

**Thesis / Entwicklungsdokumentation**

Analyse, Konzeption und Implementierung der  
Integration von arsnova.click und arsnova.voting in  
Microsoft Office 365

**Autor:** Tjark Wilhelm Hoeck

**Erstellt im:** März 2017

## Abstract

Präsentationsprogramme unterstützen den Alltag fast aller vortragenden Personen durch eine visuelle Ebene und sind heutzutage nicht mehr wegzudenken. Doch die Lehre befindet sich im Wandel, Themen wie der invertierte Klassenraum rücken immer mehr in den Vordergrund und werden bereits seit Jahren durch eigenständige Softwarelösungen abgebildet. Für einen optimalen Lernerfolg sollten beide Systeme angewandt werden, doch die Bedienung mehrerer Programme gleichzeitig lenkt den Vortragenden ab und ist damit kaum praktikabel.

Zur Verbesserung der Vortrags-Produktivität und für eine Erhöhung des Lerneffektes soll Microsoft Office PowerPoint sich die Stärken des Lernens 4.0 durch eine Integration von multiplen Audience-Response-Systemen zunutze machen.

Zu Beginn der Arbeit wird der Leser in die nötigen Grundlagen eingeführt. Danach erfolgt eine Beschreibung der Benutzeroberfläche in PowerPoint, welche die Funktionen der Audience-Response-Systeme darstellt. So wird ein Überblick über die zu Verfügung stehenden Funktionalitäten gewonnen. Im gleichen Kapitel werden auch die Anforderungen an die zu integrierenden Systeme definiert und analysiert. Der nächste Abschnitt befasst sich mit der Konzeption der Anwendung und geht vor allem auf die eingesetzten Instrumente der Softwareentwicklung ein. Zum Ende hin werden einzelne Besonderheiten der Implementierung vorgestellt und ein Rück- sowie Ausblick auf das Projekt gewährt.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>5</b>
<b>Abkürzungsverzeichnis</b>	<b>6</b>
<b>1 Einleitung</b>	<b>8</b>
1.1 Ausgangssituation und Zielsetzung . . . . .	8
1.2 Gliederung . . . . .	9
<b>2 Grundlagen</b>	<b>10</b>
2.1 Begriff "Audience Response System" . . . . .	10
2.2 Begriff "Integration" . . . . .	10
2.3 ARSnova . . . . .	11
2.3.1 arsnova.voting . . . . .	14
2.3.2 arsnova.click . . . . .	17
2.4 Microsoft Office PowerPoint . . . . .	17
2.5 Technologien . . . . .	17
2.5.1 VSTO . . . . .	18
2.5.2 WPF . . . . .	18
2.5.3 XAML . . . . .	18
2.5.4 Meteor . . . . .	19
<b>3 Analyse</b>	<b>21</b>
3.1 Einstiegspunkte . . . . .	21
3.1.1 Menüband . . . . .	21
3.1.2 Kontextmenüs . . . . .	22
3.1.3 Präsentation . . . . .	23
3.2 Anwendungsfälle . . . . .	24
3.2.1 Installation . . . . .	25
3.2.2 ARSnova App auswählen . . . . .	26
3.2.3 Verwalten von Quizfragen . . . . .	27
3.2.4 Anlegen und Durchführen . . . . .	28
3.2.5 Bearbeiten . . . . .	32
3.2.6 Ausblenden . . . . .	32
3.2.7 Löschen . . . . .	32
3.2.8 Informationen . . . . .	33
3.3 Anforderungen . . . . .	33
3.3.1 Weitere funktionale Anforderungen . . . . .	33
3.3.2 Nicht-funktionale Anforderungen . . . . .	34
3.3.3 ARSnova Systeme . . . . .	34
3.4 Anforderungsanalyse . . . . .	35
3.4.1 Analyse von PowerPoint Add-ins . . . . .	35
3.4.2 Alternativen . . . . .	38
3.4.3 Analyse von ARSnova . . . . .	38
<b>4 Konzeption</b>	<b>41</b>
4.1 Technologien und Bibliotheken . . . . .	41
4.2 Architektur . . . . .	42
4.3 Projektstruktur . . . . .	42

4.4	Entwurfsmuster . . . . .	54
4.4.1	Separated Interface . . . . .	54
4.4.2	Service Locator . . . . .	55
4.4.3	Dependency Injection . . . . .	56
4.5	Model-View-ViewModel . . . . .	59
4.6	Installation . . . . .	59
4.7	Integration von ARSnova in PowerPoint . . . . .	61
4.7.1	Technische Abläufe . . . . .	62
4.7.2	Schnittstellenverwendung von arsnova.voting . . . . .	66
4.7.3	Schnittstellendefinition von arsnova.click . . . . .	67
<b>5</b>	<b>Implementierung</b>	<b>68</b>
5.1	Anbindung an PowerPoint . . . . .	68
5.2	WPF-Erweiterung . . . . .	71
5.2.1	ViewPresenter . . . . .	72
5.2.2	Watermark-Service . . . . .	73
5.3	LINQ . . . . .	74
5.4	Dependency Injection . . . . .	76
5.5	Lokalisierung . . . . .	77
5.6	Object-Mapper . . . . .	78
5.7	Build und Setup Erstellung . . . . .	80
5.8	Grenzen der VSTO . . . . .	81
5.8.1	Start Button . . . . .	81
5.8.2	Beeinträchtigte Folien-Navigation . . . . .	81
5.8.3	Sperren von Folien . . . . .	82
5.9	arsnova.click . . . . .	82
<b>6</b>	<b>Zusammenfassung</b>	<b>85</b>
6.1	Rückblick . . . . .	85
6.2	Ausblick . . . . .	87
6.2.1	arsnova.voting Backend . . . . .	88
6.2.2	arsnova.click API . . . . .	88
6.3	Fazit . . . . .	89
	<b>Anhang</b>	<b>90</b>
	<b>Literaturverzeichnis</b>	<b>92</b>

## Abbildungsverzeichnis

1	Das PowerPoint-Menüband mit dem ARSnova-Eintrag . . . . .	21
2	Das Kontextmenü einer PowerPoint-Folie . . . . .	22
3	Das ARSnova Menüband . . . . .	24
4	Annahme der Lizenzvereinbarungen . . . . .	25
5	Der Windows-Installationseintrag . . . . .	26
6	Das Fenster zur Auswahl des zu verwendenden ARS . . . . .	26
7	Eine nicht erfolgreiche Validierung des Hashtag-Feldes . . . . .	27
8	Eine Übersicht über alle Fragen der aktuellen Präsentation . . . . .	28
9	Erstellen einer Frage: Teil 1 . . . . .	29
10	Erstellen einer Frage: Teil 2 . . . . .	30
11	Festlegen der Antwortoptionen: Typ Freitext-Frage . . . . .	31
12	Schichtabhängigkeiten . . . . .	43
13	Die Projektliste . . . . .	44
14	Auflistung der Dokumente im Projekt "Business" . . . . .	44
15	Auflistung der Dokumente im Projekt "Business.Contract" . . . . .	45
16	Auflistung der Dokumente im Projekt "Business.Model" . . . . .	45
17	Auflistung der Dokumente im Projekt "Common" . . . . .	46
18	Auflistung der Dokumente im Projekt "Common.Contract" . . . . .	47
19	Auflistung der Dokumente im Projekt "Common.Enum" . . . . .	47
20	Auflistung der Dokumente im Projekt "Common.Helpers" . . . . .	47
21	Auflistung der Dokumente im Projekt "Common.Resources" . . . . .	48
22	Auflistung der Dokumente im Projekt "Communication" . . . . .	48
23	Auflistung der Dokumente im Projekt "Communication.Contract" . . . . .	49
24	Auflistung der Dokumente im Projekt "Communication.Model" . . . . .	49
25	Auflistung der Dokumente im Projekt "Presentation" . . . . .	50
26	Auflistung der Dokumente im Projekt "Presentation.Configuration" . . . . .	53
27	Auflistung der Dokumente im Projekt "Test" . . . . .	53
28	Beispiel am Projekt: Separated Interface-Pattern . . . . .	54
29	Sequenzdiagramm: Eine Umfrage in arsnova.click . . . . .	63
30	Sequenzdiagramm: Eine Umfrage in arsnova.voting . . . . .	65
31	Beispiel: Ein Platzhalter in einer Text-Box . . . . .	71
32	Die LINQ-Architektur . . . . .	75

## Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface
<b>ARS</b>	Audience Response System
<b>BYOD</b>	Bring Your Own Device
<b>CLR</b>	Common Language Runtime
<b>CRUD</b>	Create, Read, Update and Delete
<b>CSS</b>	Cascading Style Sheets
<b>DLL</b>	Dynamic Link Library
<b>GDI</b>	Graphics Device Interface
<b>GUI</b>	Graphical User Interface
<b>HTML</b>	Hypertext Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IDE</b>	Integrated development environment
<b>IoT</b>	Inversion of Control
<b>JS</b>	JavaScript
<b>JSON</b>	JavaScript Object Notation
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>LINQ</b>	Language Integrated Query
<b>MEAN</b>	MongoDB, Express, Angular und Node.js
<b>MSDN</b>	Microsoft Developer Network
<b>MVC</b>	Model-View-Controller
<b>MVP</b>	Model-View-Presenter
<b>MVVM</b>	Model-View-ViewModel
<b>REST</b>	Representational State Transfer
<b>SOAP</b>	Simple Object Access Protocol
<b>SSL</b>	Secure Sockets Layer
<b>SQL</b>	Structured Query Language
<b>UTF</b>	Unicode Transformation Format
<b>URL</b>	Uniform Resource Locator
<b>VB</b>	Visual Basic

**VSTO** Visual Studio Tools for Office

**WPF** Windows Presentation Foundation

**WS** WebSocket

**XAML** Extensible Application Markup Language

**XML** Extensible Markup Language

# 1 Einleitung

Innerhalb eines Informatik Studiums werden viele Techniken und Methoden zur Entwicklung von Software gelehrt. Eine Abschlussthesis soll neben der Vertiefung einer Thematik auch das erlernte Wissen repräsentieren. Dieses zeigt sich am besten an der praktischen Umsetzung des theoretisch Erlernten: "Für das Können gibt es nur einen Beweis: das Tun." (von Ebner-Eschenbach, 1911). Aus diesem Grund spezialisiert sich diese Thesis nicht nur auf ein Thema, sondern behandelt auch den gesamten Entwicklungsprozess einer Softwarelösung.

Es gibt viele gute Anwendungen, welche die Produktivität im Alltag immens steigern können. Doch oft werden diese Produkte aus ähnlichen Gründen nicht eingesetzt: Die Einarbeitungszeit schreckt ab. Dabei kann die Usability noch so gut sein, der Mensch gewöhnt sich nur ungern an neue Oberflächen oder neue Wertschöpfungsprozesse. Viele Erfahrungswerte mit Programmen, deren Anwendung zu Beginn mehr Aufwand als tatsächliche Zeitersparnisse eingebracht haben, festigen den Eindruck, dass das bestehende System nicht ersetzt werden sollte. Und damit liegen die Anwender nicht falsch. Eine Anwendung ist erst hilfreich, wenn der Nutzer diese vollends durchblickt hat, ihren Funktionsumfang kennt und sich mit der Oberfläche soweit auseinandergesetzt hat, dass er diese zügig bedienen kann. Dabei kann eine Bedienungsanleitung helfen, doch wird zu dieser nur selten gegriffen. Erst wenn eine Anwendung intuitiv verwendbar ist, kann man von einer gelungenen Usability sprechen. In jedem anderen Fall wirkt das Designkonzept undurchsichtig auf den Endanwender.

Aus dieser Problematik ergibt sich der zentrale Antrieb dieser Thesis: Einer Anwendung eine Usability zu verleihen, die für alle Anwender intuitiv bedienbar ist, ist nur schwer möglich. Doch wenn eine Software bekannt ist, die ein großer Teil der Nutzer bereits verwendet und deren Oberflächenbedienung somit keinen Lernaufwand für diese Zielgruppe darstellt, kann eine Integration in diese Anwendung erfolgen. Damit kann der Nutzer die neuen Funktionalitäten ohne die Auseinandersetzung mit einer neuen Benutzeroberfläche verwenden. Einzige Vorbedingung: Die bestehende sowie die Integrationsanwendung sollten thematisch so nah beieinander liegen, dass ein Zusammenspiel konzeptionell sinnvoll ist und das Ergebnis homogen erscheint.

## 1.1 Ausgangssituation und Zielsetzung

Diese Thesis wurde an der Technischen Hochschule Mittelhessen erstellt. Eine Erläuterung der bestehenden Audience Response System (ARS), die in der hier konzipierten und implementierten Anwendungslösung angebunden werden, findet sich in dem Kapitel ARSnova auf S. 11. Um eine einfache Anwendung der Systeme und damit eine Verbesserung der Hochschullehre zu ermöglichen, ergibt sich folgende Zielsetzung:

Es soll die Möglichkeit bestehen, die Funktionalitäten der beiden ARS arsnova.voting und arsnova.click während einer Präsentation mit Microsoft Office PowerPoint anzuwenden. Auch die Vorbereitungsphase soll mit PowerPoint erfolgen, die zur Zeit bestehenden Web-Oberflächen zur Bedienung der beiden ARSnova-Systeme müssen dem Vortragenden nicht bekannt sein.

Dabei sollen alle Funktionalitäten nahtlos integriert werden, die PowerPoint nicht schon von Haus aus mitbringt. So ist beispielsweise die Integration der Vortragsfolien-Funktion nicht nötig. Die Bedienung muss nativ, als wäre sie schon bei der Auslie-



ferung des Präsentationsprogrammes vorhanden gewesen, anzuwenden sein. Während der Vortragende nur mit PowerPoint arbeitet, nutzt das Auditorium die bestehenden Web-Applikationen. Für die Zuhörerschaft darf das System zum Verwalten der Anwendungen nicht durch das Teilnehmen an einer Sitzung sichtbar werden.

Neben der Analyse der Umsetzbarkeit sowie der Findung entsprechender Frameworks und Entwicklungstechnologien besteht ebenfalls die Anforderung der Backend-Definition und dessen Implementierung für arsnova.click.

## 1.2 Gliederung

Zunächst befasst sich diese Arbeit mit zwingend erforderlichen Grundlagen, um alle Leser für das spätere Verständnis auf einen Wissensstand zu bringen. Dies umfasst die Softwaresysteme, welche im Rahmen dieser Arbeit aneinander angebunden werden sollen sowie die Definitionen von Begrifflichkeiten und eine Einführung in die verwendeten Technologien.

Die Analyse beschäftigt sich mit den Anforderungen an die Integrationslösung. Dies betrifft neben den Funktionalitäten für den Endanwender auch die funktionalen und nicht-funktionalen Anforderungen an die beiden anzubindenden Systeme.

Der Schwerpunkt dieser Arbeit spiegelt sich in dem Abschnitt Konzeption wieder. Das Hauptaugenmerk liegt auf der technischen Realisierung der Integration, der Anbindung der beiden externen Systeme sowie der Umsetzung aller in der Analyse definierten Anwendungsszenarien. Des Weiteren werden auch softwareseitige Konzeptionsfragen wie die Architektur der Anwendung geklärt.

Der Bereich Implementierung beinhaltet eine Darstellung über den derzeitigen Entwicklungsstand mit einigen Ausschnitten aus dem Quellcode. Da dies nur ein kurzer Codeausschnitt ist, wird hier Wert auf die Besonderheiten in der Entwicklung gelegt. Auch wird auf spezielle Techniken sowie technologische Hürden eingegangen.

In dem letzten Kapitel Zusammenfassung erfolgt ein Resümee über den gesamten Ablauf des Softwareentwicklungs-Prozesses. Dies schließt auch einen Vergleich der Zielsetzung mit dem Endstadium ein und gibt dann einen Ausblick auf den zukünftigen Weg der Anwendung.

## 2 Grundlagen

In diesem Kapitel wird ein grundlegendes Wissen zum Verständnis der späteren Arbeit aufgebaut. Dazu gehören die Beschreibungen zu der Integrationslösung selbst und den anzubindenden Systemen, den dabei verwendeten Technologien und einigen zentralen Begriffen.

### 2.1 Begriff "Audience Response System"

In dem Kern der Arbeit geht es um die Integration von zwei bestehenden ARS. Daher ist es unabdingbar, den Begriff "Audience Response System" einzuführen. Dieser stammt aus dem Englischen und bedeutet soviel wie "Publikums-Rückmelde-System". Gerade an Hochschulen erfreuen sich die Systeme immer größerer Beliebtheit, da die Kommunikations- und Interaktionsbarriere zwischen der vortragenden Person und dem Publikum proportional zur Größe des Publikums wächst und die Vorlesungsräume nicht selten von weit mehr als einhundert Studenten gefüllt werden.

Diese Barrieren können heutzutage durch Softwarelösungen wie den ARS aufgehoben werden. Die Kommunikation zwischen Menschen kann vereinfacht werden, indem die auszutauschenden Informationen von einem System verteilt werden. Die Anwendung sorgt für eine didaktisch vollständige Formulierung der Fragen von Seiten der vortragenden Person und verteilt diese gezielt an jeden einzelnen Zuhörer und sorgt damit dafür, dass ein beidseitiges Verständnis gewährleistet ist. Die Antworten werden zwingend einfach gehalten und die Ergebnisse anschließend zusammenfassend, beispielsweise in Form einer grafischen Statistik, dargestellt. Mit Hilfe eines solchen Systems ist die Abbildung der verschiedensten Meinungen eines großen Publikums innerhalb einer kurzen Zeitspanne gewährleistet. Dies ist ein Beispiel für den Einsatz eines Systemes, welches die Kommunikation unter Menschen verbessert und erleichtert. Sobald die Zuhörerschaft eine Größe annimmt, mit der ein Meinungs- oder Wissensbild auf Grund der vielen verschiedenen Individuen nicht mehr in einer kurzen und damit akzeptablen Zeitspanne erreichbar ist, kann ein ARS die Stimmen mit gezielten Fragen zusammenfassen und dem Vortragenden nach einem kurzen Zeitraum gebündelt zur Verfügung stellen (Kibler, 2015).

Das genannte Beispiel ist nur eines der vielen Anwendungsfälle von ARS und verdeutlicht die Aufgabe dieser Softwaregruppe. Der volle Funktionsumfang eines ARS ist beispielhaft in dem Kapitel `arsnova.voting` nachzulesen.

### 2.2 Begriff "Integration"

Der Begriff der Integration variiert in seiner Bedeutung stark, je nachdem in welchem Kontext dieser verwendet wird. Bei der Betrachtung im Rahmen der Softwareentwicklung ähnelt die Integration dem Begriff der Kopplung. Bei der Kopplung wird in der Regel eine Schnittstelle definiert, über die zwei vorher unabhängige Systeme Informationen austauschen können. Bei einer Integration sollen die Systeme ebenfalls aneinander angebunden werden, doch soll dies unter der Vermeidung von neuen Schnittstellendefinitionen vonstatten gehen. Dabei wird zwischen drei verschiedenen Arten der Integration unterschieden:

- Funktionsintegration

Bei einer Integration zur Vermeidung der Implementierungen von ähnlichen oder gleichen Funktionen oder dem Ziel, großen Kommunikationsbedarf durch Zentralisierung von Funktionalitäten zu verringern, spricht man von Funktionsintegration.

- Datenintegration

Wenn mehrere Applikationen zur Vermeidung von Redundanzen auf den gleichen Datenmodellen arbeiten, handelt es sich um einen Fall der Datenintegration.

- Geschäftsprozessintegration

Mit Hilfe einer Integrationssoftware werden mindestens zwei verschiedene Geschäftsprozesse, die in unabhängigen Applikationen abgebildet werden, miteinander integriert. Das Besondere an dieser Stelle ist, dass die beiden Anwendungen in ihrer Funktionalität nicht durch den Integrationsprozess beeinträchtigt werden.

Bei dem Vergleich der drei Begriffsdefinitionen mit den Anforderungen kristallisiert sich heraus, dass es sich bei der Integration der beiden ARS um eine “volle” Integration handelt. Eine volle Integrationslösung bildet die Integration aller drei Unterdefinitionen ab.

Die Funktionsintegration beschreibt in dem Kontext dieser Thesis Fall den Willen, die beiden ARS in Microsoft PowerPoint integrieren zu wollen, um die Möglichkeiten des interaktiven und invertierten Vortrages in der Präsentationssoftware abzubilden, ohne die Funktionalität ein weiteres Mal zu implementieren.

Die Datenintegration spricht für sich. Um Redundanzen in der Datenhaltung zu vermeiden, werden die Datenmodelle der beiden ARS-Serveranwendungen verwendet - auch von der Integrationslösung.

Die Geschäftsprozessintegration erfolgt an dem Punkt, an dem die beiden unabhängigen Prozesse der Präsentation und des interaktiven Vortrages nach wie vor mit den einzelnen Komponenten durchführbar sind, doch durch die Integrationslösung miteinander gekoppelt werden können.

## 2.3 ARSnova

ARSnova ist, was der Name verspricht: ein neues und innovatives Projekt für ein ARS. Dabei bringt es alle Features mit, die eine Anwendung zu einem ARS machen und im nachfolgenden Abschnitt arsnova.voting dargelegt werden. Die Anwendung macht sich die Verbreitung von Smartphones und Tablets zunutze und realisiert das Konzept von Bring Your Own Device (BYOD). Hierbei erhält die Zuhörerschaft keine weitere Hardware, die als Eingabemittel für die Anwendung dient, sondern verwendet seine eigenen, portablen Geräte wie das Smartphone oder ein Tablet. ARSnova steht dabei als skalierende Web-Anwendung bereit, kann also ohne Installation und in der Darstellung für jedes Endgerät optimiert eingesetzt werden. Dies ermöglicht einen schnellen Einstieg, da bereits jeder Anwender mit der Bedienung seines persönlichen Gerätes vertraut ist, und reduziert gleichzeitig die zusätzlichen Hardwarekosten gegen Null.

Durch die inzwischen mehrjährige Entwicklung von ARSnova gab es viele kleine Optimierungen, wie beispielsweise das Aufteilen der stark angewachsenen Funktionsauswahl in Use Cases (zu dt.: Anwendungsfälle). Je nach gewähltem Use Case wird die Oberfläche entschlackt, um dem Anwender ein aufgeräumtes und intuitives Bedienungserlebnis

bieten zu können.

Im weiteren Entwicklungsverlauf wurde die Erkenntnis gewonnen, dass man trotz dieser Usability-verbessernden Maßnahmen nicht alle Anwendungsfälle, die für eine digitale Optimierung in der Lehre in Frage kommen, auf einem System abbilden kann. Zu den bereits umgesetzten und beschriebenen Features kamen Anforderungen wie die “Gamifizierung” oder das “nachhaltige Lernen” auf.

Kurz zusammengefasst beschreibt die Gamifizierung das Anwenden von spieltypischen Elementen in einem nicht Spiele-üblichen Kontext. Während dieser Aspekt in der Wirtschaft zur Mitarbeitermotivation oder der Kundenbindung eingesetzt wird, soll in der Lehre eine Verbesserung der Aufmerksamkeit erreicht werden (Schuh & Stich, 2014, S. 191). Zur Realisierung werden folgende Elemente eingesetzt:

- Wettbewerb

Wettbewerbsbedingungen sind ein großer Teil der Gamifizierung. In vielen Spielen geht es darum, seine Konkurrenten auszusteichen, sei es nun im kognitiven oder reaktionsschnellen Bereich. Hierbei wird den Teilnehmern ein ganz einfacher Grund gegeben, aufmerksam dabei zu sein: Sie wollen besser sein als der Rest. Dieses Verhalten wäre nicht so stark ausgebildet, wenn man den Personen klar machen würde, dass sie für sich und ihre Zukunft lernen - kurzfristige Ziele sind hierbei viel motivierender (Slaghuis, 2013).

- Zeitdruck

Druck ist Stress, allerdings nur in zu hohem Maße negativer. In kleinen Dosen fördert er die Konzentration und setzt den Körper in eine Art kurzen “Ausnahmestand”. Diese Art der Gamifizierung kann man wunderbar an die Wettbewerbsbedingung knüpfen. Dabei werden gleich zwei Ziele erfüllt: Mehr Einsatz der Zuhörerschaft durch den Wettbewerb mit Zeitdruck sowie ein festgesetztes Ende der Aufgabe, wodurch der Einsatz dieser Mittel in vorgegebenen Zeiten wie Vorlesungen oder Vorträgen kalkulierbarer wird.

- Belohnung

Eine Belohnung ist immer ein guter Anreiz, um jemanden zur freiwilligen Aktion zu motivieren. Dies kann auch schon im kleinen Stile erfolgen, wie einer Rangliste nach einem Wettbewerb. Wer seinen Namen ganz oben sieht, weiß wie gut er war und wird durch die Anerkennung und auffällige Darstellung entlohnt.

- Design

Das Design einer Anwendung vermittelt dem Anwender unterbewusst ein Gefühl. Bei einem eintönigen und eckigen Design kann somit etwas Seriosität erzeugt werden. Bunte, runde und damit verspielte Elemente lockern das Verhältnis spürbar auf. Dadurch vermittelt man sofort den Eindruck, dass es mehr um ein Spiel als um eine Aufgabe oder wichtige Prüfung geht. Folglich lockert sich die Stimmung und es wird eine deutlich bessere Lernatmosphäre geschaffen. Beispielsweise ist es nicht schlimm, in einem Spiel etwas nicht zu wissen oder einen Fehler zu machen, in einer Prüfung oder vor dem Vorgesetzten ist dies peinlich und führt bei vielen Personen zu einem introvertierten Verhalten: Es wird lieber nichts gesagt als eventuell etwas Falsches von sich zu geben. Doch auch Fehler haben einen hohen Lerneffekt und sollten in der Lehre auf keinen Fall zwingend vermieden werden.

- Musik

Musik beeinflusst die Stimmung von ihren Zuhörern. Durch gezielten Einsatz kann diese aufputschend wirken (Dramatik und Spannung). Ein Beispiel hierfür wäre die Untermalung von Wettbewerbsbedingungen, dadurch wird das Konkurrenzdenken und der Lauf gegen die Uhrzeit weiter verstärkt. Ebenfalls könnte eine locker-fröhliche Melodie die Teilnehmer bei Laune halten, wenn man auf andere Teilnehmer warten muss. Dies ist auch der Grund, warum viele Lehr-Videos, die einen rein informellen Hintergrund haben, oft mit solchen Musikrichtungen hinterlegt werden.

Der Begriff “nachhaltiges Lernen” beschreibt ein Thema, das heutzutage viele Lehrkräfte, sei es an Schulen oder Hochschulen, beschäftigt. Die Klausur beziehungsweise das Kolloquium sind die wohl anerkanntesten und meist verbreitetsten Prüfungsverfahren. Wenn jemand eine solche Prüfung besteht, wird davon ausgegangen, dass das entsprechende Wissen vorhanden ist, auch Monate oder sogar Jahre später. Doch oft müssen Lehrkräfte feststellen, dass die Schüler oder Studenten Wissen, welches sie auf dem Papier bereits erlangt haben, nicht auf Abruf parat haben. Denn das stellen die beiden verbreitetsten Lehrmethoden nicht sicher. Hier wird lediglich geprüft, ob das Wissen zu einem bestimmten Zeitpunkt vorhanden ist. Ob dies durch kurzfristiges oder langfristiges Lernen erreicht wurde, ist dabei nicht wichtig. Nachhaltig wurde erst gelernt, wenn das Wissen sich im Langzeitgedächtnis befindet. Nun wird nach Methoden gesucht, den Schülern das Wissen langfristig zu vermitteln (Schmidt, 2003, S. 40-50). Dabei gibt es schon mehrere Algorithmen und entsprechende Programme zum nachhaltigen Lernen. Gerade der Leitner-Algorithmus ist eines der bekanntesten Mittel für ein erfolgreiches, langfristiges Lernen. Das Wissen wird dabei über mehrere Stufen, in immer länger werdenden Intervallen, abgefragt. Die nächste Stufe wird dabei erst erreicht, wenn das Wissen nach Ablauf des jeweiligen Intervalles zur Verfügung stand. Falls nicht, beginnt der Prozess für dieses spezielle Wissen, meist niedergeschrieben auf einer Lernkarte, wieder bei der niedrigsten Stufe. Nachhaltig wurde die jeweilige Karte gelernt, wenn die letzte Stufe erreicht ist. In der Regel benötigt dies eine Zeitdauer von drei bis sechs Monaten. Wenn nach einer solchen Zeitspanne das Wissen noch zur Verfügung steht, kann man davon ausgehen, dass sich das Wissen im Langzeitgedächtnis befindet und im Alltag schnell abrufbar ist (Leitner, 2011).

Da sich diese beiden Anforderungen ebenfalls um die Verbesserung der Lehre, genau wie das ursprüngliche ARS, drehen, wurden diese in Form von neuen Anwendungen von dem ARSRnova-Entwicklungsteam konzipiert und implementiert. Daraus erfolgt eine Reformation der Namensgebung, um zwischen den einzelnen Produkten differenzieren zu können:

- arsnova.voting

Das ursprüngliche ARS, bestehend aus den typischen Konzepten zum “Flipped Classroom”. Diese Anwendung ist geprägt durch ein seriöses Auftreten und wissenschaftliche Aspekte wie der Unterstützung von LaTeX-Auszeichnungen. Es stehen Versionen mit angepassten Texten zur Verfügung, um den Einsatz auch außerhalb der Lehre attraktiv und passend zu gestalten. Die Anwendung unterteilt sich in ein Frontend (ARSnova-Team der TH Mittelhessen, 2017d) und ein Backend (ARSnova-Team der TH Mittelhessen, 2017c).

- arsnova.click

Ein abgespecktes ARS, doch dafür voller Gamification-Aspekte. Hier geht es um den Anreiz, die Zuhörerschaft zur Aufmerksamkeit zu motivieren sowie dem Schaffen einer lockeren Atmosphäre. Gerade zu Beginn oder zum Ende von Vorträgen ist der Einsatz dieses Systemes zu empfehlen. Auch Auflockerungen während einer Präsentation, gerade bei einem schläfrigen Publikum, sind ein gewinnbringendes Mittel (ARSnova-Team der TH Mittelhessen, 2017a).

- arsnova.cards

Die Lernkarten-Software, welche mit dem Leitner-Algorithmus arbeitet, schafft den Zuhörern eine Möglichkeit zum nachhaltigen Lernen. Durch die starke Anbindung an die restlichen ARSnova-Systeme sowie dem hochschuleigenen Lightweight Directory Access Protocol (LDAP)-System ist es ohne großen Aufwand möglich, Vorlesungsinhalte zum nachhaltigen Lernen zur Verfügung zu stellen und den Fortschritt zu kontrollieren bzw. sogar zu benoten.

Im Rahmen dieser Arbeit werden zwei der drei Systeme in PowerPoint integriert: arsnova.voting und arsnova.click, also die beiden Softwarelösungen, die aktiv während eines Vortrages eingesetzt werden können. In den folgenden zwei Abschnitten werden diese beiden Anwendungen näher erläutert und ihre genauen Einsatzzwecke dargestellt, um die Möglichkeiten der anzubindenden Funktionen sowie deren Sinn verständlicher zu gestalten (Quibeldey-Cirkel, 2016).

### 2.3.1 arsnova.voting

Natürlich gibt es verschiedene Zuhörerschaften und Gründe, einen Vortrag zu halten, und mit diesen ändert sich stetig auch der Use Case eines ARS. Auf diesen Punkt hat sich dieses System spezialisiert. Es hat mehrere, vordefinierte Anwendungsfälle und passt sich damit den individuellen Anforderungen des aktuellen Vortrages an. Dadurch ist das System nicht nur an einer Hochschule in einer Standard-Vorlesung mit einer hohen Zuhörerschaft einsetzbar, sondern ebenfalls in kleinen Übungsgruppen oder Produktvorstellungen in Unternehmen, in denen sich der Vortragende im Anschluss ein schnelles und vor allem ehrliches Feedback zu seiner Persönlichkeit und dem Vortragsinhalt selbst wünscht. Durch die Vorbereitung genau dieser Fragestellungen im Anschluss an den Vortrag erhält die vortragende Person das gewünschte Feedback. Durch den ausgewählten Use Case verschwinden alle nicht benötigten Funktionen und erleichtern so das Nutzungsverhältnis für Zuhörer sowie den Vortragenden selbst - die Anwendung erscheint schlank und leicht zu bedienen. Folgende Use Cases stehen zur Verfügung (Quibeldey-Cirkel, 2017):

- Publikumsfragen à la Günther Jauch: A|B|C|D ohne Fragetext

Die einfachste und wohl bekannteste Art und Weise ist, eine Frage mit mehreren Antwortoptionen zu stellen. Der Vortragende präsentiert eine Frage mit vier Antwortmöglichkeiten, sortiert nach den ersten vier Buchstaben des Alphabetes. Antwortmöglichkeit A repräsentiert somit die erste Antwortoption, B steht für die zweite Möglichkeit und die letzten beiden Wahlmöglichkeiten entsprechen dem gleichen Prinzip. An dieser Stelle möchte man von den Zuhörern für eine repräsentative Umfrage lediglich einen der vier Buchstaben als Antwort bekommen. Der Gewinn besteht darin, dass die Abstimmungsgeräte immer die gleichen sein

können, da sich die Aufschrift (A, B, C und D) nicht ändert. Dies kann natürlich auch durch eine Software widergespiegelt werden, in diesem Fall bietet ARSnova den Teilnehmern nur die vier Optionen an und dem Vortragenden einen Graphen mit einer Zusammenfassung der Ergebnisse.

