

Mobile Collaboration App

Vordiplomarbeit - HF Uster

Version [0.0]

[Date]

Presented by: Christian B. Sax

Dorfstrasse 11, CH 9545 Wängi

PlexByte.com

Inhalt

[1 Zusammenfassung / Management-Summary 3](#_Toc447379731)

[1.1 Inhalt der Zusammenfassung / Management Summary 3](#_Toc447379732)

[1.1.1 Kurzbeschreibung des Projekts, Projektidee 3](#_Toc447379733)

[1.1.2 Schwierigkeiten 3](#_Toc447379734)

[1.1.3 Lösung ganz grob 3](#_Toc447379735)

[1.1.4 Rückblick / Erkenntnisse 3](#_Toc447379736)

[1.1.5 Zeitaufwand (soll-Ist Vergleich) 3](#_Toc447379737)

[1.1.6 Ausblick 4](#_Toc447379738)

[2 Einleitung 4](#_Toc447379739)

[2.1 Ausgangslage /Motivation 4](#_Toc447379740)

[2.2 Aufgabenstellung 4](#_Toc447379741)

[2.3 Entscheidungs Matrix (Technologie) 5](#_Toc447379742)

[2.4 Vorgehen / Lösungsansatz 5](#_Toc447379743)

[2.4.1 Technology Evaluation 5](#_Toc447379744)

[3 Projektplanung 7](#_Toc447379745)

[4 Pflichtenheft 7](#_Toc447379746)

[4.1 Mögliche Struktur 7](#_Toc447379747)

[4.1.1 Zweck 7](#_Toc447379748)

[4.1.2 Geltungsbereich 7](#_Toc447379749)

[4.1.3 Referenzierte Dokumente 7](#_Toc447379750)

[4.1.4 Situationsübersicht 7](#_Toc447379751)

[4.1.5 Zielsetzung 7](#_Toc447379752)

[4.1.6 Schnittstellen 8](#_Toc447379753)

[4.2 Abgrenzung der Aufgabe 8](#_Toc447379754)

[4.3 Abweichungen und Korrekturen 8](#_Toc447379755)

[5 Analyse / Vorstudie 8](#_Toc447379756)

[5.1 Zweck und Umfang der Vorstudie / Analyse 8](#_Toc447379757)

[5.2 Zielsetzungen 8](#_Toc447379758)

[5.3 Analyse von 8](#_Toc447379759)

[6 Konzept-Varianten / Hauptstudie 8](#_Toc447379760)

[6.1 Zweck und Umfang der Hauptstudie 8](#_Toc447379761)

[6.2 Konzeptvarianten 8](#_Toc447379762)

[6.2.1 Beschreibung der Konzeptvarianten 8](#_Toc447379763)

[6.2.2 Beurteilung der Konzeptvarianten 8](#_Toc447379764)

[6.2.3 Auswahl der Konzeptvarianten 8](#_Toc447379765)

[7 realisation / Umsetzung 9](#_Toc447379766)

[7.1 Zweck und Umfang der Realisation 9](#_Toc447379767)

[7.2 Realisation gemäss Konzept im Top-Down Verfahren 9](#_Toc447379768)

[7.3 Realisation von SW 9](#_Toc447379769)

[7.3.1 Project 9](#_Toc447379770)

[7.3.2 Accounting 10](#_Toc447379771)

[8 Test 11](#_Toc447379772)

[9 Dokumentation 11](#_Toc447379773)

[10 Glossar und Verzeichniss 11](#_Toc447379774)

[11 Beilagen 11](#_Toc447379775)

[11.1 Test the plan 12](#_Toc447379776)

[11.1.1 Consumer perceptions 12](#_Toc447379777)

[11.1.2 Quantitative metrics 12](#_Toc447379778)

[11.2 Adjust the plan 12](#_Toc447379779)

[11.3 Put the plan into action 13](#_Toc447379780)

# Zusammenfassung / Management-Summary

## Inhalt der Zusammenfassung / Management Summary

### Kurzbeschreibung des Projekts, Projektidee

Die Idee war ein Tool zu entwickeln bei dem man einfach Arbeiten in einer selbst zusammengestellten Gruppe erstellen und gemeinsam erledigen kann.

Die Lösung soll das Planen von Events, Firmenprojekten, Haushaltseinkäufe, für private als auch geschäftliche Zwecke ermögliche. Ebenso die Möglichkeit, Ausgaben finanzieller als auch zeitlicher Art zu erfassen. Dadurch soll Transparenz zur Einhaltung von Budgets oder Zeitaufwände geschaffen werden.

Idealerweise wird die Lösung als portable Applikation entwickelt, um jederzeit und überall zugreifen zu können.

Die Lösung wird MoCap bezeichnet, was für «Mobile Collaboration Application» steht. Sie ermöglicht es, Projekte, Tasks, Survey, Expenses und Timeslices zu erfassen und diese entsprechend einem Projekt zuzuordnen. Anschliessend können die Objekte von den Projektmitgliedern bearbeitet und verwaltet werden.

### Schwierigkeiten

* Teamarbeit  
  Das Arbeiten im Team erfordert das Definieren von Schnittstellen, abstrahieren von Komponenten und erhöht dadurch die Komplexität ungemein. Auch das Einhalten von Deadlines ist essentiell, da andere Teammitglieder ansonsten nicht weiterarbeiten können.

