**Technologische Grundlagen Cloudbasierter Anwendungen**

Thema 5 „User Experience“

Projektdokumentation

Wirtschaftsinformatik Master an der Hochschule der Medien

Von

Florian Blessing

Lisa Böhler

Andrea Breimayer

Wendelin Herrmann

Markus Götz

Wintersemester 2016/2017

Inhaltsverzeichnis

[Darstellung des Themas 4](#_Toc467441863)

[Theoretische Grundlagen 4](#_Toc467441864)

[Cloudbasierte Anwendungen 5](#_Toc467441865)

[User Experience 5](#_Toc467441866)

[Google Hangouts 6](#_Toc467441867)

[Projektorganisation 6](#_Toc467441868)

[Teammitglieder 6](#_Toc467441869)

[Aufgabenverteilung und Meilensteine 7](#_Toc467441870)

[Grundlegende Ansätze 9](#_Toc467441871)

[Anforderungsanalyse 12](#_Toc467441872)

[Konstruktion Anforderungskatalog 13](#_Toc467441873)

[Anforderungskatalog 13](#_Toc467441874)

[Visualisierungskonzepte 14](#_Toc467441875)

[Visualisierungskonzept I - Konventionell 14](#_Toc467441876)

[Visualisierungskonzept II - MindMap 15](#_Toc467441877)

[Visualisierungskonzept III - Pyramide 15](#_Toc467441878)

[Visualisierungskonzept IV - Litfaßsäule 16](#_Toc467441879)

[Technologie Scouting 17](#_Toc467441880)

[Auswahl der Visualisierungs-Frameworks 17](#_Toc467441881)

[Auswahl der Code-Frameworks 18](#_Toc467441882)

[Auswahl der Programmiermethodik 19](#_Toc467441883)

[Entwicklungsumgebung 20](#_Toc467441884)

[Zwischenfazit 20](#_Toc467441885)

Abbildungsverzeichnis

[Abbildung 1: Google Hangouts Layout 6](#_Toc467441947)

[Abbildung 2: Übersicht der Meilensteine in Projektphase 1 7](#_Toc467441948)

[Abbildung 3: Möglichkeiten der Systemintegration 9](#_Toc467441949)

[Abbildung 4: Google Hangouts mit eingebundener App 10](#_Toc467441950)

[Abbildung 5: Google Hangouts mit eingebundener Erweiterung 11](#_Toc467441951)

[Abbildung 6: Google Hangouts "Problembutton" 12](#_Toc467441952)

[Abbildung 7: Visualisierungskonzept I - Konventionell 14](#_Toc467441953)

[Abbildung 8: Visualisierungskonzept II - MindMap 15](#_Toc467441954)

[Abbildung 9: Visualisierungskonzept III - Pyramide 16](#_Toc467441955)

[Abbildung 10: Visualisierungskonzept IV - Litfaßsäule 16](#_Toc467441956)

[Abbildung 11: Vergleich Visualisierungs-Frameworks 18](#_Toc467441957)

[Abbildung 12: Vergleich Code-Frameworks 18](#_Toc467441958)

[Abbildung 13: Vergleich Programmiermethodik 19](file:///C:\Users\Lisa\Downloads\aktuellDoku.docx.docx#_Toc467441959)

# Darstellung des Themas

In global vernetzten Unternehmen stehen Mitarbeiter immer öfter vor der Herausforderung mit Kollegen in Außenstandorten oder Tochtergesellschaften zusammen zu arbeiten, um den Unternehmenserfolg zu garantieren. Absprachen und Meetings werden nicht mehr nur vor Ort abgehalten, sondern finden mittlerweile auch vermehrt online mittels verschiedener digitaler Medien statt.

Im Rahmen der Vorlesung „Technologische Grundlagen cloudbasierter Internet-Anwendungen“ soll nun eine Cloudanwendung auf Basis von Google Hangouts entwickelt werden, welche es ermöglicht Videokonferenzen mit einem System zu verknüpfen, das mittels Speech Tokenization das Gespräch analysiert und relevante Dokumente automatisch ermittelt, vorschlägt und anzeigt.

Das zu entwickelnde Zielsystem wurde unterteilt in fünf einzeln zu betrachtende Challenges. In kleineren Projektteams sollen die verschiedenen Aufgabenstellungen bearbeitet werden. Am Markt verfügbare Module und vorgefertigte Softwarepakete sollen unter technischen und wirtschaftlichen Gesichtspunkten betrachtet und verglichen werden. Ziel ist es dabei, dass die Studierenden sich mit den Herausforderungen des Cloud Computing und der Entwicklung einer Software in Teamarbeit auseinandersetzen. Die Studenten sollen sich dabei an einem professionellen Software-Engineering-Prozess orientieren und während der Projektarbeit einen wasserfallartigen oder agilen Ansatz der Softwareentwicklung verfolgen. Die Studierenden sollen dabei Wissen zu cloudbasierten Anwendungen eigenständig erwerben und im Bachelor erworbenes fachliches Wissen vertiefen und anwenden.

Das Gesamtsystem soll innerhalb einer Drei-Schichten-Architektur aus Präsentationsschicht, Applikationsschicht und Datenbankschicht implementiert werden. Konkrete Aufgabenstellung des Projektteams ist es dabei die Challenge Fünf „User Experience“ umzusetzen. Hier soll eine Komponente innerhalb der Präsentationsschicht entwickelt werden, welche eine adäquate Visualisierung und Interaktion mit den in der Applikationsschicht erarbeiteten Recommendations ermöglicht. Weitere technische Anforderungen an das zu entwickelnde System werden im Anforderungskatalog erläutert (siehe Kapitel 4).

Diese Projektarbeit ist untergliedert in zwei Phasen. In Phase eins geht es im ersten Schritt darum, Anforderungen an die Software zu ermitteln und in einem Pflichtenheft bzw. einem Anforderungskatalog zu dokumentieren. Im zweiten Schritt dieser Phase gilt es passende Frameworks zu finden, mit denen die Anforderungen erfüllt und das System umgesetzt werden können. In der zweiten Phase werden verschiedene Umsetzungsformen des GUI in Fallstudien evaluiert, um bis zum Ende des Semesters ein funktionierendes GUI-System entwickeln zu können. Dieses soll im kommenden Semester mit den Entwicklungen der anderen Projektteams zusammengeführt werden, um das gewünschte Gesamtsystem realisieren zu können.

# Theoretische Grundlagen

Um ein Verständnis für das vorliegende Problem zu erhalten und ein gemeinsames Verständnis zu schaffen, werden im nachfolgenden Kapitel die theoretischen Grundlagen erarbeitet. Hierfür wird zum einen die Charakteristik einer Cloud-Anwendung im Vergleich zu einer Web-Anwendung herausgearbeitet und zum anderen der Begriff “User Experience” näher definiert.

## Cloudbasierte Anwendungen

Vor einer Betrachtung von Cloud Anwendungen bietet es sich an zuerst Web Applikationen näher zu erläutern. **Web Apps** zeichnen sich vor allem durch die Erreichbarkeit einer bestimmten Funktion über eine Webseite aus. Es wird somit ein Webbrowser benötigt um auf die gewünschten Funktionalitäten zuzugreifen. Web Anwendungen bestehen serverseitig häufig aus Scriptsprachen wie PHP oder ASP und clientseitig aus HTML, JavaScript und Adobe Flash. Ein wichtiges Merkmal von Web Apps ist, dass diese über einen Webbrowser erreicht werden können, ohne dass eine Installation einer App von Nöten ist.[[1]](#footnote-1) Typische Beispiele für Web Apps sind etwa Onlinebanking-Systeme und Webshops.

**Cloud Apps** sind ebenfalls Funktionen die auf einem im Internet stehenden Server betrieben werden, allerdings mit dem entscheidenden Unterschied, dass sie nicht zwingend über ein Browserfenster angesteuert werden müssen. Sie gleichen hierbei einer Kombination aus klassischen Web Apps wie auch klassischen Desktop Apps. Hierbei kommunizieren diese Apps z.B. über eine App oder eine Desktopanwendung mit der Cloud, bieten jedoch meist ein Webinterface als Alternative an. Cloud Apps besitzen meist fortgeschrittene Funktionen wie etwa, dass Daten offline bearbeitet werden können und erst später mit dem Server synchronisiert werden. Es gibt Zusatzfunktionen wie Backups, Datenkompression und Security Maßnahmen.[[2]](#footnote-2) Typische Beispiele für Cloud Apps sind etwa Dropbox, Google Drive oder aber auch Evernote.

