

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)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **Duration Est.** | **Duration Eff. Chris** | **Duration Eff. Fabian** | **In Charge Est.** | **In Charge Eff.** |
| **Project preparation** |  |  |  |  |  |
| Documentation template | 16.00 | 0.50 | 0.00 | CS |  |
| Timesheets | 8.00 | 0.50 | 0.00 | CS |  |
| Proposal | 12.00 | 1.50 | 0.00 | CS |  |
| **Project Preparation Completed** |  | **2.50** | 0.00 |  |  |
| **Requirement Engineering** |  |  |  |  |  |
| Project | 8.00 |  | 9.50 | FO |  |
| Chat | 12.00 | 0.00 | 7.00 | FO |  |
| Task | 16.00 | 9.50 | 0.00 | CS |  |
| Survey | 24.00 | 6.50 | 0.00 | CS |  |
| Accounting | 8.00 | 5.00 | 6.00 | FO | CS, FO |
| Reporting | 24.00 | 0.00 | 0.00 |  |  |
| Security | 32.00 | 2.50 | 0.00 | CS |  |
| **Backend** |  |  |  |  |  |
| DB Model | 16.00 | 29.50 | 0.00 | CS |  |
| Interaction Layer | 12.00 | 10.50 | 0.00 | CS |  |
| Config Layer | 8.00 | 0.00 | 0.00 | CS |  |
| Backend Completed | 0.00 |  |  |  |  |
| **Webservice** |  |  |  |  |  |
| Query Specification | 32.00 | 9.50 | 0.00 | CS |  |
| Update Specification | 40.00 | 11.50 | 0.00 | CS |  |
| Update Documentation | 48.00 | 3.50 | 0.00 | CS, FO |  |
| UI Design | 32.00 | 21.00 | 19.00 | FO | CS, FO |
| **Requirement Engineering Completed** |  | **109.00** | 41.50 |  |  |
| **Imlementation** |  |  |  |  |  |
| Project | 16.00 | 3.50 | 38.00 | FO | FO, CS |
| Chat | 24.00 | 0.00 | 0.00 | FO |  |
| Task | 24.00 | 9.50 | 0.00 | CS |  |
| Survey | 32.00 | 17.50 | 0.00 | CS |  |
| Logging | 32.00 | 0.00 | 0.00 | CS |  |
| Accounting | 16.00 | 1.00 | 22.00 | FO | FO, CS |
| Reporting | 40.00 | 0.00 | 0.00 |  |  |
| Security | 48.00 | 4.50 | 0.00 | CS |  |
| Webservice | 80.00 | 0.00 | 0.00 | CS |  |
| UI Implementation | 32.00 | 32.50 | 29.00 | FO |  |
| **Implementation Completed** |  | **68.50** | 89.00 |  |  |
| **Testing** |  |  |  |  |  |
| Specify TestCases | 40.00 | 7.50 | 9.00 | FO, CS | CS, FO |
| Project | 8.00 | 0.00 | 6.00 | FO |  |
| Chat | 12.00 | 0.00 | 0.00 | FO |  |
| Task | 12.00 | 8.50 | 0.00 | CS |  |
| Survey | 16.00 | 7.50 | 0.00 | CS |  |
| Logging | 8.00 | 0.00 | 0.00 | CS |  |
| Accounting | 8.00 | 4.50 | 2.00 | FO |  |
| Reporting | 16.00 | 0.00 | 0.00 |  |  |
| Security | 16.00 | 5.00 | 0.00 |  |  |
| Webservice | 40.00 | 0.00 | 0.00 |  |  |
| UI Implementation | 16.00 | 15.50 | 16.00 | FO | CS, FO |
| **Testing Completed** |  | **48.50** | 33.00 |  |  |
| **Follow up** |  |  |  |  |  |
| Prepare official Documents | 16.00 | 12.50 | 6.00 | CS |  |
| Prepare Media | 16.00 | 2.50 | 0.00 | CS |  |
| Handover | 8.00 | 5.50 | 0.00 | CS |  |
| **Grand Total** | **924.00** | **249.00** | **169.50** |  |  |

