

# Video Streaming Web Application

Ein Prototyp für moderne Webtechnologien

## **BACHELOR THESIS**

Verfasser: Tobias Studer

Betreuung: Marcus Hudritsch

Olten, 15. August 2012



## Zusammenfassung

Diese Bachelor Thesis beschäftigt sich mit der Portierung einer Desktopsoftware ins Web. In einem ersten Teil wird eine bereits bestehende Teilimplementierung analysiert. In einem zweiten, praktischen Teil wird ein vereinfachter Prototyp einer Webapplikation mit modernen Technologien erstellt.

## Keywords

.NET 4	Video	Visual Studio 2010
C#	JavaScript	NuGet
ASP.NET	KnockoutJS	TeamCity
ASP.NET MVC4	Upshot.js	Git/GitHub
ASP.NET Web API	Bootstrap, from Twitter	WebDeploy
Entity Framework (Code First)	HTML5	IIS
Razor View Engine	CSS3	Continuous Integration
LINQ	LESS	Continuous Deployment
Silverlight	JSON	ELMAH
Smooth Streaming	Single Page Applications	Glimpse

## Vorwort

Das Erarbeiten einer Bachelor Thesis ist ein spannendes Unterfangen und ich möchte mich an dieser Stelle bei den beteiligten Personen und Parteien herzlich bedanken. Insbesondere betrifft dies den Betreuer dieser Arbeit, Marcus Hudritsch. Seinem Interesse und Engagement ist es zu verdanken, dass diese Arbeit stets vorangetrieben wurde. Auch der CDLab AG in Murten als Auftragsgeber, vertreten durch Martin Hien und Beat Brand, gebührt ein grosses Dankeschön für die Offenheit und Kooperation.



## Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Ausgangslage .....	1
1.2	Problemstellung.....	1
1.3	Aufgabenstellung.....	1
2	Analytischer Teil.....	2
2.1	Performance .....	2
2.2	Business Model vs. Database Model .....	2
2.3	Namensgebung bei Funktionen.....	3
2.4	Fehlerbehandlung.....	3
2.5	Entity Framework .....	3
2.6	Debugging.....	3
2.7	Rückgabewerte .....	4
2.8	Code Coverage.....	4
3	Praktischer Teil.....	5
3.1	Single Page Applications .....	5
3.2	Server Technologien .....	6
3.2.1	ASP.NET MVC 4 (Beta).....	6
3.2.2	Entity Framework (Code First) .....	7
3.2.3	ASP.NET Web API .....	8
3.3	Client Technologien .....	8
3.3.1	KnockoutJS.....	9
3.3.2	Upshot.js.....	10
3.3.3	Bootstrap, from Twitter .....	10
3.4	Ergänzende Technologien.....	12
3.4.1	Validierung.....	12
3.4.2	Globalisierung.....	13
3.5	Videos im Web.....	14
3.5.1	Video Container .....	15
3.5.2	Video Codecs .....	15
3.5.3	Was funktioniert im Web.....	16
3.5.4	Microsoft Media Platform Player Framework .....	16
3.6	Entwicklung .....	17
3.6.1	NuGet .....	17
3.6.2	Chrome Entwickler Tools .....	17
3.6.3	Fiddler.....	18

3.6.4	ELMAH .....	18
3.7	Entwicklungsumgebung.....	19
3.7.1	Git und GitHub.....	19
3.7.2	Continuous Integration.....	20
3.7.3	Continuous Deployment.....	21
3.8	Endprodukt.....	21
3.8.1	Infrastruktur.....	22
3.8.2	Informationen.....	23
4	Schluss .....	24
4.1	Zukunftsblicke.....	24
5	Anhang.....	A
5.1	Bibliographie.....	A
5.2	Abbildungsverzeichnis .....	B
6	Ehrlichkeitserklärung.....	C

# 1 Einleitung

Diese technische Dokumentation der Bachelor Thesis ist eine Ergänzung zu den praktischen Arbeiten, die während diesem Semester und teilweise auch aus dem Vorgängerprojekt des letzten Semesters entstanden sind. Sie beschreibt und erklärt die eingesetzten Methoden und Technologien in vereinfachten Szenarien und Beispielen, dient jedoch nicht als Handbuch oder detaillierte Beschreibung der in diesem Semester entstandenen Webapplikation.

## 1.1 Ausgangslage



Die Firma CDLab AG in Murten ist eine der führenden Softwarehersteller in der Kanalinspektionsbranche. Bei einer modernen Kanalinspektion werden verschiedenartige Daten, insbesondere Videodaten, Textdaten sowie Sensordaten (wie z.B. die Neigung des Kanalabschnittes) erfasst. Diese Daten werden bisher noch bei vielen Stadtverwaltungen auf DVD gebrannt und dann archiviert. Die Verwaltung dieser Daten und der Zugriff darauf ist daher sehr umständlich.

## 1.2 Problemstellung

Da die bisherige Datensicherung sehr aufwändig ist, soll diese durch eine Webapplikation erheblich vereinfacht werden. Folgende Probleme sind zu lösen:

- Die Wiedergabe von Videos auf verschiedenen Endgeräten mit unterschiedlich schnellen Datenleitungen zu ermöglichen.
- Eine desktopähnliche Webapplikation mit schnellen Reaktionszeiten zu schaffen, um ein gutes Benutzererlebnis zu erzeugen.

## 1.3 Aufgabenstellung

Die CDLab arbeitet bereits daran, ihre Desktopsoftware WinCan<sup>1</sup> ins Web zu portieren. Deshalb wird in einem ersten analytischen Teil die bestehende Implementierung untersucht. Danach wird ein Konzept erarbeitet und umgesetzt, das ein vereinfachtes Model der kompletten Software mit modernsten Webtechnologien darstellt. Dabei werden die aus dem Vorgängerprojekt entstandenen Erkenntnisse ergänzt und eingebaut.



---

<sup>1</sup> <http://wincan.com/>

## 2 Analytischer Teil

Die Softwarearchitektur der WinCan Web Lösung sieht vor, dass sämtliche Abfragen von Daten über Webservices erfolgen. Dadurch kann die Kommunikation des Webserver mit der Programmlogik und der dahinterliegenden Datenbank genau definiert werden. Im Folgenden werden verschiedene Aspekte dieser Implementierung betrachtet. Dabei ist zu beachten, dass die Webapplikation als Webserver die Internet Information Services (IIS) benutzt.

### 2.1 Performance

In einer Webumgebung ist die Geschwindigkeit, mit welcher einfache Daten an den Client gelangen, entscheidend. Um das Verhalten dieser Geschwindigkeit abzuschätzen, wurden verschiedene Szenarien von Datenanfragen als automatisierte Tests programmiert. Diese schreiben die Ergebnisse während der Ausführung in Textdateien. Dabei hat sich gezeigt, dass die jeweils erste Anfrage von Daten über die Webservices durchschnittlich fünf bis zwanzig Sekunden dauert. Darauf folgende Anfragen werden im Millisekunden Bereich behandelt.

```
1 *****
2 Executing method: GetSingleProjectByPrimaryKey
3     Calling method: ProjectRepository.GetProject(Guid 4388b72c-9748-4d6d-a4bb-a5731bf359fd)
4         1. Execution time: 00:00:20.5009870
5         2. Execution time: 00:00:00.0837521
6         3. Execution time: 00:00:00.1253946
7         4. Execution time: 00:00:00.0747736
8         5. Execution time: 00:00:00.1533918
9         6. Execution time: 00:00:00.1289978
10        7. Execution time: 00:00:00.0600016
11        8. Execution time: 00:00:00.1211588
12        9. Execution time: 00:00:00.0585266
13        10. Execution time: 00:00:00.1312703
14        -----
15        Total Execution time: 00:00:21.4382542
16        Average Execution time: 00:00:02.1438254
17        =====
```

Abbildung 1 - Logfile eines automatisierten Tests

Dieser Umstand ist auf den Application Pool zurückzuführen, in welchem die Webapplikation ausgeführt wird. Ein Application Pool eines IIS wird dazu benutzt, eine oder mehrere Webapplikation zusammen zu fassen und von anderen Webapplikationen auf demselben Webserver zu isolieren. Jeder Application Pool läuft in einem eigenen Prozess und muss bei der ersten Anfrage zuerst gestartet werden. Daraus resultieren die relativ langen ersten Reaktionszeiten. Problematisch wird es nun, weil jeder Application Pool ein Timeout hat. Das heisst, er wird nach einer definierten Leerlaufzeit wieder heruntergefahren. Daraus ergibt sich wiederum eine sehr lange Wartezeit bei der nächsten Datenanfrage. Es gibt grundsätzlich zwei Möglichkeiten, dies zu umgehen. Bei der ersten Möglichkeit wird das Timeout so gross gewählt, dass es fast nie ausgelöst wird. Die zweite Variante ist die sogenannte Keep-Alive Methode, welche in regelmässigen Abständen eine Anfrage an die Webapplikation schickt, um so das Timeout zu verhindern.

### 2.2 Business Model vs. Database Model

Die Daten der Webapplikation befinden sich in einer relationalen Datenbank. Die Entitäten dieser Datenbank werden im Programmcode durch Database Models abgebildet. Dies sind C# Objekte, die automatisch aus den Tabellen in der Datenbank erzeugt werden. Durch unvorsichtige Erzeugung haben sich mehrere Versionen im Code eingeschlichen. Man findet deshalb als Repräsentation der abgespeicherten Projektdaten aus der PROJECTS Tabelle Objekte mit den Namen PROJECT1, PROJECT2 usw. Dies macht es für einen externen Entwickler sehr schwierig bis unmöglich, zu wissen mit welcher Version der Objekte man beim Programmieren arbeiten soll. Zusätzlich erhöht es die



Fehlerwahrscheinlichkeit und führt dazu, dass an mehreren Stellen im Code Änderungen vorgenommen werden müssen, falls das Database Model zu einem späteren Zeitpunkt noch einmal generiert wird.

In der Businesslogik der Applikation werden die Database Models in Business Models abgefüllt. Die Business Models haben zusätzliche Funktionen, sind aber grundsätzlich alle gleich aufgebaut. Sie enthalten das Database Model als direkten Objektverweis. Diese Kapselung macht Sinn, wenn damit die direkte Verwendung der Database Models in den höheren Programmschichten unterbunden werden soll. In den Funktionen der Webservices wird dies aber nicht konsequent umgesetzt.

## 2.3 Namensgebung bei Funktionen

Die Namensgebung der Funktionen der Webservices lehnt sich an den versionierten Namen der Database Models an. So gibt es zum Beispiel im UserService folgende zwei Funktionen:

- `public void NewProject1(...)`
- `public void NewProject2(...)`

Bei diesen Versionen ergeben sich dieselben Probleme, wie bei den Database Models. Zudem sind die Funktionsnamen inkonsequent und nicht immer sehr aussagekräftig, wenn man beispielsweise das Auslesen einzelner Objekte aus verschiedenen Tabellen mit bekannter ID vergleicht:

- `service.FindNodeInspectionByPK(id);`
- `var criteria = String.Format(„OBJ_PK={0}“, id);  
service.NODES(criteria, String.Empty, 0, 1);`

Da die einzelnen Funktionen nicht dokumentiert sind, ist es teilweise schwierig, die korrekten Schnittstellen der Webservices anzusprechen.