- Quizfragen in den Formaten Multiple- und Single-Choice, Ja|Nein

An dieser Stelle stehen mehrere Möglichkeiten der Fragestellung zur Verfügung. Es können Umfragen gestellt werden (keine richtigen Antworten), Single- oder Multiple-Choice-Fragen. Ebenfalls gibt es bereits vordefinierte Frageformate wie die Ja|Nein-Frage: Eine Single-Choice-Frage mit zwei Antwortoptionen, bei denen die beiden Texte bereits vorgegeben sind (Ja und Nein). Die Anzahl der Fragen, Antwortoptionen, die jeweiligen Texte, ob Enthaltungen erlaubt sind oder Lösungshinweise gegeben werden sollen, ist frei konfigurierbar.

- Zwischenfragen & Kommentare (Kummerkasten)

Der Anwendungsfall “Kummerkasten” baut einen Sprachkanal von der Zuhörerschaft zum Vortragenden auf, der immer offen steht. So können jederzeit Kommentare und Fragen an den Vortragenden gestellt werden. Dieser entscheidet selbst, wann und ob die Einwürfe bearbeitet werden. Bei wiederkehrenden Veranstaltungen, beispielsweise bei einer wöchentlichen Vorlesung, können die Anmerkungen auch außerhalb der eigentlichen Vorlesungszeit erfolgen. Hier wird nicht nur eine Möglichkeit für den Zuhörer geschaffen, seine Aussagen oder Fragen anonym und schnell zu veröffentlichen, sondern auch dafür gesorgt, dass Fragen nicht vergessen werden, wenn sie nicht in der gleichen Veranstaltung behandelt werden konnten. Auch können bestehende Foliensätze anhand der Verbesserungsvorschläge angepasst werden, da die Anmerkungen auch ein halbes Jahr später, bei der Überarbeitung eines Vorlesungsskriptes, zur Verfügung stehen.

- Live Feedback (Verständnisbarometer)

Mit dem Live-Feedback kann der Vortragende schnell eruieren, welcher Teil der Zuhörerschaft nicht mehr zuhört, für welchen Teil er zu langsam ist und für wen zu schnell. Auch kann angegeben werden, dass das Tempo genau richtig ist. Dadurch kann ein kurzer Blick auf das Display schon verraten, ob die Zuhörerschaft einem noch folgt oder der Inhalt noch ein weiteres Mal erklärt werden muss. Es ist schwierig, bei einer Frage mit Handzeichenabstimmung zwischen den Zuständen zu differenzieren. Sollte man darum bitten dass sich alle melden, die der Vorlesung noch folgen, weiß man nicht ob die restlichen kein Interesse haben, man diese abgehängt hat oder man so langsam war, dass man viele gelangweilt hat. Ebenfalls muss beachtet werden, dass die Anonymität des Systemes zu einem ehrlichen Meinungsbild führt. Manchmal traut sich ein Student nicht, mit einem Handzeichen zu bestätigen, dass er dem Vortragenden gedanklich nicht folgenden kann: Es ist ihm peinlich. Das Verständnisbarometer gibt Ausschluss darüber, in welchen Zustand das Auditorium sich befindet - ohne lange Auszählung.

- Evaluationsfragen mit fünfstufiger Likert-Skala und Auswertung

Die Likert-Skala ist ein bewährtes Mittel zur Evaluierung und Auswertung. Da gerade Vortragende im Anschluss an eine Präsentation ein vergleichbares Feedback wünschen, ist die fünfstufige Likert-Skala eine verbreitete und anerkannte Möglichkeit, eine solche Funktion in einem ARS zur Verfügung zu stellen (Wuensch, 2015).

- Lernkarten für das Selbststudium: Session als Lernkartei

Im vorherigen Abschnitt wurde bereits erläutert, dass Lernkarten in Verbindung mit dem Leitner-Algorithmus eine wissenschaftlich belegte Methode sind, um sich Wissen auch langfristig anzueignen. Da der Inhalt dieser Lernkarten vor dem Lernen auch verstanden werden muss, ist es sinnvoll, diese in einem Auditorium gemeinsam durchzugehen. Die Karten stehen der Zuhörerschaft dann direkt in dem besprochenen Format zur Verfügung. Sollte sich, beispielsweise aufgrund eines Eintrages im Kummerkasten, noch etwas an dem Inhalt einer Lernkarte ändern, bekommt der Rest des Kurses sofort den aktuellsten Inhalt zur Verfügung gestellt.

- Interaktiver Vortrag

Der interaktive Vortrag ist die Kernkomponente von arsnova.voting. Sie stellt eine Vortragsoberfläche zur Verfügung. In die Folien selbst können teilweise schon erwähnte Inhalte mit eingegliedert werden, zum Beispiel:

- Single-Choice-Frage
- Multiple-Choice-Frage
- Ja|Nein-Frage
- Freitext-Frage
- Evaluation
- Benotung
- Hot-Spots

Das Verwenden von verschiedenen Inhalten wie bereits erstellten Folien, Bildern, Videos oder LaTeX-Auszeichnungen ist ebenfalls möglich. Neben diesem aktiv innerhalb des Vortrages eingesetzten Medium steht der “Kummerkasten”, die Lernkartei, das Live-Feedback und die optionale Erstellung von Vorbereitungsaufgaben zur Verfügung.

- Eigene Funktions- und Formatauswahl

Neben dem Auswählen eines einzelnen Anwendungsfalles aus der oberen Liste kann auch eine beliebige Kombination der Funktionen erfolgen. Wählt man die Option “Eigene Funktions- und Formatauswahl” erhält man eine Liste aller Funktionen, unter denen man jede einzelne an- und abwählen kann:

- Hauptfunktionen
  - \* Hörsaalfragen
  - \* Vorbereitungsaufgaben
  - \* Lernkarten
  - \* Fragen & Kommentare der Studierenden
  - \* Live Feedback
  - \* Vortragsfolien
- Optionale Funktionen
  - \* 2-Runden-Abstimmungen
  - \* Lernstandsberechnung



### 2.3.2 arsnova.click

Die oben beschriebenen Beispiele zum Einsatz von arsnova.voting haben eines gemeinsam: Sie werden in einen ernsten, also einen wissenschaftlichen oder unternehmerischen Zusammenhang eingesetzt. Dies stellt an die Software die Ansprüche, seriös aufzutreten und didaktisch korrekte und wertvolle Ansätze zu liefern. Adjektive wie verspielt, bunt oder laut sollen nicht in Assoziation gebracht werden. Doch gibt es auch hierfür didaktisch begründete Anwendungsszenarien. Sei es nun eine junge Zuhörerschaft oder höhere Altersgruppen, die verspielt aufgeweckt werden sollen, ohne dass diese gleich das Gefühl haben, “arbeiten” zu müssen oder alles richtig zu beantworten. Hierfür eignet sich das bedeutend jüngere arsnova.click, in dem wie bereits beschrieben viele Aspekte der Gamifizierung angewendet werden, besser.

In arsnova.click stehen die ARS-Funktionen nur sehr begrenzt zur Verfügung, da die Anwendung gezielt für kleine Quizrunden zwischendurch eingesetzt werden soll und nicht als unterstützende Software für die gesamte Dauer des Vortrages. Folgende Frageformate werden dabei unterstützt:

- Single-Choice-Frage
- Multiple-Choice-Frage
- Ja-Nein-Frage
- Wahr-Falsch-Frage
- Schätzfrage
- Kurzantwort-Frage
- Umfrage

## 2.4 Microsoft Office PowerPoint

Microsoft Office PowerPoint ist eine Software zur Erstellung und dem Halten von visuellen Präsentationsinhalten. Das Programm wird als Teil der Office-Suite vertrieben, welches neben anderen Services Teil von Microsoft Office 365 ist.

Durch die weite Verbreitung und viele Funktionalitäten, die eine erhebliche Unterstützung bei Vorträgen darstellen, ist diese Software schon lange und mit Abstand das meist verwendete Präsentationstool. Nach einer Umfrage mit über 1.300 Personen, die alle regelmäßig Vorträge halten und den verschiedensten Berufsgruppen angehörten, kommt PowerPoint mit 82.8 % zum Einsatz (Thielsch & Förster, 2007, S. 4). Durch die geringe Innovation bei den Präsentationstools, die auf die bereits vollkommene Unterstützung eines Vortragenden zurückzuführen ist, wird keine ausschlaggebende Abweichung der Umfragewerte von damals zu heute erwartet.

## 2.5 Technologien

Heutzutage gibt es nicht die eine Technologie, mit deren Hilfe eine Softwarelösung für ein Problem entwickelt werden kann. Vielmehr gilt es zu evaluieren, welche der gebotenen Möglichkeiten zu den besten Ergebnissen führen. Dieser Teil findet in der Analyse statt, die Ergebnisse werden für ein besseres Verständnis bereits hier erläutert. Eine Kurzvorstellung der Alternativen findet ebenfalls in der Analyse statt, hier werden diese nicht näher ausgeführt, da sie keinen direkten Einfluss auf die Arbeit haben.

### 2.5.1 VSTO

Die Visual Studio Tools for Office (VSTO) sind eine Reihe von Entwicklungs-Werkzeugen von Microsoft, welche in Form eines Visual Studio Add-ins, also einer Entwicklungsvorlage, zur Verfügung stehen. Sie erlauben es, Anwendungen in Office zu integrieren. Dabei werden die mit den von VSTO erstellten Applikationen von der jeweiligen Office-Anwendung gehostet, sind also nicht eigenständig ausführbar. Um andere Applikationen ausführen zu können, wird das Common Language Runtime (CLR) benötigt. Sie ist zentraler Bestandteil des .NET-Frameworks und die Laufzeitumgebung, in der .NET-Anwendungen ausgeführt werden. Das CLR stellt Microsoft Office seit der Version 2003 zur Verfügung. Damit ist die Verwendung eines VSTO-Add-ins seit der Version 2003 möglich (Rahman, 2014, S. 16).

Neben den Funktionalitäten der Microsoft-Office Palette, die das Framework in Form von Members, Methods, Properties und Events zur Verfügung stellt, ist es ebenfalls möglich, die Benutzeroberflächen zu erweitern. Bestehende Oberflächenelemente oder Funktionen können dabei nicht manipuliert werden.

### 2.5.2 WPF

Die Windows Presentation Foundation (WPF) ist die aktuelle Schnittstelle für grafische Benutzeroberflächen unter Windows und hat den Vorgänger, Windows Forms, mit der Version 3.0 abgelöst. Für die Darstellung verwendet WPF DirectX anstelle des veralteten Graphics Device Interface (GDI). Durch die Verwendung der DirectX-Schnittstelle kann die teilweise enorme Leistung der Grafikkarten-Chips genutzt werden, um Windows-Anwendungen flexibel und performant zu gestalten. Ein weiterer Vorteil besteht darin, dass sich mit DirectX dargestellte Control-Elemente durch Styles und Templates im Erscheinungsbild frei anpassen lassen. In früheren Versionen waren diese nicht anpassbar, welches zu dem typischen und immer gleichen Erscheinungsbild von älteren Windows-Anwendungen führte. Auch die vektorbasierte Definition der Elemente durch WPF ist wichtig für heutige Anwendungen: Sie sind voll skalierbar und somit auf jedem Gerät ansehnlich darstellbar (Huber, 2015, S. 51-53).

### 2.5.3 XAML

Extensible Application Markup Language (XAML) ist die Beschreibungssprache, mit der die grafischen WPF-Elemente definiert werden. Wie in der Namensbezeichnung selbst beschrieben, basiert XAML auf Extensible Markup Language (XML). Dabei beschreibt XAML die Elemente rein deklarativ und trennt somit Aussehen und Funktion der Elemente, ähnlich dem Prinzip von Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS). Dies sorgt im Code für eine wesentlich kompaktere und verständlichere Beschreibung einer Oberfläche.

Der nächste Vorteil von XAML ergibt sich aus der Abschottung zum restlichen Code. C#- oder Visual Basic (VB)-Code muss, wie in fast allen Sprachen üblich, in eigenen Dateien mit speziellen Endungen vorliegen, damit der Compiler mit diesen arbeiten kann. Dadurch wird gewährleistet, dass die Geschäftslogik und die Oberflächenbeschreibung zumindest im Speicherort getrennt sind. Die dadurch gewonnene Aufgabenteilung kann dafür genutzt werden, die Oberflächengestaltung mit Tools wie Expression Blend von Usability-Experten vornehmen zu lassen, während sich die Softwareentwickler auf die Anbindung und Geschäftslogik konzentrieren können.

### 2.5.4 Meteor

Durch die Anpassungen an die bestehende arsnova.click-Applikation wird neben den .NET-Technologien auch mit dem Meteor-Framework gearbeitet. Meteor ist ein privat wie auch kommerziell kostenlos nutzbares Web-Framework, welches durch seine breit aufgestellten Komponenten ein einfaches Entwickeln von reaktiven Webanwendungen ermöglicht. Für das wirklich einfache Entwickeln ist das bereits implementierte Zusammenspiel von vielen Komponenten verantwortlich (Dascalescu, 2016a). Die wichtigsten sind dabei:

- MongoDB, Express, Angular und Node.js (MEAN)-Stack
- Socket.IO
- grunt/gulp
- Cordova
- hot code reload

Oft wirken andere Web-Frameworks wie viele einzelne Komponenten, die aneinander angebunden werden müssen. Bei dem Meteor-Framework arbeiten diese bereits alle gemeinsam, ein einziger Befehl genügt zum Herunterladen der benötigten Pakete, dem Kompilieren der Anwendung sowie dem Bereitstellen auf einem lokalen Webserver. Sogar das direkte Updaten von geändertem Code, dem hot code reload, funktioniert out-of-the-box. Dies entspricht einem der Grundprinzipien der Meteor-Entwickler: “Einfachheit heißt Produktivität”. Damit ist das Framework auch für Anfänger geeignet. Folgende Aspekte tragen zu dem Erfolg der einfachen Anwendung bei Dascalescu (2016b):

- Das Einsteiger-Tutorial beansprucht lediglich eine Stunde und das Ergebnis ist bereits eine reaktive und voll funktionsfähige TODO-Anwendung.
- Alles ist mit einer Sprache zu entwickeln: JavaScript (JS). Auf dem Server läuft JS durch Node.js und Underscore, auf dem Client durch jQuery.
- Client und Server bedienen sich der gleichen MongoDB-Schnittstelle, daher ist es nicht nötig, neuen Model-Code zu lernen.
- Eine gute Dokumentation.
- Das Buch “Discover Meteor” von Tom Coleman & Sacha Greif beinhaltet alle wichtigen Informationen zu Meteor - andere Werke werden nicht benötigt.
- Die Community auf GitHub und StackOverflow sprechen für sich - das Meteor-Projekt ist mit Abstand eines der aktivsten Web-Frameworks.
- Neben der eigenen Blaze-View können auch React oder Angular verwendet werden.

Dabei gibt es nur wenige Probleme, die Meteor mit sich bringt (C., 2015):

- Meteor ist eine Client-Server-Plattform und damit nicht geeignet für Web-Services oder eine Representational State Transfer (REST) Application Programming Interface (API).
- Das Framework ist sehr hardwareintensiv - dies ist vor allem dem Wunsch der Echtzeit und Reaktivität geschuldet. Sollten diese beiden Aspekte nicht benötigt werden, ist Meteor auf keinen Fall empfehlenswert.
- Der Wartungsaufwand ist momentan aufgrund der geringen Zeitspannen zwischen den Releases sehr hoch. Oft sind Pakete nicht kompatibel oder die bestehende Software hat Fehler. Selbst wenn dies nicht der Fall ist, sollte vor einem Framework-Update immer ein ausführlicher Test erfolgen. Da Meteor direkt alle Komponenten der Web-Applikation bereitstellt, müssen die Tests hier besonders gründlich durchgeführt werden.
- Die Fehlerausgaben sind nicht immer aussagekräftig. Gerade durch die vielen einzelnen Komponenten des Frameworks und deren Abhängigkeiten untereinander gibt es oft viele verschiedene Ausgaben von mehreren Komponenten, obwohl Blaze lediglich einen Template-Namen nicht auflösen konnte. Oft hilft hierbei nur die Verbose-Fehlernachricht - sollte diese generischer Natur sein, bleibt nur das Raten und Ausprobieren. Dieses Problem bleibt in der Regel nur den Meteor-Einsteigern, im Laufe der Entwicklung lernt man, zwischen den Fehlermeldungen zu lesen und deren Bedeutungen zu erkennen.

Alles in Allem ist Meteor eines der besseren, neuen Web-Frameworks. Es ist nicht modellgetrieben, einfach zu entwickeln und reaktiv, gerade kleine Anwendungen sind hier schnell fertiggestellt. Bei größeren Anwendungen, bei denen die Performance durch das gleichzeitige Verwenden vieler Nutzer und serverseitige Funktionalitäten wichtig sind, gibt es andere empfehlenswerte Frameworks.

## 3 Analyse

Die Analyse beschäftigt sich mit den Anforderungen an das zu entwickelnde System und deren Umsetzbarkeit. In diesem Fall werden zwei Seiten der Problemstellung betrachtet. Zum einen gilt es, die Anforderungen aus der Sicht des Anwenders in Form von Anwendungsfällen zu beschreiben. In den entsprechenden Gliederungspunkten findet gleichzeitig die zugehörige Oberflächenbeschreibung statt. Dieser Teil hat lediglich die Aufgabe, ein Verständnis für die späteren Workflows (dt.: Arbeitsabläufe) in PowerPoint zu erlangen.

Im nächsten Teil dieses Kapitels geht es um die Anforderungen an die externen ARS und das Integrationsframework. Diese müssen, um eine Integration in PowerPoint überhaupt zu ermöglichen, zwingend erfüllt werden. Nach der Definition der Anforderungen werden die genauen Anpassungen an den beiden ARS und der Integrationslösung in dem Kapitel Konzeption aufgeführt.

### 3.1 Einstiegspunkte

Da es sich bei dieser Softwarelösung um eine vollständige Integration, wie in dem gleichnamigen Kapitel auf Seite 10 definiert, handelt, muss spezifiziert werden, an welchen Einstiegspunkten eine Anknüpfung an den PowerPoint-Workflow stattfinden soll. Hierbei kann es zur Optimierung der Bedienung durchaus vorkommen, dass eine Funktion von zwei verschiedenen Einstiegspunkten heraus aufrufbar sein soll.

An den folgenden Stellen soll die Integrationslösung in PowerPoint erreichbar sein:

- Im Menüband
- In bestimmten Kontextmenüs
- Während der Präsentation

Nachfolgend sind die Einstiegspunkte genau spezifiziert. Da deren Möglichkeiten von dem Framework vorgegeben werden, wurde an dieser Stelle bereits vorgegriffen: Es werden nur die Einstiegspunkte beschrieben, die die VSTO auch unterstützen.

#### 3.1.1 Menüband

Das Menüband, auch unter der englischen Übersetzung “Ribbon Bar” bekannt, ist die Haupt-Funktionsleiste in Microsoft Office. An dieser Stelle werden die verschiedenen Bearbeitungsbereiche der jeweiligen Office-Applikation aufgelistet. In PowerPoint sind dies zum Beispiel Menüpunkte wie “Datei”, “Einfügen”, “Animation” oder “Bildschirmpräsentation”. Die Menübänder kapseln also für eine erhöhte Übersicht Funktionsbereiche. Da die Integration der beiden ARS nicht nur wenige, sondern einen ganze Menge an Funktionalitäten in PowerPoint einbinden möchten, ist es unabdingbar, diese inklusive zusätzlicher Verwaltungs- und Informationsmöglichkeiten in ein eigenes Menüband einzugliedern:



Abbildung 1: Das PowerPoint-Menüband mit dem ARSnova-Eintrag

Wenn lediglich ein kleiner Funktionsumfang hinzugefügt werden sollte, wäre es sinnvoll, die bestehenden Menübänder um diese zu ergänzen. Ein Menüband mit nur wenigen Funktionsangeboten wirkt umständlich und schwer zu erreichen. Auch dies ermöglichen VSTO, lediglich das Entfernen bestehender Einträge oder das Verändern derer Funktionalität wird unterbunden.

### 3.1.2 Kontextmenüs

Ein Kontextmenü ist eine Auflistung von Funktionen, welche durch einen Rechtsklick auf eine Oberfläche aktiviert, beziehungsweise ausgefahren wird. In der Regel gelten die Funktionen speziell für das Element, auf welches geklickt wurde. Über das Anwählen einer beliebigen, anderen Oberfläche als des Kontextmenüs selbst wird dieses wieder deaktiviert.

Da es gerade in den Office-Applikationen aufgrund der vielen verschiedenen Oberflächen-Elemente auch viele verschiedene Kontextmenüs gibt, gilt es hier genau zu bestimmen, für welche Oberflächenelemente eine Kontextmenü-Erweiterung effektivitätsfördernd ist. Zu viele Einträge verwirren den Anwender, fehlende könnten den Trugschluss erzeugen, dass es an dieser Stelle keine Möglichkeit gibt, eine ARSnova-bezogene Interaktion mit dem Element durchzuführen. Dass alle Funktionen ebenfalls immer im Menüband hinterlegt sind und ein Kontextmenü lediglich eine Art “Schnellzugriff” für auf dem angewählten Element oft ausgeführte Aktionen darstellt, ist nicht jedem Benutzer bewusst.

Da die Integrationslösung immer komplette Folien mit den eigenen Inhalten erzeugt beziehungsweise anpasst, muss das Folien-Kontextmenü um sinnvolle, ARSnova-spezifische Funktionalitäten erweitert werden. Die zur Verfügung gestellten Aktionen sollen dabei über einen separaten Kontextmenü-Eintrag gekapselt sein, damit den Anwendern klar ersichtlich ist, welche Interaktionen ARSnova-spezifisch sind:

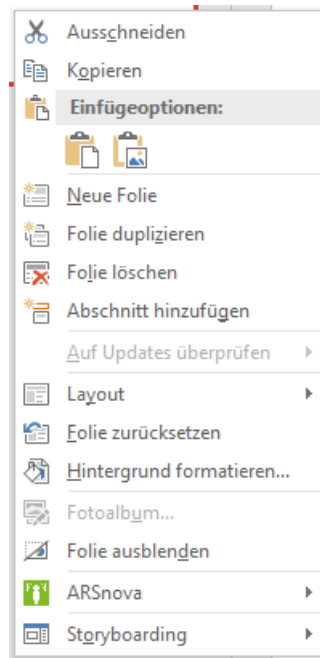


Abbildung 2: Das Kontextmenü einer PowerPoint-Folie

Das Erstellen neuer Kontextmenüs ist mit den VSTO nicht möglich. Wirklich benötigt werden diese allerdings auch nicht, da die Menüs bereits für fast alle Oberflächenelemente existieren.

### 3.1.3 Präsentation

Die beiden vorherigen Abschnitte haben sich hauptsächlich einer Form der Integration gewidmet: Eine bereits existierende Anwendung um neue Funktionalitäten zu erweitern. Ob diese nun über ein bestehendes Menüband oder einen Kontextmenü-Eintrag aufgerufen werden, ist nur ein Ergebnis des Versuches, dem Nutzer an den richtigen Stellen die benötigten Möglichkeiten zur Verfügung zu stellen. Allerdings ist es das Ziel einer Integration, auch in die bestehenden Workflows einer Software einzugreifen, statt nur neue und eigenständige Vorgehensweisen hinzuzufügen. Ansonsten wäre dies lediglich eine Erweiterung, die auch in einer separaten Anwendung realisiert werden könnte.

Die VSTO ermöglichen den Eingriff in bestehende Arbeitsabläufe durch Event-Handler. Diese werden in der Regel bei dem Auftreten eines bestimmten Events aktiv und rufen alle Methodendefinitionen, die ihnen vorher zugeteilt wurden, auf. Dabei gibt es nicht die Möglichkeit, andere, ebenfalls aufgerufene Methoden, zu beeinflussen oder zu beenden, wie man es unter anderem von der `preventDefault()`-Methode in JS kennt. Dadurch ist es möglich, bestehende Workflows zu erweitern, beispielsweise indem einem Event-Handler, der bei dem Starten der Präsentation aktiviert wird, eine Methodendefinition hinzugefügt wird, die das Abspielen eines Start-Signals auslöst. Alle restlichen Start-Interaktionen bleiben unverändert. Die nachfolgenden, umschriebenen Events, und damit Einstiegspunkte, sind einige der wichtigsten Einstiegsmöglichkeiten für diese Integrationslösung (Microsoft, 2017a). Im entsprechenden Literaturverweis können alle zur Verfügung gestellten Events nachgelesen werden.

- Nach dem Erstellen einer Präsentation und nachdem es der Präsentations-Kollektion hinzugefügt wurde
- Nach dem Öffnen einer existierenden Präsentation und dem Hinzufügen zur Präsentations-Kollektion
- Nach dem Ändern eines Farbschemas
- Vor dem Schließen einer Präsentation
- In dem Moment vor der Schließung, in dem das Präsentations-Objekt bereits aus der Präsentations-Kollektion entfernt wurde
- Vor dem Speichern einer Präsentation
- Nach dem Erstellen einer neuen Folie und dem Hinzufügen zu der Folien-Kollektion
- Vor dem Drucken einer Präsentation
- Während der Synchronisation einer lokalen Präsentation mit der Server-Version inklusive Statusinformationen
- Bei dem Ändern der Folienauswahl
- Bei dem Starten des Präsentations-Modus

- Bei dem Beenden des Präsentations-Modus
- Vor dem Anzeigen der nächsten Folie im Präsentations-Modus

Die Verwendung so vieler Events anstelle von einer geringen Kopplung ist nötig, um das Gefühl einer vollen Integration zu schaffen, die Konsistenz der Präsentation zu wahren und möglichst viele Synchronisations-, Vorbereitungs- und Einstellungsprozesse automatisiert ablaufen zu lassen und nicht vom Nutzer initiiert.

## 3.2 Anwendungsfälle

Zu Beginn werden die einzelnen Anwendungsfälle inklusive ihrer Umsetzung beschrieben. Dabei werden die einzelnen Fälle nicht gänzlich separat betrachtet, es wird immer ein Zusammenhang zwischen den anderen Fällen und dem aktuellen hergestellt. Der Gedanke ist, dass sich später auch der Nutzer durch mehrere Ansichten klicken muss, um an eine gewünschte Stelle zu gelangen. Diese Wege sollen unter anderem auch in den Anwendungsfällen ersichtlich werden.

In dem Menüband von PowerPoint wird nach der Installation des Add-ins der Menüpunkt “ARSnova”, wie auf dem Bild in Kapitel arsnova.voting dargestellt, angeboten:



Abbildung 3: Das ARSnova Menüband

Über diesen stehen alle neuen Funktionen der ARSnova-Integration zur Verfügung. Jeder Anwendungsfall ist über einen eigenen Button gekapselt. Bei dem Darüberhalten des Mauszeigers über eine Schaltfläche erhält der Anwender in einem Tooltip ausführliche Informationen bezüglich der entsprechenden Aktion. Folgende Anwendungsszenarien müssen von der Softwarelösung unterstützt werden:

- Installation
- ARSnova App auswählen
- Verwalten von Quizfragen
- Anlegen und Durchführen
- Bearbeiten
- Ausblenden
- Löschen
- Informationen



Auffällig ist, dass nicht alle im Kapitel arsnova.voting beschriebenen Use Cases von arsnova.voting umgesetzt werden sollen. Das liegt zum einen an PowerPoint selbst, welches bestimmte Anwendungsszenarien wie den interaktiven Vortrag teilweise schon abbildet. Zum anderen liegt es auch an Funktionen, die in der Präsentation selbst nicht sinnvoll aufgehoben sind. So soll der “Kummerkasten” über den Zeitraum des Vortrages noch weiter zur Verfügung stehen, allerdings kann die Integrationslösung nur mit der Session interagieren, solange PowerPoint selbst auf einem Computer ausgeführt wird. Spätere Einträge können nur über die bestehende Web-Oberfläche hinzugefügt und abgerufen werden. An dieser Stelle ist eine Integration nicht sinnvoll, da es dem Anwendungsfall selbst nicht mehr gerecht werden würde.

Die Intention dieser Integrationslösung ist, einen invertierten Vortrag zu gestalten, also einen Sprachkanal von der Zuhörerschaft zum Vortragenden zu öffnen. Dies geschieht in Form von Fragen, die in eine PowerPoint-Präsentation eingefügt werden.

### 3.2.1 Installation

Da die Endanwendung aus mehreren Assemblies, also Dynamic Link Library (DLL)-Dateien, bestehen wird und die Installation eines VSTO-Add-ins für einen durchschnittlichen Office-Anwender etwas Neues ist, soll ein Setup zur Vereinfachung der Installation zur Verfügung gestellt werden. Aus Gründen der Einfachheit wird das Setup nur in englischer Sprache zur Verfügung stehen. In der Installationsroutine sollen lediglich die Lizenzvereinbarungen angenommen werden, danach erfolgt eine automatisierte Installation des Add-ins:

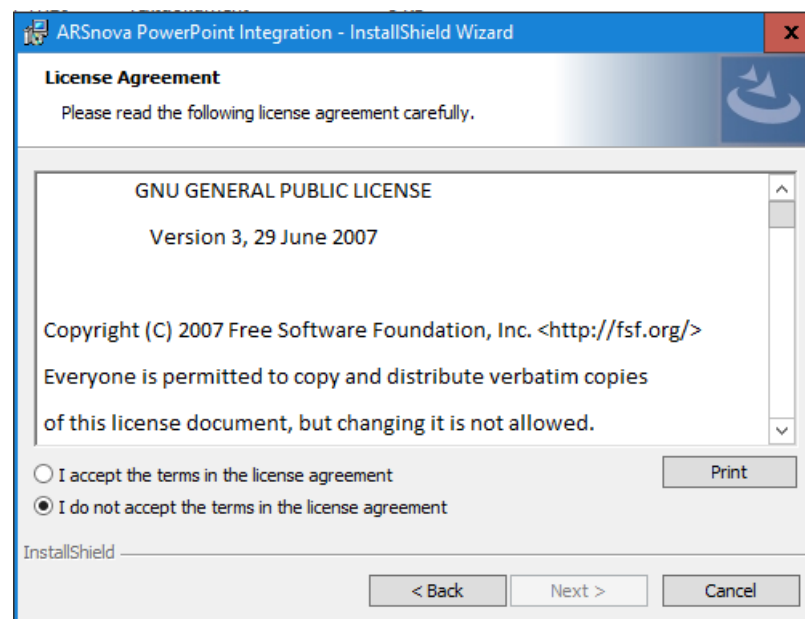


Abbildung 4: Annahme der Lizenzvereinbarungen

Im Anschluss wird die Integrationslösung in den Windows-Programmen aufgelistet und steht dort zur Deinstallation zur Verfügung, sodass sich der Nutzer auch hier nicht mit der Office-Add-in-Verwaltung auseinandersetzen muss:



Abbildung 5: Der Windows-Installationseintrag

Wenn die Installationsroutine ausgeführt wird, obwohl die Software bereits installiert ist, stehen dem Anwender anstelle der Lizenzvereinbarungen und einer erneuten Installation die Optionen der Reparatur oder der Deinstallation zur Auswahl bereit. Das Setup kann jederzeit abgebrochen werden.

### 3.2.2 ARSnova App auswählen

Bevor eine Frage zu den Folien hinzugefügt werden kann, muss ausgewählt werden, ob es sich um einen arsnova.voting- oder arsnova.click-Foliensatz handelt. Für die Zuhörerschaft ist es umständlich, während eines Vortrages in zwei verschiedenen Anwendungen abstimmen zu müssen, daher ist das Erstellen von arsnova.voting- und arsnova.click-Inhalten innerhalb einer Präsentation nicht möglich.

Um dem Anwender die Auswahl der Session-Art bewusst zu machen, wird keine einfache Vorauswahl getroffen, sondern folgendes Auswahl-Fenster vor dem Erstellen einer Frage, wenn es nicht bereits vorher geöffnet wurde, automatisch eingeblendet. Dies geschieht pro Präsentation nur einmal, bei der folgenden Fragestellung wird davon ausgegangen, dass sich der Nutzer der Systemauswahl bewusst ist. Nach dem Anwählen des Buttons “ARSnova App auswählen” in der Menüleiste öffnet sich folgendes Fenster:

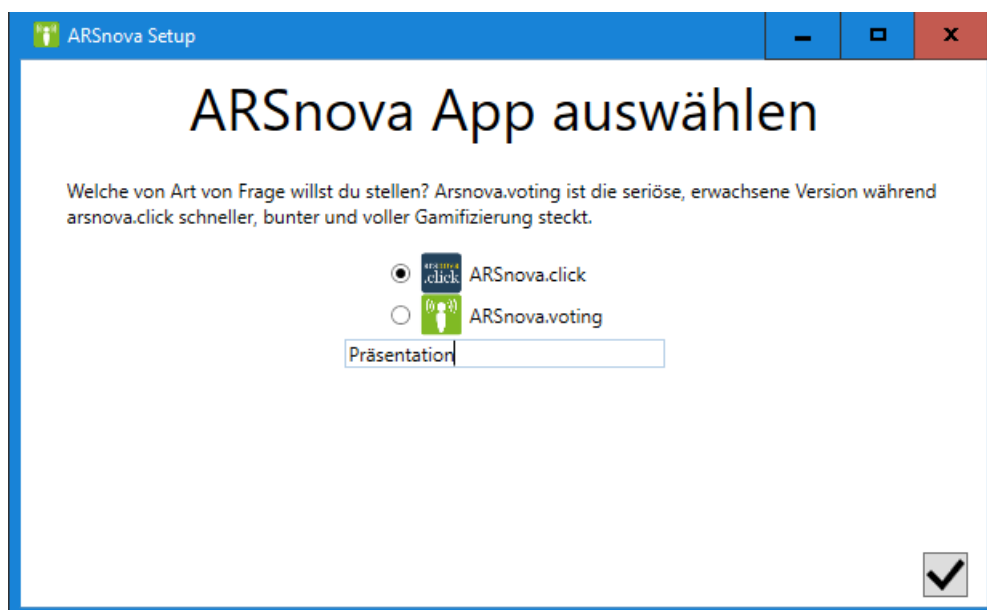


Abbildung 6: Das Fenster zur Auswahl des zu verwendenden ARS

Bei der Auswahl von arsnova.click erscheint das untere Textfeld, in dem der Name für die Quizrunde festgelegt wird. Bei arsnova.voting wird diese Option ausgeblendet, da bei dieser Auswahl die Session-ID automatisch vergeben wird. Stattdessen werden hier jeweils ein Feld zum Festlegen des Namens und des Kurznamens der Session angeboten. Während an dieser Stelle keine Werte vorbelegt sind, wird bei der gamifizierten Variante standardmäßig das Text-Eingabefeld mit dem Namen der Präsentation vorbelegt. Wenn dieser bereits verwendet wird, also nicht mehr verfügbar ist, werden hinter dem vergebenen Namen ganze Zahlen hochgezählt, bis ein noch nicht verbogener Name gefunden wurde. Sollte ein bereits verbogener Name eingetippt werden oder die Zeichenlänge liegt nicht zwischen drei und fünfundzwanzig Zeichen, wird das Textfeld rot hinterlegt und erhält bei dem Darüberhalten des Mauszeigers eine entsprechende Fehlermeldung:

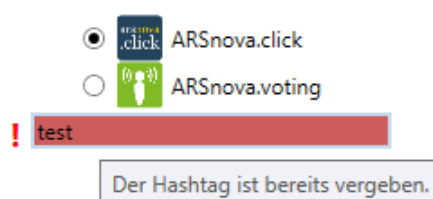


Abbildung 7: Eine nicht erfolgreiche Validierung des Hashtag-Feldes

Das Fenster kann über den Schließen-Button rechts oben jederzeit geschlossen werden, in diesem Falle erscheint ein entsprechender Warnhinweis und die Änderungen werden nicht übernommen. Die Möglichkeit, das aktuelle Fenster zu minimieren, maximieren oder zu schließen (mit einem Warnhinweis) besteht in jedem Fenster dieser Integrationslösung und wird im Folgenden nicht mehr explizit erwähnt. Die Speichern- und Schließen-Aktion durch den Button rechts unten wird erst erfolgreich ausgeführt, wenn die Validierung erfolgreich ist. Falls nicht, wird der Nutzer mit einem Warnhinweis auf die nicht korrekten Felder hingewiesen und das Fenster bleibt nach dem Bestätigen der Meldung geöffnet.

