6.2 GLOBALE BEHANDLUNG

Da nicht alle Request erst bei der Behandlung der eigentlichen Anfrage Fehler provozieren können, sondern auch direkt beim Empfangen und Überprüfen (speziell in Bezug auf die Authentifizierung), wurde eine globale Fehlerbehandlung eingerichtet. Aus diesem Grund wurde hier auch das Logging eingebaut um sicherlich jeden Fehler im System in einem Log zu erfassen. Die lokale Fehlerbehandlung ist aber natürlich weiterhin möglich.

Wird nun bereits ein Fehler in der lokalen Fehlerbehandlung richtig umgewandelt, wird dieser anschliessend in der globalen ignoriert. Die globale Fehlerbehandlung kommt erst zum Zug, wenn ein Fehler noch in der systeminternen Repräsentation vorhanden ist.

Die Umwandlung in der globalen Fehlerbehandlung gliedert sich wiederum ein zwei Teile. Wird ein Fehler abgefangen, für den ein speziell geeigneter HTTP-Statuscode besteht (zum Beispiel „401 Unauthorized“, „403 Forbidden“ oder „404 Not Found“), wird dieser auch gewählt. Kann hingegen kein dafür vorgesehener Statuscode genommen werden, wird der Fehler generell in einen „500 Internal Server Error“ umgewandelt.

Vorteile

Jeder Fehler, der im System auftreten kann, kann hier abgefangen und entsprechend behandelt werden. Somit können auch keine systeminternen Fehler, die nicht lokal abgefangen wurde, nach aussen treten.

Nachteile

Es müssen alle Fehler bekannt sein, die mit einem speziellen HTTP-Statuscode versehen werden können. Ansonsten werden sie dem Aufrufer durch einen 500er verschleiert.

2.6.3 ENTSCHLUSS

Der Entscheid zu einer zwei-stufigen Fehlerbehandlung kam daher, dass man sicherlich jeden Fehler im Backend bereits behandeln kann und die Möglichkeit der HTTP-Statuscodes komplett ausnutzt. Die lokale Fehlerbehandlung macht insofern Sinn, dass sie dem Entwickler die Möglichkeit gibt, die globale zu übersteuern, in dem man den Fehler selbst behandelt. Sie führt auch zu mehr Transparenz, da mögliche Fehler entsprechend den Anforderungen behandelt werden können.

2.7 TESTING

Da Guidelines als grössere Plattform geplant ist, wurde ein grosser Teil auch dem Testing gewidmet. Allem voran wurde das Backend grössten Teils nach dem Test-Driven Prinzip entwickelt um zu garantieren, dass zukünftige Erweiterungen nicht bestehende Funktonalität zerstören.

2.7.1 TEST-DRIVEN DEVELOPMENT

Test-Driven Development ist ein Prinzip der Softwareentwicklung, beidem die Test die vorantreibende Kraft bei der Implementation sind. So wird zu einer gewünschten Funktionalität immer zuerst ein Test geschrieben, der diese auf die Richtigkeit prüft. Anschliessend folgt die Implementation, die den Test erfolgreich durchlaufen muss. Ist dies geschehen, kann der Test um ein weiteres Kriterium erweitert werden und man fährt weiter mit der Implementation. Dadurch, dass zu jedem Feature diverse Tests bestehen, kann bei einer Änderung im Code und/oder Refactoring schnell geprüft werden, ob die bestehende Funktionalität noch garantiert werden kann.

Vorteile

Es werden während der Implementation bereits Tests geschrieben. Die Software erhält ein Design, welches gut getestet werden kann. Fehler und Bugs nach Änderungen können schneller lokalisiert werden.

Nachteile

Mehraufwand, da zu jeder Funktionalität Tests geschrieben werden. Der Fortschritt ist, zumindest zum Beginn des Projektes, nicht so schnell wie wenn keine Tests geschrieben würden.

2.7.2 ENTSCHLUSS

Da Guidelines potentiell eine sehr grosse Plattform werden kann, ist die Notwendigkeit einer guten Testing-Strategie unumgänglich. Hier hat sich Test-Driven Ansatz gut angeboten. Dadurch, dass dieser auch bereits an anderen Orten im Studium thematisiert wurde, waren grundlegende Kenntnisse bei allen Projektmitgliedern vorhanden.

Durch Test-Driven Development kann garantiert werden, dass bei Änderungen und/oder Erweiterungen keine negativen Effekte auf bestehe Funktionalität auftreten. Es ist zwar keine Garantie gegeben, dass alles richtig funktioniert, aber zumindest so wie im Test erwartet.

2.8 SUCHE

Eine der wichtigsten Funktionalitäten von Guidelines ist die Suche, dabei müssen mehrere Aspekte beachtet werden. Die Suche muss sowohl performant, aber auch benutzerfreundlich sein. Für die Implementation der Suche gab es verschiedene Möglichkeiten, zum einen der Einsatz einer eigenständigen Search Engine und zum anderen das Umsetzen einer Eigenimplementation auf Basis der Suchfunktionalitäten des SQL Servers.

2.8.1 LUCENE

Lucene ist ein opensource Suchframework von Apache. Es bietet eine Rangliste für die Suchresultate, Feldsuche, etc. an. Des Weiteren werden verschiedene Querytypen unterstützt, wie:

- **Phrase queries**: Suche nach genau diesem Suchterm in ein einem Dokument, Teiltreffer werden nicht berücksichtigt.
- **Wildcard queries**: erlaubt das suchen mit Hilfe von *, wobei * als irgenwas interpretiert wird.
- **Proximity queries**: Suche nach Wörtern, die eine maximale Distanz zwischen sich haben. Je näher sich die Wörter befinden, desto höher ist das Ranking.
- **Range queries**: Suche nach Begriffen, die zwischen zwei Parametern liegen.

Vorteile

Grossteil der Implementation wird vom Framework übernommen. Viele Suchalgorithmen sind bereits vorhanden und können genutzt werden. Es kann schneller mehr Funktionalität angeboten werden. Mit Lucene.net wäre eine Integrität mit der von uns gewählten Infrastruktur geboten. Performantes suchen wird gewährleistet.

Nachteile

Anpassungen des Rankingssystems und der Suchalgorithmen an die Problemdomäne können unter Umständen viel Zeit in Anspruch nehmen. Zudem ist durch die komplexität von Apache Lucene viel Einarbeitungszeit erforderlich.

2.8.2 ENTSCHLUSS

Dadurch dass mit der Volltextsuche, welche vom MSSQL Server 2012 Express unterstützt wird, bereits in einem Vorprojekt Erfahrung gesammelt wurde, war dieser Weg für die SA am besten geeignet. Mit der Volltextsuche sind schon gewisse Querytypen unterstützt und weitere können genutzt werden, sobald ein Thesaurus Dokument hinterlegt wird.

Mit dieser Entscheidung, war die ganze Gestaltung: Datenbankwahl, Datenmodell etc. unter eigener Kontrolle und somit flexibel erweiterbar.

