Hey, I'm in

autor(en) : Pascal Honegger

version : 1.0

status : Final

quelle : Atos

dokumentendatum : 21 März 2018

anzahl der seiten : 63

Änderungshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Beschreibung | Autor(en) |
| 0.1 | 06.03.2018 | Initiale Version erstellt | Pascal Honegger |
| 0.2 | 06.03.2018 | Zeitplan & Informieren abgeschlossen | Pascal Honegger |
| 0.3 | 08.03.2018 | Planen abgeschlossen | Pascal Honegger |
| 0.4 | 08.03.2018 | Entscheiden abgeschlossen | Pascal Honegger |
| 0.5 | 16.03.2018 | Implementieren abgeschlossen | Pascal Honegger |
| 0.6 | 20.03.2018 | Realisieren abgeschlossen | Pascal Honegger |
| 0.7 | 20.03.2018 | Kontrollieren abgeschlossen | Pascal Honegger |
| 0.8 | 21.03.2018 | Auswerten abgeschlossen | Pascal Honegger |
| 1.0 | 21.03.2018 | Erster Release & Abgabe | Pascal Honegger |

# Inhalt

**Teil 1**

1 Aufgabenstellung 4

1.1 Ausgangslage 4

1.2 Detaillierte Aufgabenstellung 4

1.3 Mittel und Methoden 5

1.4 Vorkenntnisse 6

1.5 Vorarbeiten 6

1.6 Neue Lerninhalte 6

1.7 Arbeiten in den letzten 6 Monaten 6

2 Projektorganisation 7

2.1 Beteiligte Personen 7

2.2 Projektmanagementmethode 7

2.3 Backup-Konzept 7

3 Detaillierte Ausgangslage 8

3.1 Vorkenntnisse 8

3.2 Konkretisierung der Vorarbeit 8

3.3 Firmenstandards 8

4 Zeitplanung 9

4.1 Meilensteine 9

4.2 Verhalten im Falle von Verzug 9

4.3 Gantt-Diagramm 10

5 Arbeitsprotokoll 11

5.1 Tag 1 (06.03.2018) 11

5.2 Tag 2 (07.03.2018) 12

5.3 Tag 3 (08.03.2018) 13

5.4 Tag 4 (09.03.2018) 14

5.5 Tag 5 (13.03.2018) 15

5.6 Tag 6 (14.03.2018) 16

5.7 Tag 7 (15.03.2018) 17

5.8 Tag 8 (16.03.2018) 18

5.9 Tag 9 (20.03.2018) 19

5.10 Tag 10 (21.03.2018) 20

**Teil 2**

6 Kurzfassung 21

6.1 Kurze Ausgangssituation 21

6.2 Umsetzung 21

6.3 Ergebnis 21

7 Informieren 22

7.1 Ist-Analyse 22

7.2 Soll-Analyse 22

7.3 Systemübersicht 25

8 Planen 26

8.1 Use-Cases 26

8.2 Testkonzept 33

8.3 Software-Architektur 38

8.4 Datenbankdesign 40

8.5 Logging-Konzept 44

9 Entscheiden 45

9.1 Authentifizierung 45

9.2 Umgang mit Zeitzonen 45

9.3 Umgang mit sensitiven Konfigurationswerten 45

9.4 Cron-Jobs 46

9.5 Verwendete Tools / Frameworks 46

10 Realisieren 47

10.1 Deployment (Azure) 47

10.2 Übergreifend 47

10.3 Backend 48

10.4 Frontend 51

11 Kontrollieren 53

11.1 Zeitmanagement 53

11.2 Tests 53

11.3 Testresultate: Browserkompatibilität 56

11.4 Testfazit 56

12 Auswerten 57

12.1 Reflexion 57

12.2 Erfahrungen 57

12.3 Einsetzbarkeit 57

12.4 Erweiterungen 57

12.5 Schlusswort 57

**Anhang**

13 Glossar 58

14 Verzeichnisse 59

14.1 Abbildverzeichnis 59

14.2 Tabellenverzeichnis 59

15 Quellcode 61

15.1 Logdateien der Tests 61

15.2 Frontend 63

15.3 Backend 63

# 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 bevorstehenden 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 Monate 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 evtl. 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

Betriebssystem: 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

## Vorkenntnisse

Projekterfahrungen:

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

## Vorarbeiten

* Wissen für das Senden von Mails aufbauen.
* Erarbeitung GUI Mockups mit entsprechender Abstimmung mit Auftraggeber (welche Seiten braucht es, welchen Inhalt sollten sie haben, Entscheidung über Priorität bei der Umsetzung, genauere Definition der Anforderungen).
* Wissensaufbau Frontend-Logging anhand des aktuellen Projektes.

## Neue Lerninhalte

* MS SQL Server (keine Projekterfahrung, aber Schul- und Wettkampferfahrung)
* Entity Framework (keine Projekterfahrung, aber Schulprojekterfahrung)
* Mails senden

## Arbeiten in den letzten 6 Monaten

### Erster grosser Auftrag:

Backend:

* Implementierung von User-Management in C# und Web API 2.0
* Implementierung von Authentifizierung und Autorisierung

### Zweiter grosser Auftrag:

Frontend:

* GUI Entwicklung mit HTML und Angular 4
* Bootstrap 3
* Webpack
* Implementierung User-Management

# 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.

# Detaillierte Ausgangslage

## Vorkenntnisse

* Ein Jahr Entwicklung einer Client-Anwendung
* Anderthalb Jahre Entwicklung in einem Webprojekt
* Fünftägige Probe-IPA mit gleichem Technologie-Stack
* Diverse Schulprojekte in der Azure-Cloud gehostet

## Konkretisierung der 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 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, nur noch kleine Fehlerkorrekturen vonnöten | 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 zusammen mit der ersten IPERKA-Phase (Informieren) abschliessen und mit der Planung starten.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 2h 30min | ✘ |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 3: Konkrete Planung Tag 1

### Erledigte Arbeit

Heute war der erste Tag des Projektes. Nach dem (späten) Ankommen begann die Arbeit flott und ohne Probleme. Zu Beginn habe ich ein Git-Repository erstellt, damit ich meine Arbeit möglichst schnell auf GitHub sichern kann. Danach habe ich meine Vorlage kopiert und mit dem Zeitplan begonnen. Dabei habe ich festgestellt, dass die Zeit ziemlich genau aufgeht. Jedoch habe ich nur wenige Stunden für das Erstellen der Use-Cases und Testfälle eingeplant, da ich lieber mehr Zeit in die Realisierung investiere.

### Probleme

Leider war dieser erste Tag der IPA ein holpriger Einstig. Während der Anreise hatte ich Probleme mit dem Öffentlichen Verkehr, weshalb ich erst um 09:30 mit der Arbeit beginnen konnte. Zum Glück hatte ich heute keine fixen Termine und kann auch am Abend länger arbeiten, damit ich acht Arbeitsstunden erreiche. Während dem Arbeiten hatte ich Probleme bei der Darstellung des Use-Case-Diagramms. Da dieses Projekt viele Use-Cases umfasst, wusste ich nicht, wie ich diese darstellen sollte, ohne eine zu grosse oder unübersichtliche Grafik zu erstellen. Schliesslich habe ich mich für die erst beste Darstellungsform entschieden und diese beibehalten, damit ich nicht mehr Zeit verschwende.

Der Hauptexperte Guido Jansen wollte über PkOrg wissen, zu welcher Uhrzeit der erste Expertenbesuch stattfinden soll. Unglücklicherweise habe ich beim Antworten auf diese Frage eine Nachricht an den Chefexperten gesendet. Diese Art von Nachricht ist eigentlich für Beschwerden über den Hauptexperten gedacht. Zum Glück ist mir dieses Missgeschick aufgefallen, und ich konnte die Nachricht erneut über den korrekten Weg senden.

### Reflexion

Ausnahmsweise habe ich heute während der Arbeit Kopfhörer verwendet, um weniger abgelenkt zu werden. Da ich während dem Tag nur ein einziges Mal abgelenkt wurde, haben sich die Kopfhörer als gute Idee bewiesen, weshalb ich diese auch weiterhin verwenden werde.

Während der Planungsphase viel mir ein Stein vom Herzen als ich sah, dass die Zeitplanung aufgeht. Ich hatte befürchtet, dass ich bereits während der Planung auf Probleme stossen würde, was zum Glück nicht eingetreten ist. Alles in allem bin ich optimistisch, dass ich dieses grosse Projekt termingerecht fertig stellen werde. Jedoch muss hierfür die Implementation ohne grosse Problem funktionieren, da ich keine nennenswerte Zeitreserve eingeplant habe.

## Tag 2 (07.03.2018)

### Planung

Heute Morgen möchte ich meine Arbeit an den Use-Cases abschliessen und daraus Testfälle ableiten. Am Nachmittag beginne ich dann mit der Softwarearchitektur.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Use-Cases | 1h | 45min | ✔ |
| Testkonzept | 3h | 2h 30min | ✔ |
| Komponentenübersicht | 2h | 2h 15min | ✔ |
| Datenbankstruktur | 1h 30min | 2h | ✔ |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 4: Konkrete Planung Tag 2

### Erledigte Arbeit

Am Morgen konnte ich rasch meine Use-Cases abschliessen und mit den Testfällen beginnen. Hier musste ich Grenzen setzen, welche Grenzfälle alle getestet werden. So sind alle definierten Anforderungen inklusive realistischen Ausnahmefällen getestet. Dafür habe ich die Testfälle zu den Notifikations-Emails gekürzt und Testfälle für allgemeine Themen wie die Fehlerbehandlung, Logs und Input-Validierung weggelassen. Diese Themen werde ich zwar implementieren und testen, jedoch wäre der Mehraufwand der Dokumentation zu gross.

Bei der Softwarearchitektur konnte ich trotz Problemen bei der Darstellung der UML-Diagramme gute Fortschritte erzielen. So konnte ich mithilfe von <https://github.com/ErikEJ/EntityFramework6PowerTools> mein in Klassen formuliertes Datenbankmodell grafisch darstellen. So bin ich nun zu einem grossen Teil mit der Planung fertig und kann diese morgen vor dem ersten Expertenbesuch abschliessen.

### Probleme

Das Erstellen des ERMs hat viel mehr Zeit in Anspruch genommen als geplant war. Das Datenmodell verfügt über mehrere Relationen und diverse Felder. Zwar hatte ich berücksichtigt, dass ich das Modell Code-First, also direkt als Programmcode, erstellen möchte. Ich hatte jedoch nicht berücksichtigt, dass ich ein Test-Projekt erstellen, die Klassen dokumentieren und eine grafische Version exportieren muss. Zudem hatte ich Probleme beim Erarbeiten des logischen Modells, da ich diverse Problemfälle berücksichtigen musste. Damit ich dieses Datenmodell nicht mehr editieren muss, habe ich mir gleich überlegt, wie ich die diversen Anwendungsfälle implementieren könnte. Dadurch sollte ich bei der Realisierung das Datenmodell verwenden können, ohne es überarbeiten zu müssen.

Beispielsweise sollte ein Benutzer eine Zusammenfassungs-E-Mail nur einmal erhalten. Das Problem fällt erst auf, wenn man sich die Implementierung der Notifikationen vor Augen führt. Da ein Webserver immer eine Anfrage braucht, um etwas umzusetzen, werde ich auch hier durch einen Cron-Job regelmässige Anfragen starten. Das heisst, dass eine Notifikation nie exakt zur korrekten Zeit versendet wird, sondern erst, wenn die nächste Cron-Job-Anfrage eintrifft. Damit beim Überprüfen der zu sendenden Notifikationen keine Nachricht doppelt gesendet wird, ist bei einer Termin-Teilnahme auch gespeichert, ob bereits eine Zusammenfassung versendet wurde. Diese Wert kann ich ebenfalls für das Versenden von aktualisierten Zusammenfassungen verwenden, falls es kurzfristige Änderungen gegeben hat.

### Reflexion

Trotz des Verzugs mit den Use-Cases konnte ich heute die geplante Arbeit abschliessen. Leider musste ich zur Fertigstellung des Arbeitsprotokolls etwas länger als acht Stunden arbeiten, was ich morgen ausgleichen möchte. Ich blicke immer noch optimistisch auf dieses Projekt und denke, dass ich morgen das Projekt fertig aufsetzen und mit der Programmierung beginnen werde.

## Tag 3 (08.03.2018)

### Planung

Heute möchte ich die Planung abschliessen, das Expertengespräch führen und durch das Abschliessen der Entscheidungs-Phase in die Realisierungs-Phase übergehen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Logging-Konzept | 1h 15min | 2h | ✔ |
| Planung abschliessen | 15min | 30min | ✔ |
| Vorbereitungen Expertenbesuch | 15min | 15min | ✔ |
| Expertenbesuch | 1h 45min | 1h 15min | ✔ |
| Entscheiden – Authentifizierung | 45min | 30min | ✔ |
| Entscheiden – Sensitive Konfigurationswerte | 45min | 30min | ✔ |
| Entscheiden – Zeitzonen | 30min | 15min | ✔ |
| Entscheiden – Tools / Frameworks | 30min | 15min | ✔ |
| Projekteinrichtung inklusive DI | 1h | 1h 15min | ✔ |
| Entscheiden – Cron-Job | - | 30min | ✔ |
| Azure-Konfiguration | - | 30min | ✘ |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 5: Konkrete Planung Tag 3

### Erledigte Arbeit

Heute konnte ich die Dokumentationsphase abschliessen und bereits mit der Realisierung beginnen. Am Morgen wurde ich mit der Planung fertig und hatte den ersten Expertenbesuch. Am Nachmittag konnte ich meine Entscheidungen gut dokumentieren und früher als geplant mein Projekt aufsetzen. Auch dort gab es keine Probleme und ich konnte die beiden Projekte für das C#-Backend und Angular-Frontend aufsetzen. Die Dependency Injection habe ich nach der Autofac-Anleitung <http://autofaccn.readthedocs.io/en/latest/integration/webapi.html> konfiguriert, jedoch noch nicht testen können. Jedoch sollte dank dem übernommenen AssemblyScanner aus einem anderen Projekt alles funktionieren. Kurz vor Feierabend habe ich noch mit der Konfiguration für Azure begonnen, diese jedoch nicht mehr abschliessen können. Zuhause habe ich danach weitere fünfzehn Minuten investiert, damit das automatische Bereitstellen der ASP.NET Anwendung grundsätzlich funktioniert. Da ich die neusten Tools und somit die aktuellste Projektstruktur verwende, musste ich explizit angeben, dass ich die neue Version verwenden möchte. Glücklicherweise hat ein anderer GitHub-Benutzer unter <https://github.com/projectkudu/kudu/issues/2376#issuecomment-309936988> beschrieben, wie man die neue Version aktivieren kann.

### Expertenbesuch

Beim ersten Expertenbesuch trat ich auch zum ersten Mal persönlich in Kontakt mit meinem Hauptexperten Guido Jansen. Dieser wirkt sympathisch und hat mir und Rolf den weiteren Ablauf der IPA nähergebracht. Auch konnten alle offenen Fragen geklärt werden. Als ich den Stand meiner Arbeit präsentiert habe, konnte Herr Jansen fast nicht glauben, dass ich nach zwei Tagen bereits so viel geschafft habe. Mein schnelles Vorankommen ist auf jeden Fall auch den erstellten Mockups anzurechnen. Dank diesen hatte ich immer einen guten Referenzpunk für Fragen und eine grafische Darstellung der abstrakten Klassenmodelle. Eine direkte Verbesserung war der ausgedruckte Zeitplan, bei welchem die verschiedenen Blautöne kaum zu unterscheiden sind. Aus diesem Grund möchte ich die Farben bei Gelegenheit anpassen.

### Reflexion

Ich konnte heute den «langweiligen» aber auf keinen Fall unwichtigen Teil der Arbeit abschliessen und langsam in die Programmierung übergehen. Das Expertengespräch war sehr lehrreich und meine offenen Fragen konnten geklärt werden. Falls alles nach Plan verläuft, sollte ich nach dem morgigen Tag einen Grossteil des Servers so definiert haben, dass ich im Verlauf der nächsten Woche mit der Implementierung fertig werde.

## Tag 4 (09.03.2018)

### Planung

Heute möchte ich mit der Realisierung beginnen und die Authentifizierung inklusive Login und Registration abschliessen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Azure-Konfiguration abschliessen | 30min | 45min | ✔ |
| Logger konfigurieren und testen | 1h 15min | 2h 45min | ✔ |
| Authentifizierung als Pseudocode | 30min | 30min | ✔ |
| Authentifizierung im Backend implementieren | 45min | 1h | ✔ |
| Authentifizierung im Frontend implementieren | 1h | 45min | ✔ |
| Frontend Design & Routing | - | 45min | ✔ |
| Login / Registration | 2h | 1h 15min | ✔ |
| Controller-Methoden definieren | 1h 30min | - | ✘ |
| Arbeitsprotokoll | 30min | 30min | ✔ |
| Korrekturlesen | - | 45min | ✔ |

Tabelle 6: Konkrete Planung Tag 4

### Erledigte Arbeit

Heute konnte ich mit der Konfiguration beginnen und dabei auf viele mühselige Probleme stossen. Trotzdem konnte ich diese Arbeit abschliessen und mit der ersten Funktion beginnen: der Authentifizierung. Diese war auf Serverseite für sich alleine genommen schnell implementiert, da ich im gleichen Zug auch noch das Erstellen und Verwalten des Benutzers implementiert habe, war der Zeitaufwand trotzdem grösser als geplant. Als ich mit dem Frontend beginnen wollte, habe ich mich dazu entschieden, dass ich möglichst früh die Projektstruktur, das Routing und das globale Layout mit der Navigationsleiste erstellen möchte. Dies habe ich dann auch getan, wobei ich das Grundkonzept der Navigation aus einem anderen Projekt übernehmen konnte. Im gleichen Schritt habe ich die Atos-Farben als Theme-Farben definiert, damit diese nicht später vergessen gehen. Leider wurde ich durch das Logging und Frontend soweit aufgehalten, dass ich mit den weiteren Server-Schnittstellen nicht mehr beginnen konnte.

Da ich mit dem ersten Teil der Dokumentation fertig bin, habe ich am Abend und Wochenende eine Dreiviertelstunde investiert, um sprachliche Mängel zu beseitigen.

### Probleme

Das Hauptproblem war, dass ich den Aufwand für den Logger stark unterschätzt hatte. Das Konfigurieren an sich ging schnell von der Hand, jedoch musste ich noch Log-Meldungen einbauen, um diese Konfigurationen zu testen. Zusätzlich konnte ich vom Netzwerk am Arbeitsplatz nur begrenzt auf Azure zugreifen, weshalb ich die Konfiguration am Abend von zuhause aus abschliessen musste. Während der Konfiguration haben mir folgende Blog-Einträge weitergeholfen: <https://stackify.com/log4net-guide-dotnet-logging/> & <https://blogs.msdn.microsoft.com/benjaminperkins/2017/09/27/how-to-configure-log4net-on-azure-app-service/>. Die Log-Dateien scheinen nun ohne Probleme zu funktionieren und werden mir bei weiteren Fehlersuchen behilflich sein.

Als ich die zuvor erstellte Datenbank verwenden wollte, stiess ich auf einen Fehler bezüglich des kaskadierenden Löschens. Dieses Problem war mir bereits bewusst, da ich die Referenzen zum Versenden von Notifikationen ohnehin manuell löschen werde.

### Reflexion

Heute habe ich objektiv betrachtet viel geschafft, konnte meinen Zeitplan aber nicht eingehalten. Ich wurde durch viele zuerst nicht eingeplanten Aufgaben aufgehalten, was mich in der Planung um ungefähr zwei Stunden zurückwarf. Dies ist zwar noch kein grosser Rückfall, trotzdem könnten weitere unerwartete Probleme das Projekt gefährden. Ich werde fürs Erste keine Massnahmen ergreifen und vermute, dass ich mit den Schnittstellen schneller vorankomme als geplant.

## Tag 5 (13.03.2018)

### Planung

Heute möchte ich die Serverschnittstellen inklusive Pseudocode implementieren und Notifikationsmails versenden können.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Bestehenden Code-Dokumentation erweitern | 15min | 15min | ✔ |
| Controller und Methoden erstellen | 2h | 3h 30min | ✔ |
| Pseudocode einfügen | 45min | 2h | ✔ |
| Server-Wrapper im Frontend erstellen | 30min | 0min | ✘ |
| Schnittstellen der Notifikation fertigstellen | 1h | 1h | ✔ |
| Alle Notifikationen implementieren | 2h | 30min | ✘ |
| «Passwort vergessen» implementieren | 1h | 45min | ✔ |
| Arbeitsprotokoll | 30min | 0min | ✘ |

Tabelle 7: Konkrete Planung Tag 5

### Erledigte Arbeit

Heute habe ich nicht alles geschafft, was ich mir vorgenommen hatte. Konkret habe ich die Serverschnittstellen und Notifikationen nicht ganz abgeschlossen und bin somit nach dem Zeitplan im Verzug. Jedoch habe ich die Schnittstellen nicht nur definiert, sondern den zuvor erstellten Pseudocode ausgearbeitet. Somit ist das Backend zu einem Grossteil abgeschlossen, was bei den zukünftigen Arbeitsschritten Zeit sparen sollte. Dies entspricht zwar nicht meiner Planung, trotzdem sollte ich in der Lage sein, die Meilensteine rechtzeitig einzuhalten.

### Probleme

Bei der (voreiligen) Implementation der Funktionen stiess ich auf viele Sonderfälle, welche ich nicht bedacht hatte. So wusste ich, dass ich beim Löschen der Einträge auch die Relationen löschen muss. Jedoch habe ich nicht berücksichtigt, dass diese Relationen teilweise automatisch erstellt werden müssen. So zum Beispiel die Verbindung eines Benutzers zu einem Event (Eventteilnahme). Sobald ein Benutzer an einem Termin teilnimmt, sollte er automatisch am Event teilnehmen. Da der Organisator eine Termin-Teilnahme für den Benutzer setzen kann, könnte ein böswilliger Organisator Benutzer zu seinem Event hinzufügen, ohne deren Einwilligung. Solche Sonderfälle habe ich wo möglich berücksichtigt, was mich unnötig Zeit gekostet hat. Im Nachhinein hätte ich einen TODO-Kommentar hinterlassen und mich auf das wichtige konzentrieren sollen.

Mir ist aufgefallen, dass ich eventuell Probleme bezüglich Race Conditions bekommen könnte. Um dieses Problem zu umgehen verwende ich die Datenbank als synchronisierungspunkt. Da alle Speichervorgänge in einer Transaktion durchgeführt werden musstet ich zusätzlich sicherstellen, dass problematische Überprüfungen auch auf Datenbankebene durchgeführt werden. Ein Beispiel dafür ist die Teilnahme an einem Event oder Termin. Falls ein Benutzer schnell nacheinander an einem Event teilnimmt, könnten theoretisch mehrere Speicheroperationen zur Datenbank gesendet werden. Damit nur eine Teilnahme akzeptiert wird habe ich eindeutige Indexe konfiguriert.

### Reflexion

Im Gegensatz zu den vorherigen Tagen bin ich heute stark hinter dem Zeitplan. Zwar gehe ich davon aus, dass ich diesen Verzug morgen aufholen kann, trotzdem habe ich bedenken. Ein weiterer Punkt ist, dass ich das ERM leicht anpassen muss, da ich simple Werte wie den Titel eines Events vergessen habe. Auch habe ich bis jetzt viel Code geschrieben, ohne diesen durchzuführen. So könnten böse Überraschungen auf mich warten, sobald ich mit dem Frontend beginne. Ich werde mich die nächsten Tage auf jeden Fall auf die Kernfunktionalität fokussieren, damit diese fertig wird.

## Tag 6 (14.03.2018)

### Planung

Heute möchte ich meinen Verzug aufholen und mit dem Verwalten der Events grösstenteils fertig werden. Vorerst muss ich jedoch die bestehende Dokumentation abschliessen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Arbeitsprotokoll nachtragen | 30min | 15min | ✔ |
| ERM aktualisieren | 30min | 15min | ✔ |
| Notifikationen im Backend implementieren | 1h 30min | 2h | ✔ |
| Backend Schnittstellen abschliessen | 30min | 45min | ✔ |
| Server-Wrapper im Frontend erstellen | 30min | 1h | ✔ |
| Profil-Seite umsetzen | 1h | 1h | ✔ |
| Event-Erstellen umsetzen | 1h 30min | 1h 30min | ✔ |
| Event-Editieren umsetzen (Ohne Termine) | 1h 30min | 1h | ✘ |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 8: Konkrete Planung Tag 6

### Erledigte Arbeit

Am Morgen habe ich die Dokumentation aktualisiert und bin mit der Implementation der Notifikationen fortgefahren. Hier stiess ich auf Probleme mit den Daten und dem Cron-Service, weshalb ich länger gebraucht habe als erwartet. Da die Implementation der weiteren Funktionen, wie das Löschen eines Events, noch nicht implementiert sind, werde ich dazugehörigen Notifikationen fortlaufend testen. Am Nachmittag konnte ich mit der richtigen Frontend-Web-Entwicklung beginnen und habe die Profilverwaltung und einen Teil der Eventverwaltung abgeschlossen. Nebenbei musste ich allgemeine Änderungen am Design vornehmen und den «Sind Sie sicher?»-Dialog implementieren, welcher beispielsweise beim Löschen eines Events angezeigt wird. Beim Testen der Löschfunktion viel mir ausserdem auf, dass ich die Relationen in der falschen Reihenfolge gelöschte hatte, was ebenfalls korrigiert werden musste. Um einen letzten Fehler innerhalb der privaten Events zu korrigieren, habe ich fünfzehn Minuten Überzeit geleistet.

### Probleme

Bei der Planung habe ich mich dafür entschieden alle Daten als UTC zu formatieren. Im Frontend wird das Datum dann in der Browserzeit dargestellt. Ein nicht bedachtes Problem sind die Notifikations-Emails. Da der Server die Texte aufbereitet müssen diese in der Zeitzone des Empfängers formatiert werden. Um dieses Problem zu lösen habe ich eine konfigurierbare Zeitzone eingefügt, mit welcher die in UTC gespeicherten Daten umgewandelt werden. Diese Lösung setzt jedoch voraus, dass alle Empfänger der E-Mail-Adressen in der gleichen Zeitzone leben, was hier kein Problem ist. Analog habe ich das Format konfigurierbar gemacht, damit die Daten im de-CH Format gesendet werden.

Beim Testen meiner Cron-Implementation musste ich mit Schreck feststellen, dass die Datenbankoperationen nicht funktionierten. Konkret wurde der Context, in diesem Fall die Datenbankanbindung, nach der ersten Abfrage aufgelöst, weshalb weitere Abfragen fehlschlugen. Erst nach mühseligem Probieren fand ich heraus, dass der Fehler in «Parallel.ForEach» steckte. Die Idee hinter meiner Implementation war, dass die verschiedenen Cron-Dienste parallel laufen können. «Parallel.ForEach» erstellt dafür mehrere Threads, die sich mit dem Context nicht vertragen haben. Als Lösung werden die verschiedenen Cron-Handler nacheinander abgehandelt.

### Reflexion

Leider bin ich nicht so weit gekommen, wie ich es erhofft hatte, trotzdem liege ich jetzt wieder im Zeitplan. Ich versuche weiterhin die Hauptfunktionalität umzusetzen, bevor ich die bestehende Funktionalität vergolde. Da das Projekt nun zu grossen Teilen implementiert ist steigt auch die Code-Komplexität, was den Code teilweise unübersichtlich gestaltet. Falls ich mit der Funktionalität und Dokumentation früh genug fertig werde möchte ich auf jeden Fall an der Leserlichkeit des Codes feilen.

## Tag 7 (15.03.2018)

### Planung

Heute möchte ich die Eventverwaltung abschliessen, den zweiten Expertenbesuch durchführen und mit der Eventübersicht beginnen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Teilnehmerverwaltung | 1h | 1h | ✔ |
| Terminverwaltung | 1h 30min | 2h 15min | ✔ |
| Terminübersicht und Teilnahme | 2h | 1h 15min | ✔ |
| Expertenbesuch | 1h 30min | 1h 30min | ✔ |
| Events-Übersicht | 1h 30min | 1h 30min | ✔ |
| Arbeitsprotokoll | 30min | 30min | ✘ |
| Fix E-Mail-Spam | - | 15min | ✔ |

Tabelle 9: Konkrete Planung Tag 7

### Erledigte Arbeit

Die Arbeit ging gut voran und ich konnte mit der Implementierung der Funktionalität fertig werden. Trotzdem ist der Code noch nicht ganz abgeschlossen, da ich noch den Mobile-Design und Browsersupport sicherstellen möchte. Die Frontend-anforderungen konnte ich dank den Mockups schnell umsetzen und musste nur kleine Anpassungen vornehmen. Dank Angular konnte ich Komponenten nach ihren Funktionen trennen und den Code möglichst übersichtlich gestalten. Bei der Terminverwaltung stiess ich auf interessante Funktionen von Moment.js, der Datums-Bibliothek, welche ich verwende. So gibt ein Benutzer nur eine Reihe von Texten ein, welche durch RegEx überprüft werden.

Am Abend hat mich Rolf darüber informiert, dass er jede Stunde durch eine E-Mail auf seinem Eventteilnamen hingewiesen wurde, obwohl dies nur einmal geschehen soll. Eine Log-Analyse bestätigt, dass jede Stunde ein Reminder und Summary versendet werden. Beim suchen des Fehlers bin ich darauf gestossen, dass «SentEmail» nie auf true gesetzt wird, weshalb bei jeder Durchführung E-Mails versendet wurden. Ähnliche Fehler wie diesen möchte morgen durch exploratives Testen korrigieren, bevor ich nächste Woche den Testplan abarbeite.

### Probleme

Erneut stiess ich auf Probleme in Bezug auf das OR-Mapping. Die dabei generierte kryptische Fehlermeldung «Unable to create a constant value of type Event» half mir leider auch nicht weiter, weshalb ich mich an das Internet wandte. Glücklicherweise hat ein Benutzer unter <https://stackoverflow.com/a/18944508> erklärt, dass dieser Fehler beim Vermische von In-Memory- und Datenbank-Objekten auftritt. Die Lösung des Problems war, die Ids anstelle der Objekte miteinander zu vergleichen

### Expertenbesuch

Beim zweiten Expertenbesuchen haben wir zuerst den Ablauf der Präsentation besprochen, meine offenen Fragen beantwortet und anschliessend eine kleine Demo der Anwendung durchgeführt. Ein guter Punkt für die richtige Demo ist das bereithalten von Logs / Datenbanken und die Einbeziehung des Publikums. So konnte ich live Rolf und den Experten Herr Jansen zu einem zuvor erstellten Event einladen. Als wir über die Dokumentation gingen, war ich überrascht, als Herr Jansen nach der Definition eines Salts fragte. Da wir in der Schule Dutzende Male von Passwort-Hashing und Salts gehört hatten, habe ich betriebsblind angenommen, dass dies Allgemeinwissen ist. Um dies zu vermeiden möchte ich zum Schluss die exotischen Begriffe im Glossar erleutern.

### Reflexion

Da ich mit der Implementierung sogar vor meinem Zeitplan bin, möchte ich wie vom Experten geraten den Fokus auf die Dokumentation verschieben. So habe ich noch drei Tage Zeit, wobei Fehlerkorrekturen und Testen Potential zur Zeitverbrennung aufweisen. Alles in allem bin ich zufrieden mit dem bisherigen Projektablauf und hoffe beim Testen, auf keine bösen Überraschungen zu stossen.