Wenn der Session-Typ nach der Erstellung von Fragen geändert wird, werden alle bestehenden Fragen, Antwortoptionen und zugehörige Folien gelöscht. Der Nutzer wird darauf mit einer entsprechenden Warnmeldung hingewiesen.

### 3.2.3 Verwalten von Quizfragen

Der zweite Button in dem ARSnova-Menüband mit der Beschriftung “Fragen verwalten” bietet die Möglichkeit, sich alle bestehenden Fragen in der aktuellen Präsentation anzusehen und diese zu bearbeiten oder zu löschen. Das Anlegen neuer Fragen steht an dieser Stelle nicht zur Verfügung, da dazu immer ein Einhängpunkt angegeben werden muss. Das bedeutet, dass der Anwender davon ausgeht, dass eine neue Folie immer hinter der aktuell angewählten eingefügt wird. Sollte sich der Nutzer in einem zusätzlichen Fenster wie der Quizfragen-Verwaltung befinden, dann ist er sich der zur Zeit selektierten Folien nicht bewusst und könnte von der Einfügungsposition der neuen Folien verwirrt sein.

Nach dem Anwählen des Buttons öffnet sich folgendes Fenster:

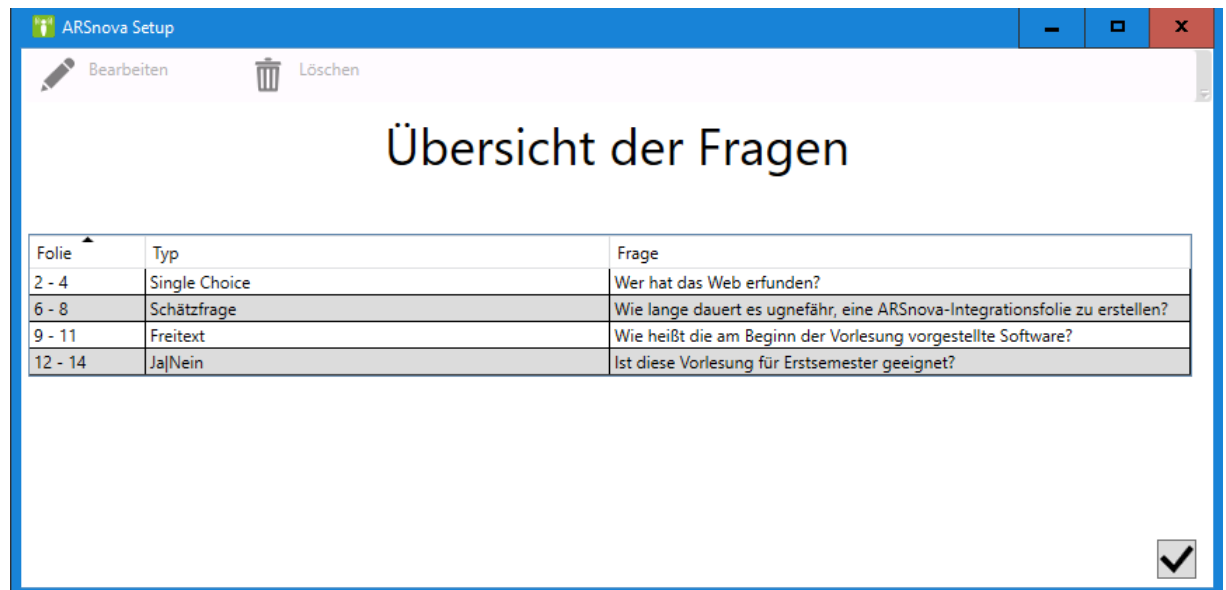


Abbildung 8: Eine Übersicht über alle Fragen der aktuellen Präsentation

Die Lösungs- und Bearbeitungsinteraktionen werden folgend in den gleichnamigen Kapiteln beschrieben. Durch den Bestätigungs-Button rechts unten im Fenster oder den Schließen-Button rechts oben wird das Fenster geschlossen. Das PowerPoint-Fenster kann nicht angewählt und die Präsentation somit nicht bearbeitet werden, während das Verwaltungsfenster geöffnet ist. Dadurch wird verhindert, dass Inhalte oder Folienreferenzen, mit denen das Verwaltungsfenster zu dem Zeitpunkt arbeitet, ungültig werden.

### 3.2.4 Anlegen und Durchführen

Eine neue Quizfrage kann auf zwei verschiedene Arten angelegt werden. Zum einen ist das Anlegen über den Menüband-Eintrag “Füge ein Quiz auf neuen Folien ein” möglich, zum anderen kann über das Folien-Kontextmenü der gleichnamige Eintrag gewählt werden. Sollte die Session-Art noch nicht festgelegt worden sein, öffnet sich, wie im Abschnitt ARSnova App auswählen beschrieben, das Fenster, um diesen auszuwählen. Anschließend wird der Anwender durch eine zweistufige Installationsansicht geführt, um alle Einstellungen zu der Frage vorzunehmen:

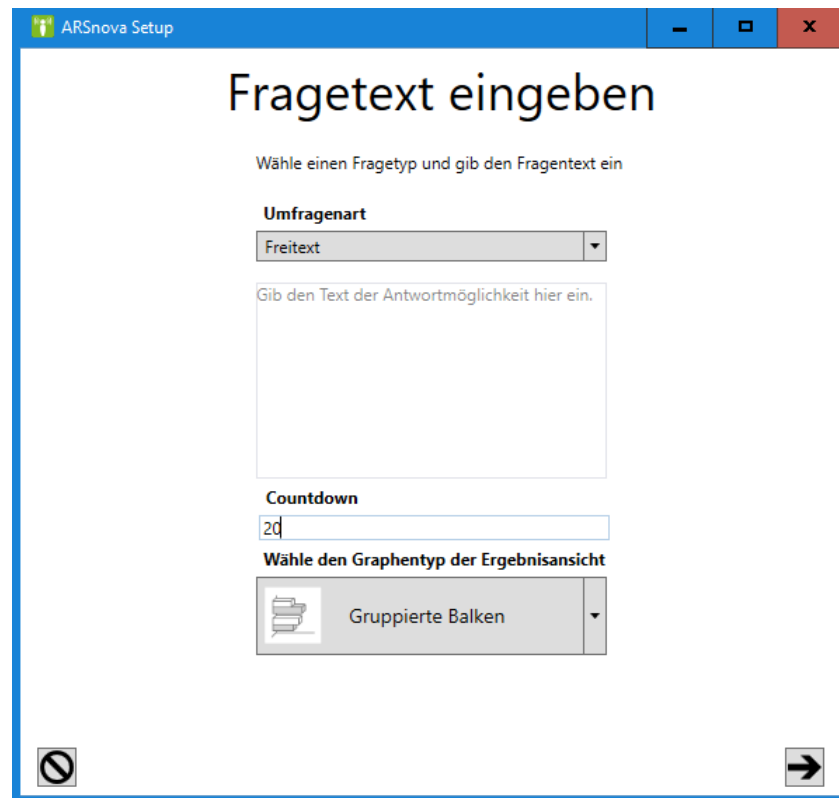
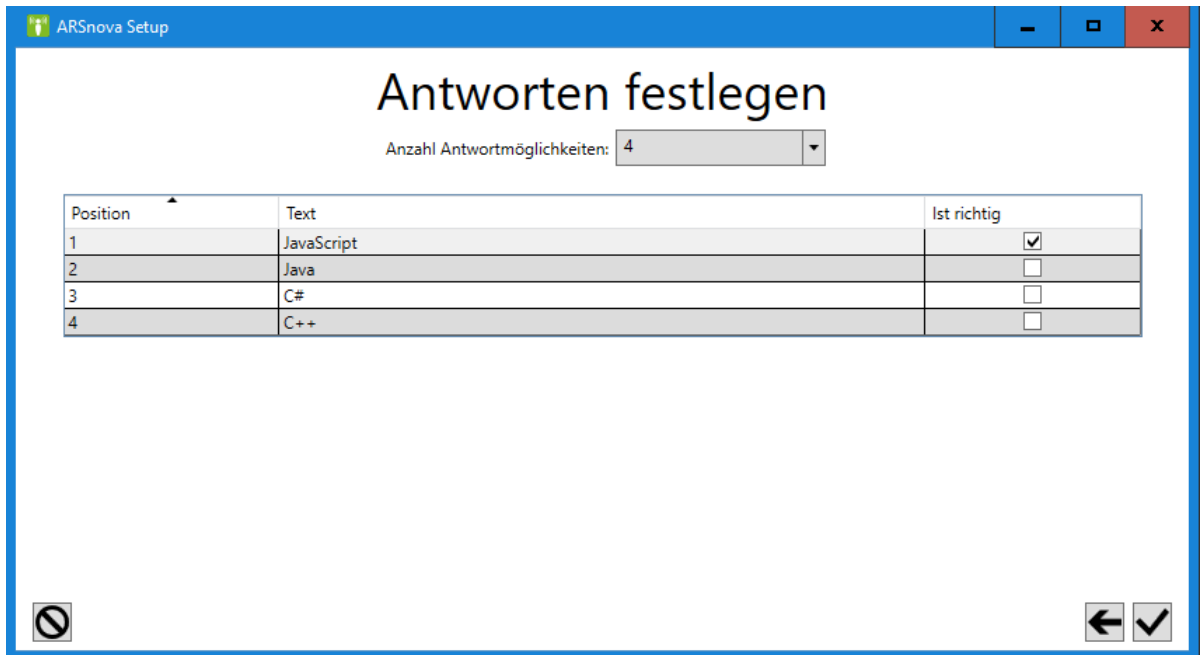


Abbildung 9: Erstellen einer Frage: Teil 1

In dieser ersten Ansicht wird der Fragetyp, dessen Text und die gewünschte Darstellung des Graphen gewählt. Sollte arsnova.click als Session-Art gewählt worden sein, muss ebenfalls ein Countdown in Sekunden angegeben werden. In diesem Feld ist ein numerischer Wert größer fünf einzutragen. Die Umfragearten unterscheiden sich je nach dem gewähltem System, folgende Auswahl steht zur Verfügung:

- arsnova.voting
  - Single Choice
  - Multiple Choice
  - Ja|Nein
  - Freitext
  - Evaluation
  - Benotung
- arsnova.click
  - Single Choice
  - Multiple Choice
  - Ja|Nein
  - Wahr|Falsch
  - Schätzfrage
  - Freitext
  - Umfrage

Als Graphen-Art stehen ein Balken- und ein Säulendiagramm zur Verfügung. Die Graphen werden jeweils in einer 3D-Ansicht dargestellt. Über den Abbrechen-Button links unten sowie den Schließen-Button rechts oben kann die Fragerstellung in beiden Ansichten jederzeit abgebrochen werden. Die bis dahin erstellten Inhalte dieser einen Frage werden nicht gespeichert. Über den Weiter-Button rechts unten kann, wenn die Validierung der Eingabefelder erfolgreich ist, in die zweite Ansicht gewechselt werden. Neben dem Countdown-Feld wird auch überprüft, ob ein Fragentext eingegeben wurde. Bei einer nicht erfolgreichen Validierung wird eine Meldung mit entsprechenden Warnhinweisen eingeblendet und der Wechsel in die zweite Ansicht ist nicht möglich.



ARSnova Setup

## Antworten festlegen

Anzahl Antwortmöglichkeiten: 4

Position	Text	Ist richtig
1	JavaScript	<input checked="" type="checkbox"/>
2	Java	<input type="checkbox"/>
3	C#	<input type="checkbox"/>
4	C++	<input type="checkbox"/>

Navigation: [Previous] [Next]

Abbildung 10: Erstellen einer Frage: Teil 2

In dieser zweiten Instanz werden die Antwortoptionen für die Quizfrage festgelegt. Je nach Umfrageart unterscheidet sich diese Ansicht. Bei einer Schätzfrage werden drei Eingabefelder für den richtigen, den niedrigsten sowie den höchsten Eingabewert angeboten. Bei einer Umfrage gibt es keine richtigen oder falschen Antworten, daher fehlt hier die Tabellenspalte "Ist richtig". Für Freitext-Fragen werden verschiedene, spezielle Konfigurationsmöglichkeiten zur Verfügung gestellt:

Abbildung 11: Festlegen der Antwortoptionen: Typ Freitext-Frage

Über den Zurück-Button rechts unten kann in den ersten Teil der Fragerstellung gewechselt werden, beispielsweise um die Umfrageart zu ändern. Sollte dabei zwischen nicht kompatiblen Antwortoptionen gewechselt werden, also zwischen einer Schätzfrage, einer Freitextfrage oder einem beliebigen anderem Typ, wird der Nutzer darüber informiert, dass die Änderung des Umfragetyps eine Löschung der bisherigen Antwortoptionen bedeutet. Über den Bestätigen-Button wird die Frage, nach einer Validierung der Antwortoptionen, erstellt. Die Validierung beinhaltet dabei:

- Jede Antwortoption muss einen Text enthalten
- “Ist Richtig”-Feld: Eine richtige Antwort für Single-Choice-Fragen, mindestens zwei für Multiple-Choice-Frage
- Schätzfrage: Der Minimum-Wert muss gleich oder unter dem richtigen und maximalen Wert liegen, der maximale gleich oder über dem richtigen Wert

Nach dem erfolgreichen Anlegen einer neuen Quizfrage werden nach der vorher selektierten Folie drei neue Folien bei einer arsnova.click Umfrage eingefügt, zwei bei der arsnova.voting-Version. Sollte die vorher selektierte Folie bereits eine Quiz-Folie gewesen sein, wird sichergestellt, dass die anderen beiden damit zusammenhängenden Folien auch weiterhin zusammenhängend bleiben. Sollte beispielsweise vorher die Folie Nummer drei angewählt gewesen sein, dies aber gleichzeitig auch die zweite Folie einer anderen Quizfrage sein, die sich über die Folien zwei bis vier erstreckt, werden die neuen Quizfolien an der Position fünf eingefügt, nicht an der Stelle vier.

Die erste Folie beinhaltet in beiden Fällen die Frage mit den Antwortoptionen, diese wird vor dem Starten des Quizzes eingeblendet. An dieser Stelle werden dem Auditorium die Frage und deren Antwortoptionen auf verständliche Art und Weise vorgestellt. In arsnova.voting wird die Frage gleichzeitig freigeschaltet, um den Zuhörern das Abstimmen zu ermöglichen. Bei arsnova.click wird die Frage erst später freigegeben, da der Countdown nur ein begrenztes Abstimmungsfenster zulässt. Davor muss das Publikum die Frage vollständig erfasst haben. Hierfür ist die zusätzliche und zweite Folie vorgesehen. Diese blendet den gleichen Inhalt der vorherigen ein, allerdings erweitert

um die Anzeige des Countdowns. Nach der Beendigung der Umfrage, in arsnova.click nach Ablauf des Timers und in arsnova.voting zu einem beliebigen Zeitpunkt, blendet der Vortragende die letzte Folie der Quizfrage ein. Darauf werden die aktuell vorliegenden Ergebnisse grafisch dargestellt. Bei der Verwendung von arsnova.click wird der grafische Inhalt noch um eine Bestenliste mit den ersten 10 Plätzen ergänzt. Wenn der Vortragende die aktuelle Präsentation nach Ablauf des Vortrages speichert, werden die eingefügten Umfrageergebnisse mit gespeichert und stehen so auch später noch zur Verfügung.

Das Design der angelegten Folien kann nach der Erstellung von dem Anwender angepasst werden. Nur die Inhalte selbst, beispielsweise der Text einer Frage, darf lediglich über die Fragen-Verwaltung angepasst werden. Eine Manipulation kann zu Fehlern während der Vorstellung führen, wenn zwingend benötigte Elemente entfernt wurden. Auch sind inkonsistente Daten möglich, da angepasst Folientexte noch in ihrer alten Form auf dem Server hinterlegt sind.

### 3.2.5 Bearbeiten

Über den Button “Bearbeiten” gelangt der Anwender in das gleiche zweiteilige Installationsmenü, welches in dem vorherigen Kapitel zum Anlegen verwendet wurde. Die vorher eingestellten Werte sind hier bereits eingetragen und die schon existierenden Folien werden nach einem erfolgreichen Abschluss des Editierens inhaltlich angepasst. Durch das Verlassen der Ansicht oder das Betätigen des Abbrechen-Buttons, bleibt die Frage nach der Bestätigung einer entsprechenden Rückfrage so bestehen, wie sie vor dem Editieren war.

### 3.2.6 Ausblenden

Der Button “Ausblenden” hat eine ähnliche Auswirkung wie die bereits in PowerPoint bestehende, gleichnamige Funktion: Ausgeschlossene Folien werden in der echten Präsentation nicht angezeigt. Diese kann bei der Arbeit mit arsnova.voting auch verwendet werden, da keine Fragen zur Abstimmung freigegeben werden, wenn die entsprechenden Folien während der Präsentation nicht angezeigt werden. Allerdings müssen hier alle zwei beziehungsweise drei Folien einzeln ausgeblendet werden. Mit der Ausschließen-Funktion von der ARSnova-Integration werden direkt alle Folien, die der Quizfrage angehören, ausgeblendet.

arsnova.click hat im Gegensatz zu arsnova.voting eine feste Reihenfolge der Fragen und kann keine davon überspringen. Sollten die Folien trotzdem über die PowerPoint-Funktionalität ausgeschlossen werden, führt dies zu Fehlern auf dem arsnova.click-Server. Die Lösung ist, ausgeblendete Fragen nicht an den Server zu übermitteln. Damit die Integration über den ausgeblendeten Zustand informiert wird, ist an dieser Stelle das Verwenden der eigenen Ausschließen-Funktion zwingend nötig.

### 3.2.7 Löschen

Das Löschen, welches über den gleichnamigen Button gesteuert wird, fordert Nutzer nach der Betätigung auf, die Löschung erneut zu bestätigen. Gerade bei dem Entfernen von Daten ist es wichtig, dass dies nicht aus Versehen geschieht, daher die erneute Rückfrage. Die Quizfrage wird aus der Verwaltungsauflistung, den lokalen Sicherungen und im Falle von arsnova.voting auch vom Server entfernt. Ebenfalls werden die zugehörigen Folien automatisch aus der Präsentation gelöscht.



### 3.2.8 Informationen

Der Bereich “Info” im Menüband beinhaltet die beiden Schaltflächen “Hilfe” und “Über”. Der erste Button öffnet den ARSnova-Blog, auf dem Informationen über das gesamte ARSnova-System und die einzelnen Use Cases zu finden sind. Hier werden dem Anwender die Einsatzzwecke und Herangehensweise schrittweise dargelegt. Auch gibt es verschiedene Tutorials, um die Anwendungen besser kennen zu lernen.

Der “Über”-Button listet in einem gesonderten Fenster Informationen über die ARSnova-Integration für PowerPoint auf. Dazu gehören:

- Produktbeschreibung
- Version
- Lizenzierung

## 3.3 Anforderungen

Dieser Abschnitt befasst sich mit den Anforderungen an alle drei beteiligten Systeme. Hierbei wird zwischen den funktionalen und nicht-funktionalen Anforderungen unterschieden. Die funktionalen Anforderungen der Integrationslösung wurden größtenteils bereits im ersten Teil der Analyse in Form von Anwendungsfällen beschrieben und werden daher folgend nur noch ergänzt.

Wie aus dem Titel der Arbeit hervorgeht, wird Microsoft Office PowerPoint die erforderlichen Bedingungen erfüllen, trotzdem werden hier alle alternativen Lösungen beschrieben und die entsprechenden Vor- und Nachteile bewertet. Aus diesem Teil der Arbeit geht hervor, warum sich PowerPoint für eine volle Integration eignet.

Da die Integrationslösung die funktionalen Anforderungen der beiden ARS abbildet, diese allerdings nicht erweitert werden sollen, werden in dem Abschnitt ARSnova Systeme lediglich die funktionalen Anforderungen an die beiden ARS behandelt. Anschließend werden arsnova.voting sowie arsnova.click in dem Kapitel Anforderungsanalyse in Hinblick auf die vordefinierten Parameter analysiert. Beide Systeme müssen um die daraus hervorgehenden Anforderungen ergänzt werden.

### 3.3.1 Weitere funktionale Anforderungen

Die Integrationslösung soll einfach zu erlernen und intuitiv zu verwenden sein. Genauer gesagt: Die Software sollte sich in der Bedienung an Microsoft Office orientieren, da die Nutzer dieses Bedienungskonzept bereits kennen. Somit ist der Einarbeitungsaufwand für die ARS in PowerPoint möglichst gering. Sollte dies nicht der Fall sein, verliert die Integration seinen größten Mehrwert: ARSnova ohne zusätzliche Einarbeitungszeit zu verwenden. Diese Situation darf nicht eintreten.

Außerdem ist es wichtig, dass die jeweiligen Server-Sessions der Präsentationen an das PowerPoint-Dokument selbst geknüpft sind und nicht an einen Benutzer. Zwar ist es möglich, den Office-Account zur Authentifizierung zu verwenden, doch soll die Präsentation auch von anderen Endgeräten aus oder von Kollegen, denen der Vortrag zur Verfügung gestellt wurde, genutzt werden können. Dies trägt dazu bei, dass das Dokument in sich konsistent bleibt und nicht von einem Nutzer abhängig ist. Dies würde dem Konzept eines Dokumentes widersprechenden, welches beliebig an andere Personen weitergegeben werden kann. Möchte man nicht, dass diese die Datei bearbeiten,

kann die Sperren-Funktion von Office genutzt werden. Sollte gewünscht werden, dass die Präsentation nicht von einer anderen Person in PowerPoint gehalten wird und damit auch die Umfragen verwendet werden, werden die Folien lediglich ausgedruckt. Es besteht kein Grund, die Sessions an einen Nutzer zu knüpfen.

### 3.3.2 Nicht-funktionale Anforderungen

Im Folgenden werden alle nicht-funktionalen Parameter für die Integrationslösung beschrieben. Diese Anforderungen müssen zwingend erfüllt werden, um die Verwendbarkeit von arsnova.voting und arsnova.click in dem Host-System zu ermöglichen.

Um als Softwarelösung wirklich nützlich zu sein, muss die Anwendung, in die ARSnova integriert wird, mindestens von 50 % der Personen, die regelmäßig Vorträge halten, verwendet werden. Wenn das Programm, in welches die beiden ARS integriert beziehungsweise angebunden werden, von zu wenigen, vortragenden Personen verwendet wird, ist das Integrationsprojekt nicht als erfolgreich anzusehen.

Das Hostsystem muss über eine hohe Usability verfügen. Neben einer intuitiven Bedienung wird auch erwartet, dass eine Hilfsfunktion zur Verfügung steht. Das ARSnova-Hostsystem muss mindestens eine so hohe Oberflächenqualität haben wie ARSnova selbst, um ein einfaches Arbeiten mit den zusammenhängenden Funktionen zu ermöglichen.

Die Präsentationssoftware muss eine technische Möglichkeit zur Verfügung stellen, Anwendungen anzubinden beziehungsweise zu integrieren. Dabei müssen folgende Aspekte unterstützt werden:

- Erweiterung der bestehenden Oberflächen, inklusive Kontextmenüs
- Implementierung neuer Funktionalitäten
- Erstellen, versenden und empfangen von Web-Requests
- Sichern von zusätzlichen Daten

Eine einfache “Overlay”-Lösung, bei der bestimmte Darstellungen einfach über Elemente einer anderen Software gelegt werden, wirkt inkonsistent und läuft Gefahr, bei der kleinsten grafischen Anpassung nicht mehr den Designansprüchen zu genügen. Der Wartungsaufwand könnte hier immens sein. Daher ist eine ordentliche und vollständige Integration nur durch die Bereitstellung der gelisteten Anforderungen möglich.

### 3.3.3 ARSnova Systeme

Wie bereits in der Kapitelbeschreibung erwähnt, werden an dieser Stelle lediglich die nicht-funktionalen Anforderungen an arsnova.voting und arsnova.click aufgeführt. Die funktionalen Aspekte der beiden ARS sollen in der Integrationslösung abgebildet und nicht erweitert werden.

Die Verfügbarkeit der beiden ARSnova-Systeme muss mindestens 99,9 % oder höher betragen. Bei einer geringeren Zuverlässigkeit kann es passieren, dass dem Benutzer auffällt, dass die angebundenen Anwendungen nicht zur Verfügung stehen, obwohl die Integrationslösung selbst funktioniert. Eventuell geht er sogar davon aus, dass es ein

Problem mit dem Hostsystem oder der integrierten Lösung selbst gibt, obwohl lediglich die Endpunkte zur Zeit nicht erreichbar sind. Das ist nicht vertretbar.

Unabhängig von den Anfragen oder den gesendeten Daten müssen die Systeme immer schnell antworten. Je nach Anfrage und Menge der Daten gibt es dabei eine gewisse Toleranz, gerade weil viele Anfragen asynchron ausgeführt werden können und dem Anwender eine kleine Verzögerung dadurch nicht direkt auffällt. Trotzdem muss die Performance immer flüssig erscheinen, regelmäßige Abfragen dürfen nie länger als 500 Millisekunden betragen, seltenere Abfragen mit einem größerem Datenvolumen nicht länger als eine Sekunde. Diese Anforderung bezieht sich rein auf die technischen Möglichkeiten des Servers und seiner Software, für eine längere Antwortzeit aufgrund einer schlechten Internetanbindung des Endnutzers muss nicht Sorge getragen werden.

Sicherheitsanforderungen sind keinesfalls zu vernachlässigen. Die Daten müssen sicher verwaltet werden, sodass sie nicht verloren gehen. Darüber hinaus müssen sie vertraulich behandelt und bereitgestellt werden – eine Secure Sockets Layer (SSL) Verschlüsselung und zertifizierte Rechenzentren sind in Europa Pflicht.

Für eine einwandfreie Benutzbarkeit müssen die bestehenden Webanwendungen über eine Hilfe verfügen. Hierdurch können sich die Anwender selbst erklären, welche Logik hinter bestimmten Anwendungsfällen steht oder wie die Zuhörerschaft mit den Systemen zu interagieren hat. Dies geht über eine Online-Hilfe bedeutend schneller, als einen Support zu kontaktieren oder erst im Internet recherchieren zu müssen.

Um die Anbindung zu ermöglichen, müssen die Systeme ein Backend zur Verfügung stellen, über das alle in den Anwendungsfällen erwähnten Funktionalitäten angeboten werden. Aus Gründen der Einfachheit soll dies eine REST-API sein, Alternativen wie Simple Object Access Protocol (SOAP) sind für diese Anwendungsfälle zu komplex und schwerfällig. Ein Beispiel für die erhöhte Komplexität ist das Umwandeln von .NET-Objekten in JavaScript Object Notation (JSON) und umgekehrt. Dies ist mit bestehenden .NET-Bibliotheken einfach zu bewerkstelligen, während die Arbeit mit dem XML-Format einen erhöhten Implementierungsaufwand darstellen würde.

## **3.4 Anforderungsanalyse**

Zu Beginn dieses Abschnittes wird analysiert, ob und welche Hostsysteme die vorher definierten Anforderungen erfüllen und somit zur Anbindung an die ARS in Frage kommen. Anschließend wird überprüft, inwiefern die bestehenden beiden ARSnova-Systeme den nicht-funktionalen Anforderungen genügen beziehungsweise um welche Aspekte die Systeme erweitert oder angepasst werden müssen.

### **3.4.1 Analyse von PowerPoint Add-ins**

Schon vor dem Beginn der Analyse ist Microsoft Office PowerPoint der Favorit als Hostsystem für die Integrationslösung. Dies liegt unter anderem an zwei einfachen Gründen:

1. Eine Studie hat gezeigt, dass PowerPoint die mit Abstand am weitesten verbreitetste Software zur Präsentationsunterstützung ist (Thielsch & Förster, 2007, S. 4). Es ist ein starkes Argument, 82,8 % der Vortragenden eine ARSnova-Integration in Ihrer favorisierten und alltäglich angewendeten Präsentationssoftware zu bieten.
2. An fast allen Hochschulen wird Microsoft Office kostenfrei für alle Studenten und Dozenten angeboten. Damit müssen zumindest die Personen in der Lehre keine Software erwerben. In der Wirtschaft ist das Office-Paket in der Regel ebenfalls Standard (Miedl, 2013).

Gerade der zweite Punkt ist für den Hochschuleinsatz zu beachten, da zusätzliche Kosten für den Vorlesungsbetrieb mit einem organisatorischen Aufwand verbunden sind: Die Gelder müssen erst beantragt und gerechtfertigt werden. Dieser Aufwand oder das alternative private Bezahlen der Anwendung würde abschreckend wirken. Gut, dass fast alle Hochschulen das Office-Paket bereits zur Verfügung stellen.

Im Folgenden gilt es zu analysieren, ob PowerPoint auch die restlichen, nicht-funktionalen Anforderungen erfüllen kann und damit als Hostsystem in Frage kommt. Die funktionalen Anforderungen sind an dieser Stelle nicht weiter interessant, da diese durch die Implementierung selbst realisiert werden und das Hostsystem darauf keine Einflüsse hat. Anschließend werden in den folgenden Kapiteln die möglichen Alternativen erläutert und die beiden anzubindenden ARS analysiert.

Im ersten Punkt der nicht-funktionalen Anforderungen für das Hostsystem wurde gefordert, dass mindestens 50 % der Personen, die regelmäßig Vorträge halten, die entsprechende Präsentationssoftware verwenden sollen. Wie bereits angedeutet, liegt PowerPoint an dieser Stelle laut einer Studie aus dem Jahr 2007 mit 82,8 % eindeutig vorne (Thielsch & Förster, 2007, S. 4). Zwar ist diese Studie inzwischen 10 Jahre alt und zum Zeitpunkt der Evaluation war keine neuere Studie bekannt, doch gibt es keinen Grund zur Annahme, dass sich die Verteilung stark verschoben hätte. Die aktuellen Präsentationssysteme haben sich in den letzten Jahren nur wenig entwickelt, da es nicht viele weitere Methoden gibt, einen Vortrag weiter zu unterstützen: Die zentrale Komponente bleibt nach wie vor die vortragende Person selbst.

Weiterhin wurde eine gute Usability inklusive moderner Hilfsfunktionen gefordert. An dieser Stelle glänzt PowerPoint mit einer jahrelangen Entwicklung in der Oberflächensteuerung. Durch die allgemeine Steuerung der gesamten Office-Suite und der weiten Verbreitung sind die Oberflächen bei den meisten Windows-Nutzern so weit verbreitet, dass trotz dem großen Funktionsumfang von einer intuitiven Steuerung gesprochen werden kann. Neben der Hilfsfunktion, welche alle Interaktionen der Office-Suite abdeckt, werden auch Elemente der Barrierefreiheit wie die Screenreader-Tauglichkeit voll unterstützt. An dieser Stelle gilt das gesamte Office-Paket als Vorreiter und ist mehr als geeignet, ARSnova zu integrieren.

