Hey I'm in

autor(en) : Pascal Honegger

version : 0.1

status : Draft

quelle : Atos

dokumentendatum : 06 März 2018

anzahl der seiten : 22

Änderungshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Beschreibung | Autor(en) |
| 0.1 | 06.03.2018 | Initiale Version erstellt | Pascal Honegger |
| 1.0 | 21.03.2018 | Erster Release & Abgabe | Pascal Honegger |

# Inhalt

**Teil 1**

1 Aufgabenstellung 4

2 Projektorganisation 5

2.1 Beteiligte Personen 5

2.2 Projektmanagementmethode 5

2.3 Backup-Konzept 5

3 Ausgangslage 6

3.1 Vorkenntnisse 6

3.2 Vorarbeit 6

3.3 Firmenstandards 6

4 Zeitplanung 7

4.1 Meilensteine 7

4.2 Verhalten im Falle von Verzug 7

4.3 Gantt-Diagramm 8

5 Arbeitsprotokoll 9

5.1 Tag 1 (06.03.2018) 9

6 Kurzfassung 10

6.1 Kurze Ausgangssituation 10

6.2 Umsetzung 10

6.3 Ergebnis 10

**Teil 2**

7 Informieren 11

7.1 Ist-Analyse 11

7.2 Soll-Analyse 11

7.3 Systemübersicht 11

8 Planen 12

8.1 Use-Cases 12

8.2 Testkonzept 13

8.3 Software-Architektur 14

8.4 Datenbankdesign 15

9 Entscheiden 16

10 Realisieren 17

10.1 Backend-Klassenübersicht 17

10.2 Klassenname 17

10.3 Frontend-Klassenübersicht 17

11 Kontrollieren 18

11.1 Zeitmanagement 18

11.2 Tests 18

11.3 Testresultate: Browserkompatibilität 18

11.4 Testfazit 18

12 Auswerten 19

12.1 Reflexion 19

12.2 Erfahrungen 19

12.3 Einsetzbarkeit 19

12.4 Erweiterungen 19

12.5 Schlusswort 19

**Anhang**

13 Glossar 20

14 Verzeichnisse 21

14.1 Abbildverzeichnis 21

14.2 Tabellenverzeichnis 21

15 Quellcode 22

15.1 Frontend 22

15.2 Backend 22

# Aufgabenstellung

Hier ist die detaillierte Aufgabenstellung abgebildet, welche vom Auftraggeber vorgegeben wird.

## Ausgangslage

Es gibt bereits zahlreiche grosse Anbieter für die Planung der Teilnehmer von Einzelevents. Einer der bekanntesten ist sicherlich doodle.com. Wiederkehrende Ereignisse werden hingegen nur sehr rudimentär unterstützt. Im Arbeitsumfeld von der Atos gibt es zahlreiche Ereignisse, welche sich regelmässig wiederholen und viele Rahmenbedingungen (z.B. der potentielle Teilnehmerkreis) unverändert bleiben.

## Detaillierte Aufgabenstellung

Ziel dieser IPA Arbeit ist die Erstellung einer responsive Web Applikation mit welcher die Benutzer der Rolle Organisator ihre Eventserien planen und publizieren können. Dabei können sie die Metadaten (Titel, Beschreibung, Ort, Einzeltermine, Kosten, Antwortmöglichkeiten für die Teilnehmer) erfassen, die Notifikationen an die potentiellen Teilnehmer festlegen und die Antworten und Kosten der Teilnehmer verwalten. Termine eines Events können vom Organisator abgesagt, verschoben oder neu erstellt werden.

Benutzer mit der Rolle Teilnehmer können sich für eine Eventserie registrieren und zu jedem Termin ihre Teilnahme planen. Über eine Kommentarfunktion können die Teilnehmer zusätzliche Informationen zu ihrer Teilnahme bereitstellen. Sie können die Notifikationen der Organisatoren nutzen, um sich an einen bevorstehenden Termin erinnern zu lassen und sich eine Zusammenfassung (Metadaten + Teilnehmer) des bevorstehendes Termins zusenden zu lassen. Die Notifikationen werden per Mail versendet und sollen einen temporär gültigen Zugangslink (inkl. Authentisierung) zur Web Applikation enthalten.

Für Benutzer ist ein Self-Service zum Anlegen, Löschen und Passwort zurücksetzen erforderlich.