3 API CLIENT FÜR .NET

Im Rahmen der Studienarbeit wurde auch ein wiederverwendbares Assembly entwickelt, mit dem .NET Applikationen den Zugriff auf die Guidelines API ermöglicht wird. Das Assembly ist unter dem Namespace GuiDot.Client.API zu finden und bietet folgende Funktionalitäten:

- Bereitstellen der API Methoden in .NET (z.B. GetWorkgroups())
- Authentifizierung
- Konvertieren von HTTP Status Codes zu Exceptions im Fehlerfall
- Weiterleiten von ASP.NET HTTP Requests zur API

3.1 RESTSHARP

Der API Client für .NET baut auf dem RestSharp REST Client[1] auf. RestSharp stellt dazu Funktionalitäten wie JSON Serialisierung und Deserialisierung, das Erzeugen von HTTP Requests und das Interpretieren der HTTP Responses zur Verfügung.

[1] http://restsharp.org/

4 WEB FRONTEND

Das Web Frontend stellt ein Benutzerinterface für die Arbeit mit der Guidelines API zur Verfügung, welches mit jedem modernen Browser bedienbar ist.

Somit ist das Web Frontend ein API Client der Eingaben des Benutzers an das Backend weiterleitet und die Antwort als HTML Applikation darstellt.

4.1 ANFORDERUNGEN

Da das Web Frontend bloss zwischen Backend und Webapplikation vermittelt, werden an die dort verwendeten Technologien relativ wenig Anforderungen gestellt:

- **Geringe Reaktionszeiten auf Requests**
  Insbesondere sollen AJAX Requests schnell beantwortet werden können (< 200ms), da bei zu langen Antwortszeiten das Benutzerinterface ansonsten als träge empfunden wird.
- **Einfaches HTML Templating**
  Das ausgeben von HTML ist gewissermassen die Hauptaufgabe des Web Frontends und sollte dementsprechend schnell und einfach handhabbar sein.
- **Session Management**
  Da das Backend komplett Zustandslos arbeitet, muss bei jeder Anfrage an die API alle Authentifizierungsinformationen mitgesendet werden. Damit sich der Benutzer aber trotzdem nur einmal authentifizieren muss, soll das Frontend die Zugriffsinformationen für den Benutzer (API Access Token) an der Usersession zuweisen können.
- **Integrierter HTTP Client**
  Die Kommunikation zwischen Backend und Frontend wird über REST Requests stattfinden. Das Frontend nimmt deshalb die Rolle eines HTTP Servers wie auch eines HTTP Clients ein.

4.2 EVALUATION WEBFRAMEWORK

4.2.1 ASP.NET MVC

ASP.NET MVC ist ein sehr umfangreiches Web Framework von Microsoft und baut auf .NET auf. Das Framework verfolgt konsequent das MVC Pattern (Model – View – Controller), welches auch von vielen anderen erfolgreichen Web Frameworks wie Spring MVC, Ruby On Rails oder Zend (PHP) eingesetzt wird.

Vorteile

Da das Backend mit .NET Technologien entwickelt wird, bietet der Einsatz eines Web Frameworks das auf den selben Technologien aufbaut einige Vorteile.

So können einige für das Backend entwickelte Komponenten im Frontend wiederverwendet werden. Insbesondere Klassen, die Daten zwischen API und Clients transportieren (Data Transfer Objects), werden mit Sicherheit in Frontend und Backend zu verfügung stehen müssen.

Auch das Deployment der Software wird vereinfacht, da Frontend und Backend auf der selben Serverinfrastruktur verteilt werden können (Microsoft IIS).

Nachteile

Da ASP.NET MVC für die Entwicklung von Full Stack Applikationen ausgelegt ist, werden viele Komponenten des Frameworks für das Web Frontend gar nicht benötigt. Das Frontend muss zum Beispiel keine Daten persistieren können. Deshalb besteht die Gefahr, das die hohe Komplexität eines so umfangreichen Frameworks die Entwicklung unnötig verlangsamen kann.

4.2.2 NODE.JS

Node.js ist eine Plattform, die es ermöglicht auch serverseitig JavaScript einzusetzen. Als Laufzeitumgebung wird die, unter anderem in Google Chrome eingesetzte, V8 JavaScript Engine verwendet. Basierend auf dieser Plattform gibt es schon einige Web Frameworks, die meist auch das MVC Pattern anwenden.

Das weiterverbreitetste und wahrscheinlich auch ausgereifteste dieser Frameworks ist „Express“.

Vorteile

JavaScript auch serverseitig einzusetzen bietet in erster Linie den Vorteil, das Server und Client mit der selben Sprache entwickelt und gewisse Komponenten so einfach wiederverwendet werden können.

Eine weitere Stärke von Node.js ist die konsequent asynchrone Verarbeitung von Requests und die stark optimierte Laufzeitumgebung (V8), was zu sehr geringen Reaktionszeiten auf Anfragen führt. Desweitern ist durch die asynchrone Architektur auch sichergestellt, das Funktionen immer atomar ausgeführt und nicht von anderen Tasks unterbrochen werden, was klassische Nebenläufigkeitsprobleme fast gänzlich eliminiert.

Nachteile

Dadurch das Node.js jeweils nur ein Prozessorkern auslastet, kann die Skalierung nur begrenzt über leistungsfähigere Hardware erfolgen. Können jedoch mehrere Instanzen der Applikation parallel ausgeführt werden (wie z.B. in einer Cloudumgebung), wird dadurch dieses Problem wieder entschärft.

Da der Einsatz von JavaScript auf serverseite noch relativ jung ist, stehen auch noch nicht so viele fortgeschrittene Entwicklerwerzeuge zur verfügung, wie bei etablierteren Plattformen wie etwa .NET oder Java. Desweitern ist durch die fehlende Typsicherheit der Einsatz von JavaScript in grösseren Projekten schwierig.

4.2.3 WEITERE ALTERNATIVEN

Es gibt unzählige weitere Web Frameworks, welche die gestellten Anforderungen wohl ebenfalls problemlos erfüllen könnten. Allerdings hat der Einsatz zusätzlicher Programmiersprachen und Technologien immer auch den Preis, das mehr Einarbeitungszeit benötigt wird. Zudem würde fast jedes andere Web Framework einen anderen Web Server als IIS voraussetzen. Zwei unterschiedliche

Webserver (möglicherweise auch auf unterschiedlichen Betriebssystemen) würde die Wartung der Infrastruktur im produktiven Umfeld wesentlich erschweren.

4.2.4 ENTSCHEIDUNG

Da das Framework alle Anforderungen an das Web Frontend zu erfüllen scheint und schon einige Erfahrungen damit innerhalb des Teams vorhanden sind, fiel die Entscheidung auf ASP.NET MVC.

4.3 ARCHITEKTURÜBERSICHT

Da das Web Frontend sehr wenige Business Logik abbilden muss, konnte eine sehr einfache Architektur verwendet werden. Durch das ASP.NET MVC Framework wird die Grundstruktur auf Basis des MVC Patterns auch schon relativ stark vorgegeben. Da die Business Logik und die Persistenz auf dem Backend implementiert ist, konnte jedoch die Model Komponente (das M von MVC) durch den Guidelines API Client ersetzt werden.