# 

# 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.

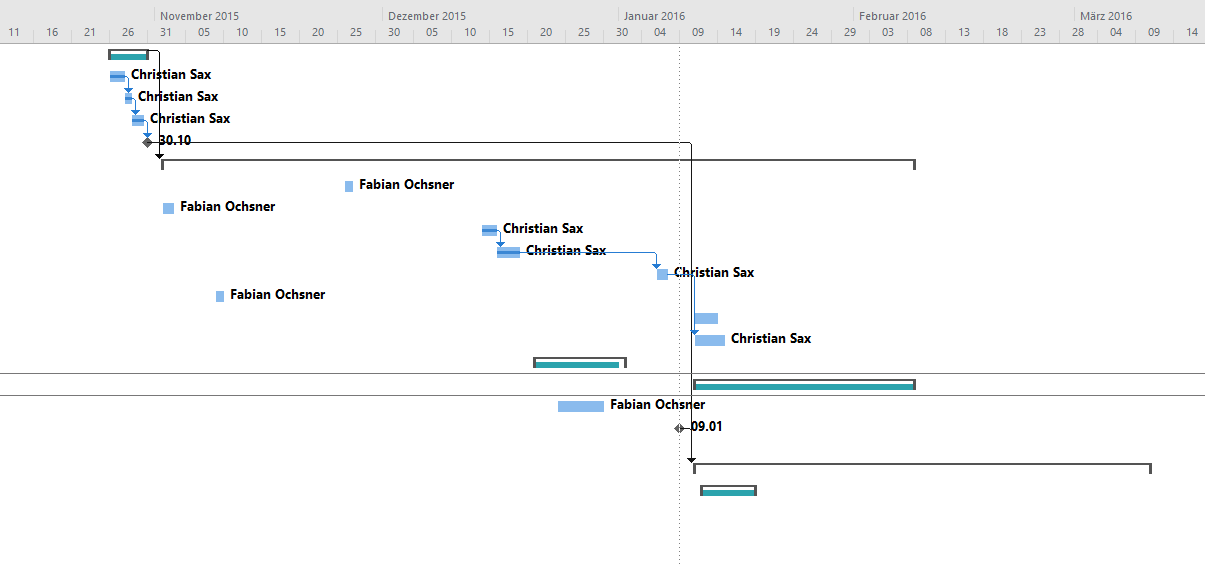
## Entscheidungs Matrix (Technologie)

## 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 MoCap

Ein Projekt ist eine Sammlung von verschiedenen Interaktionen, die von den Projektmitgliedern ausgeführt oder bearbeitet werden.

* **Project**Allem voran steht das Projekt. Es definiert einen Namen und die Mitglieder, welche für dieses Projekt arbeiten. Ebenso ist ein Start- und Endtermin gesetzt in welchem das Projekt aktiv ist.
* **Task**Anschliessend werden dem Projekt Tätigkeiten zugewiesen, sogenannte Tasks. Tasks beschreiben eine Aufgabe und haben ebenfalls einen Start- und Endzeitpunkt. Aufgaben haben ein unter Umständen ein zeitliches als auch finanzielles Budget, welches einzuhalten ist und ein bestimmtes Fälligkeitsdatum.
* **Chats**Im Rahmen eines Projektes können Nachrichten oder Gespräche geführt werden, welche als Chats bezeichnet werden. Auch Chats werden automatisch einem Projekt angehängt.
* **Survey**Darüber hinaus kann es nötig sein, gewisse Umfragen in Bezug auf das Projekt zu starten, genannt Surveys. Diese Umfragen enthalten verschieden Optionen. Das Projektmitglied kann nun seine Stimme für eines dieser Optionen abgeben. Dies wird ein Vote genannt. Auch ein Survey kann in seiner Gültigkeit beschränkt sein und ein Fälligkeitsdatum aufweisen.
* **Outlay**Letztlich gibt es Ausgaben. Es werden hier zwei Fälle unterschieden. Zum einen die Expenses was finanzielle Ausgaben sind und zum anderen die Timeslices, was einen Zeitlichen Aufwand darstellt. Diese Expenses können nun im Rahmen eines Projektes direkt oder aber für einen Task anfallen.