In den Vorarbeiten werden zusammen mit dem Auftragsgeber GUI Mockups erstellt. Der Auftraggeber soll sich dabei auf die Anforderungen konzentrieren, während es die Aufgabe des Kandidaten sein soll, diese Anforderungen benutzerfreundlich umzusetzen. Die Mockups sind nicht fix und können während der Arbeit durch den Kandidaten verbessert werden.

Folgende Features sind optional und werden in einem späteren Release nachgeliefert (grau hinterlegt in den Mockups):

* Kommentarfunktion
* Verwaltung der Kosten

### Fehlerbehandlung

* In keinem Fall darf das Backend abstürzen
* Alle Fehler müssen im Minimum geloggt werden
* Exceptions müssen mit den "Inner Exceptions" und allen Callstacks geloggt werden
* Fehler sollen dem Benutzer in einem für ihn verständlichen Format angezeigt werden
* Alle Eingabefelder sollen validiert werden. Validierungsfehler werden dem Benutzer angezeigt.

### Ereignisprotokoll / Logging

* Wichtige Ereignisse sollen in eine eigene Textdatei geschrieben werden und mindestens 6 Montate erhalten bleiben (Audit-Log). Diese sind:
* Benutzer erstellen / löschen
* Benutzer loggt sich ein / loggt sich aus
* Änderungen an Terminen und Zusagen
* Normale Logs sollen:
* begrenzt auf maximal 5 Dateien à 20MB sein
* sollten für die Auswertung möglichst maschinenlesbar sein (z.B. per logstash). Mögliche Formate: JSON, XML oder Text per Regex parsbar
* Wichtige Eigenschaften sollten zur besseren Auswertung als "Log Context Properties" geloggt werden, z.B. Benutzername, Client Hostname, Dauer einer Anfrage
* Folgende Eigenschaften müssen in der Logdatei vorhanden sein: Zeit des Ereignisses, Severity (TRACE ... FATAL), Namespace mit Klasse, Methodenname, Threadname, Meldungstext und evt. Kontextvariablen
* Die Konfiguration des Loggers soll per Datei editierbar sein. Änderungen sollten ohne Neustart erkannt und angewendet werden. Die Datei muss nicht per GUI veränderbar sein.

### Weitere Anforderungen

* Browser: >= IE11, Chrome >= V63

### Erwartete Lieferartefakte

* Code abgelegt in GIT Projekt (z.B. privates GitHub Projekt)
* funktionsfähiges Kompilat in GIT abgelegt
* Applikation in Azure deployed
* Einfaches Deployment von Updates in die Azure Cloud
* Technische Dokumentation, Architekturbeschreibung

## Mittel und Methoden

Betriebsystem: Azure Windows Web-App (lokal Windows)

Versionsverwaltung: GIT

GUI: Farben und Logos gemäss Atos Firmenstandard

### Backend:

* C# / .NET 4.7.x
* ASP.NET Web API 2.0
* SQL Server 2017
* Entity Framework (OR Mapper)
* Visual Studio 2017
* Resharper
* DI mit Autofac

### Frontend:

* HTML 5
* Angular 5
* Typescript
* Webpack
* Material Design

# Projektorganisation

## Beteiligte Personen

|  |  |  |  |
| --- | --- | --- | --- |
| Person | Rolle | Aufgaben / Verantwortung | E-Mail-Adresse |
| Christoph Fauti | Auftraggeber | Betrieb der Software | [christoph.fauti@atos.net](mailto:christoph.fauti@atos.net) |
| Rolf Fux | Verantwortliche Fachkraft | Unterstützung Kandidat | [rolf.fux@atos.net](mailto:rolf.fux@atos.net) |
| Pascal Honegger | Kandidat | Umsetzung der Arbeit | [pascal.honegger@bluewin.](mailto:pascal.honegger@bluewin.)ch |
| Guido Jansen | Hauptexperte | Festlegung Termine, Bewertung der Arbeit | [guidojansen@gmx.ch](mailto:guidojansen@gmx.ch) |
| Philip Karrer | Zweit-Experte | Unterstützung Hauptexperte | [pipo.karrer+pkorg@gmail.com](mailto:pipo.karrer+pkorg@gmail.com) |
| Tilo Steiger | Validierungsexperte | Validierung der Aufgabenstellung | [steiger@id.ethz.ch](mailto:steiger@id.ethz.ch) |
| Johan Widmer | Chefexperte | Sicherstellung korrekter Ablauf der IPA | [widmer@atap.ch](mailto:widmer@atap.ch) |

Tabelle 1: Projektteilnehmer

## Projektmanagementmethode

Ich verwende die Projektmanagementmethode IPERKA (**I**nformieren, **P**lanen, **E**ntscheiden, **R**ealisieren, **K**ontrollieren, **A**uswerten). Diese Wasserfallmethode hat den Vorteil, dass die Struktur gut mit den Vorgaben der IPA übereinstimmen. Ausserdem habe ich sowohl im Betrieb wie in der Schule Erfahrungen mit IPERKA sammeln können.