## 2.4 Fehlerbehandlung

Sämtliche Fehler, die bei der Datenabfrage auftreten, werden durch die Business Models geschluckt. Das heisst, es gibt keine Möglichkeit, die geworfenen Exceptions in einer höheren Schicht abzufangen. Es bleibt nur den Wert in `result.ObjectState.Exception` zu überprüfen. Dieser ist ein einfacher Textwert und die verschiedenen Fehlertypen können so nicht auf einfache Art differenziert werden.

## 2.5 Entity Framework

Für das Filtern, Sortieren und Verknüpfen von Datensätzen werden keine streng-typisierten Abfragen verwendet, wie es das Entity Framework unterstützt. Die textbasierten Kriterien benötigen ein vertieftes Wissen über die inneren Strukturen der Datenbank und sind fehleranfällig:

- `var criteria = String.Format(„INS_Section_FK={0}“, section.OBJ_PK);`

Besser wäre die Verwendung von LINQ<sup>2</sup> wie im folgenden Beispiel:

- `GetPermissionMatrix(filter: p => p.OrganizationalUnit.Equals(„internal“),  
orderBy: p => p.OrderBy(m => m.Name));`

## 2.6 Debugging

Die Webservices sind nicht in der Visual Studio Solution eingebettet. Sie müssen für das Testen lokal installiert werden. Dadurch ist es nicht möglich, die Services direkt in der Entwicklungsumgebung zu debuggen. Das Debuggen der zurückgegebenen Objekte der Webservices erweist sich durch die oben genannte Fehlerbehandlung als umständlich, weil man immer sehr tief in die Struktur der Rückgabeobjekte schauen muss, um mögliche Fehler zu entdecken.

---

<sup>2</sup> Language Integrated Query ([http://en.wikipedia.org/wiki/Language\\_Integrated\\_Query](http://en.wikipedia.org/wiki/Language_Integrated_Query))

## 2.7 Rückgabewerte

Die Rückgabewerte der Webservice Funktionen sind immer Business Models. Diese haben immer dieselbe Struktur. Wie bereits erwähnt, befindet sich im Business Model eine Referenz auf das Database Model. Diese wird entweder in der \*.Data oder \*.DataSet Eigenschaft abgelegt. Aus Sicht des Programmcodes ist es nicht vorhersagbar, mit welcher Eigenschaft man auf die gewünschte Datenstruktur zugreifen kann. Ob man als Rückgabewert einen einzelnen Wert oder eine Liste von Werten erwartet, wird nicht unterschieden, denn es wird immer genau ein Business Objekt zurückgegeben.

## 2.8 Code Coverage<sup>3</sup>

Der komplette Programmcode verfügt über keine automatisierten Tests. Programmierfehler werden dadurch nur schwer und erst spät gefunden. Aufwändiges Debuggen ist notwendig, um zu testen oder Fehler zu finden. Die dafür aufgewendete Zeit könnte für das Schreiben von automatisierten Tests verwendet werden. Damit wird auch die Code Qualität gesteigert.

---

<sup>3</sup> Prozentualer Anteil an Code der durch automatisierte Tests abgedeckt wird

### 3 Praktischer Teil

Die Webentwicklung ist ein sehr spannendes und dynamisches Umfeld. Viele verschiedene Faktoren spielen dabei zusammen. Es erfordert ein breites Wissen und gute Kenntnisse der einzelnen Komponenten, welche im Webumfeld zusammenspielen. Serverseitig benötigt man einen Webserver und in den meisten Fällen einen Datenspeicher, zum Beispiel in Form einer Datenbank. Zusätzlich wird eine Laufzeitumgebung vorausgesetzt, die den geschriebenen Programmcode interpretieren kann. Da sich dieses Projekt im Windowsbereich bewegt, basiert es auf dem .NET Framework in der Version 4 und dem IIS<sup>4</sup> als Webserver. Als Grundlage der Programmierung der Webanwendung wird ASP.NET eingesetzt.



Clientseitig befindet sich der Internetbrowser, der den vom Server angelieferten Markupcode interpretieren kann. Die Hypertext Markup Language, oder kurz auch HTML, definiert die Struktur einer Webseite. Alle graphischen Komponenten werden mit CSS<sup>5</sup> gesteuert. Beide Konzepte existieren im Webumfeld schon länger und haben sich mit der Zeit weiterentwickelt. HTML5 ist die neuste Entwicklung, ist aber noch nicht fertig definiert. Trotzdem setzen immer mehr Webseiten diese Version ein und auch

die verschiedenen Browser unterstützen immer mehr die neuen Funktionen dieser Version. Leider sind nicht alle Browser auf demselben Stand und deshalb ist es für einen Webentwickler auch immer eine Herausforderung, Kompromisse zu schließen, um möglichst viele verwendete Browser so gut als möglich zu unterstützen. Auch CSS hat sich mit der Zeit entwickelt und steht vor derselben Herausforderung. Hat der Webserver die HTML und CSS Dateien einmal an den Browser, also den Client gesendet, ist diese Seite grundsätzlich statisch. Das heisst, um neue Inhalte auf einer Webseite anzuzeigen muss eine neue Anfrage an den Webserver geschickt werden, welche dieser bearbeitet und anschliessend die neuen Inhalten wieder an den Client zurückschickt. Um die Inhalte im Browser dynamischer zu machen, kann JavaScript verwendet werden. JavaScript ist eine Skriptsprache und kann von allen gängigen Browsern interpretiert werden. So ist es möglich, nur einzelne Teile einer Seite zu aktualisieren oder Daten nachzuladen. Die Verlagerung von Programmlogik auf den Client hat in den letzten Jahren zugenommen und so entstanden völlig neue Möglichkeiten im Web.

Ein weiterer wichtiger Teil der Webentwicklung ist die Kommunikation zwischen dem Server und den Clients. Diese wird in den meisten Fällen über HTTP<sup>6</sup> oder der verschlüsselten Variante davon (HTTPS) durchgeführt. In welcher Form diese Daten übertragen werden, ist davon abhängig, wie diese Daten später verarbeitet werden sollen. Diese Arbeit beschäftigt sich nicht vertieft mit den Netzwerkkommunikationstechnologien. Trotzdem ist es ein wichtiger Bestandteil der Webprogrammierung, zum Beispiel wenn es darum geht, Daten asynchron nachzuladen.

#### 3.1 Single Page Applications<sup>7</sup>

Für Webapplikationen zeigt sich der Trend, dass sich das Konzept der Single Page Applications durchsetzt. Dieses Konzept bringt reaktive Applikationen hervor, die das Beste aus Web und Desktop kombinieren. HTML5 und JavaScript spielen dabei eine wichtige Rolle. In einer Single Page Application werden für einzelne Teile einer Webapplikation nicht mehr komplett eigenständige Seiten geladen, sondern grundsätzlich eine, die je nach Zustand der Applikation, andere Inhalte darstellt. Grosse Teile der Logik werden dafür auf die Clientseite verlagert und Daten werden nach Bedarf asynchron nachgeladen. Dies führt zu einem angenehmeren Benutzererlebnis und ist zudem zusätzlich

---

<sup>4</sup> Internet Information Services

<sup>5</sup> Cascading Style Sheets

<sup>6</sup> Hypertext Transfer Protocol

<sup>7</sup> <http://www.asp.net/single-page-application>

unabhängig vom jeweiligen Endgerät. Beispiele für Single Page Applications sind moderne E-Mail Clients wie Gmail.

### 3.2 Server Technologien

Dieses Projekt besteht aus einer Vielfalt verschiedenster Technologien. Es müssen jedoch für Single Page Applications nicht genau die für diese Arbeit ausgewählten Frameworks und Libraries eingesetzt werden. Die grosse Stärke einer solchen Programmierung liegt sogar darin, dass einzelne Komponenten ausgetauscht werden können. Einzelne Frameworks befinden sich während der Entwicklung noch in Beta-Versionen und können sich deshalb bis zu der finalen Version noch verändern.

#### 3.2.1 ASP.NET MVC 4 (Beta)<sup>8</sup>

Das Model-View-Controller (MVC) Architektur Muster separiert eine Applikation in drei Hauptkomponenten: die Models, die Views und die Controller. MVC ist ein Standard Design Pattern, mit welchem viele Entwickler vertraut sind.

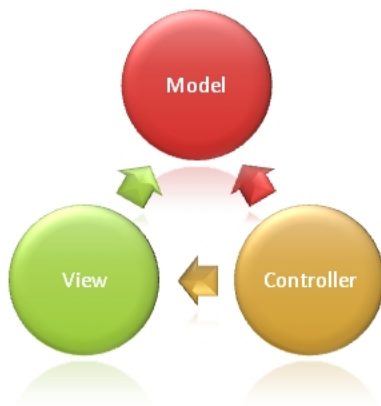


Abbildung 2 - Die Hauptkomponenten von MVC

**Models.** Model Objekte sind diejenigen Teile der Applikation, welche die Datendomäne definieren. Oft lesen und schreiben sie ihren Zustand aus einer Datenbank. Die Models werden als POCOs<sup>9</sup> implementiert. Im vereinfachten Datenmodell werden für das Projekt die folgenden vier Entitäten verwendet:

- Project
- Section
- Inspection
- Observation

```
1 namespace Sewerage.Models
2 {
3     public class Project
4     {
5         public int ProjectId { get; set; }
6         public string Name { get; set; }
7         public string Description { get; set; }
8     }
9 }
```

Abbildung 3 - POCO Implementierung eines Projekts

<sup>8</sup> <http://www.asp.net/mvc/mvc4>

<sup>9</sup> Plain Old CLR Object ([http://en.wikipedia.org/wiki/Plain\\_Old\\_CLR\\_Object](http://en.wikipedia.org/wiki/Plain_Old_CLR_Object))

Ein Project beinhaltet Informationen über ein einzelnes Projekt, das in der Applikation bearbeitet werden kann. Zusätzlich hat ein Project eine oder mehrere Sections. Eine Section repräsentiert einen Kanalabschnitt, für den Beobachtungsdaten erfasst werden. Details zu diesem Abschnitt sowie eine oder mehrere Inspections sind Teile einer Section. Inspections sind einzelne Durchführungen von Inspektionen dieses Abschnitts, falls eine Stelle einer Kanalisation mehrfach untersucht wird. Innerhalb einer Inspection werden Observations erfasst, das heisst, Beobachtungen, die während der Untersuchung gemacht werden. Damit ist das vollständige Datenmodell der umgesetzten Applikation definiert.