Die letzte und wichtigste nicht-funktionale Anforderung ist die technische Möglichkeit einer Integration. An dieser Stelle bietet Microsoft mit den VSTO eine Möglichkeit. Diese sind zuerst in der Version VSTO 2003 erschienen und unterstützen seit der Version VSTO 2005 SE ab Office 2007 auch PowerPoint. Vorher waren nur Anpassungen an Word und Excel möglich. Im Folgenden ist die Überprüfung der vier spezifischen, technischen Anforderungen aufgelistet (Microsoft, 2017d):

- Mit den VSTO können alle bestehenden Oberflächen erweitert werden, nur können bestehende Einträge nicht angepasst werden. Des Weiteren unterstützen die VSTO in der aktuellen Version 4.0 die Verwendung der WPF. Damit eröffnet sich die Möglichkeit, komplett neue Fenster mit eigenen Funktionalitäten anzubieten, die in bestehenden PowerPoint-Oberflächen keinen Platz finden würden.
- Durch die CLR-Unterstützung seit der ersten Version der VSTO stellt Office als Hostsystem die gesamte Bandbreite des .NET-Frameworks zur Verfügung. Daher ist es möglich, alle Funktionen, inklusive der Web-Requests, zu implementieren. Auch hier entspricht Office den nicht-funktionalen Anforderungen vollkommen.
- Ebenfalls soll es eine Möglichkeit geben, weitere Inhalte neben der Präsentation zu speichern - allerdings nicht in weiteren Dateien. Die Session-Informationen gehören zu einer Präsentation und sollen mit dieser gesichert werden. Wenn das Präsentations-Dokument auf einen anderen Rechner kopiert wird, sollen die Dokumente hier ebenfalls vorliegen. Hierfür gibt es bei Office-Dokumenten zwei Möglichkeiten. Zum einen verfügt jedes Dokument über sogenannte "CustomDocumentProperties", welche allerdings auf 255 Zeichen pro Property begrenzt sind. Natürlich könnte man die zu speichernden Elemente in 255-Zeichen-Elemente splitten und beim Auslesen wieder zusammenfügen, doch das ist aufwendig, kostet Rechenleistung und ist nicht Sinn der CustomDocumentProperties. Die zweite Möglichkeit bildet die Liste "CustomXMLParts", welche ebenfalls zu jedem Office-Dokument gehört. Ein XML-String kann beliebig lang sein, lediglich die Serialisierung der Information in das XML-Format muss erfolgen. Dann stellt VSTO mit den CustomXMLParts eine gute Lösung zum Sichern der Sessiondaten einer Präsentation dar.

Als Alternative zu dem Office Add-in mit den VSTO gibt es das Web-Add-in für die Office-Plattform. Der große Vorteil ist hier, dass die Web-Add-ins in allen Office Versionen funktionsfähig sind, also auch für die Mac-Versionen, ebenso für das in der Cloud und im Browser betriebene Office 365. Der Nachteil besteht in den mit Abstand geringeren Möglichkeiten. Im Web müssen zusätzliche Oberflächen mit HTML und CSS dargestellt werden, unter Windows mit WPF und auf einem Mac mit dem dortigen Darstellungs-Framework. Der funktionale Umfang der Integrationslösung ist so groß, dass dieser nicht ohne zusätzliche Oberflächen abgebildet werden kann. Allerdings gibt es kein universelles Darstellungs-Framework für alle Oberflächen - daher muss eine Entscheidung für eines der Systeme fallen. Office 365 wäre eine gute Wahl, da die Web-Version von jedem Betriebssystem aus zu erreichen ist, allerdings bietet dies auch nur einen kleinen Funktionsumfang der eigentlichen Office-Suite und ist unter den Anwendern noch nicht sehr weit verbreitet. Die Nutzer an die Web-Version zu gewöhnen würde für diese die Verwendung einer anderen Office Version als der bekannten, lokalen bedeuten und damit eine Umstellung. Das ist genau das, was durch die Integration verhindert werden soll und widerspräche somit dem gesamten Konzept. Daher fällt die Wahl hier auf die VSTO und einem Add-in für die lokale Windows-Version von Microsoft Office PowerPoint.

### 3.4.2 Alternativen

Die zwei bekanntesten Alternativen zu Microsoft Office sind wohl Apache OpenOffice und LibreOffice. Die Open-Source Varianten kommen jedoch nicht als Hostsystem für diese Problemstellung in Frage, da beide die nicht-funktionale Anforderung der minimalen Anwenderschaft im Verhältnis zu den Gesamtanwendern von Präsentationssoftware nicht erfüllen. Ebenfalls sind die zwingend benötigten technischen Voraussetzungen für die Entwicklung einer vollständigen Integration kritisch zu betrachten. Zwar gibt es die sogenannten “Extensions” für Apache OpenOffice, doch sind diese lange nicht so mächtig wie die VSTO. Für das Sichern von Daten sowie auch für das Hinzufügen neuer Fenster und Inhalte findet man nur wenige bis keine Informationen. Dies könnte aber auch an der Dokumentation liegen, die im Gegensatz zu den auf Microsoft Developer Network (MSDN) ausführlich dokumentierten und mit Beispielen versehenen VSTO-Informationen mangelhaft erscheint. Ebenso werden die Extensions mehr als OpenOffice-API dargestellt als wirklich mächtige Entwicklertools für die Microsoft Office-Alternative. Auch eine schlechte Dokumentation ist aufgrund des entstehenden Mehraufwandes in der Entwicklung schon ein valides Argument, ein Framework niedriger zu priorisieren (Apache, 2017).

Aus dem ersten Abschnitt hat sich deutlich herauskristallisiert, dass Microsoft Office PowerPoint durch seine Entwicklertools und die weite Verbreitung besticht. Eine letzte Alternative wären die Google Apps, doch diese sind noch weniger verbreitet als die bereits genannten Alternativen und kommen somit ebenfalls nicht für eine Integration in Frage.

### 3.4.3 Analyse von ARSnova

Die nicht-funktionalen Anforderungen an die beiden zu integrierenden ARS sind deckungsgleich. Zu Beginn werden die Verfügbarkeit, die Sicherheit sowie die Hilfsfunktion analysiert, da diese für beide Systeme im gleichen Maße gelten. Die restlichen beiden Aspekte unterscheiden sich je nach System, daher werden diese Fälle differenziert betrachtet.

Zertifizierte Rechenzentren geben viele Sicherheiten. Über die Datensicherheit in Form von dezentraler Spiegelung, skalierbarer Hardware, wenn eine größere Anzahl an Ressourcen benötigt wird oder eben der Sicherheits- und Verfügbarkeits-Zertifizierung. ARSnova ist ein Open Source Hochschul-Projekt, welches auch an diesen zur Verfügung gestellt wird. Aus Kostengründen und der Einfachheit, den eigenen Server im eigenen Haus leichter verwalten zu können, ist dieses für ein solches Open Source-Projekt natürlich sinnvoll. Für Sicherheit in Form von Verschlüsselung und Servern, die in Deutschland stehen und dementsprechend den deutschen Gesetzen unterliegen, wird hier also gesorgt. Lediglich die dezentralen Backups im Falle eines Brandes oder eines Gebäude-Einsturzes werden hier nicht durchgeführt. Diese Situation ist zwar sehr abwegig, da öffentliche Gebäude strengen Sicherheits- und Kontrollauflagen unterliegen, doch es kann passieren, dass Daten eventuell nicht gesichert sind. Im Falle von arsnova.click ist dies nicht so wichtig, da dort aus rechtlichen Gründen keine Daten auf dem Server, sondern lediglich im lokalen Browser-Speicher des Anwenders gespeichert werden, doch im Falle von arsnova.voting könnte ein solches Szenario verheerend sein. Trotz allem genügen die Ansprüche einem Hochschulprojekt und aufgrund der Unwahrscheinlichkeit dieser Fälle kann von einer Verfügbarkeit von über 99,9 % sowie ausreichender Sicherheit für beide Anwendungen gesprochen werden, auch ohne eine

Zertifizierung.

Der ARSnova-Blog (<https://arsnova.io>) verfügt über viele Informationen und Hilfestellungen zu der Verwendung der ARSnova-Systeme. Neben der Erläuterung von Anwendungsfällen und deren Hintergründen gibt es auch Tutorials und einen Helpdesk, der sich um größere technische, bedienungsorientierte und didaktische Probleme kümmert. Für einen ausreichenden Support ist somit ebenfalls gesorgt.

#### **3.4.3.1 Performance**

Um dem Anwender ein flüssiges Anwendungserlebnis bieten zu können, ist eine gute Performance unabdingbar. Meteor hat, wie in dem Kapitel Grundlagen bereits erläutert, durch seine Reaktivität und die stetige Synchronisation vieler Mongo-Datenbanken nicht die beste Performance. In dieser Betrachtung sind die genannten Geschwindigkeits-Einbußen allerdings nicht von Bedeutung, da durch die REST-API lediglich mit der Serverinstanz kommuniziert wird. Ob Meteor bestimmte Informationen weiter verteilt und ob daraus leichte Performance-Differenzen entstehen, ist für den Akteur der Integrationslösung nicht von Belang. Da dem Server viel Hardware zur Verfügung steht, gibt es im Falle von arsnova.click keine Bedenken bezüglich der Performance.

Bei arsnova.voting hingegen sieht der Fall anders aus. Auch dieser Serverinstanz sind großzügige Hardwareressourcen zur Verfügung gestellt, doch ist die Anwendung an vielen Stellen sehr langsam. So ist es nicht unüblich, dass der Client bei dem Erstellen einer neuen Session bis zu 10 Sekunden oder mehr auf eine Antwort vonseiten des Servers liegt. Dies hat mehrere Ursachen, nur ein Beispiel ist das Session-Model mit über 70 primitiven Attributen, von denen in der Regel nicht einmal 10 verwendet werden. Hier wären wenige Objekte, die lediglich als Referenzen übergeben werden und in den wenigen Fällen, in denen sie benötigt werden, nachgeladen werden, sinnvoller. Aufgrund der vielen Ursachen, die mit dem langsamen Backend zusammenhängen, soll anstelle eines kleinen Hotfixes eine komplette Neuentwicklung der Hintergrundlogik erfolgen. Zum Zeitpunkt der Verfassung dieser Thesis ist diese bereits in Arbeit und verspricht, Abhilfe zu schaffen. Daraus resultierend entspricht arsnova.voting in seinem jetzigen Zustand nicht den Performance-Anforderungen, sollte allerdings schon in naher Zukunft für Besserung sorgen. Mit diesen Aussichten wird der Prototyp der Integrationslösung an arsnova.voting angebunden. Später müssen dann lediglich die Endpunkt-Zugriffe und die entsprechenden Parameter angepasst werden, um eine performante Kommunikation zu gewährleisten.

#### **3.4.3.2 REST-Schnittstelle**

Die Anforderungsdefinition der Schnittstellen ist ein Ausschlusskriterium, da sie die Verbindung zwischen den ARSnova-Systemen und der Integrationslösung herstellt. Sollte diese noch nicht implementiert sein, muss dies zwingend nachgeholt werden. arsnova.voting ist explizit in ein Front- und Backend aufgeteilt. Das Backend besitzt, wie gefordert, eine REST-API, über die alle Funktionalitäten angeboten werden. Zusätzlich arbeitet die Anwendung teilweise auf WebSocket-Basis, um einige Funktionalitäten wie das Live-Feedback in Echtzeit darzustellen. Damit ist der Anforderung in diesem Fall Genüge getan.

Die gamifizierte Version arsnova.click wurde als App konzipiert und entwickelt. Auch das Framework Meteor sieht kein eigenständiges Backend vor. An dieser Stelle muss, um eine Integration zu ermöglichen, eine REST-API definiert und implementiert wer-

den. In Meteor gibt es ein serverseitiges Skript, welches bereits verwendet wird, um Abfragen außerhalb des Meteor-Frameworks mit WebSocket-Verbindung auszuführen. Dies ist zur Zeit lediglich der Verbindungstest, bei dem der Client die Latenz zum Server über wiederholte REST-Aufrufe testet, die Ergebnisse mitteilt und den Nutzer warnt, wenn die Verbindung zu schwach ist, um eine performante Benutzererfahrung zu gewährleisten. Gerade bei der Nutzung des mobilen Netzes mit Smartphones kann dies sinnvoll sein, damit der Anwender weiß, dass die schlechte Performance nicht von der App selbst herrührt. Hier können beliebige weitere Requests eingefügt werden, um eine Schnittstelle aufzubauen. Des Weiteren gibt es zusätzliche Meteor-Pakete, welche automatisch REST-Methoden aus bestehenden Datenbank-Zugriffs-Definitionen generiert. Aus den vielen Angeboten sticht ein aktuelles und weit verbreitetes Paket hervor: `simple:rest` (<https://atmospherejs.com/simple/rest>). Mit regelmäßigen Updates in einem Zeitraum von über einem Jahr sowie fast 3.800 Installationen in bestehenden Projekten ist dieses Paket eindeutig empfehlenswert. Lediglich mit einem anderen Paket, welches bei schreibenden Zugriffen auf die Mongo-Datenbank für eine automatische Authentifizierung sorgt, hat `simple:rest` Probleme: Die nicht offizielle und standardmäßig in Meteor inkludierte Authentifizierung wird nicht unterstützt. Da dies nicht für die lesenden Zugriffe gilt, ergibt sich folgende Implementierungsempfehlung für die REST-Schnittstelle in `arsnova.click`:

- Aus Gründen eines geringen Implementierungsaufwandes, der Vermeidung von Fehlern und Erhöhung der Wartbarkeit wird das Meteor Paket `simple:rest` für die automatische Generierung aller benötigten GET-Anfragen empfohlen.
- Alle anderen Create, Read, Update and Delete (CRUD)-Befehle, die Schreibrechte benötigen, müssen händisch implementiert werden. Dabei müssen Sicherheitsabfragen, Validierungen und Authentifizierungen bei der Entwicklung berücksichtigt werden.

Die Definition und Entwicklung der REST-API für `arsnova.click` ist in den folgenden Kapiteln Konzeption und Implementierung beschrieben. Nach einer erfolgreichen Implementierung von dieser entsprechen die beiden ARS den definierten, nicht-funktionalen Anforderungen und können somit in PowerPoint integriert werden.



## 4 Konzeption

Aufgrund der bisher definierten Anforderungen kann nun eine Softwarelösung konzipiert werden. Der Fokus von diesem Teil wird zu Beginn auf der Integrationslösung selbst liegen. Neben dem Grenzverhalten der VSTO wird die Architektur der Anwendung sowie die verwendeten Entwurfsmuster erläutert. Danach werden alle technischen Abläufe im Zusammenhang mit der Dokumentenverwaltung beschrieben.

Auf der Seite der ARS wird ein genauer Blick auf arsnova.click geworfen, da dieses System noch keine REST-API besitzt. Die entsprechenden Schnittstellendefinition erfolgen am Ende dieses Kapitels.

### 4.1 Technologien und Bibliotheken

Für die PowerPoint-Integration werden, wie in der Analyse beschrieben, die VSTO verwendet. Da dieses Tool-Paket für die Verwendung mit Visual Studio und C# .NET entwickelt wurde, ist die Wahl der Programmiersprache und der entsprechenden Entwicklungsumgebung obligatorisch. .NET wird in der neusten Version 4.5.2 verwendet, bei Visual Studio die Version Enterprise 2015. Ausgeführt wird die Software auf dem Betriebssystem Windows 10. Zwar hätte die Entwicklung auch mit VB erfolgen können, doch ist C# die mit Abstand modernere Sprache und erleichtert die Entwicklung durch seine angepassten Syntax-Konzepte erheblich. So werden Schleifen- und Klassen-Blöcke nicht mehr durch Bezeichner wie “Class ... End Class” oder “For ... Next” abgegrenzt, sondern durch einfache Klammern. Hier machen sich die Programmiersprachen bemerkbar, die C# geprägt haben: C++ und Java. Auch wurden die vielen sprechenden, dafür aber umständlichen und nicht etablierten Bezeichner angepasst. Aus “MustInherit” wurde “abstract”, aus “NotInheritable” wurde “sealed” und aus “Overrideable” wurde “virtual”.

Zur Verbesserung der Code-Qualität und Bereitstellung von Entwicklungshilfe-Tools wurde ReSharper, eine Erweiterung der Visual Studio Integrated development environment (IDE), von dem Unternehmen JetBrains verwendet. Folgender Funktionsumfang wurde eingesetzt:

- Just-in-time Code-Analyse, Formatierung und Bereinigung
- Automatisches Refactoring. Hierbei werden unter anderem die Implementierungs- und Verwendungsbezeichnungen mit angepasst, wenn ein Interface umbenannt wird.
- Verbesserte Navigations- und Suchfunktionen
- Code-Generierung, beispielsweise für Vertrags-Implementierungen

Für die Versionsverwaltung des Quellcodes von diesem Open-Source-Projekt wird die freie und verteilte Software Git, gehostet auf dem hauseigenen Server der Technischen Hochschule Mittelhessen, verwendet:

<https://git.thm.de/arsnova/powerpoint-integration>.

Folgende Bibliotheken kommen neben dem Funktionsumfang des .NET-Frameworks zum Einsatz:

- Microsoft.VSTO v4.0: Alle Informationen finden sich im gleichnamigen Kapitel auf Seite 18.
- Newtonsoft.Json: Stellt Methoden zur Arbeit mit Daten im JSON-Format zur Verfügung.
- Microsoft.Unity: Ein leichtgewichtiger, erweiterbarer Container für Dependency Injection mit der Unterstützung von Konstruktor-, Property- und Methoden-Injection. Mehr zu dem Thema im gleichnamigen Implementierungs-Kapitel auf Seite 76.

## 4.2 Architektur

Die Integrationssoftware unterteilt sich in drei Schichten, die Benutzerschnittstelle, die Geschäftslogik und die Kommunikationsschicht. Während in der obersten Ebene, der Benutzerschnittstelle, die Anbindung an PowerPoint mit Hilfe der VSTO erfolgt und die Benutzeroberflächen und Interaktionen mit der WPF abgedeckt werden, befindet sich in der Geschäftslogik die Manipulation und Vorbereitung der Folien sowie allen weiteren Berechnungen wie beispielsweise die Auswertung der Abstimmungsergebnisse. In der Kommunikationsschnittstelle befinden sich alle nötigen Funktionalitäten um mit den ARSnova-Systemen zu interagieren, also die REST-APIs anzusprechen. Die Funktionen einer jeden Ebene, die für andere Schichten zur Verfügung gestellt werden sollen, sind über Interfaces definiert. Die restlichen Ebenen kennen nur diese Verträge, die Objektverwaltung übernimmt der Service Locator. Dadurch wird verhindert, dass die Anwenderschicht und damit auch PowerPoint die Kommunikationschnittstelle und die gespeicherten Sessiondaten verwenden kann. Dies soll aus Gründen der Datensicherheit allein der Geschäftslogik vorbehalten sein. Weitere Details zu den einzelnen Ebenen sowie den verwendeten Technologien finden sich in den nächsten Abschnitten Projektstruktur und Entwurfsmuster.

## 4.3 Projektstruktur

Um die einzelnen architektonischen Ebenen und die Kapselung der jeweiligen Elemente besser einordnen zu können, werden alle Projekte folgend in einen bildlichen Kontext gebracht. Anschließend wird auf dieser Basis jedes Projekt für sich untersucht. Aus dem unten gelisteten Modell wurden der Bootstrapper mit der Anwendungskonfiguration, wie beispielsweise der Initialisierung des Service Locators für die Dependency Injection, sowie das Test-Projekt für eine bessere Übersicht entfernt.

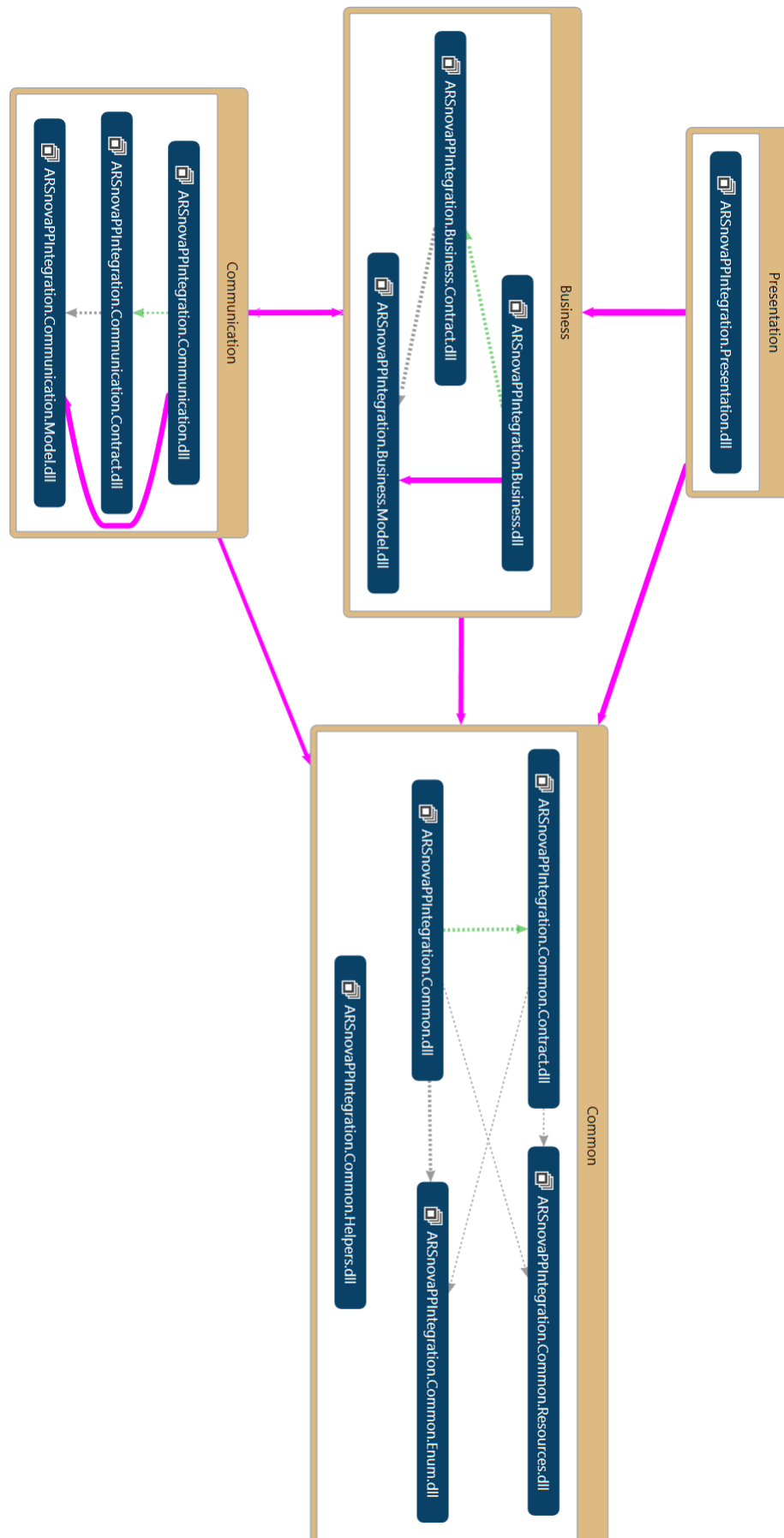


Abbildung 12: Schichtabhängigkeiten

Die orangefarbenen Umrandungen geben die jeweiligen Schichten an, die blauen und hellgrünen Kästchen innerhalb dieser die einzelnen Projekte, welche Klassen oder Interfaces enthalten. Die pinken Linien zeigen an, wenn ein Projekt Elemente aus einem anderen verwendet. Die kleinen, grünen Linien geben an, dass in den jeweiligen Projekten Verträge aus dem referenzierten implementiert werden.

Die Darstellungsschicht, im Projekt selbst “Presentation” genannt, greift auf das Business-Projekt, also die Geschäftslogik, zu, nicht aber auf das Communication-Projekt, in dem die Zugriffe auf die ARSnova-Systeme gekapselt sind. Auch kann in jeder Schicht das Contract-Projekt ausgemacht werden, in dem alle Interfaces der jeweiligen Schicht definiert sind. Im Anschluss eine Auflistung aller vierzehn Projekte:

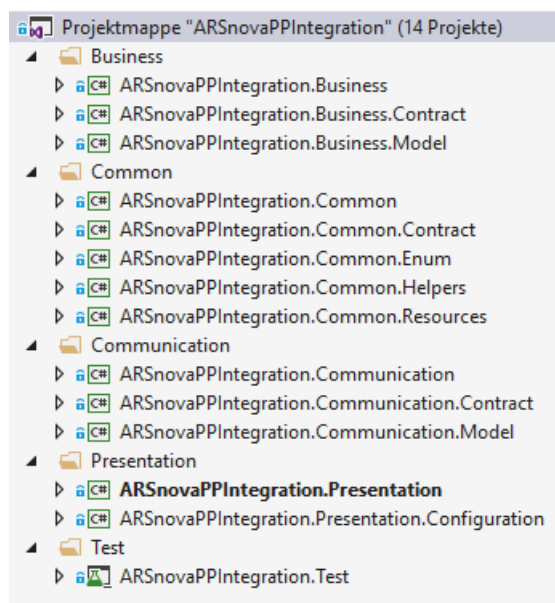


Abbildung 13: Die Projektliste

Die nachfolgende Auflistung enthält eine Kurzbeschreibung der jeweiligen Projekte, der entsprechenden Aufgaben und Inhalten. Gerade für die Entscheidung, welche Funktionen und Aufgaben von welchem Projekt übernommen werden, ist eine klare Abgrenzung unabdingbar. Eine Vermischung der Aufgabengebiete zersetzt die Architektur der Anwendung und damit auch die Wartbarkeit. Unerwünschte Seiteneffekte wie Performance-Einbußen können durchaus die Folge von solch unsauberen Implementierungsansätzen sein. Gerade zu Beginn eines neuen Projektes ist es daher sehr wichtig, sich die Struktur sehr genau zu überlegen. Die Entscheidungen sind grundlegend für den gesamten weiteren Entwicklungsverlauf der Anwendung.

- Business

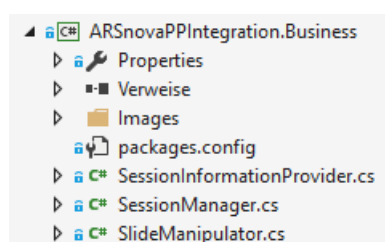


Abbildung 14: Auflistung der Dokumente im Projekt “Business”

Das Business-Projekt enthält die eigentliche Geschäftslogik der Anwendung. Diese ist in drei Klassen unterteilt. Eine Klasse ist der `SessionInformationProvider`, der lediglich Informationen zur Verfügung stellt. Beispielsweise gibt es eine Methode, die auf Basis des Fragentyps mitteilt, ob es sich um eine `arsnova.voting`- oder `arsnova.click`-Session handelt.

Der `SessionManager` beinhaltet alle Methoden, die zur Steuerung einer Session benötigt werden. Hier werden Fragen übergeben, die angelegt werden sollen, zu Beginn des Vortrages wird hierüber die Session freigegeben und die einzelnen Fragen selbst werden ebenfalls über den `SessionManager` zur Abstimmung zur Verfügung gestellt. Die Aufrufe erfolgen dabei unabhängig von der gewählten Session. Erst der `SessionManager` differenziert zwischen `arsnova.voting` und `arsnova.click` und entscheidet, welche Server-Kommunikationen durchgeführt werden müssen. So wird eine neue Frage einer `arsnova.voting`-Session direkt auf den Server übertragen, bei `arsnova.click` allerdings nur intern gespeichert, da auf dem Server langfristig keine Session-Daten gespeichert werden. Erst zum Beginn eines Vortrages werden die Fragen zur Verfügung gestellt.

Die dritte Klasse, der `SlideManipulator`, ist für das Erzeugen und Bearbeiten der Inhalte auf den Folien selbst verantwortlich. Nur in dieser Klasse wird innerhalb der Business-Logik an bestehenden Folien gearbeitet. Im `Images`-Ordner liegen einige Bilder wie beispielsweise das `arsnova.click`-Logo, die der `SlideManipulator` als Inhalt einer Folie einfügt.

- Business.Contract

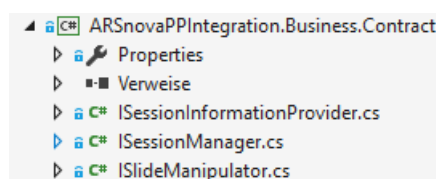


Abbildung 15: Auflistung der Dokumente im Projekt “Business.Contract”

Hier sind alle Interfaces der vorher beschriebenen drei Komponenten des Business-Projektes enthalten. Andere Schichten der Applikation verweisen lediglich auf dieses Vertrags-Projekt, niemals erfolgt ein direkter Verweis auf das Business-Projekt. Dies stellt die Kapselung der zwischen den einzelnen Schichten dar.

- Business.Model

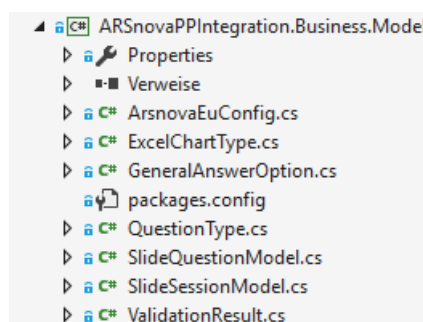


Abbildung 16: Auflistung der Dokumente im Projekt “Business.Model”

In dem Model-Projekt werden alle Datenstrukturen in Form von Klassen, die zur Darstellung von Daten dienen, gesichert. Das Projekt kann ebenfalls von den äußeren Schichten referenziert werden, da diese die Formate, in denen Daten übergeben und zurückgesendet werden, kennen müssen. Es ist auch möglich, die Model-Klassen in einem eigenen Ordner im Business.Contracts-Projekt unterzubringen, da sie zu den vertraglichen Vereinbarungen der Schicht dazugehören. Sie definieren das Format, in dem die Daten übertragen werden. Aus Gründen der Übersichtlichkeit und der damit verbundenen Reduktion von tiefen Verzweigungen, liegen die Models in der Integrationslösung in einem eigenen Projekt mit dem Namespace “ARSnovaPPIntegration.Business.Models” vor. Wären diese im Contracts-Projekt gesichert, würde der Namespace “ARSnovaPPIntegration.Business.Contract.Models” lauten, da Ordner wie auch Projekte eine eigene Ebene im Verweispfad einnehmen. So wurde eine Ebene gespart und damit die Komplexität verringert.

- Common

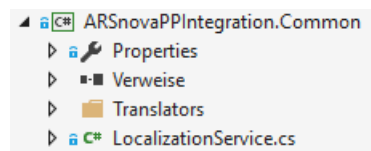


Abbildung 17: Auflistung der Dokumente im Projekt “Common”

Das Projekt Common beinhaltet alle programmatischen Elemente, die nicht auf eine Ebene gekapselt werden können, da sie in mehreren Schichten benötigt werden. Wichtig ist dabei, dass hier keine Elemente enthalten sind, die sicherheitskritisch sind oder von ihrer Definition her in eine andere Schicht gehören müssen. Ein Beispiel dafür wäre ein Zugriff auf die ARSnova-Systeme. Diese sollen nur in der Kommunikations-Ebene erfolgen, einen solchen hier unterzubringen, würde die Architektur zerstören.

Um der obigen Struktur-Regelung gerecht zu werden, beinhaltet das allgemeine Projekt Common hauptsächlich Übersetzungsangelegenheiten. Diese werden überall benötigt, wo Oberflächeninhalte erzeugt werden. Das trifft auch auf die SlideManipulator-Klasse in der Business-Schicht zu, demnach kann keine Kapselung in die Präsentations-Ebene erfolgen. Es wäre eine Option, die sprachlichen Elemente in der Geschäftslogik immer in einer Sprache zu erzeugen und vor der Darstellung in der Präsentations-Ebene zu übersetzen, doch dies wäre umständlich und verwirrend. Für eine bessere Übersicht soll immer bei der Zuweisung von dargestelltem Text zu einem Property die direkte Übersetzung erfolgen. Dadurch wird verhindert, dass im Nachhinein auffällt, dass eine Übersetzung vergessen wurde.

Neben dem Lokalisierungs-Service beinhaltet das Common-Projekt noch den Ordner “Translators”. Dieser enthält die Übersetzungen für die Enums. Beispielsweise wird der Fragentyp im Code als Enum gespeichert, da ein solcher durch seinen Bezeichner aussagekräftiger ist als eine Zahl. Soll der Anwender nun diesen Typ auswählen, muss eine Lokalisierung durchgeführt werden. Da allerdings keine Enums, sondern lediglich Texte übersetzt werden können, wird hier eine spezielle Zuordnung benötigt. Die Realisierung erfolgt durch einen Übersetzer für jeden

Enum-Typ. Dieser arbeitet ähnlich dem Lokalisierungs-Service, erwartet anstelle des Textes, der übersetzt werden soll, allerdings einen Enum-Typ.

- Common.Contract

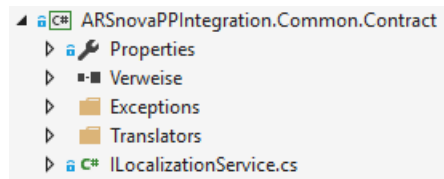


Abbildung 18: Auflistung der Dokumente im Projekt “Common.Contract”

Ähnlich der Geschäftslogik befinden sich auch hier die Vertrags-Definitionen für das Common-Projekt, auf welches die anderen Ebenen zugreifen können. An dieser Stelle gibt es allerdings noch einen Exceptions-Ordner, in dem alle Fehler-Definitionen vorliegen, welche die Integrationslösung verwendet. Da alle Fehler, die geworfen werden können, in der Präsentationsschicht für eine Nutzerausgabe behandelt werden müssen, ist auch hier ein globaler Zugriff von allen Ebenen nötig. Allerdings ist es nicht sinnvoll, Verträge in Form von Interfaces hierfür zu verwenden, da diese Klassen lediglich zur Datenrepräsentation genutzt werden, ähnlich den Models, nicht aber zur Verarbeitung von Informationen.

- Common.Enum

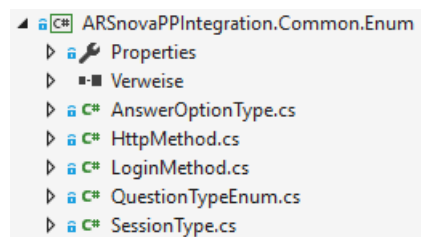


Abbildung 19: Auflistung der Dokumente im Projekt “Common.Enum”

Enum-Definitionen wie der Fragentyp können global verwendet werden, genau wie sie durch die Translators in dem Common-Projekt global übersetzt werden können. Es ist nicht sinnvoll, Übersetzer global zur Verfügung zu stellen, wenn die zu übersetzenden Datentypen nicht ebenfalls projektübergreifend bekannt sind. Für eine bessere Übersicht sind auch diese in einem eigenen Projekt gekapselt. Sie sind weder eine Vertragsdefinition, noch deren Implementierung oder eine Datenstruktur.