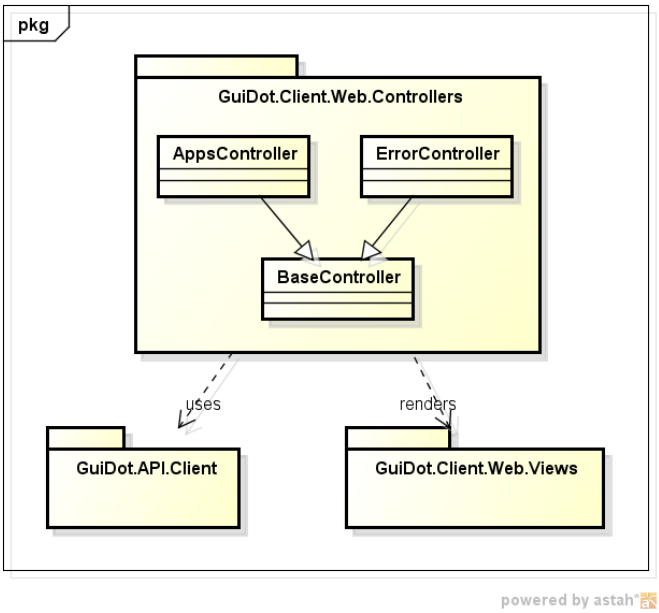


Abbildung 2 Die wichtigsten Komponenten des Web Frontends

Der **BaseController** ist die Basisklasse für alle Controller, die Zugriff auf die Guidelines API benötigen. Er stellt eine Instanz der Guidelines API zur Verfügung und verwaltet die API Access Tokens in den User Sessions.

5 BENUTZERINTERFACE

5.1 ANFORDERUNGEN

Das Benutzerinterface soll ein einfaches und effizientes Arbeiten mit der Guidelines API ermöglichen.

Um die Erweiterbarkeit und Wartbarkeit der Applikation zu garantieren, soll besonders auch bei der Entwicklung des Benutzerinterfaces auf gängige Software Engineering Praktiken zurückgegriffen werden. Dies ist ein Punkte der bei vielen Webprojekten stark vernachlässigt wird, da die Browserumgebung nur sehr wenige Vorgaben an die Strukturierung und Modularisierung des Codes stellt.

5.2 CODING STYLEGUIDE

Da JavaScript dem Entwickler sehr viele Freiheiten lässt, muss bei der Implementation besonders auf eine gewisse Konsistenz im Code und das Vermeiden fehleranfälliger Praktiken geachtet werden. Deshalb wurde als erster Schritt eine JavaScript Coding Guideline erarbeitet, die während der Umsetzung des Projektes zwingend eingehalten werden soll (siehe Anhang).

5.3 TECHNOLOGIEN

5.3.1 MODULARISIERUNG MIT REQUIREJS[2]

Beim Entwickeln von JavaScript für Browserplattformen stehen nur sehr einfache Mechanismen zur Modularisierung des Codes zur Verfügung. Der gängige Ansatz ist, das der Code auf mehrere Dateien verteilt wird, und diese anschliessend im HTML eingebunden werden. Dies ist für kleine Projekte zwar ausreichend, kann bei grösseren Applikationen aber schnell zu einigen Problemen führen:

- Verschmutzung des globalen Scopes
- Verwalten der einzubindenden Dateien wird unübersichtlich oder es wird mehr JavaScript Code geladen, als für das Ausführen der aktuell angeforderten Seite überhaupt notwendig
- Abhängigkeiten zwischen Modulen sind nur sehr schwer überschaubar

Deshalb wird ein Werkzeug benötigt, das - ähnlich wie Javas „import“ Anweisung - Module anderen Modulen zur Verfügung stellen kann und diese bei Bedarf in den Browser lädt.

Neben RequireJS gibt es noch einige andere Tools, welche ähnliche Funktionalitäten zur Verfügung stellen. RequireJS hebt sich aber durch einige Eigenschaften besonders hervor:

- **Asynchrones Nachladen der Module**: Der JavaScript Interpreter des Browsers wird während dem Nachladen der benötigten Dateien nicht blockiert (im Gegensatz zu synchronen Requests)
- **Name des Moduls nicht zwingend von dessen Position im Dateisystem abhängig**: Dadurch wird das einbinden von Libraries von Drittanbieter vereinfacht.

[2] http://requirejs.org/

- **Integriertes Code Minifying:** Dadurch das alle Abhängigkeiten von Modulen bekannt sind, kann für jede Seite ein Script File zur Verfügung gestellt werden, das nur die auch wirklich benötigten Module in minimierter Form enthält. Zwar müssen diese Dateien extra im Build Prozess erzeugt werden, können aber die Ladezeit einer Seite massiv verringern.
  Code Minifying umfasst in der Regel das Entfernen aller unnötigen Whitespaces und Kommentare und das verkürzen von nur lokal benötigten Symbole (Variablen- und Funktionsnamen).

5.3.2 UNIT TESTING MIT JASMINE[3]

Jasmine ist eines der meistgenutzten Unit Testing Frameworks für JavaScript und ermöglicht Tests im Stile von Behavior Driven Development (BDD). Durch das Anlehnen an BDD können aussagekräftige und übersichtliche Tests geschrieben werden.

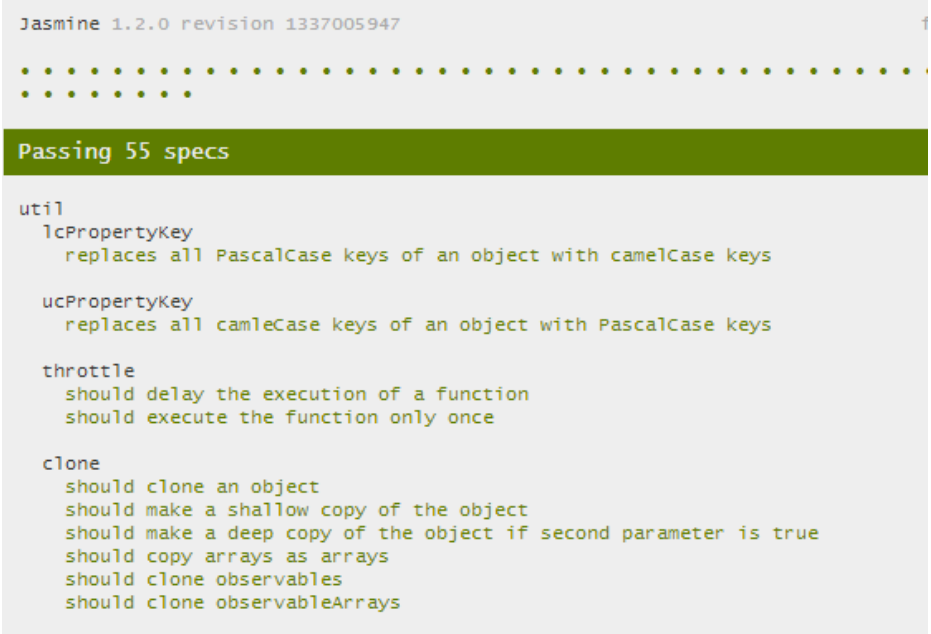


Abbildung 3 Auszug des Jasmine Testreports von GuiDot

5.3.2.1 AUSFÜHREN DER UNIT TESTS

Alle JavaScript UnitTests können im Browser ausgeführt werden.