## Tag 8 (16.03.2018)

### Planung

Heute möchte ich am Morgen die Implementation endgültig abschliessen und am Nachmittag die dazugehörige Dokumentation schreiben.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Arbeitsprotokoll nachtragen | 15min | 30min | ✔ |
| Code-Cleanup | 1h | 1h | ✔ |
| Exploratives Testen | 30min | 30min | ✔ |
| Bugfixes | 45min | 45min | ✔ |
| GUI-Vergoldung | - | 15min | ✔ |
| Dokumentation Backend | 3h | 2h 30min | ✔ |
| Dokumentation Frontend | 2h | 1h | ✔ |
| Arbeitsprotokoll | 30min | 30min | ✔ |
| Erweiterte Fehlerkorrekturen | - | 30min | ✔ |
| Allgemeine Dokumentationskorrektur | - | 1h | ✔ |

Tabelle 10: Konkrete Planung Tag 8

### Erledigte Arbeit

Am Morgen konnte ich das gestrige Arbeitsprotokoll abschliessen und durch das Einführen eines DeleteServices die Code-Qualität verbessern. Zuvor hatte ich den Code für das Löschen von Entitäten wild verteilt, wobei eine Implementation nach meiner Analyse sogar fehlerhaft war. Dank meines Umbaus ist der Code leserlicher und wartbarer. Beim explorativen Testen habe ich diverse Kleinigkeiten bemerkt, welche ich korrigiert habe. Durch das verwalten eines realen Events, dem Mittagessen mit einem anderen Lehrling, konnte ich die Online-Version ausgiebig testen. Dabei habe ich noch eine Vergoldung eingebaut, dass der Laden-Dialog bei schnell aufeinanderfolgenden Abfragen nicht mehrmals aufflackert. Am Nachmittag habe ich die Dokumentation der Realisierung ausgearbeitet und grundsätzlich abgeschlossen. Den Tag habe ich eine Stunde früher als geplant beendet, damit ich diese Stunde am Wochenende in das Korrekturlesen der Arbeit investieren kann. Am Wochenende habe ich neben der Dokumentation auch weitere Fehlerkorrekturen vorgenommen und Version 1.0.1 veröffentlicht. Darunter gab es einen kritischen Fehler, wodurch zeitabhängige E-Mails viel zu früh versendet wurden. Der Fehler lag in dem Aufruf der «DiffHours»-Methode, welche für die Daten 01.01.18 12:00 und 02.01.18 13:00 eine Differenz von einer Stunde zurückgab, wobei ich 25 Stunden erwartet hatte.

### Probleme

Beim Betrachten der Notifikationen ist mir aufgefallen, dass die «zufällig» von der Datenbank erstellten GUIDs nicht so zufällig sind, wie ich das erhofft hatte. In einigen Fällen variierten die verschiedenen GUIDs, in diesem Fall Session-Tokens, nur um eine Zahl. Nach Recherche im Internet habe ich unter anderem dank <https://blogs.msdn.microsoft.com/oldnewthing/20120523-00/?p=7553/> herausgefunden, dass diese Verhalten bis zu einem gewissen Grad beabsichtigt ist. Konkret sind G**U**IDs wie der Name schon sagt dafür ausgelegt **U**nique zu sein. Dabei wird nicht beachtet, ob diese zufällig sind, weshalb diese auch für kryptografische Funktionen nicht sinnvoll wären. In meinem konkreten Fall könnte ein Benutzer mit ein wenig Glück und Brute-Force die Session eines anderen Benutzers ausfindig machen. Da die bestehende, veraltete Lösung noch gar keine Benutzerauthentifizierung besitzt, ist die Gefahr für einen solchen Angriff sehr minimal.

### Reflexion

Heute konnte ich die Realisierungsphase grösstenteils abschliessen und langsam in die Testphase übergehen. Obwohl ich vor dem Zeitplan liege habe ich am Wochenende Zeit investiert, damit ich auf keinen Fall in Verzug komme. Das durchgeführte explorative Testen hat Fehler ans Licht gebracht, welche ich zum Glück allesamt schnell korrigieren konnte. Ein grosses Problem der Arbeit ist, dass ich niemals alles dokumentieren kann.

## Tag 9 (20.03.2018)

### Planung

Heute möchte ich die Testfälle durchführen, ein Testfazit ziehen und mit der Auswertung beginnen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Realisieren-Kapitel abschliessen | 1h | 45min | ✔ |
| Exploratives Testen – Dokumentation | 1h | 1h 15min | ✔ |
| Testfälle durchführen | 3h 30min | 2h 30min | ✔ |
| Testfazit ziehen | 1h | 1h | ✔ |
| Glossar | - | 30min | ✔ |
| Korrekturen am Word-Layout | - | 15min | ✔ |
| Kurzfassung | - | 1h | ✔ |
| Projektauswertung beginnen | 1h | 15min | ✘ |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 11: Konkrete Planung Tag 9

### Erledigte Arbeit

Heute habe ich zu Beginn die Realisierung abgeschlossen und das explorative Testen dokumentiert. Hier habe ich auch viele bereits getestete Funktionen dokumentiert, welche nur einen schnellen Nachtest benötigten. Bei der Durchführung der Testfälle habe ich ein paar Fehler der Applikation und der Testfälle gefunden, welche ich korrigiert habe. Dies waren grösstenteils Tippfehler oder eine falsche Weiterleitung. Trotzdem konnte ich die Testphase vorzeitig abschliessen und mich auf die bestehende Dokumentation fokussieren. Da ich vor dem Zeitplan war, habe ich die Zeit genommen die Verzeichnisse und das allgemeine Layout zu überprüfen und beispielsweise die Farbe des Zeitplans geändert, damit die Soll- und Ist-Zeit nach einem Ausdruck besser zu unterscheiden ist.

Beim Überprüfen der Aufgabenstellung ist mir aufgefallen, dass ich den Namen des Projekts konsequent falsch geschrieben habe. Ich habe jeweils «Hey I’m in» anstelle von «Hey**,** I’m in» geschrieben. Glücklicherweise musste ich dies nur an zwei Stellen korrigieren.

### Probleme

Beim Durchführen der Testfälle habe ich als Zweitgerät mein privates Smartphone verwendet. Als E-Mail-Adresse habe ich eine private Bluewin-Adresse angegeben, welche ich schon zuvor getestet hatte. Als ich zu den Testfällen der Notifikation kam, musste ich feststellen, dass ich keine Notifikationen erhielt. Als ich in den Log-Dateien gesehen habe, dass die E-Mails aus meiner Sicht versendet wurden, habe ich zu den Statistiken von SendGrid gewechselt. Dabei sieht man einen klaren Trend, dass alle E-Mails an meine Privatadresse den Status «Block» haben. Dank einem Informations-Tooltip wurde ich auf eine Seite verwiesen, die bekannte Spam-Mail-Server blockiert. Ich kenne leider den Grund für die Sperrung nicht, gehen jedoch davon aus, dass ein anderer Benutzer desselben SendGrid-Servers Spam-Wellen versendet hat. Auf der Website konnte ich eine Austragung der Spam-Liste «ix.dnsbl.manitu.net» beantragen, was ich auch getan hatte. Da die E-Mails trotzdem nicht verarbeitet wurden habe ich zu einer anderen Test-E-Mail-Adresse zurückgegriffen.

### Reflexion

Da ich beim Erstellen des Zeitplans die Zeit für das Kopieren des Quellcodes vergessen hatte, möchte ich mit der Auswertung möglichst rasch fertig werden. Leider habe ich mein Ziel bezüglich der Auswertung nicht erreicht, dafür den ersten Teil des IPA-Berichts abgeschlossen. Ich bin zuversichtlich, dass ich die Arbeit morgen Mittag abschliessen kann und einen Teil des Nachmittags mit Überarbeiten verbringen kann. Falls ich frühzeitig fertig bin werde ich die IPA auch abschliessen und somit einen Teil der geleisteten Überzeit ausgleichen.

## Tag 10 (21.03.2018)

### Planung

Heute möchte ich am Morgen die Reflexion abschliessen, die Arbeit auf grobe Fehler durchsuchen und den Anhang aufbereiten. Am Nachmittag möchte ich die Arbeit frühzeitig abgeben.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Kurzfassung abschliessen | 30min | 30min | ✔ |
| Reflexion verfassen | 1h | 1h 15min | ✔ |
| Schlusswort | 30min | 30min | ✔ |
| Anhang | 2h | 15min |  |
| Arbeitsprotokolle überfliegen | 30min |  |  |
| Teil 1 korrigieren | 30min |  |  |
| Wichtige Abschnitte Teil 2 korrigieren | 30min |  |  |
| Leserlichkeit nach Druck überprüfen | 30min |  |  |
| Arbeitsprotokoll | 30min |  | ✔ |

Tabelle 12: Konkrete Planung Tag 10

### Erledigte Arbeit

### Probleme

### Reflexion

# Kurzfassung

In diesem Kapitel folgt eine Zusammenfassung über das Projekt «Hey, I’m in» - Doodle für wiederkehrende Ereignisse

## Kurze Ausgangssituation

Christoph Fauti, der Auftraggeber dieses Projekts und Teamleiter bei der Atos, organisiert jede Woche Badminton. Für die Organisation hat er ein eigenes, nach heutigen Standards veraltetes, Webtool implementiert und Lösungen wie Doodle ausprobiert. Um den Zeitaufwand für das Organisieren solcher wiederkehrenden Events zu optimieren, wird die Webanwendung «Hey, I’m in» entwickelt. In einem ersten Schritt werden Kernfunktionen wie das Verwalten von Events und Terminen umgesetzt. Wichtig ist, dass sich jede Person selbst registrieren kann und über E-Mails auf dem Laufenden gehalten wird. In einem späteren Release werden Komfortfunktionen wie eine integrierte Kostenverwaltung ergänzt, um den Aufwand für den Organisator weiter zu optimieren.

## Umsetzung

Das Projekt ist nach der Projektmanagementmethode IPERKA geplant und realisiert. Der Fokus der Implementation liegt auf der Leserlichkeit des Codes, einer sinnvollen Architektur und einer robusten Fehlerbehandlung inklusive Logging. Für die Implementierung wurden aufgerundet fünf Tage benötigt.

Das Frontend ist mit Angular 5 und Angular Material realisiert. Angular Material bietet Komponenten zur Umsetzung des Material Designs im Web, wodurch die Webanwendung wie eine Android-Applikation aussieht. Alle Komponenten entsprechen dieser Designsprache und haben ein einheitliches Look-and-feel. Das Responsive Design ist mit Flexbox realisiert und passt sich der Bildschirmgrösse an. Die verschiedenen Kern-Komponenten sind auf drei Module aufgeteilt und verwenden weitere Hilfs-Komponenten, um die Erweiterbarkeit zu gewährleisten. Unter Verwendung des Angular-Frameworks wurden globale Funktionen wie ein Lade- oder Fehler-Dialog zentral umgesetzt, ohne Business-Logik anzupassen. Jegliche Logik ist in TypeScript verfasst und wird für den Browser zu JavaScript transpiliert. Analog sind Style-Regeln in Sass geschrieben und werden als CSS aufbereitet.

Das Backend ist in C# verfasst und verwendet das ASP.NET WebApi 2 Framework. Zur Übersicht und Wartbarkeit ist das Backend in fünf Projekte unterteilt, die verschiedene Verantwortlichkeiten übernehmen. Die Services sind durch dokumentierte Schnittstellen gekapselt und lösen ihre Abhängigkeiten über die Dependency Injection «Autofac» auf. Die Kommunikation vom Frontend zum Backend läuft über REST und HTTPS, wobei die Objekte als JSON übermittelt werden. Um ein Hosting ausserhalb der Schweiz zu ermöglichen, werden alle Zeitpunkte in UTC gespeichert. Die zur Datenhaltung verwendete MSSQL-Datenbank wird über das OR-Mapping «Entity Framework 6» aufgesetzt und angefragt.

## Ergebnis

Das Projekt ist vollumfänglich realisiert und auf Azure gehostet. Eine neue Instanz kann innert Minuten von GitHub installiert und in Azure konfiguriert werden. Christoph Fauti wird das Tool die nächsten Wochen testen und entscheiden, wie die Entwicklung weitergeht.

# 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 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, die 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 endgültig löschen. Bei einer solchen Löschung 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 oder organisert, wird dieser unter «Meine Events» aufgeführt. Um an einem Event teilzunehmen, kann ein Benutzer entweder allgemein am Event oder explizit an einem Termin teilnehmen. Ein Event kann verlassen werden, wobei Terminteilnahmen 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 zukünftigen Terminen zu- oder absagen. Optional kann jeder Teilnehmer Notifikationen deaktivieren, die 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. Der Organisator kann den Event als privat deklarieren, woraufhin nur eingeladene Personen teilnehmen dürfen. Falls ein Event nie mehr stattfinden wird, kann man diesen löschen. Über Änderungen an diesen Eigenschaften werden alle Teilnehmer informiert.

Neben dem Event an sich können auch Termine verwaltet werden. So kann ein Administrator neue Termine erfassen oder die bestehenden absagen. Events können nicht explizit verschoben werden, da alle Teilnehmer für die geänderte Startzeit ihre Teilnahme überdenken müssen. Beim Erfassen von neuen Terminen werden alle Startdaten mit Zeilenumbruch getrennt erfasst. Dies hat den Vorteil, dass man über Programme wie Excel sehr effizient Dutzende 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 können, darf jeder Organisator die Zeitperiode für Reminder und die Zusammenfassung selbst festlegen. Der Reminder ist personalisiert und wird an die bisherigen Teilnehmer des Termins gesendet. Die Zusammenfassung ist für alle identisch und beinhaltet eine Übersicht der (finalen) 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 den Benutzer automatisch anmeldet. Diese Zugangslinks sollten aus Sicherheitsgründen nach einer gewissen Zeitspanne 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, die in «Abbildung 8: Systemübersicht» mit durchgezogenen Verbindungen dargestellt ist. So kommuniziert die Angular-App (Frontend) mit dem Angular HttpClient, der 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, die 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

# C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\PlantUMLs\UseCases_Relationen.pngPlanen

## Use-Cases

Die Use-Cases sind aus drei Sichten betrachtet

* Benutzer
* Teilnehmer
* Organisator

Wobei ein Benutzer der Applikation immer ein «Benutzer» ist. Sobald er an einem Event teilnehmen möchte, wird er zum Teilnehmer, sobald er einen Event organisiert zum Organisator. Ein Benutzer kann auch gleichzeitig Organisator und Teilnehmer sein.

Abbildung 9: Use-Cases – Übersicht

### C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\PlantUMLs\UseCases_Benutzer.pngUse-Cases 1: Benutzer

Alle Use-Cases in diesem Kapitel werden von einem beliebigen Benutzer durchgeführt. Als Benutzerdaten sind die E-Mail-Adresse und das Passwort zu verstehen.

Abbildung 10: Use-Cases – Benutzer

|  |  |
| --- | --- |
| Use-Case 1.1: Anmelden | |
| Beschreibung | Als Benutzer möchte ich mich anmelden können |
| Ergebnis / Nachbedingung | Man wird angemeldet und bekommt Zugang zu den Events |
| Vorbedingung | Ein Benutzerkonto mit den Benutzerdaten muss registriert sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Login-Seite öffnen | | 2 | Benutzerdaten eingeben | | 3 | «Login» drücken | |

Tabelle 13: Use-Case 1.1 – Anmelden

|  |  |
| --- | --- |
| Use-Case 1.2: Registrieren | |
| Beschreibung | Als Benutzer möchte ich mich registrieren können |
| Ergebnis / Nachbedingung | Man wird angemeldet und bekommt Zugang zu den Events |
| Vorbedingung | Es existiert noch kein Benutzerkonto mit den Benutzerdaten |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Registrieren-Seite öffnen | | 2 | Benutzerdaten eingeben | | 3 | «Registrieren» drücken | |

Tabelle 14: Use-Case 1.2 – Registrieren

|  |  |
| --- | --- |
| Use-Case 1.3: Passwort zurücksetzen | |
| Beschreibung | Als Benutzer möchte ich mein Passwort zurücksetzen, falls ich diese vergessen habe |
| Ergebnis / Nachbedingung | Anmelden mit zurückgesetztem Passwort ist möglich |
| Vorbedingung | Ein Benutzerkonto mit der E-Mail-Adresse muss registriert sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Login-Seite öffnen | | 2 | Auf «Passwort vergessen» drücken | | 3 | Die eigene E-Mail-Adresse eingeben | | 4 | Code anfordern | | 5 | Per E-Mail erhaltenen Code eingeben | | 6 | Neues Passwort eingeben | | 7 | «Passwort setzen» drücken | |

Tabelle 15: Use-Case 1.3 – Passwort zurücksetzen

|  |  |
| --- | --- |
| Use-Case 1.4: Persönliche Daten aktualisieren | |
| Beschreibung | Als Benutzer möchte ich meine persönlichen Daten (Vollständiger Name, E-Mail-Adresse) ändern |
| Ergebnis / Nachbedingung | Das Benutzerkonto wurde aktualisiert |
| Vorbedingung | Man muss angemeldet sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Profil-Seite öffnen | | 2 | Gewünschte persönliche Daten ändern | | 3 | «Speichern» drücken | |

Tabelle 16: Use-Case 1.4 – Persönliche Daten aktualisieren^

|  |  |
| --- | --- |
| Use-Case 1.5: Passwort ändern | |
| Beschreibung | Als Benutzer möchte ich mein Passwort ändern |
| Ergebnis / Nachbedingung | Anmelden mit dem neuen Passwort ist möglich |
| Vorbedingung | Man muss angemeldet sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Profil-Seite öffnen | | 2 | Jetziges Passwort eingeben | | 3 | Gewünschtes neues Passwort eingeben | | 4 | «Speichern» drücken | |

Tabelle 17: Use-Case 1.5 – Anmelden

|  |  |
| --- | --- |
| Use-Case 1.6: Benutzerkonto löschen | |
| Beschreibung | Als Benutzer möchte ich mein Konto und alle damit verbundenen Daten löschen können |
| Ergebnis / Nachbedingung | Das Benutzerkonto existiert nicht mehr |
| Vorbedingung | Man muss angemeldet sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Profil-Seite öffnen | | 2 | «Profil löschen» drücken | | 3 | «Sind Sie sicher?»-Dialog bestätigen | |

Tabelle 18: Use-Case 1.6 – Benutzerkonto löschen

### Use-Cases 2: Teilnehmer

Alle Use-Cases in diesem Kapitel werden von einem angemeldeten Benutzer durchgeführt.

Abbildung 11: Use-Cases – Teilnehmer

|  |  |
| --- | --- |
| Use-Case 2.1: Event teilnehmen | |
| Beschreibung | Als (zukünftiger) Teilnehmer möchte ich ausdrücken, dass ich an einem Event interessiert bin und gerne teilnehmen würde |
| Ergebnis / Nachbedingung | Der Event taucht unter «Meine Events» auf |
| Vorbedingung | Es muss ein öffentlicher Event existieren, an welchem man noch nicht teilnimmt |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | Event unter «Öffentliche Events» aufklappen | | 3 | «Teilnehmen» drücken | |

Tabelle 19: Use-Case 2.1 – Event teilnehmen

|  |  |
| --- | --- |
| Use-Case 2.2: Event verlassen | |
| Beschreibung | Als Teilnehmer möchte ich einen Event verlassen |
| Ergebnis / Nachbedingung | Der Event taucht nicht mehr unter «Meine Events» auf |
| Vorbedingung | Man muss teil eines Events sein |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | Event, an welchem man teilnimmt, aufklappen | | 3 | «Verlassen» drücken | | 4 | «Sind Sie sicher?»-Dialog bestätigen | |

Tabelle 20: Use-Case 2.2 – Event verlassen

|  |  |
| --- | --- |
| Use-Case 2.3: Zusage eines Termins verwalten | |
| Beschreibung | Als Benutzer möchte ich ausdrücken, ob ich an einem Termin teilnehme oder nicht |
| Ergebnis / Nachbedingung | Dem Termin wurde zu- oder abgesagt |
| Vorbedingung | Es muss ein öffentlicher Event existieren |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | Event aufklappen | | 3 | «Details» drücken | | 4 | In der Termin-Übersicht bei einem Termin zu- oder absagen | |

Tabelle 21: Use-Case 2.3 – Zusage eines Termins verwalten

|  |  |
| --- | --- |
| Use-Case 2.4: Notifikationen zu Änderungen erhalten | |
| Beschreibung | Als Benutzer möchte ich informiert werden, wenn sich bei einem Event oder Termin etwas ändert |
| Ergebnis / Nachbedingung | Man erhält eine E-Mail, welche über die Änderungen informiert |
| Vorbedingung | Man muss einem Event teilnehmen, von welchem man selbst der Organisator ist, und man muss eine gültige E-Mail-Adresse besitzen |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | Eigenen Event aufklappen | | 3 | «Details» drücken | | 4 | «Editieren» drücken | | 5 | Den Titel des Events anpassen | | 6 | «Speichern» drücken | |

Tabelle 22: Use-Case 2.4 – Zusage eines Termins verwalten

|  |  |
| --- | --- |
| Use-Case 2.5: Notifikationen deaktivieren | |
| Beschreibung | Als Benutzer möchte ich gewisse Notifikationen deaktivieren können |
| Ergebnis / Nachbedingung | Man erhält zukünftig keine E-Mails zu den Themen, welche man deaktiviert hat |
| Vorbedingung | Man muss einem Event teilnehmen |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | Event, an welchem man teilnimmt, aufklappen | | 3 | «Details» drücken | | 4 | Häkchen bei zu deaktivierenden Notifikationen entfernen | |

Tabelle 23: Use-Case 2.5 – Notifikationen deaktivieren

### Use-Cases 3: Organisator

Abbildung 12: Use-Cases – Organisator

Alle Use-Cases in diesem Kapitel werden von einem angemeldeten Benutzer durchgeführt, welcher einen Event organisiert und auf der Editieren-Seite dieses Events ist.

|  |  |
| --- | --- |
| Use-Case 3.1: Neuen Event organisieren | |
| Beschreibung | Als (zukünftiger) Organisator möchte ich einen neuen Event organisieren |
| Ergebnis / Nachbedingung | Der Event existiert und kann verwaltet werden |
| Vorbedingung |  |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Events-Seite öffnen | | 2 | «Event erstellen» drücken | | 3 | Allgemeine Informationen eingeben | | 4 | «Speichern» drücken | |

Tabelle 24: Use-Case 3.1 – Neuen Event organisieren

|  |  |
| --- | --- |
| Use-Case 3.2: Informationen eines Events anpassen | |
| Beschreibung | Als Organisator möchte ich die Informationen meines Events anpassen |
| Ergebnis / Nachbedingung | Der Event verfügt über die geänderten Informationen |
| Vorbedingung |  |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Informationen ändern | | 2 | «Speichern» drücken | |

Tabelle 25: Use-Case 3.2 – Informationen eines Events anpassen

|  |  |
| --- | --- |
| Use-Case 3.3: Termine zu einem Event hinzufügen | |
| Beschreibung | Als Organisator möchte ich Termine zu meinem Event hinzufügen |
| Ergebnis / Nachbedingung | Der Termin wird im Event angezeigt |
| Vorbedingung |  |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | «Termine hinzufügen» drücken | | 2 | Mehrere Startdaten mit Zeilenumbruch getrennt eingeben | | 3 | Dialog bestätigen | |

Tabelle 26: Use-Case 3.3 – Termine zu einem Event hinzufügen

|  |  |
| --- | --- |
| Use-Case 3.4: Termin absagen | |
| Beschreibung | Als Organisator möchte ich bereits geplante Termin wieder absagen können |
| Ergebnis / Nachbedingung | Der Termin wird nicht mehr angezeigt |
| Vorbedingung | Der Event muss mindestens einen Termin haben |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Termin, welchen man absagen möchte, aufklappen | | 2 | «Absagen» drücken | | 3 | «Sind Sie sicher?»-Dialog bestätigen | |

Tabelle 27: Use-Case 3.4 – Termin absagen

|  |  |
| --- | --- |
| Use-Case 3.5: Teilnahme für verhinderten Teilnehmer absagen | |
| Beschreibung | Als Organisator möchte ich die Teilnahme für einen Teilnehmer absagen können, falls dieser sich persönlich abgemeldet hat, ohne dies online einzutragen |
| Ergebnis / Nachbedingung | Die Teilnahme der Person wurde abgesagt |
| Vorbedingung | Der Event muss mindestens einen Termin haben, bei welchem eine Person teilnimmt |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Termin, welchen man verändern möchte, aufklappen | | 2 | Gewünschte Person in der Liste finden | | 3 | Für die Person absagen | |

Tabelle 28: Use-Case 3.5 – Teilnahme für verhinderten Teilnehmer absagen

|  |  |
| --- | --- |
| Use-Case 3.6: Teilnehmer einladen | |
| Beschreibung | Als Organisator möchte ich explizit Teilnehmer einladen können, auch wenn sie noch kein Benutzerkonto besitzen |
| Ergebnis / Nachbedingung | Die Person hat eine Einladung per E-Mail erhalten |
| Vorbedingung | Die Person, welche man einlädt, nimmt nicht bereits am Event teil |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | «Teilnehmer einladen» drücken | | 2 | Die E-Mail-Adresse der einzuladenden Person eintragen | | 3 | Dialog bestätigen | |

Tabelle 29: Use-Case 3.6 – Teilnehmer einladen

|  |  |
| --- | --- |
| Use-Case 3.7: Inaktiven Benutzer entfernen | |
| Beschreibung | Als Organisator möchte ich inaktive Teilnehmer von einem Event entfernen können |
| Ergebnis / Nachbedingung | Die Person ist nicht mehr Teil des Events |
| Vorbedingung | Die Person, welche man entfernen möchte, muss am Event teilnehmen |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | In der Liste der Teilnehmer die gewünschte Person suchen | | 2 | «Entfernen» drücken | | 3 | «Sind Sie sicher?»-Dialog bestätigen | |

Tabelle 30: Use-Case 3.7 – Inaktiven Benutzer entfernen

|  |  |
| --- | --- |
| Use-Case 3.8: Event löschen | |
| Beschreibung | Als Organisator möchte ich Event löschen, welche nicht mehr durchgeführt werden |
| Ergebnis / Nachbedingung | Der Event existiert nicht mehr |
| Vorbedingung |  |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | «Löschen» drücken | | 2 | «Sind Sie sicher?»-Dialog bestätigen | |

Tabelle 31: Use-Case 3.8 – Event löschen

## Testkonzept

Das Testkonzept beschreibt, wie nach der Realisierung sichergestellt wird, dass alle gewünschten Funktionen vorhanden sind. Dazu werden Testfälle definiert und später in der Kontrollieren-Phase durchgeführt. Für jeden Use-Case existiert mindestens ein Testfall, wobei Testfallnummer als Präfix die Nummer des dazugehörigen Use-Cases besitzt.

### Eingesetzte Testmittel und -Methoden

Alle Tests werden, ausser explizit angegeben, mit folgender Hard- und Software durchgeführt:

* Dell-Laptop mit 16 Gigabyte Arbeitsspeicher, einem Intel i7 Prozessor und fungierender Internetverbindung
* Bildschirmauflösung von 1920x1080 ohne Zoom
* Google Chrome 65.0.3325.162
* Internet Explorer 11.0.9600.18952
* Windows 7 (Systemsprache: Deutsch)

### Funktionsübergreifende Tests

Alle Eingabemasken verfügen über eine Eingabevalidierung und Markierung der Pflichtfelder. Diese überall mit allen möglichen Werten zu testen, würde den Testrahmen sprengen. Aus diesem Grund wird angenommen, dass die Validierung überall funktioniert, falls während allen Testfällen und durch exploratives Testen keine Fehler festgestellt wurden.