Das Planen, Abgleichen und Synchronisieren der Arbeitsschritte bringt zudem einen gewissen Overhead mit sich, der nicht zu unterschätzen ist.

Letztlich ist auch der unterschiedliche Wissensstand und Erfahrungen im Bereich Softwareentwicklung von Bedeutung um die Teilnehmer nicht zu überfordern

* Architektur  
  Die Client Server Architektur birgt ebenfalls eine gewisse Herausforderung, da Kommunikation zwischen mehreren Endpunkten und Komponenten sauber abgeglichen und vereinbart werden muss
* Aufwandsabschätzung  
  Oft wird der Umfang an Arbeit unterschätzt. Je komplexer eine Applikation wird umso umfangreicher werden die einzelnen Komponenten und deren Schnittstellen.

### High Level Lösungsübersicht

Die Lösung basiert auf einer lokalen Applikation, welche sich via Internet mit der MoCap Datanbank verbindet und Daten überträgt. Sobald eine Änderung an einem Objekt gemacht wurde, wird dieses in der Datenbank ebenfalls, automatisiert im Hintergrund, entsprechend aktualisiert.

Jedes Objekt welche zu einem späteren Zeitpunkt geöffnet wird, gleicht die Applikation mit der Datenbank ab. Somit ist sichergestellt, dass der Benutzer immer den aktuellsten Stand hat.



Die lokale Applikation wiederum gliedert sich grob in folgende Komponenten, welche im Kapitel Realisation genauer beschrieben werden.



Wie auf der Abbildung zu erkennen, bildet das «Interactions» Packet das Herz der Lösung ab, welches die einzelnen Objekte, sogenannte Interaktionen, definieren und implementieren.

Sämtliche Komponenten laufen zentral in den Manager zusammen und werden von diesen auch gesteuert. Sie bilden mitunter die Businesslogik, durch verketten diverser technische Funktionen, ab.

Das Backend regelt die Anbindung an den SQL Server und kapselt diesen komplett. Die zur Verfügung gestellten Funktionen liefern oder verarbeiten jeweils Objekte aus dem «Interactions» Packet.

Schliesslich bildet das «Forms» Packet alle Sichten, welche der User verwendet und damit die Interaktionen ver-/bearbeitet.

Um die Verwendung der Applikation so einfach wie möglich zu gestalten wurde auf eine Setuproutine verzichtet. Somit kann die Lösung durch starten der ausführbaren Datei MoCap.exe verwendet werden.

### Rückblick / Erkenntnisse

Aufgrund der gewählten Architektur und dem Bestreben möglichst abstrahierte und generische Ansätze zu verwenden entstanden sehr komplexe Konstrukte. Das führte schliesslich zur Überforderung des Teammitgliedes und der Überlastung des anderen.

Auch wurde der Aufwand als auch die Aufgabe an sich unterschätzt. Schwierigkeiten in der Kommunikation, unterschiedliches Engagement und verzögerte Deliverables verschärften die Problematik.

Die daraus gezogene Erkenntnis ist, dass die Architektur so gewählt werden sollte, dass sie unterschiedliche Fähigkeiten und Erfahrungen der Teammitglieder berücksichtigt und die Einhaltung von Terminen fokussierter verfolgt werden muss.

### Zeitaufwand (soll-Ist Vergleich)

|  |  |  |
| --- | --- | --- |
| **Module** | **Must Duration** | **Optional** |
| Balance | 42 | 42 |
| Chat | 38 | 50 |
| Communication | 46 | 46 |
| Logging | 18 | 18 |
| Persistance Layer | 62 | 68 |
| Poll | 28 | 38 |
| Project | 20 | 25 |
| Reporting | 0 | 24 |
| Security | 0 | 6 |
| Task | 27 | 39 |
| Misc | 34 | 34 |
| **Total** | **315** | **390** |

# Einleitung

## Ausgangslage /Motivation

Bei der Planung von Events ist es so, dass alle Beteiligten einen gemeinsamen freien Termin finden müssen um das Event zu Planen.

Die Schwierigkeiten kommen nach dem Treffen. Meistens werden Sachen vergessen oder nicht genau Abgemacht. Ausserdem werden meistens die Ausgaben nicht genau festgehalten.

Bei einer Hand voll guter Freunde lässt sich das noch relativ gut regeln, aber wenn die Gruppe aus 10 – 20 Personen besteht wird das schon schwieriger, da sich viele untereinander schlecht bis gar nicht kennen.