Um eine Web App zu einer Cloud App weiterzuentwickeln, sollte sichergestellt werden, dass diese mandantenfähig[[3]](#footnote-3) ist. Dies wird benötigt, um verschiedenen Anforderungen und Bedürfnisse für die Verbraucher zu unterstützen. Außerdem sollte sie Virtualisierungstechnologien, die für Cloud-App-Anwendungen eine führende Rolle spielen unterstützen. Web-Anwendungen sollten entweder weiterentwickelt oder neu designed werden, um zu Cloud-Anwendungen zu werden.[[4]](#footnote-4)

## User Experience

User Experience beschreibt das Nutzer- bzw. Nutzungserleben eines Produktes. In der aktuellen ISO 9241-210 wird User Experience wie folgt beschrieben: “all aspects of the user’s experience when interacting with the product, service, environment or facility. [...] It includes all aspects of usability and desirability of a product, system or service from the user’s perspective”.[[5]](#footnote-5)

In der Forschung gibt es eine Reihe unterschiedlicher Ansätze, User Experience zu definieren. Dabei stößt man immer wieder auf drei Charakteristiken: Ganzheitlich, Subjektiv und Positiv. Ganzheitlich meint dabei, die “Balance zwischen instrumentellen und nicht-instrumentellen Qualitäten, wie Schönheit, Neuartigkeit, Herausforderung oder Selbstausdruck”. Die Subjektivität beschreibt die wahrgenommene und nicht die tatsächliche Qualität des Produktes. Diese Wahrnehmung bestimmt zukünftig die Nutzung und Kommunikation über das Produkt. Die Charakteristik des Positiven fokussiert sich auf Freude, Spaß, Attraktivität, Herausforderungen und Schönheit.[[6]](#footnote-6)

## Google Hangouts

Google Hangouts ist ein Programm des Entwicklers Google. Es kann zur klassischen Videotelefonie verwendet werden und hierbei durch eingebaute Applikationen um Funktionen erweitert werden. Hangouts kann sowohl im Browser, über eine eigene Applikation und auch auf Mobilgeräten gestartet werden. In Abbildung 1 ist der typische Layout eines Hangout-Gesprächs erkennbar. Während am unteren Rand eine Übersicht aller Gesprächsteilnehmer platziert ist, wird der Großteil des Bildes automatisch mit dem Video des aktuell Sprechenden gefüllt.

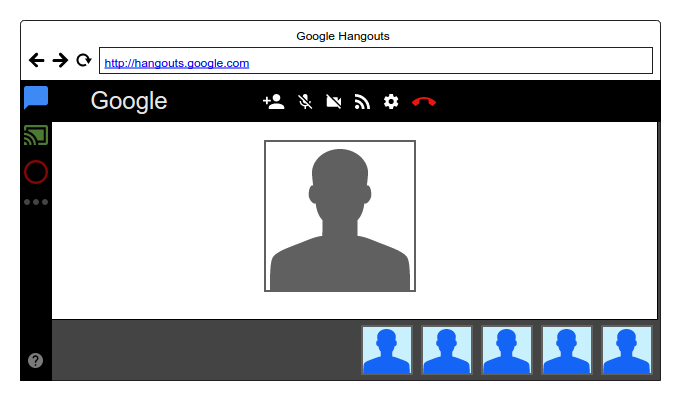


Abbildung 1: Google Hangouts Layout

# Projektorganisation

Zu Beginn des Projekts gilt es zunächst, Aufgaben und Ziele zu analysieren, um Fehlern präventiv entgegen zu wirken. Das Team erarbeitet gemeinsam die zu erledigenden Schritte und definiert Meilensteine. Um den hohen Arbeitsaufwand zu bewältigen, werden Aufgaben gemäß der individuellen Stärken und Vorlieben der einzelnen Teammitglieder verteilt. Die nachfolgenden Kapitel zeigen die identifizierten Problemstellungen sowie die Aufgabenverteilung und das Projekttiming.

## Teammitglieder

Das Team setzt sich zusammen aus fünf Masterstudenten des Studiengangs Wirtschaftsinformatik an der Hochschule der Medien:

Florian Blessing

Lisa Böhler

Andrea Breimayer

Wendelin Herrmann

Markus Götz

## Aufgabenverteilung und Meilensteine

**Meilensteine Phase 1**

Nachfolgend ist eine kurze Übersicht über die definierten Meilensteine gegeben, anhand denen die Definition der Aufgabenpakete im Detail sowie die Aufgabenverteilung vorgenommen wurden. Details zu den technischen Rahmenbedingungen und den aufgetretenen Fragestellungen werden im Kapitel „Pflichtenheft“ aufgeführt. EPK und Wireframes sind als separate Dateien auf der Projekthomepage enthalten.

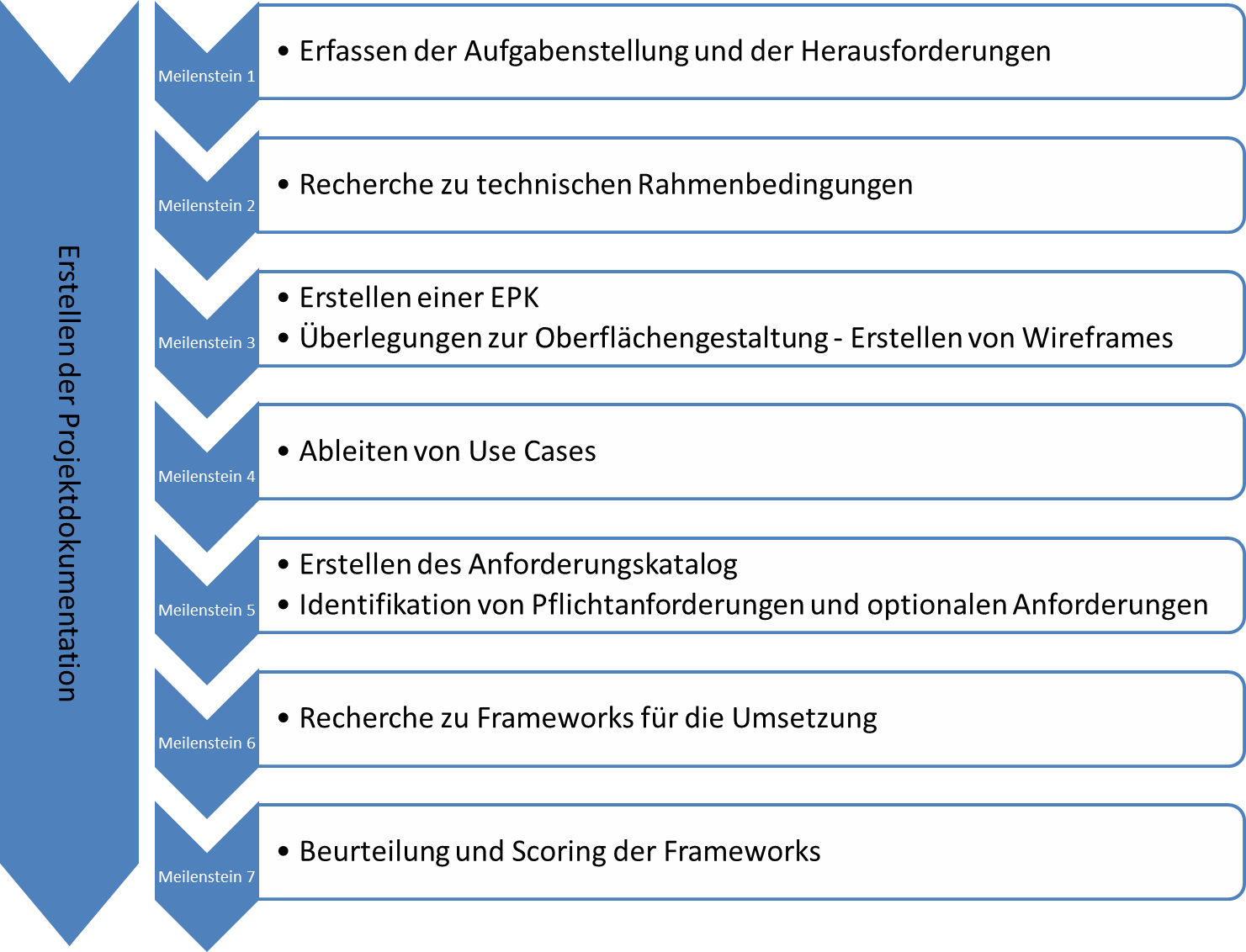


Abbildung 2: Übersicht der Meilensteine in Projektphase 1

**Aufgabenverteilung Projekt-Phase 1:**

|  |  |
| --- | --- |
| Erstellen der Gliederung/Aufbau der Projektdokumentation | Andrea |
| Identifikation von Pflichtanforderungen und optionalen Anforderungen | Alle |
| Erstellen von Wireframes und EPK | Andrea, Lisa, Wendelin |
| Definieren der Use Cases | Alle |
| Identifizieren von Tools zur Erstellung von javabasierten Webapplikationen | Florian, Markus |
| Richtlinien für die Erstellung des Anforderungskatalogs | Lisa, Wendelin |
| Erstellen des Anforderungskatalogs | Lisa, Wendelin |
| Review Anforderungskatalog | Alle |
| Scoren und beurteilen der einzelnen Frameworks | Alle |
| Erstellen der Zwischenpräsentation | Alle |

Nachfolgend ist eine kurze Übersicht über die durchgeführten Projektmeetings des Teams sowie eine kurze Zusammenfassung der dort besprochenen Inhalte aufgeführt.