### Testfälle 1: Benutzer

Diese Tests werden von einem Anwender durchgeführt, der grundsätzlich nicht angemeldet ist. Als Benutzerdaten werden, ausser explizit angegeben, eine beliebige funktionierende E-Mail-Adresse, der vollständige Name «Max Muster» und das Passwort «123» verwendet.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 1.1.1 | Ein Benutzer mit den Benutzerdaten ist registriert | |  |  | | --- | --- | | 1 | Login-Seite öffnen | | 2 | Benutzerdaten eingeben | | 3 | «Anmelden» drücken | | Man wird angemeldet |
| 1.1.2 | Ein Benutzer mit den Benutzerdaten ist registriert | |  |  | | --- | --- | | 1 | Login-Seite öffnen | | 2 | E-Mail-Adresse eingeben | | 3 | Das falsche Passwort *qwer* eingeben | | 4 | «Anmelden» drücken | | Man wird nicht angemeldet und wird darüber informiert, dass die Benutzerdaten ungültig waren |
| 1.2.1 | Es ist noch kein Benutzer mit den Benutzerdaten registriert | |  |  | | --- | --- | | 1 | Registrieren-Seite öffnen | | 2 | Benutzerdaten eingeben | | 3 | «Registrieren» drücken | | Man wird angemeldet |
| 1.2.2 | Ein Benutzer mit den Benutzerdaten ist registriert | |  |  | | --- | --- | | 1 | Registrieren-Seite öffnen | | 2 | Benutzerdaten eingeben | | 3 | «Registrieren» drücken | | Man wird nicht angemeldet und darüber informiert, dass die E-Mail-Adresse bereits registriert ist |
| 1.3.1 | Ein Benutzer mit den Benutzerdaten ist registriert | |  |  | | --- | --- | | 1 | Login-Seite öffnen | | 2 | «Passwort vergessen» drücken | | 3 | Die E-Mail-Adresse eingeben | | 4 | «Code anfordern» drücken | | 5 | Den per E-Mail erhaltenen Code eingeben | | 6 | Als neues Passwort *1234* eingeben | | 7 | «Passwort setzen» drücken | | Anmeldung mit dem Passwort *1234* ist möglich |
| 1.3.2 | Ein Benutzer mit den Benutzerdaten ist registriert | |  |  | | --- | --- | | 1 | Login-Seite öffnen | | 2 | «Passwort vergessen» drücken | | 3 | Die E-Mail-Adresse eingeben | | 4 | «Code anfordern» drücken | | 5 | Einen anderen Code eingaben, als man per E-Mail erhalten hat (Bsp. *e8c1423d-5478-4c19-926b-8d74d9eb736d*) | | 6 | Als neues Passwort «1234» eingeben | | 7 | «Passwort setzen» drücken | | Das Passwort wurde nicht geändert und der Benutzer wird darüber informiert, dass der Code ungültig ist |
| 1.4.1 | Man ist als Benutzer mit den Benutzerdaten registriert und angemeldet | |  |  | | --- | --- | | 1 | Profil-Seite öffnen | | 2 | Den vollständigen Namen auf «Max Muster der Grosse» ändern | | 3 | «Speichern» drücken | | Der vollständige Name wurde geändert und wird neu angezeigt |
| 1.5.1 | Man ist als Benutzer mit den Benutzerdaten registriert und angemeldet | |  |  | | --- | --- | | 1 | Profil-Seite öffnen | | 2 | «Profil löschen» drücken | | 3 | «Sind Sie sicher?»-Dialog bestätigen | | Man wird abgemeldet und kann sich nicht mehr anmelden, da der Benutzer nicht mehr existiert |

Tabelle 32: Testfälle 1 – Benutzer

### Testfälle 2: Teilnehmer

Diese Tests werden von einem beliebigen angemeldeten Benutzer durchgeführt, welcher eine gültige E-Mail-Adresse besitzt. Zu Testzwecken muss der öffentliche Event «Badminton» existieren (Siehe Kapitel «8.2.5 Testfälle 3: Organisator»), welcher von einem anderen Benutzer organisiert wird. In den Testfällen wird auf diesen «Badminton»-Event übersichtshalber mit «dem Event» verwiesen.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 2.1.1 | Der Benutzer nimmt nicht an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Öffentliche Events» aufklappen | | 3 | «Teilnehmen» drücken | | Der Event taucht unter «Meine Events» auf |
| 2.1.2 | Der Benutzer nimmt nicht an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Öffentliche Events» aufklappen | | 3 | «Details» drücken | | 4 | «Teilnehmen» drücken | | 5 | Events-Seite öffnen | | Der Event taucht unter «Meine Events» auf |
| 2.2.1 | Der Benutzer nimmt bereits an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Meine Events» aufklappen | | 3 | «Verlassen» drücken | | 4 | «Sind Sie sicher?»-Dialog bestätigen | | Der Event taucht unter «Öffentliche Events» auf |
| 2.2.2 | Der Benutzer nimmt bereits an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Meine Events» aufklappen | | 3 | «Details» drücken | | 4 | «Verlassen» drücken | | 5 | «Sind Sie sicher?»-Dialog bestätigen | | 6 | Events-Seite öffnen | | Der Event taucht unter «Öffentliche Events» auf |
| 2.3.1 | Der Benutzer nimmt nicht an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Öffentliche Events» finden | | 3 | Neustem Termin durch drücken des Häkchens zusagen | | Der Event taucht unter «Meine Events» auf und bleibt zugesagt |
| 2.3.2 | Der Benutzer nimmt nicht an dem Event teil | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Öffentliche Events» finden | | 3 | «Details» drücken | | 4 | Beiden Terminen durch Drücken der Häkchen zusagen | | 5 | Events-Seite öffnen | | Der Event taucht unter «Meine Events» auf und bleibt zugesagt |
| 2.3.3 | Der Benutzer nimmt am Event teil und hat den Terminen zugesagt | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Meine Events» finden | | 3 | Neustem Termin durch drücken des Kreuzchens absagen | | Der Event bleibt unter «Meine Events» und ist abgesagt |
| 2.4.1 | Der Benutzer nimmt am Event teil und hat den Terminen zugesagt | |  |  | | --- | --- | | 1 | «Badminton»-Event editieren (Siehe Testfall 3.2.1) | | Der Benutzer wird per E-Mail über die Änderung informiert |
| 2.4.2 | Der Benutzer nimmt am Event teil | |  |  | | --- | --- | | 1 | «Badminton»-Event löschen (Siehe Testfall 3.8.1) | | Der Benutzer wird per E-Mail über die Löschung informiert |
| 2.4.3 | Der Benutzer nimmt am Event teil und hat den Terminen zugesagt | |  |  | | --- | --- | | 1 | «Badminton»-Event Termin absagen (Siehe Testfall 3.4.1) | | Der Benutzer wird per E-Mail über die Absagung informiert |
| 2.5.1 | Der Benutzer nimmt am Event teil und hat den Terminen zugesagt | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | Event unter «Meine Events» finden | | 3 | «Details» drücken | | 4 | Notifikationen abwählen | | 5 | «Speichern» drücken | | Bis zum Eventstart werden keine Informations-E-Mails erhalten |

Tabelle 33: Testfälle 2 – Teilnehmer

### Testfälle 3: Organisator

Diese Tests werden von einem beliebigen angemeldeten Benutzer durchgeführt, welcher den Event «Badminton» organisiert. Um diesen Test-Event zu erstellen, müssen die Testfälle 3.1.1 und 3.3.1 durchgeführt werden. Wird der Event als Teil des Testfalles editiert, sollte man diesen, wie im Testfall 3.8.1 beschrieben, löschen und erneut erstellen. In den Testfällen wird auf diesen «Badminton»-Event übersichtshalber mit «dem Event» verwiesen.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 3.1.1 | «Badminton»-Event existiert noch nicht | |  |  | | --- | --- | | 1 | Events-Seite öffnen | | 2 | «Event erstellen» drücken | | 3 | Folgende Daten eingeben:  Titel: *Badminton*  Treffpunkt: *4. Stock*  Beschreibung: *Badminton*  Zugriff: *Öffentlich*  Reminder: 1*h*  Zusammenfassung: *1h* | | 4 | «Erstellen» drücken | | Der Event wurde erstellt und das Event-Editieren wurde geöffnet |
| 3.2.1 |  | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | Treffpunkt zu *5. Stock* ändern | | 3 | «Speichern» drücken | | Der Treffpunkt wurde geändert |
| 3.3.1 |  | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | «Termine hinzufügen» drücken | | 3 | Die Startpunkte von zwei Terminen eingeben, welche in zwei beziehungsweise vier Stunden beginnen (Bsp. *07.03.2018 12:00*  *07.03.2018 14:00*) | | 4 | Dialog bestätigen | | Die zwei Termine werden in den Event-Details angezeigt |
| 3.4.1 |  | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | Ersten Termin aufklappen | | 3 | «Absagen» drücken | | 4 | «Sind Sie sicher?»-Dialog bestätigen | | Der Termin wird nicht mehr angezeigt |
| 3.5.1 | Eine Person muss am Event teilnehmen und den Terminen zugesagt haben (Testfall 2.3.2) | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | Ersten Termin aufklappen | | 3 | Person in der Liste finden | | 4 | Für die Person absagen | | Für die Person wurde abgesagt |
| 3.6.1 | Es existiert kein Benutzerkonto mit der Testmail | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | «Teilnehmer einladen» drücken | | 3 | Eine Testmail eingeben, welche funktioniert | | 4 | Dialog bestätigen | | Auf der Testmail wird eine Einladung empfangen |
| 3.6.2 | Es existiert ein Benutzer mit der Testmail, welcher nicht am Event teilnimmt | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | «Teilnehmer einladen» drücken | | 3 | Eine Testmail eingeben, welche funktioniert | | 4 | Dialog bestätigen | | Auf der Testmail wird eine Einladung empfangen, welche personalisiert an die Person gerichtet ist |
| 3.6.3 | Es existiert ein Benutzer mit der Testmail, welcher am Event teilnimmt | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | «Teilnehmer einladen» drücken | | 3 | Die Testmail eingeben | | 4 | Dialog bestätigen | | Es wird keine Einladung versendet und der Benutzer wird darüber informiert, dass die Person bereits Mitglied des Events ist |
| 3.7.1 | Eine Person muss am Event teilnehmen (Testfall 2.1.1) | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | Person in der Liste finden | | 3 | «Entfernen» drücken | | 4 | «Sind Sie sicher?»-Dialog bestätigen | | Die Person ist nicht mehr Teil des Events |
| 3.8.1 |  | |  |  | | --- | --- | | 1 | Event-Editieren öffnen | | 2 | «Löschen» drücken | | 3 | «Sind Sie sicher?»-Dialog bestätigen | | Der Event existiert nicht mehr |

Tabelle 34: Testfälle 3 – Organisator

## Software-Architektur

In diesem Kapitel wird die allgemeine Architektur festgelegt. Es werden alle essenziellen Klassen und Schnittstellen definiert, ohne eine Implementation vorzugeben. Dadurch können während der Realisierung immer noch Verbesserungen integriert werden, ohne die grundlegende Architektur ändern zu müssen.

### Backend

|  |  |  |
| --- | --- | --- |
| Modul | Inhalt | Externe Abhängigkeiten |
| WebApplication | REST-Schnittstellen  Business-Logik für die Eventverwaltung | ASP.NET WebApi 2  Autofac |
| Authentication | Services rund um die Authentifizierung | Bcrypt |
| MailNotifier | Services zum Senden von Notifikationen | SendGrid |
| Database | Datenbankanbindung und Model | Entity Framework |
| External | Codeschnipsel aus anderen Projekten |  |

Tabelle 35: Backend Modulübersicht

Das Backend ist ein grosses Projekt (Visual Studio Solution) mit fünf Modulen (Visual Studio Project). Das Kernmodul ist die WebApplication, welches die Schnittstellen zur Aussenwelt und Business-Logik realisiert. Zudem ist dieses Modul der Einstiegspunkt der Applikation, weshalb globale Konzepte wie die DI (**D**ependency **I**njection) hier eingerichtet werden.

Abbildung 13: Backend Komponentendiagramm

Konkret muss die Klasse WebApiApplication, der Einstiegspunkt beim Start der Webanwendung, die REST-Schnittstelle konfigurieren, sensitive Konfigurationsdaten bereitstellen, das DI-Framework mit allen Klassen und Interfaces initialisieren und den Logger mit der korrekten Konfiguration versorgen.

Für Autofac, das verwendete DI-Framework, wird zur Konfiguration ein AssemblyScanner verwendet, welcher aus einem anderen Projekt übernommen wird. Als Logger wird die log4net-Bibliothek eingesetzt, welche über eine Konfigurationsdatei eingerichtet wird. Diese Konfigurationsdatei kann ohne Neustart angepasst werden, wobei Änderungen direkt in Kraft treten.

### Frontend

Das Frontend verwendet diverse Angular-Konzepte um redundanten Code zu vermeiden. So werden Interceptors eingesetzt, damit global auf allen Abfragen Fehler abgehandelt werden. Auch werden GUI-Elemente, welche mehrmals benötigt werden, in eine eigene Komponente extrahiert. Dadurch können Änderungen an oft vorkommenden Komponenten, wie beispielsweise die An- und Ablehnen-Knöpfe, schnell integriert werden. Da das Frontend zum Grossteil die Darstellung übernimmt, gibt es keinen Bedarf für komplexe Abhängigkeiten.

## 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 (Code-First). Hierbei werden auch Migrationen beim Umbau des Models erstellt und das korrekte Mapping von Klassen zu Datenbanktypen übernommen.

### ERD

Abbildung 14: ERD

### Entitätsbeschreibungen

|  |  |
| --- | --- |
| Session | |
| Beschreibung | Eine Session |
| Verbindungen | Hat einen Benutzer |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Token | Guid | Die eindeutige Identifikation | Pflichtfeld  Generiert | | UserId | Int32 | Fremdschlüssel Benutzer | Pflichtfeld | | ValidUntil | DateTime | Bis wann die Session gültig ist |  | | Created | DateTime | Wann diese Session erstellt wurde | Pflichtfeld | |

Tabelle 36: Entitätsbeschreibung Session

|  |  |
| --- | --- |
| User | |
| Beschreibung | Ein registrierter Benutzer |
| Verbindungen | Kann von einem oder mehreren Events der Organisator sein  Kann an einem oder mehreren Events teilnehmen  Kann an einem oder mehreren Terminen teilnehmen  Kann eine oder mehrere Sessions haben  Kann eine oder mehrere Passwort-Zurücksetzungen haben |
| Primärschlüssel | Id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Id | Int32 | Die eindeutige Identifikation | Pflichtfeld  Generiert | | FullName | String | Der vollständige Name | Pflichtfeld | | Email | String | Die einmalige E-Mail-Adresse | Pflichtfeld | | PasswordHash | String | Ein sicherer Hash des Passworts | Pflichtfeld | |

Tabelle 37: Entitätsbeschreibung User

|  |  |
| --- | --- |
| PasswordReset | |
| Beschreibung | Eine Anfrage auf die Zurücksetzung des Passworts |
| Verbindungen | Hat einen Benutzer |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Token | Guid | Die eindeutige Identifikation | Pflichtfeld  Generiert | | UserId | Int32 | Fremdschlüssel Benutzer | Pflichtfeld | | Requested | DateTime | Wann die Anfrage gestellt wurde | Pflichtfeld | | Used | Boolean | Ob diese Anfrage bereits verwendet wurde |  | |

Tabelle 38: Entitätsbeschreibung PasswordReset

|  |  |
| --- | --- |
| EventParticipation | |
| Beschreibung | Die Teilnahme eines Benutzers an einem Event |
| Verbindungen | Hat einen Benutzer  Hat einen Event |
| Primärschlüssel | Id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Id | Int32 | Die eindeutige Identifikation | Pflichtfeld  Generiert | | ParticipantId | Int32 | Fremdschlüssel Benutzer | Pflichtfeld | | EventId | Int32 | Fremdschlüssel Event | Pflichtfeld | | SendReminderEmail | Boolean | Ob der Teilnehmer Reminder-E-Mails erhalten möchte |  | | SendSummaryEmail | Boolean | Ob der Teilnehmer Zusammenfassungs-E-Mails erhalten möchte |  | | SendLastMinute-ChangesEmail | Boolean | Ob der Teilnehmer E-Mails zu kurzfristigen Änderungen erhalten möchte |  | |

Tabelle 39: Entitätsbeschreibung EventParticipation

|  |  |
| --- | --- |
| Event | |
| Beschreibung | Ein Event |
| Verbindungen | Hat einen Organisator (Benutzer)  Kann eine oder mehrere Teilnahmen haben  Kann eine oder mehrere Einladungen haben  Kann eine oder mehrere Termine haben |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Id | Int32 | Die eindeutige Identifikation | Pflichtfeld  Generiert | | OrganizerId | Int32 | Fremdschlüssel Benutzer | Pflichtfeld | | Title | String | Titel | Pflichtfeld | | MeetingPlace | String | Der Treffpunkt | Pflichtfeld | | Description | String | Eine kurze Beschreibung | Pflichtfeld | | IsPrivate | Boolean | Private Events benötigen eine Einladung | Pflichtfeld | | ReminderTime-WindowInHours | Int32 | Wie viele Stunden vor Terminbeginn der Reminder versendet wird | Pflichtfeld | | SummaryTime-WindowInHours | Int32 | Wie viele Stunden vor Terminbeginn die Zusammenfassung versendet wird | Pflichtfeld | |

Tabelle 40: Entitätsbeschreibung Event

|  |  |
| --- | --- |
| EventInvitation | |
| Beschreibung | Eine Einladung für einen Event |
| Verbindungen | Hat einen Event |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Token | Guid | Die eindeutige Identifikation | Pflichtfeld  Generiert | | EventId | Int32 | Fremdschlüssel Event | Pflichtfeld | | Requested | DateTime | Wann die Einladung versendet wurde | Pflichtfeld | | Used | Boolean | Ob diese Einladung bereits verwendet wurde |  | |

Tabelle 41: Entitätsbeschreibung EventInvitation

|  |  |
| --- | --- |
| Appointment | |
| Beschreibung | Ein Termin eines Events |
| Verbindungen | Hat einen Event  Kann eine oder mehrere Teilnahmen haben |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Id | Int32 | Die eindeutige Identifikation | Pflichtfeld  Generiert | | EventId | Int32 | Fremdschlüssel Event | Pflichtfeld | | StartTime | DateTime | Wann die der Termin stattfindet | Pflichtfeld | |

Tabelle 42: Entitätsbeschreibung Appointment

|  |  |
| --- | --- |
| AppointmentParticipation | |
| Beschreibung | Die Teilnahme eines Benutzers an einem Termin, sowohl für Zusagen wie Absagen |
| Verbindungen | Hat einen Event  Hat einen Benutzer |
| Primärschlüssel | Token |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | Id | Int32 | Die eindeutige Identifikation | Pflichtfeld  Generiert | | AppointmentId | Int32 | Fremdschlüssel Termin | Pflichtfeld | | ParticipantId | Int32 | Fremdschlüssel Benutzer | Pflichtfeld | | Appointment-ParticipationAnswer | Enum(  Accepted,  Declined) | Die explizite Antwort dieser Teilnahme | Pflichtfeld | | SentReminder | Boolean | Ob der Reminder versendet wurde | Pflichtfeld | | SentSummary | Boolean | Ob eine Zusammenfassung versendet wurde | Pflichtfeld | |

Tabelle 43: Entitätsbeschreibung AppointmentParticipation

### Anmerkung

Da ich nie direkt mit der Datenbank in Kontakt trete, habe ich mich dazu entschieden, die Typen der Klassen zu übernehmen, anstelle der Datenbanktypen. So werden gewisse Typen in der Datenbank anders abgebildet, so beispielsweise der Boolean (true, false) als bit (1, 0). Dies wird glücklicherweise vom Entity Framework intern umgewandelt und ist aus meiner Sicht irrelevant. Die Maximallängen der String-Werte wurden noch nicht durchgehend definiert, da diese von der Darstellung in der Benutzeroberfläche abhängig sind. So könnte die Datenbank Hunderte von Zeichen speichern, jedoch müssen diese auch auf dem Bildschirm Platz finden. Die Definition sinnvoller Maximalwerte wird während der Realisierung fortlaufend nachgetragen.

## Logging-Konzept

Ereignisse werden zur Fehleranalyse und Nachvollziehbarkeit vom Backend in ein Log geschrieben.

### Log-Dateien

Die Log-Dateien werden während der Entwicklung im Root-Verzeichnis der Webanwendung unter dem Ordner «App\_data» gespeichert. Auf Azure werden die Log-Dateien unter D:\home\LogFiles gespeichert. Das Speichern und Verwalten der Log-Dateien übernimmt das Log-Framework log4net. Nach der Konfiguration werden folgende Dateien erstellt & rotiert:

|  |  |  |
| --- | --- | --- |
| Log | Lebensdauer | Inhalt |
| All.log | Fünf Dateien à 20MB – Ältere Dateien werden gelöscht | Alle Einträge |
| Audit.log | Eine Datei pro Monat – Dateien werden nach sechs Monaten gelöscht | Benutzer erstellen / löschen  Benutzer loggt sich ein / loggt sich aus  Änderungen an Terminen und Zusagen |

Tabelle 44: Logdateien

### Log-Level

Jede Logmitteilung hat ein zugewiesenes Log-Level (Gewichtung / Severity). Für jede Mitteilung wird passend zur Situation ein passendes Log-Level gewählt.

|  |  |  |
| --- | --- | --- |
| Log-Level | Beschreibung | Beispiele |
| Debug | Entwickler-Meldungen, welche deaktiviert werden könnten | HTTP-Anfrage wurde bearbeitet |
| Info | Informationen, welche im produktiven Betrieb relevant sein können | Audit-Meldungen  Server wurde gestartet / gestoppt |
| Warn | Unerwartetes Verhalten, nach welchem der Programmfluss nicht gestört wird | Konfigurationswert invalid, Standardwert wird verwendet |
| Error | Fehler, welche beim Entwickeln berücksichtigt und behandelt wurden, aber trotzdem nicht auftreten sollten | Try-Catch-Blöcke  Senden einer E-Mail fehlgeschlagen |
| Fatal | Kritische Fehler, welche den Programmfluss unterbrochen haben | Unbehandelte Ausnahmen |

Tabelle 45: Log-Level

### Nachrichtenformat

Die Nachrichten innerhalb der Log-Meldungen sind analog zum Programmcode englisch gehalten. Die Log-Zeiten sind in UTC gehalten, da die Azure-Maschine in einer anderen Zeitzone liegt. Werte wie der Benutzer / Session werden für die Anfrage konfiguriert und müssen nicht bei allen Log-Meldungen explizit angegeben werden. Die Log-Meldungen sind wie folgt formatiert, um maschinelle Auswertungen zu ermöglichen:

ZEIT [THREAD] LEVEL NAMESPACE.KLASSE [BENUTZER | SESSION] METHODE(): NACHRICHT

Beispiel eines solchen Logs:

09:15 [1] DEBUG HeyImIn.WebServer.Class1 [3 | 4A0AF93A] StuffAsync(): Started operation

09:16 [2] INFO HeyImIn.WebServer.Class2 [5 | B56CAD34] Logout (): User was logged out

09:18 [1] DEBUG HeyImIn.WebServer.Class1 [3 | 4A0AF93A] StuffAsync(): Finished successfully

# Entscheiden

In diesem Kapitel werden die getroffenen Entscheidungen über die verwendeten Frameworks und Umsetzungsdetails begründet.

## Authentifizierung

Die Benutzerauthentifizierung ist ein wichtiger Aspekt der Anwendung. So wird bei jeder Abfrage überprüft, welcher Benutzer diese ausführt. Die Informationen zur Benutzeridentifikation werden vom Browser bei jeder Anfrage beigelegt. Eine konkrete Anforderung an die Authentifizierung ist, dass Zugangslinks über die E-Mail-Adresse möglich sind, welche nach einer gewissen Zeitspanne ungültig werden.

### Bestehende Protokolle

Zum einen könnte man grosse Protokolle wie OAuth2 verwenden, welche standardisierte und sichere Schnittstellen anbieten. Da ich keine Erfahrungen mit OAuth oder ähnlichen Protokollen besitze, habe ich diese Umsetzung ausgeschlossen.

### Übermittlung des Passworts

Die naivste Implementation wäre das Übermitteln von E-Mail-Adresse und Password bei jeder Abfrage. Dieser Ansatz würde das Erstellen der Zugangslink erschweren, da es weder sicher noch möglich ist, das Klartext-Passwort per E-Mail zu versenden. Somit müsste man für diesen Ansatz neben der Passwort-Authentifizierung auch die Authentifizierung über einen Zugangslink implementieren.

### Eigene Sessions

Ein Ansatz ist das Erstellen und Übermitteln von Session-Tokens. Diese Tokens müssen eindeutig und mit einem Benutzer verknüpft sein. Ich habe mich für diese Variante entschieden, damit ich die volle Kontrolle über die Verwaltung und Gültigkeit der Sessions besitze. Als Session-Token setze ich eine GUID ein, welche beim Versenden des Zugangslinks beigelegt wird. In der Datenbank werden diese Tokens zusammen mit einem Ablaufdatum und dem Benutzer gespeichert. Da dieses Token in keinerlei Relation zum Passwort des Benutzers steht, wird es zur Bequemlichkeit im Local Storage des Browsers gespeichert. Hierdurch muss sich der Benutzer nur selten erneut anmelden.

## Umgang mit Zeitzonen

Im Umgang mit Daten stellt man sich immer die Frage, wie man mit der Zeitverschiebung umgeht. Eigentlich wäre dieses Problem nicht existent, da alle Anwender dieser Applikation aus der Schweiz kommen und somit in der gleichen Zeitzone leben. Problematisch ist jedoch, dass die Zeitzone des Azure-Servers variieren kann. Eine Lösung hierfür wäre das explizite Konfigurieren der Zeitzone, was ich jedoch nicht ideal finde. Zum einen könnte das Konfigurieren vergessen gehen, zum anderen könnten Probleme bei der Sommer- und Winterzeit auftreten. So wäre es sehr dubios, wenn Log-Dateien überlappende Zeitstempel hätten. Aus diesen Gründen habe ich mich entschieden, alle Uhrzeiten in UTC abzuspeichern. Dies gilt für alle Zeitstempel in der Datenbank und den Log-Dateien. Zur Folge sind alle lokalen Log-Ereignisse eine Stunde (beziehungsweise zwei in der Sommerzeit) zurückversetzt. Dies macht das Auswerten per Hand etwas umständlicher, sollte jedoch das maschinelle Einlesen und verarbeiten leichter gestalten.

## Umgang mit sensitiven Konfigurationswerten

Eine verpönte Praxis in der Informatik ist das Einprogrammieren von Passwörtern im Quellcode. Für dieses Projekt werden zwei Konfigurationswerte benötigt, welche als sensitiv zu betrachten sind und ebenso wie Passwörter nicht direkt im Quellcode stehen dürfen. Diese zwei Konfigurationswerte sind die Verbindungszeichenfolge der MSSQL-Datenbank und der API-Schlüssel der SendGrid-Plattform. Ich habe mich dazu entschieden, diese Werte über die Azure-Plattform zu verwalten. Konkret wird die Verbindungszeichenfolge mit dem Schlüssel «HeyImIn» und die Anwendungseinstellung mit dem Schlüssel «SENDGRID\_API\_KEY» gesetzt. Diese werden von der Anwendung beim Start ausgelesen und fortan verwendet. Die lokale Testdatenbank wird im Projekt konfiguriert, während jeder Entwickler seine lokalen Umgebungsvariablen um den SENDGRID\_API\_KEY ergänzen muss.

## Cron-Jobs

Ein Webserver bekommt normalerweise eine Anfrage, bearbeitet diese und liefert eine Antwort. Eine Ausnahme zu diesem Muster stellen in diesem Projekt die Reminder- und Zusammenfassungs-E-Mails dar. Diese werden eine gewisse Zeitspanne vor Eventbeginn versendet, auch wenn kein Benutzer die Website geöffnet hat. Um dieses Problem zu lösen habe ich mich entschieden einen Cron-Job einzurichten. Dieser Cron-Job sollte in regelmässigen Abständen eine Schnittstelle aufrufen, woraufhin der Webserver interne Aufträge abarbeitet. Konkret geht es momentan einzig um das Versenden von Notifikationen, zukünftig könnte man aber auch beispielsweise die Datenbank bereinigen. Zur Umsetzung der wiederholenden Webanfragen wird der Azure Scheduler verwendet. Dieser Azure-Dienst erlaubt gratis das stündliche Versenden einer Webanfrage.

## Verwendete Tools / Frameworks

Die meisten verwendeten Technologien / Frameworks sind von der Aufgabenstellung vorgegeben. Die weiteren Frameworks sind in diesem Kapitel aufgelistet.

### Logger

Aufgrund meiner Erfahrungen setze ich log4net ein. Dieser Logger wird auch in dem MediaGateway-Projekt eingesetzt, in welchem ich seit eineinhalb Jahren mitentwickle. Log4net bietet alle Funktionen an, welche vom Logging-Konzept verlangt werden und ist daher die logische Wahl.

### Hashfunktion

Die Benutzerpasswörter sollen nie in Klartext, sondern sicher als Hash gespeichert werden. Eine kryptografisch sichere und einfach zu bedienende Hashfunktion ist bcrypt, welche sogar automatisch Salts generiert. Zudem habe ich bcrypt bereits in mehreren Schulprojekten verwendet. Da es mehrere C#-Implementationen gibt, musste ich eine auswählen. Ich habe mich für die aktuellste Umsetzung «BCrypt.Net-Next» entschieden. Diese Implementation ermöglicht neben dem Hashen und Überprüfen eines Passworts auch die Angabe eines Arbeitsfaktors. Dieser Wert bestimmt die Rechenleistung und den dementsprechenden Zeitaufwand, welcher für das Hashen benötigt wird. Ein grösserer Wert sorgt dafür, dass die gespeicherten Passwörter auch mit dem Anstieg der Rechenleistung gegen Rainbow-Table-Angriffe sicher bleiben.

# Realisieren

In diesem Kapitel wird auf die Details der Implementierung eingegangen.

## Deployment (Azure)

### Benötigte Ressourcen

1. Web-App
2. SQL-Datenbank
3. SendGrid Account
4. Scheduler

### Web-App Deployment & Updates

Innerhalb der Web-App-Einstellungen kann unter dem Reiter «Bereitstellungsoptionen» GitHub selektiert werden. Danach muss über einen GitHub-Account auf das GitHub-Repository <https://github.com/PascalHonegger/HeyImIn> und die Verzweigung (Branch) «azure» verwiesen werden. Nach dieser Konfiguration werden Updates automatisch eingespielt. Eine zusätzliche empfehlenswerte Konfiguration ist «Nur HTTPS», was unter dem Reiter «Benutzerdefinierte Domänen» aktiviert werden kann, wodurch die Seite zwingend über HTTPS verschlüsselt wird.

### Datenbank

Als Datenbank wird eine SQL-Datenbank in Azure erstellt. Innerhalb der Datenbankeinstellungen kann man die Verbindungszeichenfolge auslesen, welche man bei der Web-App unter «Anwendungseinstellungen» -> «Verbindungszeichenfolgen» -> «HeyImIn» eintragen muss.

### SendGrid

Mithilfe von <https://sendgrid.com/docs/User_Guide/Settings/api_keys.html> kann ein API-Schlüssel erstellt werden. Dieser Schlüssel muss unter der Anwendungseinstellung «SENDGRID\_API\_KEY» eingetragen werden. Es ist wichtig, dass Click Tracking deaktivier ist, da die URLs ansonsten nicht funktionieren! Dies kann innerhalb der SendGrid-Website unter Settings -> Tracking -> Click Tracking deaktiviert werden.

### Scheduler (Cron-Job)

Um die Cron-Jobs regelmässig durchzuführen, muss der Scheduler richtig konfiguriert sein. Konkret muss eine POST-Anfrage über HTTPS auf die URL «/api/Cron/Run» ausgeführt werden. Die Zeitabstände sind für den Programmcode irrelevant, in diesem Fall wurde jede Stunde gewählt.

### KUDU

Da dieses Projekt die neueste Version von C# verwendet, werden die aktuellsten MSBuild-Tools benötigt. Zum Zeitpunkt des Deployments sind diese noch nicht standardmässig aktiviert und wurden manuell konfiguriert. Diese Konfiguration von KUDU, dem automatischen Deployment-Tool von Azure, ist in Git in der Datei «.deployment» aufzufinden. Falls das automatische Bereitstellen in Zukunft Probleme bereiten sollte, wäre die Ursache höchstwahrscheinlich diese Konfiguration.

## Übergreifend

Eine Beschreibung der wichtigsten Themen, welche das Front- und Backend betreffen.

### Routing

Das grundsätzliche Routing ist zuerst im Backend umgesetzt, wobei Anfragen an die Schnittstelle verarbeitet und Anfragen an das Frontend weitergeleitet werden. Anfragen zu einer REST-Schnittstelle beginnen per Konvention mit /api. Die Weiterleitungen an das Frontend funktioniert, indem auf die index.html Datei verwiesen wird. Im Frontend (Angular-Anwendung) angekommen, übernimmt das Angular-Routing, welches jede Route zur dazugehörigen Komponente weiterleitet. Vor dem Öffnen einer Route überprüft das Frontend über einen Guard, ob der jetzige Benutzer diese Route öffnen darf. Falls dieser nicht authentifiziert ist wird auf die Login-Seite verwiesen. Nach dem sich ein Benutzer angemeldet hat wird er wieder auf die Website verwiesen, welcher er zuvor angestrebt hat. Dies ermöglicht unter anderem das Annehmen einer Einladung, obwohl man sich zuerst anmelden oder registrieren muss.

### Authentifizierung

Die Authentifizierung ist über eigene Sessions umgesetzt. Das Frontend wird über eine Session authentifiziert, welche üblicherweise beim Anmelden erstellt wird. Das Frontend speichert diese Session lokal im Browser ab, damit der Benutzer sich möglichst selten anmelden muss. Optional kann eine Session über die URL mitgegeben werden, indem an eine beliebige URL «?authToken={Token}» angehängt wird. Dies wird für die Zugangslinks verwendet und ist im Frontend durch einen Guard realisiert. Dieser Guard überprüft beim Aufruf einer Route, ob der Benutzer eine Session besitzt. Ist dies nicht der Fall, aber ein Token wurde über die URL angegeben oder im Browser-Speicher gefunden, wird versucht die Session vom Server zu laden. Das Backend erkennt dabei keinen Unterschied, ob das Token aus einer E-Mail oder dem Browser-Speicher kommt. Bei allen Anfragen an das Backend wird das geprüfte Token mitgegeben. Im Backend stellt das AuthenticateUserAttribute sicher, dass die Session noch gültig ist und speichert die jetzige Benutzer-Id in der Abfrage, damit weiterer Code einfach auf diese zugreifen kann. Ist die Session nicht mehr gültig, wird die Anfrage mit dem HTTP-Code 401: Unauthorized abgewiesen.

### UTC

Obwohl alle Benutzer der Anwendung aus der gleichen Zeitzone kommen, stellt das Verwalten der Termin-Zeitpunkte ein Problem dar, weil der Server in einer anderen Zeitzone liegt. Um diese Problematik zu lösen werden alle Zeitpunkte in UTC gespeichert. Ein weiteres Problem stellt die Sommer- und Winterzeit dar, wodurch die Abweichung von UTC je nach Datum variiert. Dieses Problem wird durch die Verwendung von Datums-Frameworks wie Moment.js korrigiert. Konkret wird bereits beim Speichern der Zeitpunkte das Datum berücksichtigt, wodurch für die Zeitverschiebung kompensiert wird. Dieses Jahr beginnt die Sommerzeit am 25. März. Plant man beispielsweise zwei Termine um 12:00 für den 24.03 und 26.03, so wird die Uhrzeit in der Datenbank anders gespeichert: Am 24.03 wird 11:00 erfasst und am 26.03 10:00. Diese kompensierte Zeit wird bei der Darstellung an den Benutzer wieder in die andere Richtung angeglichen, weshalb dieser korrekterweise zweimal 12:00 sieht.