1. Starten der Solution in Visual Studio
2. In beliebigen Browser http://localhost:3973/Public/tests/SpecRunner.html öffnen
3. Die Tests werden automatisch ausgeführt

5.3.2.2 HINZUFÜGEN WEITERER TESTSPEZIFIKATIONEN

Jasmine Tests werden in sogenannten Testspezifikationen geschrieben. Dies sind Dateien, die per Konvention im Ordner **GuiDot.Client.Web/Public/tests** abgelegt sind und die Endung **.spec.js** haben. Für die Ausführung der Tests ist die HTML Datei **SpecRunner.html** zuständig.

[3] http://pivotal.github.com/jasmine/

Der SpecRunner ist so konfiguriert, dass Tests die benötigten Module ebenfalls über RequireJS einbinden können. Um neue Testspezifikation ausführen zu lassen, muss die Spezifikation dem SpecRunner bekanntgegeben werden.

5.3.3 MVVM MIT KNOCKOUT[4]

Damit die Trennung zwischen Darstellung und Applikationslogik auch beim Benutzerinterface angewendet werden kann, empfiehlt sich die Verwendung eines JavaScript MVC oder MVVM Frameworks.

Die Anforderungen an ein solches Framework für GuiDot sind:

- **Deklaratives HTML**: HTML Dokumente enthalten keinerlei Applikationslogik. In HTML soll lediglich definiert sein, wo welche Daten angezeigt werden und welche Methoden des Controllers bei Benutzerinteraktionen ausgeführt werden.
- **Keine zusätzliche Template Engine**: Die Views sollen komplett in HTML geschrieben werden können. Dies ist insbesondere auch wichtig, da auf Serverseite die Views mit der Razor Engine von ASP.NET MVC generiert werden. Durch das verwenden einer zusätzlichen, clientseitigen View Engine würde die Übersichtlichkeit wohl sehr stark verringert und es könnten auch Konflikte auftreten (bei zu ähnlicher Syntax).
- **Bidirektionales Data Binding**: Änderungen im Modell sollten auch automatisch im View aktualisiert werden. Analog dazu sollen Änderungen im View (z.B. Benutzereingaben in Textfeldern) auch an das Modell gebunden werden können.
- **Einfach erweiterbar**: Es soll einfach möglich sein, eigene Data Bindings zu definieren, um zum Beispiel WYSIWYG Editoren von Drittanbietern einzubinden.

Knockout ist ein JavaScript MVVM Framework das diese Anforderungen erfüllen kann und dessen Entwicklung unter anderem auch von Microsoft vorangetrieben wird.

5.3.4 DRY STYLESHEETS MIT LESS[5]

Mit CSS steht in der Web Entwicklung ein Werkzeug zur Verfügung, mit dem die Darstellung der Elemente einer Seite definiert werden kann. Ein häufiges Problem beim Einsatz von CSS in grösseren Projekten ist jedoch, dass es keine Möglichkeit gibt CSS Regeln und Werte wiederzuverwenden und das DRY Prinzip (Don't Repeat Yourself) kaum eingehalten werden kann.

LESS ist eine Stylesheet Sprache, die dieses Problem löst, indem CSS unter anderem um Konstanten und Mixins erweitert wird. Im Gegensatz zu ähnlichen Sprachen ist der LESS Compiler in JavaScript geschrieben, was das Kompilieren der LESS Dateien direkt im Browser ermöglicht. Dadurch können LESS Stylesheets während der Entwicklungsphase mit `<link rel="stylesheet/less" type="text/css" href="style.less">` wie CSS Dateien eingebunden werden. Dies ist komfortabler als das manuelle Übersetzen nach Änderungen der Stylesheets und leider notwendig, da zu Beginn des Projektes noch keine Plugins für Visual Studio 2012 zur Verfügung standen, die diese Aufgabe direkt in der IDE ausführen konnten.

[4] http://knockoutjs.com/
[5] http://lesscss.org/

```
@red: #ef1122;

.important{
	border: 1px solid @red;
	font-weight: bold;
}

header{
	.important;
}

footer{
	.important;
}
```

```
header{
	border: 1px solid #ef1122;
	font-weight: bold;
}

footer{
	border: 1px solid #ef1122;
	font-weight: bold;
}
```

Abbildung 4 LESS Beispiel mit entsprechendem CSS (rechts)

5.3.5 BOOTSTRAP[6]

Bootstrap ist ein weitverbreitetes CSS Framework und bietet eine Sammlung wiederverwendbarer UI Komponenten sowie ein gitterbasiertes Layoutsystem. Zudem werden die Standardformatierungen der verschiedenen Browser vereinheitlicht, was die Entwicklung eines browserübergreifend konsistenten Benutzerinterfaces stark vereinfacht.

Neben Bootstrap existieren noch zahlreiche andere CSS Frameworks mit ähnlichem Umfang. Jedoch bietet Bootstrap den Vorteil, dass das Framework in LESS entwickelt wird und aus unabhängig voneinander verwendbaren Modulen besteht. Dadurch können in Bootstrap verwendete Mixins auch in den Guidelines spezifischen Stylesheets verwendet werden und es können nicht benötigte Module ausgeschlossen werden.

5.4 EVALUATION WYSIWYG EDITOR

Die Abschnitte einer Guideline werden in Form eines HTML Dokumentes abgelegt. Deshalb wird für das Benutzerinterface ein Editor benötigt, der es dem Benutzer erlaubt, strukturierte Texte zu verfassen.

Der HTML Standard bietet zwar mit dem **contenteditable** Attribut[7] bereits eine Schnittstelle die das Bearbeiten von Teilbäumen der DOM erlaubt, ist aber in allen Browsern sehr unterschiedlich umgesetzt und beinhaltet kein Benutzerinterface um Formatierungen vorzunehmen.

5.4.1 ANFORDERUNGEN

- **Semantische Formatierungen:** Der Benutzer soll die Möglichkeit haben, auszudrücken was für eine Bedeutung oder Stellenwert die verschiedenen Segmente seines Textes haben (Semantik). Eine Überschrift soll also als Überschrift markiert werden können, und nicht als unterstrichener Text mit erhöhter Schriftgrösse. Semantisch strukturierte Texte bieten zum einen den Vorteil, dass Dokumente konsistent dargestellt werden können und zum anderen können semantische Analysen verwendet werden um beispielsweise die Relevanz von

[6] http://twitter.github.com/bootstrap/
[7] http://www.whatwg.org/specs/web-apps/current-work/multipage/editing.html#contenteditable

Suchresultaten besser einzustufen (z.B. Treffer in einer Überschrift sind relevanter als im Fliesstext).