Das Team hat zur Bewältigung der Challenge eine agile Vorgehensweise verfolgt und zu Beginn ein Ziel definiert, welches bis zum Ende von Phase 1 erreicht sein soll. Innerhalb der Projektmeetings wurden dann jeweils die nächsten Steps und die Aufgabenverteilungen im Detail festgelegt. Die Aufgabenverteilung ist später im Kapitel vorzufinden.

**Übersicht über die Projektteammeetings in Phase 1 des Projekts:**

|  |  |
| --- | --- |
| Datum | Kurzprotokoll |
| 20.10.2016 | Grundüberlegungen zum Vorgehen und zur Aufgabenstellung  Erstellen eines Fragenkatalogs für die Absprache mit dem Dozenten |
| 24.10.2016 | Definition der Erwartungen mit dem Dozenten  Definieren von ersten Arbeitspaketen |
| 29.10.2016 | Hangouts-Session:  Absprache der Möglichkeiten zur Integration von Hangouts:  Hangouts in eine Eigenentwicklung zu integrieren ist nicht möglich.  Hangouts bietet die Möglichkeit eigenentwickelte Apps via Schnittstelle zu integrieren.  Möglichkeit 1: Entwicklung der zu integrierenden App in Javascript  Möglichkeit 2: Entwicklung der zu integrierenden App in Java, Compilierung mit Google Web Toolkit in Javascript |
| 4.11.2016 | Erstellen des Konzepts für die Usecase Diagramme |
| 7.11.2016 | Absprache des Vorgehens mit dem Dozenten  Präsentation erster Ergebnisse (App innerhalb von Google Hangouts) und Vorstellung der Wireframes  Anregung Dozent: genau prüfen, ob eine Integration von Google Hangouts in eine Eigenentwicklung nicht möglich ist  Weitere Anregung: dynamisches „Markieren“ der vorgeschlagenen Dokumente innerhalb der gesamten Dokumentenablage zum Projekt (kein klassischer Dialog zur Abfrage, ob das Dokument angezeigt werden soll)  Grobe Absprache der Fallbeispiele für Teil zwei der Projektphase (mehrere verschiedene grafische Oberflächen sollen präsentiert werden) |
| 14.11.2016 | Bewertung der einzelnen Frameworks Kurze Absprache zur Aufbau der  Projektdokumentation  Erstellen der Präsentation |
| 20.11.2016 | Abschluss der Projektdokumentation und Zwischenpräsentation |

# Grundlegende Ansätze

Nach gründlicher Recherche ist das Projektteam zu dem Schluss gekommen, dass es keine Anbieter auf dem Markt gibt, welche ein zur Fragestellung der Projektarbeit passendes GUI anbieten. Daher hat sich das Projektteam mit dem Dozenten darauf verständigt, eine Eigenentwicklung der Software vorzunehmen. Gescoutet werden in diesem Falle nicht die Anbieter von Software bzw. Webservices, sondern verschiedene Technologien und Frameworks zur Umsetzung der definierten Anforderungen.

Betrachtet man die Ausgangslage so sind drei grundsätzliche Ansätze zur Umsetzung vorstellbar:

1. Hangouts in eine eigenentwickelte Cloud Anwendung/Homepage einbetten
2. Aufspaltung von Videotelefonie und erarbeiteten Vorschlägen in zwei separate Browserfenster
3. Entwicklung einer Hangouts-Applikation

Die nachfolgende Grafik stellt die drei Möglichkeiten gegenüber und enthält eine kurze Bewertung:

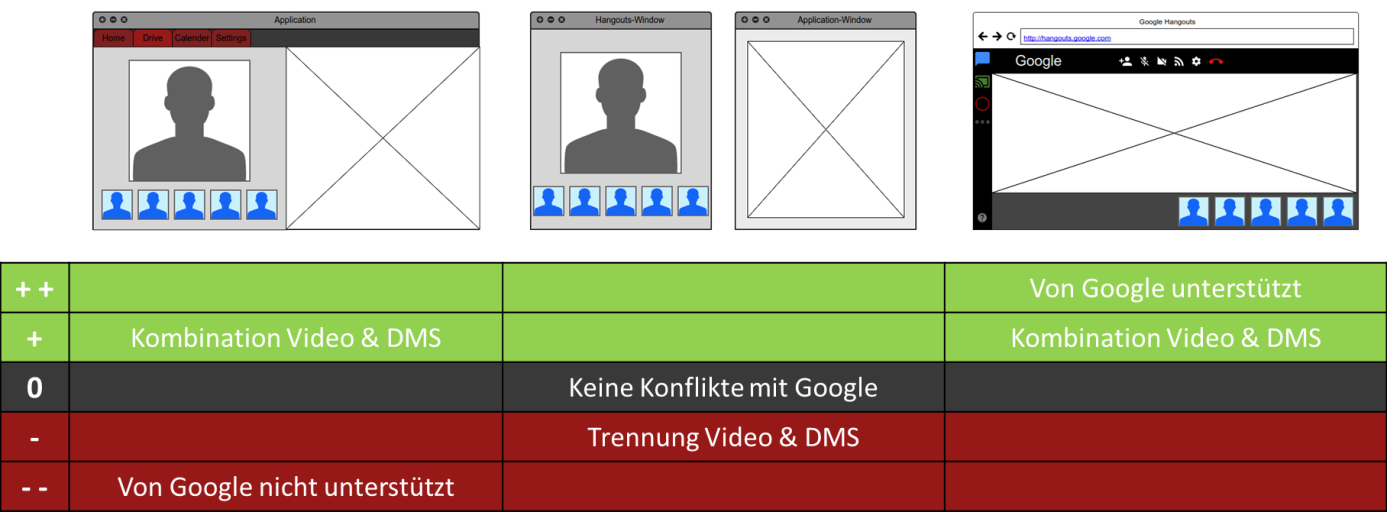


Abbildung 3: Möglichkeiten der Systemintegration

So hat eine Recherche und Nachfrage bei Google ergeben, dass eine Einbindung von Google Hangouts in eine eigenentwickelte App oder Software nicht möglich ist, da die benötigten Schnittstellen von Google nicht zur Verfügung gestellt werden. Google empfiehlt hier die Entwicklung einer Hangouts Applikation. Somit wurde Ansatz 1 auf Grund der fehlenden Unterstützung Googles, die auch am seit mehr als fünf Jahren bestehenden Feature Request ausgemacht werden kann[[7]](#footnote-7), nicht weiter verfolgt. [[8]](#footnote-8) [[9]](#footnote-9)

Eine Trennung von Hangouts auf der einen Seite und Sprachanalyse inklusive Dokumentenvorschlägen auf der anderen Seite ist insofern nicht ergonomisch, als dass zwei Browserfenster verwendet werden müssen, welche sich unter Umständen überlappen können und somit dem User ein einfaches Arbeiten erschweren würden. Dies wird gerade bei kleinen Bildschirmen ohne Möglichkeit der Erweiterung auf einen zweiten Bildschirm zum Problem. Da dieses Szenario im heutigen sehr mobilen Geschäftsleben des öfteren auftreten kann, hat sich das Team dagegen entschieden Ansatz 2 weiter zu verfolgen.

Bei der Evaluation des dritten Ansatzes zeigt sich, dass Google die Möglichkeit bietet eigenentwickelte Apps in Hangouts zu integrieren. Hierbei können zwei Arten von Hangouts-Applikationen unterschieden werden. So kann die Integration sowohl in Form einer klassischen App, die in der Mitte des Hangouts-Fensters dargestellt wird, als auch in Form einer sogenannten Erweiterung, die am rechten Seitenrand des Fensters angezeigt wird, erfolgen. Der größte Unterschied der beiden Applikationstypen ist die Größe der zur Verfügung stehenden Fläche.