- Common.Helpers

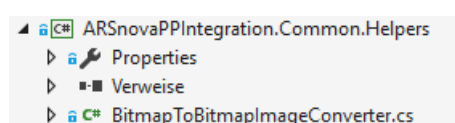


Abbildung 20: Auflistung der Dokumente im Projekt “Common.Helpers”

Das Helpers-Projekt der Common-Ebene ist eine Besonderheit. Auch sie stellt eine Hilfsklasse bereit, ähnlich wie der Lokalisierungsservice. Der Unterschied besteht hier in der Beschaffenheit: Der `BitmapToBitmapImageConverter` ist rein statischer Natur. Mit der Dependency-Injection, mit deren Hilfe alle anderen Objekte zur Verfügung gestellt werden, können keine statischen Klassen behandelt werden. Den Converter nun instanzierbar zu gestalten, nur damit mit der Dependency-Injection gearbeitet werden kann, ist Ressourcenverschwendung. Auch wenn die Menge der benötigten Ressourcen nicht groß ist, soll hier der optimale Weg gewählt werden. Da nicht bekannt ist, wie sich das Projekt weiterentwickeln wird, ist es immer sinnvoll, den saubersten Ansatz zu wählen. Daraus resultiert das Helpers-Projekt. Wäre die hier enthaltene Klasse in dem Common-Projekt abgelegt, müssten andere Schichten auf diese verweisen, um mit dem Converter arbeiten zu können. Das wäre allerdings ein Verstoß gegen die Architekturrichtlinien, daher werden alle statischen Hilfsklassen hier abgelegt.

- Common.Resources

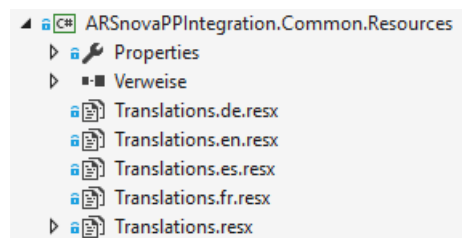


Abbildung 21: Auflistung der Dokumente im Projekt “Common.Resources”

In dem Resources-Projekt werden lediglich Informationen in Form von \*.resx-Dateien zur Verfügung gestellt. An dieser Stelle dürfen keine Datenstrukturen oder Logiken abgelegt werden. Auch dies dient der sorgsamsten Dokumenten- und Funktionstrennung. Da das Common-Projekt zur Zeit hauptsächlich mit den Übersetzungen arbeitet, sind die entsprechenden Dateien hier abgelegt. Der für diese Thesis entwickelte Prototyp unterstützt zwar nur die Sprachen Deutsch und Englisch, doch sind bereits Dateien für Französisch und Englisch angelegt. Die beiden ARSnova-Systeme unterstützen die genannten vier Sprachen, daher sollte auch diese Anwendung in naher Zukunft den gleichen Lokalisierungsumfang anbieten.

- Communication

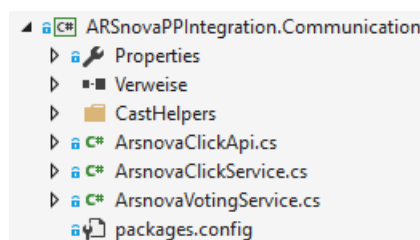


Abbildung 22: Auflistung der Dokumente im Projekt “Communication”



Das Projekt Communication beinhaltet die REST-API-Zugriffe auf die beiden ARS. Diese sind aus Gründen der Wartbarkeit im Falle einer Schnittstellen-Änderung ausgelagert. Neben einer Klasse für jede der beiden Services wurden mehrere arsnova.click-Aufrufe in eine weitere Datei entkoppelt, um eine höhere Kapselung zu erlangen. Viele Aufrufe konnten dadurch vereinfacht werden, gerade weil die arsnova.click-API sehr spartanisch ist und wenig überprüft. Um sicherzustellen, dass die Daten nicht korrupt sind und keine unerwarteten Fehler bei der Kommunikation mit arsnova.click aufkommen, ist der Implementierungsaufwand an dieser Stelle größer als bei der arsnova.voting-Schnittstelle. Des Weiteren beinhaltet der Ordner CastHelpers alle benötigten Klassen, um die zurückgegebenen JSON-Daten zu de-serialisieren und in die eigenen Models der Geschäftslogik zu konvertieren. Oft sind die Models lediglich abstrahiert und um nicht relevante Referenzen wie interne ID-Verweise ergänzt, damit ein problemloses Casten ermöglicht werden kann. Da diese Werte später nur verwirren, werden diese vor der Rückgabe an die aufrufenden Schichten auf die offiziellen Communication-Models reduziert. Dafür ist ein ObjectMapper nötig, der lediglich die Aufgabe hat, alle Werte zwischen zwei Objekten verschiedener Klassen, welche den gleichen Namen und Typ tragen, zu kopieren. Durch diese generische Lösung wird der große Aufwand gespart, für jede zu mappende Klassen-Paarung zwei Kopiermethoden zu schreiben, da eine Implementierung aufgrund der festen Übergabe- und Rückgabewerte lediglich ein unidirektionales Mapping zulässt.

- Communication.Contract

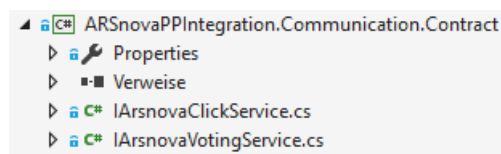


Abbildung 23: Auflistung der Dokumente im Projekt “Communication.Contract”

Wie bereits in den anderen Contract-Projekten beschrieben, befinden sich hier die Vertragsdefinitionen für die Kommunikations-Schicht.

- Communication.Model

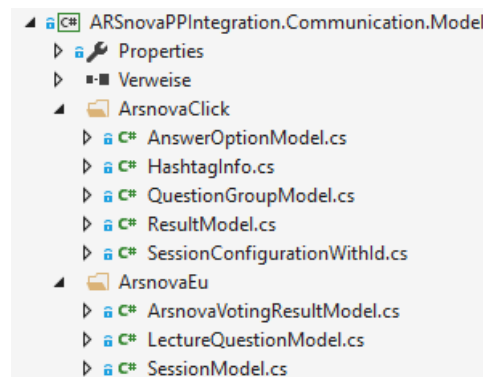


Abbildung 24: Auflistung der Dokumente im Projekt “Communication.Model”

Der Inhalt dieses Projektes ist analog zu dem Models-Projekt in der Geschäftslogik. Die Datenstrukturen sind nach der Schnittstellen-Zugehörigkeit in zwei Ordner gegliedert, um bei der Verwendung durch Namespaces Klarheit zu schaffen, ob die Models für den richtigen ARSnova-Service verwendet werden.

- Presentation

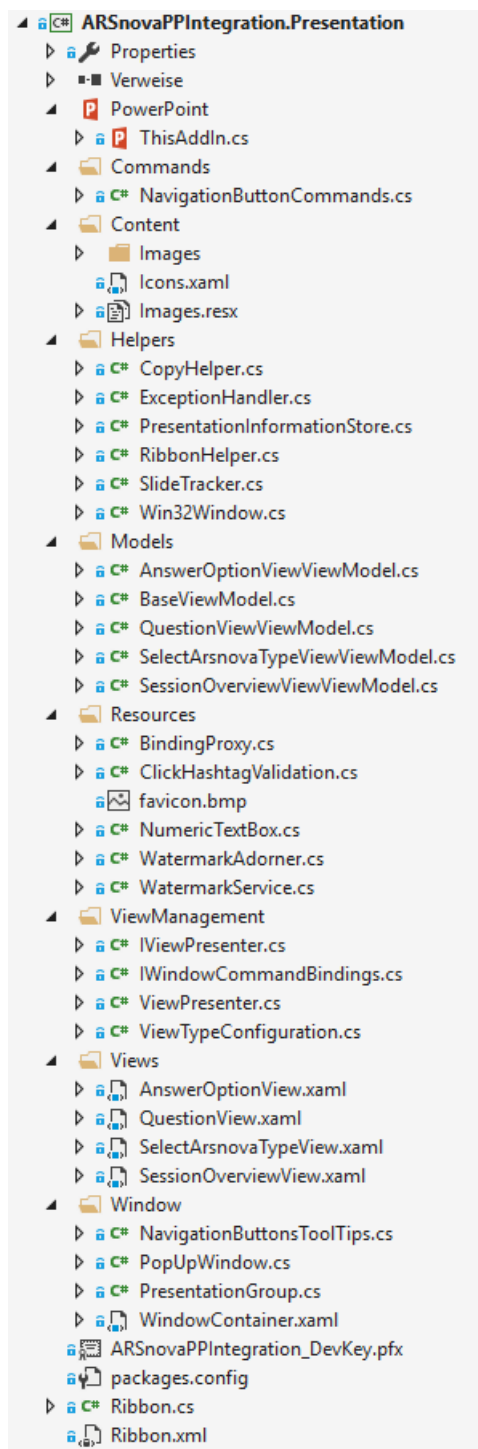


Abbildung 25: Auflistung der Dokumente im Projekt “Presentation”

Das Präsentations-Projekt beinhaltet vor allem die Darstellung mit der WPF und dem Model-View-ViewModel (MVVM)-Pattern und die Anbindung an PowerPoint. Für eine einfachere Zuordnung wurden viele Teile der Präsentations-Schicht in eigene Ordner und damit Namespaces aufgeteilt, so sollen diese auch erläutert werden:

- PowerPoint (ThisAddIn.cs): Die eigentliche Anbindung an PowerPoint. Hier werden die im Kapitel Präsentation beschriebenen Events gebunden, um auf Basis von PowerPoint-Geschehnissen interagieren zu können. Dies betrifft unter anderem das Selektieren einer Folie im Bearbeitungs-Modus, das Starten einer Präsentation oder das Einblenden einer neuen Folie im Präsentations-Modus. Auch wird die Anwendung von hier aus mit Hilfe der Bootstrapper-Klasse im Presentation.Configuration-Projekt initialisiert. Hierzu gehören auch die in der Projektexplorer-Abbildung 25 ganz unten aufgeführten Dateien “Ribbon.cs” und “Ribbon.xml”. Die XAML-Datei enthält die Erweiterungen der PowerPoint-Oberflächen, während die dafür benötigten Funktionen in der zugehörigen “cs”-Datei definiert sind. Dies sind unter anderem Icons, übersetzte Tooltip- oder Beschriftungstexte, Indikatoren für die Aktivität der Steuerelemente oder die nach einem Klick ausgeführten Methoden.
- Commands: Eine Definition aller Kommandos für die Views. In dem MVVM-Pattern existieren sogenannte “Commands”, die an bestimmte Oberflächenelemente wie Buttons gebunden werden können. Wird der Button betätigt, wird der Command getriggert. Das zugehörige Model kann diesen Command ebenfalls binden und bestimmte Interaktionen ausführen, wenn dieser ausgelöst wird. Auf diese Art und Weise werden Oberflächen-Aktionen im Vergleich zu dem bekannten Model-View-Controller (MVC)-Pattern ohne einen Controller ausgelöst.
- Content: In dem Content-Namespace sind alle für die Oberflächen benötigten Bilder sowie die im Vektor-Format definierten Icons enthalten.
- Helpers: Der Helpers-Ordner beinhaltet alle Klassen und Funktionen, die zur Aufarbeitung oder Darstellung der Daten benötigt werden und nicht von der Geschäftslogik übernommen werden. Der RibbonHelper wird von der vorher beschriebenen PowerPoint-Oberflächen-Erweiterung Ribbon.cs angesprochen, um alle benötigten Daten zu erhalten oder die Geschäftslogik anzusprechen. In dem ExceptionHandler werden die Fehlermeldungen für eine Darstellung aufbereitet, der PresentationInformationStore übernimmt das Sichern und Abrufen der Session-Daten in dem Präsentationsobjekt. Die SlideTracker-Klasse kapselt alle benötigten Zugriffe auf die Folien von PowerPoint. Win32Window dient zur Erzeugung von Hilfsobjekten zum Lokalisieren der aktuellen Fenster von PowerPoint, während der CopyHelper das Duplizieren aller möglichen Model-Objekte übernimmt, was bei dem Abbrechen von Änderungsfunktionen wichtig ist. Hierbei wird das Zurücksetzen dadurch simuliert, dass vor dem Ändern eine Kopie des aufgerufenen Objektes angefertigt wird. Wenn die Interaktion abgebrochen wird, ist immer eine Kopie des Objektes im alten Zustand vorhanden. Für das Zurücksetzen muss also lediglich die Objekt-Referenz ausgetauscht werden.

- Models: Der Models-Bereich beinhaltet alle Darstellungs-Models für das MVVM-Pattern. Dabei enthalten die Models möglichst wenig Aufbereitungslogik, da an dieser Stelle lediglich eine Repräsentation der Daten erfolgen soll.
- Resources: Dieser Bereich beinhaltet erweiterte Elemente für die Views. So abstrahiert die NumericTextBox-Klasse das WPF-Steuerelement TextBox, um ein Eingabefeld zu erzeugen, welches nur numerische Werte annimmt. Der WatermarkService ermöglicht das Anzeigen von Platzhaltern in Textfeldern. Auch reine Eingabe-Validierungsfunktionen wie die Hashtagvalidierung von arsnova.click wird an dieser Stelle durchgeführt. Durch diesen Service findet eine Validierung direkt in der Darstellungsschicht statt, ohne regelmäßige Abfragen an die Geschäftslogik und damit eventuell eingehende Performance-Einbußen zu erhalten, wenn der Anwender schnell tippt.
- ViewManagement: Innerhalb des ViewManagement werden die Views und Models verwaltet. Hier wird das Zuweisen einzelner Objekte durchgeführt, das Vorbelegen bestimmter Werte im Model und auch das Verwalten der Ressourcen. So müssen bereits aufgerufene Models beispielsweise nicht erneut gebaut werden, sondern werden eine Zeit lang gesichert. Dadurch wird Performance gewonnen. Ebenfalls bietet der ViewPresenter an, direkt nach dem Erzeugen eines ViewModels bestimmte Interaktionen in diesem auszuführen. Sollte beispielsweise in einer Ansicht durch das Betätigen eines “Weiter”-Buttons ausgelöst werden, dass eine neue Ansicht samt Model erzeugt wird, soll die vorherige Ansicht eventuell terminiert werden, bevor die neue angezeigt wird. Sollte der Aufrufer vor der Erzeugung einer neuen Ansicht terminieren, hat dieser nicht mehr die Chance, die Initialisierungsmethode bei der neuen Instanz aufzurufen. Diese sollte sich aber auch nicht selbst vorbereiten, da sie im Konstruktor nicht wissen kann, wer der Aufrufer war und ob beispielsweise bestimmte Buttons aktiviert werden sollen, die normalerweise nicht zur Verfügung stehen. Dies übernimmt an dieser Stelle der ViewPresenter, er hat die Übersicht über alle Darstellungsansichten.
- Views: Hier liegen alle Views im XAML-Format vor. Die jeweilige Bezeichnung ergibt sich dabei aus dem Namen des zugehörigen ViewModels, lediglich ohne das Suffix “ViewModel”.
- Window: Alle Views enthalten nur den Inhalt eines Fensters selbst. Das liegt daran, dass die meisten Views das gleiche Fenster verwenden können, wenn sich dieses dynamisch an die unterschiedlichen Größen der Inhalte anpasst. Daher muss das Fenster selbst, der WindowContainer, nur einmal definiert werden und liegt in dem gesonderten Window-Namespace vor. Zusätzlich sind hier auch schon die Navigations-Buttons des unteren Oberflächen-Bereichs definiert, mit denen es unter anderem möglich ist, das nächste Fenster aufzurufen, das aktuelle zu schließen oder auch Vor- und Zurück-Funktionen auszuführen. Über die zugehörigen CommandBindings entscheidet das darin dargestellte ViewModel, welche dieser Buttons eine Funktion besitzen sollen. Alle Command-Bindings, denen keine Ausführungsmethode zugewiesen ist, werden automatisch ausgeblendet. Die Klasse PopUpWindow stellt gleichnamiges Verhalten zur Verfügung. Dabei wird zwischen Informations-, Bestätigungs- und Fehler-Pop-up-Fenstern

unterschieden, die sich in Ihrer Darstellung beispielsweise durch das Icon im oberen, linken Teil des Fensters unterscheiden.

- Presentation.Configuration

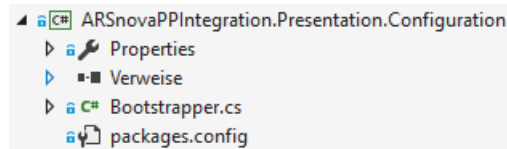


Abbildung 26: Auflistung der Dokumente im Projekt “Presentation.Configuration”

An dieser Stelle werden alle benötigten Konfigurationen vor dem Start des Add-ins getroffen. Das Bootstrapper-Objekt wird zum Start des Add-ins, welches nach der Installations-Konfiguration direkt bei dem Start von PowerPoint geladen wird, erzeugt und übernimmt das Vorbereiten der Anwendung. Da die meisten wichtigen Einstellungen bereits von dem Hostsystem getroffen wurden, bleibt hier lediglich noch die Erstellung des Unity-Containers für die Dependency-Injection. Dieser Prozess ist in dem gleichnamigen Kapitel auf Seite 56 beschrieben.

- Test

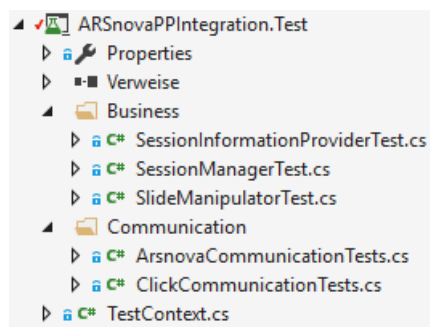


Abbildung 27: Auflistung der Dokumente im Projekt “Test”

Das Test-Projekt deckt in erster Linie die Überprüfung der Geschäftslogik und der Kommunikations-Schnittstelle ab. Dabei sind diese getrennt und der Test der Kommunikationsschnittstelle erfolgt zuerst. Da der größte Teil dieser Applikation auf die Konnektivität zu den ARS aufbaut, ist sicherzustellen, dass die Schnittstellenzugriffe funktionieren. Die Tests der Geschäftslogik erfolgen dabei mit einem eigenen Textkontext, der die Schnittstellen zu den ARSnova-Systemen simuliert und vordefinierte Daten zurückgibt. Dadurch ist eine Überprüfung der Geschäftslogik-Implementierung unabhängig von den beiden ARS möglich.

Jedes einzelne Projekt enthält jeweils ein Properties sowie Verweise-Verzeichnis. In den Properties werden die Projekteigenschaften hinterlegt, beispielsweise die Assembly-Informationen der erzeugten “dll”-Dateien. Auch können in dieses Verzeichnis andere Dokumente wie Ressourcen-Dateien hinzugefügt werden. In den Verweisen werden alle benötigten Projekt- oder Assembly-Referenzen hinterlegt, die zur Ausführung des Projektes benötigt werden. Die packages.config-Datei enthält Informationen zu den zur Ausführung benötigten NuGet-Paketen. NuGet ist der Paketmanager für die .NET-Plattform.

## 4.4 Entwurfsmuster

Die erläuterte Architektur definiert die Struktur einer Software, nicht aber die Lösungswege für diese. Lösungsansätze werden in der Softwareentwicklung in Form von Entwurfsmustern beschrieben. Einige davon werden bereits bei der Architektur selbst benötigt, beispielsweise das Single Responsibility Principle, welches besagt, dass jede Klasse lediglich eine Aufgabe zu erfüllen hat und auch nur Methoden für dieses eine Ziel beinhaltet (Martin, 2003).

Viele andere Entwurfsmuster werden benötigt, um die definierte Projektstruktur wirklich umzusetzen. Hauptsächlich betrifft das die Kopplung der Komponenten, da genau diese Beziehungen untereinander in der Projektstruktur abgebildet werden. Alle dafür eingesetzten Entwurfsmuster sind in diesem Abschnitt beschrieben.

### 4.4.1 Separated Interface

Eine Entkoppelung von einzelnen Softwarekomponenten kann erzielt werden, indem diese nur über eine Form von Abstraktion miteinander kommunizieren. Das heißt, es muss eine abstrakte Klasse oder ein Interface geben, das einen Service innerhalb der Anwendung definiert. Die Implementierung des Interfaces beziehungsweise die abgeleitete Klasse ist den restlichen Softwarekomponenten nicht bekannt und somit entkoppelt. Die entsprechende Dienst-Implementierung kann durch eine andere Realisierung der gleichen Abstraktion ausgetauscht werden, ohne dass restlicher Quellcode angepasst werden muss. Wichtig ist dabei, dass die Schnittstellen-Definition, in diesem Projekt Interface, in einer eigenen Assembly definiert sind, welche von der Implementierung und den Verwender referenziert werden kann. (Fowler, 2003, S. 476 ff.).

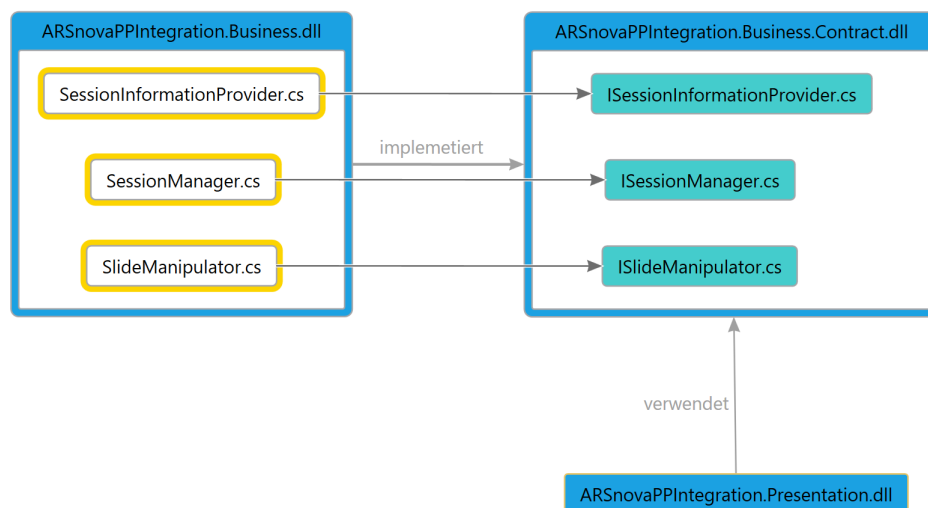


Abbildung 28: Beispiel am Projekt: Separated Interface-Pattern

Das Beispiel zeigt das Entwurfsmuster Separated Interface an dem Integrationsprojekt. Die Assembly `ARSnovaPPIntegration.Business.dll` implementiert die Interfaces der Contract-Assembly, während andere Schichten, wie beispielhaft in der Grafik dargestellt die Präsentations-Ebene, lediglich auf die abstrakte Definition (in der Abbildung türkis markiert) verweist und damit arbeitet.

#### 4.4.2 Service Locator

Das Service Locator Entwurfsmuster beschreibt ein Objekt, welches innerhalb einer Anwendung die Implementierungen von Diensten zur Verfügung stellt (Fowler, 2004). Zur Identifikation des benötigten Services kann eine Zeichenkette oder ein Typ übergeben werden. In diesem Fall erfolgt hierbei immer die Übergabe des Interface-Typs, da die Implementierungen der Dienste selbst durch das Separated Interface-Pattern nicht bekannt sind. Gleichzeitig wird hiermit die Erfüllung des vorherigen Patterns sichergestellt, da es ein Objekt geben muss, welches zur Laufzeit die Implementierungen der Service-Interfaces übernehmen muss, damit auf die Logik dahinter zugegriffen werden kann.

Interessant ist hierbei die Abgrenzung zu dem Registry-Entwurfsmuster, welches technisch gesehen dem gleichen Prinzip entspricht, aber eine andere Zielsetzung verfolgt (Fowler, 2003, S. 480). Während das Registry-Pattern anstrebt, beliebige Objekte durch einen Schlüssel zu identifizieren, dient das Service Locator-Pattern einzig der Entkopplung von Softwarekomponenten. Hierbei werden nicht beliebige Objekt-Schlüssel-Beziehungen aufgelöst, sondern nur die spezielle Beziehung zwischen dem Interface eines Services und seiner Implementierung.

Ein Service Locator muss zum Start der Anwendung initialisiert werden. Hierbei werden dem Interface jeweils die entsprechenden Implementierungen zugewiesen, damit dem Locator die Zugehörigkeiten bewusst werden. In der Regel kann hier über einen Parameter festgelegt werden, ob lediglich eine Instanz der Realisierung erzeugt wird und die entsprechende Referenz bei einer Auflösungsanfrage zurückgegeben wird oder jedes mal ein neues Objekt angelegt wird. Hier spielen die Aspekte der Datentrennung oder Performance eine wichtige Rolle. In manchen Anwendungsfällen sollen nicht viele Instanzen eines komplexen Objektes erzeugt werden, wenn dieses oft aufgelöst werden soll. Allerdings könnte es sich auch um eine Serveranwendung handeln, bei der nutzerspezifische Daten im Service hinterlegt werden - an dieser Stelle muss eine strikte Trennung erfolgen und jeder Nutzer benötigt seine eigene Service-Instanz.

Um die bestmögliche Performance zu gewährleisten, wird in der PowerPoint-Integration von ARSnova jede Service-Klasse nur einmal instantiiert. Alle Services sind so aufgebaut, dass diese keine sensiblen Daten kennen oder sichern, sondern lediglich Daten verarbeiten und die Ergebnisse ohne eine Zwischenspeicherung zurückgeben. Dadurch kann es niemals zu Problemen bei der Wiederverwendung der Instanzen kommen, beispielsweise durch korrupte Daten, da es von der vorherigen Verwendung keinen Kontext gibt, der zu veralteten oder einfach falschen Daten führen könnte.

#### 4.4.3 Dependency Injection

Wenn der beschriebene Service Locator verwendet wird, muss eine Klasse die benötigten Service-Instanzen, meist im Konstruktor, auflösen um damit arbeiten zu können:

---

```
public class RibbonHelper
{
    private ILocalizationService localizationService;

    public RibbonHelper ()
    {
        this.localizationService =
            ServiceLocator.Current.GetInstance<ILocalizationService>();
    }
}
```

---

Listing 1: Abhängigkeitsauflösung mit einem Service Locator

In diesem Beispiel ist lediglich das Abrufen einer Implementierung enthalten, nicht das Erstellen des Service Locators und dessen Initialisierung. Hierbei hat die Klasse die volle Kontrolle darüber, wann und von wem der benötigte Dienst aufgelöst wird. Durch den Freiraum bei dem Abrufen wird die Möglichkeit eröffnet, die Implementierungen auf eine andere Art und Weise als den eigentlichen Service Locator zur Verfügung zu stellen. Diese Freiheit soll die Klasse aber gar nicht besitzen, neben der Verwalter-Rolle des Locators wird ein Pattern für die Verteilung der Dienstaufösungen benötigt: Die Dependency Injection, eine Umsetzung des Inversion of Control (IoT)-Ansatzes. Hierbei wird einer Klasse die Referenz auf die Implementierung eines Interfaces hereingereicht, ohne dass diese in dem Moment weiß, wer den überreichenden Part darstellt. Man spricht von einer Injizierung der benötigten Abhängigkeiten. Daraus leitet sich der Name dieses Patterns ab.

Die Injektion der Abhängigkeiten kann dabei auf eine von drei bekannten Arten und Weisen stattfinden: Konstruktor-, Interface- oder Setter-Injection. In diesem Entwicklungsprojekt wird die Konstruktor-Injection zur Anwendung kommen, da der Entwickler mit dieser Variante im Konstruktor selbst entscheiden kann, wie die Abhängigkeit weiter verwendet werden soll. Auch kann die Abhängigkeit in privaten Attributen gesichert werden und ist damit nicht mehr von außen zugänglich. Je nach Art des Services soll dieser nicht von anderen Teilen der Anwendung manipulierbar sein. Im Gegenzug dazu muss der Setter-Injection eine Eigenschaft zur Verfügung stellen, über die die Abhängigkeit geladen werden kann. Da dieser öffentlich sein muss, kann über diesen Zugriff auch die Manipulation von außerhalb erfolgen. Mit internen Abfragen kann einem solchen Missbrauch zwar vorgebeugt werden, doch bedeutet dies einen zusätzlichen Implementierungsaufwand. Bei der Interface-Injection werden die benötigten Abhängigkeiten durch das Interface des Services erkannt und können über einen Methodenaufruf der Schnittstellenbeschreibung geladen werden. Sollte ein Service eine neue Abhängigkeit benötigen, muss mit der Implementierung auch das Interface mit angepasst werden. Das widerspricht dem Entkopplungs-Prinzip Separated Interfaces, nach dem eine Änderung der Dienst-Logik im besten Falle unabhängig von der Schnittstellenbeschreibung sein soll. Um einen Vergleich zu der Auflösung mit dem Service Locator zu schaffen, folgt eine Implementierung der gleichen Logik aus dem vorherigen Code-Beispiel Abhängigkeitsauflösung mit einem Service Locator:



---

```
public class RibbonHelper
{
    private ILocalizationService localizationService;

    public RibbonHelper (ILocalizationService
        localizationService)
    {
        this.localizationService = localizationService;
    }
}
```

---

Listing 2: Abhängigkeitsauflösung mit Konstruktor-Injektion

Unschwer ist zu erkennen, dass der Code neben den oben beschriebenen Vorteilen nun auch deutlich übersichtlicher wirkt. Das eigentlich Interessante ist hier jedoch das Zusammenspiel mit dem Service-Locator. Dieser verwendet einen Container, in dem die Zuordnungen und bereits instantiierten Klassen verwaltet werden. Der Container kann Dependency-Injection-fähig sein, in diesem Beispiel der Unity-Container, welcher die Konstruktoren auf benötigte Parameter durchsucht und automatisch injiziert, wenn sie in der Container-Kollektion bereits zur Verfügung stehen. Folglich muss die Reihenfolge der Registrierung nach den entsprechenden Abhängigkeiten sortiert werden, da alle Konstruktor-Parameter der nächsten Klasse bereits im Unity-Container vorhanden sein müssen, um das neue Objekt initialisieren zu können. Beispielsweise kann `IExampleServiceB`, welcher die Konstruktor-Injektion von einem Parameter `IExampleServiceA` erwartet, erst nach dessen Registrierung angelegt werden. Ansonsten ist dem Container der Typ nicht bekannt und es gibt entsprechende Kompilierungs-Fehler. Einem solchen Verhalten kann vorgebeugt werden, indem ein Lazy-Container verwendet wird, in dem die Objekt-Erzeugung nicht zum Registrierungszeitpunkt, sondern erst zum Zeitpunkt der Auflösung einer Abhängigkeit mit dem benötigten Typ ausgeführt wird. Dies würde eine spätere Registrierung für den Kompilierungsprozess zulassen, sollte die Eintragung allerdings nicht erfolgen, gibt es einen Laufzeitfehler bei der ersten Auflösung und damit auch Erzeugung der entsprechenden Abhängigkeit.

Nach der obigen Entwurfsmuster-Beschreibung ergibt sich folgende Implementierung:

---

```
public class Bootstrapper
{
    public IUnityContainer UnityContainer { get; private
        set; }

    public void SetupUnityServiceLocator()
    {
        this.UnityContainer =
            this.GetRegisteredUnityContainer();
        ServiceLocator.SetLocatorProvider(() => new
            UnityServiceLocator(this.UnityContainer));
    }

    private IUnityContainer GetRegisteredUnityContainer()
    {
        var unityContainer = new UnityContainer();

        // Type registration
        unityContainer
            .RegisterType<ILocalizationService,
                LocalizationService>(new
                ContainerControlledLifetimeManager())
            .RegisterType<IArsnovaVotingService,
                ArsnovaVotingService>(new
                ContainerControlledLifetimeManager())
            .RegisterType<ISlideManipulator,
                SlideManipulator>(new
                ContainerControlledLifetimeManager())
            .RegisterType<IArsnovaClickService,
                ArsnovaClickService>(new
                ContainerControlledLifetimeManager())
            .RegisterType<IQuestionTypeTranslator,
                QuestionTypeTranslator>(new
                ContainerControlledLifetimeManager())
            .RegisterType<ISessionInformationProvider,
                SessionInformationProvider>(
                new ContainerControlledLifetimeManager())
            .RegisterType<ISessionManager,
                SessionManager>(new
                ContainerControlledLifetimeManager());

        return unityContainer;
    }
}
```

---

Listing 3: Initialisierung des Dependency-Injection Containers

Die `ContainerControlledLifetimeManager`-Instanz, die jeder Unity-Container-Registrierung mit übergeben wird, sorgt für die einmalige Instanziierung der Interface-Implementierung und Rückgabe der erzeugten Objekt-Referenz bei jeder Auflösungsanfrage durch Konstruktor-Injektion. Damit wird einer nicht ressourcenschonenden, zweiten Erzeugung vorgebeugt.

## 4.5 Model-View-ViewModel

MVVM ist ein für WPF optimiertes Entwurfsmuster für die Präsentations-Ebene. Dabei übernehmen Model und ViewModel die abstrakte Darstellung der Daten, die Datenhaltung, während die View für die Darstellung dieser verantwortlich ist. Das ViewModel stellt in der Drei-Komponenten-Beziehung die Steuerungslogik dar, es vermittelt zwischen View und Model. Änderungen von Daten in der View oder im Model werden direkt durchgereicht und im Hintergrund oder der Darstellung angepasst. Das wäre auch durch ein direktes Binding der View auf das Model möglich, doch übernimmt das ViewModel noch die Aufgabe der Nachrichten- und Command-Verwaltung. Bestimmte Nutzereingaben triggern Commands, welche an View und ViewModel gebunden sind - das Model interessiert sich als reine Datenhaltung nicht für Nutzerinteraktionen wie das Schließen einer Ansicht. Das ViewModel hingegen enthält die Logik für solche Interaktionen und löst entsprechende Events aus, um beispielsweise neue Ansichten anzuzeigen oder eine Methode in der Geschäftslogik aufzurufen. Folglich ist das ViewModel als Kommunikationsmittel zwischen Darstellung und Datenhaltung zu sehen, welches alle Aufgaben übernimmt, die über die Aufgaben der abstrakten und konkreten Darstellung hinausgeht (Microsoft, 2016).