**Views.** Views sind diejenigen Komponenten, welche das UI<sup>10</sup> der Applikation darstellen. Die Views werden durch HTML in Kombination mit der Razor ViewEngine<sup>11</sup> aufgebaut. Zur besseren Strukturierung werden einzelne Bereiche in sogenannte PartialViews ausgelagert. Ein Layout View definiert die Grundstruktur der Seiten, die in der gesamten Webapplikation verwendet wird.

```
1 using Sewerage.Resources.Views.Home
2
3 <h2>@IndexStrings.Projects</h2>
4
5 <div class="well">
6     <ul class="nav nav-pills nav-stacked" data-bind="foreach: Projects">
7         <li class="active" data-bind="css: { active: ProjectId == $parent.ChosenProjectId() }, click: $parent.selectProject">
8             <a href="#">
9                 <strong data-bind="text: Name"></strong><br/>
10                <em data-bind="text: Description"></em>
11            </a>
12        </li>
13    </ul>
14</div><!--/.well -->
```

Abbildung 4 - Die Projektübersicht als PartialView

**Controller.** Controller verarbeiten die Benutzerinteraktion, arbeiten mit den Models und bestimmen den View, welcher das UI rendert. Da das Projekt nur wenige verschiedene Views benötigt ist auch die Anzahl Controller klein. Der HomeController kümmert sich um die Hauptseite, die Funktionen der Webapplikation zur Verfügung stellt und um eine kleine statische Seite mit Informationen zum Projekt. Der Registrierungs- und Anmeldeprozess wird durch den AccountController gemanagt. Dieser stellt die genannten Funktionen bereit. Zusätzlich gibt es nur einen weiteren Controller, der jedoch eine Spezialfunktion übernimmt und im Abschnitt Upshot.js besprochen wird.

### 3.2.2 Entity Framework<sup>12</sup> (Code First)

Das Microsoft ADO.NET Entity Framework ist ein Object/Relational Mapping (ORM) Framework, welches dem Entwickler erlaubt, mit relationalen Daten als Domänenobjekten zu arbeiten. Datenabfragen können so mit LINQ auf streng-typisierten Objekten ausgeführt werden. Das Entity Framework (EF) unterstützt den Code First Ansatz. Code First heisst, das Datenmodell kann komplett durch den Code definiert werden und das EF kümmert sich selbst um die Datenbankstruktur. Dafür definiert man einen DbContext. Der DbContext beinhaltet DbSet der Objekte, die in der Datenbank abgespeichert werden sollen. Die Verbindung zur Datenbank kann in der Konfiguration der Applikation hinterlegt werden. Über die Verbindung wird zudem die zu verwendende Datenbanktechnologie definiert.

Das Datenmodell verändert sich normalerweise während der Entwicklung. Deshalb wird über einen DropOrCreateDatabaseIfModelChanges-Initializer festgelegt, dass die Datenbank im Falle einer Änderung komplett gelöscht und neu angelegt wird. Will man nicht nach jeder Änderung eine leere Datenbank haben, kann man die Seed-Methode im Initializer überschreiben und die Datenbank mit

<sup>10</sup> User Interface

<sup>11</sup> <http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx>

<sup>12</sup> <http://www.asp.net/entity-framework>

Testdaten füllen. Dieses Konzept ist während der Entwicklung sehr praktisch, macht aber in einem späteren Produktsystem wenig Sinn. Die neueste Version des EF unterstützt aus diesem Grund neue Migrations<sup>13</sup>. Migrations können verschiedene Versionen des Datenmodells festhalten und die benötigten Veränderungen der Datenbank abspeichern und gegebenenfalls automatisch ausführen.

### 3.2.3 ASP.NET Web API<sup>14</sup>

ASP.NET Web API ist ein Framework mit dem es einfach ist HTTP Services zu erstellen, die von einem breiten Spektrum an Clients – Browsern und mobilen Geräten – konsumiert werden können. Die ASP.NET Web API ist eine ideale Plattform, um „RESTful applications“<sup>15</sup> auf Basis des .NET Frameworks zu programmieren. Als Übertragungsformat kann XML<sup>16</sup> oder JSON<sup>17</sup> verwendet werden. JSON ist aufgrund seines sehr kleinen Overheads besonders geeignet für mobile Applikationen mit langsamen Verbindungen. Die Web API Services können unter anderem aus dem JavaScript Code aufgerufen werden. In der entwickelten Webapplikation wird diese Technologie verwendet, um die Daten nach dem Laden der Seite asynchron vom Server nachzuladen.

```
1 GET http://fhnw.iunodesign.com/api/Sewerage/GetProjects HTTP/1.1
2 Host: fhnw.iunodesign.com
3 Connection: keep-alive
4 X-Requested-With: XMLHttpRequest
5 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.77 Safari/537.1
6 Accept: application/json, text/javascript, */*; q=0.01
7 Referer: http://fhnw.iunodesign.com/
8 Accept-Encoding: gzip, deflate, sdch
9 Accept-Language: en-US,en;q=0.8,de-CH;q=0.6,de;q=0.4
10 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Abbildung 5 - Request-Header der in der Response JSON akzeptiert

Im Projekt umgesetzt wird dies durch einen Controller, der von `DbDataController` abgeleitet wird. In diesem Controller werden die Funktionen definiert, welche als Services aufgerufen werden können. Aufgrund des Request-Headers der Clientanfrage werden Daten im XML oder JSON Format zurückgeliefert. Der gesamte Datenaustausch zwischen Client und Server findet über diesen Controller statt.

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Transfer-Encoding: chunked
5 Content-Type: application/json; charset=utf-8
6 Expires: -1
7 Server: Microsoft-IIS/7.5
8 X-AspNet-Version: 4.0.30319
9 X-Powered-By: ASP.NET
10 Date: Tue, 14 Aug 2012 13:19:51 GMT
11
12 [
13   {"Description": "Project 1 Description", "Name": "Project 1", "ProjectId": 1},
14   {"Description": "Project 2 Description", "Name": "Project 2", "ProjectId": 2},
15   {"Description": "Project 3 Description", "Name": "Project 3", "ProjectId": 3}
16 ]
```

Abbildung 6 - Response mit drei Datensätzen im JSON Format

## 3.3 Client Technologien

Auf dem Server werden Technologien eingesetzt, um Daten in einen Speicher zu schreiben, beziehungsweise daraus zu lesen und an die Clients in einem geeigneten Format zu übermitteln. Zudem stellt das MVC Framework die Struktur der Webapplikation bereit. Clientseitig werden nun die

<sup>13</sup> <http://visualstudiomagazine.com/articles/2012/04/10/entity-framework-code-first-migrations.aspx>

<sup>14</sup> <http://www.asp.net/web-api>

<sup>15</sup> [http://en.wikipedia.org/wiki/Representational\\_state\\_transfer](http://en.wikipedia.org/wiki/Representational_state_transfer)

<sup>16</sup> <http://en.wikipedia.org/wiki/XML>

<sup>17</sup> <http://www.json.org/>

Darstellung und die Logik, um Daten nachzuladen und zu aktualisieren, definiert. Auch für diese Aufgaben werden unterschiedliche Frameworks und Libraries eingesetzt.

### 3.3.1 KnockoutJS<sup>18</sup>

KnockoutJS (KO) ist eine JavaScript Library, um komplexe dynamische UIs mit einem klar definierten Datenmodell und deklarativen Bindings zu erstellen. Es aktualisiert automatisch das UI, wenn sich das darunterliegende Modell ändert und benutzt dazu Two-Way Bindings und Dependency Tracking. Die Templating Engine von KO kann direkt in die Views des MVC Frameworks eingebunden werden.



Das Datenmodell für KnockoutJS wird in JavaScript geschrieben und widerspiegelt das Datenmodell des Servers. Für die automatische Generierung dieses Modells auf dem Client wird ein weiteres Framework eingesetzt: Upshot.js. Eine genauere Beschreibung davon folgt in einem späteren Abschnitt unter Upshot.js.

KO ist ein clientseitiges MVVM<sup>19</sup> Muster. Das in JavaScript programmierte ViewModel beinhaltet alle in den Views benötigten Daten und Funktionen. Objekte des ViewModels, die in den Views automatisch aktualisiert werden sollen, wenn sich deren Daten ändern, müssen als Observables definiert werden. Dadurch werden die Two-Way Bindings festgelegt. Ändern sich nun die Werte eines Objekts im ViewModel, wird dies automatisch im View abgebildet. Ändert sich andererseits ein Wert im View, so wird dieser auch automatisch im ViewModel aktualisiert.

```
58      // define the view model
59      Sewerage.SewerageViewModel = function (options) {
60          var self = this;
61
62          // public array to bind to HTML
63          self.Projects = ko.observableArray();
64          self.Sections = ko.observableArray();
65          self.Inspections = ko.observableArray();
66          self.Observations = ko.observableArray();
```

Abbildung 7 - Ausschnitt des KnockoutJS ViewModels

In den Views können Attribute von HTML-Elementen an Werte des KO-ViewModels angebunden werden, um beispielsweise Texte von Labels oder Listenelementen darzustellen. Diese Technik wird des Weiteren dazu verwendet, die verschiedenen Zustände der Webapplikation zu widerspiegeln. Befindet sich die Applikation in einem Edit-Modus, muss dies nur im ViewModel in einer Funktion gesetzt werden und im entsprechenden View wird ein Teil der HTML-Struktur ein- oder ausgeblendet.