Die soeben beschriebenen Objekte werden als Interaction zusammengefasst, da sie ähnlicher Art sind. Aufgrund dieser Tatsache wird ein Interface, genannt «IInteraction» eingeführt, für die Definition von gemeinsamen Attributen, Funktionen und Ereignissen definiert und eingeführt.

Jede Applikation benötigt ein User Management. Der User wird in der Applikation selbst registriert und kann anschliessend das Tool verwenden.

Die Darstellung wird über WinForms gemacht und erhalten ebenfalls einen eingenen Namespace. Da die Forms selbst keine Logik abbilden sollen, sondern lediglich darstellen, wird die Business Logik in sogenannte Managers ausgelagert. Die Manager bilden das Zentrale Element der Applikation.

Das Backend regelt sämtliche Datenbank Funktionen. Hier geschieht auch die Konvertierung von einem Datensatz zu einem Objekt und umgekehrt.

Jede Komponente verfügt über ihren eigenen Namespace, der analog zur Komponente benannt wurde.

### Systemkontext



### Grundlegende Strukturierung



### MoCap.Interactions NameSpace

In diesem Namensraum werden folgende Komponenten, inklusive deren Use Cases, abgebildet.

#### 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.

Jeder Nutzer von MoCap kann Projekte eröffnen und andere Mitglieder dazu einladen. Jeder Nutzer von MoCap kann zu mehreren Projekten gehören.

Beim Kreieren eines Projektes kann man auswählen ob Umfragen und/oder Ausgaben bei diesem vorhanden sein werden oder nicht.

Nur Mitglieder eines Projektes können Aufgaben übernehmen oder an Umfragen teilnehmen.



##### 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 |

#### Task



##### Create

|  |  |
| --- | --- |
| **Use Case Id:** | 1 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user creates a new task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User clicks "New Task" button 2. User enters a title 3. User enters a description of the task 4. User estimates the duration 5. User specifies the start date 6. User specified the due date (if any) 7. User associates task with project 8. User specifies a budget (if any) 9. User selects either "auto-assign", "manual-assign" or "poll-assign" 10. User clicks "Save" button |
| **Alternative Flow:** | 9a. If the user selects "poll-assign"  10. => Goto Use Case **"Create" (poll)** 7b. Project does not exist yet  7. => Goto Use Case **"Create" (project)** |
| **Exceptions:** | None |

##### Update

|  |  |
| --- | --- |
| **Use Case Id:** | 2 |
| **Use Case Name:** | Update |
| **Actors:** | User |
| **Description:** | The user updates task information |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task to be updated from within the task list 2. User updates corresponding fields 3. User saves changes |
| **Alternative Flow:** | 2a. If the user adds subtasks to the task  Goto Use Case **"Create" (task)** 2b. If the user re-assigns the task  Goto Use Case **"Assign" (task)** |
| **Exceptions:** | None |

##### Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 3 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be deleted 2. User clicks the "delete task" button 3. User confirms the deletion |
| **Alternative Flow:** | 3a. If the user cancels the deletion  3. => abort process |
| **Exceptions:** | None |

##### Forward

|  |  |
| --- | --- |
| **Use Case Id:** | 4 |
| **Use Case Name:** | Forward |
| **Actors:** | User |
| **Description:** | The user forwards the task to another user |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be forwarded 2. User clicks the dispatch button 3. User selects either "auto-dispatch", "manual-dispatch" or "poll-dispatch" 4. User clicks "save" button |
| **Alternative Flow:** |  |
| **Exceptions:** | 2a. The user is not permitted to forward the task  3. => abort process |