Damit die View die darzustellenden Daten abrufen kann, benötigt es einen Zugriff auf das ViewModel. Das erfolgt, indem das ViewModel als DataContext der View gesetzt wird. Dadurch stehen alle Eigenschaften des ViewModels für die View zur Verfügung und diese kann auf diese "binden", also eine Verknüpfung herstellen. Dadurch ist eine direkte Aktualisierung der Daten bei einer Änderung im Model gewährleistet, wenn die entsprechenden Eigenschaften Dependency-Properties sind oder das ViewModel die INotifyPropertyChanged-Schnittstelle implementiert.

## 4.6 Installation

Wie bereits in dem Analyse-Abschnitt Installation festgesetzt, soll für die optimale Verbreitung der Integrations-Software eine Installationsroutine bereitgestellt werden. Um die Komplexität möglichst gering zu halten, wird hierfür der InstallShield Limited-Edition verwendet, welcher für alle Visual Studio Installationen gratis verfügbar ist. Die Konfiguration des InstallShields kann in sechs verschiedene Aufgabenbereiche gegliedert werden (Microsoft, 2017b):

- Anwendungsinformation

Der InstallShield benötigt mehrere Metadaten, um die Installationsroutine für die entsprechende Anwendung zu individualisieren. Dazu gehört beispielsweise der Name des Unternehmens, der Anwendungsname selbst, die aktuelle Version sowie eine Informationswebsite und ein Logo. Diese Daten tauchen in der Windows Software-Übersicht wieder auf und identifizieren damit die Anwendung.

- Voraussetzungen

Das Setup soll neben dem PowerPoint Add-in und den benötigten DLL-Dateien wie der VSTO-Bibliothek auch das benötigte .NET-Framework 4.6 installieren, wenn dieses unter der bestehenden Windows-Installation noch nicht zur Verfügung steht. Alle benötigten Elemente müssen bereits im InstallShield zur Verfügung stehen, ein Download der fehlenden Bibliotheken oder des Frameworks ist nicht gewünscht. Sollte es Komponenten geben, die bereits vor der Installation

vorhanden sein müssen, können diese bei den Voraussetzungen angegeben werden. In dem Fall der Integrationslösung muss eine Microsoft Office Installation vorhanden sein, sonst macht ein PowerPoint Add-in keinen Sinn.

- Anwendungsdateien

Um eine Anwendung zu installieren, müssen die Assemblies der Anwendung angegeben werden. Neben der ARSnovaPPIntegration.Presentation.vsto-Datei, welche das Add-in selbst darstellt, gehören auch die restlichen Projektausgaben dazu, in denen sich die benötigten Geschäftslogiken oder die Kommunikations-Schnittstelle befinden. Man kann den InstallShield beauftragen, die primären Projektausgaben hinzuzufügen oder man wählt die DLL-Dateien direkt aus dem Build-Verzeichnis. Im ersten Fall wird das Projekt vor der Zusammenstellung der Installationsroutine gebaut um die benötigten Ausgaben zu erzeugen.

- Verknüpfungen

Im Bereich der Verknüpfungen kann der Startmenü-Eintrag und eine Desktopverknüpfung definiert werden. Da die Integrationsanwendung lediglich in PowerPoint zur Verfügung steht und nicht alleinstehend gestartet werden kann, wird dieser Definitionsbereich ignoriert.

- Registrierdatenbank

Die Registrierdatenbank, auch bekannt unter der Kurzbezeichnung "Registry", ist die zentrale und hierarchische Konfigurationsdatenbank von Windows. Office Add-ins müssen an dieser Stelle registriert werden, damit sich PowerPoint bewusst wird, dass ein neues Add-in installiert wurde. Je nach Installationsart gibt es dabei zwei Orte, an denen die benötigten Einträge hinterlegt werden können. Wird die Anwendung nur für den aktuell angemeldeten Anwender installiert, müssen Einträge in den übergeordneten Schlüssel "HKEY\_CURRENT\_USER" erfolgen. Soll die Anwendung für alle Benutzer der Windows-Version zur Verfügung gestellt werden, müssen die Werte unter dem Pfad "HKEY\_LOCAL\_MACHINE" gesichert werden. Im letzteren Fall müssen die Werte doppelt erfasst werden, einmal für die 32-Bit-Anwendungen und einmal für die 64-Bit-Programme. Die doppelte Erfassung ist nötig, da vor der Installation nicht bekannt ist, ob eine 32- oder 64-Bit-Variante von Office installiert ist. Wäre dies bekannt, könnten lediglich die Schlüssel für den entsprechenden Bereich angelegt werden.

In dem jeweiligen Software-Ordner des übergeordneten Schlüssels und der Bit-Version muss folgende Verzeichnishierarchie erzeugt werden: "Microsoft ⇒ Office ⇒ PowerPoint ⇒ *Addins*". Die nachfolgende Tabelle beinhaltet die Schlüssel, die dort hinterlegt werden müssen:

Name	Typ	Beschreibung
Description	String	Die Beschreibung des Add-ins, welche in der PowerPoint Add-in-Verwaltung geführt wird.
FriendlyName	String	Der Name des Add-ins, welcher in der PowerPoint Add-in-Verwaltung geführt wird.
LoadBehaviour	DWORD	Gibt an, wann das Add-in geladen wird. Hier wird in der Regel '3' gewählt. Das bedeutet, dass das Add-in bei dem Start von PowerPoint mitgeladen wird. Ein anderes Verhalten ist selten erwünscht, die restlichen Möglichkeiten sind hier nachzulesen: <a href="https://msdn.microsoft.com/de-de/library/bb386106.aspx#LoadBehavior">https://msdn.microsoft.com/de-de/library/bb386106.aspx#LoadBehavior</a> .

- Komponenten-Auswahl

InstallShield stellt verschiedene Optionen für die Installationsroutine zur Verfügung, unter denen gewählt werden kann:

- Einblenden und zustimmen zu einer beliebigen Lizenzvereinbarung.
- Eine Aufforderung, den Nutzer- und Firmen-Namen einzugeben.
- Den Nutzer den Installationspfad frei wählen zu lassen.
- Den Anwender entscheiden lassen, die Anwendung direkt nach der Installation auszuführen.

Da es sich hierbei um ein Add-in handelt, fallen die letzten drei Optionen weg. Der Pfad darf nicht geändert werden und eine direkte Ausführung ist ohne PowerPoint nicht möglich. Auch werden keine Daten des Anwenders wie beispielsweise der Name benötigt. Lediglich das Einblenden und Bestätigen der allgemeinen Lizenzvereinbarungen des ARSnova-Projektes, der GNU General Public License v3.0, ist gewünscht.

Nach der erfolgreichen Konfiguration und dem Kompilieren der gesamten Anwendung im Release-Modus, kann das Setup-Projekt mit einer Single-Image-Konfiguration erstellt werden. Es ist darauf zu achten, dass die Projektmappe im Release- und nicht im Debug-Modus erstellt wird, da ansonsten unnötige Meta- und Debug-Informationen mit erstellt werden, die für die reine Ausführung nicht benötigt werden.

Eine solche Setup-Installation sollte in eine eigene Projektmappe ausgelagert werden, da der Inhalt nichts zu der eigentlichen Anwendung beiträgt, sondern diese lediglich zur Verfügung stellt. Um den Einstieg für weitere Entwickler möglichst einfach zu halten, ist der InstallShield in der Integrations-Projektmappe ARSnovaPPIntegration mit enthalten. In der Projektstruktur wurde der entsprechende Eintrag (ARSnovaPPIntegration.Installer.Setup) aufgrund der fehlenden Laufzeitabhängigkeit nicht mit beschrieben.

## 4.7 Integration von ARSnova in PowerPoint

Zur Integration von arsnova.voting in PowerPoint wird die bereits bestehende REST-Schnittstelle verwendet. arsnova.click hingegen bietet noch keine Schnittstelle, mit der eine Kommunikation von außen möglich ist. Folglich muss diese Softwarelösung noch

erweitert werden. In der Analyse wurden bereits die verschiedenen Aspekte der möglichen REST-API-Implementierungen mit Meteor erörtert. Für die GET-Requests, welche keine schreibenden Zugriffe benötigen, wird das Simple-Rest-Paket verwendet, alle anderen Schnittstellendefinitionen werden in eine eigene Methode in der “route.js”-Datei des Servers implementiert.

#### **4.7.1 Technische Abläufe**

Um ein Verständnis für die Kommunikation zwischen der Integrationslösung und den beiden ARS zu erlangen, wird ein beispielhaftes Anwendungsszenario mit Hilfe von Sequenzdiagrammen erstellt. Der Anwender entscheidet sich für eine Session-Art, legt eine Frage an und bearbeitet diese im Anschluss wieder, da der Fragetext nicht aussagekräftig war. Später wird die Präsentation gestartet, die Umfrage durchgeführt und die Ergebnisse dargestellt. Da sich die Abläufe und API-Zugriffe je nach gewähltem System unterscheiden, werden `arsnova.click` und `arsnova.voting` jeweils in einzelnen Sequenzdiagrammen dargestellt:

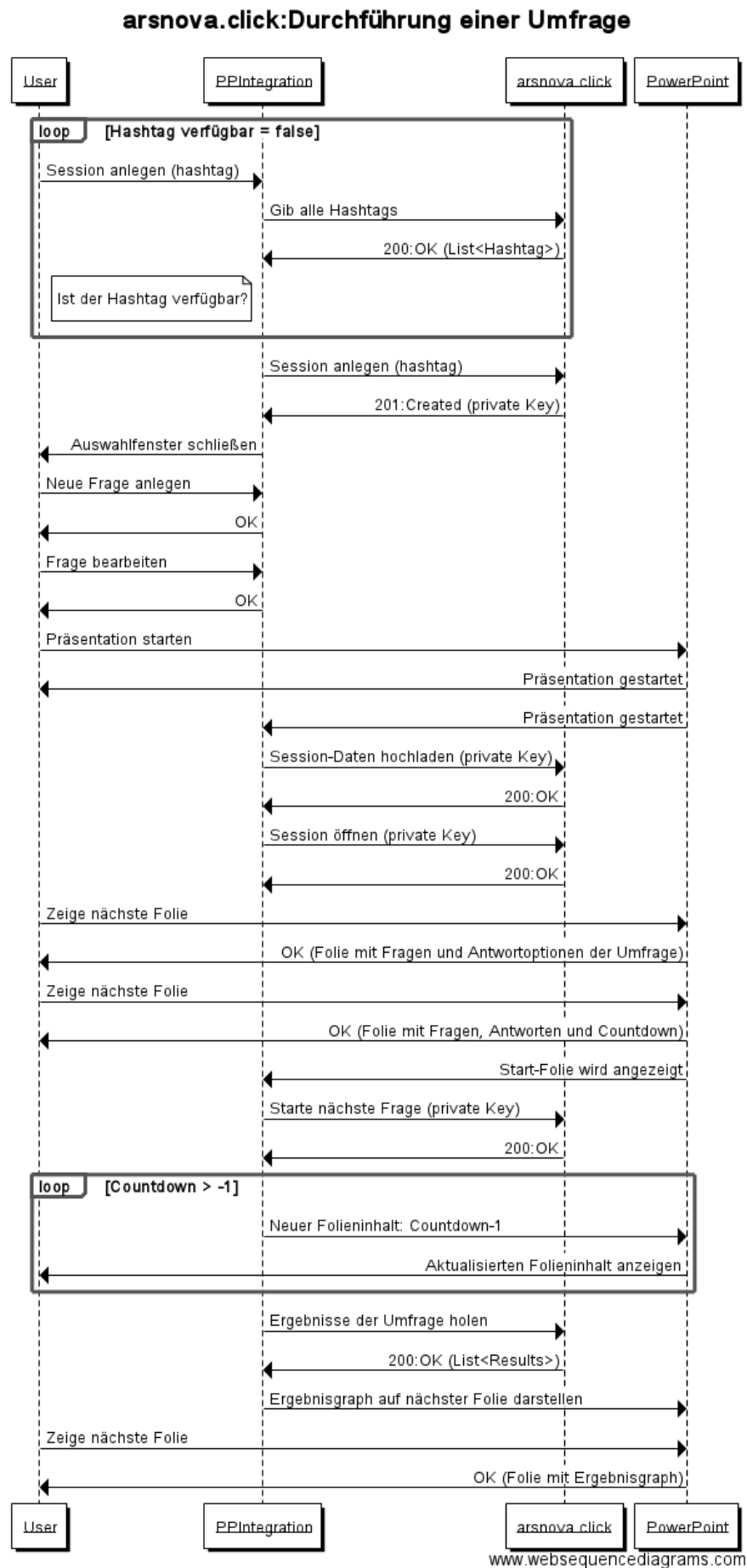


Abbildung 29: Sequenzdiagramm: Eine Umfrage in arsnova.click

Bei dem beispielhaften Anwendungsszenario von arsnova.click fehlt eine Schnittstellenkommunikation. Um den strengen Datenschutzbestimmungen von Hochschulen gerecht zu werden, entfernt das gamifizierte ARSnova-Projekt alle Session-bezogenen Daten vom Server, sobald diese nicht mehr aktiv ist. Dies wird mit Hilfe eines “Keep-Alives” durchgeführt. Die Session des Vortragenden teilt dem Server alle fünf Minuten mit, dass sich der Anwender noch aktiv in der Anwendung befindet. Wird die Website und damit die Applikation geschlossen, bleibt der “Heartbeat” aus. Hat sich die Anwendung länger als 10 Minuten nicht bei dem Server gemeldet, geht dieser von einer Beendigung der Session aus und entfernt alle verbleibenden Session-Daten. Um diesen “Heartbeat” zu simulieren gibt es eine eigene Schnittstellen-Methode, welche alle fünf Minuten von der Integrationslösung angesprochen wird. Da ein Sequenzdiagramm einen Ablauf ohne Zeitbezug darstellt, wurde dieser fünf-minütige Intervall nicht mit aufgeführt.

Die automatisierte Aufräumen-Funktion von arsnova.click ist auch der Grund, warum die Session Daten, also die Session-Konfiguration, die Fragen und die Antwortoptionen, erst zu Beginn der Präsentation zum Server übertragen werden. Eine vorherige Übertragung wäre nicht sinnvoll, da die Daten nach spätestens zehn Minuten wieder von dem Server entfernt werden. Außerdem wäre das Datenschutz-Konzept durchbrochen, wenn außerhalb der eigentlichen Umfragen Zugriffe auf Serverdaten erfolgen.

Die Authentifizierung erfolgt in arsnova.click mit dem “private Key”, welcher eindeutig einem Hashtag zugeordnet ist. Dieser muss bei jedem Schnittstellenzugriff, welcher nur für den Besitzer der Session verfügbar ist, mit übergeben werden, damit arsnova.click die Zugriffsberechtigungen überprüfen kann.

Der exemplarische Ablauf zur Erstellung einer arsnova.voting-Session nach dem vorherig definierten Ablauf:



## arsnova.voting: Durchführung einer Umfrage

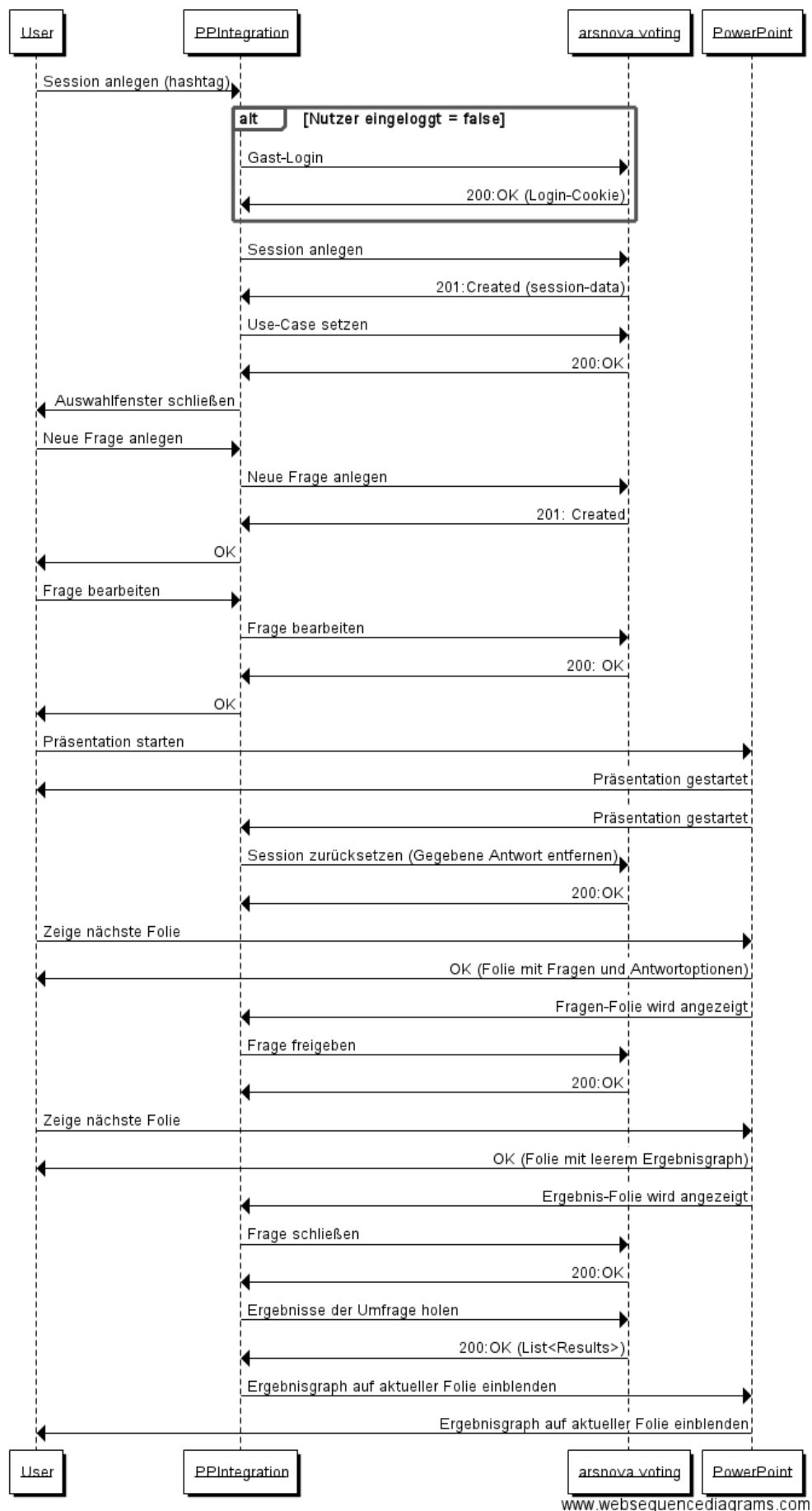


Abbildung 30: Sequenzdiagramm: Eine Umfrage in arsnova.voting

Hier erfolgt die Authentifizierung über einen Cookie, welcher eine “JSession-ID” enthält. Dieser wird nach dem Login als Gast automatisch an jeden Request als Cookie mit angehängt.

Ein weiterer Unterschied der beiden Sequenzdiagramme besteht in dem Darstellen der Umfragen-Ergebnisse. Bei einer arsnova.click Frage weiß die Anwendung, dass nach dem Ende des Countdowns keine neuen Ergebnisse mehr dazukommen und kann daher bereits die aktuellen Ergebnisse von dem Server anfordern und den Inhalt der nächsten Folie vorbereiten. Bei arsnova.voting hingegen wird die Frage-Runde beendet, sobald die Ergebnisfolie geöffnet wird. Erst dann können die Ergebnisse von dem Server abgerufen und dargestellt werden. Da der Graph-Erzeugungsprozess sowie das Abrufen der Ergebnisse selbst, je nach Hardware und aktueller Internetverbindung, einige hundert Millisekunden dauern kann, ist hier eine Verzögerung ersichtlich, nach der die Ergebnisse auf der entsprechenden Folie dargestellt werden.

#### 4.7.2 Schnittstellenverwendung von arsnova.voting

Für die Anbindung von arsnova.voting müssen keine weiteren Möglichkeiten evaluiert werden, da die Web-Anwendung in ein Back- und Frontend unterteilt ist. Die Kommunikation zwischen diesen beiden Schichten erfolgt über eine REST-API und eine WebSocket (WS)-Verbindung. Alle über die WS-Verbindung zur Verfügung gestellten Daten sind ebenfalls über eine REST-Schnittstelle verfügbar. Demnach sind bereits alle benötigten Operationen implementiert, folgend werden lediglich alle verwendeten Funktionen aufgelistet, damit diese für den Fall einer späteren Anpassung dokumentiert sind. Gerade in Hinblick auf die geplante Einführung des neuen arsnova.voting-Backends in 2017 (siehe arsnova.voting) ist diese Art der Informationssicherung unabdingbar. Die benötigten Zugriffe ergeben sich zum größten Teil aus dem vorher definierten technischen Ablauf der arsnova.voting Integration.

Pfad	Methode	Beschreibung	Body-Content
/auth/login?type=guest &user=user_name	GET	Gast-Login	-
/lecturerquestion/ question_id	DELETE	Löscht eine Frage	-
/lecturerquestion/ question_id	PUT	Updatet den Inhalt einer Frage	Frage-Model
/lecturerquestion/ question_id/answer	DELETE	Entfernt alle gegebenen Antworten der Frage	-
/lecturerquestion/ question_id/answer/ ?piround=1	GET	Liefert die gegebenen Antworten zu der Frage	-
/lecturerquestion/ question_id/publish	POST	Veröffentlichen oder Deaktivieren einer Frage	Frage-Model
/session	POST	Erstellt eine neue Session	Session-Model
/session/session_id/ question	POST	Legt eine neue Frage an	Frage-Model

Die Body-Parameter wurden größtenteils als “Models” zusammengefasst, da zur Zeit sehr viele, meistens nicht benötigte, Parameter übergeben werden, deren Auflistung den Rahmen der Tabelle gesprengt hätte und einer Übersicht nicht mehr dienlich sind.

#### 4.7.3 Schnittstellendefinition von arsnova.click

Aus dem definierten, technischen Ablauf ergibt sich bereits der größte Teil der benötigten Schnittstellendefinitionen. Die weiteren Pfade haben sich im Laufe der Implementierung als Anforderung ergeben und wurden ergänzt.

Pfad	Methode	Beschreibung	Body-Content
/addHashtag	POST	Fügt einen neuen Hashtag hinzu, wenn dieser noch nicht verwendet wird	Hashtag und private Key
/createPrivateKey	GET	Erstellt einen neuen private Key und sendet diesen zurück	-
/getResultsFromHashtag	POST	Liefert eine Liste mit allen Ergebnissen zu einem Hashtag	Hashtag
/hashtags	GET	Gibt eine Liste aller bereits verwendeten Hashtags zurück	-
/keepalive	POST	Simuliert den "Heartbeat" des Anwenders für den Server	Hashtag und private Key
/openSession	POST	Öffnet die Session für die Teilnehmer	Hashtag und private Key
/removeLocalData	POST	Entfernt alle Session-Daten des Hashtags vom Server	Hashtag und private Key
/startNextQuestion	POST	Startet die nächste Frage	Hashtag und private Key
/updateQuestionGroup	POST	Erstellt oder überschreibt die Session-Daten zu einem Hashtag	Hashtag und private Key

## 5 Implementierung

Nach der Analyse und der erfolgreichen Konzeption einer Lösung zur Integration der beiden ARSnova-Systeme in PowerPoint kann die eigentliche Implementierung beginnen. Hierbei wird kein Wert auf die grafische Oberfläche gelegt, da diese bereits in der Sektion Analyse ersichtlich wird und lediglich “nachgebaut” werden muss. Es sollen die Besonderheiten, Herausforderungen und umgesetzten Entwurfsmuster der Implementierung herausgestellt werden, dazu werden einige der komplexeren Prozesse geschildert, um einen tieferen Einblick in die programmatische Realisierung der Konzeption zu ermöglichen (ARSnova-Team der TH Mittelhessen, 2017b).

### 5.1 Anbindung an PowerPoint

Im Bereich der Implementierung der Integrationslösung ist die genaue Anbindung an PowerPoint das erste Thema. Für diesen Zweck wird eine Klasse benötigt, welche “Microsoft.Office.Tools.AddInBase” abstrahiert. Durch die Vererbung wird vertraglich sichergestellt, dass alle benötigten Eigenschaften wie beispielsweise das “Application”-Property zur Verfügung stehen, um die Integration zu gewährleisten. Gleichzeitig wird auch dafür gesorgt, dass die zwingend erforderlichen Methoden wie “Initialize” implementiert werden. Die Implementierung der Abstraktion wird größtenteils durch das PowerPoint Add-in-Template zur Verfügung gestellt und ist in eine eigene Quelldatei gekapselt. Die zweite, partielle und damit namensgleiche Datei enthält bereits zwei Methoden, die klassischerweise für ein Office Add-in benötigt werden: “Startup” und “Shutdown”. Des Weiteren ist auch die Methode “InternalStartup” schon definiert, welche automatisch von PowerPoint aufgerufen wird und die Event-Bindung zwischen dem Starten und Schließen von PowerPoint und der Integrationslösung übernimmt und das Add-in somit wirklich nativ wirken lässt, da es nicht extra gestartet werden muss:

---

```
public partial class ThisAddIn
{
    private void ThisAddIn_Startup(object sender,
        System.EventArgs e) {}
    private void ThisAddIn_Shutdown(object sender,
        System.EventArgs e) {}

    private void InternalStartup()
    {
        this.Startup += new
            System.EventHandler(ThisAddIn_Startup);
        this.Shutdown += new
            System.EventHandler(ThisAddIn_Shutdown);
    }
}
```

---

Listing 4: Ein unbearbeitetes PowerPoint Add-in Template (ohne Kommentare und Regions)

Für eine bessere Übersicht zwischen dem Standard-Verhalten eines Add-ins und der eigenen Implementierung empfiehlt es sich, nur in der im Beispiel 4 gezeigten, partiellen Datei zu arbeiten.

In der “ThisAddIn.cs”-Datei der ARSnova-Integrationslösung wird die Startup-Methode dazu verwendet, die benötigte PowerPoint-Anbindung herzustellen. Schon bei dem Starten der Anwendung soll diese Verbindung erzeugt werden, um direkt bei allen benötigten Events benachrichtigt zu werden. Dabei ist die Reihenfolge der Events ausschlaggebend, da diese die Priorität der Ausführung definiert. Die äußeren Elemente der Oberfläche sollen immer bevorzugt werden. Wenn das PowerPoint-Fenster geschlossen werden soll, ist der Befehl wichtiger als die Anweisung, die nächste Folie anzuzeigen, da letzterer Befehl nach dem Beenden der Anwendung nicht mehr zum Tragen kommt. Ersteres hat sozusagen die entscheidende Macht über die inneren Elemente. Nach dem klassischen Office Aufbau ergibt sich daraus folgende Reihenfolge:

- Hohe Priorität: Fenster Aktivitäten (Neue Präsentation erstellen, Öffnen oder Schließen einer Präsentation)
- Mittlere Priorität: Innere Fenster Aktivitäten (Vortrag starten / beenden, nächste / vorherige Folie anzeigen)
- Niedrige Priorität: Folien-Interaktionen (Anwählen einer anderen Folie im Bearbeitungsmodus)

Hier werden die wichtigen Bindings wie das Öffnen einer Präsentation, das Anzeigen der nächsten Folie oder das Selektieren einer ARSnova-Folie im Bearbeitungsmodus abgefangen und die entsprechenden Methoden im RibbonHelper ausgeführt, um die jeweilige Reaktion der Integrationsanwendung zu triggern.

Neben dem beschriebenen Teil der VSTO-Schnittstelle, der Anbindung an PowerPoint-Events, existiert noch die Ergänzung um ein neues Menüband sowie die Erweiterung der bestehenden Kontext-Menüs. Das wird bei den VSTO durch die Methode “CreateRibbonExtensibilityObject” in der Klasse “ThisAddIn.cs” erreicht, welche ebenfalls automatisch von PowerPoint während dem Laden der Oberflächen ausgeführt wird und ein Objekt mit der Implementierung des Interfaces “IRibbonExtensibility” erwartet, um welches die Office-Oberfläche erweitert wird. Zugehörig ist immer ein gleichnamiges XML-Dokument, welches die neuen Elemente für das Graphical User Interface (GUI) beinhaltet. In der Instanz selbst sind die Hilfsmethoden für die Oberfläche enthalten. Die Methoden stellen klassischerweise zur Verfügung:

Control-Eigenschaft	Rückgabewert	Zweck
getLabel	string	Gibt die Beschriftung für ein Element zurück.
getSuperTip onAction	string void	Liefert den Tooltip für ein Element. Legt fest, was bei der Aktivierung des Steuerelementes geschieht.
getImage	Bitmap	Gibt ein Bild für das Element zurück.
getEnabled	Integer	Entscheidet, ob ein Element als aktiv oder inaktiv (ausgegraut und nicht selektierbar) dargestellt wird.

Alle Methoden bekommen nach Vertragsdefinition ein Objekt der Klasse “Control” übergeben, der allgemeinen Definition eines Oberflächenelementes in Office. Damit ist eine Manipulation oder Weiterarbeit mit Inhalten des Elements möglich.

Die einzelnen Control-Elemente in der XML-Datei werden anhand von Identifikatoren (ID) eindeutig gehalten. Existiert eine ID bereits, handelt es sich für PowerPoint um die Erweiterung eines bestehenden Elementes. Ist ein Identifikator noch nicht bekannt, wird das GUI-Element neu angelegt. Je nach dem genauen Typ eines Control-Elementes unterscheiden sich die Strukturen leicht. Eine beispielhafte Struktur eines neuen Menübandes ist folgend dargestellt:

---

```
<ribbon>
  <tabs>
    <tab idMso="TabAddIns" label="ARSnova">
      <group id="sessionTypeGroup"
        getLabel="GetSessionTypeGroupLabel"
        visible="true">
        <button id="AddNewSlidesButton"
          getLabel="GetAddNewSlidesLabel"
          keytip="CA"
          getSupertip="GetAddNewSlidesSupertip"
          onAction="AddQuizToNewSlideButtonClick"
          getImage="GetAddNewSlidesButtonImage"
          size="large" />
        <separator id="quizGroupSeparator"/>
        <button id="EditButton"
          getLabel="GetEditButtonLabel"
          keytip="CA"
          getSupertip="GetEditButtonSupertip"
          getEnabled="IsOneArsnovaSlideSelected"
          onAction="EditButtonClick"
          getImage="GetEditButtonImage"
          size="large" />
      </group>
    </tab>
  </tabs>
</ribbon>
```

---

Listing 5: Struktur eines neuen Menübandes

Die Erweiterung des Kontext-Menüs einer Folie hingegen ist anders strukturiert:

---

```

<contextMenus>
  <contextMenu idMso="ContextMenuThumbnail">
    <menuSeparator id="separatorThumbnail"/>
    <menu id="menuThumbnail"
      getLabel="GetArsnovaSlideContextMenuLabel"
      getImage="GetArsnovaFavIcon">
      <button id="addQuizContentToNewSlide"
        getLabel="GetAddNewSlidesLabel"
        getImage="GetAddNewSlidesButtonImage"
        getSupertip="GetAddNewSlidesSupertip"
        onAction="AddQuizToNewSlideButtonClick"/>
      </menu>
    </contextMenu>
  </contextMenus>

```

---

Listing 6: Struktur eines neuen Kontextmenü-Eintrages

Die zugehörigen und benötigten Methoden der Control-Eigenschaften müssen wie oben erläutert in dem gleichnamigen “IRibbonExtensibility”-Objekt im selbigen Ordner definiert sein. Alle XML-Beschreibungen werden von dem Tag “customUI” umschlossen, welches einer Methoden-Angabe ähnlich der “get”-Methode der einzelnen Oberflächenelemente enthält, die bei dem Laden der GUI-Erweiterung ausgeführt wird.

## 5.2 WPF-Erweiterung

Neben der Erweiterung der PowerPoint-Oberflächen gibt es noch eigenständige WPF-Elemente, in welche die Interaktionslogik zur Verwaltung der Fragen gekapselt ist. Die einzelnen Fenster werden durch das “IRibbonExtensibility”-Objekt getriggert, beispielsweise nachdem der Button “Fragen verwalten” in der ARSnova-Ribbon-Bar ausgewählt wurde. Da wie in der Analyse beschrieben das MVVM-Pattern zum Einsatz kommt, muss es eine Klasse zum Verwalten und Koppeln der einzelnen Views, Models und ViewModels geben. Diese Aufgabe übernimmt der im nächsten Abschnitt erläuterte *ViewPresenter*.

Neben den standardmäßigen WPF-Techniken wie den Wertkonvertern, die beliebige, gebundene Attribute in einen Visibility-Wert übersetzen, um die Sichtbarkeit der Oberfläche mit beispielsweise Wahrheitswerten aus dem ViewModel steuern zu können, gab es lediglich eine Erweiterung der WPF-Controls. Ein “Wasserzeichen-Service”, um leere Textboxen oder Drop-down-Listen mit Platzhalter-Texten zu versehen:

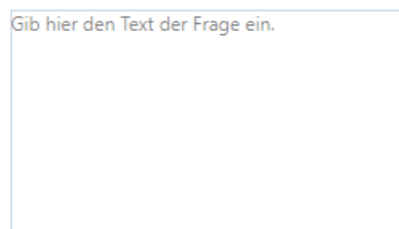


Abbildung 31: Beispiel: Ein Platzhalter in einer Text-Box

Der Watermark-Service wird auf Seite 73 erläutert.