<sup>18</sup> <http://knockoutjs.com/>

<sup>19</sup> Model-View-ViewModel ([http://en.wikipedia.org/wiki/Model\\_View\\_ViewModel](http://en.wikipedia.org/wiki/Model_View_ViewModel))



```

5 <div id="list-inspection" data-bind="if: $root.ChosenSectionId() != null">
6   <ul class="nav nav-tabs">
7     <!-- ko foreach: Inspections -->
8     <li class="dropdown" data-bind="css: {active: InspectionId == $parent.ChosenInspectionId()}">
9       <a class="dropdown-toggle" data-toggle="dropdown" href="#">
10        @IndexStrings.Inspection <span data-bind="text: InspectionId"></span>
11        <b class="caret"></b>
12      </a>
13      <ul class="dropdown-menu">
14        <li>
15          <a href="#" data-bind="click: $parent.selectInspection">
16            <i class="icon-ok"></i> @GlobalStrings.SelectButton
17          </a>
18        </li>
19        [...]
20      </ul>
21    </li>
22    <!-- /ko -->
23  </ul>
24 </div>

```

Abbildung 8 - KnockoutJS Bindings im HTML Code

### 3.3.2 Upshot.js<sup>20</sup>

Das serverseitige Datenmodell muss auf der Clientseite widerspiegelt werden. Hier kommt Upshot.js ins Spiel. Upshot.js ist ein von Microsoft entwickeltes Framework und befindet sich in einem noch sehr frühen Stadium. Es ist in der Lage, die Metadaten des Datenmodells im View einzubinden. Nun können mit Hilfe dieser Metadaten im JavaScript Code Konstruktoren für die Objekte des ViewModels definiert werden. Danach werden für die Datenabfrage RemoteDataSources erstellt, die mit den Objekten und den jeweiligen Methoden des Web API Controllers assoziiert werden. Somit übernimmt Upshot.js clientseitig die Kommunikation zwischen dem JavaScript ViewModel und dem Server. Upshot.js bietet die Möglichkeit, Datenänderungen auf dem Client zwischenspeichern und erst, durch einen Event ausgelöst, auf den Server zu übertragen.

```

98 // data sources
99 var projectsDataSourceOptions = {
100   providerParameters: { url: options.serviceUrl, operationName: "GetProjects" },
101   bufferChanges: true,
102   entityType: projectEntityType,
103   mapping: Sewerage.Project,
104   result: self.Projects
105 };
106 var projectsDataSource = new upshot.RemoteDataSource(projectsDataSourceOptions).refresh();

```

Abbildung 9 - Definition einer Upshot.js RemoteDataSource

### 3.3.3 Bootstrap, from Twitter<sup>21</sup>



Zu diesem Zeitpunkt verfügt das Projekt über Server- und Clientlogik. Auch die Struktur der Views ist per HTML definiert und die Verknüpfung dazwischen mit KnockoutJS und Upshot.js erstellt. Für die noch offenen Designaspekte wird ein Open Source Projekt verwendet, das von Entwicklern und Designer von Twitter<sup>22</sup> betrieben wird: *Bootstrap, from Twitter*. Bootstrap stellt CSS Elemente und JavaScript Funktionen zur Verfügung, um auf einfache Weise anspruchsvolle und plattformübergreifende Designs zu erstellen. Es bietet ein 12-Kolonnen Raster und folgt den Prinzipien des Responsive Design<sup>23</sup>. Dabei wird das Layout mit Hilfe von Media Queries<sup>24</sup> der Displaygröße angepasst.

<sup>20</sup> <http://visualstudiomagazine.com/articles/2012/06/01/the-upshot-its-a-knockout.aspx>

<sup>21</sup> <http://twitter.github.com/bootstrap/>

<sup>22</sup> <http://twitter.com/>

<sup>23</sup> [http://de.wikipedia.org/wiki/Responsive\\_Design](http://de.wikipedia.org/wiki/Responsive_Design)



Sewerage

About Sewerage

student

## Projects

Project 1  
Project 1 Description

Project 2  
Project 2 Description

Project 3  
Project 3 Description

Project 4  
Project 4 Description

Project 5  
Project 5 Description

Project 6  
Project 6 Description

Project 7  
Project 7 Description

Project 8  
Project 8 Description

Project 9  
Project 9 Description

Project 10  
Project 10 Description

## Sections

Number	Length	City	Street	+
1	37.3	Murten	Insweg	
2	54.3	Murten	Insweg	
3	84	Murten	Insweg	

Inspection 1 Inspection 2 +

## Observations

Position	Tag	Description	Video Offset	+
0	BCDXP	Observation 1	0	
10	BAFAE	Observation 2	10	
20	BBCC	Observation 3	20	

© 2012 by University of Applied Sciences Northern Switzerland

Sewerage

Sewerage

Sewerage

## Projects

Project 1  
Project 1 Description

Project 2  
Project 2 Description

Project 3  
Project 3 Description

Project 4  
Project 4 Description

Project 5  
Project 5 Description

Project 6  
Project 6 Description

Project 7  
Project 7 Description

Project 8  
Project 8 Description

Project 9  
Project 9 Description

Project 10  
Project 10 Description

## Sections

Number	Length	City	Street	+
1	37.3	Murten	Insweg	
2	54.3	Murten	Insweg	
3	84	Murten	Insweg	

Inspection 1 Inspection 2 +

## Observations

Position	Tag	Description	Video Offset	+
0	BCDXP	Observation 1	0	
10	BAFAE	Observation 2	10	
20	BBCC	Observation 3	20	

© 2012 by University of Applied Sciences Northern Switzerland

© 2012 by University of Applied Sciences Northern Switzerland

Abbildung 10 - Responsive Design für Desktop (oben), Mobile (u. links) und Tablets (u. rechts)

Das Framework selbst basiert auf dem CSS Framework LESS<sup>25</sup>. LESS ist eine Erweiterung von CSS um ein dynamisches Verhalten wie Variablen, Verschachtelungen und Funktionen. Aus den LESS Dateien werden durch einen Compiler wiederum CSS Dateien erstellt.

<sup>24</sup> <http://mediaqueri.es/>

<sup>25</sup> <http://lesscss.org/>

```

8  .navbar {
9    // Fix for IE7's bad z-indexing so dropdowns don't appear below content that follows the navbar
10   *position: relative;
11   *z-index: 2;
12
13   overflow: visible;
14   margin-bottom: @baselineHeight;
15 }
16
17 // Gradient is applied to it's own element because overflow visible is not honored by IE when filter is present
18 .navbar-inner {
19   padding-left: 20px;
20   padding-right: 20px;
21   #gradient > .vertical(@navbarBackgroundHighlight, @navbarBackground);
22   .border-radius(4px);
23   @shadow: 0 1px 3px rgba(0,0,0,.25), inset 0 -1px 0 rgba(0,0,0,.1);
24   .box-shadow(@shadow);
25 }

```

Abbildung 11 - Ausschnitt des LESS Codes für die Navigation

Bootstrap bietet auch eine Vielzahl an jQuery<sup>26</sup> Plug-Ins. Damit können Dropdowns, Tooltips oder Dialogfenster einfach in die Webapplikation eingebunden werden.

### 3.4 Ergänzende Technologien

Neben den Technologien, die klar der Server- oder Clientseite zugeordnet werden können, gibt es weitere, die auf beiden Seiten Aufgaben übernehmen.

#### 3.4.1 Validierung

Die als POCO implementierten Models können mit Validierungsregel ergänzt werden. Mittels Attributen werden sie auf die Eigenschaften der Models angewendet. Durch das Prinzip der Unobtrusive Validations müssen die Regeln nur an einem Ort festgelegt werden und können danach im Server- und im Clientcode überprüft werden. Die doppelte Prüfung stellt immer einen Sicherheitsaspekt dar. Es kann in der Webprogrammierung nicht davon ausgegangen werden, dass die an den Server geschickten Daten nur über die Webmaske eingegeben werden. Man will jedoch dem Benutzer eine schnelle Rückmeldung geben, ob seine Daten im erwarteten Format sind ohne dabei die Zeit eines Roundtrips zum Server in Kauf zu nehmen.

Die Unobtrusive Validations werden in der Konfiguration der Webapplikation aktiviert. Dadurch werden die einzelnen HTML Elemente mit den HTML5 data-\* Attributen ergänzt. Diese wiederum können vom jQuery Unobtrusive Validations Plug-In interpretiert werden. Mit einem kleinen Fix erreicht man zusätzlich auf sehr einfache Weise eine optische Repräsentation im UI durch die Kombination von jQuery und Bootstrap.

<sup>26</sup> <http://jquery.com/>

```

6  <!-- Fix for jQuery Validation -->
7  <script type="text/javascript">
8      var page = function () {
9          //Update that validator
10         $.validator.setDefaults({
11             highlight: function (element) {
12                 $(element).closest(".control-group").addClass("error");
13             },
14             unhighlight: function (element) {
15                 $(element).closest(".control-group").removeClass("error");
16             }
17         });
18     } ();
19 </script>

```

Abbildung 12 - jQuery Validation Fix

### 3.4.2 Globalisierung

Bei einer Software, die weltweit eingesetzt werden soll, spielt die Mehrsprachigkeit eine entscheidende Rolle. Die Implementierung kann auf unterschiedliche Weise erfolgen. Das Projekt verwendet dafür die vom .NET Framework bereitgestellten Resource Dateien. In der Konfigurationsdatei wird die automatische Erkennung der vom Browser eingestellten Sprache aktiviert. Danach werden eine allgemein gültige Sprachdatei und eine oder mehrere sprachspezifische Dateien angelegt. Das .NET Framework wählt automatisch die am besten geeignete Sprachdatei aus. Damit können Überschriften, ganze Texte oder auch Validierungsnachrichten in mehreren Sprachen ausgegeben werden. Die Übersetzung der extern abgelegten XML Dateien kann so auch von externen Mitarbeitern durchgeführt werden.

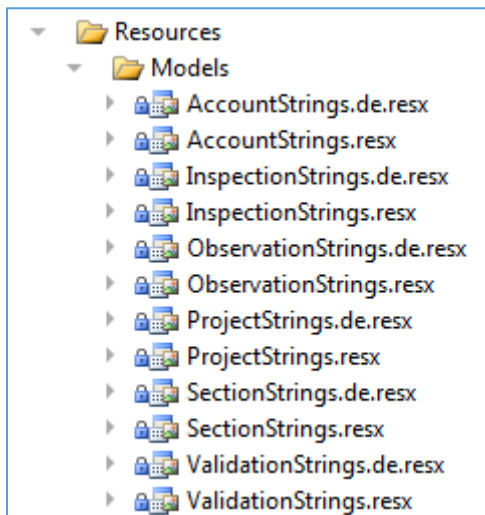


Abbildung 13 - Resource Dateien in zwei Sprachen

### 3.5 Videos im Web<sup>27</sup>

Bei den Kanalinspektionen spielt neben der Erfassung von Daten das dazugehörige Video die grösste Rolle. Da die Videodateien unter Umständen sehr gross sein können (>>1GB) wurde dieses Thema im vorangehenden Semester in einem separaten Projekt angegangen. Dabei galt es die Smooth Streaming<sup>28</sup> Technologie von Microsoft zu evaluieren. Um ein Video mit der Smooth Streaming Technologie im Web bereitzustellen, muss das Video speziell enkodiert werden. Ein Teil des Vorgängerprojektes war es, diesen Prozess zu automatisieren. Für die Übertragung der grossen Videos auf den Server wurde eine eigenständige Applikation geschrieben. Diese überträgt die Dateien per FTP-Protokoll. Die vollständig übertragenen Videodateien werden von einer Webapplikation erkannt und der Enkodierungsprozess wird gestartet.



Job Monitor   Finished   Failed

### Smooth Streaming Transcoder.

**Job Monitor.** Active Encodes

Id	Submitted	Asset	Status	Progress	Tasks
<a href="#">78b280ed-74dc-497b-9f26-c55e396544ef</a>	24.01.2012 11:21:38	Kanalinspektion_klein.mpg	Running	<div></div>	1/1

© 2012 Fachhochschule Nordwestschweiz

Abbildung 14 – Webapplikation für die automatische Enkodierung aus dem Vorgängerprojekt

Die Wiedergabe auf dem Client erfordert zudem eine spezielle Komponente auf dem Webserver (IIS Media Services<sup>29</sup>) und einen Silverlight<sup>30</sup> Player im Browser. Um Videos über diesen Player abzuspielen muss im Browser das Silverlight Plug-In installiert sein. Das stellt aber auf mobilen Geräten ein Problem dar, weil bis jetzt kein mobiles Gerät Silverlight unterstützt<sup>31</sup>.