Als Alternative wurde die agile Projektmanagementmethode Scrum verworfen, da Nutzen und Aufwand nicht im Verhältnis stehen. Zwei grosse Vorteile von Scrum sind das iterative Anpassen der Anforderungen und die transparente Kommunikation im Entwicklerteam. Da dieses Projekt von mir alleine umgesetzt wird und die Anforderungen klar spezifiziert sind, haben diese Vorteile hier keine Anwendung. Deshalb wäre der Mehraufwand der formalen Scrum-Meetings zu gross.

## Backup-Konzept

Um die Chance eines Datenverlusts zu minimieren sind regelmässige Backups essenziell. Teil des Backups ist das gesamte Repository, namentlich der Quellcode und der IPA-Bericht. Zur Durchführung der Backups werden in einem ersten Schritt regelmässig Commits erstellt. Ein solcher Commit stellt einen Entwicklungsstand dar, auf welchen man jederzeit zurückgreifen kann. Diese Commits werden mindestens zwei Mal am Tag auf dem externen Anbieter GitHub gesichert. Zusätzlich wird täglich ein komplettes Backup auf einen USB-Stick geladen, welcher nur zur Zeit des Backups mit dem Laptop verbunden ist.

Im Falle eines Laptopfehlers kann eine relative aktuelle Version von GitHub geladen werden. Sollte dies nicht funktionieren wird auf eines der täglichen Backups auf dem USB-Stick zurückgegriffen.

# Ausgangslage

## Vorkenntnisse

* C# (2 Jahre)
* Visual Studio (2 Jahre)
* HTML 5 (1 Jahr)
* Angular (1 Jahr)
* TypeScript (1 Jahr)
* GIT (2 Jahre)
* DI (2 Jahre)

## Vorarbeit

### GUI Mockups

Zusammen mit dem Auftraggeber Christoph Fauti wurden Mockups erarbeitet. Die konkreten Mockups sind im Kapitel «7.2 Soll-Analyse» aufgelistet und erläutert. Als Vorarbeit wurden ausschliesslich die Grafiken ohne schriftliche Beschreibung erstellt. Die Grau markierten Bereiche sind nicht Teil der IPA und werden in einem späteren Release implementiert.

### Frontend-Logging

Als Vorarbeit wurde angeschaut, wie man Log-Einträge des Browsers zum Server überträgt. Dies wurde jedoch als zu aufwendig erachtet und verworfen.

### Senden von Mails

Zum Erarbeiten von Know-how wurde eine kleine Applikation erstellt, welche mithilfe der SendGrid-Plattform E-Mails versendet. SendGrid ist ein Dienst, welcher das kostenlose Versenden von Mails erlaubt, solange man sein Kontingent nicht überschreitet.

### Projekt

Analog zu meinen anderen Projekten basiert die Angular-Anwendung auf dem [Angular-Starter](https://github.com/AngularClass/angular-starter) GitHub-Projekt. Als Vorarbeit habe ich dieses Starter-Projekt kopiert und mit [Angular Material](https://github.com/angular/material2) ein „Hallo Welt“ Projekt erstellt, welches ich als Grundlage verwende. Nebenbei wurde eine simple Word-Vorlage für die IPA aufbereitet, welche auch für sonstige Dokumentationen verwendet wird.

## Firmenstandards

### OR-Mapper

Zur Datenbankanbindung wird das Entity Framework 6 von Microsoft eingesetzt. Dieses direkt in die IDE integrierte OR-Mapping ist Atos-Firmenstandard für C#-basierte Projekte.

### Frontend

Die IDE-Konfigurationen wurden von der Abteilung übernommen, damit der korrekte Coding-Style eingehalten wird.

### Farbschema

Als Primärfarbe wurde die Farbe «Atos-Primary #0066A1» gewählt (Dunkles Blau). Als Akzentfarbe wurde die Farbe «Atos-Secondary #A626AA» gewählt (Violett).