### 5.2.1 ViewPresenter

Prinzipiell wird das MVVM-Pattern ohne einen Presenter umgesetzt - dies würde dem Model-View-Presenter (MVP)-Pattern näher kommen, bei dem der Presenter die Logik enthält. In diesem Projekt ist die Funktionslogik allerdings in dem ViewModel abgelegt. Der ViewPresenter trägt lediglich die Verantwortung für das Paaren von Views und ViewModels, deren Darstellung und der Entscheidung, welches Fenster nach der Schließung eines anderen angezeigt wird. Ebenfalls wird geregelt, ob ein neues Fenster für ein ViewModel geöffnet wird oder lediglich der Inhalt eines bereits geöffneten ausgetauscht werden soll. Gerade in letzterem Fall ist das Wissen nötig, welche Fenster zur Zeit geöffnet sind. Da ein ViewModel bei einer Schließung nicht weiß, welches Fenster vorher geöffnet wurde und nach der Schließung in der Vordergrund rücken soll, muss diese Aufgabe von einem externen Service übernommen werden. Ein ViewModel sollte niemals die Kontrolle über andere Inhalte einer Anwendung besitzen. Ein solches Verhalten würde einer Gott-Klasse nahekommen und widerspricht den in der Analyse definierten Design-Richtlinien, dass jede Klasse nur eine Aufgabe erfüllen soll. Die exakte Funktionalität der ViewPresenters lässt sich gut von dem Interface des Services ablesen:

---

```
public interface IViewPresenter
{
    void Add<TViewModel, TView>();

    void ShowInNewWindow<TViewModel>(Action<TViewModel>
        viewModelAction = null) where TViewModel : class;

    void Show<TViewModel>(Action<TViewModel> viewModelAction =
        null) where TViewModel : class;

    void CloseWithoutWarningPrompt();

    void CloseWithWarningPrompt();

    void Close(Guid windowId, bool removeWindow = true);
}
```

---

Listing 7: Das Interface des ViewPresenters

Die “Add”-Methode wird lediglich zur Initialisierung des Services verwendet. Durch die bekannte Zugehörigkeit von ViewModel zu View muss später nur das ViewModel übergeben werden und der ViewPresenter ordnet automatisch die entsprechende View zu.

“Show” und “ShowInNewWindow” zeigen jeweils das übergebene ViewModel an. Hilfreich ist dabei der zweite Parameter der “ShowInNewWindow”-Methode, einem Delegate des jeweiligen ViewModel-Typs. Zwar hat der ViewPresenter die totale Gewalt über die ViewModels und ob er bei einem Aufruf ein neues Objekt erzeugt oder eine bereits bestehende Instanz des Typs verwendet, doch kann der Aufrufende Actions angeben, die später auf dem Objekt ausgeführt werden sollen. Beispielsweise möchte der Aufrufer bei der Terminierung der neuen Ansicht eine bestimmte Methode ausführen, um das Beenden “mitzubekommen”. Gleichzeitig soll der Aufrufer allerdings nicht das neue ViewModel kennen, denn damit wären die Entwurfsmuster zur modularen



Programmierung durchbrochen. Auf diese Art und Weise werden die benötigten Kopp-  
lungen an den ViewPresenter übergeben und dieser stellt die benötigten Verbindungen  
her:

---

```
this.viewPresenter.ShowInNewWindow
    <SelectArsnovaTypeViewViewModel>(
    vm =>
    {
        vm.OnSelectArsnovaTypeViewClose += this.ShowManageSession;
    });
```

---

Listing 8: Funktionale Action-Definition

Der ViewPresenter aktiviert später die Actions, sofern welche vorhanden sind:

---

```
viewModelAction?.Invoke(viewModel);
```

---

Listing 9: Aktivieren von Actions für ein ViewModel

Die ersten beiden Close-Methoden sind für das Schließen von ViewModels über die  
entsprechenden Buttons in der Fußzeile gedacht. Da es neben dieser Funktion aller-  
dings noch den bekannten, roten Schließen-Button oben rechts in einem Windows-  
Fenster gibt, muss auch dieser behandelt werden. Zwar könnte der standardmäßige  
Schließ-Mechanismus eines Windows-Fensters weiter verwendet werden, doch würde  
der ViewPresenter diese Interaktion nicht mitbekommen und weiterhin denken, dass  
das Fenster noch existiert. Das kann bei späteren Darstellungsversuchen zu Fehlern wie  
einer “NullReferenceException” führen. Über die “*Close(Guid windowId, bool remove-  
Window = true)*”-Methode werden eben solche Schließungen des Fensters behandelt,  
um ein korrektes Weiterlaufen der Anwendung zu garantieren.

### 5.2.2 Watermark-Service

Standardisierte WPF-Textboxen enthalten keine Möglichkeiten, einen Platzhalter für  
leere Inhalte zu hinterlassen. Ein Platzhalter besteht in der Regel aus einer Kurzbe-  
schreibung des Zwecks eines leeren Oberflächenelementes, um den Nutzer mitzuteilen,  
welche Eintragung er vornehmen muss. Dieses Verhalten ist weit verbreitet und darf  
nicht fehlen. Aus diesem Grund beinhaltet die Integrationslösung eine Watermark-  
Implementierung für Textboxen. Um die Herangehensweise zu erläutern, muss zuerst  
ein grundsätzliches Verständnis für die XAML-Views in WPF geschaffen werden.  
Das WPF-Konzept ermöglicht die Verwendung von “Resources” in allen Controls, also  
allen Oberflächenelementen, denn ein Objekt kann erst als ein GUI-Element eingesetzt  
werden, wenn es die Control-Klasse abstrahiert. “Resources” können dabei jede mög-  
liche Form von Daten darstellen. Oft sind es Styles wie eine Farbgebung, die einem  
Window-Control als “Resource” angehängt wird und damit für alle Kind-Elemente  
verfügbar ist. Dadurch muss ein Farbton nicht mehrfach definiert werden und kann  
durch Anpassung des “Resource”-Wertes für alle verwendeten Elemente geändert wer-  
den. Das “Resources”-Konzept wird genutzt, um ein Property des Watermark-Services  
einzubinden, welches nur die Aufgabe hat, eine Methode auszulösen, wenn sich der  
Wert der äußeren Textbox ändert. Da in WPF die Sub-Elemente standardmäßig auf  
die Eigenschaften ihrer übergeordneten Elemente zugreifen können müssen, beispiels-  
weise um Daten zur Darstellung durchzureichen, kann neben dem Event zur Änderung

des Textbox-Inhaltes auch auf andere Eigenschaftswerte zugegriffen werden. Interessant sind dabei:

- `Loaded`: Wird getriggert, wenn das Oberflächenelement geladen wurde.
- `GotKeyboardFocus`: Wird ausgelöst, wenn das GUI-Element über die Tastatur ausgewählt wird.
- `LostKeyboardFocus`: Wird angestoßen, wenn der Focus der Tastatur auf dieses Element verloren geht.

Diese Events werden zur Steuerung des Platzhalters benötigt. Bei dem Laden des Elementes wird geprüft, ob bereits ein Text in der Textbox vorhanden ist. Wenn dem nicht so ist, wird ein entsprechender Platzhalter-Text eingefügt. Sobald der Tastatur-Fokus auf das Element gelegt wird, bearbeitet der Anwender das Feld. Dann wird der Platzhalter sofort entfernt. Bei dem Verlassen des Fokus wird erneut geprüft, ob sich ein Inhalt in der Textbox befindet. Ist dem nicht so, wird der Platzhalter erneut eingefügt. Neben dem Setzen des Textes erfolgt auch eine Anpassung des Styles der Textbox, da das Wasserzeichen in grau dargestellt werden soll. Damit wird dem Anwender suggeriert, dass es sich nicht um wirklichen Inhalt des Feldes handelt.

### 5.3 LINQ

Language Integrated Query (LINQ) wurde Ende 2007 mit dem .NET-Framework 3.5 releast und vereint mehrere Daten-APIs:

- `System.Collections`: Zugriff auf Objekte im Speicher
- `Structured Query Language (SQL)`: Zugang zu relationalen Datenbanken
- `XQuery`: Umgang mit XML-Dokumenten

Um auf diese drei Datenmodelle zuzugreifen, mussten die Entwickler unterschiedliche Programmierschnittstellen verwenden, die teilweise sogar mit verschiedenen Abfragesprachen (SQL und XQuery) bedient werden mussten. Um Abfragen Datenmodell-unabhängig erstellen zu können, wurde LINQ als einheitliche Schnittstelle für die unterschiedlichsten Datenquellen eingeführt:

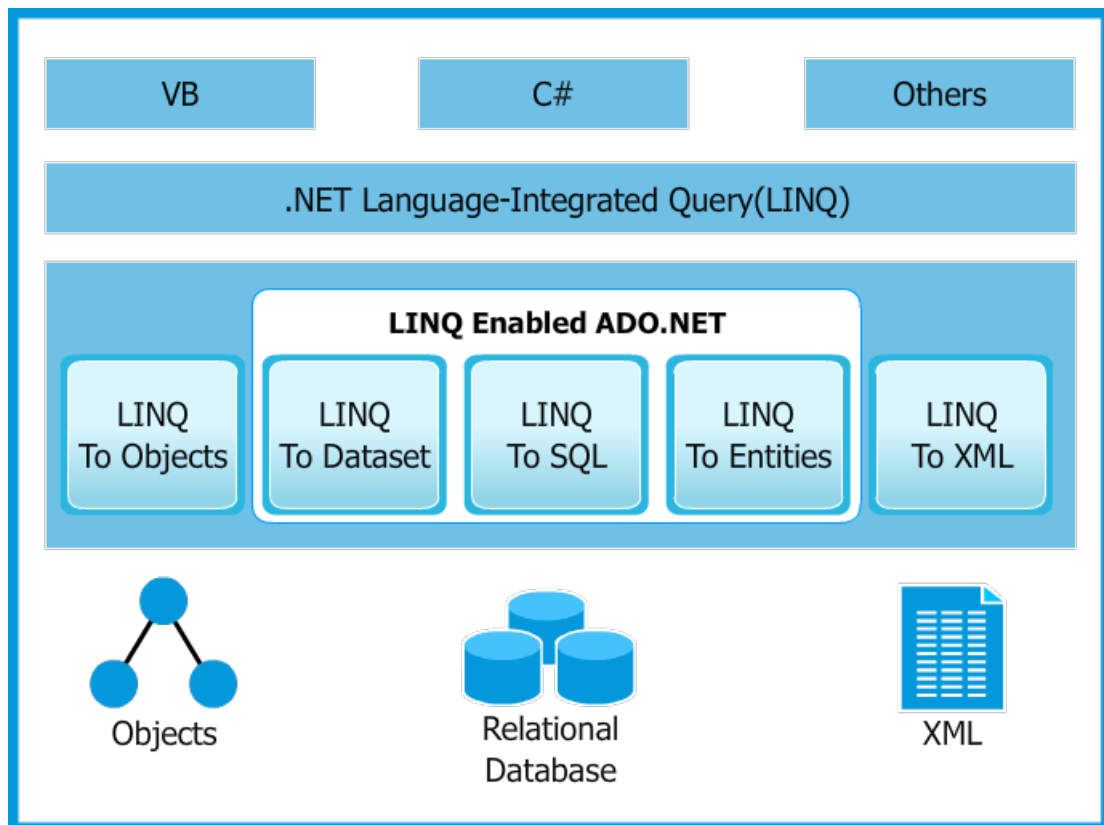


Abbildung 32: Die LINQ-Architektur

Quelle: <https://www.edureka.co/blog/unleash-the-power-of-linq-the-net-way/>, Abgerufen am 01.03.2017

Zusätzlich enthält das LINQ-Framework auch neue Schlüsselwörter und Sprachkonzepte, die eine SQL-ähnliche Abfragesyntax zulassen. Um den Entwicklern noch weiter entgegen zu kommen, wurde neben der Abfragesyntax die Erweiterungsmethodensyntax eingeführt. Bei dieser Art der Abfrage werden Methodenaufrufe verkettet. Unter Entwicklern ist eine solche Variante besser lesbar. Folgend ein Beispiel um einen direkten Vergleich der beiden Abfragetypen zu schaffen:

---

```
// Abfragesyntax
var correctUsers = from answer in this.GetAllResultsList()
    where answer.answerOption == correctAnswerOption
    select answer.User;

// Erweiterungsmethodensyntax
var correctUsers = this.GetAllResultsList()
    .Where(a => a.answerOption == correctAnswerOption)
    .Select(a => a.User);
```

---

Listing 10: LINQ-Syntax im Vergleich

Die Ergebnisse sind in beiden Fällen identisch, da der Compiler die beiden Abfragen gleich interpretiert. Wichtig ist zu wissen, dass LINQ mit einer verzögerten Ausführung arbeitet. Zunächst werden alle Abfragenverkettungen gesammelt, um eine eventuell mögliche Optimierung des Datenzugriffes durchzuführen. Erst wenn die Ergebnisse wirklich benötigt werden, wird die Abfrage ausgeführt, davor arbeiten die Erweiterungsmethoden auf dem Datentyp "Enumerable" - der Repräsentation einer Sequenz.

Durch die Methoden *toList()* oder *toArray()* wird LINQ zu einer Ausführung gezwungen und gibt statt einem Enumerable-Objekt eine Liste oder ein Array zurück. Folglich sollten diese Anweisungen erst zum Ende einer Abfragenverkettung ausgeführt werden, um dem Framework die Möglichkeit zur Optimierung der Abfrage zu geben (Microsoft, 2017c).

Innerhalb der Integrationsanwendung wird weder mit XML-Dokumenten, noch mit anderweitigen Datenbank-Zugriffen gearbeitet. Dementsprechend wird LINQ lediglich für Listen-Operationen eingesetzt, um performante und gut lesbare Abfragen zu garantieren. In anderen Programmiersprachen würden Schleifen und Abfragen kombiniert werden, um eine Liste auf die gleiche Art und Weise zu filtern. Das nachfolgende Beispiel zeigt klar die gewonnene Lesbarkeit einer LINQ-Abfrage, gleichzeitig wird der Speicher durch weniger Datenstrukturen optimaler ausgenutzt:

---

```
// herkömmliches Listen filtern
var correctUsers = new List<User>();
for(var answer in this.GetAllResultsList())
{
    if (answer.answerOption == correctAnswerOption)
    {
        correctUsers.Add(answer.User);
    }
}

// Listenfilterung mit LINQ
var correctUsers = this.GetAllResultsList()
    .Where(a => a.answerOption == correctAnswerOption)
    .Select(a => a.User);
```

---

Listing 11: LINQ im Vergleich zu herkömmlichen Listenoperationen

## 5.4 Dependency Injection

Wie bereits im gleichnamigen Abschnitt auf Seite 56 der Konzeption festgelegt, wird innerhalb der Integrationslösung Dependency Injection eingesetzt. Folglich ist es nicht möglich, in den über den Unity-Container verwalteten Instanzen individuelle Eingabeparameter über den Konstruktor zu definieren, da diese zu dem Zeitpunkt der Erzeugung und Constructor Injection nicht bekannt sind. Die Objekte sollen alle als Service interagieren, dass heißt, sie stehen für viele Teile der Anwendung gleichzeitig bereit und übernehmen gekapselte Aufgaben der Informationsverarbeitung oder Datenaufbereitung. Daher sollen alle benötigten Daten über den Service-Aufruf verfügbar sein und alle benötigten Ergebnisse werden sofort zurückgegeben. Diese Struktur soll auch in zukünftigen Versionen beibehalten werden. Die aktuellen mit Dependency Injection instantiierten Services mit einer Kurzbeschreibung der jeweiligen Aufgabengebiete sind:

- ArsnovaClickService: Kapselt die Verbindung zu dem arsnova.click-Server.
- ArsnovaVotingService: Beinhaltet die Kommunikation zu dem arsnova.voting-Backend.

- LocalizationService: Zuständig für Übersetzungen.
- QuestionTypeTranslator: Übersetzt das Enum “QuestionType” in einen lokalisierten Text zur Anzeige in dem GUI.
- SessionInformationProvider: Stellt alle möglichen Auskünfte für die Verwaltung von ARSnova-Sessions zur Verfügung.
- SessionManager: Beinhaltet die Geschäftsprozesse
- SlideManipulator: Einfügen und Manipulieren von Inhalten auf PowerPoint-Folien.

Die zugehörigen Interfaces tragen die gleiche Bezeichnung der Implementierung um das Präfix 'I' ergänzt, dem Anfangsbuchstaben von “Interface”.

## 5.5 Lokalisierung

Internationalisierung ist in der Integrations-Applikation durch den Einsatz des “LocalizationService” umgesetzt. Dieser verfügt lediglich über eine Translate-Methode, die eine Zeichenkette erwartet und zurückgibt. Der Übergabeparameter wird auf Zeichenfolgen durchsucht, die von XML nicht unterstützt werden, wie beispielsweise das '&'-Symbol, und ersetzen diese durch alternative Darstellungen. Aus Gründen der Lesbarkeit und XML-Kodierung werden alle Leerzeichen durch einen Unterstrich ersetzt. Die aus der Ersetzung resultierende Zeichenkette wird verwendet, um in der aktuellen Sprachdatei den zugehörigen Wert zu ermitteln: Die übersetzte Zeichenfolge. Der Vorteil bei dieser Vorgehensweise im Gegensatz zu der Arbeit mit Labels besteht darin, dass die Labeltexte nicht hinzugefügt werden müssen. Auch gibt der Lokalisierungs-Service Zeichenfolgen, zu denen er keine zugehörige Übersetzung findet, direkt zurück. Aus der Gegebenheit heraus, dass die Texte im Quellcode in englischer Sprache eingepflegt werden, ist daher theoretisch keine einzelne englische Übersetzungsdatei nötig. Trotzdem sind die Eintragungen zu empfehlen, da später der Wunsch nach einer Anpassung der englischen Texte entstehen könnte, nicht aber der anderen Sprachen. Wenn kein Übersetzungseintrag für den Text existiert, wird die Stelle im Code, also als Übergabeparameter der Translate-Methode angepasst, abgeändert. Da sich damit der Wert ändert, mit dem der Übersetzer nach der zugehörigen Zeichenkette sucht, müssen alle Übersetzungs-Zuordnungen der anderen Sprachen angepasst werden. Damit ist in Hinblick auf die Wartbarkeit ein Übersetzungs-Dokument für die englische Sprache die langfristige Lösung.

Für den Fall, dass bei einem neuen Text vergessen wurde, eine Übersetzung zu erstellen, werden in allen Sprachen immer die englischen Texte aus dem Code angezeigt. Zwar ist dies nicht immer die gewünschte Sprache, doch im Vergleich zu der Arbeit mit Labels werden hier keine Fehlermeldungen erzeugt oder der Label-Platzhalter in der grafischen Benutzeroberfläche angezeigt.

Sollen nur kleine Änderungen an bestehenden Texten erfolgen, können die Übersetzungen in den jeweiligen Lokalisierungs-Dateien, die in dem Projekt “ARSnovaPPIntegration.Common.Resources” vorliegen, angepasst werden. Eine Übersetzung kann von mehreren Stellen aus verwendet werden, daher empfiehlt es sich, bei größeren textuellen Änderungen und Erweiterungen den neuen Text in englischer Sprache im Quellcode einzutragen und alle entsprechenden Übersetzungs-Zuordnungen anzulegen.

Die Sprache, in der die Anwendung dargestellt wird, wird bei dem Starten des Menübandes initialisiert und von den aktuellen Office-Einstellungen übernommen. Sollte

die aktuelle PowerPoint-Sprache nicht in den Übersetzungen verfügbar sein, wird immer die englische Version verwendet.

---

```
var app = this.GetHostItem<Application>(typeof(Application),  
    "Application");  
var languageId = app.LanguageSettings.LanguageID  
    [MsoAppLanguageID.msoLanguageIDUI];  
System.Threading.Thread.CurrentThread.CurrentCulture = new  
    CultureInfo(languageId);
```

---

Listing 12: Festlegen der Spracheinstellungen anhand der Office-Lokalisierung

In dem Lokalisierungs-Service verwaltet ein Objekt der Klasse “ResourceManager” die “Resource”-Dokumente mit den Übersetzungen. Ein “ResourceManager” ist in der Lage, die benötigte Datei anhand einer Sprachkonfiguration und einem zugehörigen Sprach-Suffix in dem Datei-Namen aus einer Liste von “Ressourcen” zu extrahieren. Die im obigen Code-Beispiel gesetzte Sprachkonfiguration für den aktuellen Thread kann dafür einfach wieder ausgelesen werden:

---

```
var translatedString =  
    this.resourceManagedTranslations.GetString(escapedString,  
    Thread.CurrentThread.CurrentCulture);
```

---

Listing 13: Automatische “Resource”-Auswahl anhand des CurrentCulture-Properties durch einen “ResourceManager”

## 5.6 Object-Mapper

Manche Models, die von den Schnittstellen von `arsnova.click` und `arsnova.voting` zurückgegeben werden, enthalten viel mehr Information als wirklich benötigt. Auf `arsnova.click` trifft das zu, weil der dortige Typ “QuestionGroup” die komplette Session mit allen benötigten Informationen enthält und die Datenstruktur nicht für eine schlanke Kommunikation über eine REST-API entworfen wurde. Auf `arsnova.voting` trifft es zu, weil die Models über Jahre hinweg von vielen Studenten verwendet wurden, die oft Eigenschaften doppelt definiert oder Wertgruppen für neue Features, beispielsweise für die Formatierung von Bildern, nicht in neue Objekte ausgelagert haben, sondern viele einzelne Werte in eine Fragen-Klasse eingefügt haben.

Da die Arbeit mit solchen Arten von Rückgabewerten umständlich ist, werden diese vor der Rückgabe zu den aufrufenden Komponenten des Kommunikations-Services erst in ein neues, schlankeres und für seine Anwendung passenderes Model übertragen. Aufgrund der vielen verschiedenen Models, die zum Einsatz kommen, wäre es ein großer Aufwand, für jedes dieser Models eine eigene Methode zum Kopieren und Zurückkopieren zu entwerfen, kommt ein ObjectMapper zum Einsatz. Dieser überträgt alle Werte von einem Model in ein anderes, solange die Bezeichnungen und Typen der Eigenschaften identisch sind. Dadurch kann, solange die Typen und die Bezeichnungen übernommen werden, ein großer Implementierungsaufwand verhindert werden:

---

```
public class ObjectMapper<TSourceType, TTargetType>
{
    private PropertyInfo[] sourceProperties;
    private PropertyInfo[] targetProperties;

    public void Map(TSourceType source, TTargetType target)
    {
        var tSource = typeof(TSourceType);
        var tTarget = typeof(TTargetType);

        this.sourceProperties = tSource.GetProperties();
        this.targetProperties = tTarget.GetProperties();

        this.SyncProperties(source, target);
    }

    private void SyncProperties(TSourceType objSource,
        TTargetType objTarget)
    {
        if (this.sourceProperties != null &&
            this.sourceProperties.Any())
        {
            foreach (var sourceProperty in this.sourceProperties)
            {
                var targetProperty =
                    this.targetProperties.FirstOrDefault(x => x.Name ==
                        sourceProperty.Name);

                if (targetProperty != null)
                {
                    var val = sourceProperty.GetValue(objSource, null);
                    targetProperty.SetValue(objTarget,
                        Convert.ChangeType(val,
                            sourceProperty.PropertyType), null);
                }
            }
        }
    }
}
```

---

Listing 14: Object Mapper

Der ObjectMapper ist an seine beiden Typen, die bei der Erzeugung angegeben werden müssen, gebunden. Dadurch kommt die Hilfs-Klasse nicht als Service in Frage. Ein zur Verfügung Stellen von vielen ObjectMappern, um jede mögliche Typ-Paarung abzudecken, ist eine Verschwendung von Ressourcen, da nur selten alle Services der Art bei einer Ausführung verwendet werden. Sollte es sich um eine arsnova.click-Session handeln, werden die Mapper für die arsnova.voting-Models nicht zum Einsatz kommen und umgekehrt. Auch kann diese Klasse nicht von Dependency Injection profitieren, da es weder bei der Erzeugung und noch im späteren Anwendungsverlauf Abhängigkeiten besitzt, die geladen werden müssen. Daher wird ein ObjectMapper immer zu dem Zeitpunkt instantiiert, indem er benötigt wird.

## 5.7 Build und Setup Erstellung

Bei dem Hinzufügen eines neuen Projektes zu der Mappe müssen die Projekt-Einstellungen angepasst werden, um später eine brauchbare Assembly für die Installationsroutine zu generieren. Ohne diese Konfiguration erschwert sich die spätere Setup-Erstellung. Zuerst müssen der Name der Assembly sowie das Zielframework überprüft werden. Um Verwirrungen zu vermeiden, sollte der Name der Assembly immer der des Namespaces des zugehörigen Projektes sein. Bei dem Zielframework ist es wichtig, dass alle Projekte der Mappe dieselbe Version angeben. Zwar sind verschiedene .NET-Versionen untereinander kompatibel, doch benötigt der Anwender später alle angegebenen Framework-Versionen, um die gesamte Anwendung zu installieren. Dies gilt es zu vermeiden, die zur Zeit von der Integrationslösung verwendete .NET-Framework-Version ist die 4.6.1, welche die aktuellste zum Beginn der Entwicklung ist. Weitere, wichtige Konfigurationsparameter sind:

Parameter	Wert	Erläuterung
Ausgabetyyp	Klassenbibliothek	Stellt sicher, dass eine Assembly-Datei erzeugt wird.
Ausgabepfad	bin\Release\	Alle Ausgaben werden in diesen Ordner umgelenkt, damit alle für das Setup benötigten Dateien in einem Ordner gesammelt sind.
Code Analyse beim Erstellen aktivieren	true	Beim Erstellen des Projektes wird eine Code-Analyse durchgeführt. Der Vorgang kostet Zeit, ersetzt aber zumindest teilweise die fehlende Code-Qualitätsüberprüfung durch einen Build-Server.

Für die Zukunft ist zu überlegen, eine Signierung zu erwerben. Zurzeit werden die Assemblies mit einem lokalen Schlüssel signiert, doch trotzdem wird die Anwendung als "nicht vertrauenswürdig" eingestuft. Vertrauenswürdig wird eine Applikation für Windows erst, wenn es sich bei der Signierung um ein beglaubigtes Zertifikat von einer "Certification Authority"-Instanz handelt. Für einen Prototypen ist das nicht notwendig.

Abschließend werden die Daten innerhalb des folgenden Listings in der "AssemblyInfo" eines jeden Projektes eingetragen, um jeder Assembly einzeln aussagekräftige Metadaten zur Verfügung zu stellen:

---

```
[assembly:
    AssemblyTitle("ARSnovaPPIntegration.Common.Resources")] //
    Beispiel, die Zeichenkette muss dem Assembly-Namen
    beziehungsweise dem Projekt-Namespace entsprechen
[assembly: AssemblyCompany("Technische Hochschule
    Mittelhessen")]
[assembly: AssemblyProduct("ARSnova PowerPoint Integration")]
[assembly: AssemblyCopyright("Copyright \textcopyright
    ARSnova-Team der TH Mittelhessen (2017)")]
```

---

Listing 15: AssemblyInfo eines ARSnovaPPIntegration-Projektes



## 5.8 Grenzen der VSTO

Mit den VSTO werden einige Möglichkeiten zur Integration einer Anwendung in PowerPoint geboten. Doch auch dem sind Grenzen gesetzt. So konnten einige anfängliche Ideen nicht umgesetzt werden, da die nötigen Anbindungen zu der Office-Applikation nicht durch die VSTO gegeben sind. Als Folge musste teilweise das Konzept angepasst werden und mehr Verantwortung in die Hände des Anwenders gelegt werden. Die drei größten Anpassungen beziehungsweise nicht umsetzbare Features aufgrund der Grenzen der VSTO werden in diesem Abschnitt dargelegt.

### 5.8.1 Start Button

Während bei einer arsnova.voting-Session die Frage freigegeben wird, sobald der Vortragende die Folie mit den Fragen und Antworten aufruft, muss bei einem von arsnova.click unterstützten Vortrag erst die Vorstellung der Frage selbst mit all seinen möglichen Antwortoptionen durchgeführt werden. Da die Runden countdown-basiert sind und fairer Wettbewerb nur gegeben ist, wenn alle Teilnehmer vor dem Beginn des Countdowns die Frage wirklich verstanden haben, ist dieses Verhalten unabdingbar. Wäre dem nicht so, würde mit bewertet werden, wie schnell ein Teilnehmer die Frage erfasst - das soll unterbunden werden. Dies ist zur Zeit so gelöst, dass der Vortragende eine zweite Folie mit der gleichen Frage und den entsprechenden Antworten hat. Mit der ersten werden die Inhalte vorgestellt, bei dem Anzeigen der zweiten beginnt die Umfrage selbst und damit der Countdown. Zu Beginn sollte es nur eine Folie geben, auf der sich ein "Start"-Button befindet.

Ein solcher Button muss von PowerPoint zur Verfügung gestellt werden, da nur Inhalte auf einer PowerPoint-Folie eingefügt werden können, die die Office-Anwendung selbst unterstützt. In diesem Bereich gibt es den "ActionButton", einen Knopf, der bei dem Auswählen während einer Präsentation eine von vier Aktionen auslösen kann:

- Hyperlink öffnen
- Programm ausführen
- Makro ausführen
- Objektaktion (Animation)
- Sound wiedergeben

Keine dieser Interaktionen ist hilfreich. Zur Umsetzung des Features müsste das ARSnova Add-in auf ein Event-Property des Buttons binden können, damit eine Methode zur Freischaltung der Quiz-Frage ausgeführt werden kann, wenn der Anwender den Knopf betätigt. Allerdings kann der ActionButton nur die oben gelisteten Aktionen ausführen, das gewünschte Verhalten entspricht mehr einem WPF-Control-Button, dafür ist die PowerPoint-Folien-Variante allerdings nicht konzipiert.

### 5.8.2 Beeinträchtigte Folien-Navigation

Während einer Präsentation wird das Freigeben und Schließen von Fragen in arsnova.voting und arsnova.click daran festgemacht, auf welcher Folie sich der Vortragende momentan befindet. Es werden zwei Prozesse ineinander integriert: Das Durchführen einer Umfrage sowie das Präsentieren von Folieninhalten. Während der Präsentations-Prozess auch das Vor- und Zurückwechseln von Folien beinhaltet, ist der Ablauf

einer Umfrage unidirektional. Nach dem Freigeben einer Frage kommt das Schließen, abschließend das Darstellen der Ergebnisse und nicht etwa eine erneute Freigabe. Durch das in PowerPoint erlaubte Zurücknavigieren von der Folie mit der Darstellung der Ergebnisse auf die vorherige Folie zum Starten der Frage wird diese erneut gestartet - obwohl dies in dem Umfragen-Prozess nicht eingeplant ist.

Wie bereits in der Analyse erörtert, können in PowerPoint bestehende Prozesse erweitert beziehungsweise neue Abläufe hinzugefügt werden, nicht allerdings bestehende verändert oder entfernt werden. Um den Umfragen-Prozess korrekt abbilden zu können, müsste ein Zurück-navigieren auf den Ergebnis- und Countdown-Folien unterbunden werden. Das wird allerdings nicht von PowerPoint genehmigt. An dieser Stelle muss sich der Anwender darüber im Klaren sein, dass ein erneutes Aufrufen der Folie zum Starten der Umfrage diese wirklich ein erneutes Mal startet.

### 5.8.3 Sperren von Folien

Als erste Idee kam der Wunsch auf, dass die Integrationslösung alle Folien mit ARSnova-bezogenen Inhalten voll verwaltet und das Bearbeiten durch einen Anwender nicht zulässt. Wenn dieser beispielsweise den Hashtag auf einer Folie ändert, würde die Nutzer durch Eingabe der falschen Session-Bezeichnung in keine oder die falsche Session gelangen. Dies wäre durch das Ausschließen des Bearbeitens durch den Nutzer verhindert. Allerdings stellt PowerPoint eine solche Funktion nicht zur Verfügung. Dem Anwender gehören die Folien und diese sind voll bearbeitbar - so die Philosophie der Anwendung. Das Verhalten könnte nur durch eine Art "Hack" erzwungen werden. Durch die PowerPoint-Bindings kann überprüft werden, welche Folie vom Anwender selektiert wird. Sollte es sich dabei um eine ARSnova-Folie handeln, kann einfach wieder die nächste vorherige Folie, welche keine ARSnova-Inhalte besitzt, angezeigt werden. Zusätzlich kann der Nutzer über ein Pop-up darüber informiert werden, dass ein Bearbeiten nicht möglich ist. Neben der Tatsache, dass es sehr unschön ist, eine Folie nicht wirklich zu sperren, sondern einfach eine andere anzuzeigen, kann der Fall eintreten, dass der Anwender mit dem Design nicht zufrieden ist. Eventuell soll die Schrift größer oder dicker sein, oder er möchte bestimmte Inhalte der Frage unterstreichen beziehungsweise farbig markieren. Dies wäre über PowerPoint nicht mehr möglich, da die Folie nicht angezeigt wird. Folglich müssen alle möglichen Design-Änderungen über eine zusätzliche Oberfläche zur Verfügung gestellt werden. Abgesehen von dem erheblichen Mehraufwand in der Implementierung für eine in PowerPoint bereits bestehende Funktionalität, ist es für den Anwender sehr umständlich, zwei Menüs zum Anpassen des Designs zu kennen und je nach Folie ein unterschiedliches zu verwenden. Das ist nicht das Verhalten, welches bei einer vollen Integration erwartet wird.