## Backend

Eine Beschreibung der wichtigsten Konzepte und Services im Backend.

### Logging-Konfiguration

Das Logging ist wie in der Aufgabenstellung verlangt konfiguriert. Der Betreiber der Website kann in der Datei «log4net.config» diverse Konfigurationen zur Laufzeit vornehmen und beispielsweise das Log-Level ändern. Die erstellten Log-Dateien werden unter dem Verzeichnis «D:\home\LogFiles\HeyImIn» der Azure-Maschine gespeichert. Die Log-Dateien und Konfiguration können über FTP oder direkt in Azure über «Erweiterte Tools» verwaltet werden. Zusätzlich werden alle Logs in Azure über den Log-Stream ausgegeben, was eine Real-Time-Analyse ermöglicht. Ein Ausschnitt solcher Log-Dateien kann im Anhang unter «15.1 Logdateien der Tests» gefunden werden.

Die All-Log-Dateien werden wie spezifiziert nach 20 Megabyte getrennt, wobei bis zu fünf Backups behalten werden. Die Backups können neben der aktiven Datei gefunden werden und enden mit einer Zahl. Beispielsweise wäre die Datei «HeyImIn-All.5.log» das fünfte Backup, welches in kürze gelöscht wird. Die Audit-Log-Dateien enden mit dem jetzigen Jahr und Monat, wodurch jede Datei einem Monat zugewiesen werden kann. Hier werden Dateien nach 6 Monaten gelöscht.

### Logging einer Anfrage & Log-Context – LogActionAttribute

Für jede Anfrage wird zu Beginn eine Stoppuhr gestartet und der Log-Context definiert. In dem Context sind die Benutzer-Id und ein Teil des Session-Tokens hinterlegt. Zusätzlich wird zum Beginn jeder Anfrage eine Stoppuhr gestartet. Am Ende der Abfrage wird dann über einen Logeintrag festgehalten, wie lange der Aufruf der Aktion gedauert hat. Dadurch könnte ein Entwickler beispielsweise analysieren, welche Abfragen Optimierungspotential besitzen.

### Error-Handling – LogFatalExceptionLogger

Das Programm ist gegenüber erwarteten Fehlerfällen, wie der Registration einer bereits verwendeten E-Mail-Adresse, abgesichert und behandelt diese explizit. Im Falle einer unerwarteten und unbehandelten Exception reagiert der Code minimal. Einen Grossteil der Fehlerbehandlung übernimmt das ASP.NET Framework. Ein registrierter ExceptionLogger schreibt Details wie den Stacktrace zu dem unbehandelten Fehler mit dem Log-Level FATAL in die Log-Datei, während die Anfrage mit dem HTTP-Code 500: Internal Server Error abbricht.

### Konfigurationswerte

Alle Konfigurationswerte werden beim Start der Webanwendung geladen. Die sensitiven Konfigurationswerte sind über Umgebungsvariablen, während weitere Konfigurationen über die Web.config Datei festgelegt werden. Diese Datei kann über FTP oder direkt in Azure über «Erweiterte Tools» verwaltet werden. Folgende Werte sind konfigurierbar:

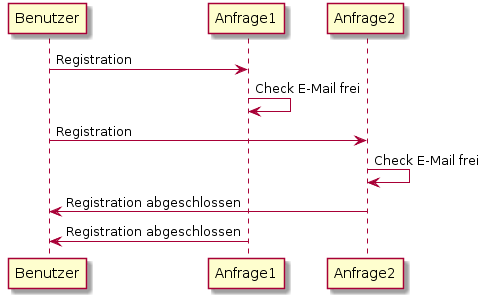
|  |  |  |
| --- | --- | --- |
| Name | Beschreibung | Standard |
| PasswordHashWorkFactor | Die Hash-Stärke der Passwörter. Ein höherer Wert benötigt mehr Rechenleistung, um einen Passwort-Hash zu erzeugen. | 10 |
| FrontendBaseUrl | Die URL, welche bei E-Mails verwendet wird. Entspricht normalerweise der Hosting-URL. | https://hey-im-in.azurewebsites.net/#/ |
| MailTimezoneName | Name der Zeitzone, welche für Daten innerhalb von E-Mails verwendet wird | W. Europe Standard Time |

Tabelle 46: Konfigurationswerte

### Dependency Injection

Um zukünftige Erweiterungen und automatisierte Testfälle zu ermöglichen, wurde die Dependency Injection Autofac konfiguriert. Zur grundlegenden Konfiguration wurde der AssemblyScanner aus einem anderen Projekt verwendet. Zusätzlich werden die konfigurierten Werte werden über die Dependency Injection an die konkreten Implementationen übergeben, wodurch die Klassen keinen Kontakt mit der Beschaffung der Konfigurationswerte haben.

### Input-Validierung

Die grundlegende Input-Validierung wird im Frontend umgesetzt. Zusätzlich werden alle Anfragen im Backend auf ihre Integrität überprüft. Für simple Parameter-Überprüfungen wird der ModelState verwendet, welcher auf die Attribute der Modelle reagiert. Dadurch wird im Code einheitlich der ModelState überprüft, ohne die konkreten Grenzwerte der Parameter kennen zu müssen. Auf den Parameter werden die Grenzwerte über die zuvor erwähnten Attribute in einem leserlichen Format beschrieben und können einfach angepasst werden.

Weitere Validierungen, wie die Sicherstellung einer eindeutigen E-Mail-Adresse, werden im Code explizit durchgeführt. Theoretisch könnte ein Benutzer seine Registration zweimal nacheinander absenden und somit die Validierung umgehen. Um dies zu verhindern, sind diese Überprüfungen auf der Datenbank ebenfalls durch die Verwendung von Indices realisiert.

Abbildung 15: Beispiel Race Condition – Zwei Registrationen mit derselben E-Mail-Adresse

### Datenbankmigrationen

Das verwendete OR-Mapping wurde in der Konfiguration Code-First verwendet. Dies erlaubt das automatische Aufsetzen der Tabellenstruktur, sobald diese von der Web-Anwendung zum ersten Mal verwendet wird. Ein zusätzlicher Anwendungsfall ist das aktualisieren des Schemas, beispielsweise durch das Hinzufügen einer Teilnahmegebühr. Schemaänderungen müssen vom Entwickler lokal durchgeführt werden, woraufhin dieser eine Migration erstellt und dem Source-Code beilegt. Um das Aktualisieren der Software möglichst reibungslos zu gestalten, wird diese Migration beim Start automatisch durchgeführt.

### Async-Await

Im Backend wurden alle IO-bedingten Operationen, namentlich die Datenbankabfragen, asynchron durch async-await umgesetzt. Dies erlaubt einem Thread mehrere Anfragen anzunehmen, während die laufenden Anfragen auf die Datenbank warten. Als Resultat müssen unter Last weniger Threads erstellt werden.

### Lazy-Loading

Das eingesetzte OR-Mapping lädt nur die Daten der Datenbank, welche angefragt werden. Dies hat zur Folge, dass ein Benutzer beispielsweise ohne seine organisierten Events und Eventteilnahmen geladen wird. Möchte man diese Werte trotzdem verwenden, werden weitere Datenbankabfragen durchgeführt. Um die Datenbank nicht zu überfordern, wurden bei häufigen Anfragen angegeben, welche Relationen direkt mitgeladen werden sollten.

### Benutzer-Passwort – IAuthenticationService & IPasswordService

Bei jeder Anmeldung eines Benutzers wird zuerst überprüft, ob sein Passwort korrekt ist. Falls seine Benutzerdaten korrekt sind, wird sichergestellt, dass sein Passwort nach dem konfigurierten Sicherheitsstandard gespeichert wird. Falls das Sicherheitsniveau des Passwort-Hashs unter dem konfigurierten «PasswordHashWorkFactor» liegt, wird ein aktualisierter Hash erstellt und gespeichert. Dies garantiert den fliessenden Übergang zu sichereren Passwort-Hashes, sobald die Rechenleistung stark zugenommen hat.

### Session-Verwaltung – ISessionService

Bei jeder authentifizierten Anfrage verlängert das Backend die (gültige) aktive Session automatisch. Dies soll dafür sorgen, dass ein Benutzer bei täglicher Benutzung keine Login-Maske ausfüllen muss.

### Löschen von Daten – IDeleteService

Da beim Löschen von Daten üblicherweise Notifikationen versendet werden, werden Relationen manuell und explizit gelöscht. Auf der Datenbank ist kaskadierendes löschen deaktiviert, was verhindert, dass unbeabsichtigte Daten gelöscht werden. Da die Notifikationen erst nach dem erfolgreichen Löschen versendet werden, müssen gewisse Daten, wie beispielsweise die Teilnehmer, im Arbeitsspeicher gehalten werden. Da die Löschreihenfolge und das «In-Memory-Backup» der Daten an mehreren Orten benötigt wird, wurde ein IDeleteService implementiert. Dieser wird für das Löschen von Benutzern, Events und Terminen verwendet.

### Cron-Jobs – ICronService

Obwohl es momentan nur einen Cron-Job zum Versenden von Notifikationen gibt, wären zukünftige Jobs zum Aufräumen der Datenbank bestimmt sinnvoll. Damit diese einfach implementiert werden können, wurde ein ICronService-Interface erstellt, welches von den verschiedenen Cron-Jobs implementiert wird. Um einen weiteren Cron-Job hinzuzufügen muss der Typ lediglich in der Dependency Injection hinterlegt werden, damit dieser regelmässig aufgerufen wird.

## Frontend

Eine Beschreibung der wichtigsten Konzepte und Services im Backend.

### AppModule

Das App-Modul ist der Einstiegspunkt der Angular-Applikation und lädt die weiteren Module, Routen und globalen Styles. Zusätzlich wird die Sprache auf «de-CH» festgelegt, damit die Startzeiten der Events im korrekten Format dargestellt werden.

### SharedModule

Das Modul, welches geteilten Code zwischen mehreren Modulen beherbergt. Dies beinhaltet die Authentifizierung, Main-Layout und die drei Dialoge für Fehlermeldungen, Ladeindikation und «Sind Sie sicher?».

### AnonymousModule

Das Modul, welches die drei Komponenten Login, Register und ResetPasswort beinhaltet.

### AuthenticatedModule

Das Modul, welches die Komponenten zur Eventverwaltung und -teilnahme beinhaltet.

### Interceptors

Ein Interceptor schaltet sich zwischen jede Web-Anfrage und führt Code aus. In unserem Fall gibt es drei Code-Blöcke, welche für alle Anfragen durchgeführt werden: Im AppendSessionTokenInterceptor wird das Session-Token zur Authentifizierung in den Header geschrieben, damit dies nicht bei jeder Anfrage explizit gemacht werden muss. In einem zweiten Schritt notiert sich der ShowLoadingDialogInterceptor in einem internen Zähler, dass eine Anfrage gestartet hat. Sobald die Anfrage abgeschlossen wurde wird der Zähler wieder reduziert. Damit schnell aufeinanderfolgende Abfragen kein Flackern in der Benutzeroberfläche auslösen bleibt der Dialog für bis zu 200 Millisekunden geöffnet, nachdem keine Anfragen mehr laufen. Startet in dieser Zeit eine neue Anfrage, bleibt der bestehende Dialog geöffnet. In dem letzten Schritt fängt der ErrorHandlerInterceptor fehlerhafte HTTP-Antworten ab. Für Fehler bezüglich der Benutzereingabe (400) wird eine SnackBar-Meldung unten am Bildschirm eingeblendet. Im Falle eines Authentifizerungsproblems (401) wird die lokale Session als invalid deklariert und auf das Login weitergeleitet. Für weitere Fehler wird ein generischer «Etwas ist schief gelaufen»-Dialog angezeigt.

### Dialoge

Zur Darstellung der Dialoge wird das von Angular-Material bereitgestellte Framework verwendet. Dazu wird für jeden anzuzeigenden Dialog eine eigene Komponente deklariert. Ein häufig verwendeter Dialog ist der «Sind Sie sicher?»-Dialog, welcher einen Beschreibungstext als Eingabe verlangt, worin beispielsweise «Möchten Sie diesen Event wirklich löschen?» steht. Falls der Benutzer den Dialog mit Ja bestätigt wird true zurückgegeben, ansonsten false. Die beiden weiteren Dialoge beziehen sich auf das Editieren eines Events. Zum einen wird ein Dialog für das Einladen von Eventteilnehmern erstellt, welcher die einzuladenden E-Mail-Adressen zurückgibt. Analog wird ein Dialog für das Hinzufügen von Terminen erstellt, welcher die hinzuzufügenden Terminstartpunkte zurückgibt.

### Eventverwaltung

Für die Eventverwaltung wurden diverse Komponenten erstellt. Die wichtigste Komponente ist die AppointmentParticipationComponent, welche die drei Auswahlmöglichkeiten durch Knöpfe wiederspiegelt. Zusätzlich wurden separate Komponenten erstellt, wo eine Funktion an mehreren Orten der Benutzeroberfläche verwendet wird. Dies beinhaltet das Verwalten der allgemeinen Eventinformationen und die Teilnehmerübersicht eines Termins.

### Main-Layout

Das globale Layout wurde in einer Main-Layout-Komponente umgesetzt, welche von den drei Modulen verwendet wird. Da die Navigationsleiste je nach Seite ändert, sind die jeweiligen Layouts der Module im Routing als übergeordnete Komponente deklariert. Diese modulspezifischen Layouts ergänzen das globale Layout um eine Liste von Links, welche in der Navigationsleiste angezeigt wird. Die Darstellung der Navigationsleiste wurde in Bezug auf das Design aus einem anderen Projekt inspiriert und könnte leicht angepasst werden.

### Responsive Design

Das Grunddesign ist bereits Mobilefreundlich gehalten und wurde auf kleinen Bildschirmen nur durch kleinere Seitenabstände optimiert. Stattdessen wurden Verbesserungen für grössere Bildschirme umgesetzt. Beispielsweise wird die Beschreibung eines Events in der Übersicht und den Details neben den sonstigen Informationen angezeigt, sobald der Bildschirm eine gewisse Grösse besitzt.

### Input-Validierung

Für die Benutzereingaben wurden die von Angular und Angular-Material bereitgestellten Werkzeuge verwendet. Die verwendeten Input-Elemente besitzen eine Kennzeichnung für Pflichtfelder und können vom Browser automatisch ausgefüllt werden. Durch das Verwenden von ReactiveForms, einer Möglichkeit Formulare via Code zu konfigurierten, wurden Validierungen wie Maximallängen und Pflichtfelder deklariert. Diese Validierungsfehler können in der Benutzeroberfläche spezifiziert und dargestellt werden. Die Fehlermeldungen werden erst dargestellt, wenn das Formular dirty ist, also ein Wert verändert wurde. Dies verhindert, dass der Benutzer beim Öffnen eines Formulars mit Fehlermeldungen überhäuft wird. Sobald ein Feld innerhalb eines Formulars ungültig ist, kann das Formular nicht mehr abgesendet werden und der Senden-Knopf wird ausgegraut. Dasselbe gilt, wenn ein Formular nicht dirty ist und noch nicht verändert wurde. Dies verhindert, dass ein Benutzer auf einer Seite «Speichern»-Knöpfe aufleuchten sieht, obwohl er nichts geändert hat.

# 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 | 06.03.2018 15:30 |
| Planen | 08.03.2018 12:00 | 08.03.2018 12:00 |
| Entscheiden | 08.03.2018 17:00 | 08.03.2018 16:30 |
| Implementation | 16.03.2018 12:00 | 16.03.2018 11:00 |
| Realisieren | 20.03.2018 10:00 | 16.03.2018 09:30 |
| Kontrollieren | 20.03.2018 17:00 | 20.03.2018 15:00 |
| 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 und erweiterte explorative Tests durchgeführt.

### Legende

✔ = Bestanden

✔ (**~**) = Bestanden mit Abweichung

✘ = Fehlgeschlagen

### Explorative Tests

Vor den funktionalen Testfällen wurden explorative Testfälle durchgeführt. In diesen ging es darum, die Applikation wie ein normaler Benutzer mit realen Events auf die Probe zu stellen, ohne strikte Testdaten zu verwenden.

|  |  |  |
| --- | --- | --- |
| Testfall | Beschreibung | Resultat |
| Allgemeine Funktionalität | Mehrere reale Events wurden exemplarisch getestet | Keine Fehler festgestellt |
| Input-Validierung | Für alle Formulare wurden Grenzwerte und Fehleingaben versucht. Auch exotische Schriftzeichen wie Emojis wurden eingegeben. | Für die Zeitangaben in Stunden waren keine plausiblen Grenzwerte definiert, korrigiert in Commit a7e0b88687d64dadb530d3e89fe2d2d528ec47c4 |
| Erweiterter Browsersupport | Zusätzlich zu den vorgegebenen Browsern wurden die neuste Version des Mozilla Firefox (Desktop) und Safari (Mobile) getestet. Es wurden nur die wichtigsten und nicht alle Testfälle durchgeführt. | Keine Fehler festgestellt |
| Konsistente Darstellung | Damit die Applikation seriös wirkt, sollten die verschiedenen Seiten ein durchgängiges Layout haben. | Die Seitenüberschrift war auf gewissen Seiten abwesend oder falsch formatiert, korrigiert in Commit 9b84dd420892479fa145666f5d24d7735b8badad |
| Responsive Design | Neben dem Test auf einem Mobilgerät wurden im Chrome alle Seiten mit diversen Bildschirmauflösungen getestet. | Profil-Knopf war unter gewissen Geräten abgeschnitten, korrigiert in Commit dee513b590a22f37f1f31a23f72e56a3e1b038db |
| Log-Rotation | Da ich weder sechs Monate testen noch 20MB an Logdateien erstellen kann, wurden diese Werte für die Tests angepasst. Hierdurch kann sichergestellt werden, dass die Konfiguration auf längere Zeit hinaus wie erwartet funktioniert. | Wegen einer fehlerhaften Konfiguration wurden alte Dateien nicht gelöscht, korrigiert in Commit 6d7d5e9160a71033f523f4ed67f68d01b7e606c4 |
| Notifikationen | Für alle Notifikationen wurde getestet, ob ein Notifikationslink vorhanden ist und funktioniert. Zusätzlich zu den explizit getesteten Notifikationen wurden folgende E-Mails getestet:   * Kurzfristige Änderungen * Organisator Ändert Teilnahme eines anderen Teilnehmers * Organisator entfernt Teilnehmer | Notifikationen wurden teilweise mehrere Tage zu früh versendet, korrigiert in Commit 6d7d5e9160a71033f523f4ed67f68d01b7e606c4 |
| Mehrfache Verwendung von Token | Die per E-Mail versendeten Tokens sollten nach einer Zeitspanne ablaufen. Zusätzlich sollten die Einladungen zu privaten Events und Passwort-Zurücksetzungs-Anfragen nur einmal funktionieren. Um das Ablaufen zu testen, wurde die Ablauffrist von mehreren Tagen auf drei Minuten reduziert. | Keine Fehler festgestellt |
| Authentifizierung | Folgende erweiterte Tests wurden durchgeführt:   * Abmelden erklärt die Session auf dem Server für ungültig und löscht die gespeicherten Daten aus dem Browser * Nicht angemeldete Benutzer werden konsequent auf das Login weitergeleitet * Mit dem Detail-Link zu einem privaten Event, an dem man nicht teilnimmt, werden keine Daten angezeigt | Keine Fehler festgestellt |

### Testresultate 1: Benutzer

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 1.1.1 | ✔ |  |
| 1.1.2 | ✔ |  |
| 1.2.1 | ✔ |  |
| 1.2.2 | ✔ |  |
| 1.3.1 | ✔ |  |
| 1.3.2 | ✔ |  |
| 1.4.1 | ✔ (**~**) | Beim Öffnen der Profilseite, ohne sich erneuten anzumelden, wurden die alten Daten angezeigt, korrigiert in Commit eaa84ea3dde21ad06694c3ecc0b497fb7d65407d |
| 1.5.1 | ✔ |  |

Tabelle 47: Testresultate 1

### Testresultate 2: Teilnehmer

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 2.1.1 | ✔ |  |
| 2.1.2 | ✔ |  |
| 2.2.1 | ✔ |  |
| 2.2.2 | ✔ |  |
| 2.3.1 | ✔ |  |
| 2.3.2 | ✔ |  |
| 2.3.3 | ✔ |  |
| 2.4.1 | ✔ |  |
| 2.4.2 | ✔ |  |
| 2.4.3 | ✔ |  |
| 2.5.1 | ✔ |  |

Tabelle 48: Testresultate 2

### Testresultate 3: Organisator

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 3.1.1 | ✔ (**~**) | Weiterleitung war auf die Übersichts-Seite anstelle der Edit-Seite, korrigiert in Commit ae79b7e17c99e19101ec5bedbb90c3d60117571c |
| 3.2.1 | ✔ |  |
| 3.3.1 | ✔ |  |
| 3.4.1 | ✔ |  |
| 3.5.1 | ✔ |  |
| 3.6.1 | ✔ |  |
| 3.6.2 | ✔ |  |
| 3.6.3 | ✔ |  |
| 3.7.1 | ✔ |  |
| 3.8.1 | ✔ |  |

Tabelle 49: Testresultate 3

## Testresultate: Browserkompatibilität

|  |  |  |  |
| --- | --- | --- | --- |
| Browser | Durchgeführte Tests | Abweichung | Unterstützt |
| Mozilla Firefox 59 | Exploratives Testen & ausgiebiges Testen während der Entwicklung | - | ✔ (~) |
| Safari 11 (Mobile) | Exploratives Testen | - | ✔ (~) |
| Google Chrome 65 | Exploratives Testen & Testfälle | - | ✔ |
| Internet Explorer 11 | Exploratives Testen & Testfälle | Animationen nicht immer flüssig, jedoch nicht störend. | ✔ |

Tabelle 50: Testresultate Browserkompatibilität

Der Browsersupport ist gewährleistet.

## Testfazit

Die Funktionalität der Applikation wurde durch definierte Testfälle und exploratives Testen sichergestellt. Eine Erweiterung zu den manuellen Tests wären automatisierte Unit-Tests, die im Rahmen dieses Projekts wegen der zeitlichen Einschränkung nicht umgesetzt wurden. Das Projekt wurde so konzipiert, dass sowohl Back- wie Frontend automatisiert getestet werden könnten. Von den 29 Testfällen wurden 27 fehlerfrei bestanden. Bei den Testfällen 1.4.1 (Profil bearbeiten) und 3.1.1 (Event erstellen) wurden jeweils Fehler entdeckt und behoben. In beiden Fällen waren die nötigen Korrekturen minimal und bedenkenlos einzupflegen. Zusätzlich wurden während der Durchführung der explorativen Tests fünf weitere Fehler festgestellt, welche etwas grössere Korrekturen benötigten, aber auch korrigiert wurden. Mit der aktuellsten Version 1.0.2 sind alle bekannten Fehler korrigiert und Testfälle bestanden, weshalb das Testfazit sehr positiv ausfällt. Die gewünschte Funktionalität ist komplett umgesetzt und das Programm kann bedenkenlos produktiv eingesetzt werden.

# Auswerten

Das komplette Projekt wird nochmals reflektiert und ausgewertet.

## Reflexion

Das Projekt ist abgeschlossen und es wird Zeit zurückzublicken. Ich bin sehr zufrieden damit, wie sich das Projekt entwickelt hat. Meiner Meinung nach sieht die Webanwendung professionell aus und ist voll funktionsfähig. Ich bin immer noch verblüfft darüber, wie viel Quellcode in einer so kurzen Zeit entstanden ist. Nach den Git-Statistiken habe ich, die initiale Aufsetzung und generierte Dateien weggelassen, elf tausend Zeilen Code geschrieben. Dazu habe ich diesen IPA-Bericht verfasst, welcher auch genügend Wörter umfasst. Natürlich ist Quantität nicht alles, jedoch bin ich auch mit der Qualität zufrieden. Zu Beginn gab es einige Codestellen, welche ich nicht ideal fand. Diese konnte ich zum Glück fortlaufend aufbessern und leserlich gestalten. Jedoch lief im Projekt nicht alles perfekt. Der Grösste Schwachpunkt der Arbeit ist der Zeitplan, bei welchem ich gewisse Arbeitsschritte sogar ganz weggelassen habe. So habe ich keine Zeit für das Aufbereiten IPA-Berichts eingeplant. Konkret wurden die Kurzfassung und der Anhang nicht eingeplant, obwohl diese mehrere Stunden in Anspruch nehmen. Glücklicherweise konnte ich die letzten beiden IPERKA-Phasen, das Kontrollieren und Auswerten, frühzeitig abschliessen und die Arbeit erfolgreich abschliessen.

## Erfahrungen

Das Projekt war sehr spannend und ich konnte viel Wissen zu bekannten Teilbereichen verbessern. So habe ich in anderen Projekten eine Dependency Injection angewandt, jedoch musste ich diese noch nie selber aufsetzen. Für Schulprojekte hatte ich aus Zeitgründen immer den herkömmlichen Weg gewählt und die konkreten Abhängigkeiten von Hand aufgelöst. Da ich nun eine DI konfigurieren kann, werde ich keine zukünftigen Projekte ohne DI durchführen. Dasselbe gilt für Logging, welches ich zwar verwendet, jedoch nie konfiguriert habe. Die Log-Dateien waren mir bei mehreren Fehleranalysen behilflich, weshalb ich diese auch für weitere Projekte erstellen möchte.

Eine erstaunlich schmerzfreie Erfahrung hatte ich mit dem Umgang von Zeitzonen, der Datenbankanbindung und dem Senden von E-Mails. So habe ich alle Komponenten und Klassen für das Speichern und Darstellen der Zeitpunkte korrekt angewandt, wodurch sogar Sommer- und Winterzeit berücksichtigt werden. Ähnlich musste ich für das Entity Framework die korrekte Datenbankanbindung konfigurieren und konnte die Datenbanktabellen wie normale Listen verwenden. Das Versenden von E-Mails über die SendGrid-Bibliothek war analog zu einem normalen Methodenaufruf, ohne dass ich Probleme mit unerreichbaren E-Mail-Adressen oder E-Mail-Servern hatte. Für diese Konfigurationen musste ich auch zum ersten Mal Konfigurationen aus den Umgebungsvariablen lesen, damit keine sensitiven Konfigurationen im Quellcode stehen. Diesen Ansatz möchte ich auch für weitere private Projekte verwenden, da diese normalerweise Open-Source und somit für jeden Einsehbar sind.

## Einsetzbarkeit

Die Webapplikation wird in einem nächsten Schritt mit dem Auftraggeber Christoph Fauti angeschaut und nach Bedarf angepasst. Danach wird die Organisation des Badmintons migriert, während weitere Projekte fortlaufend auf die neue Organisationsplattform wechseln. Da die geforderte Funktionalität in allen Browsern und Mobile funktioniert, ist dieses Tool bedenkenlos einsetzbar.

## Erweiterungen

Es gab bereits weitere Mitarbeiter, welche mich bezüglich einer Erweiterung der jetzigen Lösung kontaktiert haben, obwohl diese noch nicht veröffentlich ist. Bevor ich jedoch weitere Funktionen erweitern möchte, plane ich das Aufsetzen von automatisierten Tests, damit keine Regressionen entstehen. Die ersten Erweiterungen wurden bereits in den Mockups berücksichtigt und grau hinterlegt. Die Preisverwaltung benötigt eine kleine Erweiterung des Datenschemas und sollte in ein bis zwei Tagen integriert sein. Die Integration einer Chat-Funktion könnte mehr Aufwand bedeuten, weshalb ich den Aufwand nicht abschätzen kann. Weitere Funktionswünsche können agil und schrittweise hinzugefügt werden.

## Schlusswort

Das Projekt «Hey, I’m in» ist abgeschlossen und ich schaue auf ein erfolgreiches Projekt zurück. Ich konnte mein Wissen in der Webentwicklung vertiefen und durfte zum ersten Mal ein grösseres Projekt ausserhalb der Schule aufsetzen. Während der Entwicklung kam ich mit diversen Themen wie Zeitzonen, Responsive Design und E-Mail-Notifikationen in Kontakt, wobei ich vieles für zukünftige Projekte gelernt habe. Leider war die Zeitplanung nicht immer zutreffend, was an ungeplanter und vergessen gegangener Arbeit lag, die zu Beginn nicht berücksichtigt wurden. Im Grossen und Ganzen betrachtet ging trotzdem alles auf und das Projekt wurde rechtzeitig fertig.

«Hey, I’m in» wird für Doodle in naher Zukunft zwar keine Konkurrenz darstellen, dies war jedoch auch nie das Ziel. Für die Organisation von wiederkehrende Events könnte diese neue Lösung mindestens in der Atos auf eine positive Resonanz stossen und vielleicht auch über die Atos hinaus Einsatz finden. Alles in allem bin ich mit der Planung, Umsetzung und Dokumentation der Arbeit sehr zufrieden und freue mich auf die Entwicklung der nächsten Version.

# Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Azure | Die Cloud- Plattform von Microsoft, auf welcher unter anderem Datenbanken und Webanwendung gehostet werden können. |
| Backend | Die Business-Logik, welche auf einem Server läuft. |
| Cron-Job | Eine regelmässig durchzuführende Aufgabe. |
| Dependency Injection (DI) | Mithilfe einer Dependency Injection können die konkreten Implementation einer Schnittstelle zur Laufzeit ermittelt werden. Dadurch sind die Abhängigkeiten einer Klasse als Schnittstellen definiert und von der effektiven Implementation losgelöst. Dies verbessert die Wartbarkeit und Testbarkeit des Codes drastisch. |
| DTO (Data Transfer Object) | Ein Datentransferobjekt ist ein Objekt, welches keine Logik besitzt und nur zum Übertragen von Daten verwendet wird. Dies sind beispielsweise Objekte, welche mithilfe von Serialisierung über das Netzwerk transportiert werden. |
| Frontend | Die Darstellungs-Logik, welche im Browser läuft. |
| GUID (UUID) | Eine **U**niversally **U**nique **Id**entifier (UUID) ist eine eindeutige Identifikation einer Entität. Microsofts Implementierung einer UUID heisst **G**lobally **U**nique **Id**entifier (GUID). |
| Rainbow Table | Eine Rainbow Table wird üblicherweise zum Herausfinden von Passwörtern verwendet. Da diese normalerweise als Hash gespeichert werden, kann man den originalen Text auf diese Weise nicht herausfinden. Eine Möglichkeit dieses Problem zu umgehen ist, eine Tabelle mit den am meisten verwendeten Passwörtern zu erstellen. Als Angreifer kann man denn Hash in der Tabelle suchen und findet dazu das passende Passwort. Dies funktioniert am besten, wenn alle Passwörter gleich gehasht werden, also kein Salt verwendet wird. |
| Regex (Regular Expression) | Regex ist eine Syntax um Muster zu definieren, mit welchen Text durchsucht werden kann. So kann ein Regex-Muster bestimmen, dass eine Eingabe wie eine E-Mail-Adresse auszusehen hat. |
| Salt (Password Hash) | Ein Salt ist ein üblicherweise generierter Wert, welcher einem Passwort vor dem Hashen hinzugefügt wird. Der verwendete Salt wird mit dem Passwort gespeichert und beim Überprüfen verwendet. Salts werden eingesetzt, um Rainbow Table Angriffe zu erschweren |

Tabelle 51: Glossar

# Verzeichnisse

## Abbildverzeichnis

Die Abbildungen in diesem Dokument sind alle selbst erstellt.