Vor HTML5 gab es keinen standardisierten Weg, um Videos im Browser wiederzugeben. Praktisch alle Videos im Web wurden von Third-Party Plug-Ins wiedergegeben – manchmal QuickTime, manchmal RealPlayer oder manchmal auch Flash. Diese Plug-Ins integrieren sich teilweise so gut in den Browsern, dass der Benutzer dies nicht bemerkt. Aber nur solange, bis man das Video versucht auf einer Plattform zu schauen, die dieses Plug-In nicht unterstützt.

HTML5 definiert einen Standard, Videos in eine Webseite mittels des <video> Elements einzubinden. Die Unterstützung dieses <video> Elements ist noch in der Entwicklung. Das heisst, es funktioniert noch nicht überall, aber es gibt wie immer in der Programmierung Umwege. Doch die Unterstützung des <video> Elements selbst ist nur ein kleiner Teil der ganzen Geschichte. Um HTML5 Videos zu verstehen, muss man zuerst ein bisschen mehr über Videos selbst verstehen.

<sup>27</sup> <http://fortuito.us/diveintohtml5/video.html>

<sup>28</sup> <http://www.iis.net/download/SmoothStreaming>

<sup>29</sup> <http://www.iis.net/media>

<sup>30</sup> <http://www.silverlight.net/>

<sup>31</sup> Nicht einmal das Windows Phone 7, welches zu einem grossen Teil auf Silverlight basiert.

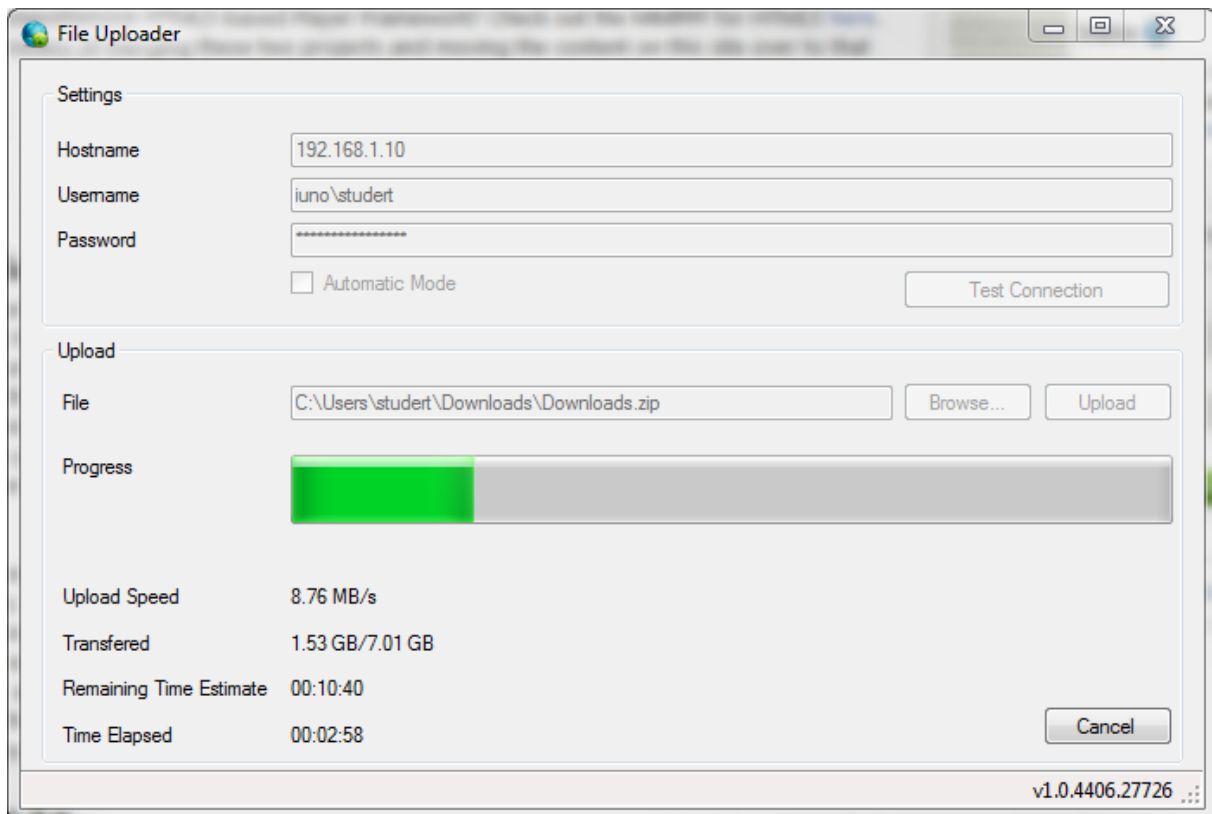


Abbildung 15 - FTP Client für die Übertragung grosser Videodateien

### 3.5.1 Video Container

Man stellt sich unter Videodateien oft .AVI Dateien oder .MP4 Dateien vor. In Wirklichkeit sind .AVI und .MP4 nur Containerformate. Genau wie eine .ZIP Datei, die unterschiedlichste Dateien enthalten kann. Video Container definieren nur, wie Informationen darin abgespeichert werden, nicht was für Informationen. Eine Videodatei beinhaltet normalerweise mehrere Tracks – ein Videotrack (ohne Audio), plus einen oder mehrere Audiotracks (ohne Video).

### 3.5.2 Video Codecs

Wenn man ein Video schaut, macht der Videoplayer mindestens drei Dinge gleichzeitig:

1. Das Containerformat interpretieren, um herauszufinden, welche Video- und Audiotracks verfügbar sind und wie sie innerhalb der Datei abgespeichert sind.
2. Den Videostream dekodieren und eine Bilderserie am Bildschirm anzeigen.
3. Den Audiostream dekodieren und den Sound an die Lautsprecher schicken.

Ein Video Codec ist ein Algorithmus, mit welchem ein Videostream enkodiert wird. Der Videoplayer dekodiert den Videostream anhand des Video Codecs und zeigt dann eine Bilderserie, auch Frames genannt, am Bildschirm an. Es gibt eine Vielzahl an Video Codecs.

#### Codec

Als **Codec** (Kunstwort aus englisch **coder** und **decoder**) bezeichnet man ein Verfahren, das Daten oder Signale digital kodiert und dekodiert. Beim direkten Umwandeln von einem Format in ein anderes (bspw. MPEG-2 zu MPEG-4 oder MP3 zu WMA) spricht man auch von Transkodierung.

Quelle: <http://de.wikipedia.org/wiki/Codec>

Unter den bekanntesten Codecs sind H.264, MPEG (1-4), WMV und DivX. Ein Hauptproblem beim Verwenden des .MP4 Formats ist, dass dessen Video Codec (H.264) lizenzpflichtig ist. Das Lizenzmodell ist zudem sehr komplex<sup>32</sup>.

### 3.5.3 Was funktioniert im Web

HTML5 beinhaltet ein <video> Element, um Videos in eine Webseite zu integrieren. Es gibt keine Einschränkungen beim Video Codec, Audio Codec oder Containerformat, die für Videos eingesetzt werden können. Ein <video> Element kann mehrere Videodateien verlinken und der Browser wird das erste Video auswählen, dass er abspielen kann. Dabei gilt es zu beachten welche Browser welche Container und Codecs unterstützen. Konkret heisst das, es gibt keine einzelne Kombination von Containern und Codecs, die in allen gängigen HTML5 Browsern funktioniert. Sehr wahrscheinlich wird sich das in Zukunft ändern. Zurzeit wird man jedoch jedes Video mehrmals enkodieren müssen.

#### Video Codec Support in verfügbaren Browsern

Codec/Container	IE	Firefox	Safari	Chrome	Opera	iPhone	Android
Theora+Vorbis+Ogg	-	3.5+	*	5.0+	10.5+	-	-
H.264+AAC+MP4	-	-	3.0+	5.0-?*	-	3.0+	2.0+
WebM	-	-	*	6.0+	10.6+	-	-

\*Safari kann alles abspielen, was QuickTime abspielen kann. QuickTime hat H.264/AAC/MP4 Support bereits vorinstalliert. Third-Party Plug-Ins können Theora und WebM Unterstützung hinzufügen.

\*\*Google Chrome wird die H.264 Unterstützung bald einstellen.<sup>33</sup>

Abbildung 16 - Quelle: <http://fortuito.us/diveintohtml5/video.html>

Folgende Kombination hat sich als maximal kompatibel herausgestellt:

1. Eine Version, die VP8 Video und Vorbis Audio in einem WebM Container benutzt.
2. Eine Version, die H.264 Video und AAC Audio in einem MP4 Container benutzt.
3. Eine Version, die Theora Video und Vorbis Audio in einem Ogg Container benutzt.
4. Verknüpfe alle drei Videodateien mit einem <video> Element und stelle ein Plug-In basiertes Fallback Szenario mit Flash oder Silverlight bereit.

### 3.5.4 Microsoft Media Platform Player Framework<sup>34</sup>

Trotz der Möglichkeiten des HTML5 <video> Elements, soll die bereits erarbeitete Silverlight Variante der primäre Videoplayer bleiben. Mit Hilfe des Player Frameworks der Microsoft Media Plattform kann dies realisiert werden. Es handelt sich dabei um ein robustes Videoplayer Framework für Windows 8, HTML5, Silverlight, Windows Phone und andere Applikationsplattformen. Der Video Player ist Open Source und wird von Microsoft weiterentwickelt. Um den Player in der Webapplikation anzusteuern, benötigt man einerseits das HTML5 Player Framework bestehend aus JavaScript und CSS Dateien und andererseits den Player als .xap<sup>35</sup> Datei.



<sup>32</sup> <http://diveintohtml5.info/video.html#licensing>

<sup>33</sup> <http://blog.chromium.org/2011/01/more-about-chrome-html-video-codec.html>

<sup>34</sup> <http://playerframework.codeplex.com/>

<sup>35</sup> <http://debugmode.net/2011/03/05/what-is-xap-file-in-silverlight/>

Im HTML kann man für den Player unterschiedliche Videoquellen angeben. Auch die Reihenfolge des Fallbacks kann einfach definiert werden, sprich ob der Silverlight Player als erste oder letzte Variante versucht werden soll. Jedoch sieht das Framework nicht vor, dass die Quellen dynamisch geändert werden können. Deshalb muss für das Projekt ein Umweg programmiert werden, welcher für jedes neue Video den Markup des Players aus dem DOM<sup>36</sup> entfernt und mit den neuen Quellen wieder einfügt.

### 3.6 Entwicklung

Als Entwicklungsumgebung wurde hauptsächlich Visual Studio 2010<sup>37</sup> eingesetzt. Visual Studio bietet für die .NET Entwicklung die ausgereiftesten Tools. Ein MVC Projekt kann aus einem Template erstellt werden und generiert die benötigte Ordnerstruktur mit einem Minimum an Dateien für eine lauffähige Webapplikation. Zusätzlich benötigte Assemblies werden mit dem Paketmanager von Visual Studio ins Projekt geladen und referenziert.