* Effizienz  
  Durch Software liesse sich der Aufwand senken und im Falle einer mobilen Anwendung (web basiert) flexibler gestalten. So können beteiligte in einem Gruppenchat verlinkt und die Aufgaben verteilt werden. Zusätzlich lassen sich die Aufwände, welche durch eine bestimmte Aufgabe entstehen erfassen und dem Projekt zuweisen.
* Einfache Bedienung  
  Wir wollen ein leicht zu bedienendes Planungstool entwickeln das uns zukünftig bei unseren Projekten die Arbeit erleichtert und das zusammensetzen aller Beteiligter reduziert.
* All Facts, one place  
  Es soll alle Interaktionen zentral abgespeichern, zwischen den Mitgliedern verteilen, aktuell halten und die Ausgaben verwalten.
* Vordiplom  
  Letztlich möchten wir mit der Arbeit die Vordiplomprüfung der HFU gut und erfolgreich abschliessen.
* Erfahrungen sammeln  
  Das Sammeln von Erfahrungen beim Entwickeln von interaktiven Applikationen und deren Design

## Aufgabenstellung

Die Anwendung soll portable Kommunikation bieten und dabei einfach zu bedienen sein.

Auf das wiederverwenden bestehenden Chat APIs wird bewusst verzichtet, um die Hoheit über dieses, sehr zentrale, Module zu haben und, in Bezug auf Designfragen, frei zu bleiben.

Projekte können aufgesetzt und anschliessend Aufgaben zugewiesen werden, welche wiederum den Projektmitgliedern verteilt werden können. Die Aufgaben können ein Budget aufweisen, welches mit den effektivkosten ergänzt werden kann.  
Es soll den Projektmitgliedern auch ermöglicht werden, Zeit auf eine bestimmte Aufgabe oder das Projekt als solches zu buchen.

Umfragen können ebenfalls erfasst und den Projektmitgliedern publiziert werden. Die Benutzer geben Ihre Stimme für eine der definierten Optionen ab.

Die Userverwaltung und Registrierung soll direkt im Frontend geschehen. Somit wird gewährleitet, dass man unmittelbar loslegen kann ohne einen grossen Registrierungsprozess zu durchlaufen.

Der zentrale Dreh- und Angelpunkt stellt damit die Applikation dar, welche der User bedient.

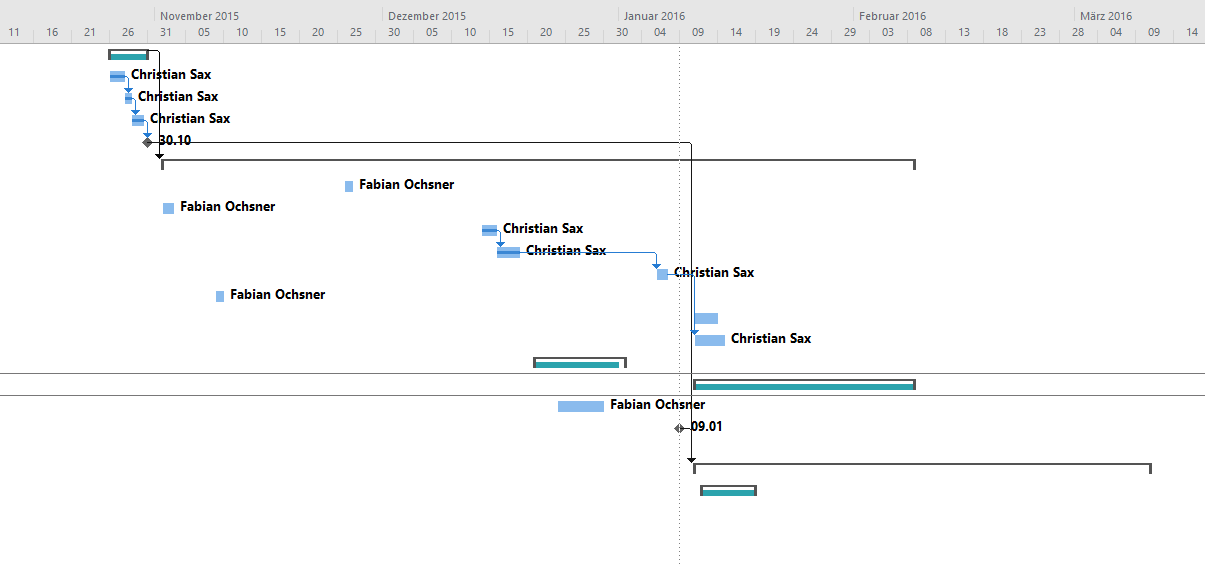
## Entscheidungs Matrix (Technologie)

## Vorgehen / Lösungsansatz

Im Rahmen einer Vorstudie werden verschieden Lösungsansätze und Technologien erörtert. Diese werden anschliessend über eine gewichtete Entscheidungsmatrix bewertet, was zur Wahl einer entsprechenden Vorgehensweise und den verwendeten Technologien führt.

Die Entscheidungsmatrix berücksichtigt im Vorfeld definierte Punkte und gewichtet deren Erreichung. Die Punkte ergeben sich aus dem Pflichtenheft und den daraus resultierenden Anforderungen an Technik und Entwicklung.