##### UpdateProgress

|  |  |
| --- | --- |
| **Use Case Id:** | 5 |
| **Use Case Name:** | UpdateProgress |
| **Actors:** | User |
| **Description:** | The user updates the progress of the task |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be updated 2. User changes the progress value 3. User clicks the "save" button 4. System updates task progress and subtask(-s) progress accordingly 5. Dialog closes |
| **Alternative Flow:** | 2a. The user updates a subtask  3. => Goto "UpdateProgress" use case for subtask  4. System updates the progress based on subtask status  5. User clicks the "save" button  6. Dialog closes 2b. User adds a subtask to the task  3. => Goto use case "Create" (task)   4. => Continue as per 2a.4 2c. User deletes a subtask  3. => Goto use case "Delete" (task)  4. => Continue as per 2a.4 2d. User completes the task  3. => Goto use case "Complete" (task) 2e. User cancels the dialog  3. Dialog closes |
| **Exceptions:** | 2a. The user specifies a negative or lower value than before  3. Message is shown  4. => Goto 2. |

##### ManualDispatch

|  |  |
| --- | --- |
| **Use Case Id:** | 6 |
| **Use Case Name:** | ManualDispatch |
| **Actors:** | User |
| **Description:** | The user dispatches the task manually |
| **Preconditions:** | 1. User is logged in 2. User opened the task panel |
| **Normal Flow:** | 1. User selects the task (-s) to be dispatched 2. User selects "Manual Dispatch" 3. User selects project member to assign the task 4. System evaluates availability of user => Goto use case "GetMatchingMembers" (task) 5. System displays availability stats 6. User clicks the "Dispatch" button 7. System dispatches the task to selected member 8. Dialog closes |
| **Alternative Flow:** | 5a. The selected user does not have anough availability  6. => Goto use case "NotifyInsufficientAvailability" (task)  7. Dialog closes 2a. User selects "Auto-Dispatch"  3. System evaluates availability of user => Use Case "GetIdealMemebers" (task)  4. User clicks "Dispatch" button  5. Dialog closes   6. System selects project member  7. System assigns task to member 2b. User selects "Poll-Dispatch"  3. => Goto use case "Create" (Poll)  4. Dialog closes  5. System assigns task based on poll result |
| **Exceptions:** | none |

##### AutoDispatch

|  |  |
| --- | --- |
| **Use Case Id:** | 7 |
| **Use Case Name:** | AutoDispatch |
| **Actors:** | System |
| **Description:** | The system dispatches a task based on availability, and preferences defined |
| **Preconditions:** | 1. A task exists or was created 2. Auto-Dispatch function was called |
| **Normal Flow:** | 1. System retrieves project members 2. System calculates scor card by executing "Get Ideal Members" use case 3. System identifies member with highest score card 4. System assigns task to user 5. System raises "Task Assigned" event 6. System adds users to task |
| **Alternative Flow:** | 2a. No due date specified  3. Randomly select user  4. => Goto use case step 4 2b. System does not find user with sufficient availability  3. => Raise "Insufficient Resources" Exception  4. End process |
| **Exceptions:** | 2b. Insufficient availability |

##### Complete

|  |  |
| --- | --- |
| **Use Case Id:** | 8 |
| **Use Case Name:** | Complete |
| **Actors:** | System |
| **Description:** | The system detects that the task is completed after the user has updated the task progress |
| **Preconditions:** | 1. User has updated the task progress 2. Task is completed after update progress ends |
| **Normal Flow:** | 1. System marks the task as "Completed" 2. System notifies the users (Creator and current task owner) 3. System checks if the task is a subtask and updates the parent task if required 4. System raises "Task Completed" event |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

#### Survey



##### Create

|  |  |
| --- | --- |
| **Use Case Id:** | 10 |
| **Use Case Name:** | Create |
| **Actors:** | User |
| **Description:** | The user is creating a new Poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User clicks the create poll button 2. User specifies a poll title 3. User enters a description of the poll (optional) 4. User defines the date this poll ends 5. User defines whether or not custom options are allowed to vote for 6. User enters a list of options to vote 7. User selects the people eligible to vote 8. User specifies the number of votes per users 9. User saves the vote 2. System identifies member offering sufficient time to this project 3. Process ends |
| **Alternative Flow:** | 4a. No end date specified  4. Poll ends once each user has voted  5. Goto use case step 5 5a. User selects a task to distribute  6. Options are set automatically (including all project members)  7. Eligible users are project members only  8. Goto use case step 8 |
| **Exceptions:** | None |