#### 3.6.1 NuGet<sup>38</sup>

NuGet ist eine Visual Studio Erweiterung, die es einfach macht, Libraries und Tools zu einem Visual Studio Projekt hinzuzufügen, zu entfernen und zu aktualisieren. Wenn man eine Library oder ein Tool entwickelt und anderen Entwicklern zur Verfügung stellen will, dann erstellt man ein NuGet Paket und lädt es in ein NuGet Repository. Wenn man eine Library oder ein Tool eines anderen Entwicklers verwenden will, kann man dieses direkt aus einem NuGet Repository beziehen und in das Visual Studio Projekt installieren.



Wenn man ein Paket installiert, kopiert NuGet die Dateien in das Projekt und macht automatisch alle benötigten Veränderungen, wie Referenzen hinzufügen oder die Konfigurationsdateien anpassen. Falls man sich dazu entscheidet, ein Paket wieder zu entfernen, macht NuGet

die Änderungen rückgängig und hinterlässt keine verwahrlosten Dateien.

NuGet führt eine Liste aller installierten Pakete und kann während des Kompilierens des Projekts fehlende Pakete nachladen. Dies hat den Vorteil, dass die Dateien der installierten Pakete nicht in die Versionskontrolle eingecheckt werden müssen.

#### 3.6.2 Chrome Entwickler Tools<sup>39</sup>

Ein grosser Teil der programmierten Logik wird als JavaScript direkt im Browser ausgeführt. Um diesen Code während der Entwicklung zu debuggen, benötigt ein Entwickler ausgereifte Entwickler Tools. Google Chrome hat sich dabei als äusserst hilfreich herausgestellt. Sämtliche JavaScript Dateien, die durch die Webseite geladen werden, können mit Breakpoints versehen werden. Dadurch wird die Ausführung des JavaScript Codes an den entsprechenden Stellen unterbrochen und die Werte der einzelnen Variablen können untersucht werden.

### Open Source Software

*(kurz **OSS**) steht unter einer von der Open Source Initiative (OSI) anerkannten Lizenz. Diese Organisation stützt sich bei ihrer Bewertung auf die Kriterien der Open Source Definition, die weit über die Verfügbarkeit des Quelltexts hinausgeht. Sie ist fast deckungsgleich mit der Definition freier Software.*

Quelle: [http://de.wikipedia.org/wiki/Open\\_Source](http://de.wikipedia.org/wiki/Open_Source)

<sup>36</sup> <http://www.w3schools.com/html/default.asp>

<sup>37</sup> <http://www.microsoft.com/visualstudio/en-us>

<sup>38</sup> <http://docs.nuget.org/docs/start-here/overview>

<sup>39</sup> <https://developers.google.com/chrome-developer-tools/docs/overview>







<div> <span>Elements</span> <span>Resources</span> <span>Network</span> <span>Sources</span> <span>Timeline</span> <span>Profiles</span> <span>Audits</span> <span>Console</span> <span>PageSpeed</span> </div>							
Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	
 <b>GetProjects</b> /api/Sewerage	GET	200 OK	application/json	jquery-1.7.2.min.js:4 Script	1019B 734B	226ms 14ms	
 <b>GetProjectSections</b> /api/Sewerage	GET	200 OK	application/json	jquery-1.7.2.min.js:4 Script	587B 314B	28ms 22ms	
 <b>GetSectionInspections</b> /api/Sewerage	GET	200 OK	application/json	jquery-1.7.2.min.js:4 Script	654B 381B	22ms 16ms	
 <b>GetInspectionObservations</b> /api/Sewerage	GET	200 OK	application/json	jquery-1.7.2.min.js:4 Script	676B 403B	79ms 17ms	

Abbildung 17 - Google Entwickler Tools mit XHR Aufrufen

Zudem vereinfacht es die Untersuchung des generierten HTML und dessen Verknüpfung mit den jeweiligen CSS Elementen. Spannend wird es aber vor allem dann, wenn man genau beobachten kann, was auf Netzwerkebene zwischen dem Server und dem Client ausgetauscht wird. Die Netzwerkaktivitäten können zusätzlich gefiltert werden, um zum Beispiel nur die XHR<sup>40</sup> Aufrufe genauer zu betrachten. Damit kann man sicherstellen, dass wirklich nur ein Minimum an Daten zum gewünschten Zeitpunkt zwischen Server und Client ausgetauscht werden.

### 3.6.3 Fiddler<sup>41</sup>

Fiddler ist ein Web Debugging Proxy, der den kompletten HTTP(S) Verkehr zwischen dem Computer und dem Internet dokumentiert. Mit Fiddler können auch eigene Requests generiert werden. Die ASP.NET Web API kann dadurch einfach getestet werden und es hilft dem Entwickler zu verstehen, wie die Daten genau aussehen, welche zwischen der Web API und dem Client ausgetauscht werden.

#### XHR

*XMLHttpRequest (XHR) ist eine API zum Transfer von beliebigen Daten über das Protokoll HTTP. Dabei können sämtliche HTTP-Anfragemethoden (unter anderem GET, POST, HEAD, PUT) verwendet werden.*

Quelle: <http://de.wikipedia.org/wiki/XMLHttpRequest>

### 3.6.4 ELMAH<sup>42</sup>

Während der Entwicklung schleichen sich immer wieder Fehler ein. Nicht alle Fehler werden erwartet und können entsprechend behandelt werden. Aus diesem Grund wird via NuGet das ELMAH Paket in die Webapplikation installiert. ELMAH ist eine applikationsweite Fehlerbehandlungskomponente. Sie kann dynamisch einer ASP.NET Webapplikation hinzugefügt werden, ohne dass diese neu kompiliert werden muss. Sobald ELMAH mit einer Webapplikation verknüpft ist, zeichnet sie alle unbehandelten Fehler auf.

<sup>40</sup> XMLHttpRequest (<http://en.wikipedia.org/wiki/XMLHttpRequest>)

<sup>41</sup> <http://www.fiddler2.com/fiddler2/>

<sup>42</sup> Error Logging Modules and Handlers (<http://code.google.com/p/elmah/>)



## Error Log for ROOT on SV03

[RSS FEED](#) | [RSS DIGEST](#) | [DOWNLOAD LOG](#) | [HELP](#) | [ABOUT](#)

Errors 1 to 10 of total 90 (page 1 of 9). Start with [10](#), [15](#), [20](#), [25](#), [30](#), [50](#) or [100](#) errors per page.

Host	Code	Type	Error	User	Date	Time
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/9/2012	5:01 PM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/9/2012	11:06 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/9/2012	5:56 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/9/2012	1:57 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/8/2012	6:30 PM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/8/2012	7:35 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/8/2012	5:42 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/8/2012	5:42 AM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/7/2012	5:57 PM
SV03	404	Http	The controller for path '/robots.txt' was not found or does not implement IController. <a href="#">Details...</a>		8/7/2012	1:04 PM

[Next errors](#)

Powered by [ELMAH](#), version 1.2.14706.955. Copyright (c) 2004, Atif Aziz. All rights reserved. Licensed under [Apache License, Version 2.0](#). Server date is Thursday, 09 August 2012. Server time is 20:42:50. All dates and times displayed are in the W. Europe Daylight Time zone. This log is provided by the SQL Server Compact Error Log.

Abbildung 18 - Fehlerlog von ELMAH

### 3.7 Entwicklungsumgebung

Bei der Softwareentwicklung ist die Versionskontrolle ein wichtiges Thema. Welches System dafür verwendet wird, spielt keine entscheidende Rolle. Diese Arbeit wird auf GitHub verwaltet und verwendet deshalb Git als Versionskontrolle.

#### 3.7.1 Git<sup>43</sup> und GitHub<sup>44</sup>

Eine Versionskontrolle ist ein System, das Änderungen an einer Datei oder einer Kollektion von Dateien zeitlich aufzeichnet, um so auf frühere Versionen zurückzugreifen. Es erlaubt daher, Dateien in einen vorangehenden Zustand zurückzubringen und Veränderungen über die Zeit zu vergleichen. Ein Versionskontrollsystem (VCS<sup>45</sup>) zu verwenden, bedeutet einfaches Wiederherstellen eines früheren Zustandes der Entwicklung.

Git ist im Gegensatz zu Subversion<sup>46</sup> ein verteiltes VCS. In einem verteilten Versionskontrollsystem (DVCS<sup>47</sup>) holt sich der Entwickler nicht nur den letzten Snapshot aller Dateien auf sein lokales System, sondern das komplette Repository wird auf die Festplatte gespiegelt. Somit ist jeder Checkout ein vollständiges Backup aller Daten.

<sup>43</sup> <http://git-scm.com/>

<sup>44</sup> <https://github.com/>

<sup>45</sup> Version Control System

<sup>46</sup> <http://subversion.apache.org/>

<sup>47</sup> Distributed Version Control System

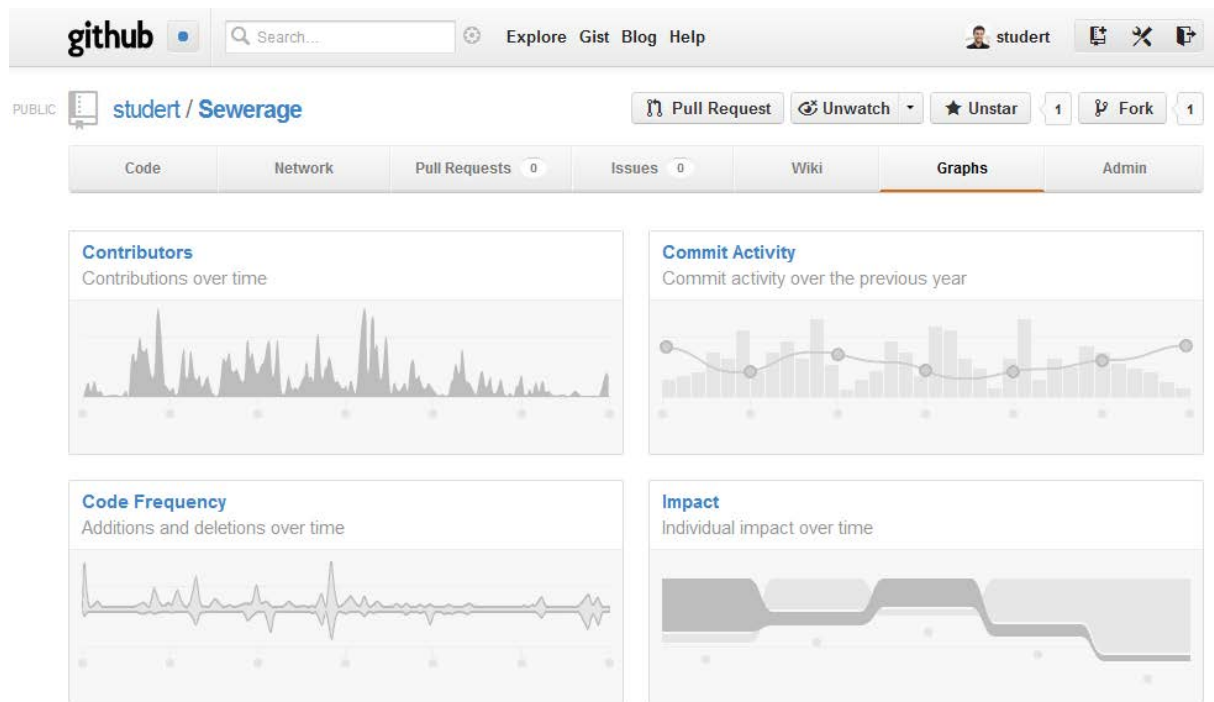


Abbildung 19 - Übersicht der Aktivitäten im GitHub Repository

Es ist durchaus möglich, sich ein solches DVCS selbst einzurichten. Jedoch haben sich in der Open Source Entwicklergemeinschaft verschiedene populäre Onlinedienste, wie zum Beispiel GitHub, durchgesetzt.