# Projektplanung



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vorgangsmodus | Vorgangsname | Dauer | Anfang | Ende | Vorgänger | Ressourcennamen |
| **Manuell geplant** | **Project preparation** | **5 d** | **Mon 26.10.15** | **Fre 30.10.15** |  |  |
| Automatisch geplant | Documentation template | 2 d | Mon 26.10.15 | Die 27.10.15 |  | Christian Sax |
| Automatisch geplant | Timesheets | 1 d | Mit 28.10.15 | Mit 28.10.15 | 2 | Christian Sax |
| Automatisch geplant | Proposal | 1.5 d | Don 29.10.15 | Fre 30.10.15 | 3 | Christian Sax |
| Automatisch geplant | Project Preparation Completed | 0 d | Fre 30.10.15 | Fre 30.10.15 | 4 |  |
| **Automatisch geplant** | **Requirement Engineering** | **71 d** | Mon 02.11.15 | **Mon 08.02.16** | **1** |  |
| Automatisch geplant | Project | 1 d | Don 26.11.15 | Don 26.11.15 |  | Fabian Ochsner |
| Automatisch geplant | Chat | 1.5 d | Mon 02.11.15 | Die 03.11.15 |  | Fabian Ochsner |
| Automatisch geplant | Task | 2 d | Mon 14.12.15 | Die 15.12.15 |  | Christian Sax |
| Automatisch geplant | Poll | 3 d | Mit 16.12.15 | Fre 18.12.15 | 9 | Christian Sax |
| Automatisch geplant | Logging | 1.5 d | Mit 06.01.16 | Don 07.01.16 | 10 | Christian Sax |
| Automatisch geplant | Accounting | 1 d | Mon 09.11.15 | Mon 09.11.15 |  | Fabian Ochsner |
| Automatisch geplant | Reporting | 3 d | Mon 11.01.16 | Mit 13.01.16 |  |  |
| Automatisch geplant | Security | 4 d | Mon 11.01.16 | Don 14.01.16 | 11 | Christian Sax |
| **Manuell geplant** | **Backend** | **10 d** | Mon 21.12.15 | **Fre 01.01.16** |  |  |
| Automatisch geplant | DB Model | 2 d | Mon 21.12.15 | Die 22.12.15 | 10 | Christian Sax |
| Automatisch geplant | Interaction Layer | 1.5 d | Mit 23.12.15 | Don 24.12.15 | 16 | Christian Sax |
| Automatisch geplant | Config Layer | 1 d | Don 24.12.15 | Fre 25.12.15 | 17 | Christian Sax |
| Automatisch geplant | Backend Completed | 0 d | Fre 01.01.16 | Fre 01.01.16 |  |  |
| **Manuell geplant** | **Webservice** | **21 d** | Mon 11.01.16 | **Mon 08.02.16** |  |  |
| Automatisch geplant | Query Specification | 4 d | Mon 11.01.16 | Don 14.01.16 | 18 | Christian Sax |
| Automatisch geplant | Update Specification | 5 d | Fre 15.01.16 | Don 21.01.16 | 21 | Christian Sax |
| Automatisch geplant | Update Documentation | 6 d | Mon 01.02.16 | Mon 08.02.16 | 22 | Christian Sax, Fabian Ochsner |
| Automatisch geplant | UI Design | 4 d | Don 24.12.15 | Die 29.12.15 |  | Fabian Ochsner |
| Automatisch geplant | Requirement Engineering Completed | 0 d | Sam 09.01.16 | Sam 09.01.16 |  |  |
| **Automatisch geplant** | **Imlementation** | **44 d** | Mon 11.01.16 | **Don 10.03.16** | **5;25** |  |
| **Manuell geplant** | **Testing** | **5 d** | **Die 12.01.16** | **Mon 18.01.16** |  |  |
| **Manuell geplant** | **Follow up** | **2 d** | **Don 31.03.16** | **Fre 01.04.16** |  |  |

# Pflichtenheft

## Struktur

Das Pflichtenheft ist in einer Matrixstruktur aufgebaut. Jede Zeile beschreibt eine spezifische Funktion, welche mit einer Gewichtung, Priorität, einem Owner und dem entsprechenden «Use Case» verlinkt wird. Das Due Date beschreibt den Fertigstellungszeitpunkt, welcher mit anderen Funktionen die Abhängigkeiten aufweisen abgeglichen ist.

### Zweck

Zweck des Pflichtenheftes ist es, den Funktionsumfang der Applikation zu definieren. Zusammen mit dem Projektplan bildet es die Grundlage der Teamarbeit. Durch das Pflichtenheft wird sichergestellt, dass man sich nicht in «Details verrennt» oder Funktionen implementiert welche nicht beauftragt wurden. Der Scope ist somit klar geregelt.

### Geltungsbereich

Das Pflichtenheft bezieht sich lediglich auch die geschilderte Applikation «MoCap» und findet für Entwickler als auch Benutzer Verwendung.

### Referenzierte Dokumente

Das Projekt beinhaltet keine referenzierten Dokumente. Eventuell verwendete «3rd party API»’s werden in Kapitel 7.2 dokumentiert und mittels «html link» mit externer Dokumentation / Reference verknüpft.

### Zielsetzung

Ziel ist es, die Applikation unter Berücksichtigung folgender Pattern zu entwickeln:

* MVC Pattern (GUI Design)
* Abstract Factory Pattern (Interaction classes)
* Builder Pattern (Manager classes)
* Singleton Pattern (Manager classes)
* Adapter Pattern (DB Backend)

Für weitere Informationen zu den verwendeten Design Pattern klicken sie [hier](http://www.dofactory.com/net/design-patterns).

### Funktionsmatrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Matrix** | | | | | | |
| **Use Case Id** | **Description** | **Topic** | **Weight** | **Priority** | **Score** | **Owner** |
| 26 | Create a new project | Project | 100 | 10 | 1000 | FO |
| 27 | Invite a user to participate in the project | Project | 100 | 10 | 1000 | FO |
| 28 | Delete a project | Project | 10 | 1 | 10 | FO |
| 29 | Leave a project | Project | 10 | 1 | 10 | FO |
| 30 | Bann a user from a project | Project | 10 | 1 | 10 | FO |
|  | Report time to a project | Project | 80 | 10 | 800 | FO |
|  | Report an expense to a project | Project | 80 | 10 | 800 | FO |
|  | Show overview of Expenses for project | Project | 50 | 10 | 500 | FO |
|  | Show overview of Timeslices for project | Project | 50 | 10 | 500 | FO |
| 1 | Create a task | Task | 100 | 10 | 1000 | CS |
| 2 | Update a task (Edit) | Task | 100 | 10 | 1000 | CS |
| 3 | Delete a task | Task | 10 | 1 | 10 | CS |
| 4 | Forward a task | Task | 10 | 1 | 10 | CS |
| 8 | Complete a task | Task | 100 | 1 | 100 | CS |
| 7 | Auto dispatch | Task | 10 | 1 | 10 | CS |
| 6 | Manual dispatch | Task | 80 | 2 | 160 | CS |
| 5 | Update progress | Task | 50 | 2 | 100 | CS |
| 10 | Create Survey | Survey | 100 | 10 | 1000 | CS |
|  | Edit survey | Survey | 100 | 10 | 1000 | CS |
|  | View survey | Survey | 100 | 10 | 1000 | CS |
| 11 | Create Option | Survey | 100 | 10 | 1000 | CS |
| 12 | Delete survey | Survey | 10 | 1 | 10 | CS |
| 13 | Complete survey | Survey | 10 | 1 | 10 | CS |
| 14 | Vote | Survey | 80 | 8 | 640 | CS |
|  | Read out vote | Survey | 80 | 8 | 640 | CS |
| 15 | Invite | Survey | 10 | 1 | 10 | CS |
| 16 | Assign task | Survey | 10 | 1 | 10 | CS |
| 46 | Create timeslice | Accounting | 100 | 10 | 1000 | FO |
| 46 | Create expense | Accounting | 100 | 10 | 1000 | FO |
| 31 / 32 | Attach bill | Accounting | 50 | 1 | 50 | FO |
|  | Assign to project / task | Accounting | 100 | 10 | 1000 | FO |
| 33 | Edit | Accounting | 100 | 10 | 1000 | FO |
| 34 | View project | Accounting | 100 | 10 | 1000 | FO |
| 35 | View | Accounting | 100 | 10 | 1000 | FO |
| 40 | Insert Project in DB | Backend | 100 | 10 | 1000 | CS |
| 41 | Insert Task in DB | Backend | 100 | 10 | 1000 | CS |
| 42 | Insert Survey in DB | Backend | 100 | 10 | 1000 | CS |
| 43 | Insert SurveyOptions in DB | Backend | 100 | 10 | 1000 | CS |
| 44 | Insert User in DB | Backend | 100 | 10 | 1000 | CS |
| 45 | Insert timeslice in db | Backend | 100 | 10 | 1000 | CS |
| 46 | Insert Expense in db | Backend | 100 | 10 | 1000 | CS |
| 47 | Update Project in DB | Backend | 100 | 10 | 1000 | CS |
| 48 | Update Task in DB | Backend | 100 | 10 | 1000 | CS |
| 49 | Update Survey in DB | Backend | 100 | 10 | 1000 | CS |
| 50 | Update SurveyOptions in DB | Backend | 100 | 10 | 1000 | CS |
| 51 | Update User in DB | Backend | 100 | 10 | 1000 | CS |
| 52 | Update timeslice in db | Backend | 100 | 10 | 1000 | CS |
| 53 | Update Expense in db | Backend | 100 | 10 | 1000 | CS |
| 54 | Insert interaction in db | Backend | 100 | 10 | 1000 | CS |
| 55 | Update interaction in db | Backend | 100 | 10 | 1000 | CS |
| 56 | View Project from db | Backend | 100 | 10 | 1000 | CS |
| 57 | View Task from db | Backend | 100 | 10 | 1000 | CS |
| 58 | View Survey from db | Backend | 100 | 10 | 1000 | CS |
| 59 | View User from db | Backend | 100 | 10 | 1000 | CS |
| 60 | View Timeslice from db | Backend | 100 | 10 | 1000 | CS |
| 61 | View Expense from DB | Backend | 100 | 10 | 1000 | CS |
| 62 | View Project User Mapping from db | Backend | 100 | 10 | 1000 | CS |
| 63 | View Project Task Mapping from db | Backend | 100 | 10 | 1000 | CS |
| 64 | View Project Survey Mapping from db | Backend | 100 | 10 | 1000 | CS |
| 65 | View Project Expense Mapping from db | Backend | 100 | 10 | 1000 | CS |
| 66 | View Project Timeslice from db | Backend | 100 | 10 | 1000 | CS |
| 67 | Register User | User | 100 | 10 | 1000 | CS |
| 68 | Create User | User | 100 | 10 | 1000 | CS |
| 69 | Modify User | User | 100 | 10 | 1000 | CS |
| 70 | Create person in DB | User | 100 | 10 | 1000 | CS |
| 71 | Update user password | User | 100 | 10 | 1000 | CS |
| 26 | Create a chat | Chat | 100 | 10 | 1000 | FO |
| 27 | Invite to chat | Chat | 10 | 1 | 10 | FO |
| 28 | delete chat | Chat | 10 | 1 | 10 | FO |
| 29 | Leave chat | Chat | 10 | 1 | 10 | FO |
| 30 | Kick user | Chat | 10 | 1 | 10 | FO |
| 80 | Create form overview item | Overview | 100 | 10 | 1000 | CS |
| 81 | List active projects | Overview | 100 | 10 | 1000 | CS |
| 82 | Show overview of Interactions | Overview | 100 | 10 | 7 | CS |