- **Ausgegebenes HTML browserübergreifend konsistent:** Da die meisten WYSIWYG HTML Editoren mit der **contenteditable** API arbeiten, kann das generierte HTML je nach verwendetem Browser sehr unterschiedlich aussehen[8]. So kann zum Beispiel die Formatierungsanweisung „bold“ den ausgewählten Text mit <b>, oder <span style=“font-weight: bold“> umschliessen. Dies erschwert semantische Analysen und führt zu Konflikten sobald Benutzer mit unterschiedlichen Browsern dasselbe Dokument bearbeiten.
- **Schnittstelle für Erweiterungen:** Um beispielsweise Ressourcen wie Bilder oder Links zu anderen Guidelines in die Dokumente einzubinden, muss der Editor so erweiterbar sein, damit er mit den Ressourcen der Guidelines API arbeiten kann.
- **WYSIWYG:** Dokumente sollen während dem Bearbeiten möglichst ähnlich dargestellt werden wie im Lesemodus.

5.4.2 CKEDITOR

Der CKEditor wurde in der Vorarbeit im Rahmen des „SE2 Projektes“ verwendet. Er wird sowohl unter einer OpenSource Lizenz (GPL) wie auch in einer kommerziellen Version verteilt. Damit der Editor in einem Closed Source Projekt wie Guidelines eingesetzt werden darf, muss eine Lizenz erworben werden (ab $ 375.-).

Vorteile

Der CKEditor wird sehr oft eingesetzt und verfügt über eine relativ grosse Community, die auch einige Plugins zur Verfügung stellt. Zudem verfügt er über einen sehr grossen Funktionsumfang und ist relativ frei konfigurierbar.

Nachteile

Der Editor legt keinen Fokus auf die semantische Formatierung und generiert je nach Browser auch unterschiedliches HTML. Zudem bietet die Dokumentation nur sehr spärliche Informationen zum erweitern des Editors.

5.4.3 ALOHA EDITOR

Wie der CKEditor ist Aloha unter der GPL Lizenz erhältlich und ist dehalb für kommerzielle / Closed Source Projekte ebenfalls kostenpflichtig.

Vorteile

Der Aloha Editor wird als semantischer HTML Editor beworben. Dementsprechend ist auch das Benutzerinterface auf die semantische Formatierung ausgelegt. Auch lehnt sich die Oberfläche des Editors relativ stark an den Ribbons der neueren Microsoft Office Versionen an, was vielen Benutzern entgegenkommen wird.

Nachteile

[8] http://www.quirksmode.org/dom/execCommand.html

Obwohl der Aloha Editor nicht mehr Funktionalitäten bietet, als der CKEditor, ist der Umfang des Quellcodes um ein vielfaches grösser. In optimierter und komprimierter Form ist der Editor grösser als 1 MB und kann dadurch das Laden einer Webapplikation deutlich verlangsamen.

Des weiteren ist es auch hier den Entwicklern nicht gelungen, HTML vollumfänglich semantisch korrekt und browserunabhängig auszugeben.

5.4.4 REDACTOR

Redactor ist ein relativ kleiner WYSIWYG Editor und ist ausschliesslich unter einer kommerzieller Lizenz erhältlich (ab $ 100.-). Es kann aber eine Probeversion bezogen werden.

Vorteile

Redactor ist sehr einfach aufgebaut und verfügt über eine schlanke, übersichtliche Dokumentation. Der Funktionsumfang ist aber nicht viel geringer als bei den anderen Editoren.

Nachteile

Auch Redactor kann die Kriterien bezüglich Semantik und Konsitenz nicht erfüllen.

5.4.5 ENTSCHEIDUNG

Da keiner der untersuchten Editoren die Anforderungen vollständig erfüllen kann, jedoch alle über etwa die selben Grundfunktionalitäten verfügen, wurde schlussendlich bei der Auswahl das Produkt berücksichtigt, das am einfachsten in die Applikation eingebunden werden kann und den geringsten Overhead bietet.

Durch die schlanke Architektur und die übersichtliche Dokumentation schien **Redactor** diese Kriterien am besten zu erfüllen.

6 SCHLUSSFOLGERUNGEN

Während der Studienarbeit wurde eine solide Grundlage für die Neuentwicklung von Guidelines.ch geschaffen. Dast System baut auf einer mehrschichtigen Architektur mit starker Entkoppelung von Backend und Benutzerschnittstelle auf.

Die Arbeit umfasst ein sehr breites Spektrum an verwendeten Technologien und Paradigmen und erstreckt sich von der Umsetzung der Persistierung und Suche auf Datenbankebene über die Implementation eines RESTful Webservices mit .NET bis zur Erstellung einer moderenen Browserapplikation mit JavaScript und HTML5.

Es konnten auch bereits schon einige grundlegende Use Cases umgesetzt werden:

- Das Suchen von Guidelines mit einem auf Benutzerfreundlichkeit und Schnelligkeit ausgelegten Suchkonzept
- Das Lesen, Bearbeiten und Erstellen von Guidelines inklusive Versionsverwaltung, Historisierung und WYSIWYG Editor
- Das An- und Abmelden von Benutzern

Weitere Use Cases wurden konzeptionell ausgearbeitet und wurden in der Entwicklung teilweise bereits berücksichtigt.

Desweitern wurde eine ACL basierte Zugriffskontrolle, die Authentifizierung von Clientanwendungen und Benutzer, die durchgängige Fehlerbehandlung auf Basis der HTTP Status Codes und die Fehlerprotokollierung umgesetzt.

6.1.1 VERGLEICH ZUR AKTUELLEN LÖSUNG

Im gegensatz zum aktuell eingesetzten System wurde bei der Entwicklung der neuen Lösung auch von Beginn weg grosser Wert auf die Skalierbarkeit der Applikatione gelegt. Dadurch dass der Webservice zustandslos arbeitet, können je nach Bedarf mehrere Instanzen gestartet werden. Dies ist auch auf Seite Frontend möglich, wobei jedoch der Zustand der Benutzersessions zwischen den Instanzen geteilt werden muss.

Bei der Analyse des bestehenden Systems wurden zusätzlich zur fehlenden Möglichkeit der quantitativen Skalierung auch konzeptionelle Probleme festgestellt, die den Einsatz der Software in mehreren Abteilung oder sogar Spitalübergreifend erschweren könnten. So kann zwar das aktuelle Konzept der Autoren und Co-Autoren bei einem kleinen Benutzerfeld durch seine Einfachheit überzeugen, stösst aber bei einer grösseren und verteilteren Benutzergruppe an seine Grenzen und ist sehr hierarchisch ausgelegt. Aus diesen Gründen wurde während der Domainanalyse das etwas offenere Konzept der Arbeitsgruppen eingeführt. Guidelines gehören nicht mehr einem einzelnen Autor sondern werden von Arbeitsgruppen verwaltet in denen Personen oder Organisationen in, je nach Fachwissen und Position, verschiedenen Rollen wie Contributor (Mitwirkender), Validator oder Administrator teilnehmen können. Dadurch soll Administrationsproblemen bei Mitarbeiterwechseln entgegengewirkt und zusätzlich auch die Zusammenarbeit von grösseren Gruppen ermöglicht werden.