##### CreateOptions

|  |  |
| --- | --- |
| **Use Case Id:** | 11 |
| **Use Case Name:** | CreateOptions |
| **Actors:** | User |
| **Description:** | The user is creating a new option selectable in a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User clicks the create option button 2. User specifies an option title 3. User enters a description for the option (optional) 9. User saves the option |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### Delete

|  |  |
| --- | --- |
| **Use Case Id:** | 12 |
| **Use Case Name:** | Delete |
| **Actors:** | User |
| **Description:** | The user deletes a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User selects the poll to be deleted 2. User clicks the delete button 3. Process ends |
| **Alternative Flow:** | 2a. The user cancels the delete request  3. Dialog closes |
| **Exceptions:** | None |

##### Complete

|  |  |
| --- | --- |
| **Use Case Id:** | 13 |
| **Use Case Name:** | Complete |
| **Actors:** | User / System |
| **Description:** | Occurs once the poll is completed. They happens either when the due date expires or all users have left their vote |
| **Preconditions:** | 1. User is logged in 2. User opened the poll panel |
| **Normal Flow:** | 1. User votes 2. System checks if all users have voted or due date expired 3. System closes the poll 4. System notifies users 5. System marks poll as completed |
| **Alternative Flow:** | 2a. Some votes are still open  3. The process ends 5a. The poll was a assign poll  6. The system assigns the task to user  7. System notifies the task owner  8. The process ends |
| **Exceptions:** | None |

##### Vote

|  |  |
| --- | --- |
| **Use Case Id:** | 14 |
| **Use Case Name:** | Vote |
| **Actors:** | User |
| **Description:** | The user votes for an option |
| **Preconditions:** | 1. User is logged in 2. User opened the poll |
| **Normal Flow:** | 1. User selects an option 2. User clicks save button to submit his vote 3. Dialog closes |
| **Alternative Flow:** | 1a. User files a custom option  2. The user defines a new option  3. User saves the new option  4. User selects the new option  5. User saves the vote  6. The dialog closes |
| **Exceptions:** | None |

##### Invite

|  |  |
| --- | --- |
| **Use Case Id:** | 15 |
| **Use Case Name:** | Invite |
| **Actors:** | User |
| **Description:** | The user invites participants for a poll |
| **Preconditions:** | 1. User is logged in 2. User opened the poll |
| **Normal Flow:** | 1. User opens the poll 2. User clicks the participants button 3. User adds participants from either a project or individually from his / her addressbook 4. User saves the poll 5. The dialog closes 6. The system notifies the user 7. Process ends |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### AssignTask

|  |  |
| --- | --- |
| **Use Case Id:** | 16 |
| **Use Case Name:** | AssignTask |
| **Actors:** | System |
| **Description:** | Occurs if a poll has completed and a task was attached to it |
| **Preconditions:** | 1. Poll completed 2. Task attached to poll |
| **Normal Flow:** | 1. System couts most user votes 2. System assigns user to task 3. System updates the task 4. System updates the project 5. Process ends |
| **Alternative Flow:** | None |
| **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.

Es zeigt die Ausgaben von jedem Task in einem Projekt, es vergleicht die Ausgaben alle Mitglieder und zeigt wieviel jeder jedem schuldet.

Jeder User kann den Tasks die ihm sind expenses anhängen. Die anderen Mitglieder können diese einsehen und sehen wieviel jeder für seine Tasks ausgegeben hat.



##### 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 |

#### Klassendiagram



### MoCap.Security NameSpace

Dieser Namespace beinhaltet das User Objekt und dessen Person Interface.

#### User

##### LoginUser

|  |  |
| --- | --- |
| **Use Case Id:** | 67a |
| **Use Case Name:** | Login User |
| **Actors:** | User |
| **Description:** | The user starts the application and logs in |
| **Preconditions:** | 1. User is registered |
| **Normal Flow:** | 1. User enters credentials 2. Clicks the login button |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

##### Logout User

|  |  |
| --- | --- |
| **Use Case Id:** | 67b |
| **Use Case Name:** | Logout User |
| **Actors:** | User |
| **Description:** | The user logs out |
| **Preconditions:** | 1. User is registered 2. User is logged in |
| **Normal Flow:** | 1. User clicks the logout button |
| **Alternative Flow:** | None |
| **Exceptions:** | None |