### Schnittstellen

Objekte welche im Rahmen der Applikation erstellt und manipuliert werden, sind zentral auf einem, im Internet gehosteten, SQL gespeichert.

Um unabhängiger zu sein wurde versucht «Dependencies» mittels Interfaces klein zu halten und so weit als möglich zu abstrahieren. Grundsätzlich wurden sechs Komponenten identifiziert, welche über entsprechende Schnittstellen miteinander kommunizieren. Diese Komponenten können wiederum in zwei Kategorien aufgeteilt werden, Interaktionsobjekte und «normale» Objekte.   
Jede Interaktion besitz ein Interface, um Abhängigkeiten bei der Entwicklung weitestgehend zu vermeiden.

* IProject  
  Das Projektobjekt woran diverse andere Objekte geknüpft sind.
* ITask  
  Das Taskobjekt, welches eine zu erledigende Aufgabe darstellt.
* ISurvey  
  Eine Umfrage, welche wiederum Objekte «Vote» und «Option» miteinander verknüpft.
* IChat  
  Das Chat Modul, welches live Kommunikation verarbeitet
* ITimeslice und IExpense  
  Das Timeslice Objekt bildet eine zeitliche Arbeitsleistung ab, wo hingegen Expense eine finanzielle Leistung deklariert. Beide werden unter dem Begriff Accounting zusammengefasst.
* IUser  
  Der eigentliche Benutzer, welcher das System bedient und Interaktionen bearbeitet.

Diese Komponenten werden wiederum über zwei weitere Schnittstellen gekapselt und darüber erstellt.

* IInteractionFactory  
  Sämtliche Interaktionen (Objekte) werden über dieses Interface erstellt.
* IObjectFactory  
  Objekte welche nicht vom Typ IInteraction sind, werden über dieses Interface erstellt.

Die Schnittstellen werden im Kapitel 7.4 Realisation der Software eingehend beschrieben.

## Abgrenzung der Aufgabe

Es wird lediglich die Funktion der Applikation, wie in der Funktionsmatrix beschrieben, abgebildet. Um das Tool anschliessend verwenden zu können, wird eine Internetverbindung vorausgesetzt.

Des Weiteren muss sich jeder Benutzer über die Applikation registrieren. Dieser Prozess beinhaltet das Erstellen eines Users, welcher anschliessend für die Arbeit mit dem Tool verwendet wird.

# Analyse / Konzept-Varianten

## Zweck und Umfang der Vorstudie / Analyse

Die Analyse bezieht sich auf die zu entwickelnde Lösung. Sie umfasst die Bereiche Architektur, Entwicklungsumgebungen, Datenbanktechnologien und deren Anbindung.

## Zielsetzungen

Im Rahmen der Analyse sollen verschiedene Ansatzweisen als auch Technologien erörtert und bewertet werden, um im Anschluss eine Grundlage, für das Design und der zu verwendenden Technologien, zu bilden.

## Analyse

### Technology Evaluation

#### Programmiersprache

1. Die Applikation muss auf Windows 8 oder höher laufen
2. Die verwendete Technologie muss über die IDE einen GUI-Designer bieten
3. Technologie muss OS Integration bieten um Alarme und/oder Benachrichtigungen zu generieren
4. Lässt MVC Programmiermuster zu
5. Technologie muss benutzerfreundlich und einfach zu handhaben sein
6. Die Programmiersprache muss vielseitig sein (Portierbarer Code)
7. Technologie muss eine hohe Effizienz bieten, wenn es um GUI-Design geht

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Descision Matrix** | | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** |  |
| WinForm C# | 10 | 10 | 10 | 10 | 9 | 9 | 10 |  |
| ASP.net c# | 10 | 9 | 8 | 10 | 7 | 10 | 7 |  |
| Java JSF | 10 | 8 | 7 | 10 | 6 | 10 | 6 |  |
| Java Swing | 8 | 10 | 9 | 10 | 9 | 10 | 8 |  |
|  |  |  |  |  |  |  |  |  |
| **Criterion Weight** | 10 | 10 | 8 | 10 | 10 | 8 | 10 |  |
|  |  |  |  |  |  |  |  |  |
| **Weighted Scores** | | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **Total Score** |
| WinForm C# | 100 | 100 | 80 | 100 | 90 | 72 | 100 | **642** |
| ASP.net c# | 100 | 90 | 64 | 100 | 70 | 80 | 70 | **574** |
| Java JSF | 100 | 80 | 56 | 100 | 60 | 80 | 60 | **536** |
| Java Swing | 80 | 100 | 72 | 100 | 90 | 80 | 80 | **602** |