# Zeitplanung

## Meilensteine

|  |  |  |
| --- | --- | --- |
| Meilenstein | Beschreibung | Datum |
| Projektbeginn | Start des Projektes | 06.03.2018 09:30 |
| Informieren | Aufgabenstellung fertig analysiert | 06.03.2018 16:00 |
| Planen | Weiteres Vorgehen, grundlegende Architektur und Testfälle geplant | 08.03.2018 12:00 |
| Entscheiden | Für konkrete Umsetzungen entschieden, wo mehrere Lösungswege existieren | 08.03.2018 17:00 |
| Implementation | Applikation funktional abgeschlossen | 16.03.2018 12:00 |
| Realisieren | Umsetzung fertig dokumentiert | 20.03.2018 10:00 |
| Kontrollieren | Applikation durch zuvor definierte Testfälle kontrolliert | 20.03.2018 17:00 |
| Auswerten | Projekt reflektiert und ein Fazit gezogen | 21.03.2018 14:00 |
| Projektabschluss | Erfolgreicher Projektabschluss | 21.03.2018 16:00 |

Tabelle 2: Meilensteine

## Verhalten im Falle von Verzug

Falls Meilensteine nicht erreicht werden können, müssen Aktionen getroffen werden. Primär werden nicht kritische Teilaufgaben vernachlässigt, damit die Kernaufgaben erfolgreich abgeschlossen werden können. Die Teilaufgaben in der Reihenfolge, in welcher sie vernachlässigt werden:

1. Responsive-Design
2. Erweiterte Logs
3. Browser-Kompatibilität
4. Löschen eines Profils
5. Löschen eines Events.

## Gantt-Diagramm



Abbildung 1: Zeitplan

# Arbeitsprotokoll

## Tag 1 (06.03.2018)

### Planung

Heute möchte ich mit dem Projekt beginnen, die Zeitplanung abschliessen und die erste IPERKA-Phase (Informieren) abschliessen. Die restliche Zeit investiere ich in die Use-Cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Projektbeginn | 15min | 15min | ✔ |
| Zeitplanung & Meilensteine | 2h 15min | 1h 30min | ✔ |
| Projektorganisation dokumentieren | 30min | 15min | ✔ |
| Ausgangslage dokumentieren | 30min | 30min | ✔ |
| Ist- und Soll-Analyse | 1h | 2h |  |
| Systemübersicht | 30min | 30min |  |
| Use-Cases | 2h 30min |  |  |
| Arbeitsprotokoll | 30min |  |  |

Tabelle 3: Konkrete Planung Tag 1

### Erledigte Arbeit

Zusammenfassung der Tabelle

Bsp. GitHub aufgesetzt, Komponenten realisiert etc.

### Probleme

Bsp. Zeitplant zu knapp, unerwartete Fehler…

### Reflexion

Fazit / Gelerntes

# Kurzfassung

## Kurze Ausgangssituation

## Umsetzung

## Ergebnis

# Informieren

## Ist-Analyse

Im Arbeitsumfeld der Atos gibt es zahlreiche Events, welche sich regelmässig wiederholen. Ein konkretes Beispiel hierfür ist das wöchentliche Badminton. Christoph Fauti, der Auftraggeber dieses Projekts und Organisator des wöchentlichen Badmintons, hat vor mehreren Jahren bereits ein Tool zur Planung solcher Events realisiert. Dieses web basierte Tool ist jedoch technisch nicht mehr auf dem neusten Stand und besitzt beispielsweise kein mobilefreundliches Design, weshalb es abgelöst wird. Als Alternative gibt es den grossen Anbieter Doodle, welcher die Terminfindung leicht gestaltet. Jedoch fehlen auf Doodle Komfortfunktionen wie das schnelle Erfassen von wiederholenden Terminen oder eine integrierte Kostenverwaltung.

## Soll-Analyse

Eine genauere Analyse der Anforderungen und des Soll-Zustands. Das Frontend umfasst die Angular-Anwendung, welche im Browser läuft, und das Backend umfasst die Serverlogik, welcher auf Azure gehostet wird.

### Frontend – Benutzer Self-Service

Benutzer können sich anmelden oder ein neues Benutzerkonto anlegen. Zusätzlich können registrierte Benutzer ihr Passwort zurücksetzen, wobei ihnen eine E-Mail mit den benötigten Informationen zugesendet wird. Angemeldete Benutzer können auch ihre persönlichen Daten aktualisieren oder ihr Benutzerkonto entgültig löschen. Bei einer solchen Löschen werden auch alle Teilnahmen und organiserten Events gelöscht.