Ein weiterer Punkt, der zumindest konzeptionell beachtet wurde und sich stark von der bestehenden Lösung unterscheidet, ist das Erweitern der Kommunikationsmöglichkeiten innerhalb einer Arbeitsgruppe, damit sich Benutzer einfacher über Änderungsvorschläge und Probleme einer Guideline austauschen können.

6.1.2 WEITERE SCHRITTE

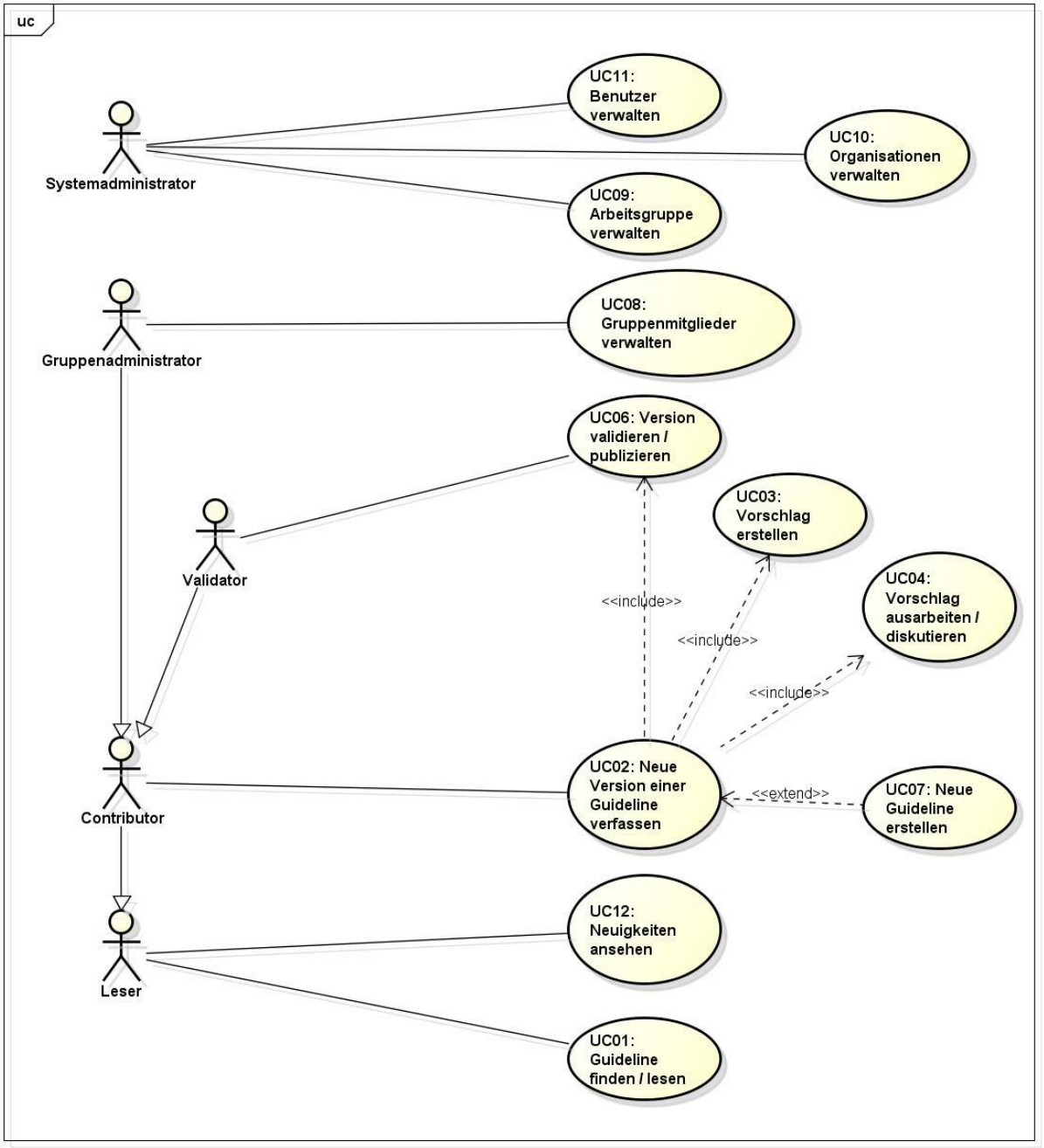
Zwar konnten in der Studienarbeit schon sehr viele Erkenntnisse gewonnen und umgesetzt werden, allerdings fehlen noch sehr viele Funktionalitäten bis die Software auch produktiv eingesetzt werden kann. Neben den noch nicht implementierten Use Cases gibt es auch noch weitere Aspekte die weiterverfolgt werden könnten. Zum Beispiel ist das Persistieren der Revisionen noch sehr speicherintensiv umgesetzt. Eine weitere Herausforderung die sich noch stellt, ist die freie Konfigurierbarkeit der Validierungsprozesse durch die Benutzer.

Auch bezüglich **Sicherheit** konnten noch nicht alle geplanten Aspekte berücksichtigt werden. Dazu gehört die Verschlüsselung des Login Vorgangs per HTTPS oder der Schutz vor Cross-Site Request Forgery Attacken (CSRF).

7 ANFORDERUNGSSPEZIFIKATION

7.1 USE CASES

7.1.1 DIAGRAMM



7.1.2 AKTOREN

Aktoren	Interessen

Leser	• Guidelines möglichst einfach finden • Guidelines anwenden und von den Informationen profitieren
Contributor	• Ideen einbringen können • Wissen teilen
Validator	• Sicherstellen, dass nur korrekte Informationen veröffentlicht werden
Gruppenadministrator	• Gruppe einfach administrieren können
Systemadministrator	• System einfach unterhalten • Möglichkeit zur Systemüberwachung

7.1.3 BESCHREIBUNGEN

7.1.3.1 UC01: GUIDELINE FINDEN

Ein Besucher der Seite www.guidelines.ch möchte zu einem bestimmten Thema eine Guideline finden.

Er gibt den ersten Suchbegriff in die Suchleiste ein und erhält eine Liste mit treffenden Guidelines als Vorschlag, dazu werden ihm die Tags der gefundenen Guidelines geliefert. Falls er einen Tag auswählt, wird die Suche auf Guidelines mit diesem Tag eingeschränkt. Es kann auch nach mehreren Tags gefiltert werden (und-Verknüpfung). Das Suchresultat enthält nur die Guidelines, auf denen der angemeldete Benutzer die Leserechte hat. Ist kein Benutzer angemeldet, werden nur öffentliche Guidelines angezeigt.

Wählt der Benutzer eine Guideline aus, so wird diese angezeigt.

7.1.3.2 UC02: NEUE VERSION EINER GUIDELINE VERFASSEN

Ein Contributor will einen Fehler in einer Guideline korrigieren, Ergänzungen anbringen oder den Inhalt aktualisieren.

Er erstellt einen neuen Vorschlag (siehe UC03) welcher anschliessend durch die Mitglieder der Arbeitsgruppe diskutiert und verfeinert wird (UC04). Nachdem ein Konsens über die Änderungen gefunden wurde, wird die Version validiert (UC06), was deren Publikation zur Folge hat.