#### WebServer

1. Must be at hand
2. Geringe Kosten
3. Einfach einzurichten
4. c# Web-Service ist einfach bereitzustellen
5. Uptime muss über 99.9% liegen
6. Fernverwaltung

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Descision Matrix** | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** |  |
| Hosted WebSvr IIS7 | 10 | 6 | 7 | 10 | 10 | 10 |  |
| Hosted WebSvr Linux | 1 | 8 | 6 | 4 | 10 | 10 |  |
| Hosted Notebook | 10 | 10 | 10 | 10 | 1 | 1 |  |
| Hosted Home Server | 10 | 10 | 5 | 8 | 8 | 1 |  |
|  |  |  |  |  |  |  |  |
| **Criterion Weight** | 8 | 8 | 6 | 10 | 9 | 5 |  |
|  |  |  |  |  |  |  |  |
| **Weighted Scores** | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **Total Score** |
| Hosted WebSvr IIS7 | 80 | 48 | 42 | 100 | 90 | 50 | **410** |
| Hosted WebSvr Linux | 8 | 64 | 36 | 40 | 90 | 50 | **288** |
| Hosted Notebook | 80 | 80 | 60 | 100 | 9 | 5 | **334** |
| Hosted Home Server | 80 | 80 | 30 | 80 | 72 | 5 | **347** |

#### Datenbank Technologie

1. Must be at hand
2. Geringe Kosten
3. Fernverwaltung
4. Muss zusammen mit einem Webserver kombiniert sein
5. Uptime muss 99.9% übersteigen
6. Muss entity framework unterstützen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Descision Matrix** | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** |  |
| MS SQL | 10 | 6 | 10 | 10 | 10 | 10 |  |
| Oracle | 1 | 5 | 10 | 7 | 10 | 10 |  |
| MySQL | 1 | 10 | 10 | 10 | 10 | 10 |  |
| SQL Express | 10 | 10 | 5 | 8 | 8 | 10 |  |
|  |  |  |  |  |  |  |  |
| **Criterion Weight** | 8 | 8 | 6 | 10 | 9 | 8 |  |
|  |  |  |  |  |  |  |  |
| **Weighted Scores** | | | | | | |  |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **Total Score** |
| MS SQL | 80 | 48 | 60 | 100 | 90 | 80 | **458** |
| Oracle | 8 | 40 | 60 | 70 | 90 | 80 | **348** |
| MySQL | 8 | 80 | 60 | 100 | 90 | 80 | **418** |
| SQL Express | 80 | 80 | 30 | 80 | 72 | 80 | **422** |

### Architecture Evaluation

Es gilt folgende Punkt betreffend Architektur zu berücksichtigen und zu bewerten

#### Datenbankanbindung

1. Flexibilität und Hoheit bei der Anbindung an Datenbank
2. Geringe Einarbeitungszeit
3. Möglichst direkte Anbindung ohne zusätzliche Overhead

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Descision Matrix** | | | |  |
|  | **A** | **B** | **C** |  |
| EntityFramework | 8 | 6 | 8 |  |
| ADO | 10 | 10 | 10 |  |
| WebService | 4 | 8 | 6 |  |
|  |  |  |  |  |
| **Criterion Weight** | 10 | 8 | 7 |  |
|  |  |  |  |  |
| **Weighted Scores** | | | |  |
|  | **A** | **B** | **C** | **Total Score** |
| EntityFramework | 80 | 48 | 56 | **184** |
| ADO | 100 | 80 | 70 | **250** |
| WebService | 40 | 64 | 42 | **146** |

### Fazit

Aufgrund der Evaluation gibt es folgende Varianten,

1. Webbasierter Client, welche sich jeweils direkt auf die Datenbank zugreift
2. Standalone Webclient, welcher seine Daten direkt mit anderen Clients austauscht
3. WinForm Client mit zentralem DB Server (ADO)
4. WinForm Client mit Webservice als DB Zugriff
5. WinForm Client mit Entity Framework
6. Web Client mit Entity Framework

Woraus Variante “C” als Lösungsarchitektur ergibt. Es entfallen somit der Webservice oder das Entity Framework, was einen Zusätzliche Layer bedeuten würde.

# Realisierung / Umsetzung

## Zweck und Umfang der Realisation

Aus der Realisierung soll eine Lauffähige Applikation geschaffen werden, welche der evaluierten Variante aus [Kapitel 5](#_Analyse_/_Konzept-Varianten) entspricht.

Die finale Lösung setzt sich grob aus zwei Einheiten zusammen, dem Client und dem Backend (Datenbank). Der Client bezieht seine Daten von einem zentralen Datenbankserver, welcher die Objekte in Form von Datensätzen speichert.

Nach beenden der Realisation sollten beide Einheiten funktionsfähig zur Verfügung stehen. Die Applikation soll keiner Installation bedürfen, sondern direkt über eine ausführbare Datei gestartet werden.