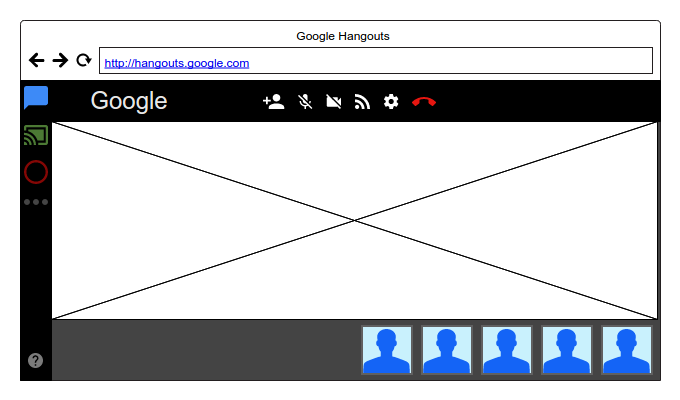


Abbildung 4: Google Hangouts mit eingebundener App

Während die klassische App die Großansicht des Gesprächspartner ersetzt und somit den größten Teil des Fensters zur Verfügung hat, wird die Erweiterung am rechten Fensterrand eingeschoben und verfügt somit über eine stark limitierte Fläche im Vergleich zur Alternative, zeigt allerdings nach wie vor die Großansicht des sprechenden Gesprächspartners.

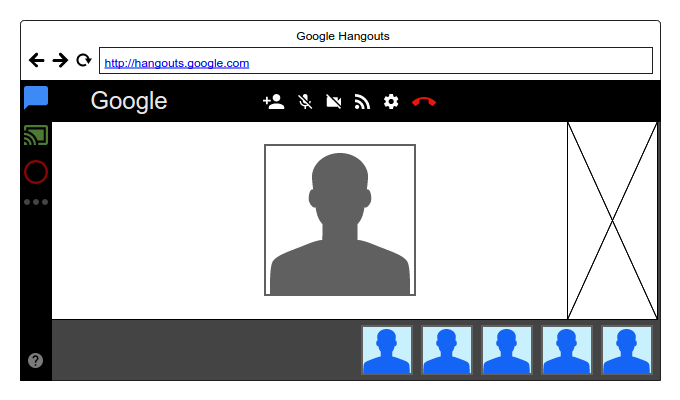


Abbildung 5: Google Hangouts mit eingebundener Erweiterung

Das Team präferiert in diesem Fall die Umsetzung als klassische Hangouts-Applikation, da für das zu betrachtende Kollaborationssystem der Darstellung der erarbeiteten Vorschläge Vorzug gegenüber der Großansicht des Gesprächspartners einzuräumen ist. Zudem sind die Gesprächspartner in beiden Darstellungsformen zu jedem Zeitpunkt am unteren Bildschirmrand sichtbar. Sollte dies nicht ausreichen, kann die Applikation temporär geschlossen werden, wodurch die Großansicht des Gesprächspartners gegeben wäre. Die Applikation kann jederzeit während einem Videotelefonat geöffnet und geschlossen werden, ohne dass dies einen Einfluss auf das Hangouts-Gespräch hätte.

Durch den beschriebenen dritten Ansatz könnte also die Darstellung von Videotelefonat und Applikationslogik in einem Fenster unter Einhaltung der Richtlinien von Google erreicht werden. Da dies exklusiv bei diesem Ansatz der Fall ist, entscheidet sich das Team für die Umsetzung der GUI als Hangouts-Applikation in klassischer Form.

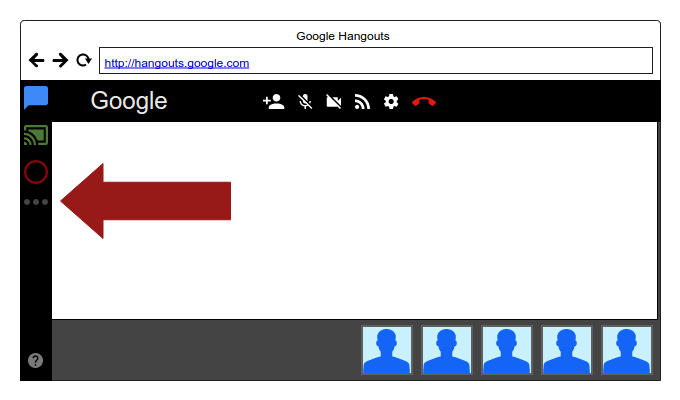


Abbildung 6: Google Hangouts "Problembutton"

Bei der weiteren Evaluation zur Verwendung einer Hangouts-Applikation ist das Team auf ein Problem gestoßen. So wurde die in Abbildung 6 markierte Applikationsleiste zwar bei jedem Gesprächsteilnehmer angezeigt, allerdings fehlten bei mehreren Mitgliedern das dargestellte Symbol mit den drei Punkten, das zur Einbindung einer Applikation notwendigen ist. Diese Komplikation konnte auch nicht durch den Neustart des Telefonats, den Wechsel des Browsers oder die Verwendung der expliziten Hangouts Erweiterung für Google Chrome behoben werden. Die Ursache des Problems konnte somit nicht ausgemacht und isoliert werden. Es gelang allerdings durch einen einfachen Workaround die Komplikationen zu beheben und die Funktionalität der Applikation zu gewährleisten. Es gelang die Erstellung eines Links, der Google Hangouts automatisch mit der gewünschten Applikation öffnet und somit neben der Behebung des Problems weitere Vorteile mit sich bringt. Durch die Einbindung dieses Links, der im Look & Feel von Google Hangouts gehalten ist, in eine Landingpage, kann das Frontend des Systems beliebigen Corporate Identities angepasst werden. Zudem wird dem Nutzer die Suche und Installation der Applikation abgenommen und hierdurch das Risiko der Installation einer falschen Applikation vermieden. Insgesamt kann also durch die Erweiterung des Ansatzes einer Hangout-Applikation durch eine vorgeschaltete Landingpage der gewünschte Prozess gewährleistet, optimiert und userfreundlich angeboten werden.

Es kann somit festgehalten werden, dass das GUI im Rahmen einer Hangouts-Applikation umgesetzt werden soll.

# Anforderungsanalyse

Zu Beginn der darauf aufbauenden Anforderungsanalyse definierte das Team die einzelnen Schritte und Aktionen des Tools während eines Videotelefonats. Hierfür wurden als Zwischenprodukte eine EPK sowie ein Wireframe mit integriertem Prozess erarbeitet, die als grafische Grundlagen für die Erfassung der Anforderungen dienen sollten. Beide Diagramme sind auf der Projektsite unter “3.5 Workflow User” einsehbar.

## Konstruktion Anforderungskatalog

Um die erfassenden Anforderungen geeignet dokumentieren zu können wurde in Google Tabellen die Schablone eines Anforderungskataloges eingerichtet. Google Tabellen wurde gewählt, da es lizenzfrei erhältlich ist, die gleichzeitige Bearbeitung der Anforderungen durch mehrere Autoren ermöglicht und durch die gegebenen Filtermechaniken eine einfache Verwaltung der Anforderungen an Hand deren Attribute bietet. Bei der Wahl der zu verwendeten Attribute hat sich das Team dazu entschieden, sich auf das mindeste zu beschränken, um zusätzlichen Pflegeaufwand und das Risiko von Redundanzen und veralteten Daten zu minimieren. Zudem wurden Regeln erarbeitet, die eine einheitliche Dokumentation der Anforderungen fördern und somit die Validität des Anforderungskataloges gewährleisten sollen. Außerdem wurde ein Review-Prozess eingerichtet, in dem zwei Studierende, die aus ihrem Bachelorstudium bereits Erfahrung mit der Anforderungsanalyse haben, nach dem Vier-Augen-Prinzip die neuen Anforderungen überprüfen und anschließend freigeben. Eine Erklärung der Attribute, die Liste der Regeln sowie eine grafische Darstellung des Review-Prozesses können auf der Projektsite unter “4.1 Funktionale Anforderungen” im Leitfaden eingesehen werden.

## Anforderungskatalog

Der gesamte Anforderungskatalog wird aus Gründen der Übersichtlichkeit separat auf der Projektwebsite als Excel-Datei zur Verfügung gestellt. Der Katalog dient als strukturierte Sammlung sämtlicher im Zuge der Projektarbeit erfassten Anforderungen.