7.1.3.3 UC03: VORSCHLAG ERSTELLEN

Ein Änderungsvorschlag wird durch das Editieren der Guideline erstellt. Eine Änderung betrifft ein oder mehrere Attribute der Guideline (z.B. der Inhalt, Tags, die Sichtbarkeit...). Zudem muss ein Grund für die Änderung angegeben werden, der das Problem mit der aktuellen Version beschreibt.

7.1.3.4 UC04: VORSCHLAG AUSARBEITEN / DISKUTIEREN

Die Mitglieder einer Arbeitsgruppe in der der Änderungsvorschlag erstellt wird, können Anmerkungen in einem Diskussionsthread verfassen. Zudem hat jedes Mitglied die Möglichkeit weitere Änderungen vorzunehmen.

7.1.3.5 UC05: DIFFERENZ ANZEIGEN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Um die Diskussion zu erleichtern, besteht auch die Möglichkeit Differenzen zwischen zwei Vorschlägen anzuzeigen.

7.1.3.6 UC06: VERSION VALIDIEREN / PUBLIZIEREN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Nachdem sich die Mitglieder der Arbeitsgruppe über einen Änderungsvorschlag einig geworden sind, wird die Validierung durch einen der Validatoren gestartet. Dazu bestätigt er die aktuelle Version der Änderung.

Sobald die Validierung gestartet wurde, können keine Änderungen an der Guideline vorgenommen werden. Kommentare im Diskussionsthread können weiterhin von allen Contributoren verfasst werden.

Die Änderung muss von allen Validatoren der Arbeitsgruppe bestätigt werden. Lehnt einer der Validatoren den Änderungsvorschlag ab, wird die Validierung zurückgesetzt. Um eine Änderung abzulehnen muss zwingend ein Grund angegeben werden.

Sobald der Validierungsprozess erfolgreich abgeschlossen wird, werden die Änderungen im festgelegten Sichtbarkeitsbereich publiziert. Die Guideline erhält damit auch eine neue Versionsnummer.

7.1.3.7 UC07: NEUE GUIDELINE ERSTELLEN

Ein Benutzer von Guidelines bemerkt, dass eine relevante Guideline noch nicht vorhanden ist und möchte diese erstellen.

Er erstellt eine neue Guideline und schlägt diese einer Arbeitsgruppe vor (entweder muss er Mitglied dieser Arbeitsgruppe sein oder einfach nur ein Benutzer mit Account). Der weitere Ablauf ist identisch mit dem Verfassen einer neuen Version einer Guideline (UC02).

7.1.3.8 UC08: GRUPPENMITGLIEDER VERWALTEN

7.1.3.8.1 UC8.1: GRUPPENMITGLIEDER BEARBEITEN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Ein neuer Mitarbeiter soll Mitglied einer Arbeitsgruppe werden.

Dazu erstellt der Gruppenadministrator eine neue Mitgliedschaft und wählt die Rolle des Mitarbeiter (Gruppenadministrator, Validator oder Contributor) aus. Falls die Rolle Validator ausgewählt wurde, kann es sinnvoll sein, dass zusätzlich auch ein Themengebiet angegeben wird.

Mitgliedschaften können auch gelöscht und bearbeitet werden.

7.1.3.8.2 UC8.2: GRUPPENMITGLIEDER ANZEIGEN

Es können die Mitglieder einer Arbeitsgruppe angezeigt werden.

Hierfür öffnet der Benutzer eine Arbeitsgruppe. Daraufhin werden ihm die Benutzer sowie die Guidelines der Arbeitsgruppe angezeigt.

7.1.3.9 UC09: ARBEITSGRUPPEN VERWALTEN

7.1.3.9.1 UC9.1: ARBEITSGRUPPEN BEARBEITEN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Ein Systemadministrator kann neue Arbeitsgruppen erstellen. Dazu legt er einen Gruppennamen und eine Beschreibung der Gruppe fest und wählt mindestens einen Gruppenadministrator.

Gruppen können auch bearbeitet und deaktiviert werden.

7.1.3.9.2 UC9.2: ARBEITSGRUPPEN ANZEIGEN

Wird auf die Homepage zugegriffen werden dem Benutzer die Arbeitsgruppen des Systems angezeigt. Ist der Benutzer zusätzlich noch eingeloggt, werden ihm die Arbeitsgruppen angezeigt, in denen er Mitglied ist.

7.1.3.10 UC10: ORGANISATIONEN VERWALTEN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Ein Systemadministrator kann die Organisationsstruktur editieren. Dazu kann er neue Organisationen (Spitäler, Spitalverbunde, Abteilungen etc.) erstellen oder Organisationen umbenennen oder löschen. Zudem kann er Benutzer und Organisationen einer Organisation zuordnen.

7.1.3.11 UC11: BENUTZER VERWALTEN

Ein Systemadministrator kann Benutzer erstellen, bearbeiten oder löschen.

7.1.3.12 UC12: NEUIGKEITEN ANSEHEN

Dieser Use Case wird nicht im Rahmen der SA umgesetzt.

Nachdem sich ein Benutzer eingeloggt hat, wird er auf sein Dashboard weitergeleitet. Dieses zeigt Informationen an, die den Benutzer interessieren. Dazu gehört:

- Ein Änderungsvorschlag in einer Arbeitsgruppe wurde erstellt
- Die Validierung eines Änderungsvorschlages wurde eingeleitet
- In einem Änderungsvorschlag in dem der Benutzer mitwirkt, wird ein Kommentar erfasst oder etwas abgeändert
- Der Benutzer wird einer Arbeitsgruppe hinzugefügt / aus einer Arbeitsgruppe ausgeschlossen

Der Benutzer hat auch die Möglichkeit, sich ausgewählte Typen von Neuigkeiten per E-Mail zustellen zu lassen.

7.2 WEITERE ANFORDERUNGEN

Die weiteren Anforderungen umfassen Punkte, die für das Projekt Guidelines.NET besonders ausschlaggebend sind.

7.2.1 QUALITÄTSMERKMALE

7.2.1.1 BENUTZERFREUNDLICHKEIT

Damit Guidelines.NET erfolgreich eingesetzt werden kann, ist die Mitarbeit der Benutzer besonders wichtig. Um die Einstiegshürde für die Benutzer möglichst gering zu halten, muss besonders Wert auf eine hohe Benutzerfreundlichkeit gelegt werden.

Die Benutzerführung soll möglichst Aufgabenorientiert erfolgen. Dies bedeutet zum Beispiel auch, dass Beschriftungen von Schaltflächen klar und verständlich formuliert sind und einen Bezug zum entsprechenden Task haben (z.B. „Suchen“, „Guideline bearbeiten“, ...).

7.2.1.2 EFFIZIENZ

Die Ausführung der einzelnen Operationen sollen nicht mehr als 0.5 Sekunde in Anspruch nehmen. Dies betrifft insbesondere Operationen wie das Suchen nach Guidelines, das Erstellen und Öffnen von Guidelines. Das heisst, dass diese Operationen auf dem Server innerhalb von 300ms abgearbeitet werden müssen. Bei der Suche könne Teilresultate zurückgeliefert werden, um die zeitlichen Vorgaben zu erreichen. Erwähnte Werte gehören zu Standardvorgaben für die Webprogrammierung und behindern den Benutzer bei der Arbeit nicht. Von dieser Regel ausgeschlossen sind der Download und Upload von Dokumenten, was je nach Grösse mehr Zeit in Anspruch nehmen kann, und das Nachladen von grösserem Inhalt (z.B.: Bilder).