## Fremdbibliotheken / API’s

### Docking Content

Zur Darstellung von «dockable content» wird die Open-Source Bibliothek «WeifenLuo.WinFormsUI.Docking» verwendet. Die Reference zur API finden sie [hier](http://docs.dockpanelsuite.com/en/latest/).

Die API findet ausschliesslich in den «WinForms» Komponenten Anwendung.

## Realisation gemäss Konzept im Top-Down Verfahren

Im Folgenden wird die Applikation stufenweise heruntergebrochen, wobei auf oberster Ebene jeweils die Unterscheidung zwischen Client und Backend gemacht wird.

### Systemkontext



### Grundlegender Nachrichtenverlauf



## Realisation von SW

### Project

Das Project ist eine Komponente mit der jeder Benutzer Aufgaben(Task) und Umfragen(Survey) Gruppieren und mit anderen Mitgliedern abarbeiten kann. Auf dem Project können Zeit und Geld abgebucht und somit den gesamten Aufwand und Ausgaben ermittelt werden.

#### Create

|  |  |
| --- | --- |
| **Use Case Id:** | 26 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user creates a new project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User clicks "New" button  2. User enters a title 3. User enters a description of the project 4. User specified the end date  5. User specifies if poll or balance is included in project 7. User clicks "Save" button  8. Project created |
| **Alternative Flow:** | 7a. The user cancels the create request  8. Dialog closes |
| **Exceptions:** | None |

#### Invite

|  |  |
| --- | --- |
| **Use Case Id:** | 27 |
| **Use Case Name:** | Invite |
| **Actors:** | User |
| **Description:** | The user invites other users to an existing project |
| **Preconditions:** | 1. User is logged in  2. User opened the project panel 3. User has created a project |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Invite User" button 3. User selects another users 4. User clicks "Invite" button  5. User added |
| **Alternative Flow:** | 4a. The user cancels the invite request  5. Dialog closes |
| **Exceptions:** | None |

#### Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 28 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Edit" button  3. User clicks “Delete Project” button 4. User clicks the “Delete” button 5. Process ends |
| **Alternative Flow:** | 4a. The user cancels the delete request  5. Dialog closes |
| **Exceptions:** | None |

#### Leave

|  |  |
| --- | --- |
| **Use Case Id:** | 29 |
| **Use Case Name:** | Leave |
| **Actors:** | User |
| **Description:** | The user leaves a project |
| **Preconditions:** | 1. User is logged in 2. User opened the project panel |
| **Normal Flow:** | 1. User enters the project  2. User clicks "Options" button 3. User clicks "Leave Project" button  4. User clicks the “Delete Project” button 5. Process ends |
| **Alternative Flow:** | 3a. The user cancels the leave request  5. Dialog closes  4a. The User cancels the delete request  6. Dialog closes |
| **Exceptions:** | None |

### Accounting

Das Accounting verwaltet die Ausgaben(expense) und Zeitaufwand(timeslice) Objekte. In ihm ist hinterlegt welche Ausgaben/Zeitaufwand zu welchem Task gehören. Es ist jeweils immer ein Accounting Objekt mit einem Project verbunden.

#### Create Balance

|  |  |
| --- | --- |
| **Use Case Id:** | 46 |
| **Use Case Name:** | Create Balance |
| **Actors:** | User |
| **Description:** | The user creates a balance object |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel  3. The task is assigned to the user |
| **Normal Flow:** | 1. User clicks "Update" button 2. User enters expenses of the task  3. User clicks “Add” button |
| **Alternative Flow:** | 3a. The user cancels the attach request  4. Dialog closes |
| **Exceptions:** | None |

#### Create Timeslice

|  |  |
| --- | --- |
| **Use Case Id:** | 46 |
| **Use Case Name:** | Create Timeslice |
| **Actors:** | User |
| **Description:** | The user creates a balance object |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel  3. The task is assigned to the user |
| **Normal Flow:** | 1. User clicks "Update" button 2. User enters Time used for the task  3. User clicks “set” button |
| **Alternative Flow:** | 3a. The user cancels the attach request  4. Dialog closes |
| **Exceptions:** | None |

# Test

# Dokumentation

# Glossar und Verzeichniss

# Beilagen

## Test the plan

### Consumer perceptions

|  | Consumer Perceptions |
| --- | --- |
|  | [List consumer perceptions about your product or service.] |
|  |  |
|  |  |
|  |  |
|  |  |

### Quantitative metrics

|  | Quantitative Metrics |
| --- | --- |
|  | [List quantitative metrics that you will use to evaluate the effectiveness of the plan.] |
|  |  |
|  |  |
|  |  |
|  |  |

## Adjust the plan

|  | Plan Changes |
| --- | --- |
|  | [List the changes that you need to make to your plan so that it will be more successful.] |
|  |  |
|  |  |
|  |  |
|  |  |

## Put the plan into action

| Step | Action | Due date for completion | % complete |
| --- | --- | --- | --- |
|  |  | [Date] | % |
|  |  | [Date] | % |
|  |  | [Date] | % |
|  |  | [Date] | % |
|  |  | [Date] | % |
|  |  | [Date] | % |