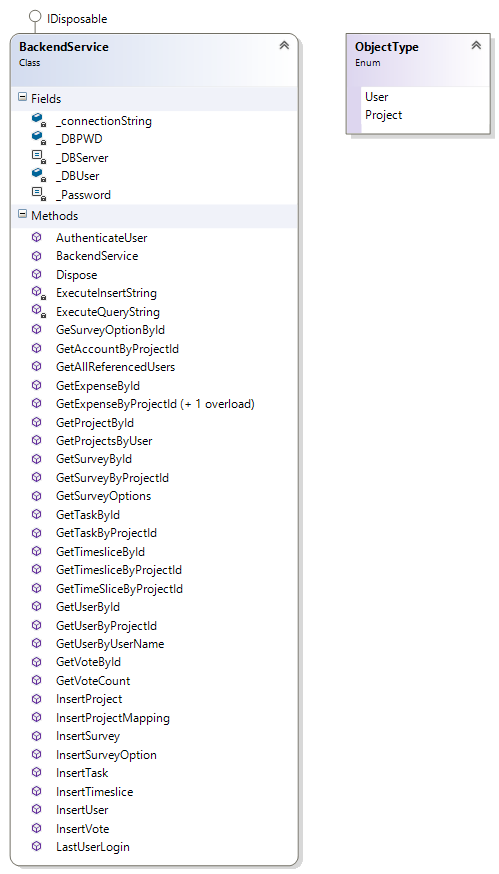
#### Klassendiagramm



### MoCap.Backend Namespace

Dieser Namespace beinhaltet die Backendklasse, welche die Kommunikation mit der Datenbank regelt. Es beinhaltet ebenfalls das Design der Datenbank und sämtliche Objekte.

Die Grundsätzliche Idee hinter dem Design ist das implementieren von Views um Daten abzufragen wo Stored Procedures für das Einfügen oder Updaten von Daten verwendet wird. Somit werden Direktzugriffe gekapselt und die zu berücksichtigende Logik, beim Speichern der Daten, konsequent angewendet.



# Test

# Dokumentation

## Project-Klasse

Die Project Klasse wird von der InteractionFactory-Klasse aufgerufen und ist das Project-Objekt, dass für das Speichern der Project Informationen verwendet wird, welche vom User über das GUI eingegeben werden. Ohne dieses Objekt könnten keine Daten über die jeweiligen Projekte gespeichert werden.

In der Project-Klasse sind folgende Eigenschaften enthalten:

Eine ID um alle Projekte voneinander unterscheiden zu können, eine StartDateTime um den Start Zeitpunkt zu speichern, eine EndDateTime un den Endzeitpunkt zu speichern, ein CreateDateTime um den Zeitpunkt zu speichern an dem das Objekt erstellt wurde, ein ModifiedDateTime um den Zeitpunkt der letzten Änderung zu speichern, ein IsActive um zu speichern ob das Objekt aktiv ist oder nicht, ein Text für eine Beschreibung des Objektes, ein Type für den Typ des Objektes hier immer Projekt, einen Creator in dem der Erschaffer gespeichert ist, einen Owner für den momentanen Besitzer, ein State für den Status des Projektes, ein EnableBalance um festzulegen ob Ausgaben in dem Projekt gespeichert werden können, ein EnableSurvey um fest zulegen ob im Projekt Umfragen gemacht werden können, ein ProjectAccount in dem der Account des Projektes gespeichert wird, eine InvitationList für alle eingeladenen User, eine TaskList für alle Aufgaben des Projektes, eine SurveyList für alle Umfragen des Projektes, eine MemberList für alle Mitglieder des Projektes und einem Namen für den Titel des Projektes.

## Task-Klasse

## Survey-Klasse

## Account-Klasse

Die Account Klasse wird von der InteractionFactory-Klasse aufgerufen und ist das Account-Objekt, dass für das Speichern der Account Informationen verwendet wird. Das Account-Objekt verwaltet die Expense- und Timeslice-Daten. Über das Account-Objekt können die gesamt ausgaben oder auch den gesamten Zeitaufwand eines Users abgerufen werden.