### 3.7.2 Continuous Integration

Den Quellcode in einem VCS zu verwalten hat einen weiteren wichtigen Vorteil. Eine Continuous Integration Umgebung kann bei jedem Commit in das VCS die neueste Version daraus beziehen, diese kompilieren und weiterverarbeiten. Für dieses Projekt wurde ein TeamCity<sup>48</sup> Build Server eingerichtet und mit dem GitHub Repository verknüpft. TeamCity reagiert auf jeden Commit und kompiliert die neueste Version des Programmcodes. Dadurch wird sehr schnell sichtbar, wenn sich ein Fehler in den Code eingeschlichen hat. Nur weil eine Applikation auf dem lokalen Test- oder Entwicklungssystem einwandfrei funktioniert, bedeutet das nicht, dass es zum Beispiel auch auf einem Produktivsystem läuft.

<sup>48</sup> <http://www.jetbrains.com/teamcity/>

[Projects](#)
[My Changes](#)
[Agents \(1\)](#)
[Build Queue \(0\)](#)

Administrator
Administration

Sewerage
Sewerage master branch
Run
Actions
Edit Configuration Settings

Overview
History
Change Log
Statistics
Compatible Agents (1)
Pending Changes (0)
Settings

Pending changes

No pending changes

Current status

Idle

Recent history

Show canceled builds and builds that failed to start

#	Results	Artifacts	Changes	Started	Duration	Agent	Tags	
#80	Success	None	student (1)	25 Jul 12 19:15	10s	SV05	None	Pin
#79	Compilation failed	None	student (1)	25 Jul 12 19:08	18s	SV05	None	Pin
#78	Success	None	student (1)	25 Jul 12 11:07	10s	SV05	None	Pin
#77	Success	None	No changes	25 Jul 12 10:48	15s	SV05	None	Pin
#76	Success	None	student (1)	25 Jul 12 10:42	19s	SV05	None	Pin
#75	Compilation failed	None	student (1)	25 Jul 12 10:36	20s	SV05	None	Pin
#74	Compilation failed	None	No changes	25 Jul 12 10:15	4s	SV05	None	Pin
#73	Compilation failed	None	student (1)	24 Jul 12 18:27	5s	SV05	None	Pin
#72	Compilation failed	None	student (2)	24 Jul 12 17:02	1m:15s	SV05	None	Pin
#71	Success	None	student (1)	13 May 12 16:35	14s	SV05	None	Pin

Showing 10 builds, see entire history

Abbildung 20 - TeamCity Build Server Übersicht

### 3.7.3 Continuous Deployment

Um sicherzustellen, dass die Webapplikation auch auf einem von der Entwicklungsumgebung unabhängigen System funktioniert, wurde die Continuous Integration um ein Continuous Deployment ergänzt. Dafür wurde ein Produktivsystem mit einem eigenen Webserver, einer eigenen Datenbank und einer öffentlichen URL<sup>49</sup> eingerichtet. Jedes Mal wenn TeamCity die neuste Version der Webapplikation erfolgreich kompiliert, wird diese mittels WebDeploy<sup>50</sup> auf dem Produktivsystem veröffentlicht. Dabei werden bestimmte Konfigurationen, wie die Datenbankverbindung, verändert. Für diesen Vorgang wurden Web.Config Transformations<sup>51</sup> eingesetzt.

```

1  <?xml version="1.0"?>
2  <configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
3    <connectionStrings>
4      <add xdt:Transform="SetAttributes" xdt:Locator="Match(name)" name="AppDbContext"
5        connectionString="Data Source=SV02;Initial Catalog=Sewerage;Integrated Security=True" />
6      <add xdt:Transform="SetAttributes" xdt:Locator="Match(name)" name="MembershipDatabase"
7        connectionString="Data Source=SV02;Initial Catalog=SewerageMembership;Integrated Security=True" />
8    </connectionStrings>
9    <system.web>
10     <compilation xdt:Transform="RemoveAttributes(debug)" />
11   </system.web>
12 </configuration>

```

Abbildung 21 - Web.Config Transformations für das Produktivsystem

## 3.8 Endprodukt

Die bei der Entwicklung entstandene Single Page Application folgt dem Prinzip des Responsive Design. Die Hauptnavigation befindet sich am oberen Rand der Webseite. Auf der linken Seite ist die Liste aller verfügbaren Projects. Das jeweils ausgewählte ist farblich hervorgehoben. Wird ein Project ausgewählt, werden im Hintergrund die Daten der dazugehörigen Sections, Inspections und Observations nachgeladen. Zudem wird der vom Browser unterstützte Videoplayer initialisiert und

<sup>49</sup> Uniform Resource Locator

<sup>50</sup> <http://www.iis.net/download/WebDeploy>

<sup>51</sup> <http://www.asp.net/mvc/tutorials/deployment/deployment-to-a-hosting-provider/Deployment-to-a-Hosting-Provider-Web-Config-File-Transformations-3-of-12>

das Video gestartet. Einzelne Observations haben eine Zeitmarke, an die das Video bei deren Auswahl springt.

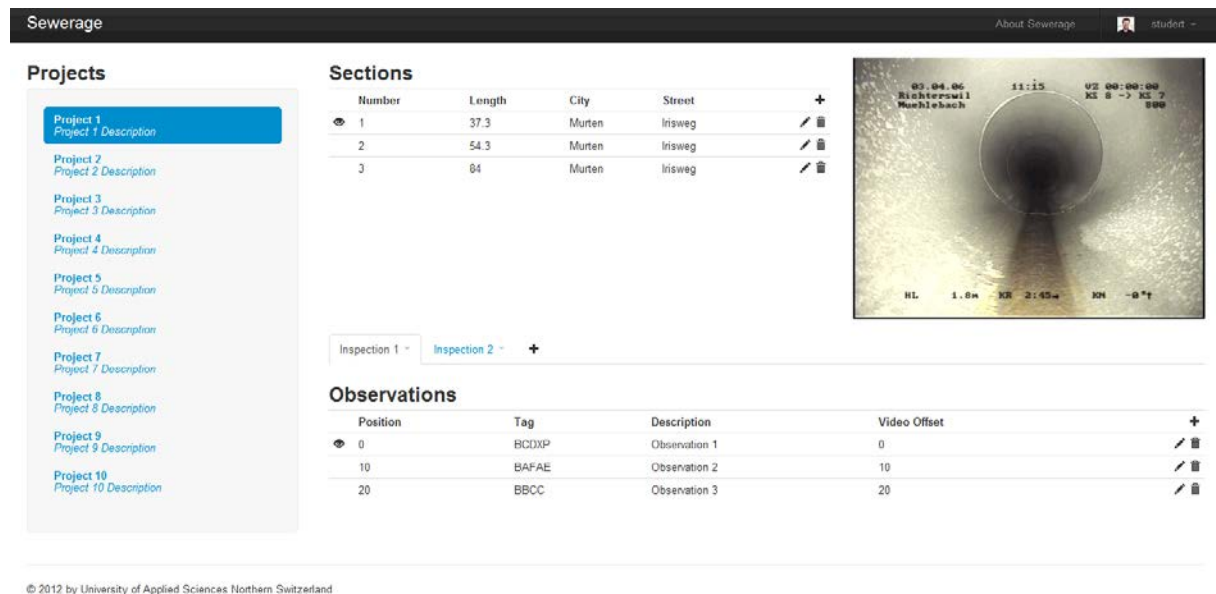


Abbildung 22 - Die aus dem Projekt entstandene Webapplikation

Sections, Inseptions und Observations können mit einem Editor bearbeitet oder neu erstellt werden. Die dafür benötigten Links sind direkt in die Darstellung der Daten integriert. Die eingegebenen Daten werden für eine schnelle Rückmeldung bereits clientseitig validiert. Zusätzlich werden sie aus Sicherheitsgründen auf dem Server erneut überprüft.

Die Benutzerdaten werden in der Navigation oben rechts angezeigt. Verfügt die angegebene E-Mail Adresse über einen Gravatar<sup>52</sup>, wird zusätzlich ein Profilbild angezeigt. Der Benutzer hat auch die Möglichkeit, sein Passwort zu ändern.

### 3.8.1 Infrastruktur

Die Webapplikation wird auf einem Windows Server 2008 R2 gehostet. Dafür wird der IIS als Webserver eingesetzt. Auf dem Webserver sind WebDeploy und die IIS Media Services installiert. Ersteres für das Continuous Deployment aus dem Buildserver, letzteres für das Smooth Streaming der für den Silverlightplayer enkodierten Videos. Die Videos werden auf dem Server selbst abgespeichert und über einen Virtual Directory in die Webapplikation integriert. Möglich wäre auch, die Videos auf einem dedizierten Server bereit zu stellen.

Als Datenspeicher wird ein Microsoft SQL Server 2008 eingesetzt. Grundsätzlich kann aber ein beliebiger Datenbankserver, der von Entity Framework unterstützt wird, verwendet werden. Die Verbindung zum SQL Server wird in der Konfigurationsdatei der Webapplikation definiert. Existiert die Datenbankstruktur noch nicht, wird diese automatisch erzeugt.

Alle gängigen Browser, sowohl auf Desktopcomputern wie auch auf mobilen Geräten, werden von der Applikation unterstützt. Ohne das Silverlight Plug-In kann jedoch nicht von der Smooth Streaming Technologie profitiert werden. Das Layout passt sich automatisch der verwendeten Bildschirmdimension an.