7.2.1.3 ÜBERTRAGBARKEIT

Guidelines.NET soll für alle Benutzer, die über einen modernen Browser verfügen, verwendbar sein. Unabhängig vom darunterliegenden Betriebssystem.

Als moderner Browser gilt unter anderem:

- **Internet Explorer 9** oder neuer
- **Firefox** aktuelle Version (zur Zeit 15)
- **Chrome** aktuelle Version (zur Zeit 21)

- **Opera** aktuelle Version (zur Zeit 12)
- **Safari** aktuelle Version

7.2.1.4 SKALIERBARKEIT

Da Guidelines.NET für den Einsatz in mehreren Spitälern entworfen wird und theoretisch in allen Spitälern der Schweiz eingesetzt werden könnte. Kann die Anzahl Benutzer auf mehrere 100‘000 anwachsen. Deshalb soll die Applikation leicht skalierbar sein.

7.2.1.5 LOKALISIERUNG

Guidelines.NET wird, im Rahmen der Studienarbeit, nur auf Deutsch verfügbar sein. Da sich das Projekt aber auch durchaus über die Sprachgrenzen hinweg weiterentwickeln kann, sollen einige Aspekte der Internationalisierung bei der Entwicklung beachtet werden:

- Vorbereiten für mehrere Sprachen
- Übersetzungen der Texte
- Formatierung (z.B.: Datum) auslagern

7.2.1.6 WARTBARKEIT

In Bezug auf die Wartung soll die Businesslogik soweit möglich mit Tests abgedeckt werden. So wird sichergestellt, dass bei Erweiterungen, die bestehende Funktionalität immer noch korrekt ausgeführt wird.

Zur Unterstützung der Fehlersuche wird zudem ein einfaches Logging eingebunden.

7.2.2 SCHNITTSTELLEN

Da eine gute Skalierbarkeit gefordert ist, steht eine Web-Lösung, wie sie bereits besteht im Vordergrund. Um den anderen Anforderungen gerecht zu werden und auch die nötigen Freiheiten zu gewährleisten, werden die Daten und Businesslogik über eine REST-Schnittstelle zugänglich gemacht. So können beliebige Clients diese Schnittstelle anderweitig auch einbinden. Für den Zugriff per Browser wird eigens ein Webserver bereitgestellt, der dieselbe Schnittstelle verwenden wird und die benötigen Informationen (Session, State) verwaltet.

B GLOSSAR

1 PERSONEN

Abkürzung	Erklärung
lk	Lukas Kretschmar
lw	Lukas Wegmann
qw	Quentin Willimann

2 PROJEKT

Abkürzung	Erklärung
GuiDot	Projekt Guidelines.NET
CEST	Central European Summer Time
CST	China Standard Time
GMT	Greenwich Mean Time

3 DOKUMENTATION

Bezeichnung	Beschreibung
.NET	Programmierframework, welches auf der Windows-Plattform weit verbreitet ist und von Microsoft stammt.
ASP.NET	Technologie des .NET-Frameworks für die Entwicklung von Web-Applikationen
ASP.NET MVC	Erweiterung für ASP.NET, welche auf dem MVC-Pattern aufbaut
ASP.NET WebAPI	Technologie des .NET-Framework für die Entwicklung von Web-Schnittstellen. Ist Teil von ASP.NET.
Attribute	Annotation in C#-Code um dem Compiler oder einem Custom Tool Metainformationen mitzuteilen. Beispiel: [WebGet], [WebInvoke]
Binding (WCF)	Ein Binding in WCF definiert, auf welche Transporttechnologie & -protokolle WCF aufgesetzt werden soll. Beispiel: NetTcpBinding (für TCP/IP), WebHttpBinding (für REST)
C#	*CSharp* Programmiersprache von Microsoft, welche das .NET-Framework verwendet.
Changetracking	Konzept für das Nachverfolgen von Änderungen von Daten.
CLR	*Common Language Runtime* Laufzeitumgebung, auf der alle .NET-Sprachen ausgeführt werden können.
Concurrency	Nebenläufigkeit
Dependency Injection	Konzept wie Instanzen von Klassen automatisch durch ein Framework zur Verfügung gestellt werden können.
DOM	*Document Object Model* Schnittstelle für Zugriffe auf HTML-Dokumente
DOTNET	*siehe „.NET“*
DTO	*Data Transfer Object* Objekt, welches für den Transport von Daten zwischen zwei oder mehreren Schnittstellen verwendet wird. Es enthält keine oder nur so wenig wie möglich an Funktionalität.
EF	*Entity Framework* Datenbankzugriffs Framework im .NET-Framework.

Framework	Sammlung von Programmierbibliotheken.
IIdentity	Interface, welches den Benutzer bei der Authentifizierung abstrahiert.
IPrincipal	Basis-Interface für die Authentifizierung in .NET.
JS	*JavaScript* Programmiersprache, welche auf Browsern ausgeführt werden kann.
Layer	Abstrahierte Ebene einer Software. So gibt es meist eine Ebene für den Datenzugriff, für die Datenverarbeitung und die Darstellung.
MVC	*Model-View-Controller* Weitverbreitetes Pattern für die Entwicklung von Benutzeroberflächen. Das Pattern teilt die Verantwortlichkeit für Datenhaltung (Model), das Anzeigen der Benutzeroberfläche (View) und der Behandlung von Benutzereingaben (Controller) auf.
MVVM	*Model-View-ViewModel* Pattern für die Verbindung von Daten (Model) an ein UserInterface (View) über einen zwischen Layer (ViewModel)
POCO	*Plain Old CLR Object* Objekt, welches nur Wertetypen oder andere POCO's enthält.
Request	Anfrage an einen Server/Service
Response	Antwort eines Servers/Services
REST	*Representational State Transfer* Architekturparadigma für die Entwicklung von Webservices
Self-Hosting (WCF)	Hosting eines Services nicht über ISS (und Konifugrationsdatei) sondern über eigene Applikation.
SQL	*Structured Query Language* Sprache mit der Datenbanken abgefragt werden können.
StoredProceudre	Prozedur, die in SQL programmiert ist und direkt auf der Datenbank ausgeführt werden kann.
Unity	Framework, welches für Dependency Injection verwendet wurde.
WCF	*Windows Communication Foundation* Framework für Interprozess- oder Netzwerkkommunikation.
WIF	*Windows Identity Foundation* Technologie des .NET-Frameworks für die Authentifizierung von Benutzern und/oder Systemen.
Wrapper	Konstrukt, welches Informationen kapselt und gegen aussen in einer umgebungsfreindlicheren Form bereitstellt. Beispiel: ein HTTP GET Request wird in ASP.NET WebAPI als RequestMessage()-Klass gekapselt, sodass einfach damit programmiert werden kann.