In der Project-Klasse sind folgende Eigenschaften enthalten:

Eine ID um alle Accounts voneinander unterscheiden zu können, eine StartDateTime um den Start Zeitpunkt zu speichern, eine EndDateTime un den Endzeitpunkt zu speichern, ein CreateDateTime um den Zeitpunkt zu speichern an dem das Objekt erstellt wurde, ein ModifiedDateTime um den Zeitpunkt der letzten Änderung zu speichern, ein IsActive um zu speichern ob das Objekt aktiv ist oder nicht, ein Text für eine Beschreibung des Objektes, ein Type für den Typ des Objektes hier immer Account, einen Creator in dem der Erschaffer gespeichert ist, einen Owner für den momentanen Besitzer, ein State für den Status des Accounts, eine ExpenseList für alle Ausgaben und eine TimesliceList für alle Zeitaufwände.

## Expense-Klasse

Die Expense Klasse wird von der InteractionFactory aufgerufen und ist das Expense-Objekt, dass für das Speichern der Expense Informationen verwendet wird, welche vom User über das GUI eingegeben werden.

In der Expense-Klasse sind folgende Eigenschaften enthalten:

Eine ID um alle Expense voneinander unterscheiden zu können, ein CreateDateTime um den Zeitpunkt zu speichern an dem das Objekt erstellt wurde, ein ModifiedDateTime um den Zeitpunkt der letzten Änderung zu speichern, ein IsActive um zu speichern ob das Objekt aktiv ist oder nicht, ein Text für eine Beschreibung des Objektes, ein Type für den Typ des Objektes hier immer Expense, ein State für den Status der Expense, einem Target in dem ist das Objekt gespeichert ist zudem die Expense gehört, einem Receipt in dem ein Bild der Rechnung oder Quittung gespeichert werden kann, einem Value für die Höhe der Ausgabe, einen Creator in dem der Erschaffer gespeichert ist und einen Owner für den momentanen Besitzer.

## Timeslice-Klasse

Die Timeslice Klasse wird von der InteractionFactory aufgerufen und ist das Timeslice-Objekt, dass für das Speichern der Timeslice Informationen verwendet wird, welche vom User über das GUI eingegeben werden.

In der Timeslice-Klasse sind folgende Eigenschaften enthalten:

Eine ID um alle Timeslice voneinander unterscheiden zu können, einer Duration für die verwendete Zeit in Minuten, einem User für den User der die zeit investiert hat, einem Target in dem ist das Objekt gespeichert ist zudem die Timeslice gehört, ein CreateDateTime um den Zeitpunkt zu speichern an dem das Objekt erstellt wurde und ein ModifiedDateTime um den Zeitpunkt der letzten Änderung zu speichern.

## InteractionFactory-Klasse

Die InteractionFactory Klasse wird aufgerufen um ein Objekt zu kreieren mit Hilfe einer der Methoden die in der Klasse enthalten sind. Die Klasse ist dafür zuständig, dass immer eine neue ID erzeugt wird, wenn keine mit den Daten mitgegeben wurde. Anschliessend gibt sie das Erschaffene Objekt zurück.

Die InteractionFactory ist zuständig für das erschaffen der Folgenden Objekte:

* Projekt-Objekt (Siehe 8.1 Project-Klasse)
* Task-Objekt (Siehe 8.2 Task-Klasse)
* Survey-Objekt (Siehe 8.3 Survey-Klasse)
* Account-Objekt (Siehe 8.4 Account-Klasse)
* Expense-Objekt (Siehe 8.5 Expense-Klasse)
* Timeslice-Objekt (Siehe 8.6 Timeslice-Klasse)

## BackendService-Klasse

Die BackendService Klasse wird vom DataManager aufgerufen und ist verantwortlich für den Datenaustausch zwischen der Datenbank des Servers und des Tools. Sie dient dazu um benötigte Daten über deren ID von der Datenbank abzurufen und an die Klasse weiterzuleiten von der der Aufruf kam. Wenn sich ein User einloggt liegt es ebenfalls in der Verantwortung dieser Klasse die eingegebenen Daten mit den Daten auf der Datenbank abzugleichen und falls nicht vorhanden eine Fehlermeldung zurück zu geben. Ebenfalls ist sie dafür verantwortlich die Daten von neu erschaffenen oder geänderten Objekten auf die Datenbank zu übertragen. Die erhaltenen Daten werden falls nötig noch zum richtigen Typ Konvertiert damit die Datenbank mit den erhaltenen Daten was anfangen kann.