[Abbildung 1: Zeitplan 10](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321485)

[Abbildung 2: Mockup Login / Registrieren 22](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321486)

[Abbildung 3: Mockup Profil 22](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321487)

[Abbildung 4: Mockup Passwort zurücksetzen 22](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321488)

[Abbildung 5: Mockup Event Details 23](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321489)

[Abbildung 6: Mockup Events 23](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321490)

[Abbildung 7: Mockup Event Bearbeitung 24](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321491)

[Abbildung 8: Systemübersicht 25](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321492)

[Abbildung 9: Use-Cases – Übersicht 26](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321493)

[Abbildung 10: Use-Cases – Benutzer 26](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321494)

[Abbildung 11: Use-Cases – Teilnehmer 28](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321495)

[Abbildung 12: Use-Cases – Organisator 30](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321496)

[Abbildung 13: Backend Komponentendiagramm 38](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321497)

[Abbildung 14: ERD 40](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321498)

[Abbildung 15: Beispiel Race Condition – Zwei Registrationen mit derselben E-Mail-Adresse 49](file:///C:\Users\A610222\Source\Repos\HeyImIn\Dokumentation\IPA-Dokumentation.docx#_Toc509321499)

## Tabellenverzeichnis

[Tabelle 1: Projektteilnehmer 7](#_Toc509321500)

[Tabelle 2: Meilensteine 9](#_Toc509321501)

[Tabelle 3: Konkrete Planung Tag 1 11](#_Toc509321502)

[Tabelle 4: Konkrete Planung Tag 2 12](#_Toc509321503)

[Tabelle 5: Konkrete Planung Tag 3 13](#_Toc509321504)

[Tabelle 6: Konkrete Planung Tag 4 14](#_Toc509321505)

[Tabelle 7: Konkrete Planung Tag 5 15](#_Toc509321506)

[Tabelle 8: Konkrete Planung Tag 6 16](#_Toc509321507)

[Tabelle 9: Konkrete Planung Tag 7 17](#_Toc509321508)

[Tabelle 10: Konkrete Planung Tag 8 18](#_Toc509321509)

[Tabelle 11: Konkrete Planung Tag 9 19](#_Toc509321510)

[Tabelle 12: Konkrete Planung Tag 10 20](#_Toc509321511)

[Tabelle 13: Use-Case 1.1 – Anmelden 26](#_Toc509321512)

[Tabelle 14: Use-Case 1.2 – Registrieren 27](#_Toc509321513)

[Tabelle 15: Use-Case 1.3 – Passwort zurücksetzen 27](#_Toc509321514)

[Tabelle 16: Use-Case 1.4 – Persönliche Daten aktualisieren^ 27](#_Toc509321515)

[Tabelle 17: Use-Case 1.5 – Anmelden 28](#_Toc509321516)

[Tabelle 18: Use-Case 1.6 – Benutzerkonto löschen 28](#_Toc509321517)

[Tabelle 19: Use-Case 2.1 – Event teilnehmen 28](#_Toc509321518)

[Tabelle 20: Use-Case 2.2 – Event verlassen 29](#_Toc509321519)

[Tabelle 21: Use-Case 2.3 – Zusage eines Termins verwalten 29](#_Toc509321520)

[Tabelle 22: Use-Case 2.4 – Zusage eines Termins verwalten 29](#_Toc509321521)

[Tabelle 23: Use-Case 2.5 – Notifikationen deaktivieren 30](#_Toc509321522)

[Tabelle 24: Use-Case 3.1 – Neuen Event organisieren 30](#_Toc509321523)

[Tabelle 25: Use-Case 3.2 – Informationen eines Events anpassen 31](#_Toc509321524)

[Tabelle 26: Use-Case 3.3 – Termine zu einem Event hinzufügen 31](#_Toc509321525)

[Tabelle 27: Use-Case 3.4 – Termin absagen 31](#_Toc509321526)

[Tabelle 28: Use-Case 3.5 – Teilnahme für verhinderten Teilnehmer absagen 31](#_Toc509321527)

[Tabelle 29: Use-Case 3.6 – Teilnehmer einladen 32](#_Toc509321528)

[Tabelle 30: Use-Case 3.7 – Inaktiven Benutzer entfernen 32](#_Toc509321529)

[Tabelle 31: Use-Case 3.8 – Event löschen 32](#_Toc509321530)

[Tabelle 32: Testfälle 1 – Benutzer 34](#_Toc509321531)

[Tabelle 33: Testfälle 2 – Teilnehmer 35](#_Toc509321532)

[Tabelle 34: Testfälle 3 – Organisator 37](#_Toc509321533)

[Tabelle 35: Backend Modulübersicht 38](#_Toc509321534)

[Tabelle 36: Entitätsbeschreibung Session 40](#_Toc509321535)

[Tabelle 37: Entitätsbeschreibung User 41](#_Toc509321536)

[Tabelle 38: Entitätsbeschreibung PasswordReset 41](#_Toc509321537)

[Tabelle 39: Entitätsbeschreibung EventParticipation 41](#_Toc509321538)

[Tabelle 40: Entitätsbeschreibung Event 42](#_Toc509321539)

[Tabelle 41: Entitätsbeschreibung EventInvitation 42](#_Toc509321540)

[Tabelle 42: Entitätsbeschreibung Appointment 42](#_Toc509321541)

[Tabelle 43: Entitätsbeschreibung AppointmentParticipation 43](#_Toc509321542)

[Tabelle 44: Logdateien 44](#_Toc509321543)

[Tabelle 45: Log-Level 44](#_Toc509321544)

[Tabelle 46: Konfigurationswerte 49](#_Toc509321545)

[Tabelle 47: Testresultate 1 55](#_Toc509321546)

[Tabelle 48: Testresultate 2 55](#_Toc509321547)

[Tabelle 49: Testresultate 3 55](#_Toc509321548)

[Tabelle 50: Testresultate Browserkompatibilität 56](#_Toc509321549)

[Tabelle 51: Glossar 58](#_Toc509321550)

# Quellcode

## Logdateien der Tests

### Audit.log

2018-03-09 16:45:07,445 [ 13] INFO [ | ] Register(): Registered user 1 (Pascal Honegger)

2018-03-09 16:47:40,683 [ 5] INFO [ | ] StartSession(userId=1): User logged in

2018-03-09 16:49:55,005 [ 21] INFO [ | ] StartSession(userId=1): User logged in

2018-03-13 15:34:57,213 [ 6] INFO [ | ] StartSession(userId=1): User logged in

2018-03-14 08:51:43,237 [ 12] INFO [ | ] StartSession(userId=1): User logged in

2018-03-14 12:54:17,485 [ 7] INFO [ | ] StartSession(userId=1): User logged in

2018-03-14 14:07:58,019 [ 29] INFO [ 1 | 8331a3ea] StopActiveSession(): User logged out

2018-03-14 14:08:07,024 [ 19] INFO [ | ] StartSession(userId=1): User logged in

2018-03-14 14:52:35,344 [ 6] INFO [ | ] StartSession(userId=1): User logged in

2018-03-14 15:14:26,815 [ 50] INFO [ 1 | b80c4554] CreateEvent(): Created event 2 (Mein toller Event)

2018-03-14 15:15:11,274 [ 39] INFO [ 1 | b80c4554] DeleteEvent(): Deleted event 2

2018-03-14 15:22:45,446 [ 9] INFO [ 1 | b80c4554] CreateEvent(): Created event 3 (qwer)

2018-03-14 15:22:57,092 [ 5] INFO [ 1 | b80c4554] DeleteAccount(): Deleted user 1 (Pascal Honegger) and all of his events

2018-03-14 15:23:13,924 [ 10] INFO [ | ] Register(): Registered user 2 (Pascal Honegger)

2018-03-14 16:01:06,367 [100] INFO [ | ] StartSession(userId=2): User logged in

2018-03-14 16:01:13,621 [ 97] INFO [ 2 | 6a9db4e6] StopActiveSession(): User logged out

2018-03-14 16:19:40,247 [ 8] INFO [ | ] StartSession(userId=2): User logged in

2018-03-14 16:20:27,436 [ 5] INFO [ 2 | 2207907e] CreateEvent(): Created event 4 (Mein toller Event)

2018-03-14 16:25:39,076 [ 18] INFO [ 2 | 2207907e] JoinEvent(): Joined event 4

2018-03-14 16:25:49,874 [ 27] INFO [ 2 | 2207907e] RemoveFromEvent(): Left the event 4

2018-03-15 07:33:02,726 [ 36] INFO [ | ] StartSession(userId=2): User logged in

2018-03-15 07:37:22,958 [ 55] INFO [ | ] StartSession(userId=2): User logged in

2018-03-15 07:40:50,726 [ 60] INFO [ 2 | e8a76317] CreateEvent(): Created event 5 (Mit Terminen)

2018-03-15 08:55:30,676 [ 57] INFO [ 2 | e8a76317] AddAppointments(): Added 23 appointments to event 5

2018-03-15 08:56:09,217 [ 64] INFO [ 2 | e8a76317] JoinEvent(): Joined event 5

2018-03-15 08:58:29,082 [ 42] INFO [ 2 | e8a76317] DeleteAppointment(): Deleted event 23

2018-03-15 08:59:28,072 [ 43] INFO [ 2 | e8a76317] DeleteAppointment(): Deleted event 22

2018-03-15 09:00:07,082 [ 32] INFO [ 2 | e8a76317] DeleteAppointment(): Deleted event 21

2018-03-15 09:05:40,814 [ 6] INFO [ 2 | e8a76317] DeleteAppointment(): Deleted event 19

2018-03-15 09:05:55,122 [ 16] INFO [ 2 | e8a76317] DeleteEvent(): Deleted event 5

2018-03-15 09:06:18,997 [ 12] INFO [ 2 | e8a76317] CreateEvent(): Created event 6 (Test)

2018-03-15 09:06:22,409 [ 11] INFO [ 2 | e8a76317] JoinEvent(): Joined event 6

2018-03-15 09:20:18,242 [ 12] INFO [ 2 | e8a76317] AddAppointments(): Added 5 appointments to event 6

2018-03-15 09:46:40,783 [ 7] INFO [ | ] StartSession(userId=2): User logged in

2018-03-15 10:54:31,362 [ 74] INFO [ 2 | c8d286c2] JoinEvent(): Joined event 4

2018-03-15 10:54:38,960 [ 84] INFO [ 2 | c8d286c2] RemoveFromEvent(): Left the event 4

2018-03-15 10:54:44,651 [ 82] INFO [ 2 | c8d286c2] RemoveFromEvent(): Left the event 6

2018-03-15 10:55:13,070 [ 88] INFO [ 2 | c8d286c2] JoinEvent(): Joined event 6

2018-03-15 10:55:16,145 [ 88] INFO [ 2 | c8d286c2] RemoveFromEvent(): Left the event 6

2018-03-15 10:55:18,311 [ 88] INFO [ 2 | c8d286c2] JoinEvent(): Joined event 6

2018-03-15 10:55:52,861 [ 75] INFO [ 2 | c8d286c2] RemoveFromEvent(): Left the event 6

2018-03-15 10:58:58,500 [ 11] INFO [ 2 | c8d286c2] JoinEvent(): Joined event 6

2018-03-15 10:59:05,449 [ 16] INFO [ 2 | c8d286c2] RemoveFromEvent(): Left the event 6

2018-03-15 10:59:07,863 [ 13] INFO [ 2 | c8d286c2] JoinEvent(): Joined event 6

2018-03-15 11:03:55,854 [ 5] INFO [ 2 | c8d286c2] DeleteAccount(): Deleted user 2 (Pascal Honegger) and all of his events

2018-03-15 11:07:08,787 [ 7] INFO [ | ] Register(): Registered user 3 (qwer)

2018-03-15 11:07:15,523 [ 7] INFO [ 3 | 38a45600] CreateEvent(): Created event 7 (qwerqwerqwer)

2018-03-15 11:07:17,567 [ 6] INFO [ 3 | 38a45600] JoinEvent(): Joined event 7

2018-03-15 11:07:30,753 [ 16] INFO [ 3 | 38a45600] DeleteAccount(): Deleted user 3 (qwer) and all of his events

2018-03-15 11:50:24,117 [ 10] INFO [ | ] Register(): Registered user 4 (Pascal Honegger)

2018-03-15 11:52:23,021 [ 19] INFO [ 4 | 486b460b] CreateEvent(): Created event 8 (Badminton)

2018-03-15 11:54:26,309 [ 17] INFO [ 4 | 486b460b] StopActiveSession(): User logged out

2018-03-15 11:54:45,033 [ 30] INFO [ | ] Register(): Registered user 5 (Pascal Honegger der zweite)

2018-03-15 11:55:24,768 [ 28] INFO [ | ] StartSession(userId=5): User logged in

2018-03-15 11:55:52,410 [ 24] INFO [ 5 | 481bc3a6] StopActiveSession(): User logged out

2018-03-15 11:55:58,885 [ 32] INFO [ | ] StartSession(userId=4): User logged in

2018-03-15 11:56:01,533 [ 34] INFO [ 4 | 5859c6d2] JoinEvent(): Joined event 8

2018-03-15 11:56:06,350 [ 32] INFO [ 4 | 5859c6d2] UpdateEventInfo(): Updated event 8

2018-03-15 11:56:11,287 [ 24] INFO [ 5 | f85a70be] JoinEvent(): Joined event 8

2018-03-15 11:57:40,037 [ 34] INFO [ 4 | 5859c6d2] AddAppointments(): Added 1 appointments to event 8

2018-03-15 11:58:19,408 [ 33] INFO [ 4 | 5859c6d2] UpdateEventInfo(): Updated event 8

2018-03-15 12:03:22,815 [ 5] INFO [ 4 | 5859c6d2] UpdateEventInfo(): Updated event 8

2018-03-15 12:46:12,238 [ 73] INFO [ | ] StartSession(userId=4): User logged in

2018-03-15 13:35:33,130 [ 44] INFO [ 4 | c874d5d6] RemoveFromEvent(): The organizer removed the user 5 from the event 8

2018-03-16 09:21:51,390 [ 19] INFO [ | ] StartSession(userId=5): User logged in

2018-03-16 09:22:25,332 [ 7] INFO [ 5 | f5aa3175] CreateEvent(): Created event 9 (To be deleted)

2018-03-16 09:22:46,457 [ 19] INFO [ 5 | f5aa3175] UpdateEventInfo(): Updated event 9

2018-03-16 09:23:21,523 [ 33] INFO [ 4 | c874d5d6] StopActiveSession(): User logged out

2018-03-16 09:23:26,203 [ 26] INFO [ | ] StartSession(userId=4): User logged in

2018-03-16 09:23:33,705 [ 34] INFO [ 4 | c748e1a7] StopActiveSession(): User logged out

2018-03-16 09:32:49,042 [ 71] INFO [ 5 | f5aa3175] AddAppointments(): Added 2 appointments to event 9

2018-03-16 09:33:56,932 [ 83] INFO [ 5 | f5aa3175] RemoveFromEvent(): The organizer removed the user 4 from the event 9

2018-03-16 09:43:06,679 [ 12] INFO [ 4 | c648e1a7] DeleteEvent(): Deleted event 8

2018-03-16 09:43:28,001 [ 10] INFO [ 4 | c648e1a7] DeleteAccount(): Deleted user 4 (Pascal Honegger) and all of his events

2018-03-16 09:45:29,341 [ 6] INFO [ | ] StartSession(userId=5): User logged in

2018-03-16 11:45:50,851 [ 10] INFO [ 5 | f5aa3175] AddAppointments(): Added 1 appointments to event 9

2018-03-16 12:00:45,636 [ 12] INFO [ | ] StartSession(userId=5): User logged in

2018-03-16 12:01:28,331 [ 11] INFO [ 5 | 95520aa8] AddAppointments(): Added 1 appointments to event 9

### All.log

2018-03-16 09:40:34,247 [ 1] INFO HeyImIn.WebApplication.WebApiApplication [ | ] LogStarted(): >>>>>>>>

2018-03-16 09:40:34,293 [ 1] INFO HeyImIn.WebApplication.WebApiApplication [ | ] LogStarted(): >>>>>>>> Started at version 1.0.0.0

2018-03-16 09:40:34,294 [ 1] INFO HeyImIn.WebApplication.WebApiApplication [ | ] LogStarted(): >>>>>>>>

2018-03-16 09:40:57,767 [ 10] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 4 | c648e1a7] OnActionExecuted(): WebApi method /api/OrganizeEvent/DeleteEvent returned, Duration = 0:00:14.0122611

2018-03-16 09:40:57,773 [ 10] FATAL HeyImIn.WebApplication.WebApiComponents.LogFatalExceptionLogger [ | ] LogAsync(): Unhandled exception occured, ex=System.InvalidOperationException: Die Auflistung wurde geändert. Der Enumerationsvorgang kann möglicherweise nicht ausgeführt werden.

bei System.Collections.Generic.HashSet`1.Enumerator.MoveNext()

bei HeyImIn.WebApplication.Services.Impl.DeleteService.DeleteEventLocally(IDatabaseContext context, Event event) in C:\Users\A610222\Source\Repos\HeyImIn\Backend\WebApplication\Services\Impl\DeleteService.cs:Zeile 73.

bei HeyImIn.WebApplication.Controllers.OrganizeEventController.<DeleteEvent>d\_\_1.MoveNext() in C:\Users\A610222\Source\Repos\HeyImIn\Backend\WebApplication\Controllers\OrganizeEventController.cs:Zeile 68.

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Threading.Tasks.TaskHelpersExtensions.<CastToObject>d\_\_1`1.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Controllers.ApiControllerActionInvoker.<InvokeActionAsyncCore>d\_\_1.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Filters.ActionFilterAttribute.<CallOnActionExecutedAsync>d\_\_6.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Web.Http.Filters.ActionFilterAttribute.<CallOnActionExecutedAsync>d\_\_6.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Filters.ActionFilterAttribute.<ExecuteActionFilterAsyncCore>d\_\_5.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Controllers.ActionFilterResult.<ExecuteAsync>d\_\_5.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Filters.AuthorizationFilterAttribute.<ExecuteAuthorizationFilterAsyncCore>d\_\_3.MoveNext()

--- Ende der Stapelüberwachung vom vorhergehenden Ort, an dem die Ausnahme ausgelöst wurde ---

bei System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()

bei System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)

bei System.Web.Http.Dispatcher.HttpControllerDispatcher.<SendAsync>d\_\_15.MoveNext()

2018-03-16 09:43:06,679 [ 12] INFO AuditLogger [ 4 | c648e1a7] DeleteEvent(): Deleted event 8

2018-03-16 09:43:07,864 [ 9] INFO HeyImIn.MailNotifier.Impl.NotificationService [ 4 | c648e1a7] NotifyEventDeletedAsync(): Sent 1 notifications about the deletion of event 8 with the title 'Badminton'

2018-03-16 09:43:07,879 [ 9] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 4 | c648e1a7] OnActionExecuted(): WebApi method /api/OrganizeEvent/DeleteEvent returned, Duration = 0:00:11.7200586

2018-03-16 09:43:08,411 [ 11] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 4 | c648e1a7] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.4541822

2018-03-16 09:43:28,001 [ 10] INFO AuditLogger [ 4 | c648e1a7] DeleteAccount(): Deleted user 4 (Pascal Honegger) and all of his events

2018-03-16 09:43:28,003 [ 10] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 4 | c648e1a7] OnActionExecuted(): WebApi method /api/User/DeleteAccount returned, Duration = 0:00:00.2285162

2018-03-16 09:45:29,337 [ 6] DEBUG HeyImIn.Authentication.Impl.SessionService [ | ] CreateSessionAsync(userId=5, active=True): Added new session

2018-03-16 09:45:29,341 [ 6] INFO AuditLogger [ | ] StartSession(userId=5): User logged in

2018-03-16 09:45:29,343 [ 6] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/StartSession returned, Duration = 0:00:00.2974277

2018-03-16 09:45:29,547 [ 7] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.0966491

2018-03-16 09:45:46,560 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.1003154

2018-03-16 09:47:28,727 [ 21] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0106311

2018-03-16 09:47:29,011 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.1386802

2018-03-16 09:47:33,743 [ 24] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.0981479

2018-03-16 09:47:36,624 [ 28] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.0892906

2018-03-16 09:48:58,151 [ 33] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0084094

2018-03-16 09:48:58,355 [ 32] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5da60c2] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.0956209

2018-03-16 11:43:43,799 [ 12] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:08.9815103

2018-03-16 11:43:44,747 [ 5] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.4772215

2018-03-16 11:44:50,182 [ 12] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0115379

2018-03-16 11:44:50,436 [ 20] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.1027042

2018-03-16 11:45:12,189 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0085179

2018-03-16 11:45:12,482 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.1048991

2018-03-16 11:45:31,115 [ 18] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0091835

2018-03-16 11:45:31,362 [ 11] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetOverview returned, Duration = 0:00:00.0882712

2018-03-16 11:45:39,528 [ 20] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/ParticipateEvent/GetDetails returned, Duration = 0:00:00.3871549

2018-03-16 11:45:41,662 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/OrganizeEvent/GetEditDetails returned, Duration = 0:00:00.0145682

2018-03-16 11:45:50,851 [ 10] INFO AuditLogger [ 5 | f5aa3175] AddAppointments(): Added 1 appointments to event 9

2018-03-16 11:45:50,855 [ 10] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/OrganizeAppointment/AddAppointments returned, Duration = 0:00:00.2129433

2018-03-16 11:45:51,037 [ 21] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/OrganizeEvent/GetEditDetails returned, Duration = 0:00:00.0818199

2018-03-16 11:46:23,399 [ 7] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ | ] OnActionExecuted(): WebApi method /api/Session/GetSession returned, Duration = 0:00:00.0215836

2018-03-16 11:46:23,675 [ 19] DEBUG HeyImIn.WebApplication.WebApiComponents.LogActionAttribute [ 5 | f5aa3175] OnActionExecuted(): WebApi method /api/OrganizeEvent/GetEditDetails returned, Duration = 0:00:00.0244013

2018-03-16 11:46:32,357 [ 7] INFO HeyImIn.MailNotifier.Impl.NotificationService [ 5 | f5aa3175] SendLastMinuteChangeIfRequiredAsync(): Sent 0 updated summaries for appointment 32

## Frontend

### File.html

## Backend – Authentication

### IAuthenticationService.cs

using System.Threading.Tasks;

using HeyImIn.Database.Models;

namespace HeyImIn.Authentication

{

    /// <summary>

    /// Encapsulates user authentication

    /// Automatically updates the user’s password if it <see cref="IPasswordService.NeedsRehash" />

    /// </summary>

    public interface IAuthenticationService

    {

        /// <summary>

        /// Tries to authenticate a user with the given credentials

        /// Automatically rehashes the password if required

        /// </summary>

        /// <param name="email"></param>

        /// <param name="password"></param>

        /// <returns></returns>

        Task<(bool authenticated, User foundUser)> AuthenticateAsync(string email, string password);

    }

}

### IPasswordService.cs

namespace HeyImIn.Authentication

{

    /// <summary>

    /// Encapsulates the password hashing algorithm

    /// </summary>

    public interface IPasswordService

    {

        /// <summary>

        /// Hashes the provided

        /// <param name="password"></param>

        /// </summary>

        /// <param name="password">The password to hash</param>

        /// <returns>Hash of the password</returns>

        string HashPassword(string password);

        /// <summary>

        /// Verifies the provided password

        /// </summary>

        /// <param name="password">Password to validate</param>

        /// <param name="hash">Hash to test against</param>

        /// <returns>True if the password matches the hash</returns>

        bool VerifyPassword(string password, string hash);

        /// <summary>

        /// Tests whether a password needs to be rehashed in order to stay secure (E.g. hashing algorithm changed)

        /// </summary>

        /// <param name="hash">Hash to test</param>

        /// <returns>True if the password needs to be rehashed</returns>

        bool NeedsRehash(string hash);

    }

}

### ISessionService.cs

using System;

using System.Threading.Tasks;

using HeyImIn.Database.Models;

namespace HeyImIn.Authentication

{

    /// <summary>

    /// Encapsulates access to sessions

    /// </summary>

    public interface ISessionService

    {

        /// <summary>

        /// Tries to load a session. If one is found also extends / activates it

        /// </summary>

        /// <param name="token">

        /// <see cref="Session.Token" />

        /// </param>

        /// <returns>Found session or null</returns>

        Task<Session> GetAndExtendSessionAsync(Guid token);

        /// <summary>

        /// Creates a new session for a user

        /// </summary>

        /// <param name="userId">The user the session belongs to</param>

        /// <param name="automaticallyActivate">Wheter the session should be activated automatically</param>

        /// <returns>Token of the created session</returns>

        Task<Guid> CreateSessionAsync(int userId, bool automaticallyActivate);

        /// <summary>

        /// Invalidates a session if it is valid, else does nothing

        /// </summary>

        /// <param name="token">

        /// <see cref="Session.Token" />

        /// </param>

        Task InvalidateSessionAsync(Guid token);

    }

}

### AuthenticationService.cs

using System.Data.Entity;

using System.Reflection;

using System.Threading.Tasks;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using log4net;

namespace HeyImIn.Authentication.Impl

{

    public class AuthenticationService : IAuthenticationService

    {

        private readonly IPasswordService \_passwordService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        public AuthenticationService(IPasswordService passwordService, GetDatabaseContext getDatabaseContext)

        {

            \_passwordService = passwordService;

            \_getDatabaseContext = getDatabaseContext;

        }

        public async Task<(bool authenticated, User foundUser)> AuthenticateAsync(string email, string password)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                User foundUser = await context.Users.FirstOrDefaultAsync(u => u.Email == email);

                if (foundUser == null)

                {

                    \_log.InfoFormat("{0}(): Failed authentication for not existing email {1}", nameof(AuthenticateAsync), email);

                    // Wrong email

                    return (false, null);

                }

                if (!\_passwordService.VerifyPassword(password, foundUser.PasswordHash))

                {

                    \_log.InfoFormat("{0}(): Failed authentication for user {1} because of wrong password", nameof(AuthenticateAsync), foundUser.Id);

                    // Wrong password

                    return (false, foundUser);

                }

                if (\_passwordService.NeedsRehash(foundUser.PasswordHash))

                {

                    \_log.InfoFormat("{0}(): Updating password hash for user {1} to match work factor", nameof(AuthenticateAsync), foundUser.Id);

                    // Update password hash to keep it secure in case the hash factor got updated

                    foundUser.PasswordHash = \_passwordService.HashPassword(password);

                    await context.SaveChangesAsync();

                }

                return (true, foundUser);

            }

        }

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

### PasswordService.cs

namespace HeyImIn.Authentication.Impl

{

    public class PasswordService : IPasswordService

    {

        /// <summary>

        /// Constructor for <see cref="IPasswordService" />

        /// </summary>

        /// <param name="workFactor">Defines how long the hashing for a password takes => Adjust as computing power increases</param>

        public PasswordService(int workFactor)

        {

            \_workFactor = workFactor;

        }

        public string HashPassword(string password)

        {

            return BCrypt.Net.BCrypt.HashPassword(password, \_workFactor);

        }

        public bool VerifyPassword(string password, string hash)

        {

            return BCrypt.Net.BCrypt.Verify(password, hash);

        }

        public bool NeedsRehash(string hash)

        {

            return BCrypt.Net.BCrypt.PasswordNeedsRehash(hash, \_workFactor);

        }

        private readonly int \_workFactor;

    }

}

### SessionService.cs

using System;

using System.Data.Entity;

using System.Reflection;

using System.Threading.Tasks;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using log4net;

namespace HeyImIn.Authentication.Impl

{

    public class SessionService : ISessionService

    {

        public SessionService(GetDatabaseContext getDatabaseContext)

        {

            \_getDatabaseContext = getDatabaseContext;

        }

        public async Task<Session> GetAndExtendSessionAsync(Guid token)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Session session = await context.Sessions.Include(s => s.User).FirstOrDefaultAsync(s => s.Token == token);

                if ((session == null) || !IsValidSession(session))

                {

                    return null;

                }

                ActivateOrExtendSession(session);

                await context.SaveChangesAsync();

                return session;

            }

        }

        public async Task<Guid> CreateSessionAsync(int userId, bool active)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Session session = context.Sessions.Create();

                session.UserId = userId;

                session.Created = DateTime.UtcNow;

                if (active)

                {

                    ActivateOrExtendSession(session);

                }

                context.Sessions.Add(session);

                await context.SaveChangesAsync();

                \_log.DebugFormat("{0}(userId={1}, active={2}): Added new session", nameof(CreateSessionAsync), userId, active);

                return session.Token;

            }

        }

        public async Task InvalidateSessionAsync(Guid token)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Session session = await context.Sessions.FindAsync(token);

                if ((session == null) || !IsValidSession(session))

                {

                    return;

                }

                session.ValidUntil = DateTime.UtcNow;

                await context.SaveChangesAsync();

                \_log.DebugFormat("{0}(token={1}): Invalidated session", nameof(InvalidateSessionAsync), token);

            }

        }

        private static void ActivateOrExtendSession(Session session)

        {

            // Set new valid until date as the session has been used

            session.ValidUntil = DateTime.UtcNow + \_inactiveSessionTimeout;

        }

        private static bool IsValidSession(Session session)

        {

            return session.ValidUntil == null

                ? DateTime.UtcNow - session.Created <= TimeSpan.FromDays(2)

                : session.ValidUntil >= DateTime.UtcNow;

        }

        private readonly GetDatabaseContext \_getDatabaseContext;

        /// <summary>

        /// A session gets invalidated after this time period passed without any access to the session

        /// </summary>

        private static readonly TimeSpan \_inactiveSessionTimeout = TimeSpan.FromHours(24);

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

## Backend – Database

### IDatabaseContext.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Data.Entity.Validation;

using System.Threading.Tasks;

using HeyImIn.Database.Models;

namespace HeyImIn.Database.Context

{

    /// <summary>

    /// A Delegate to prevent the copy-paste of <see cref="Func{TResult}" /> of <see cref="IDatabaseContext" />

    /// </summary>

    public delegate IDatabaseContext GetDatabaseContext();

    public interface IDatabaseContext : IDisposable

    {

        // DbContext methods we want to provide

        Task<int> SaveChangesAsync();

        DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;

        IEnumerable<DbEntityValidationResult> GetValidationErrors();

        // Main tables

        DbSet<User> Users { get; }

        DbSet<Appointment> Appointments { get; }

        DbSet<Event> Events { get; }

        DbSet<Session> Sessions { get; }

        // Many-To-Many relation tables

        DbSet<AppointmentParticipation> AppointmentParticipations { get; }

        DbSet<EventParticipation> EventParticipations { get; }

        // Token tables

        DbSet<PasswordReset> PasswordResets { get; }

        DbSet<EventInvitation> EventInvitations { get; }

    }

}

### DatabaseConfiguration.cs

using System.Data.Entity;

using HeyImIn.Database.Context.Impl;

using HeyImIn.Database.Migrations;

namespace HeyImIn.Database.Context

{

    public static class DatabaseConfiguration

    {

        /// <summary>

        /// Enables automatic migrations

        /// </summary>

        public static void ConfigureMigrations()

        {

            System.Data.Entity.Database.SetInitializer(new MigrateDatabaseToLatestVersion<HeyImInDatabaseContext, Configuration>(

                true /\* As we inject the connection string using DI, this is required \*/));

        }

    }

}

### HeyImInDatabaseContext.cs

using System.Data.Entity;

using System.Data.Entity.ModelConfiguration.Conventions;

using HeyImIn.Database.Models;

// ReSharper disable UnusedAutoPropertyAccessor.Global

namespace HeyImIn.Database.Context.Impl

{

    // ReSharper disable once ClassWithVirtualMembersNeverInherited.Global => Entity Framework

    public class HeyImInDatabaseContext : DbContext, IDatabaseContext

    {

        public HeyImInDatabaseContext(string connectionString) : base(connectionString)

        {

        }

        // Main tables

        public virtual DbSet<User> Users { get; set; }

        public virtual DbSet<Appointment> Appointments { get; set; }

        public virtual DbSet<Event> Events { get; set; }

        public virtual DbSet<Session> Sessions { get; set; }

        // Many-To-Many relation tables

        public virtual DbSet<AppointmentParticipation> AppointmentParticipations { get; set; }

        public virtual DbSet<EventParticipation> EventParticipations { get; set; }

        // Token tables

        public virtual DbSet<PasswordReset> PasswordResets { get; set; }

        public virtual DbSet<EventInvitation> EventInvitations { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)

        {

            // We manually remove / update relations before deleting the referenced entry

            modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

            modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();

        }

    }}

### Configuration.cs

using System.Data.Entity.Migrations;

using HeyImIn.Database.Context.Impl;

namespace HeyImIn.Database.Migrations

{

    internal sealed class Configuration : DbMigrationsConfiguration<HeyImInDatabaseContext>

    {

        public Configuration()

        {

            AutomaticMigrationsEnabled = true;

        }

        protected override void Seed(HeyImInDatabaseContext context)

        {

            // This method will be called after migrating to the latest version.