Da die Nachteile den Vorteilen bei einem solchen "Hack" zur nicht-Bearbeitung von ARSnova-Folien klar überwiegen, bleiben alle Folien weiterhin in der Hand des Anwenders. Dieser kann auch die automatisch generierten Folien frei gestalten und formatieren, muss sich allerdings bewusst machen, dass es bei inhaltlichen Anpassungen zu Problemen in der Ausführung kommen kann. Diese Entscheidung kann als Vergleich von Freiheit des Anwenders und Usability zu der Sicherheit einer Anwendung gesehen werden. Zu viel Sicherheit ist nie von Vorteil, wenn die Freiheit dadurch leidet.

## 5.9 arsnova.click

Da die beiden ARSnova-Server nur Unicode Transformation Format (UTF)-8 konforme Zeichenketten akzeptieren und es ansonsten zu Darstellungsfehlern aufgrund der

unterschiedlichen Codierung kommt, werden alle Body-Inhalte der REST-Anfragen vor dem Versenden des Requests von der Integrationslösung UTF-8 konform entschlüsselt. Dies gilt für Anfragen an `arsnova.click` als auch an `arsnova.voting`. Folglich müssen die Body-Parameter nach dem Empfang von der `arsnova.click`-REST-Schnittstelle auch wieder mit der `decodeURIComponent`-Methode decodiert werden.

Wie bereits in der Analyse erläutert, werden zwei verschiedene Arten der Schnittstellen-Definition verwendet. Die Verwendung des Simple-Rest-Paketes gestaltet sich problemlos, sogar die benötigten Übergabeparameter der zugehörigen Meteor-Methode werden einzeln nach dem Namen aus dem Request-Body gemappt. Leider kann diese Funktion nicht genutzt werden, da das Simple-Rest keine Decodierung vorsieht, diese allerdings wie im vorherigen Abschnitt beschrieben benötigt wird. Daher muss eine Funktion zum Parameter-Mapping an die bestehende Meteor-Methode angehängt werden, welches nur einen geringen Mehraufwand darstellt:

---

```
Meteor.publish('QuestionGroupCollection.join', function
  (hashtag) {
    new SimpleSchema({
      hashtag: {type: String}
    }).validate({hashtag});
    return QuestionGroupCollection.find({hashtag: hashtag});
  }, {
    url: "api/getQuestionGroup",
    httpMethod: "post",
    getArgsFromRequest: function (request) {
      var content = request.body;
      return [decodeURIComponent(content.hashtag)];
    }
  });
```

---

Listing 16: Definition einer REST-Methode mit dem Simple-Rest-Paket in Meteor

Des Weiteren müssen nur noch die Angaben für die Uniform Resource Locator (URL) sowie die Hypertext Transfer Protocol (HTTP)-Methode gemacht werden. Die restliche Arbeit übernimmt das Paket.

Die zweite Variante besteht in der händischen Definition der jeweiligen API-Methode in der Routen-Datei des Meteor-Servers. Als Beispiel dafür dient die auf *keepalive*-Methode:

---

```
Router.route('/api/keepalive', {where: 'server'})
  .post(function () {
    var sessionConfiguration =
      this.request.body.sessionConfiguration;
    var hashtag =
      decodeURIComponent(sessionConfiguration.hashtag);
    var privateKey =
      decodeURIComponent(sessionConfiguration.privateKey);

    if (!HashtagsCollection.findOne({hashtag: hashtag})) {
      this.response.writeHead(500);
      this.response.end("Hashtag not found");
    }

    if (HashtagsCollection.findOne({hashtag:
      hashtag}).privateKey !== privateKey) {
      this.response.writeHead(500);
      this.response.end("Missing permissions.");
    }
    Meteor.call('keepalive', hashtag);
  });
```

---

Listing 17: Definition einer REST-Methode über die Server-Routen in Meteor

Im oberen Teil der Routen-Definition werden die Body-Parameter ausgelesen und decodiert, anschließend werden Rechteüberprüfungen durchgeführt. Nur die vorletzte Zeile stößt die eigentliche Ausführung an: “*Meteor.call('keepalive', hashtag);*”. Damit wird die Meteor-Methode “keepalive” mit dem Hashtag als Übergabeparameter aufgerufen. Durch die saubere Kapselung der Funktionalität in einer eigenen Meteor-Routine ist das Aufrufen aus der API-Definition sowie aus dem regulären Quellcode des Clients möglich. Folglich erlangt arsnova.click eine geringe Komplexität durch die Kapselung der einzelnen Funktionen und gleichzeitig erhöht sich nicht die Wartbarkeit durch die Einführung der REST-API, da der Code nur innerhalb der Meteor-Methode gewartet werden muss.

Auf der anderen Seite wurde innerhalb von arsnova.click nicht immer so sauber gearbeitet wie im Falle der “keepalive”-Routine. Oft sind die einzelnen Funktionsabschnitte eng mit dem Code des Clients verflochten und auf die Verwendung des lokalen Speichers des Browsers angewiesen, der bei einem Schnittstellenaufruf nicht zur Verfügung steht. Ein Refactoring zur Aufgliederung in einzelne Meteor-Methoden hätte einen zu großen Aufwand im Rahmen des Integrationsprojektes dargestellt und wurde daher nicht vorgenommen. Zur Realisierung der Schnittstelle musste die benötigte Geschäftslogik erneut implementiert werden. Die dadurch entstandenen Redundanzen erhöhen die Wartbarkeit und können zur Inkonsistenz führen, wenn nur der Client- oder Schnittstellen-Code angepasst wird.

## 6 Zusammenfassung

Der Abschluss beinhaltet einen persönlichen Rückblick auf diese Arbeit, gewährt einen Blick auf die Zukunft des Projektes und zieht ein Fazit. Zunächst werden neben den interessanten Phasen auch die Herausforderungen vor und während der Bearbeitungszeit dargelegt. Weiterführend findet auch der klassische Vergleich der anfänglichen Ziele mit den Ergebnissen der Arbeit statt. Die letzten beiden Teile beschäftigen sich mit den nächsten Schritten des Projektes und fassen die Ergebnisse zusammen.

### 6.1 Rückblick

Das Thema der Abschlussarbeit stand aufgrund meiner vorherigen Tätigkeiten innerhalb des ARSnova-Projektes, größtenteils in Form der Entwicklung und Verbreitung von arsnova.click, schon eine ganze Zeit vor dem Beginn der Thesis fest. Die eigentlichen Forschungen begannen im Oktober 2016, die bereits gesammelten Erfahrungen innerhalb des ARSnova-Bereiches vereinfachten den Einstieg in das Thema ungemein und ließen den direkten Fokus auf den eigentlichen Inhalt der Abschlussarbeit zu: Wie und mit welchen Frameworks ist eine Integration in Office überhaupt möglich und welche Grenzen sind gesetzt. Gerade die Einarbeitung in die unbekannten Technologien wie den VSTO, das Austesten und Verstehen des überholungsbedürftigen arsnova.voting-Backends und der Analyse zur Möglichkeit einer REST-API-Definition für ein Meteor-Framework füllten die ersten Wochen gut aus.

In der Anfangszeit lag der Fokus auf einer Integrationslösung mit den Office Web-Add-ins, wie am Ende des Kapitels Analyse von PowerPoint Add-ins erläutert, stellten sich diese als nicht anforderungsgerecht heraus. Damit verschob sich die Entwicklung eines Add-ins auf die Windows-Ebene. Eine Softwarelösung für alle Betriebssysteme, ohne Kompromisse bei den funktionalen Anforderungen einzugehen, ist mit den von Microsoft zur Verfügung gestellten Integrationsmöglichkeiten nicht möglich. Von dem Zeitpunkt an war klar, dass sich das Projekt um ein PowerPoint Add-in mit dem VSTO-Framework handeln würde.

Nachdem feststand, dass das Projekt umsetzbar ist, wurden die ersten Geschäftsprozesse in Form eines Oberflächen-Konzeptes entworfen. Die einzelnen Ansichten mit ihren Funktionsweisen und Designs sollten für den Anwender einheitlich erscheinen und gut verteilt sein. Einen theoretischen Entwurf zu korrigieren ist in dieser Phase eines Projektes immer einfacher als direkt mit der Programmierung zu beginnen und die Implementierung später anpassen zu müssen beziehungsweise die Funktionalitäten umzulagern. Gerade in Bezug auf die Entwurfsmuster und damit auch die Projektarchitektur, welche nach den Oberflächendefinitionen entworfen wurde, ist es zu Beginn der Entwicklung einer neuen Software immer sinnvoll, zuerst die groben Richtlinien festzulegen. Nach der Definition der Oberflächen sowie der Projektstruktur konnten erste Entwurfsmuster wie die Dependency Injection oder der ViewPresenter implementiert werden. Damit war das Projekt für eine Implementierung der eigentlichen Inhalte vorbereitet.

Durch die Kapselung der einzelnen Ebenen in Kommunikations-, Geschäftslogik- und Darstellungsschicht konnten die jeweiligen Bereiche unabhängig voneinander entwickelt und getestet werden. Im Darstellungsbereich gab es bereits Vorkenntnisse bezüglich der WPF-Entwicklung, lediglich Neuheiten wie die Entwicklung des ViewPresenter oder

der Implementierung eines Watermark-Service nahmen größere Zeitspannen zur Nachforschung und Entwicklung in Anspruch. Die Geschäftslogik sowie die Verwendung der VSTO und damit auch die Anbindung an PowerPoint selbst stellten eine größere Herausforderung dar. Da PowerPoint bereits viele Funktionalitäten von Haus aus mitbringt, ist die Entwicklung von Add-ins ein seltener diskutiertes Thema im Internet. Generell stößt man bei der Suche nach PowerPoint-Funktionen mehr auf Anleitungen für einen Vortragenden selbst als für einen Entwickler. Durch das MSDN und entsprechende Fachliteratur war die Einarbeitung zwar etwas langwieriger, aber möglich. Die Forschungen stützten sich dabei auf drei unterschiedliche Bereiche:

- Entwicklung eines Menübandes / Erweiterung bestehender PowerPoint Menüs
- Anbindung an bestehende PowerPoint-Prozesse wie dem Selektieren einer bestimmten Folie
- Erzeugung und Manipulation von Folieninhalten, auch während der Präsentation selbst
- Erzeugung von Graphen mit Hilfe von Microsoft Office Excel

Im Anschluss an diesen sehr spannenden Teil, bei dem die genauen Grenzen der VSTO vorher noch nicht bekannt waren und die ursprünglichen Konzeptionspläne während der Forschung angepasst werden mussten, begann die Entwicklung der Kommunikationsschicht. Durch die Geschäftslogik waren die benötigten Aufrufe an die beiden Services, einen für `arsnova.click` und einen für `arsnova.voting`, bereits strikt definiert. Da die gamifizierte Variante noch keine API besaß, war die Implementierung dort zwar aufwendig, doch aus Forschungssicht betrachtet, nach der Analyse der zu verwendenden Frameworks, problemlos. Wenn der Entwickler der Schnittstelle diese anbindet, kennt er die benötigten Übergabeparameter und eventuelle Hürden sowie die genauen Auswirkungen eines API-Pfades. Bei `arsnova.voting` hingegen musste, gerade durch die aufgeblähten Models und die vielen Schnittstellen-Methoden, einiges an Forschung betrieben werden um herauszufinden, welche Pfade die benötigten Auswirkungen beinhalten und was es alles für Parameter gibt, beziehungsweise wofür diese stehen. Zur Verdeutlichung: Ein `arsnova.voting` Model, welches lediglich eine Frage repräsentiert, besitzt circa 70 Parameter. Dadurch wird der Zweck einer REST-API, nämlich eine unkomplizierte Anbindung eines Web-Services, aus meiner Sicht untergraben.

Mitte Februar 2017 neigte sich die Zeit der Forschung dem Ende zu, der verbleibende Monat sollte der Verschriftlichung des Projektes in Form dieser Thesis erfolgen. Zu diesem Zeitpunkt existierte ein Prototyp, der alle in der Analyse definierten Prozesse abbildet und erfolgreich `arsnova.click` sowie auch `arsnova.voting` in PowerPoint integriert. Eine erste Version liegt in Form eines Installers vor und kann verbreitet sowie auch verwendet werden.

Bei einem direkten Vergleich der Ziele des Projektes mit dem Stand der Abschlussarbeit kann durchaus von einer erfolgreichen Umsetzung gesprochen werden. Es ist bewiesen, dass es möglich ist, mehrere ARS-Systeme in PowerPoint zu integrieren, um die Präsentationssoftware um die Komponente des invertierten Klassenraums zu erweitern. Dabei ist das Add-in einfach zu installieren und gliedert sich scheinbar nativ in die Office-Umgebung ein. Somit stellt die entwickelte Software eine Erleichterung der Bedienung und damit einen eindeutigen Usability-Gewinn für das gesamte ARSnova-Projekt dar.

Ich selbst habe mich mit neuen Technologien auseinandersetzen dürfen und gelernt, welch einen großen Gewinn die Vernetzung verschiedener Softwaresysteme mit sich bringt. Durch Web-Services werden online viele Funktionalitäten zur Verfügung gestellt, die durch entsprechende Schnittstellen von überall aus verwendbar sind. Wenn große, weit verbreitete Anwendungen wie die Office-Suite zusätzlich eine Erweiterung des Systems in Form von Add-ins zulassen, können gute, bereits existierende Lösungen einfach in die Standardsoftware eines Anwenders integriert werden. In dieser Technik liegt ein großes Potential, da Funktionalitäten durch Web-Services nicht redundant implementiert werden müssen und Software in bestehende Systeme integriert wird, statt dem Nutzer die Einarbeitung in ein völlig neues Programm zuzumuten. Auch kann man sich so auf die Verbesserung bestehender Anwendungen konzentrieren, statt Bestehendes, wahrscheinlich weniger gut und mit “Kinderkrankheiten” versehen, zu replizieren.

## 6.2 Ausblick

Eine Anwendung ist nie wirklich fertig, es gibt immer Bugs zu beheben, die Performance zu optimieren oder neue Features einzuarbeiten. Dabei ist auch die ARSnova-Integrationslösung keine Ausnahme, gerade da die angebundenen Systeme noch viele Funktionalitäten besitzen, die aus Zeitgründen noch nicht integriert wurden. Es folgt eine Übersicht über Features, deren Entwicklung in naher Zukunft möglich ist:

- Drucken der Präsentation

Zur Zeit werden für eine Frage, je nach gewähltem ARSnova-System, 2-3 Folien erzeugt. Möchte der Vortragende die Vortragsfolien dem Auditorium im Nachhinein zur Verfügung stellen, soll die Frage mit seinen Antwortmöglichkeiten und Ergebnissen ebenfalls festgehalten werden. Doch Informationen wie der Countdown sowie mehrere Folien sind dabei uninteressant. Es ist möglich die Präsentation vor dem Drucken geringfügig zu manipulieren, damit nur eine Folie mit der zusammengefassten Umfrage darauf festgehalten wird. Auf diese Art und Weise erhält der Nutzer eine saubere und übersichtlichere Zusammenfassung von ARSnova-Inhalten nach einem Vortrag.

- Session-Konfiguration

arsnova.click wie auch arsnova.voting bringen unterschiedliche Session- und Fragen-Konfigurationen mit, die zur Zeit nicht alle implementiert sind. Diese sollen in naher Zukunft nachgezogen werden, um dem Anwender der Integrationslösung keinen Nachteil zu bieten, wenn er auf die Verwendung des Web-Systems verzichtet. Beispiele dafür sind:

- Die richtige Frage zu der Ergebnisdarstellung einblenden.
- Multiple-Choice-Fragen um den Graphen “teilweise richtig” ergänzen.
- Nur arsnova.click: Antworttexte auf Buttons darstellen.
- Nur arsnova.voting: Festlegung des Themas.

- **Markdown- und LaTeX-Unterstützung**

Ein großer Gewinn des ARSnova-Projektes ist die Markdown- und LaTeX-Unterstützung in all seinen Applikationen. Dies ist zur Zeit noch nicht implementiert und stellt einen größeren Aufwand dar, da PowerPoint-Folien diese Formate nicht unterstützen. Folglich müssen alle Markdown-Auszeichnungen in PowerPoint Formatierungen übertragen werden, um die Inhalte auch auf den Folien sauber darstellen zu können. Ob und wie eine LaTeX-Integration möglich ist, gilt es zu ermitteln. Vorstellbar wäre ein externer Generator, der ein Bild mit dem übersetzten LaTeX-Code zurückliefert, welches in eine Folie eingebunden werden kann.

- **Lokalisierung**

Zur Zeit liegt die Integrationslösung in englischer und deutscher Sprache vor. Da die restlichen ARSnova-Anwendungen noch weitere Sprachen wie Französisch, Spanisch und Italienisch unterstützen, ist es empfehlenswert, auch diese mit aufzunehmen. Solange die restlichen Sprachen nicht zur Verfügung stehen, werden die entsprechenden Anwender nicht von dem Web-System auf die Integration umsteigen.

Die Übersetzungen sollten erst angefertigt werden, wenn eine stabile Version der Anwendung erreicht ist. Da in naher Zukunft noch viele weitere Features entwickelt werden und sich damit bestehende Texte ändern beziehungsweise neue dazukommen, ist eine Übersetzung zum jetzigen Zeitpunkt nicht zukunftsicher.

### **6.2.1 arsnova.voting Backend**

Wie bereits mehrfach angedeutet befindet sich das arsnova.voting-Backend in keinem guten Zustand. Daher findet zeitgleich eine Neuentwicklung statt, mit der viele Performance-Probleme behoben werden sollen. Auch wird die Schnittstelle strukturierter und schlanker konzipiert werden. Gerade aus Gründen der Performance ist es ratsam, die Integrationslösung nach dem Release des arsnova.voting-Backends in der Version auf dieses umzustellen. Je nachdem wie stark sich die Schnittstellendefinitionen unterscheiden, ist der Aufwand unterschiedlich. Durch die Kapselung der Kommunikation mit den REST-APIs sind aus Sicht der hiesigen Implementierung die besten Voraussetzungen für eine schnelle Anpassung gegeben.

Mit dem neuen Backend wäre es denkbar, ein Login und damit eine Authentifizierung einzufügen, was bei arsnova.click aus Datenschutzgründen nie möglich sein wird. Dadurch kann der Anwender auf bereits vorher definierte Sessions mit dem angemeldeten Benutzeraccount zugreifen und diese direkt in die Vortragsfolien integrieren. Eine solche Übernahme würde den Umstieg vom Web-System auf das Add-in unglaublich erleichtern, da bestehende Umfragen nicht erneut angepasst werden müssen. Die Implementierung sieht eine solche Authentifizierung mit bestimmten Model-Eigenschaften und entsprechenden Methoden innerhalb des Kommunikations-Services bereits vor, eine Implementierung sollte sich nicht allzu aufwendig gestalten.

### **6.2.2 arsnova.click API**

In dem Abschnitt arsnova.click wurden die Schnittstellendefinitionen für das Meteor-Framework erläutert. Für neue Features müssen auf gleiche Art, wenn nicht über eine Erweiterung bestehender Methoden lösbar, neue API-Methoden definiert werden. Noch wichtiger ist bei der Weiterentwicklung von arsnova.click darauf zu achten, dass



die redundanten Codestellen, wenn sie für den Client angepasst werden, auch in der Schnittstellendefinition angepasst werden. Selbiges gilt auch andersherum. Ebenfalls ist es bei einer Anpassung der verwendeten Meteor-Methode wichtig, die abgebildeten Prozesse unverändert zu lassen. Sollte dies unumgänglich sein, ist bei einer Erweiterung oder Reduktion dieser Methoden die Kommunikation mit dem PowerPoint Add-in zu überprüfen und die Anwendung, wenn benötigt, um neue Aufrufe zu ergänzen. Die betroffenen Meteor-Methoden sind zur Zeit:

- “EventManagerCollection.add”
- “EventManagerCollection.setActiveQuestion”
- “EventManagerCollection.setSessionStatus”
- “EventManagerCollection.showReadConfirmedForIndex”
- “EventManagerCollection.startQuiz”
- “keepalive”
- “Main.killAll”
- “Question.startTimer”
- “QuestionGroupCollection.persist”
- “SessionConfiguration.addConfig”

### 6.3 Fazit

Durch die Entwicklung eines gesamten Projektes von Grund auf, inklusive einer Durchführbarkeits-Analyse, dem Vergleich verschiedener Frameworks, der Konzeption mit der Definition einer kompletten Projektarchitektur und seinen Entwurfsmustern sowie dem Auseinandersetzen und Erlernen verschiedenster .NET- und Web-Technologien sowie zwei bereits bestehender Web-Applikationen hat ein breites Spektrum von Wissen auf die Probe gestellt und erweitert. Es zeigt mir, dass man in der Softwareentwicklung aufgrund der vielen verschiedenen Applikationen und Frameworks nie auslernt, doch auch, dass mich das Studium darauf vorbereitet hat, mit diesen alltäglichen Herausforderungen umzugehen.

In naher Zukunft werde ich mich in meiner Freizeit weiter mit dem Projekt beschäftigen. Zum einen trage ich damit einen Teil zur Verbesserung der Lehre und damit der Ausbildung meiner Nachfolger bei, zum anderen ist es schön, an einem Open-Source-Projekt mitzuarbeiten, an dem man wachsen kann und welches den eigenen Entwicklungsprinzipien zugrunde liegt. Außerdem möchte niemand, dass ein öffentliches Projekt nicht gepflegt wird, in dem man als Hauptentwickler beteiligt ist. Zusätzlich hoffe ich, dass sich mehr Studenten der Technischen Hochschule Mittelhessen in die .NET-Welt finden, eventuell auch durch die Weiterarbeit an dem ARSnova-Integrationsprojekt.

Für den positiven Verlauf des Projektes spricht das Resultat: Ein einsetzbares PowerPoint Add-in, welches beide ARS-Systeme integriert. Nicht nur einen Prototyp, sondern eine einsetzbare Anwendung bereitstellen zu können, spricht für den Erfolg dieses Entwicklungsprojektes.

## Anhang

Im Anhang befindet sich eine Entwicklungsanleitung für das entwickelte PowerPoint Add-in. Mit Hilfe der Instruktionen ist eine Weiterentwicklung der Integrationslösung von jeder Person mit fundamentalen C#- und .NET-Kenntnissen möglich. Auch existiert eine Anleitung für eine rein textuelle Änderung, für die keine besonderen Vorkenntnisse vonnöten sind.

Vor größeren Anpassungen sollten sich zukünftige Entwickler das Kapitel Konzeption durchlesen, um ein Verständnis für die Projektstruktur und die verwendeten Entwurfsmuster zu erlangen. Auch gibt es zu den unten aufgeführten Themen Abschnitte in der Konzeption und Implementierung, dort werden die jeweiligen Punkte ausführlicher behandelt. Dieser Abschnitt stellt mehr eine Kurzanleitung dar als eine ausführliche Erläuterung.

- 1. Einrichten einer lokalen Entwicklungsumgebung
- 2. Build-Prozess
- 3. Textänderungen
- 4. Erzeugung einer aktuellen Setup-Datei

### Einrichten einer lokalen Entwicklungsumgebung

Als Entwicklungsumgebung ist zwingend Microsoft Visual Studio zu verwenden, da nur mit dieser IDE auch ein neues Setup erzeugt werden kann. Weiterhin muss das .NET-Framework in der Version 4.6 installiert sein sowie die VSTO in der 2010-er Version (<https://www.microsoft.com/de-DE/download/details.aspx?id=48217>). Alle weiteren, benötigten Abhängigkeiten werden über den NuGet-Paketmanager verwaltet und, wenn nicht vorhanden, bei der ersten Build-Ausführung nachgeladen. Es empfiehlt sich, für eine einfachere Verwendung den JetBrains ReSharper zu installieren. Dieser formatiert unter anderem automatisch den Code nach den festgelegten Code-Styles.

Der Quellcode wird mit Git verwaltet und das Repository ist unter der Adresse <https://git.thm.de/arsnova/powerpoint-integration> zu erreichen.

Visual Studio besitzt eine interne Code-Qualität-Überprüfung. Die Einstellungen für das jeweilige Projekt werden automatisch geladen, bei Verstößen gibt der Compiler Warnungen aus. Ebenfalls werden Verstöße gegen die Code-Richtlinien sofort in der IDE (gelbe Hinterlegung am rechten Scroll-Rand) angezeigt.

### Build-Prozess

Aufgrund eines fehlenden Windows-Servers ist es nicht möglich, die Anwendung auf den Servern der Technischen Hochschule zu kompilieren. Daher müssen Code-Qualität, Tests und der Build selbst immer lokal ausgeführt und überprüft werden.

Es besteht die Möglichkeit, unter GitLab einen Build-Job auf dem lokalen Computer einzurichten. Eine Anleitung ist hier zu finden: <https://gitlab.com/gitlab-org/gitlab-ci-multi-runner/blob/master/docs/install/windows.md>. Das dafür benötigte Build-Skript ist im Root-Verzeichnis des Integrationsprojektes zu finden (build.bat).

In dem Skript sind im oberen Teil zwei Variablen gesetzt: "msbuild", "slnPath". Die

erste verweist auf die “MSBuild.exe”, die zum Kompilieren der Anwendung benötigt wird, die zweite verweist auf die “ARSnovaPPIntegration.sln”-Datei, die Projektmappe. Diese beiden Pfadangaben müssen stimmen und eventuell angepasst werden, damit ein erfolgreicher Build möglich ist.

## Textänderungen

Die Internationalisierungs-Dateien befinden sich in dem Unterprojekt “ARSnovaPPIntegration.Common.Resources”. Für jede Sprache ist ein “Resources”-Dokument vorhanden. In diesen findet eine Zuordnung zwischen dem Platzhalter-Element, in der Regel der englischen Beschreibung eines Textes, sowie der Übersetzung statt. Bei den Platzhalter-Elementen werden die Leerzeichen für eine Besserung der Identifikation von freien Stellen durch Unterstriche ersetzt. Texte, die keinem Platzhalter zugeordnet sind, werden so zurückgegeben, wie sie sind. Das heißt, dass für Begriffe, die in allen Sprachen gleich sind (beispielsweise “ARSnova”), keine Zuordnung in den jeweiligen “Resources”-Dateien angelegt werden muss.

Um die ständige Konsistenz der Übersetzungen zu gewährleisten empfiehlt es sich, die Texte von allen gepflegten Sprachen auf einmal zu ändern beziehungsweise hinzuzufügen.

## Erzeugung einer aktuellen Setup-Datei

Vor jeder neuen Setup-Erzeugung muss ein Release-Build des gesamten Projektes (außer dem Installer selbst) erfolgen. Anschließend werden in dem Projekt “ARSnovaPPIntegration.Installer.Setup” die allgemeinen Informationen um eine erhöhte Versionsnummer angepasst. Wenn neue Projekte hinzugefügt wurden, müssen die entsprechenden Projektpfade und Assemblies im Abschnitt “Anwendungs-Dateien” referenziert werden. Bei einem solchen Schritt sollte die Erweiterung der Projektstruktur gut überlegt sein, für die meisten Fälle sollte die bestehende Struktur ausreichend sein. Die restlichen Setup-Konfigurationen müssen nicht angepasst werden.

In der Visual Studio Build-Konfiguration muss für das Projekt “ARSnovaPPIntegration.Installer.Setup” SingleImage gewählt werden, da das Setup in Form einer ausführbaren Datei und nicht als CD-Abbild erstellt werden soll. Anschließend wird das Projekt der Installationsroutine einzeln gebaut, nicht die gesamte Projektmappe. Nach der erfolgreichen Durchführung liegt die neue Installationsdatei unter dem Pfad:

“ARSnovaPPIntegration/bin/Express/SingleImage/DiskImages/DISK1/setup.exe”.

## Literatur

- Apache. (2017). *Extensions development*. Apache OpenOffice wiki. ([https://wiki.openoffice.org/wiki/Extensions\\_development](https://wiki.openoffice.org/wiki/Extensions_development) Eingesehen am 14.02.2017)
- ARSnova-Team der TH Mittelhessen. (2017a). *arsnova.click (Version 2.0) [Mobile Application Software]*. (<https://github.com/thm-projects/arsnova.click>)
- ARSnova-Team der TH Mittelhessen. (2017b). *ARSnova PowerPoint-Integration (Version 0.9) [PowerPoint Add-in]*. (<https://git.thm.de/arsnova/powerpoint-integration>)
- ARSnova-Team der TH Mittelhessen. (2017c). *Backend arsnova.voting (Version 2.5.0) [Mobile Application Software]*. (<https://github.com/thm-projects/arsnova-backend>)
- ARSnova-Team der TH Mittelhessen. (2017d). *Frontend arsnova.voting (Version 2.5.0) [Mobile Application Software]*. (<https://github.com/thm-projects/arsnova-mobile>)
- C., A. (2015). *What is your review of Meteor (Javascript platform)?* (<https://www.quora.com/What-is-your-review-of-Meteor-Javascript-platform> Eingesehen am 07.02.2017)
- Dascalescu, D. (2016a). *Comparing Meteor.js and the MEAN stack*. ([https://wiki.dandasescu.com/essays/meteor\\_js\\_vs\\_the\\_mean\\_stack](https://wiki.dandasescu.com/essays/meteor_js_vs_the_mean_stack) Eingesehen am 07.02.2017)
- Dascalescu, D. (2016b). *What framework should I choose to build a web app? ... and 10 reasons why the answer should be "Meteor"*. ([https://wiki.dandasescu.com/essays/why\\_meteor](https://wiki.dandasescu.com/essays/why_meteor) Eingesehen am 07.02.2017)
- Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley. (<https://www.martinfowler.com/eaCatalog/separatedInterface.html> Eingesehen am 21.02.2017)
- Fowler, M. (2004). *Inversion of Control Containers and the Dependency Injection pattern*. (<https://martinfowler.com/articles/injection.html> Eingesehen am 07.02.2017)
- Huber, T. C. (2015). *Windows Presentation Foundation Das umfassende Handbuch*. Rheinwerk Computing.
- Kibler, S. (2015). *Audience Response Systeme - Möglichkeiten und Grenzen ihres Einsatz bei der Vermittlung von Informationskompetenz in wissenschaftlichen Bibliotheken*. b.i.t.online 18(2015) Nr. 2. (<http://www.b-i-t-online.de/heft/2015-02-fachbeitrag-kibler.pdf> Eingesehen am 21.01.2017)
- Leitner, S. (2011). *So lernt man lernen*. (18. Aufl.). Verlag Herder.
- Martin, R. C. (2003). *Agile Software Development: Principles Patterns And Practices*. Prentice Hall.
- Microsoft. (2016). *The MVVM Pattern*. MSDN. (<https://msdn.microsoft.com/en-us/library/hh848246.aspx> Eingesehen am 21.02.2016)
- Microsoft. (2017a). *Application Events*. MSDN. ([https://msdn.microsoft.com/en-us/library/microsoft.office.interop.powerpoint.application\\_events\(v=office.14\).aspx](https://msdn.microsoft.com/en-us/library/microsoft.office.interop.powerpoint.application_events(v=office.14).aspx) Eingesehen am 10.02.2017)
- Microsoft. (2017b). *Bereitstellen einer Office-Lösung mithilfe von Windows Installer*. MSDN. (<https://msdn.microsoft.com/de-de/library/cc442767.aspx#Create> Eingesehen am 27.02.2017)
- Microsoft. (2017c). *Introduction to LINQ Queries (C#)*. MSDN. (<https://msdn.microsoft.com/de-de/library/bb397906.aspx> Eingesehen am 27.02.2017)

- Microsoft. (2017d). *Visual Studio 2010 Tools for Office Runtime*. MSDN. (<https://www.microsoft.com/en-us/download/details.aspx?id=54251> Eingesehen am 16.12.2016)
- Miedl, W. (2013). *Unternehmen bleiben bei Microsoft Office*. Cancom - Das Business IT Journal. (<http://www.cancom.info/2013/10/forrester-unternehmen-halten-microsoft-office-vorerst-die-treue/> Eingesehen am 14.02.2017)
- Quibeldey-Cirkel, K. (2016). *Lernwiderstände sichtbar machen mit dem Audience Response System ARSnova*. In T. Knaus & O. Engel (Hrsg.), *Wi(e)derstände: Digitaler Wandel in Bildungseinrichtungen* (S. 183-198). Reihe: fraMediale. Bd. 5. München: kopaed. ([https://arsnova.thm.de/blog/wp-content/uploads/2017/02/fraMediale\\_Band5\\_2016\\_Quibeldey-Cirkel.pdf](https://arsnova.thm.de/blog/wp-content/uploads/2017/02/fraMediale_Band5_2016_Quibeldey-Cirkel.pdf) Eingesehen am 27.02.2017)
- Quibeldey-Cirkel, K. (2017). *arsnova.io, der Blog des ARSnova-Projektes*. (<https://arsnova.io>)
- Rahman, M. (2014). *C# Deconstructed: Discover how C# works on the .NET Framework*. Apress.
- Schmidt, S. J. (2003). *REPORT Literatur- und Forschungsreport Weiterbildung 3/2003: Gehirn und Lernen*. (<http://www.die-bonn.de/id/1822> Eingesehen am 16.12.2016)
- Schuh, G. & Stich, V. (2014). *Enterprise-Integration*. Springer Vieweg.
- Slaghuis, B. (2013). *Ziele. Was motiviert uns am stärksten?* (<http://www.bernd-slaghuis.de/karriere-blog/ziele/> Eingesehen am 02.02.2017)
- Thielsch, M. T. & Förster, N. (2007). *Präsentationssoftware: Nutzung und funktionale Anforderungen*. Usability Professionals. ([http://www.thielsch.org/download/thielsch\\_2007.pdf](http://www.thielsch.org/download/thielsch_2007.pdf) Eingesehen am 07.02.2017)
- von Ebner-Eschenbach, M. F. (1911). *Aphorismen*. (<https://www.aphorismen.de/zitat/1580> Eingesehen am 10.12.2016)
- Wuensch, K. L. (2015). *How Do You Pronounce "Likert?" What is a Likert Scale?* East Caroline University Department of Psychology. (<http://core.ecu.edu/psyc/wuenschk/StatHelp/Likert.htm> Eingesehen am 21.01.2017)