<sup>52</sup> <http://en.gravatar.com/>

### 3.8.2 Informationen

Was	Wo
Webapplikation	<a href="http://fhnw.iunodesign.com">http://fhnw.iunodesign.com</a>
Dokumentation	<a href="http://fhnw.iunodesign.com/documentation">http://fhnw.iunodesign.com/documentation</a>
Quellcode	<a href="https://github.com/studert/Sewerage">https://github.com/studert/Sewerage</a>
Bachelor Thesis (Quellcode & Dokumentation)	<a href="http://sdrv.ms/RDtNKy">http://sdrv.ms/RDtNKy</a>

Die angegebenen Links sind mindestens bis drei Monate nach Abgabe dieser Arbeit gültig.

## 4 Schluss

In den letzten sieben Jahren, in denen ich mich mit der .NET Technologie beschäftigt habe, war der Fokus stets auf der serverseitigen Programmierung. Dieses Projekt gab mir die Möglichkeit, mich stärker mit der Clientprogrammierung auseinander zu setzen. Dabei gibt es eine so grosse Vielfalt an Möglichkeiten, dass es manchmal nicht einfach ist, sich darin zurecht zu finden. Neue Frameworks entstehen fast täglich.

Die Möglichkeit täglich neue Dinge zu entdecken, hat sich als grosse Herausforderung dargestellt, gab mir aber auch neue Perspektiven in der Softwareentwicklung. Der Umgang mit sehr grossen Videodateien hat mir auch Grenzen des heutigen Internets aufgezeigt.

Mit der aus der Bachelor Thesis entstandenen Webapplikation bin ich sehr zufrieden. Es scheint mir gelungen zu sein, die beiden Hauptziele der Arbeit zu behandeln und eine mögliche Implementierung aufzuzeigen. Die Webentwicklung wird in Zukunft, aufgrund der immer grösser werdenden Anzahl mobiler Geräte mit Internetanbindung, eine immer bedeutendere Rolle einnehmen und dabei spielt nicht mehr nur die Technologie eine Rolle. Das Benutzererlebnis erhält einen immer grösser werdenden Stellenwert. Man will heute nicht mehr einfach eine Webseite besuchen, man will sie erleben.

Die Bachelor Thesis zusammen mit dem Projekt aus dem letzten Semester legt den Grundstein für eine solide und moderne Webapplikation. Die automatische Enkodierung der Videos muss zwar noch um weitere Formate ergänzt und die Webapplikation selbst zu einem Produktivsystem weiterentwickelt werden. Dieses Projekt konnte jedoch Möglichkeiten dafür aufzeigen.

### 4.1 Zukunftsblicke

Die Videoenkodierung und die Verfügbarkeit der Webapplikation stellen in einer späteren produktiven Umgebung grosse Ansprüche an die Rechenleistung. Dabei muss man sich die Frage stellen, ob man diese Rechenleistung als Firma selbst zur Verfügung stellen kann, oder ob es eventuell mehr Sinn macht, diese einzukaufen. In dieser Hinsicht hat sich das Angebot von Windows Azure<sup>53</sup> in den letzten Monaten stark weiterentwickelt. Windows Azure stellt seit kurzem die IIS Media Services als Onlinedienst zur Verfügung. Damit kann die Enkodierung der Videos in die Cloud ausgelagert werden. Auch das Hosting der Webapplikation oder der Datenbank kann durch Windows Azure übernommen werden. Klar stellt sich dabei die Frage, ob man seine Daten bei einem Drittanbieter ablegen will. Gleichzeitig wird es aber für eine Firma eine riesige Herausforderung sein, die angesprochenen Dienste selber zu betreiben. Die Portierung einer Desktopsoftware ins Web ist kein kleines Unterfangen und benötigt eine sorgfältige Planung. Viele Konzepte müssen dabei überdenkt werden. Die Webentwicklung ist und bleibt ein spannendes Umfeld, welches auch immer wichtiger wird.

---

<sup>53</sup> <http://www.windowsazure.com/>

## 5 Anhang

### 5.1 Bibliographie

@fat&@mdo. (9. August 2012). *Bootstrap, from Twitter*. Abgerufen am 9. August 2012 von Bootstrap, from Twitter: <http://twitter.github.com/bootstrap/>

CDLab. (9. August 2012). *WinCan*. Abgerufen am 9. August 2012 von WinCan: <http://www.wincan.com/>

Guthrie, S. (2. Juli 2010). *Introducing Razor - a new view engine for ASP.NET*. Abgerufen am 9. August 2012 von ScottGu's Blog.

JSON. (9. August 2012). *Introducing JSON*. Abgerufen am 9. August 2012 von JSON: <http://json.org/>

Lawrence, E. (9. August 2012). *Introducing Fiddler*. Abgerufen am 9. August 2012 von Fiddler: <http://www.fiddler2.com/fiddler2/>

Microsoft. (27. Januar 2009). *ASP.NET MVC Overview*. Abgerufen am 9. August 2012 von ASP.NET: <http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview>

Microsoft. (9. August 2012). *ASP.NET MVC 4*. Abgerufen am 9. August 2012 von ASP.NET: <http://www.asp.net/mvc/mvc4>

Microsoft. (9. August 2012). *ASP.NET Single Page Applications*. Abgerufen am 9. August 2012 von ASP.NET: <http://www.asp.net/single-page-application>

Microsoft. (9. August 2012). *Entity Framework*. Abgerufen am 9. August 2012 von ASP.NET: <http://www.asp.net/entity-framework>

Microsoft. (9. August 2012). *Getting Started with ASP.NET Web API*. Abgerufen am 9. August 2012 von ASP.NET: <http://www.asp.net/web-api>

Microsoft. (10. August 2012). *Microsoft Media Platform Player Framework*. Abgerufen am 9. August 2012 von CodePlex: <http://playerframework.codeplex.com/>

Microsoft. (9. August 2012). *NuGet Overview*. Abgerufen am 9. August 2012 von NuGet Docs: <http://docs.nuget.org/docs/start-here/overview>

Pilgrim, M. (9. August 2012). *No 5. Video on the Web*. Abgerufen am 9. August 2012 von Dive Into HTML5: <http://fortuito.us/diveintohtml5/video.html>

Sanderson, S. (9. August 2012). *Knockout*. Abgerufen am 9. August 2012 von Knockout: <http://knockoutjs.com/>

VisualStudioMagazine. (10. April 2012). *Entity Framework Code-First Migrations*. Abgerufen am 9. August 2012 von Visual Studio Magazine: <http://visualstudiomagazine.com/articles/2012/04/10/entity-framework-code-first-migrations.aspx>

VisualStudioMagazine. (1. Juni 2012). *Powerful JavaScript With Upshot and Knockout*. Abgerufen am 9. August 2012 von Visual Studio Magazine: <http://visualstudiomagazine.com/articles/2012/06/01/the-upshot-its-a-knockout.aspx>

Wikipedia. (27. Juni 2012). *Codec*. Abgerufen am 9. August 2012 von Wikipedia: <http://de.wikipedia.org/wiki/Codec>

Wikipedia. (2. August 2012). *Language Integrated Query*. Abgerufen am 9. August 2012 von Wikipedia: [http://en.wikipedia.org/wiki/Language\\_Integrated\\_Query](http://en.wikipedia.org/wiki/Language_Integrated_Query)

Wikipedia. (8. August 2012). *Model View ViewModel*. Abgerufen am 9. August 2012 von Wikipedia: [http://en.wikipedia.org/wiki/Model\\_View\\_ViewModel](http://en.wikipedia.org/wiki/Model_View_ViewModel)

Wikipedia. (5. August 2012). *Open Source*. Abgerufen am 9. August 2012 von Wikipedia: [http://de.wikipedia.org/wiki/Open\\_Source](http://de.wikipedia.org/wiki/Open_Source)

Wikipedia. (15. Juni 2012). *Plain Old CLR Object*. Abgerufen am 9. August 2012 von Wikipedia: [http://en.wikipedia.org/wiki/Plain\\_Old\\_CLR\\_Object](http://en.wikipedia.org/wiki/Plain_Old_CLR_Object)

Wikipedia. (6. August 2012). *Representational State Transfer*. Abgerufen am 9. August 2012 von Wikipedia: [http://en.wikipedia.org/wiki/Representational\\_state\\_transfer](http://en.wikipedia.org/wiki/Representational_state_transfer)

Wikipedia. (10. August 2012). *Responsive Design*. Abgerufen am 9. August 2012 von Wikipedia: [http://de.wikipedia.org/wiki/Responsive\\_Design](http://de.wikipedia.org/wiki/Responsive_Design)

Wikipedia. (8. August 2012). *XML*. Abgerufen am 9. August 2012 von Wikipedia: <http://en.wikipedia.org/wiki/XML>

Wikipedia. (17. Juni 2012). *XMLHttpRequest*. Abgerufen am 9. August 2012 von Wikipedia: <http://de.wikipedia.org/wiki/XMLHttpRequest>

## 5.2 Abbildungsverzeichnis

Abbildung 1 - Logfile eines automatisierten Tests.....	2
Abbildung 2 - Die Hauptkomponenten von MVC .....	6
Abbildung 3 - POCO Implementierung eines Projekts.....	6
Abbildung 4 - Die Projektübersicht als PartialView .....	7
Abbildung 5 - Request-Header der in der Response JSON akzeptiert.....	8
Abbildung 6 - Response mit drei Datensätzen im JSON Format.....	8
Abbildung 7 - Ausschnitt des KnockoutJS ViewModels .....	9
Abbildung 8 - KnockoutJS Bindings im HTML Code .....	10
Abbildung 9 - Definition einer Upshot.js RemoteDataSource .....	10
Abbildung 10 - Responsive Design für Desktop (oben), Mobile (u. links) und Tablets (u. rechts).....	11
Abbildung 11 - Ausschnitt des LESS Codes für die Navigation.....	12
Abbildung 12 - jQuery Validation Fix .....	13
Abbildung 13 - Resource Dateien in zwei Sprachen .....	13
Abbildung 14 - Webapplikation für die automatische Enkodierung aus dem Vorgängerprojekt .....	14
Abbildung 15 - FTP Client für die Übertragung grosser Videodateien .....	15
Abbildung 16 - Quelle: <a href="http://fortuito.us/diveintohtml5/video.html">http://fortuito.us/diveintohtml5/video.html</a> .....	16
Abbildung 17 - Google Entwickler Tools mit XHR Aufrufen .....	18
Abbildung 18 - Fehlerlog von ELMAH.....	19
Abbildung 19 - Übersicht der Aktivitäten im GitHub Repository.....	20
Abbildung 20 - TeamCity Build Server Übersicht.....	21
Abbildung 21 - Web.Config Transformations für das Produkktivsystem .....	21
Abbildung 22 - Die aus dem Projekt entstandene Webapplikation .....	22



## 6 Ehrlichkeitserklärung

Die eingereichte Arbeit ist das Resultat meiner persönlichen, selbstständigen Beschäftigung mit dem Thema. Ich habe für sie keine anderen Quellen benutzt, als die in der Dokumentation erwähnten.

Tobias Studer, Mittwoch, 15. August 2012

Unterschrift:

A handwritten signature in blue ink, appearing to read 'Studer', with a large, stylized flourish on the left side.