        }

    }

}

### Appointment.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class Appointment

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int Id { get; set; }

        public virtual Event Event { get; set; }

        [Required]

        [ForeignKey(nameof(Event))]

        public int EventId { get; set; }

        /// <summary>

        /// At what time the appointment will take place

        /// </summary>

        [Required]

        public DateTime StartTime { get; set; }

        public virtual ICollection<AppointmentParticipation> AppointmentParticipations { get; set; }

    }

}

### AppointmentParticipation.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class AppointmentParticipation

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int Id { get; set; }

        public virtual Appointment Appointment { get; set; }

        [Required]

        [ForeignKey(nameof(Appointment))]

        [Index("UniqueAppointmentParticipation", IsUnique = true, Order = 1)]

        public int AppointmentId { get; set; }

        public virtual User Participant { get; set; }

        [Required]

        [ForeignKey(nameof(Participant))]

        [Index("UniqueAppointmentParticipation", IsUnique = true, Order = 2)]

        public int ParticipantId { get; set; }

        [Required]

        public AppointmentParticipationAnswer AppointmentParticipationAnswer { get; set; }

        /// <summary>

        /// Wheter a reminder has already been sent

        /// </summary>

        public bool SentReminder { get; set; }

        /// <summary>

        /// Wheter a summary has already been sent

        /// If this is true and any changes occure, an updated summary is sent

        /// </summary>

        public bool SentSummary { get; set; }

    }

}

### AppointmentParticipationAnswer.cs

namespace HeyImIn.Database.Models

{

    public enum AppointmentParticipationAnswer

    {

        Accepted = 0,

        Declined = 1

    }

}

### Event.cs

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class Event

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int Id { get; set; }

        /// <summary>

        /// The <see cref="User" /> which organizes this event

        /// Can manage <see cref="Appointments" />, general information and <see cref="EventParticipations" />

        /// </summary>

        public virtual User Organizer { get; set; }

        [Required]

        [ForeignKey(nameof(Organizer))]

        public int OrganizerId { get; set; }

        /// <summary>

        /// The title of the event

        /// </summary>

        [Required]

        [MaxLength(FieldLengths.TitleMaxLength)]

        public string Title { get; set; }

        /// <summary>

        /// The place to meet at

        /// </summary>

        [Required]

        [MaxLength(FieldLengths.MeetingPlaceMaxLength)]

        public string MeetingPlace { get; set; }

        /// <summary>

        /// A short description on what to expect from this event

        /// </summary>

        [Required]

        [MaxLength(FieldLengths.DescriptionMaxLength)]

        public string Description { get; set; }

        /// <summary>

        /// Wheter the event is private or public. Private events require an invitation

        /// </summary>

        public bool IsPrivate { get; set; }

        /// <summary>

        /// The number of hours before an <see cref="Appointment" /> where the <see cref="EventParticipations" /> get a

        /// reminder email

        /// </summary>

        [Required]

        [Range(FieldLengths.RealisticMinimumHours, FieldLengths.RealisticMaximumHours)]

        public int ReminderTimeWindowInHours { get; set; }

        /// <summary>

        /// The number of hours before an <see cref="Appointment" /> where the <see cref="EventParticipations" /> get a summary

        /// email

        /// Every change that happens after a summary mail was sent will trigger an updated summary to be sent

        /// </summary>

        [Required]

        [Range(FieldLengths.RealisticMinimumHours, FieldLengths.RealisticMaximumHours)]

        public int SummaryTimeWindowInHours { get; set; }

        public virtual ICollection<Appointment> Appointments { get; set; }

        public virtual ICollection<EventParticipation> EventParticipations { get; set; }

        public virtual ICollection<EventInvitation> EventInvitations { get; set; }

    }

}

### EventInvitation.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class EventInvitation

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public Guid Token { get; set; }

        public virtual Event Event { get; set; }

        [Required]

        [ForeignKey(nameof(Event))]

        public int EventId { get; set; }

        /// <summary>

        /// At what time the invitation was sent

        /// The invitation will expire after some period of time

        /// </summary>

        [Required]

        public DateTime Requested { get; set; }

        /// <summary>

        /// Wheter this invite was used

        /// </summary>

        public bool Used { get; set; }

    }

}

### EventParticipation.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class EventParticipation

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int Id { get; set; }

        public virtual Event Event { get; set; }

        [Required]

        [ForeignKey(nameof(Event))]

        [Index("UniqueEventParticipation", IsUnique = true, Order = 1)]

        public int EventId { get; set; }

        public virtual User Participant { get; set; }

        [Required]

        [ForeignKey(nameof(Participant))]

        [Index("UniqueEventParticipation", IsUnique = true, Order = 2)]

        public int ParticipantId { get; set; }

        /// <summary>

        /// Wheter or not the users wants to receive an email to remind him of his participation

        /// </summary>

        public bool SendReminderEmail { get; set; } = true;

        /// <summary>

        /// Wheter or not the users wants to receive an email with the final event information

        /// </summary>

        public bool SendSummaryEmail { get; set; } = true;

        /// <summary>

        /// Wheter or not the users wants to receive an email to inform him about last-minute changes (changes made after the

        /// summary was sent)

        /// </summary>

        public bool SendLastMinuteChangesEmail { get; set; } = true;

    }

}

### FieldLengths.cs

namespace HeyImIn.Database.Models

{

    /// <summary>

    /// Field lenghts used by the database

    /// These values should be used in the backend & frontend validation

    /// </summary>

    public static class FieldLengths

    {

        public const int UserFullNameMaxLength = 40;

        public const int UserEmailMaxLength = 40;

        public const int UserPasswordHashMaxLength = 60;

        public const int TitleMaxLength = 40;

        public const int MeetingPlaceMaxLength = 40;

        public const int DescriptionMaxLength = 120;

        public const int RealisticMinimumHours = 1;

        public const int RealisticMaximumHours = 720; // 30 days

    }

}

### PasswordReset.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class PasswordReset

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public Guid Token { get; set; }

        public virtual User User { get; set; }

        [Required]

        [ForeignKey(nameof(User))]

        public int UserId { get; set; }

        /// <summary>

        /// At what time the password reset was requested

        /// If not used for a period of time it is no longer valid

        /// </summary>

        [Required]

        public DateTime Requested { get; set; }

        /// <summary>

        /// Wheter this password reset was used

        /// </summary>

        public bool Used { get; set; }

    }

}

### Session.cs

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class Session

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public Guid Token { get; set; }

        public virtual User User { get; set; }

        [Required]

        [ForeignKey(nameof(User))]

        public int UserId { get; set; }

        /// <summary>

        /// At what point the (active) session will be invalidated

        /// </summary>

        public DateTime? ValidUntil { get; set; }

        /// <summary>

        /// At what time the session was created

        /// If a session isn't activated within a certain period of time (<see cref="ValidUntil" /> == null) it will expire

        /// </summary>

        [Required]

        public DateTime Created { get; set; }

    }

}

### User.cs

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace HeyImIn.Database.Models

{

    public class User

    {

        [Key]

        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int Id { get; set; }

        /// <summary>

        /// The full name of the user, which is used to refer to him

        /// </summary>

        [Required]

        [MaxLength(FieldLengths.UserFullNameMaxLength)]

        public string FullName { get; set; }

        /// <summary>

        /// The email of the users, which is used for notifications and <see cref="PasswordReset" />

        /// </summary>

        [Required]

        [EmailAddress]

        [MaxLength(FieldLengths.UserEmailMaxLength)]

        [Index(IsUnique = true)]

        public string Email { get; set; }

        /// <summary>

        /// The secure password hash

        /// </summary>

        [Required]

        [MaxLength(FieldLengths.UserPasswordHashMaxLength)]

        public string PasswordHash { get; set; }

        public virtual ICollection<Session> Sessions { get; set; }

        public virtual ICollection<PasswordReset> PasswordResets { get; set; }

        public virtual ICollection<Event> OrganizedEvents { get; set; }

        public virtual ICollection<EventParticipation> EventParticipations { get; set; }

        public virtual ICollection<AppointmentParticipation> AppointmentParticipations { get; set; }

    }

}

## Backend – External (Aus anderen Projekten übernommen)

### AssemblyScanner.cs

using System;

using System.Linq;

using System.Reflection;

using Autofac;

namespace HeyImIn.External.DependencyInjection

{

    /// <summary>

    /// A simple assembly scanner taken from the MediaGateway project

    /// </summary>

    public static class AssemblyScanner

    {

        public static void Scan(ContainerBuilder builder, params Assembly[] assemblies)

        {

            // register non-singleton types with name matching interfaces

            builder.RegisterAssemblyTypes(assemblies)

                .Where(t => (GetNameMatchingInterfaces(t).Length == 1) && !HasSingletonAttribute(t) && !HasDoNotRegisterAttribute(t) && !IsExcludedName(t))

                .As(t => GetNameMatchingInterfaces(t))

                .PreserveExistingDefaults();

            // register singleton types with name matching interfaces

            builder.RegisterAssemblyTypes(assemblies)

                .Where(t => (GetNameMatchingInterfaces(t).Length == 1) && HasSingletonAttribute(t) && !HasDoNotRegisterAttribute(t) && !IsExcludedName(t))

                .As(t => GetNameMatchingInterfaces(t))

                .SingleInstance()

                .PreserveExistingDefaults();

            // register non-singleton types marked the type with the RegisterAsSelf attribute

            builder.RegisterAssemblyTypes(assemblies)

                .Where(t => HasRegisterAsSelfAttribute(t) && !HasSingletonAttribute(t) && !HasDoNotRegisterAttribute(t) && !IsExcludedName(t))

                .AsSelf()

                .PreserveExistingDefaults();

            // register singleton types marked the type with the RegisterAsSelf attribute

            builder.RegisterAssemblyTypes(assemblies)

                .Where(t => HasRegisterAsSelfAttribute(t) && HasSingletonAttribute(t) && !HasDoNotRegisterAttribute(t) && !IsExcludedName(t))

                .AsSelf()

                .SingleInstance()

                .PreserveExistingDefaults();

            // register modules, allow modules to do more complex registrations themselves

            builder.RegisterAssemblyModules(assemblies); // -> calls Module.Load() of all detected modules

        }

        private static bool HasSingletonAttribute(Type type)

        {

            return type.GetCustomAttribute<SingletonAttribute>() != null;

        }

        private static bool HasDoNotRegisterAttribute(Type type)

        {

            return type.GetCustomAttribute<DoNotRegisterAttribute>() != null;

        }

        private static bool HasRegisterAsSelfAttribute(Type type)

        {

            return type.GetCustomAttribute<RegisterAsSelfAttribute>() != null;

        }

        private static Type[] GetNameMatchingInterfaces(Type type)

        {

            return type.GetInterfaces().Where(i => type.Name == i.Name.Substring(1)).ToArray();

        }

        private static bool IsExcludedName(Type type)

        {

            return type.Name.EndsWith("Attribute", StringComparison.Ordinal) || type.Name.EndsWith("Module", StringComparison.Ordinal);

        }

    }

}

### DoNotRegisterAttribute.cs

using System;

namespace HeyImIn.External.DependencyInjection

{

    /// <summary>

    /// Use attribute to mark types that shall not be registered with the DI container

    /// </summary>

    [AttributeUsage(AttributeTargets.Class)]

    public sealed class DoNotRegisterAttribute : Attribute

    {

    }

}

### RegisterAsSelfAttribute.cs

using System;

namespace HeyImIn.External.DependencyInjection

{

    /// <summary>

    /// Use attribute to mark types that shall be registered as self with the DI container

    /// </summary>

    [AttributeUsage(AttributeTargets.Class)]

    public sealed class RegisterAsSelfAttribute : Attribute

    {

    }

}

### SingletonAttribute.cs

using System;

namespace HeyImIn.External.DependencyInjection

{

    /// <summary>

    /// Use attribute to mark types that shall be registered as singletons with the DI container

    /// </summary>

    [AttributeUsage(AttributeTargets.Class)]

    public sealed class SingletonAttribute : Attribute

    {

    }

}

## Backend – MailNotifier

### IMailSender.cs

using System.Collections.Generic;

using System.Threading.Tasks;

namespace HeyImIn.MailNotifier

{

    /// <summary>

    /// Abstraction of the SendGrid client which also contains the HeyImIn From-Email

    /// </summary>

    public interface IMailSender

    {

        /// <summary>

        /// Sends a mail to a single recipient

        /// Calls <see cref="SendMailAsync(IReadOnlyCollection{string},string,string)" /> with an array containing one email

        /// </summary>

        /// <param name="recipientEmail">Email to send to</param>

        /// <param name="subject">Plain text subject</param>

        /// <param name="bodyText">Plain text body</param>

        Task SendMailAsync(string recipientEmail, string subject, string bodyText);

        /// <summary>

        /// Sends a mail to multiple recipient

        /// </summary>

        /// <param name="recipientEmails">Emails to send to</param>

        /// <param name="subject">Plain text subject</param>

        /// <param name="bodyText">Plain text body</param>

        Task SendMailAsync(IReadOnlyCollection<string> recipientEmails, string subject, string bodyText);

    }

}

### INotificationService.cs

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier.Models;

namespace HeyImIn.MailNotifier

{

    /// <summary>

    /// A service which encapsulates the sending of an email

    /// Is responsible for the formatting and wording of the email

    /// </summary>

    public interface INotificationService

    {

        /// <summary>

        /// Sends the provided token to the user, informing him about his password reset request

        /// </summary>

        /// <param name="token">Reset code to send</param>

        /// <param name="recipient">User who should receive the code</param>

        Task SendPasswordResetTokenAsync(Guid token, User recipient);

        /// <summary>

        /// Sends an invite link for each provided invitation

        /// </summary>

        /// <param name="userInvitations">Existing users</param>

        /// <param name="newInvitations">New users</param>

        Task SendInvitationLinkAsync(List<(User user, EventInvitation invite)> userInvitations, List<(string email, EventInvitation invite)> newInvitations);

        /// <summary>

        /// Sends a notifications after the organizer of an event changed some data for a user

        /// E.g. organizer removes a user

        /// </summary>

        /// <param name="event">Affected event, a link to this event will be included</param>

        /// <param name="affectedUser">The user who got updated by the organizer</param>

        /// <param name="change">The change that happened (E.g. "Organizer removed you from event")</param>

        Task NotifyOrganizerUpdatedUserInfoAsync(Event @event, User affectedUser, string change);

        /// <summary>

        /// Sends a reminder to all users who haven't received one yet

        /// Updates the appointment object -> Execute save() after this method was called

        /// </summary>

        /// <param name="appointment">Only users who accepted this appointment will get reminders</param>

        Task SendAndUpdateRemindersAsync(Appointment appointment);

        /// <summary>

        /// Sends a summary to all users who haven't recieved one yet

        /// Updates the appointment object -> Execute save() after this method was called

        /// </summary>

        /// <param name="appointment">Appointment for which to send summaries</param>

        Task SendAndUpdateSummariesAsync(Appointment appointment);

        /// <summary>

        /// Sends an updated summary to all users who have already received a summary

        /// </summary>

        /// <param name="appointment">Only users who accepted this appointment will get updated summaries</param>

        Task SendLastMinuteChangeIfRequiredAsync(Appointment appointment);

        /// <summary>

        /// Sends a notification that a specific appointment got canceled

        /// Only informs users who have accepted the appointment

        /// Is NOT called when a whole event gets deleted, as every user gets informed about that by

        /// <see cref="NotifyEventDeletedAsync" />

        /// </summary>

        /// <param name="appointment">The now deleted appointment</param>

        /// <param name="event">Event the appointment was part of</param>

        Task NotifyAppointmentExplicitlyCanceledAsync(AppointmentNotificationInformation appointment, Event @event);

        /// <summary>

        /// Sends a notification to all users who are participating in an event

        /// This happens either because the organizer deleted himself or a single event explicitly

        /// </summary>

        /// <param name="event">Event which got deleted</param>

        Task NotifyEventDeletedAsync(EventNotificationInformation @event);

        /// <summary>

        /// Sends a notification to all users who are participating in an event

        /// This happens because the organizer changed some general information about the event

        /// E.g. changed the description or meeting place

        /// </summary>

        /// <param name="event">Event which got updated</param>

        Task NotifyEventUpdatedAsync(Event @event);

    }

}

### MailSender.cs

using System.Collections.Generic;

using System.Net.Http;

using System.Reflection;

using System.Threading.Tasks;

using log4net;

using SendGrid;

using SendGrid.Helpers.Mail;

namespace HeyImIn.MailNotifier.Impl

{

    public class MailSender : IMailSender

    {

        public MailSender(ISendGridClient sendGridClient)

        {

            \_sendGridClient = sendGridClient;

        }

        public Task SendMailAsync(string recipientEmail, string subject, string bodyText)

        {

            return SendMailAsync(new[] { recipientEmail }, subject, bodyText);

        }

        public async Task SendMailAsync(IReadOnlyCollection<string> recipientEmails, string subject, string bodyText)

        {

            var message = new SendGridMessage();

            message.SetFrom("no-reply@hey-im-in.ch", "Hey, I'm in");

            foreach (string recipient in recipientEmails)

            {

                message.AddTo(recipient);

            }

            message.SetSubject(subject);

            // If desired we could create a HTML based layout

            message.PlainTextContent = bodyText;

            Response sendGridResponse = await \_sendGridClient.SendEmailAsync(message);

            // See https://stackoverflow.com/a/44634349

            if (!new HttpResponseMessage(sendGridResponse.StatusCode).IsSuccessStatusCode)

            {

                string responseBody = await sendGridResponse.Body.ReadAsStringAsync();

                \_log.ErrorFormat("{0}(subject={1}): Failed to send mail to recipients ({2}), statusCode={3}, errorBody={4}", nameof(SendMailAsync), subject, string.Join(";", recipientEmails), sendGridResponse.StatusCode, responseBody);

            }

        }

        private readonly ISendGridClient \_sendGridClient;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

### NotificationService.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using HeyImIn.Authentication;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier.Models;

using log4net;

namespace HeyImIn.MailNotifier.Impl

{

    public class NotificationService : INotificationService

    {

        public NotificationService(IMailSender mailSender, ISessionService sessionService, string baseWebUrl, string mailTimeZoneName)

        {

            \_mailSender = mailSender;

            \_sessionService = sessionService;

            \_baseWebUrl = baseWebUrl;

            try

            {

                \_mailTimeZone = TimeZoneInfo.FindSystemTimeZoneById(mailTimeZoneName);

            }

            catch (Exception e)

            {

                \_log.ErrorFormat("{0}(): Configured time zone '{1}' was not valid, error={2}", nameof(NotificationService), mailTimeZoneName, e);

                \_mailTimeZone = TimeZoneInfo.Utc;

            }

        }

        #region No data deleted

        public async Task SendPasswordResetTokenAsync(Guid token, User recipient)

        {

            const string PasswordResetSubject = "Passwort zurücksetzen";

            string message = $@"Hallo {recipient.FullName}

Soeben wurde eine Zurücksetzung des Passworts für diese E-Mail-Adresse beantragt. Der Code dafür lautet:

{token}

Sie können diesen Code unter {\_baseWebUrl}ResetPassword eingeben und Ihr Passwort zurücksetzen.";

            await \_mailSender.SendMailAsync(recipient.Email, PasswordResetSubject, message);

            \_log.InfoFormat("{0}(userId={1}, userEmail={2}): Sent password reset token", nameof(SendPasswordResetTokenAsync), recipient.Id, recipient.Email);

        }

        public async Task SendInvitationLinkAsync(List<(User user, EventInvitation invite)> userInvitations, List<(string email, EventInvitation invite)> newInvitations)

        {

            const string EventInviteSubject = "Sie wurden zu einem Event eingeladen";

            foreach ((User user, EventInvitation invite) in userInvitations)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(user.Id);

                string personalizedMessage = $@"Hallo {user.FullName}

Sie wurden von {invite.Event.Organizer.FullName} zum Event '{invite.Event.Title}' eingeladen.

Sie können diese Einladung unter {\_baseWebUrl}AcceptInvitation/{invite.Token}{authTokenSuffix} annehmen.";

                await \_mailSender.SendMailAsync(user.Email, EventInviteSubject, personalizedMessage);

            }

            foreach ((string email, EventInvitation invite) in newInvitations)

            {

                string generalizedMessage = $@"Hallo

Sie wurden von {invite.Event.Organizer.FullName} zum Event '{invite.Event.Title}' eingeladen.

Sie können diese Einladung unter {\_baseWebUrl}AcceptInvitation/{invite.Token} annehmen.";

                await \_mailSender.SendMailAsync(email, EventInviteSubject, generalizedMessage);

            }

            \_log.InfoFormat("{0}(): Sent {1} invites", nameof(SendInvitationLinkAsync), userInvitations.Count + newInvitations.Count);

        }

        public async Task NotifyOrganizerUpdatedUserInfoAsync(Event @event, User affectedUser, string change)

        {

            const string OrganizerUpdatedUserSubject = "Ein Organisator hat Änderungen für Sie vorgenommen";

            string authTokenSuffix = await CreateAuthTokenSuffixAsync(affectedUser.Id);

            string message = $@"Hallo {affectedUser.FullName}

{@event.Organizer.FullName}, der Organisator des Events '{@event.Title}', hat folgende Änderungen für Sie vorgenommen:

{change}

Sie können den betroffenen Event unter {\_baseWebUrl}ViewEvent/{@event.Id}{authTokenSuffix} ansehen.";

            await \_mailSender.SendMailAsync(affectedUser.Email, OrganizerUpdatedUserSubject, message);

            \_log.InfoFormat("{0}(): Sent notification as a organizer of event {1} changed something about the user {2}", nameof(NotifyOrganizerUpdatedUserInfoAsync), @event.Id, affectedUser.Id);

        }

        public async Task SendAndUpdateRemindersAsync(Appointment appointment)

        {

            DateTime startTime = TargetTimeZone(appointment.StartTime);

            string reminderSubject = $"Reminder Event '{appointment.Event.Title}' zum Termin am {startTime:g}";

            List<AppointmentParticipation> participationsAwaitingReminder = FilterParticipations(appointment.AppointmentParticipations, p => !p.SentReminder, e => e.SendReminderEmail);

            foreach (AppointmentParticipation participation in participationsAwaitingReminder)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(participation.Participant.Id);

                string message = $@"Hallo {participation.Participant.FullName}

Sie haben dem Termin am {startTime:g} zum Event '{appointment.Event.Title}' zugesagt.

Sie können weitere Details zum Event unter {\_baseWebUrl}ViewEvent/{appointment.Event.Id}{authTokenSuffix} ansehen.";

                await \_mailSender.SendMailAsync(participation.Participant.Email, reminderSubject, message);

                participation.SentReminder = true;

            }

            \_log.InfoFormat("{0}(): Sent {1} reminders for appointment {2}", nameof(SendAndUpdateRemindersAsync), participationsAwaitingReminder.Count, appointment.Id);

        }

        public async Task SendAndUpdateSummariesAsync(Appointment appointment)

        {

            DateTime startTime = TargetTimeZone(appointment.StartTime);

            string summarySubject = $"Zusammenfassung Event '{appointment.Event.Title}' zum Termin am {startTime:g}";

            string participants = ParticipationsList(appointment.AppointmentParticipations);

            string messageBody = $@"Die Teilnehmer treffen sich am {startTime:g} am Treffpunkt '{appointment.Event.MeetingPlace}'

Die finale Teilnehmerliste:

{participants}";

            List<AppointmentParticipation> participationsAwaitingSummary = FilterParticipations(appointment.AppointmentParticipations, p => !p.SentSummary, e => e.SendSummaryEmail);

            foreach (AppointmentParticipation participation in participationsAwaitingSummary)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(participation.Participant.Id);

                string message = $@"Hallo {participation.Participant.FullName}

{messageBody}

Sie können weitere Details zum Event unter {\_baseWebUrl}ViewEvent/{appointment.Event.Id}{authTokenSuffix} ansehen.";

                await \_mailSender.SendMailAsync(participation.Participant.Email, summarySubject, message);

                participation.SentSummary = true;

            }

            \_log.InfoFormat("{0}(): Sent {1} summaries for appointment {2}", nameof(SendAndUpdateSummariesAsync), participationsAwaitingSummary.Count, appointment.Id);

        }

        public async Task NotifyEventUpdatedAsync(Event @event)

        {

            string updatedSubject = $"Der Event '{@event.Title}' wurde editiert";

            string messageBody = $"Die Informationen zum Event '{@event.Title}', an welchem Sie teilnehmen, wurden aktualisiert.";

            foreach (EventParticipation participation in @event.EventParticipations)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(participation.Participant.Id);

                string message = $@"Hallo {participation.Participant.FullName}

{messageBody}

Sie können weitere Details zum Event unter {\_baseWebUrl}ViewEvent/{@event.Id}{authTokenSuffix} ansehen.";

                await \_mailSender.SendMailAsync(participation.Participant.Email, updatedSubject, message);

            }

            \_log.InfoFormat("{0}(): Sent {1} notifications about a change of the event {2}", nameof(NotifyEventUpdatedAsync), @event.EventParticipations.Count, @event.Id);

        }

        public async Task SendLastMinuteChangeIfRequiredAsync(Appointment appointment)

        {

            if (appointment.StartTime <= DateTime.UtcNow)

            {

                return;

            }

            DateTime startTime = TargetTimeZone(appointment.StartTime);

            string lastMinuteChangeSubject = $"Kurzfristige Änderung zum Termin am {startTime:g}";

            string participants = ParticipationsList(appointment.AppointmentParticipations);

            string messageBody = $@"Die Teilnehmer des Events '{appointment.Event.Title}' haben sich geändert. Die aktualisierte Teilnehmerliste:

{participants}";

            List<AppointmentParticipation> participationsAlreadyGotSummary = FilterParticipations(appointment.AppointmentParticipations, p => p.SentSummary, e => e.SendLastMinuteChangesEmail);

            foreach (AppointmentParticipation participation in participationsAlreadyGotSummary)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(participation.Participant.Id);

                string message = $@"Hallo {participation.Participant.FullName}

{messageBody}

Sie können weitere Details zum Event unter {\_baseWebUrl}ViewEvent/{appointment.Event.Id}{authTokenSuffix} ansehen.";

                await \_mailSender.SendMailAsync(participation.Participant.Email, lastMinuteChangeSubject, message);

            }

            \_log.InfoFormat("{0}(): Sent {1} updated summaries for appointment {2}", nameof(SendLastMinuteChangeIfRequiredAsync), participationsAlreadyGotSummary.Count, appointment.Id);

        }

        #endregion

        #region Data possibly deleted

        public async Task NotifyAppointmentExplicitlyCanceledAsync(AppointmentNotificationInformation appointment, Event @event)

        {

            DateTime startTime = TargetTimeZone(appointment.StartTime);

            string summarySubject = $"Termin am {startTime:g} zum Event '{@event.Title}' abgesagt";

            string messageBody = $"Sie haben dem Termin ({startTime:g}) zugesagt, welcher vom Organisator abgesagt wurde.";

            List<AppointmentParticipationNotificationInformation> acceptedParticipations = appointment.Participations.Where(p => p.Answer == AppointmentParticipationAnswer.Accepted).ToList();

            foreach (AppointmentParticipationNotificationInformation participation in acceptedParticipations)

            {

                string authTokenSuffix = await CreateAuthTokenSuffixAsync(participation.Participant.Id);

                string message = $@"Hallo {participation.Participant.FullName}

{messageBody}

Sie können weitere Details zum Event unter {\_baseWebUrl}ViewEvent/{@event.Id}{authTokenSuffix} ansehen.";

                await \_mailSender.SendMailAsync(participation.Participant.Email, summarySubject, message);

            }

            \_log.InfoFormat("{0}(): Sent {1} notifications about cancelation of appointment {2}", nameof(NotifyAppointmentExplicitlyCanceledAsync), acceptedParticipations.Count, appointment.Id);

        }

        public async Task NotifyEventDeletedAsync(EventNotificationInformation @event)

        {

            string deletedSubject = $"Der Event '{@event.Title}' wurde gelöscht";

            string messageBody = $"Der Event '{@event.Title}', an welchem Sie teilgenommen haben, wurde zusammen mit allen Terminen gelöscht.";

            foreach (UserNotificationInformation participant in @event.Participations)

            {

                string message = $@"Hallo {participant.FullName}

{messageBody}";

                await \_mailSender.SendMailAsync(participant.Email, deletedSubject, message);

            }

            \_log.InfoFormat("{0}(): Sent {1} notifications about the deletion of event {2} with the title '{3}'", nameof(NotifyEventDeletedAsync), @event.Participations.Count, @event.Id, @event.Title);

        }

        #endregion

        /// <summary>

        /// Returns all participations which pass the provided filters and were accepted

        /// </summary>

        private static List<AppointmentParticipation> FilterParticipations(IEnumerable<AppointmentParticipation> participations,

                                                                    Func<AppointmentParticipation, bool> participationFilter,

                                                                    Func<EventParticipation, bool> eventFilter)

        {

            return participations

                .Where(p => p.AppointmentParticipationAnswer == AppointmentParticipationAnswer.Accepted)

                .Where(participationFilter)

                .Where(p => eventFilter(p.Appointment.Event.EventParticipations.First(e => e.Participant == p.Participant)))

                .ToList();

        }

        private async Task<string> CreateAuthTokenSuffixAsync(int userId)

        {

            Guid createdSessionToken = await \_sessionService.CreateSessionAsync(userId, false);

            return $"?authToken={createdSessionToken}";

        }

        private DateTime TargetTimeZone(DateTime utcDateTime)

        {

            return TimeZoneInfo.ConvertTimeFromUtc(utcDateTime, \_mailTimeZone);

        }

        private static string ParticipationsList(IEnumerable<AppointmentParticipation> allParticipations)

        {

            return string.Join(Environment.NewLine, allParticipations

                .Where(a => a.AppointmentParticipationAnswer == AppointmentParticipationAnswer.Accepted)

                .Select(a => $"- {a.Participant.FullName}"));

        }

        private readonly IMailSender \_mailSender;

        /// <summary>

        /// The url being referd to within emails

        /// </summary>

        private readonly string \_baseWebUrl;

        private readonly ISessionService \_sessionService;

        private readonly TimeZoneInfo \_mailTimeZone;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

## Backend – WebApplication

### AssemblyInfo.cs

using System.Reflection;

[assembly: AssemblyTitle("Hey, I'm in")]

[assembly: log4net.Config.XmlConfigurator(ConfigFile = "log4net.config", Watch = true)]

### Global.asax.cs

using System;

using System.Diagnostics;

using System.IO;

using System.Reflection;

using System.Web;

using System.Web.Http;

using System.Web.Http.ExceptionHandling;

using HeyImIn.Database.Context;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

using log4net.Util;

namespace HeyImIn.WebApplication

{

    public class WebApiApplication : HttpApplication

    {

        protected void Application\_Start()

        {

            ConfigureLog4Net();

            LogStarted();