Nach aktuellem Stand werden folgende Anforderungen aufgrund des Projektauftrages oder sonstigen Rahmenbedingungen und Zusammenhängen als “kritisch” betrachtet:

|  |  |
| --- | --- |
| Eclipse | Das System muss in Eclipse entwickelt werden können |
| Java | Das System muss auf Java bzw. Java Script, CSS und HTML basieren |
| Grafische Darstellung | Das System muss eine grafische Darstellung anbieten |
| Dateienauswahl | Das System muss dem Benutzer erlauben konkrete Dateien auszuwählen |
| Dokumentenübersicht | Das System muss eine Gesamtübersicht der Dokumente anzeigen können |
| Datenweiterleitung | Das System muss die Nutzerdaten beim Wechsel in Drive weiterleiten |
| Schließen | Das System muss vom Benutzer geschlossen werden können |
| Google Docs | Das System muss einzelne Dateien in Google Docs öffnen können |
| Google Tabellen | Das System muss einzelne Dateien in Google Tabellen öffnen können |
| Google Kalender | Das System muss einzelne Daten in Google Kalender öffnen können |

Diese kritischen Faktoren waren Grundlage für den weiteren Projektverlauf, da diese auf jeden Fall erreicht werden müssen.

# Visualisierungskonzepte

Da nicht nur der gebotene Rahmen der Visualisierung sondern vielmehr die Form der Visualisierung selbst eine entscheidende Rolle im Prozess des Technologie-Scoutings spielt, sollen im folgenden Kapitel mehrere vom Team erarbeitete Visualisierungskonzepte vorgestellt werden. Diese werden eigentlich erst in der zweiten Phase des Projektes unterschieden, sind aber von immenser Bedeutung für die zu verwendeten Frameworks. So müssen die Frameworks bereits jetzt unter Beachtung der potentiellen Visualisierungsmethoden betrachtet werden, da gewährleistet sein muss, dass diese vom Framework unterstützt bzw. ermöglicht werden.

Das Team hat im Ausblick auf die zweite Phase vier Visualisierungskonzepte erarbeitet, die als Grundlage für das Technologie-Scouting dienen sollen. Im Folgenden werden diese Konzepte benannt und kurz beschrieben. Auf eine Wertung wird an dieser Stelle verzichtet, da diese für den weiteren Fortschritt in Phase 1 keine Relevanz besitzt.

Grundlegend festzuhalten hierbei ist, dass bei den folgenden Skizzen der schwarze Rahmen jeweils das Sichtfeld des Users symbolisiert. Der Nutzer sieht somit in keiner Darstellungsform die Gesamtmenge aller potentieller Vorschläge, da dies eine mit der Datenmenge skalierende Unübersichtlichkeit mit sich bringen würde. Es ist allerdings angedacht, dass über eine entsprechende Schaltfläche sämtliche Dateien angezeigt werden, um dem User eine proaktive Auswahl zu ermöglichen. Dies ist beispielsweise denkbar, wenn der Nutzer auf ein im bisherigen Gespräch noch nicht genanntes Dokument ansprechen möchte, sich zuvor jedoch nochmal über dessen Inhalt versichern möchte.

## Visualisierungskonzept I - Konventionell

Die konventionelle Visualisierung ist an die Dokumentendarstellung in Google Drive angelehnt. Einzelne Dokumente werden als Kästchen dargestellt, der Dokumentenname und Typ stehen darunter. Erlangt eine Datei im Gespräch Relevanz wird diese dynamisch in die Anzeige nachgeladen. Wird eine Datei nicht mehr als relevant betrachtet, wird sie aus der Anzeige entfernt.

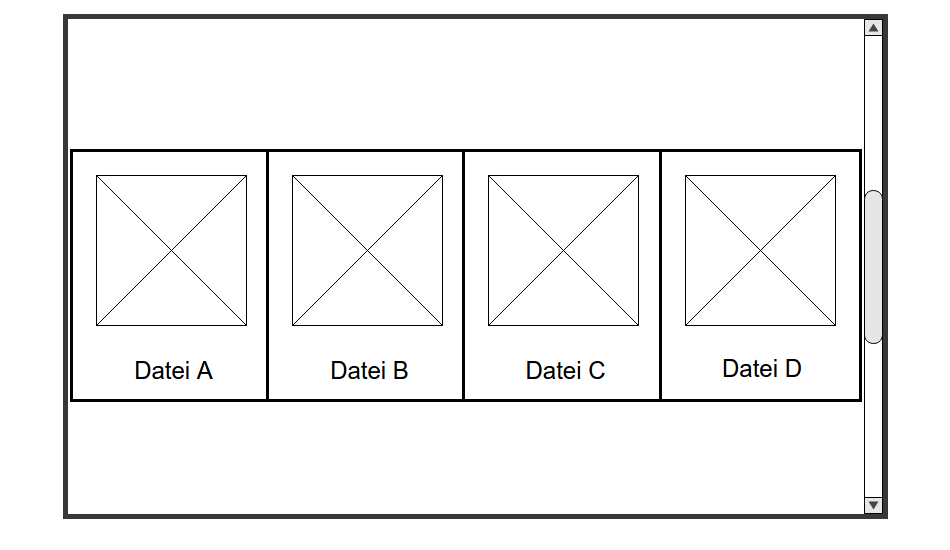


Abbildung 7: Visualisierungskonzept I - Konventionell

## Visualisierungskonzept II - MindMap

Das zweite Visualisierungskonzept arbeitet wie eine MindMap mit einer Baumstruktur, wobei die Dateiordner die Knotenpunkte darstellen. So wird zunächst der Projektordner mit seinen Unterordnern dargestellt, bis das Gespräch sich einem speziellen Bereich zuwendet. Erkennt das System, dass beispielsweise über Dokumente aus einem etwaigen Ordner “Protokolle” gesprochen wird, verändert sich der Fokus und die nächste Ebene des Protokoll-Ordners werden angezeigt, wohingegen die restlichen Ordner der ursprünglichen Ebene aus dem Sichtfeld verschwinden.

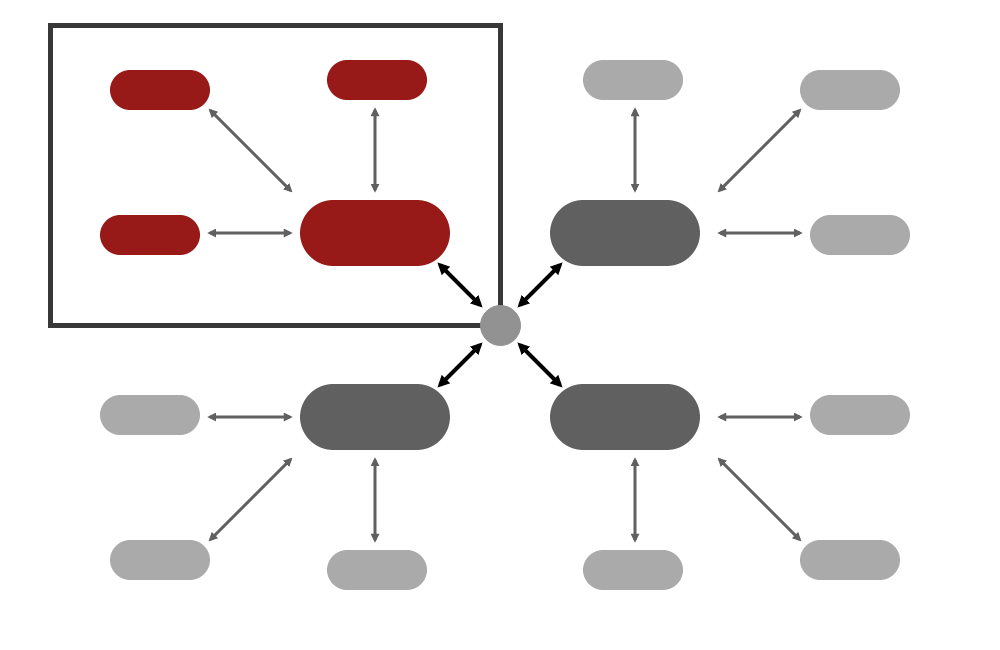


Abbildung 8: Visualisierungskonzept II - MindMap

## Visualisierungskonzept III - Pyramide

Die Pyramide ist ein hierarchisches System mit visuellem Fokus auf die Dokumentenrelevanz. Je relevanter ein Dokument auf Basis des Gesprächs erscheint, desto höher rückt es an die Spitze der Pyramide. Im Gegenzug steigen Dokumente mit geringerer Relevanz wieder ab oder verschwinden zuletzt ganz in den “Dokumentenpool”, welcher für den Nutzer nicht sichtbar ist.