Abbildung 2: Mockup Login / Registrieren

Abbildung 3: Mockup Profil

Abbildung 4: Mockup Passwort zurücksetzen

### Frontend – Events Sicht Teilnehmer

Alle Benutzer können als Teilnehmer bei einem Event mitmachen. Lediglich für private Events wird eine explizite Einladung benötigt. Für die weiteren Events kann jeder Benutzer über die Seite «Events» (Startseite) einen Event finden und an diesem teilnehmen. Sobald man an einem Event teilnimmt wird dieser unter «Meine Events» aufgeführt. Um an einem Event teilzunehmen kann ein Benutzer entweder an einem Termin teilnehmen oder explizit «Zu meine Events hinzufügen» klicken. Um das Interesse an einem Event wieder zu wiederrufen kann dieser wieder verlassen werden, wobei zukünftige Teilnahmen aufgelöst werden. Um die Anzahl Interaktionen zu optimieren kann man auf der Übersichtsseite direkt an dem nächsten Termin teilnehmen, sofern dieser existiert.

In der Detailansicht kann jeder Teilnehmer zusätzlich die nächsten Termine sehen und bereits zukünftigen Terminen zu- oder absagen. Optional kann jeder Teilnehmer Notifikationen deaktivieren, welcher er nicht bekommen möchte.

Abbildung 5: Mockup Event Details

Abbildung 6: Mockup Events

### Frontend – Events Sicht Organisator

Organisatoren können neben der Teilnehmer-Ansicht auch eine Organisator-Ansicht öffnen, welche zum Bearbeiten oder Erstellen von Events verwendet wird. In dieser können sie allgemeine Informationen wie die Beschreibung oder den Titel ändern. Auch kann der Organisator den Event als privat deklarieren, woraufhin nur eingeladene Personen teilnehmen dürfen. Falls ein Event nie mehr stattfinden wird kann man diesen auch löschen. Über Änderungen an diesen Eigenschaften werden alle Teilnehmer informiert.

Neben dem Event an sich können auch die Termine verwaltet werden. So kann ein Administrator neue Termine erfassen oder die bestehenden absagen. Beim Erfassen von neuen Terminen werden alle Startdaten mit Enter getrennt erfasst. Dies hat den Vorteil, dass man über Programme wie Excel sehr effizient duzende von Daten erfassen kann, ohne sich mit Datumsauswahlen herumzuschlagen. Falls man jedoch nur einen einzelnen Termin erfassen möchte kann man direkt in einem Textfeld das Datum und die Uhrzeit erfassen.

Da die zeitlichen Abstände zwischen Events stark variieren kann darf jeder Organisator die Zeitperiode für Reminder und die Zusammenfassung selbst festlegen. Der Reminder ist personalisiert und beinhaltet die Informationen zum Event und ob man (der Empfänger des Reminders) zugesagt hat. Die Zusammenfassung ist für alle identisch und beinhaltet eine Übersicht der Teilnehmer. Für jede Änderung an der Teilnehmerliste nach dem Versenden der Zusammenfassung wird eine aktualisierte Zusammenfassung versendet.

Der Organisator kann ausserdem die Teilnahme pro Termin und die Teilnahme an dem Event für alle Teilnehmer ändern. Hintergedanke hierbei ist, dass ein Organisator inaktive Benutzer entfernt oder im Falle von Krankheit für einen Teilnehmer absagt.

### Backend

Auf Azure gehostete REST-Schnittstelle, realisiert mit ASP.net WebApi 2. Alle oben beschriebenen Benutzeroberflächen interagieren mit dem Backend um Daten zu laden oder anzupassen.

### Datenbank

Die Daten der Events und Benutzer werden durch das Backend in eine MSSQL-Datenbank persistiert. Die Datenbank wird ebenfalls auf Azure gehostet.

### Notifikationen (E-Mail)

Als Kommunikationskanal ist E-Mail vorgesehen. Innerhalb jeder Notifikations-E-Mail sollte ein Zugangslinks eingebunden sein, welcher einen automatisch anmeldet. Diese Zugangslinks sollten aus Sicherheitsgründen nach einer gewissen Zeitspassen ablaufen.