            DatabaseConfiguration.ConfigureMigrations();

            GlobalConfiguration.Configure(WebApiConfig.Register);

            GlobalConfiguration.Configuration.Services.Add(typeof(IExceptionLogger), new LogFatalExceptionLogger());

        }

        protected void Application\_End()

        {

            LogStopped();

        }

        private static void ConfigureLog4Net()

        {

            // The mapping to the log4net.conf file is done through the AssemblyInfo.cs

            string logFileDirectory =

#if DEBUG

                Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "App\_Data");

#else

                "D:\\home\\LogFiles\\HeyImIn";

#endif

            if (!Directory.Exists(logFileDirectory))

            {

                // Make sure the directory exists

                Directory.CreateDirectory(logFileDirectory);

            }

            GlobalContext.Properties[LogHelpers.LogFileDirectoryKey] = logFileDirectory;

            // Hide (null) from logs => https://stackoverflow.com/a/22344774

            SystemInfo.NullText = string.Empty;

            \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        }

        private static void LogStarted()

        {

            Version currentVersion = typeof(WebApiApplication).Assembly.GetName().Version;

            \_log.InfoFormat("{0}(): {1}", nameof(LogStarted), StartEndPrefix);

            \_log.InfoFormat("{0}(): {1} Started at version {2}", nameof(LogStarted), StartEndPrefix, currentVersion);

            \_log.InfoFormat("{0}(): {1}", nameof(LogStarted), StartEndPrefix);

        }

        private static void LogStopped()

        {

            if (\_log == null)

            {

                Trace.TraceError("No \_log configured -> Startup probably failed");

                return;

            }

            \_log.InfoFormat("{0}(): {1}", nameof(LogStarted), StartEndPrefix);

            \_log.InfoFormat("{0}(): {1} Stopped", nameof(LogStarted), StartEndPrefix);

            \_log.InfoFormat("{0}(): {1}", nameof(LogStarted), StartEndPrefix);

        }

        private const string StartEndPrefix = ">>>>>>>>";

        private static ILog \_log;

    }

}

### log4net.config

<log4net>

    <!-- appender for Visual Studio and Azure console output -->

    <appender name="ConsoleAppender" type="log4net.Appender.TraceAppender">

        <layout type="log4net.Layout.PatternLayout">

            <param name="ConversionPattern"

                         value="%utcdate [%3thread] %-5level %-60logger [%3property{UserId} | %8property{SessionToken}] %message%newline" />

        </layout>

    </appender>

    <!-- appender for general log file -->

    <appender name="AllRollingFileAppender" type="log4net.Appender.RollingFileAppender">

        <file type="log4net.Util.PatternString" value="%property{LogFileDirectory}\HeyImIn-All.log" />

        <appendToFile value="true" />

        <encoding value="utf-8" />

        <!-- up to 5 files are allowed, each 20 megabytes in size -->

        <maximumFileSize value="20MB" />

        <maxSizeRollBackups value="5" />

        <!-- true means keep the .log extension of the rolled files -->

        <preserveLogFileNameExtension value="true" />

        <rollingStyle value="Size" />

        <!-- true means: always log to Logfile.log / false means: log to something like Logfile-20171130.0.log -->

        <staticLogFileName value="true" />

        <layout type="log4net.Layout.PatternLayout">

            <param name="ConversionPattern"

                         value="%utcdate [%3thread] %-5level %-70logger [%3property{UserId} | %8property{SessionToken}] %message%newline" />

        </layout>

    </appender>

    <!-- appender for audit log file -->

    <appender name="AuditRollingFileAppender" type="log4net.Appender.RollingFileAppender">

        <file type="log4net.Util.PatternString" value="%property{LogFileDirectory}\HeyImIn-Audit.log" />

        <appendToFile value="true" />

        <!-- filenames end with the year and month, e.g. Audit-201803.log -->

        <datePattern value="-yyyyMM" />

        <encoding value="utf-8" />

        <!-- delete logfiles older than 7 months -->

        <maxSizeRollBackups value="7" />

        <!-- true means keep the .log extension of the rolled files -->

        <preserveLogFileNameExtension value="true" />

        <rollingStyle value="Date" />

        <!-- true means: always log to Logfile.log / false means: log to something like Logfile-20171130.0.log -->

        <staticLogFileName value="false" />

        <layout type="log4net.Layout.PatternLayout">

            <!-- Same pattern as main log without the logger setting, as the logger would always be AuditLogger -->

            <param name="ConversionPattern"

                         value="%utcdate [%3thread] %-5level [%3property{UserId} | %8property{SessionToken}] %message%newline" />

        </layout>

    </appender>

    <!-- the logger specifically for audit messages, which will also be logged to the general log file -->

    <logger name="AuditLogger">

        <level value="ALL" />

        <appender-ref ref="AuditRollingFileAppender" />

    </logger>

    <root>

        <!-- options are "ALL", "DEBUG", "INFO", "WARN", "ERROR", "FATAL" and "OFF" -->

        <level value="DEBUG" />

        <appender-ref ref="ConsoleAppender" />

        <appender-ref ref="AllRollingFileAppender" />

    </root>

</log4net>

### Web.config

<?xml version="1.0" encoding="utf-8"?>

<!--

    For more information on how to configure your ASP.NET application, please visit

    https://go.microsoft.com/fwlink/?LinkId=301879

    -->

<configuration>

    <configSections>

        <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"/>

        <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">

            <section name="HeyImIn.Properties.Settings" type="System.Configuration.ClientSettingsSection, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"/>

        </sectionGroup>

    </configSections>

    <appSettings>

        <!-- Adjust this value in the future to make sure the passwords are stored securely, see https://wildlyinaccurate.com/bcrypt-choosing-a-work-factor/ for a guide on choosing a work factor -->

        <add key="PasswordHashWorkFactor" value="10"/>

        <!-- Base URL used within emails -->

        <add key="FrontendBaseUrl" value="https://hey-im-in.azurewebsites.net/#/"/>

        <!-- Time Zone used within emails, possible values: https://msdn.microsoft.com/en-us/library/gg154758.aspx -->

        <add key="MailTimeZoneName" value="W. Europe Standard Time"/>

    </appSettings>

    <system.web>

        <compilation debug="true" targetFramework="4.7"/>

        <httpRuntime targetFramework="4.7"/>

        <globalization uiCulture="de-CH" culture="de-CH"/>

    </system.web>

    <system.webServer>

        <urlCompression doStaticCompression="true" doDynamicCompression="true"/>

        <handlers>

            <remove name="ExtensionlessUrlHandler-Integrated-4.0"/>

            <remove name="OPTIONSVerbHandler"/>

            <remove name="TRACEVerbHandler"/>

            <add name="ExtensionlessUrlHandler-Integrated-4.0" path="\*." verb="\*" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0"/>

        </handlers>

    </system.webServer>

    <runtime>

        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

            <dependentAssembly>

                <assemblyIdentity name="System.IO" publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-4.1.1.0" newVersion="4.1.1.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="System.Security.Cryptography.X509Certificates" publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-4.1.1.2" newVersion="4.1.1.2"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="System.Runtime" publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-4.1.1.0" newVersion="4.1.1.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-11.0.0.0" newVersion="11.0.0.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="MySql.Data" publicKeyToken="c5687fc88969c44d" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-6.10.4.0" newVersion="6.9.10.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="System.Net.Http" publicKeyToken="b03f5f7f11d50a3a" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-4.1.1.2" newVersion="4.1.1.2"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="System.Web.Http" publicKeyToken="31bf3856ad364e35" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-5.2.4.0" newVersion="5.2.4.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="Autofac" publicKeyToken="17863af14b0044da" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-4.6.2.0" newVersion="4.6.2.0"/>

            </dependentAssembly>

            <dependentAssembly>

                <assemblyIdentity name="System.Net.Http.Formatting" publicKeyToken="31bf3856ad364e35" culture="neutral"/>

                <bindingRedirect oldVersion="0.0.0.0-5.2.4.0" newVersion="5.2.4.0"/>

            </dependentAssembly>

        </assemblyBinding>

    </runtime>

    <system.codedom>

        <compilers>

            <compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.8.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>

            <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.8.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /define:\_MYTYPE=\&quot;Web\&quot; /optionInfer+"/>

        </compilers>

    </system.codedom>

    <connectionStrings>

        <add name="HeyImIn" providerName="System.Data.SqlClient" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=HeyImIn;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\HeyImIn.mdf"/>

    </connectionStrings>

    <entityFramework>

        <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

            <parameters>

                <parameter value="mssqllocaldb"/>

            </parameters>

        </defaultConnectionFactory>

        <providers>

            <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"/>

        </providers>

    </entityFramework>

</configuration>

### CronController.cs

using System;

using System.Collections.Generic;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using HeyImIn.WebApplication.Services;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    [AllowAnonymous]

    public class CronController : ApiController

    {

        private readonly IEnumerable<ICronService> \_cronRunners;

        public CronController(IEnumerable<ICronService> cronRunners)

        {

            \_cronRunners = cronRunners;

        }

        /// <summary>

        /// Default / fallback route which redirects to index.html

        /// </summary>

        [HttpPost]

        public async Task Run()

        {

            \_log.DebugFormat("{0}(): Running Cron jobs", nameof(Run));

            try

            {

                foreach (ICronService cronService in \_cronRunners)

                {

                    await cronService.RunAsync();

                }

            }

            catch (Exception e)

            {

                \_log.ErrorFormat("{0}(): Failed to execute cron job, error={1}", nameof(Run), e);

                throw;

            }

        }

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

### HomeController.cs

using System;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Web.Http;

namespace HeyImIn.WebApplication.Controllers

{

    [AllowAnonymous]

    public class HomeController : ApiController

    {

        /// <summary>

        /// Default / fallback route which redirects to index.html

        /// </summary>

        [HttpGet]

        [Route("")]

        public HttpResponseMessage Index()

        {

            // Taken from MediaGateway project

            HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.MovedPermanently);

            response.Headers.Location = new Uri("client/index.html", UriKind.Relative);

            response.Headers.CacheControl = CacheControlHeaderValue.Parse("no-cache, no-store, must-revalidate");

            return response;

        }

    }

}

### InviteToEventController.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    [AuthenticateUser]

    public class InviteToEventController : ApiController

    {

        public InviteToEventController(INotificationService notificationService, GetDatabaseContext getDatabaseContext)

        {

            \_notificationService = notificationService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Adds new appointments to the event

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> InviteParticipants([FromBody] InviteParticipantsDto inviteParticipantsDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (inviteParticipantsDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.Organizer)

                    .Include(e => e.EventInvitations)

                    .FirstOrDefaultAsync(e => e.Id == inviteParticipantsDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (@event.Organizer != currentUser)

                {

                    \_log.InfoFormat("{0}(): Tried to add appointments to the event {1}, which he's not organizing", nameof(InviteParticipants), @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                var mailInvites = new List<(string email, EventInvitation invite)>();

                var userInvites = new List<(User user, EventInvitation invite)>();

                foreach (string emailAddress in inviteParticipantsDto.EmailAddresses)

                {

                    // Create new invite

                    EventInvitation invite = context.EventInvitations.Create();

                    invite.Requested = DateTime.UtcNow;

                    invite.Event = @event;

                    context.EventInvitations.Add(invite);

                    // Check if a profile with this email exists

                    User existingUser = await context.Users.FirstOrDefaultAsync(u => u.Email == emailAddress);

                    if (existingUser == null)

                    {

                        // No user exists

                        mailInvites.Add((emailAddress, invite));

                    }

                    else

                    {

                        // User with mail exists => Check if he's already part of the event

                        if (@event.EventParticipations.Any(e => e.Participant.Email == emailAddress))

                        {

                            return BadRequest($"Die E-Mail-Adresse {emailAddress} nimmt bereits am Event teil");

                        }

                        userInvites.Add((existingUser, invite));

                    }

                }

                await context.SaveChangesAsync();

                await \_notificationService.SendInvitationLinkAsync(userInvites, mailInvites);

                return Ok();

            }

        }

        /// <summary>

        /// Accepts an invite to an event, as long as the invite is considered valid

        /// </summary>

        /// <returns><see cref="Event.Id" /> of the accepted invite</returns>

        [HttpPost]

        [ResponseType(typeof(int))]

        public async Task<IHttpActionResult> AcceptInvitation([FromBody] AcceptInvitationDto acceptInvitationDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (acceptInvitationDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                EventInvitation invitation = await context.EventInvitations.Include(e => e.Event).FirstOrDefaultAsync(e => e.Token == acceptInvitationDto.InviteToken);

                if (invitation == null)

                {

                    return BadRequest(RequestStringMessages.InvitationInvalid);

                }

                if (invitation.Event.IsPrivate)

                {

                    // Invitations to private events require to still be valid and not already used

                    if (invitation.Used)

                    {

                        return BadRequest(RequestStringMessages.InvitationAlreadyUsed);

                    }

                    if (invitation.Requested + \_inviteTimeout < DateTime.UtcNow)

                    {

                        return BadRequest(RequestStringMessages.InvitationInvalid);

                    }

                }

                // Invitations to public events are theoretically pointless as the user could join without the invitation

                // => These invitations are always considered as valid, even if already used

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                EventParticipation participation = context.EventParticipations.Create();

                participation.Event = invitation.Event;

                participation.Participant = currentUser;

                invitation.Event.EventParticipations.Add(participation);

                invitation.Used = true;

                await context.SaveChangesAsync();

                return Ok(invitation.EventId);

            }

        }

        private readonly INotificationService \_notificationService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        /// <summary>

        /// A invite for a private event gets invalidated after this time period passed

        /// </summary>

        private static readonly TimeSpan \_inviteTimeout = TimeSpan.FromDays(7);

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

### OrganizeAppointmentController.cs

using System;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.MailNotifier.Models;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.Services;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    [AuthenticateUser]

    public class OrganizeAppointmentController : ApiController

    {

        public OrganizeAppointmentController(INotificationService notificationService, IDeleteService deleteService, GetDatabaseContext getDatabaseContext)

        {

            \_notificationService = notificationService;

            \_deleteService = deleteService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Deletes an event and all underlying appointments

        /// Informs the users about the deletion

        /// </summary>

        /// <param name="appointmentId">

        /// <see cref="Appointment.Id" />

        /// </param>

        [HttpDelete]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> DeleteAppointment(int appointmentId)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Appointment appointment = await context.Appointments

                    .Include(a => a.Event)

                    .Include(a => a.Event.Organizer)

                    .Include(a => a.AppointmentParticipations)

                    .FirstOrDefaultAsync(e => e.Id == appointmentId);

                if (appointment == null)

                {

                    return BadRequest(RequestStringMessages.AppointmentNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                Event @event = appointment.Event;

                if (@event.Organizer != currentUser)

                {

                    \_log.InfoFormat("{0}(): Tried to delete appointment {1}, which he's not organizing", nameof(DeleteAppointment), appointment.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                AppointmentNotificationInformation notificationInformation = \_deleteService.DeleteAppointmentLocally(context, appointment);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Canceled appointment {1}", nameof(DeleteAppointment), appointment.Id);

                await \_notificationService.NotifyAppointmentExplicitlyCanceledAsync(notificationInformation, @event);

                return Ok();

            }

        }

        /// <summary>

        /// Adds new appointsments to the event

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> AddAppointments([FromBody] AddAppointsmentsDto addAppointsmentsDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (addAppointsmentsDto == null))

            {

                return BadRequest();

            }

            if (addAppointsmentsDto.StartTimes.Any(a => a < DateTime.UtcNow))

            {

                return BadRequest(RequestStringMessages.AppointmentsHaveToStartInTheFuture);

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events.Include(e => e.Organizer).FirstOrDefaultAsync(e => e.Id == addAppointsmentsDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (@event.Organizer != currentUser)

                {

                    \_log.InfoFormat("{0}(): Tried to add appointments to the event {1}, which he's not organizing", nameof(AddAppointments), @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                foreach (DateTime startTime in addAppointsmentsDto.StartTimes)

                {

                    Appointment newAppointment = context.Appointments.Create();

                    newAppointment.Event = @event;

                    newAppointment.StartTime = startTime;

                    context.Appointments.Add(newAppointment);

                }

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Added {1} appointments to event {2}", nameof(AddAppointments), addAppointsmentsDto.StartTimes.Length, @event.Id);

                return Ok();

            }

        }

        /// <summary>

        /// Sets the response to an appointment

        /// Will add the user to the event if he's not part of it yet

        /// </summary>

        /// <param name="setAppointmentResponseDto">

        /// <see cref="User.Id" />

        /// <see cref="Appointment.Id" />

        /// </param>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> SetAppointmentResponse([FromBody] SetAppointmentResponseDto setAppointmentResponseDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (setAppointmentResponseDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Appointment appointment = await context.Appointments

                    .Include(a => a.Event)

                    .Include(a => a.Event.Organizer)

                    .Include(a => a.Event.EventParticipations)

                    .Include(a => a.AppointmentParticipations)

                    .FirstOrDefaultAsync(a => a.Id == setAppointmentResponseDto.AppointmentId);

                if (appointment == null)

                {

                    return BadRequest(RequestStringMessages.AppointmentNotFound);

                }

                User userToSetResponseFor = await context.Users.FindAsync(setAppointmentResponseDto.UserId);

                if (userToSetResponseFor == null)

                {

                    return BadRequest(RequestStringMessages.UserNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                bool changingOtherUser = currentUser != userToSetResponseFor;

                bool userIsPartOfEvent = appointment.Event.EventParticipations.Select(e => e.Participant).Contains(userToSetResponseFor);

                if (changingOtherUser)

                {

                    if (appointment.Event.Organizer != currentUser)

                    {

                        // Only the organizer is allowed to change another user

                        \_log.InfoFormat("{0}(): Tried to set response for user {1} for the appointment {2}, which he's not organizing", nameof(SetAppointmentResponse), userToSetResponseFor.Id, appointment.Id);

                        return BadRequest(RequestStringMessages.OrganizorRequired);

                    }

                    if (!userIsPartOfEvent)

                    {

                        // A organizer shouldn't be able to add a user to the event, unless the user is participating in the event

                        // This should prevent that a user gets added to an event he never had anything to do with

                        return BadRequest(RequestStringMessages.UserNotPartOfEvent);

                    }

                }

                // Ensure a user can't participate in a private event without an invitation

                if (!userIsPartOfEvent && appointment.Event.IsPrivate && (appointment.Event.Organizer != currentUser))

                {

                    return BadRequest(RequestStringMessages.InvitationRequired);

                }

                AppointmentParticipation appointmentParticipation = appointment.AppointmentParticipations.FirstOrDefault(e => e.Participant == userToSetResponseFor);

                if (appointmentParticipation == null)

                {

                    // Create a new participation if there isn't an existing one

                    appointmentParticipation = context.AppointmentParticipations.Create();

                    appointmentParticipation.Participant = userToSetResponseFor;

                    appointmentParticipation.Appointment = appointment;

                    context.AppointmentParticipations.Add(appointmentParticipation);

                }

                if (setAppointmentResponseDto.Response.HasValue)

                {

                    appointmentParticipation.AppointmentParticipationAnswer = setAppointmentResponseDto.Response.Value;

                }

                else

                {

                    context.AppointmentParticipations.Remove(appointmentParticipation);

                }

                if (!appointment.Event.EventParticipations.Select(e => e.Participant).Contains(currentUser))

                {

                    // Automatically add a user to an event if he's not yet part of it

                    EventParticipation eventParticipation = context.EventParticipations.Create();

                    eventParticipation.Event = appointment.Event;

                    eventParticipation.Participant = currentUser;

                    context.EventParticipations.Add(eventParticipation);

                    \_auditLog.InfoFormat("{0}(): Joined event {1} automatically after joining appointment {2}", nameof(SetAppointmentResponse), appointment.Event.Id, appointment.Id);

                }

                await context.SaveChangesAsync();

                // Handle notifications

                if (changingOtherUser)

                {

                    \_auditLog.InfoFormat("{0}(response={1}): The organizer set the response to the appointment {2} for user {3}", nameof(SetAppointmentResponse), setAppointmentResponseDto.Response, appointment.Id, userToSetResponseFor.Id);

                    await \_notificationService.NotifyOrganizerUpdatedUserInfoAsync(appointment.Event, userToSetResponseFor, "Der Organisator hat Ihre Zusage an einem Termin editiert.");

                }

                else

                {

                    \_log.DebugFormat("{0}(response={1}): Set own response for appointment {2}", nameof(SetAppointmentResponse), setAppointmentResponseDto.Response, appointment.Id);

                }

                await \_notificationService.SendLastMinuteChangeIfRequiredAsync(appointment);

                return Ok();

            }

        }

        private readonly INotificationService \_notificationService;

        private readonly IDeleteService \_deleteService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        private static readonly ILog \_auditLog = LogHelpers.GetAuditLog();

    }

}

### OrganizeEventController.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.MailNotifier.Models;

using HeyImIn.WebApplication.FrontendModels;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.FrontendModels.ResponseTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.Services;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    [AuthenticateUser]

    public class OrganizeEventController : ApiController

    {

        public OrganizeEventController(INotificationService notificationService, IDeleteService deleteService, GetDatabaseContext getDatabaseContext)

        {

            \_notificationService = notificationService;

            \_deleteService = deleteService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Deletes an event and all underlying appointments

        /// Informs the users about the deletion

        /// </summary>

        /// <param name="eventId">

        /// <see cref="Event.Id" />

        /// </param>

        [HttpDelete]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> DeleteEvent(int eventId)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.Organizer)

                    .Include(e => e.EventInvitations)

                    .Include(e => e.EventParticipations)

                    .Include(e => e.Appointments)

                    .Include(e => e.Appointments.Select(a => a.AppointmentParticipations))

                    .FirstOrDefaultAsync(e => e.Id == eventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (@event.Organizer != currentUser)

                {

                    \_log.InfoFormat("{0}(): Tried to delete event {1}, which he's not organizing", nameof(DeleteEvent), @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                EventNotificationInformation notificationInformation = \_deleteService.DeleteEventLocally(context, @event);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Deleted event {1}", nameof(DeleteEvent), @event.Id);

                await \_notificationService.NotifyEventDeletedAsync(notificationInformation);

                return Ok();

            }

        }

        /// <summary>

        /// Updates general information about an event

        /// Informs the users about the change

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> UpdateEventInfo([FromBody] UpdatedEventInfoDto updatedEventInfoDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (updatedEventInfoDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.Organizer)

                    .Include(e => e.EventParticipations)

                    .FirstOrDefaultAsync(e => e.Id == updatedEventInfoDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (@event.Organizer != currentUser)

                {

                    \_log.InfoFormat("{0}(): Tried to update event {1}, which he's not organizing", nameof(UpdateEventInfo), @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                @event.Title = updatedEventInfoDto.Title;

                @event.MeetingPlace = updatedEventInfoDto.MeetingPlace;

                @event.Description = updatedEventInfoDto.Description;

                @event.IsPrivate = updatedEventInfoDto.IsPrivate;

                @event.ReminderTimeWindowInHours = updatedEventInfoDto.ReminderTimeWindowInHours;

                @event.SummaryTimeWindowInHours = updatedEventInfoDto.SummaryTimeWindowInHours;

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Updated event {1}", nameof(UpdateEventInfo), @event.Id);

                await \_notificationService.NotifyEventUpdatedAsync(@event);

                return Ok();

            }

        }

        /// <summary>

        /// Creates a new event

        /// </summary>

        /// <returns>

        /// <see cref="Event.Id" />

        /// </returns>

        [HttpPost]

        [ResponseType(typeof(int))]

        public async Task<IHttpActionResult> CreateEvent([FromBody] GeneralEventInformation generalEventInformation)

        {

            // Validate parameters

            if (!ModelState.IsValid || (generalEventInformation == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event newEvent = context.Events.Create();

                newEvent.Title = generalEventInformation.Title;

                newEvent.Description = generalEventInformation.Description;

                newEvent.MeetingPlace = generalEventInformation.MeetingPlace;

                newEvent.IsPrivate = generalEventInformation.IsPrivate;

                newEvent.ReminderTimeWindowInHours = generalEventInformation.ReminderTimeWindowInHours;

                newEvent.SummaryTimeWindowInHours = generalEventInformation.SummaryTimeWindowInHours;

                newEvent.OrganizerId = ActionContext.Request.GetUserId();

                context.Events.Add(newEvent);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Created event {1} ({2})", nameof(CreateEvent), newEvent.Id, newEvent.Title);

                return Ok(newEvent.Id);

            }

        }

        /// <summary>

        /// Loads the details about an event which is required to edit it

        /// </summary>

        /// <param name="eventId">

        /// <see cref="Event.Id" />

        /// </param>

        /// <returns>

        /// <see cref="EditEventDetails" />

        /// </returns>

        [HttpGet]

        [ResponseType(typeof(EditEventDetails))]

        public async Task<IHttpActionResult> GetEditDetails(int eventId)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.EventParticipations)

                    .Include(e => e.Appointments)

                    .FirstOrDefaultAsync(e => e.Id == eventId);

                if (@event == null)

                {

                    return NotFound();

                }

                int currentUserId = ActionContext.Request.GetUserId();

                if (@event.OrganizerId != currentUserId)

                {

                    \_log.InfoFormat("{0}(): Tried to edit event {1}, which he's not organizing", nameof(GetEditDetails), @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                List<User> allParticipants = @event.EventParticipations.Select(e => e.Participant).ToList();

                List<AppointmentDetails> upcomingAppointments = @event.Appointments

                    .Where(a => a.StartTime >= DateTime.UtcNow)

                    .OrderBy(a => a.StartTime)

                    .Select(a => AppointmentDetails.FromAppointment(a, currentUserId, allParticipants))

                    .ToList();

                List<EventParticipantInformation> currentEventParticipation = @event.EventParticipations.Select(EventParticipantInformation.FromParticipation).ToList();

                ViewEventInformation viewEventInformation = ViewEventInformation.FromEvent(@event, currentUserId);

                return Ok(new EditEventDetails(viewEventInformation, upcomingAppointments, currentEventParticipation));

            }

        }

        private readonly INotificationService \_notificationService;

        private readonly IDeleteService \_deleteService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        private static readonly ILog \_auditLog = LogHelpers.GetAuditLog();

    }

}

### ParticipateEventController.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.FrontendModels.ResponseTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    [AuthenticateUser]

    public class ParticipateEventController : ApiController

    {

        public ParticipateEventController(INotificationService notificationService, GetDatabaseContext getDatabaseContext)

        {

            \_notificationService = notificationService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Loads the <see cref="Event" />s the current user should see including the latest upcoming

        /// <see cref="Appointment" />

        /// </summary>

        /// <returns>

        /// <see cref="EventOverview" />

        /// </returns>

        [HttpGet]

        [ResponseType(typeof(EventOverview))]

        public async Task<IHttpActionResult> GetOverview()

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                int currentUserId = ActionContext.Request.GetUserId();

                List<Event> yourEvents = await context.Events

                    .Where(e => (e.OrganizerId == currentUserId) || e.EventParticipations.Select(ep => ep.ParticipantId).Contains(currentUserId))

                    .Include(e => e.Appointments)

                    .Include(e => e.EventParticipations)

                    .ToListAsync();

                List<Event> publicEvents = await context.Events

                    .Where(e => !e.IsPrivate)

                    .Include(e => e.Appointments)

                    .Include(e => e.EventParticipations)

                    .ToListAsync();

                List<EventOverviewInformation> yourEventInformations = yourEvents

                    .Select(e => EventOverviewInformation.FromEvent(e, currentUserId))

                    .OrderBy(e => e.LatestAppointmentInformation?.StartTime ?? DateTime.MaxValue)

                    .ToList();

                List<EventOverviewInformation> publicEventInformations = publicEvents

                    .Except(yourEvents)

                    .Select(e => EventOverviewInformation.FromEvent(e, currentUserId))

                    .OrderBy(e => e.LatestAppointmentInformation?.StartTime ?? DateTime.MaxValue)

                    .ToList();

                return Ok(new EventOverview(yourEventInformations, publicEventInformations));

            }

        }

        /// <summary>

        /// Loads the <see cref="Event" /> for the provided <paramref name="eventId" /> and upcoming <see cref="Appointment" />

        /// </summary>

        /// <param name="eventId">

        /// <see cref="Event.Id" />

        /// </param>

        /// <returns>

        /// <see cref="EventDetails" />

        /// </returns>

        [HttpGet]

        [ResponseType(typeof(EventDetails))]

        public async Task<IHttpActionResult> GetDetails(int eventId)

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.Organizer)

                    .Include(e => e.EventParticipations)

                    .Include(e => e.Appointments)

                    .FirstOrDefaultAsync(e => e.Id == eventId);

                if (@event == null)

                {

                    return NotFound();

                }

                int currentUserId = ActionContext.Request.GetUserId();

                ViewEventInformation viewEventInformation = ViewEventInformation.FromEvent(@event, currentUserId);

                if (!viewEventInformation.CurrentUserDoesParticipate && @event.IsPrivate && (@event.OrganizerId != currentUserId))

                {

                    return BadRequest(RequestStringMessages.InvitationRequired);

                }

                List<User> allParticipants = @event.EventParticipations.Select(e => e.Participant).ToList();

                List<AppointmentDetails> upcomingAppointments = @event.Appointments

                    .Where(a => a.StartTime >= DateTime.UtcNow)

                    .OrderBy(a => a.StartTime)

                    .Take(ShownAppointmentsPerEvent)

                    .Select(a => AppointmentDetails.FromAppointment(a, currentUserId, allParticipants))

                    .ToList();

                EventParticipation currentEventParticipation = @event.EventParticipations.FirstOrDefault(e => e.ParticipantId == currentUserId);

                NotificationConfigurationResponse notificationConfigurationResponse = null;

                if (currentEventParticipation != null)

                {

                    notificationConfigurationResponse = NotificationConfigurationResponse.FromParticipation(currentEventParticipation);

                }

                return Ok(new EventDetails(viewEventInformation, upcomingAppointments, notificationConfigurationResponse));

            }

        }

        /// <summary>

        /// Explicitly adds the current user to an event

        /// </summary>

        /// <param name="joinEventDto">

        /// <see cref="Event.Id" />

        /// </param>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> JoinEvent([FromBody] JoinEventDto joinEventDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (joinEventDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events

                    .Include(e => e.EventParticipations)

                    .FirstOrDefaultAsync(e => e.Id == joinEventDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (@event.EventParticipations.Any(e => e.Participant == currentUser))

                {

                    return BadRequest(RequestStringMessages.UserAlreadyPartOfEvent);

                }

                if (@event.IsPrivate && (@event.Organizer != currentUser))

                {

                    return BadRequest(RequestStringMessages.InvitationRequired);

                }

                EventParticipation eventParticipation = context.EventParticipations.Create();

                eventParticipation.Event = @event;

                eventParticipation.Participant = currentUser;

                context.EventParticipations.Add(eventParticipation);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Joined event {1}", nameof(JoinEvent), @event.Id);

                return Ok();

            }

        }

        /// <summary>

        /// Removes the specified user from an event. A user can remove himself any the <see cref="Event.Organizer" /> can

        /// remove any user

        /// </summary>

        /// <param name="removeFromEventDto">

        /// <see cref="User.Id" />

        /// <see cref="Event.Id" />

        /// </param>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> RemoveFromEvent([FromBody] RemoveFromEventDto removeFromEventDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (removeFromEventDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events.FindAsync(removeFromEventDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User userToRemove = await context.Users.FindAsync(removeFromEventDto.UserId);

                if (userToRemove == null)

                {

                    return BadRequest(RequestStringMessages.UserNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                bool changingOtherUser = currentUser != userToRemove;

                if (changingOtherUser && (@event.Organizer != currentUser))

                {

                    \_log.InfoFormat("{0}(): Tried to remove user {1} from then event {2}, which he's not organizing", nameof(RemoveFromEvent), userToRemove.Id, @event.Id);

                    return BadRequest(RequestStringMessages.OrganizorRequired);