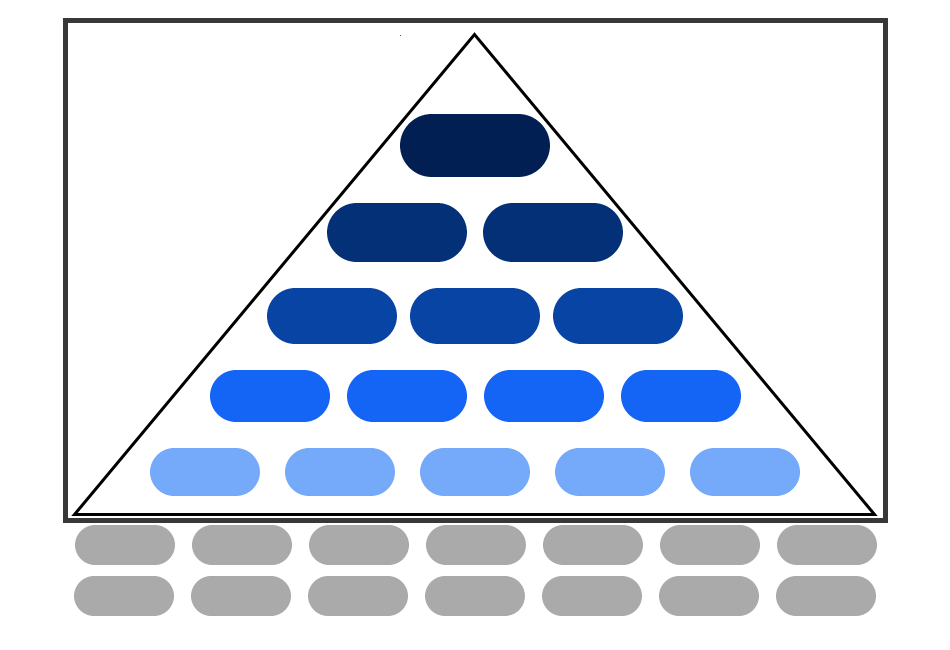


Abbildung 9: Visualisierungskonzept III - Pyramide

## Visualisierungskonzept IV - Litfaßsäule

Das vierte Visualisierungskonzept ist grafisch an einen sich drehenden Zylinder angelehnt. Dabei wird ein Ordner - vergleichbar mit einem Plakat an einer Litfaßsäule - auf der Außenseite des Zylinders aufgelistet. Ändert sich der Fokus des Gesprächs, dreht sich der Zylinder zum entsprechenden Ordner bzw. Plakat, welches von nun an im Sichtfeld des Nutzers angezeigt wird.

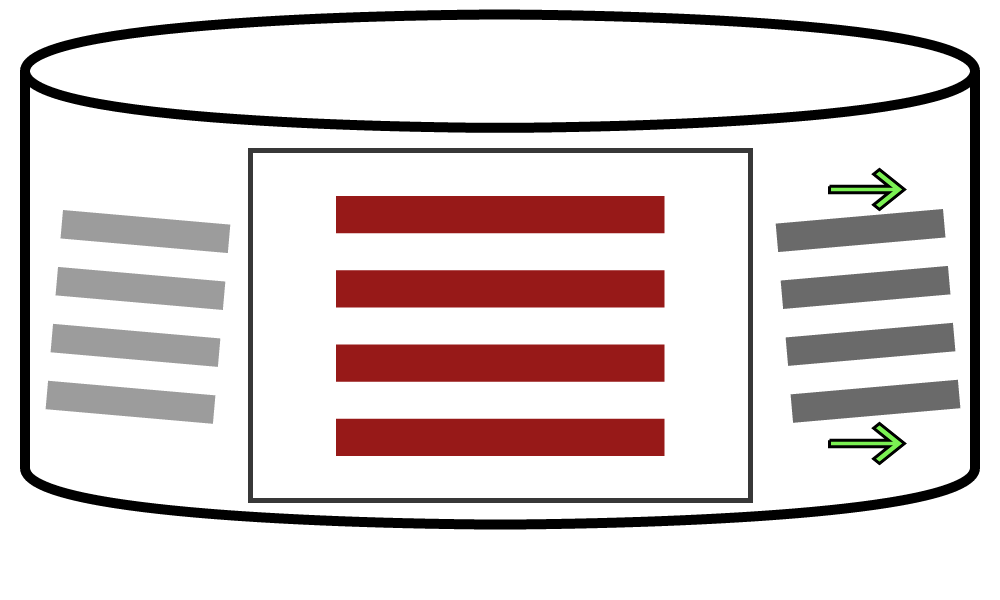


Abbildung 10: Visualisierungskonzept IV - Litfaßsäule

# Technologie Scouting

Unter Berücksichtigung der bis hierher ermittelten Rahmenbedingungen, also der Umsetzung als Hangouts-Applikation sowie den angedachten Visualisierungsformen, fand das anschließende, eigentliche Technologie Scouting statt.

Hervorzuheben ist hierbei, dass sich das Team darauf verständigt hat auf zwei Arten von Frameworks zurückzugreifen.   
Zum einen soll ein Framework als Unterstützung bei der Visualisierung der Daten eingesetzt werden. Das vom Team sogenannte “Visualisierungs-Framework” soll die für die Visualisierungskonzepte be­nötigten Animationen zur Verfügung stellen, damit bei diesen Animationen auf bewährte Methodik zurückgegriffen werden kann. Zum anderen soll ein “Code-Frameworks” benutzt werden, um große Teile der Programmierung einer Web-Applikation auf bereits bestehende Fragmente auslagern zu können.

Bei der Sichtung der zur Verfügung stehenden Frameworks wurde eine Vorauswahl hinsichtlich der Verbreitung und Erprobtheit der Tools getroffen. Da es sich bei der zu entwickelnden Lösung um ein Businessprodukt handeln soll, hat es oberste Priorität, dass ein reibungsloser Betrieb gewährleistet werden kann. Zudem erhoffen wir durch die Verwendung verbreiteter Bibliotheken, dass ein Support der Lösung durch die Bekanntheit und Dokumentation der Frameworks einfacher von statten geht und somit personenunabhängig gewährleistet werden kann.

Zur Evaluation der einzelnen Frameworks wurde ein Kriterienkatalog erstellt, der auf der Projektsite unter “4.2 Auswahl Frameworks” eingesehen werden kann.

## Auswahl der Visualisierungs-Frameworks

Bei der Recherche nach geeigneten Visualisierungs-Frameworks zeigte sich, dass die meisten verfügbaren Frameworks nicht auf die Darstellung von Dateien sowie komplexeren Datenstrukturen abzielen. Stattdessen sind sie vielmehr dafür ausgelegt, Daten in bestimmten Zusammenhängen zu veranschaulichen. Gewünschte Anwendungsfelder wären hier etwa Diagramme, Karten oder Hilfsanimationen.

Bei den verbliebenen Frameworks kristallisierten sich im folgenden zwei Varianten heraus.   
Zum einen könnten MindMap-artige Visualisierungen, wie etwa die Frameworks “arbor.js” oder aber “cola.js”, zur Darstellung von Dateibäumen verwendet werden. Eine alternative Variante bildet die Modifizierung bestimmter Diagramm-Frameworks. Hierbei könnten etwa Googles “Bubble Chart”[[10]](#footnote-10) dahingehend eingesetzt werden, dass die Relevanz der einzelnen Dateien über die Größe der Blasen und der Dokumententyp über deren Färbung ausgedrückt werden könnte.

Der schlussendlichst anpassungsfähigste und freieste Weg eine individuelle und innovative Visualisierung zu erreichen, wäre es mittels dynamischem HTML (DHTML) und JavaScript eine eigene Visualisierungskomponente zu entwerfen. Durch den Charakter der Eigenkonstruktion könnte diese in Funktionalität, Form und Umfang genau auf das Projekt und seine Bedürfnisse angepasst werden. Diesem Vorteil steht allerdings gegenüber, dass dies die mit Abstand aufwändigste Realisierungsvariante wäre. Zudem ist durch die fehlende Erprobtheit nicht sichergestellt, dass das gewünschte Ergebnis erreicht werden kann.