Abbildung 7: Mockup Event Bearbeitung

## Systemübersicht

Grundsätzlich gibt es drei Systeme in diesem Projekt:

* Darstellung durch das Frontend
* Aufbereitung durch das Backend
* Datenpersistenz durch die Datenbank

Diese drei Systeme müssen über ihre jeweiligen Grenzen hinaus miteinander kommunizieren. Durch die Verwendeten Frameworks kommunizieren die Systeme jeweils mit einer internen Schnittstelle. Diese internen Kommunikationen sind in Abbildung 8: Systemübersicht mit durchgezogenen Verbindungen dargestellt. So kommuniziert die Angular-App (Frontend) mit dem Angular HttpClient, welcher über HTTPS eine REST-Anfrage startet. Diese Anfrage wird vom Webserver entgegengenommen und vom ASP.net Framework interpretiert. Danach wird erst die Web-App (Backend) aktiv, welche über das Entity Framework die Daten der Datenbank verarbeitet. Zusätzlich hat das Backend noch eine Verbindung zu den Umgebungsvariablen, welche sensitive Konfigurationswerte wie Datenbankanbindungen oder API-Schlüssel beinhalten.

Abbildung 8: Systemübersicht

# Planen

## Use-Cases

### Use-Case Diagramm

Beschreibung zur Aufteilung des Diagrammes.

### Use-Cases 1: Gruppe 1

Alle Use-Cases in diesem Kapitel werden von Testbenutzern durchgeführt.

|  |  |
| --- | --- |
| Use-Case 1.1: Tagesabhängige Gefühlslage sehen | |
| Beschreibung | Eine aussagekräftige Beschreibung |
| Ergebnis / Nachbedingung | Detaillierte Beschreibung des erwarteten Ergebnisses |
| Vorbedingung | Vorbedingungen für diesen Use-Case |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Erster Schritt | | 2 | Weiterer Schritt | |

Tabelle 8: Use-Case 1.1

## Testkonzept

Das Testkonzept beschreibt wie nach der Realisierung sichergestellt wird, dass die Funktionalität funktioniert. Dazu werden Testfälle definiert und später in der Kontrollieren-Phase durchgeführt. Die Testfallnummer besitzt als Präfix die Nummer des dazugehörigen Use-Cases.

### Eingesetzte Testmittel und -Methoden

Alle Tests werden, ausser explizit angegeben, mit folgender Hardware & Software durchgeführt:

* Dell-Laptop mit 16 Gigabyte Arbeitsspeicher, einem Intel i7 Prozessor und fungierender Internetverbindung
* Google Chrome **VERSION**
* Windows 7 (Systemsprache: Deutsch)

### Testfälle 1.1: Gruppe 1

Diese Tests können von Benutzern und Administratoren durchgeführt werden.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 1.1.1 | Der Online-Rechner ist konfiguriert | |  |  | | --- | --- | | 1 | Startseite öffnen | | 2 | «1 + 1» eingeben | | 3 | «Rechnen» drücken | | Die Zahl 2 wird dargestellt |

Tabelle 10: Testfälle 1.1 – Gruppe 1

## Software-Architektur

In diesem Kapitel wird die allgemeine Architektur festgelegt. Zwar werden noch keine Klassendiagramme festgelegt, doch der allgemeine Aufbau der einzelnen Komponenten und ihr Zusammenspiel wird festgehalten.

### Übersicht

\*Komponenten- / Klassendiagramm\*

Das Backend ist als REST Service umgesetzt und bietet eine REST-Schnittstelle an, welche über HTTP(S) und JSON kommuniziert. Alle im Controller aufgeführten Methoden können von aussen angesprochen werden und bilden die Schnittstelle zum Frontend.

## Datenbankdesign

In diesem Kapitel wird die Struktur der Datenbank festgelegt. Die Struktur der MSSQL-Datenbank wird direkt im Code erstellt und danach durch das Entity Framework in Tabellen umgewandelt. Hierbei werden auch Migrationen beim Umbau des Models und das korrekte Mapping von Klassen- zu Datenbanktypen übernommen.

### ERD

\*Diagramm\*

### Entitätsbeschreibungen

|  |  |
| --- | --- |
| TableName (Gefühlslage) | |
| Beschreibung | Was wird in dieser Tabelle gespeichert? |
| Verbindungen | Welche Relationen bestehen weshalb? |
| Primärschlüssel | id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | id | INT | Die eindeutige Identifikation | Pflichtfeld  Generiert | | created\_date | DATE | Erstelldatum | Pflichtfeld | | Example | VARCHAR(250) | Dummy Textfeld |  | |

Tabelle 12: Entitätsbeschreibung TableName

# Entscheiden

In diesem Kapitel geht um die getroffenen Entscheidungen zu verwendeten Frameworks und dessen Versionen.