## DataManager-Klasse

Die DataManager Klasse wird vom ObjectManager aufgerufen um die mitgegebenen Daten an dem BackendServiec Klassen weiterzuleiten. Sie wird benötigt um Daten für schreiben auf die Datenbank weiterzugeben, aber auch um Daten zum Abrufen von Daten weiterzugeben zum Beispiel über deren ID. Beim Abrufen von Daten werden die erhaltenen Daten noch in das richtige Format Konvertiert bevor diese an die InteractionFactory weitergegeben werden. Die dann erhaltenen Objekte werden schliesslich wieder zurück an den ObjectManager gegeben.

Eine kleine Ausnahme sind die Vote- und SurveyOption-Objekte, diese werden nach der Abfrage von der Datenbank nicht wie die anderen Objekte an die InteractionFactory weitergegeben, sondern an die ObjectFactory die für diese zuständig ist.

## ObjectManager-Klasse

Die ObjectManager Klasse wird hauptsächlich vom UIManager aufgerufen und ist für das Managen aller Objekte zuständig. Sie dient dazu um alle Daten an die zuständige Manager Klasse weiter zu leiten.

Beim Login eines Users, sorgt der ObjectManager dafür das alle Objekte mit denen der User zutun hat von der Datenbank in den jeweiligen Listen gespeichert werden.

Werden vom User Daten für ein neues Objekt eingegeben oder ein bestehendes Editiert, leitet der ObjectManager die erhaltenen Daten an den DataManager weiter und speichert anschliessend das Objekt das dieser zurückgibt.

Möchte der User ein bestehendes Objekt öffnen, durchsucht der ObjectManager die Listen der Gespeicherten Daten und leitet das gefundene Objekt an den FormManager weiter.

Ebenfalls verantwortlich ist er für das Eventhandling. Die des Timers, des Object changes und des loggin und loggout eines Users.

In der ObjectManager-Klasse sind folgende Eigenschaften enthalten:

* UserContext Darin werden die Daten des eingeloggten Users gespeichert
* ProjectList Eine Liste in der alle Projekte des eingeloggten Users gespeichert werden. Die in denen er Creator ist und auch alle in denen er Mitglied ist.
* TaskList Eine Liste in der alle Tasks der Projekte des eingeloggten Users gespeichert werden.
* SurveyList Eine Liste in der alle Surveys der Projekte des eingeloggten Users gespeichert werden.
* ExpenseList Eine Liste in der alle Expenses der Projekte des eingeloggten Users gespeichert werden.
* TimesliceList Eine Liste in der alle Timeslices der Projekte des eingeloggten Users gespeichert werden.
* UserList Eine Liste in der alle User die an den Projekten des eingeloggten Users beteiligt sind gespeichert werden.

## FormManager-Klasse

Die FormManager Klasse wird vom ObjectManager aufgerufen und ist für die Form verantwortlich. Sie sorgt dafür das Daten über die Form angezeigt oder Daten von der Form für das Erzeugen von Objekten weitergeleitet werden.

Die vom ObjectManager erhaltenen Daten werden durch den FormManager in der richtigen WinForm oder DockPanel geöffnet und in die jeweils richtigen Controls abgefüllt.

## UIManager-Klasse

Die UIManager Klasse wird über das GUI direkt vom User angesteuert. Sie leitet alle erhaltenen Informationen und Befehle an den ObjectManager weiter. Sie überprüft ob die vom User eingegebenen Date Valid sind und füllt die created/modified Daten ab.

Ebenfalls Aktiviert oder Deaktiviert Der UIManager die controls anhand der Buttons die vom Users gedrückt werden.

# Glossar und Verzeichniss

# Beilagen