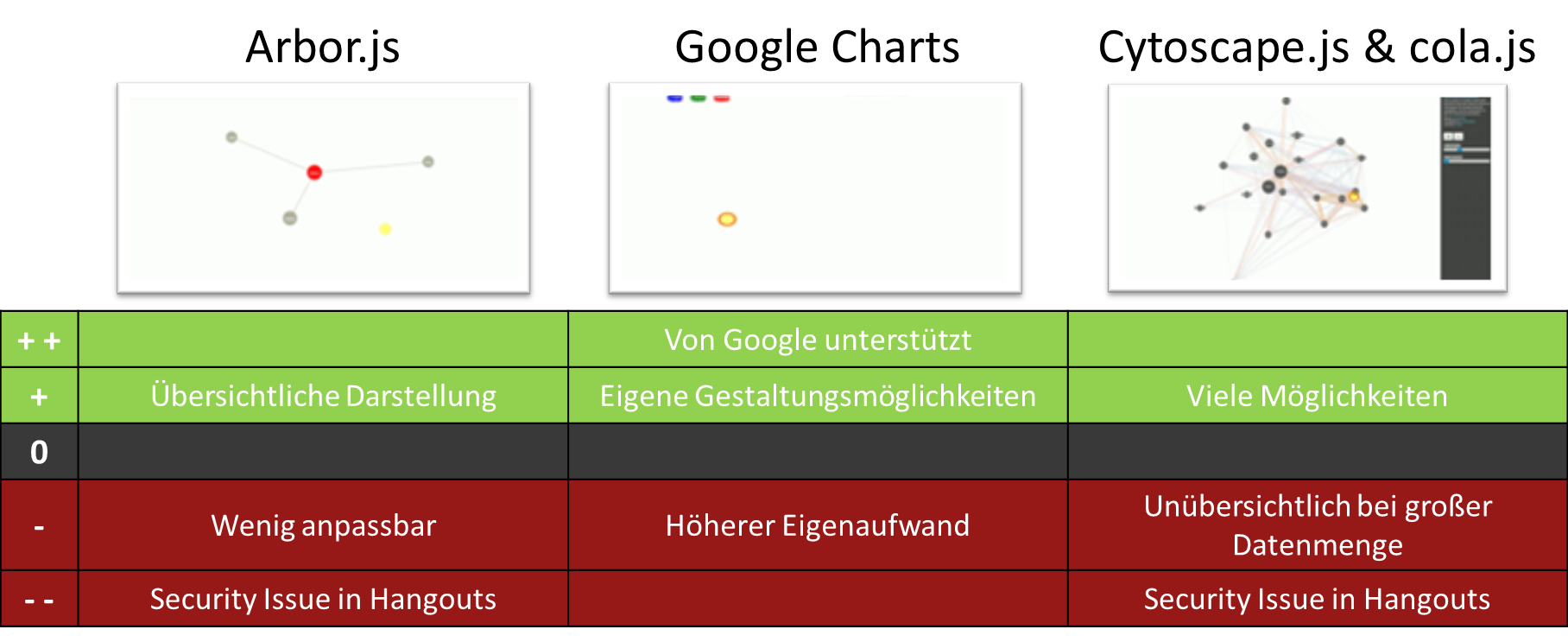


Abbildung 11: Vergleich Visualisierungs-Frameworks

Bei einer abschließenden Evaluierung wurden die am besten geeigneten MindMap- und Diagramm-Frameworks sowie die potentielle Eigenentwicklung anhand des Kriterienkatalogs getestet und auf deren Umsetzbarkeit geprüft. Eines der am vielseitigsten einsetzbaren Frameworks war Cola.js in Verbindung mit Cytoscape.js. Diese Visualisierung wird bei hohem Dateibestand jedoch schnell unübersichtlich. Arbor.js schafft es durch selbsteinklappende Unterpunkte auch bei großen Datei-Bäumen eine übersichtliche Darstellung zu erhalten. Beide Frameworks werden jedoch durch die Security Richtlinien von Google nicht für Hangouts unterstützt. Die dritte Möglichkeit bezieht sich darauf entweder ein Google Bubble Chart zu animieren oder eine eigene Visualisierung zu programmieren. Dies führt zu keinerlei Security Problemen und wird von Google unterstützt, bedeutet jedoch einen deutlichen programmiertechnischen Mehraufwand.

## Auswahl der Code-Frameworks

Die Code-Frameworks sollten neben ihrer allgemeinen Programmier Erleichterung vor allem dafür sorgen, dass eine Serververbindung durch AJAX problemlos zustande kommt und dabei helfen Webseiten oder Apps in verschiedenen Browsern gleich darzustellen.

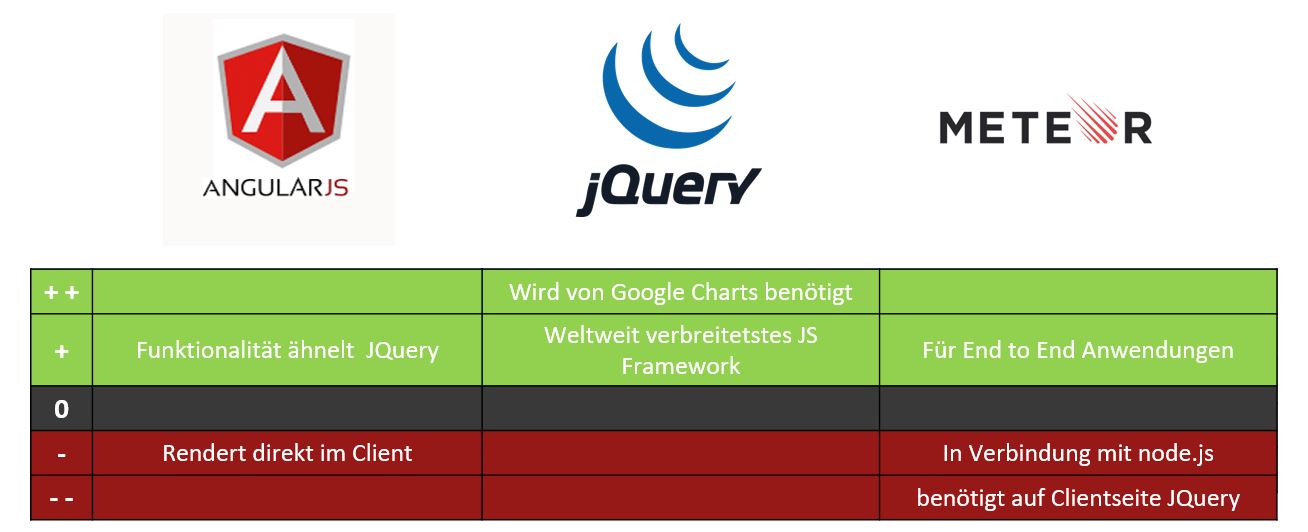


Abbildung 12: Vergleich Code-Frameworks

Drei der beliebtesten Frameworks sind derzeit jQuery, AngularJS und Meteor. Alle zeichnen sich durch eine ausgezeichnete Usability sowie eine weite Verbreitung aus. Bei der detaillierten Betrachtung mit Blick auf die gewünschte Funktionalität stellte sich Meteor als das am wenigsten geeignete Framework heraus. Dies liegt darin begründet, dass Meteor auf dem serverseitigen Node.JS basiert und clientseitig jQuery voraussetzt. Angular.js bietet zwar die meisten und vielseitigsten Möglichkeiten, ist jedoch aus mehreren Gründen für eine animierte Hangouts App ungeeignet. Im Gegensatz zu jQuery manipuliert Angular.js nicht den HTML Dom, sondern rendert clientseitig, ohne Serverbeteiligung. Dies kann bei schwachen Client-Systemen, die schon durch eine laufende Hangouts Verbindung sowie durch eine Animation ausgelastet sind, zu Problemen führen. jQuery gilt als weltweit verbreitetes JavaScript Framework und ist explizit für Animationen geeignet. Wie bereits erwähnt, manipuliert jQuery clientseitig lediglich den vom Server empfangenen DOM des HTML Dokuments und ist somit sehr performant. Erschwerend hinzukommt, dass jQuery vom gewählten Visualsierungs-Framework Google Charts als Grundvoraussetzung eingefordert wird. Somit wird jQuery als Code-Framework ausgewählt.

# Auswahl der Programmiermethodik

Vom Projektteam wurden drei Möglichkeiten für die programmiertechnische Umsetzung ermittelt.