# Realisieren

In diesem Kapitel werden auf die Details der Implementierung eingegangen.

## Backend-Klassenübersicht

Kurze Beschreibung zu den Kernkonzepten im Backend.

## Klassenname

Beschreibung

## Frontend-Klassenübersicht

Kurze Beschreibung zu den Kernkonzepten im Frontend.

### Klassenname

Beschreibung

# Kontrollieren

In diesem Kapitel wird die Zielerreichung kontrolliert und entschieden, ob das Projekt abgeschlossen ist.

## Zeitmanagement

|  |  |  |
| --- | --- | --- |
| Meilenstein | Gesetzes Datum | Effektives Datum |
| Projektbeginn | 06.03.2018 09:30 | 06.03.2018 09:30 |
| Informieren | 06.03.2018 16:00 |  |
| Planen | 08.03.2018 12:00 |  |
| Entscheiden | 08.03.2018 17:00 |  |
| Implementation | 16.03.2018 12:00 |  |
| Realisieren | 20.03.2018 10:00 |  |
| Kontrollieren | 20.03.2018 17:00 |  |
| Auswerten | 21.03.2018 14:00 |  |
| Projektabschluss | 21.03.2018 16:00 |  |

## Tests

Die Anforderungen wurden alle umgesetzt. Zur Überprüfung der umgesetzten Anforderungen werden die geplanten Tests durchgeführt.

✔ = Bestanden

✔ (**~**) = Bestanden mit Abweichung

✘ = Fehlgeschlagen

### Testresultate 1: Startseite

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 1.1 | ✔ |  |
| 1.2 | ✔ (**~**) | Was war nicht wie erwartet? |
| 1.3 | ✘ | Was lief schief? |

Tabelle 15: Testresultate 1

## Testresultate: Browserkompatibilität

|  |  |  |
| --- | --- | --- |
| Browser | Kommentar | Unterstützt |
| Mozilla Firefox 58 |  | ✔ |
| Google Chrome 64 |  | ✔ (**~**) |
| Internet Explorer 11 |  | ✘ |

Tabelle 17: Testresultate Browserkompatibilität

Der minimale Browsersupport ist gewährleistet.

## Testfazit

Anzahl bestandener Testfälle

Details zu fehlgeschlagenen Testfällen

Fazit: Ist Programm verwendbar?

Rechtfertigung Fazit

# Auswerten

Das komplette Projekt wird nochmals reflektiert und ausgewertet.

## Reflexion

Rückblick

## Erfahrungen

Gelerntes

## Einsetzbarkeit

Kann die Software verwendet werden? Wie wird sie verwendet werden?

## Erweiterungen

Ist die Erweiterbarkeit gewährleistet?

## Schlusswort

# Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Azure | Die Cloud- Plattform von Microsoft, auf welcher unter anderem Datenbanken und Webanwendung gehostet werden können. |

Tabelle 18: Glossar

# Verzeichnisse

## Abbildverzeichnis

Die Abbildungen in diesem Dokument sind alle selbst erstellt.

[Abbildung 1: Zeitplan 8](file:///C:\Users\Pascal\Desktop\HeyImIn\Dokumentation\IPA-Dokumentation_Vorlage.docx#_Toc508050375)

## Tabellenverzeichnis

[Tabelle 1: Projektteilnehmer 5](#_Toc508050366)

[Tabelle 2: Meilensteine 7](#_Toc508050367)

[Tabelle 3: Konkrete Planung Tag 1 9](#_Toc508050368)

[Tabelle 8: Use-Case 1.1 13](#_Toc508050369)

[Tabelle 10: Testfälle 1.1 – Gruppe 1 13](#_Toc508050370)

[Tabelle 12: Entitätsbeschreibung TableName 15](#_Toc508050371)

[Tabelle 15: Testresultate 1 18](#_Toc508050372)

[Tabelle 17: Testresultate Browserkompatibilität 18](#_Toc508050373)

[Tabelle 18: Glossar 20](#_Toc508050374)

# Quellcode

## Frontend

### File.html

## Backend

### File.cs