                }

                EventParticipation participation = @event.EventParticipations.FirstOrDefault(e => e.Participant == userToRemove);

                if (participation == null)

                {

                    return BadRequest(RequestStringMessages.UserNotPartOfEvent);

                }

                // Remove event participation

                context.EventParticipations.Remove(participation);

                // Remove appointment participations within the event

                List<AppointmentParticipation> appointmentParticipations = userToRemove.AppointmentParticipations.Where(a => a.Appointment.Event == @event).ToList();

                List<Appointment> appointments = appointmentParticipations.Select(a => a.Appointment).ToList();

                context.AppointmentParticipations.RemoveRange(appointmentParticipations);

                await context.SaveChangesAsync();

                // Handle notifications

                if (changingOtherUser)

                {

                    \_auditLog.InfoFormat("{0}(): The organizer removed the user {1} from the event {2}", nameof(RemoveFromEvent), userToRemove.Id, @event.Id);

                    await \_notificationService.NotifyOrganizerUpdatedUserInfoAsync(@event, userToRemove, "Der Organisator hat Sie vom Event entfernt.");

                }

                else

                {

                    \_auditLog.InfoFormat("{0}(): Left the event {1}", nameof(RemoveFromEvent), @event.Id);

                }

                foreach (Appointment appointment in appointments)

                {

                    await \_notificationService.SendLastMinuteChangeIfRequiredAsync(appointment);

                }

                return Ok();

            }

        }

        /// <summary>

        /// Configures the enabled notifications of the current user for the specified event

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> ConfigureNotifications([FromBody] NotificationConfigurationDto notificationConfigurationDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (notificationConfigurationDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                Event @event = await context.Events.FindAsync(notificationConfigurationDto.EventId);

                if (@event == null)

                {

                    return BadRequest(RequestStringMessages.EventNotFound);

                }

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                EventParticipation participation = @event.EventParticipations.FirstOrDefault(e => e.Participant == currentUser);

                if (participation == null)

                {

                    return BadRequest(RequestStringMessages.UserNotPartOfEvent);

                }

                participation.SendReminderEmail = notificationConfigurationDto.SendReminderEmail;

                participation.SendSummaryEmail = notificationConfigurationDto.SendSummaryEmail;

                participation.SendLastMinuteChangesEmail = notificationConfigurationDto.SendLastMinuteChangesEmail;

                await context.SaveChangesAsync();

                \_log.InfoFormat("{0}(): Updated notification settings", nameof(ConfigureNotifications));

                return Ok();

            }

        }

        private readonly INotificationService \_notificationService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        /// <summary>

        /// The maximum amount of appointments to show in the overview for each event

        /// In the edit view all events are shown

        /// </summary>

        private const int ShownAppointmentsPerEvent = 5;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        private static readonly ILog \_auditLog = LogHelpers.GetAuditLog();

    }

}

### ResetPasswordController.cs

using System;

using System.Data.Entity;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Authentication;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.Helpers;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    public class ResetPasswordController : ApiController

    {

        public ResetPasswordController(IPasswordService passwordService, INotificationService notificationService, GetDatabaseContext getDatabaseContext)

        {

            \_passwordService = passwordService;

            \_notificationService = notificationService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Sends a password reset code to the provided email address, if a user with than email is registered

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        [AllowAnonymous]

        public async Task<IHttpActionResult> RequestPasswordReset([FromBody] RequestPasswordResetDto requestPasswordResetDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (requestPasswordResetDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                User user = await context.Users.FirstOrDefaultAsync(u => u.Email == requestPasswordResetDto.Email);

                if (user == null)

                {

                    return BadRequest(RequestStringMessages.NoProfileWithEmailFound);

                }

                PasswordReset passwordReset = context.PasswordResets.Create();

                passwordReset.User = user;

                passwordReset.Requested = DateTime.UtcNow;

                context.PasswordResets.Add(passwordReset);

                await context.SaveChangesAsync();

                await \_notificationService.SendPasswordResetTokenAsync(passwordReset.Token, user);

                return Ok();

            }

        }

        /// <summary>

        /// Sets a new password for a password reset code

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        [AllowAnonymous]

        public async Task<IHttpActionResult> ResetPassword([FromBody] ResetPasswordDto resetPasswordDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (resetPasswordDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                PasswordReset passwordReset = await context.PasswordResets.FindAsync(resetPasswordDto.PasswordResetToken);

                if (passwordReset == null)

                {

                    \_log.DebugFormat("{0}(resetToken={1}): Couldn't find the password reset token", nameof(ResetPassword), resetPasswordDto.PasswordResetToken);

                    return BadRequest(RequestStringMessages.ResetCodeInvalid);

                }

                if (passwordReset.Used || (passwordReset.Requested - DateTime.UtcNow > \_resetTokenValidTimeSpan))

                {

                    \_log.InfoFormat("{0}(resetToken={1}): Tried to reset password for user {2} with an expired or used token", nameof(ResetPassword), resetPasswordDto.PasswordResetToken, passwordReset.UserId);

                    return BadRequest(RequestStringMessages.ResetCodeAlreadyUsed);

                }

                passwordReset.User.PasswordHash = \_passwordService.HashPassword(resetPasswordDto.NewPassword);

                passwordReset.Used = true;

                await context.SaveChangesAsync();

                \_log.InfoFormat("{0}(resetToken={1}): Reset password for user {2}", nameof(ResetPassword), resetPasswordDto.PasswordResetToken, passwordReset.UserId);

                return Ok();

            }

        }

        private readonly IPasswordService \_passwordService;

        private readonly INotificationService \_notificationService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        private static readonly TimeSpan \_resetTokenValidTimeSpan = TimeSpan.FromHours(2);

    }

}

### SessionController.cs

using System;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Authentication;

using HeyImIn.Database.Models;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.FrontendModels.ResponseTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    public class SessionController : ApiController

    {

        public SessionController(IAuthenticationService authenticationService, ISessionService sessionService)

        {

            \_authenticationService = authenticationService;

            \_sessionService = sessionService;

        }

        /// <summary>

        /// Tries to start a new session for the provided credentials

        /// </summary>

        /// <returns>The created <see cref="FrontendSession" /> containing user information</returns>

        [HttpPost]

        [ResponseType(typeof(FrontendSession))]

        [AllowAnonymous]

        public async Task<IHttpActionResult> StartSession([FromBody] StartSessionDto startSessionDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (startSessionDto == null))

            {

                return BadRequest();

            }

            var (authenticated, foundUser) = await \_authenticationService.AuthenticateAsync(startSessionDto.Email, startSessionDto.Password);

            if (!authenticated)

            {

                return Unauthorized();

            }

            Guid sessionToken = await \_sessionService.CreateSessionAsync(foundUser.Id, true);

            \_auditLog.InfoFormat("{0}(userId={1}): User logged in", nameof(StartSession), foundUser.Id);

            return Ok(new FrontendSession(sessionToken, foundUser.Id, foundUser.FullName, foundUser.Email));

        }

        /// <summary>

        /// Loads an already active session

        /// </summary>

        /// <param name="sessionToken">Unique session token</param>

        /// <returns>The found <see cref="FrontendSession" /></returns>

        [HttpGet]

        [ResponseType(typeof(FrontendSession))]

        [AllowAnonymous]

        public async Task<IHttpActionResult> GetSession(Guid sessionToken)

        {

            Session session = await \_sessionService.GetAndExtendSessionAsync(sessionToken);

            if (session == null)

            {

                return Unauthorized();

            }

            return Ok(new FrontendSession(session.Token, session.UserId, session.User.FullName, session.User.Email));

        }

        /// <summary>

        /// Stops the active session => Log out and invalidate session

        /// </summary>

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthenticateUser]

        public async Task<IHttpActionResult> StopActiveSession()

        {

            Guid token = Request.GetSessionToken();

            await \_sessionService.InvalidateSessionAsync(token);

            \_auditLog.InfoFormat("{0}(): User logged out", nameof(StopActiveSession));

            return Ok();

        }

        private readonly IAuthenticationService \_authenticationService;

        private readonly ISessionService \_sessionService;

        private static readonly ILog \_auditLog = LogHelpers.GetAuditLog();

    }

}

### UserController.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using HeyImIn.Authentication;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using HeyImIn.MailNotifier;

using HeyImIn.MailNotifier.Models;

using HeyImIn.WebApplication.FrontendModels.ParameterTypes;

using HeyImIn.WebApplication.FrontendModels.ResponseTypes;

using HeyImIn.WebApplication.Helpers;

using HeyImIn.WebApplication.Services;

using HeyImIn.WebApplication.WebApiComponents;

using log4net;

namespace HeyImIn.WebApplication.Controllers

{

    public class UserController : ApiController

    {

        public UserController(IPasswordService passwordService, ISessionService sessionService, INotificationService notificationService, IDeleteService deleteService, GetDatabaseContext getDatabaseContext)

        {

            \_passwordService = passwordService;

            \_sessionService = sessionService;

            \_notificationService = notificationService;

            \_deleteService = deleteService;

            \_getDatabaseContext = getDatabaseContext;

        }

        /// <summary>

        /// Updates the current user's data

        /// Doesn't invalidate any active sessions

        /// </summary>

        /// <param name="setUserDataDto">New user data</param>

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthenticateUser]

        public async Task<IHttpActionResult> SetNewUserData([FromBody] SetUserDataDto setUserDataDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (setUserDataDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                if (currentUser.Email != setUserDataDto.NewEmail)

                {

                    bool newMailTaken = await context.Users.AnyAsync(u => u.Email == setUserDataDto.NewEmail);

                    if (newMailTaken)

                    {

                        return BadRequest(RequestStringMessages.EmailAlreadyInUse);

                    }

                }

                currentUser.Email = setUserDataDto.NewEmail;

                currentUser.FullName = setUserDataDto.NewFullName;

                await context.SaveChangesAsync();

                \_log.InfoFormat("{0}(): Updated user data", nameof(SetNewUserData));

                return Ok();

            }

        }

        /// <summary>

        /// Changes the current user's password

        /// Doesn't invalidate any active sessions

        /// </summary>

        /// <param name="setPasswordDto">Current and new password</param>

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthenticateUser]

        public async Task<IHttpActionResult> SetNewPassword([FromBody] SetPasswordDto setPasswordDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (setPasswordDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                bool currentPasswordCorrect = \_passwordService.VerifyPassword(setPasswordDto.CurrentPassword, currentUser.PasswordHash);

                if (!currentPasswordCorrect)

                {

                    return BadRequest(RequestStringMessages.CurrentPasswordWrong);

                }

                currentUser.PasswordHash = \_passwordService.HashPassword(setPasswordDto.NewPassword);

                await context.SaveChangesAsync();

                \_log.InfoFormat("{0}(): User changed his password", nameof(SetNewPassword));

                return Ok();

            }

        }

        /// <summary>

        /// Deletes the current user's account and all connections

        /// E.g. sends cancelation for organized and participating events

        /// </summary>

        [HttpDelete]

        [ResponseType(typeof(void))]

        [AuthenticateUser]

        public async Task<IHttpActionResult> DeleteAccount()

        {

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                User currentUser = await ActionContext.Request.GetCurrentUserAsync(context);

                List<Appointment> userAppointments = currentUser.AppointmentParticipations.Select(a => a.Appointment).ToList();

                List<EventNotificationInformation> notificationInformations = \_deleteService.DeleteUserLocally(context, currentUser);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Deleted user {1} ({2}) and all of his events", nameof(DeleteAccount), currentUser.Id, currentUser.FullName);

                foreach (EventNotificationInformation notificationInformation in notificationInformations)

                {

                    await \_notificationService.NotifyEventDeletedAsync(notificationInformation);

                }

                foreach (Appointment appointment in userAppointments)

                {

                    await \_notificationService.SendLastMinuteChangeIfRequiredAsync(appointment);

                }

                return Ok();

            }

        }

        /// <summary>

        /// Registeres a new user

        /// </summary>

        /// <returns>A newly created <see cref="FrontendSession" /> so the registering user doesn't have to log in manually</returns>

        [HttpPost]

        [ResponseType(typeof(FrontendSession))]

        [AllowAnonymous]

        public async Task<IHttpActionResult> Register([FromBody] RegisterDto registerDto)

        {

            // Validate parameters

            if (!ModelState.IsValid || (registerDto == null))

            {

                return BadRequest();

            }

            using (IDatabaseContext context = \_getDatabaseContext())

            {

                bool userWithSameMailExists = await context.Users.AnyAsync(u => u.Email == registerDto.Email);

                if (userWithSameMailExists)

                {

                    return BadRequest(RequestStringMessages.EmailAlreadyInUse);

                }

                User newUser = context.Users.Create();

                newUser.FullName = registerDto.FullName;

                newUser.Email = registerDto.Email;

                newUser.PasswordHash = \_passwordService.HashPassword(registerDto.Password);

                context.Users.Add(newUser);

                await context.SaveChangesAsync();

                \_auditLog.InfoFormat("{0}(): Registered user {1} ({2})", nameof(Register), newUser.Id, newUser.FullName);

                Guid createdSessionToken = await \_sessionService.CreateSessionAsync(newUser.Id, true);

                return Ok(new FrontendSession(createdSessionToken, newUser.Id, registerDto.FullName, registerDto.Email));

            }

        }

        private readonly IPasswordService \_passwordService;

        private readonly ISessionService \_sessionService;

        private readonly INotificationService \_notificationService;

        private readonly IDeleteService \_deleteService;

        private readonly GetDatabaseContext \_getDatabaseContext;

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

        private static readonly ILog \_auditLog = LogHelpers.GetAuditLog();

    }

}

### ParameterTypes (Eine Datei pro Typ)

using System;

using System.ComponentModel.DataAnnotations;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ParameterTypes

{

    public class AcceptInvitationDto

    {

        public Guid InviteToken { get; set; }

    }

    public class AddAppointsmentsDto

    {

        public int EventId { get; set; }

        [Required]

        [MinLength(1)]

        public DateTime[] StartTimes { get; set; }

    }

    public class InviteParticipantsDto

    {

        public int EventId { get; set; }

        [Required]

        [MinLength(1)]

        public string[] EmailAddresses { get; set; }

    }

    public class JoinEventDto

    {

        public int EventId { get; set; }

    }

    public class NotificationConfigurationDto : NotificationConfiguration

    {

        public int EventId { get; set; }

    }

    public class RegisterDto

    {

        [Required]

        [MaxLength(FieldLengths.UserFullNameMaxLength)]

        public string FullName { get; set; }

        [Required]

        [EmailAddress]

        [MaxLength(FieldLengths.UserEmailMaxLength)]

        public string Email { get; set; }

        [Required]

        public string Password { get; set; }

    }

    public class RemoveFromEventDto

    {

        public int EventId { get; set; }

        public int UserId { get; set; }

    }

    public class RequestPasswordResetDto

    {

        [Required]

        [EmailAddress]

        [MaxLength(FieldLengths.UserEmailMaxLength)]

        public string Email { get; set; }

    }

    public class ResetPasswordDto

    {

        [Required]

        public Guid PasswordResetToken { get; set; }

        [Required]

        public string NewPassword { get; set; }

    }

    public class SetAppointmentResponseDto

    {

        public int AppointmentId { get; set; }

        public int UserId { get; set; }

        public AppointmentParticipationAnswer? Response { get; set; }

    }

    public class SetPasswordDto

    {

        [Required]

        public string CurrentPassword { get; set; }

        [Required]

        public string NewPassword { get; set; }

    }

    public class SetUserDataDto

    {

        [Required]

        [MaxLength(FieldLengths.UserFullNameMaxLength)]

        public string NewFullName { get; set; }

        [Required]

        [EmailAddress]

        [MaxLength(FieldLengths.UserEmailMaxLength)]

        public string NewEmail { get; set; }

    }

    public class StartSessionDto

    {

        [Required]

        public string Email { get; set; }

        [Required]

        public string Password { get; set; }

    }

    public class UpdatedEventInfoDto : GeneralEventInformation

    {

        public int EventId { get; set; }

        // In the future make the organizer changeable? => public int OrganizerId { get; }

    }

}

### AppointmentDetails.cs

using System.Collections.Generic;

using System.Linq;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class AppointmentDetails

    {

        public AppointmentDetails(AppointmentInformation appointmentInformation, List<AppointmentParticipationInformation> participations)

        {

            AppointmentInformation = appointmentInformation;

            Participations = participations;

        }

        public static AppointmentDetails FromAppointment(Appointment appointment, int currentUserId, List<User> allParticipants)

        {

            IEnumerable<AppointmentParticipationInformation> withAnswers = appointment.AppointmentParticipations.Select(p => new AppointmentParticipationInformation(p.Participant.FullName, p.Participant.Id, p.AppointmentParticipationAnswer));

            IEnumerable<User> otherParticipants = allParticipants.Except(appointment.AppointmentParticipations.Select(a => a.Participant));

            IEnumerable<AppointmentParticipationInformation> noAnswers = otherParticipants.Select(p => new AppointmentParticipationInformation(p.FullName, p.Id, null));

            List<AppointmentParticipationInformation> participations = withAnswers.Concat(noAnswers).ToList();

            AppointmentInformation appointmentInformation = AppointmentInformation.FromAppointment(appointment, currentUserId, allParticipants.Count);

            return new AppointmentDetails(appointmentInformation, participations);

        }

        public AppointmentInformation AppointmentInformation { get; }

        public List<AppointmentParticipationInformation> Participations { get; }

    }

}

### AppointmentInformation.cs

using System;

using System.Linq;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class AppointmentInformation

    {

        public AppointmentInformation(int appointmentId, DateTime startTime, AppointmentParticipationAnswer? currentResponse, int acceptedParticipants, int declinedParticipants, int notAnsweredParticipants)

        {

            AppointmentId = appointmentId;

            StartTime = startTime;

            CurrentResponse = currentResponse;

            AcceptedParticipants = acceptedParticipants;

            DeclinedParticipants = declinedParticipants;

            NotAnsweredParticipants = notAnsweredParticipants;

        }

        public static AppointmentInformation FromAppointment(Appointment appointment, int currentUserId, int totalCount)

        {

            int acceptedCount = appointment.AppointmentParticipations.Count(a => a.AppointmentParticipationAnswer == AppointmentParticipationAnswer.Accepted);

            int declinedCount = appointment.AppointmentParticipations.Count(a => a.AppointmentParticipationAnswer == AppointmentParticipationAnswer.Declined);

            int noAnswerCount = totalCount - (acceptedCount + declinedCount);

            AppointmentParticipationAnswer? currentUserAnswer = appointment.AppointmentParticipations.FirstOrDefault(a => a.ParticipantId == currentUserId)?.AppointmentParticipationAnswer;

            var appointmentInformation = new AppointmentInformation(appointment.Id, appointment.StartTime, currentUserAnswer, acceptedCount, declinedCount, noAnswerCount);

            return appointmentInformation;

        }

        public int AppointmentId { get; }

        public DateTime StartTime { get; }

        public AppointmentParticipationAnswer? CurrentResponse { get; }

        public int AcceptedParticipants { get; }

        public int DeclinedParticipants { get; }

        public int NotAnsweredParticipants { get; }

    }

}

### AppointmentParticipationInformation.cs

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class AppointmentParticipationInformation

    {

        public AppointmentParticipationInformation(string participantName, int participantId, AppointmentParticipationAnswer? response)

        {

            ParticipantName = participantName;

            ParticipantId = participantId;

            Response = response;

        }

        public string ParticipantName { get; }

        public int ParticipantId { get; }

        public AppointmentParticipationAnswer? Response { get; }

    }

}

### EditEventDetails.cs

using System.Collections.Generic;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class EditEventDetails

    {

        public EditEventDetails(GeneralEventInformation information, List<AppointmentDetails> upcomingAppointments, List<EventParticipantInformation> participants)

        {

            Information = information;

            UpcomingAppointments = upcomingAppointments;

            Participants = participants;

        }

        public GeneralEventInformation Information { get; }

        public List<AppointmentDetails> UpcomingAppointments { get; }

        public List<EventParticipantInformation> Participants { get; }

    }

}

### EventDetails.cs

using System.Collections.Generic;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class EventDetails

    {

        public EventDetails(ViewEventInformation information, List<AppointmentDetails> upcomingAppointments, NotificationConfigurationResponse notificationConfigurationResponse)

        {

            Information = information;

            UpcomingAppointments = upcomingAppointments;

            NotificationConfigurationResponse = notificationConfigurationResponse;

        }

        public ViewEventInformation Information { get; }

        public List<AppointmentDetails> UpcomingAppointments { get; }

        public NotificationConfigurationResponse NotificationConfigurationResponse { get; }

    }

}

### EventOverview.cs

using System.Collections.Generic;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class EventOverview

    {

        public EventOverview(List<EventOverviewInformation> yourEvents, List<EventOverviewInformation> publicEvents)

        {

            YourEvents = yourEvents;

            PublicEvents = publicEvents;

        }

        /// <summary>

        /// Events you participate or organize

        /// </summary>

        public List<EventOverviewInformation> YourEvents { get; }

        /// <summary>

        /// Events you could participate without an invitation

        /// </summary>

        public List<EventOverviewInformation> PublicEvents { get; }

    }

}

### EventOverviewInformation.cs

using System;

using System.Linq;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class EventOverviewInformation

    {

        public EventOverviewInformation(int eventId, ViewEventInformation viewEventInformation, AppointmentInformation latestAppointmentInformation)

        {

            EventId = eventId;

            ViewEventInformation = viewEventInformation;

            LatestAppointmentInformation = latestAppointmentInformation;

        }

        public static EventOverviewInformation FromEvent(Event @event, int currentUserId)

        {

            ViewEventInformation viewEventInformation = ViewEventInformation.FromEvent(@event, currentUserId);

            Appointment firstUpcomingAppointment = @event.Appointments.OrderBy(a => a.StartTime).FirstOrDefault(a => a.StartTime >= DateTime.UtcNow);

            if (firstUpcomingAppointment == null)

            {

                return new EventOverviewInformation(@event.Id, viewEventInformation, null);

            }

            AppointmentInformation appointmentInformation = AppointmentInformation.FromAppointment(firstUpcomingAppointment, currentUserId, @event.EventParticipations.Count);

            return new EventOverviewInformation(@event.Id, viewEventInformation, appointmentInformation);

        }

        public int EventId { get; }

        public ViewEventInformation ViewEventInformation { get; }

        public AppointmentInformation LatestAppointmentInformation { get; }

    }

}

### EventParticipantInformation.cs

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class EventParticipantInformation

    {

        public EventParticipantInformation(int participantId, string participantName, string participantEmail)

        {

            ParticipantId = participantId;

            ParticipantName = participantName;

            ParticipantEmail = participantEmail;

        }

        public static EventParticipantInformation FromParticipation(EventParticipation participation)

        {

            return new EventParticipantInformation(participation.Participant.Id, participation.Participant.FullName, participation.Participant.Email);

        }

        public int ParticipantId { get; }

        public string ParticipantName { get; }

        public string ParticipantEmail { get; }

    }

}

### FrontendSession.cs

using System;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class FrontendSession

    {

        public FrontendSession(Guid token, int userId, string fullName, string email)

        {

            Token = token;

            UserId = userId;

            FullName = fullName;

            Email = email;

        }

        public Guid Token { get; }

        public int UserId { get; }

        public string FullName { get; }

        public string Email { get; }

    }

}

### NotificationConfigurationResponse.cs

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class NotificationConfigurationResponse : NotificationConfiguration

    {

        private NotificationConfigurationResponse(bool sendReminderEmail, bool sendSummaryEmail, bool sendLastMinuteChangesEmail)

        {

            SendReminderEmail = sendReminderEmail;

            SendSummaryEmail = sendSummaryEmail;

            SendLastMinuteChangesEmail = sendLastMinuteChangesEmail;

        }

        public static NotificationConfigurationResponse FromParticipation(EventParticipation participation)

        {

            return new NotificationConfigurationResponse(participation.SendReminderEmail, participation.SendSummaryEmail, participation.SendLastMinuteChangesEmail);

        }

    }

}

### ViewEventInformation.cs

using System.Linq;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels.ResponseTypes

{

    public class ViewEventInformation : GeneralEventInformation

    {

        public ViewEventInformation(int organizerId, string organizerName, string meetingPlace, string description, int totalParticipants, bool isPrivate, bool currentUserDoesParticipate, string title, int summaryTimeWindowInHours, int reminderTimeWindowInHours)

        {

            OrganizerId = organizerId;

            OrganizerName = organizerName;

            MeetingPlace = meetingPlace;

            Description = description;

            TotalParticipants = totalParticipants;

            IsPrivate = isPrivate;

            CurrentUserDoesParticipate = currentUserDoesParticipate;

            Title = title;

            SummaryTimeWindowInHours = summaryTimeWindowInHours;

            ReminderTimeWindowInHours = reminderTimeWindowInHours;

        }

        public static ViewEventInformation FromEvent(Event @event, int currentUserId)

        {

            bool currentUserDoesParticipate = @event.EventParticipations.Select(p => p.ParticipantId).Contains(currentUserId);

            return new ViewEventInformation(@event.OrganizerId, @event.Organizer.FullName, @event.MeetingPlace, @event.Description, @event.EventParticipations.Count, @event.IsPrivate, currentUserDoesParticipate, @event.Title, @event.SummaryTimeWindowInHours, @event.ReminderTimeWindowInHours);

        }

        public int OrganizerId { get; }

        public string OrganizerName { get; }

        public int TotalParticipants { get; }

        public bool CurrentUserDoesParticipate { get; }

    }

}

### GeneralEventInformation.cs

using System.ComponentModel.DataAnnotations;

using HeyImIn.Database.Models;

namespace HeyImIn.WebApplication.FrontendModels

{

    public class GeneralEventInformation

    {

        [Required]

        [MaxLength(FieldLengths.TitleMaxLength)]

        public string Title { get; set; }

        [Required]

        [MaxLength(FieldLengths.MeetingPlaceMaxLength)]

        public string MeetingPlace { get; set; }

        [Required]

        [MaxLength(FieldLengths.DescriptionMaxLength)]

        public string Description { get; set; }

        public bool IsPrivate { get; set; }

        [Range(0, FieldLengths.RealisticMaximumHours)]

        public int ReminderTimeWindowInHours { get; set; }

        [Range(0, FieldLengths.RealisticMaximumHours)]

        public int SummaryTimeWindowInHours { get; set; }

    }

}

### NotificationConfigurationDto.cs

namespace HeyImIn.WebApplication.FrontendModels

{

    public class NotificationConfiguration

    {

        public bool SendReminderEmail { get; set; }

        public bool SendSummaryEmail { get; set; }

        public bool SendLastMinuteChangesEmail { get; set; }

    }

}

### CurrentUserNotFoundException.cs

using System;

namespace HeyImIn.WebApplication.Helpers

{

    /// <summary>

    /// Exception used if the current user could not be found even though he should exist

    /// </summary>

    public class CurrentUserNotFoundException : Exception

    {

        public CurrentUserNotFoundException(int userId)

        {

            UserId = userId;

        }

        public int UserId { get; }

    }

}

### HttpActionExtensions.cs

using System;

using System.Net.Http;

using System.Reflection;

using System.Threading.Tasks;

using HeyImIn.Database.Context;

using HeyImIn.Database.Models;

using log4net;

namespace HeyImIn.WebApplication.Helpers

{

    public static class HttpActionExtensions

    {

        private const string UserIdPropertiesKey = "UserId";

        private const string SessionTokenPropertiesKey = "SessionToken";

        // Wrapper for SessionToken property

        public static Guid? TryGetSessionToken(this HttpRequestMessage requestMessage)

        {

            if (requestMessage.Properties.TryGetValue(SessionTokenPropertiesKey, out object sessionToken))

            {

                return sessionToken as Guid?;

            }

            return null;

        }

        public static Guid GetSessionToken(this HttpRequestMessage requestMessage)

        {

            return (Guid)requestMessage.Properties[SessionTokenPropertiesKey];

        }

        public static void SetSessionToken(this HttpRequestMessage requestMessage, Guid sessionToken)

        {

            requestMessage.Properties[SessionTokenPropertiesKey] = sessionToken;

        }

        // Wrapper for UserId property

        public static int? TryGetUserId(this HttpRequestMessage requestMessage)

        {

            if (requestMessage.Properties.TryGetValue(UserIdPropertiesKey, out object userId))

            {

                return userId as int?;

            }

            return null;

        }

        public static int GetUserId(this HttpRequestMessage requestMessage)

        {

            return (int)requestMessage.Properties[UserIdPropertiesKey];

        }

        public static void SetUserId(this HttpRequestMessage requestMessage, int userId)

        {

            requestMessage.Properties[UserIdPropertiesKey] = userId;

        }

        /// <summary>

        /// Loads the current user from the database using the provided <paramref name="context" />

        /// </summary>

        /// <param name="requestMessage">Request to get UserId from</param>

        /// <param name="context">Context to load from</param>

        /// <returns>Loaded user</returns>

        public static async Task<User> GetCurrentUserAsync(this HttpRequestMessage requestMessage, IDatabaseContext context)

        {

            int currentUserId = requestMessage.GetUserId();

            User currentUser = await context.Users.FindAsync(currentUserId);

            if (currentUser == null)

            {

                \_log.ErrorFormat("{0}(): Couldn't load the current user from the database", nameof(GetCurrentUserAsync));

                throw new CurrentUserNotFoundException(currentUserId);

            }

            return currentUser;

        }

        private static readonly ILog \_log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

    }

}

### LogHelpers.cs

using log4net;

namespace HeyImIn.WebApplication.Helpers

{

    public static class LogHelpers

    {

        /// <summary>

        /// Returns the unique audit log to be used for important messages

        /// </summary>

        /// <returns></returns>

        public static ILog GetAuditLog()

        {

            return LogManager.GetLogger(AuditLoggerName);

        }

        // Have to be in sync with log4net.config

        public const string UserIdLogKey = "UserId";

        public const string SessionTokenLogKey = "SessionToken";

        public const string LogFileDirectoryKey = "LogFileDirectory";

        private const string AuditLoggerName = "AuditLogger";

    }

}

### RequestStringMessages.cs

namespace HeyImIn.WebApplication.Helpers

{

    /// <summary>

    /// Constant strings which are returned as error messages to the Frontend

    /// </summary>

    public static class RequestStringMessages

    {

        public const string EventNotFound = "Der angegebene Event wurde nicht gefunden";

        public const string UserNotFound = "Der angegebene Benutzer wurde nicht gefunden";

        public const string AppointmentNotFound = "Der angegebene Termin wurde nicht gefunden";

        public const string UserAlreadyPartOfEvent = "Der angegebene Benutzer nimmt bereits dem Event teil";

        public const string UserNotPartOfEvent = "Der angegebene Benutzer nimmt nicht an dem Event teil";

        public const string InvitationRequired = "Der angegebene Event ist privat und benötigt eine Einladung";

        public const string OrganizorRequired = "Sie müssen Organisator dieses Event sein, um diese Aktion durchzuführen";

        public const string AppointmentsHaveToStartInTheFuture = "Termine müssen in der Zukunft liegen, überprüfen Sie ihre Eingabe";

        public const string CurrentPasswordWrong = "Das angegebene jetzige Passwort ist nicht korrekt";

        public const string NoProfileWithEmailFound = "Für diese E-Mail-Adresse ist kein Konto hinterlegt";

        public const string EmailAlreadyInUse = "Für diese E-Mail-Adresse ist bereits ein Konto hinterlegt";

        public const string ResetCodeInvalid = "Der angegebene Code ist ungültig";

        public const string ResetCodeAlreadyUsed = "Der angegebene Code ist abgelaufen oder wurde bereits verwendet";

        public const string InvitationInvalid = "Die Einladung ist ungültig";

        public const string InvitationAlreadyUsed = "Die Einladung wurde bereits verwendet";

    }

}