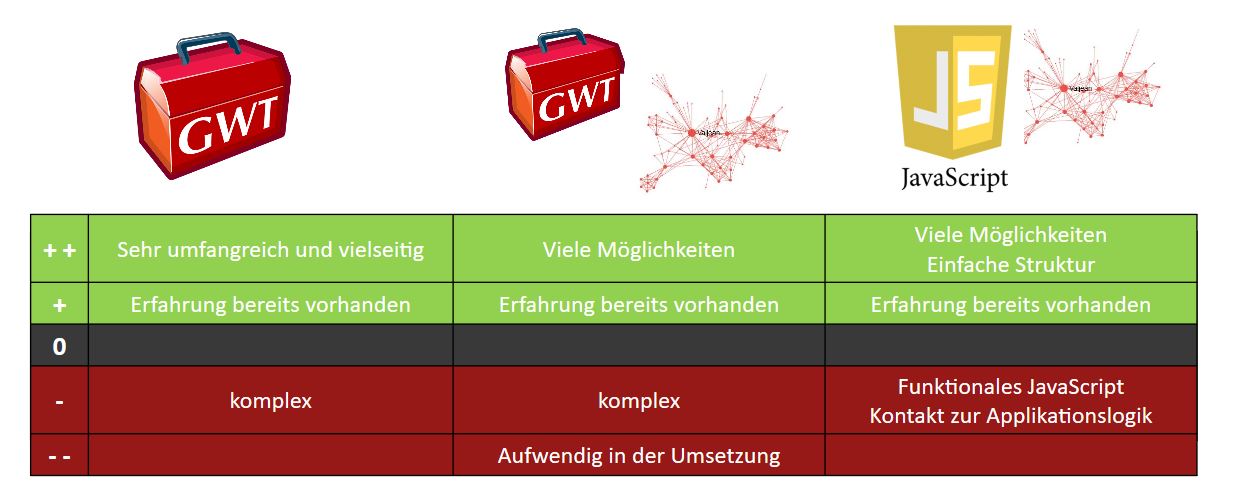
1. Die Beschränkung auf die Verwendung des Google Web Toolkits (GWT)[[11]](#footnote-11) zur Programmierung in Java und anschließenden Kompilierung in JavaScript[[12]](#footnote-12) sowie Verwendung des enthaltenen GUI-Frameworks
2. GWT als Java zu JavaScript Compiler, aber Verwendung eines zusätzlichen Visualisierungs-Frameworks
3. Natives JavaScript unter Verwendung eines Code- sowie eines Visualisierungs-Frameworks

Abbildung 13: Vergleich Programmiermethodik

Bei der reinen Nutzung von GWT würde die Anwendung vollständig in Java Code entwickelt. Das bedeutet, dass ebenfalls die Clientseitige Kommunikation in Java entwickelt wird. Zusätzlich stellt das Web Toolkit ein eigenes Framework für die Entwicklung von Grafikoberflächen. Vorteil dieser Möglichkeit ist es, dass auf die Funktionen auf Serverseite einfach zugegriffen werden kann. Die GUI Visualisierung ist jedoch komplexer.

Im Falle der Nutzung von GWT und einem zusätzlichen Visualisierungs-Frameworks würde der Source Code ebenfalls in Java entwickelt werden. Man kann somit die Vorteile von GWT und dem gewählten Framework verbinden. Nachteil ist, dass dadurch aber auch die Programmierung aufwendiger wird. Wird die Präsentationsschicht unter Verwendung eines Visualisierungs-Frameworks ausschließlich in JavaScript entwickelt, handelt es sich um keine objektorientierte Programmierung. Dadurch sinkt die Komplexität der Entwicklung, ohne dass funktionale Einschränkungen hinzunehmen wären.

Auf Grund der geringen Komplexität sowie der Gegebenheit, dass im Team bereits JavaScript-Erfahrung vorhanden ist, wurde die Entscheidung getroffen, dass System ohne GWT zu entwickeln. Somit wird eine Entwicklung mit nativen JavaScript sowie zwei verwendeter Frameworks stattfinden.

## Entwicklungsumgebung

Nachfolgend sollen noch kurz die Anforderungen des Dozenten zur Entwicklungsumgebung dargelegt werden. Vorgegeben ist die Nutzung von Java APIs – sofern diese vorhanden sind. Zur Erstellung von Source Code ist die Entwicklungsumgebung Eclipse (mit zugehörigem Papyrus-Plug-in für die UML Diagramme) zu verwenden. Der Source-Code ist den Konventionen der jeweiligen Programmiersprache entsprechend zu erstellen. Dabei ist auf eine saubere Dokumentation aller APIs und Frameworks zu achten. Zudem soll GIT als Werkzeug zur Versionsverwaltung in Form eines Plug-ins in Eclipse integriert werden. Für die Entwicklung wird Eclipse Juno in der neuesten Version verendet.

# Zwischenfazit

In dieser ersten Projektphase galt es sich mit den aktuellen Technologien zur Visualisierung einer Systemoberfläche auseinander zu setzen und durch ein Scouting diese Technologien mit den Anforderungen, die im Rahmen des Projektes an das System gestellt werden, abzugleichen. Dabei erlangte das Projektteam folgende Erkenntnisse: es ist mit Google Hangouts nicht möglich einen Hangouts Videochat in eine eigene Webanwendung zu integrieren. Aus diesem Grund ist es nötig den umgekehrten aber von Google unterstützen Weg zu gehen und eine eigene Webapp in Hangouts zu integrieren. Da es offensichtlich mehrere Hangouts Versionen gibt, welche nicht alle die Integration von Apps unterstützen, ist es weiterhin notwendig die richtige Version mit Hilfe einer Landingpage zu verlinken. Eine in Hangouts lauffähige App besteht aus den drei Technologien HTML, CSS und JavaScript. Durch das JavaScript Framework jQuery ist es außerdem möglich für Animationen geeignete Methoden in einer Hangouts App zu verwenden. Zur Visualisierung der vorgeschlagenen Dateien gibt es mehrere Ansätze, diese reichen von einer konventionellen Ordnerstruktur über eine MindMap oder Litfaßsäule bis hin zu einem Blasendiagramm. Bei der Visualisierung kommt es sowohl auf eine ansprechende Präsentation der vorgeschlagenen Dateien, als auch auf eine Feedbackfunktion für unpassende Vorschläge an. Dies ist unbedingt notwendig, um dem Gesamtsystem Feedback für die Lernfunktion zu geben. Da es für die gewünschte Darstellung keinerlei fertige Frameworks gibt, kommt das Team zu dem Schluss, dass eine selbst geschrieben Anwendung die beste Methode ist die gegebenen Projektziele zu erreichen. Diese selbst entworfene Anwendung soll optisch an ein Google Bubble Chart erinnern und durch eine Größenänderung der entsprechenden Blase Wichtigkeit symbolisieren. Für unterschiedliche Dateitypen ist es somit nicht nur möglich verschiedenen Farben, sondern auch unterschiedliche Symbole oder Formen zu verwenden.

1. Vgl. http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app [↑](#footnote-ref-1)
2. Vgl. http://searchitoperations.techtarget.com/definition/cloud-application [↑](#footnote-ref-2)
3. Als **mandantenfähig** (auch mandantentauglich) wird Informationstechnik bezeichnet, die auf demselben Server oder demselben Software-System mehrere Mandanten, also Kunden oder Auftraggeber, bedienen kann, ohne dass diese gegenseitigen Einblick in ihre Daten, Benutzerverwaltung und Ähnliches haben [↑](#footnote-ref-3)
4. Vgl. http://www.techrepublic.com/blog/the-enterprise-cloud/cloud-app-vs-web-app-understanding-the-differences/ [↑](#footnote-ref-4)
5. ISO 9241-210 [↑](#footnote-ref-5)
6. Vgl. Hassenzahl, Burmester und Koller (2008) - “Der User Experience auf der Spur” [↑](#footnote-ref-6)
7. <https://code.google.com/p/google-plus-platform/issues/detail?id=28> [↑](#footnote-ref-7)
8. <https://groups.google.com/forum/#!topic/google-plus-developers/nZ7vC3zJOx8> [↑](#footnote-ref-8)
9. <http://www.allesuebergoogleplus.de/allgemein/neue-moeglichkeiten-der-hangout-nutzung-hangout-api-fuer-entwickler/> [↑](#footnote-ref-9)
10. <https://developers.google.com/chart/interactive/docs/gallery/bubblechart> [↑](#footnote-ref-10)
11. Bei GWT handelt es sich um einen Werkzeugsatz zur Entwicklung von Webanwendungen. Das Google Web Toolkit ist ein Java-nach-JavaScript-Compiler, so dass nahezu die gesamte Entwicklung von Client und Server mit Java realisiert werden kann. Das GWT beinhaltet einen XML-Parser sowie eine Schnittstelle für Remote Procedure Calls. Zudem ist ein Widget-Paket zur Gestaltung der grafischen Oberfläche (GUI) enthalten. [↑](#footnote-ref-11)
12. JavaScript ist eine Skriptsprache, die ursprünglich für dynamisches HTML in Webbrowsern entwickelt wurde. Sie wird verwendet, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern. [↑](#footnote-ref-12)