IPA Formatvorlage

autor : Pascal Honegger

version : 0.1

status : Draft

quelle : Atos

dokumentendatum : 31 Oktober 2017

anzahl der seiten : 16

Änderungshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Beschreibung | Autor |
| 0.1 | 31.10.2017 | Initiale Version erstellt | Pascal Honegger |
| 0.2 | 01.10.2017 | Zeitplan & Informieren abgeschlossen | Pascal Honegger |
| 0.3 | 02.11.2017 | Planen & Entscheiden abgeschlossen | Pascal Honegger |
| 0.4 | 08.11.2017 | Realisieren abgeschlossen | Pascal Honegger |
| 0.5 | 10.11.2017 | Kontrollieren & Auswerten abgeschlossen | Pascal Honegger |
| 1.0 | 10.11.2017 | Erster Release & Abgabe | Pascal Honegger |

# Inhaltsverzeichnis

**Teil 1**

1 Aufgabenstellung 4

1.1 Hintergrund 4

1.2 Funktionale Anforderungen 4

1.3 Technische Anforderungen 4

1.4 Erweiterungen 5

1.5 Wahlkriterien 5

2 Projektorganisation 6

2.1 Beteiligte Personen 6

2.2 Projektmanagementmethode 6

2.3 Backup-Konzept 6

3 Ausgangslage 7

3.1 Vorkenntnisse 7

3.2 Vorarbeit 7

3.3 Firmenstandards 7

4 Zeitplanung 8

4.1 Meilensteine 8

4.2 Verhalten im Falle von Verzug 8

4.3 Gantt-Diagramm 9

5 Arbeitsprotokoll 10

5.1 Tag 1 (01.11.2017) 10

5.2 Tag 2 (02.11.2017) 11

5.3 Tag 3 (03.11.2017) 13

5.4 Tag 4 (08.11.2017) 14

5.5 Tag 5 (10.11.2017) 15

**Teil 2**

6 Management Summary 15

6.1 Projektbeschreibung 15

6.2 System-Beschreibung 15

7 Informieren 16

7.1 Ist-Analyse 16

7.2 Soll-Analyse 16

8 Planen 17

8.1 Use-Cases 17

8.2 Testkonzept 23

8.3 Software-Architektur 28

8.4 Datenbankdesign 30

9 Entscheiden 32

9.1 Versionskontrolle 32

9.2 Backend-Framework 32

9.3 Frontend-Framework 32

9.4 Smiley-Umsetzung 32

9.5 Diagramm-Visualisierung 32

10 Realisieren 33

10.1 Backend 33

10.2 Frontend 33

11 Kontrollieren 34

11.1 Zeitmanagement 34

11.2 Funktionalität 34

11.3 Codequalität 34

11.4 Fertigstellung 34

12 Auswerten 35

12.1 Reflexion 35

12.2 Verbesserungen / Erfahrungen 35

12.3 Einsetzbarkeit 35

12.4 Fazit 35

**Anhang**

13 Verzeichnisse 36

13.1 Quellenverzeichnis 36

13.2 Abbildverzeichnis 36

13.3 Tabellenverzeichnis 36

14 Quellcode 37

# Aufgabenstellung

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

## Hintergrund

Der Projektleiter und C#-Teamleiter Christoph Fauti bemüht sich stets um die Zufriedenheit seiner Arbeitskollegen. Deshalb wird das AHM (Atos Happy Meter - Gefühlslage-Messer) als Website realisiert. Die Idee hierbei ist, dass ein Mitarbeiter schnell seine jetzige Gefühlslage festhalten kann und sieht, wie es den Arbeitskollegen geht.

## Funktionale Anforderungen

### Gefühlslage Erfassen

Das Ziel der Website liegt darin, dass ein Benutzer schnell und anonym seine Gefühlslage eintragen kann. Auf der Startseite sieht man eine Auswahl von Smileys (von sehr glücklich bis zu sehr unglücklich). Hier kann man basierend auf seiner jetzigen Gefühlslage einen Smiley drücken. Zusätzlich kann optional ein Kommentar verfasst werden, welcher die eigene Gefühlslage beschreibt. Nach dem Absenden seiner Gefühlslage wird der ausgewählte Smiley und optionale Kommentar zurückgesetzt. Dies erlaubt beispielsweise das Aufsetzen eines dedizierten Laptops im Pausenbereich, auf welchem jeder seine Gefühlslage eintragen kann, ohne die vorherige Auswahl zu sehen. Als ein simpler Spam-Schutz wird das erneute Absenden einer Gefühlslage für eine Minute blockiert.

### Gefühlslage auswerten

Durch herunterscrollen findet man unterhalb der Erfassung eine Übersicht der allgemeinen Gefühlslage. Man sieht die gleichen Smileys, welche man beim Erfassen auswählen kann. Die Grösse der Smileys variiert je nachdem, wie oft dieser an diesem Tag bereits ausgewählt wurde. Diese Daten werden automatisch aktualisiert, ohne dass ein Benutzer die Seite neu laden muss. Zusätzlich zu jedem Smiley wird ein zufälliger Kommentar angezeigt, falls einer für diesen Smiley an diesem Tag existiert.

### Administration

Zusätzlich zu dieser Funktionalität gibt es einen Administrator-Modus. Dieser ist durch ein Login geschützt und für die normalen Benutzer nicht sichtbar. Der Administrator bleibt während der Browser-Session angemeldet und kann sich bei Bedarf abmelden.

Der Administrator kann einen Smiley-Verlauf ansehen. Er kann einen Datumsbereich auswählen und sieht in Form eines Liniendiagramms an welchem Tag welche Smileys wie oft gewählt wurden.

Der Administrator kann die verfügbaren Smileys verwalten. Man kann neue hinzufügen, bestehende bearbeiten und ungebrauchte deaktivieren. Durch das deaktivieren können keine neuen Gefühlslagen mit diesem Smiley erstellt werden, die Auswertungen bleiben trotzdem in der Administration erhalten.

Der Administrator kann seine eigenen Anmeldedaten (Benutzername & Passwort) bearbeiten.

## Technische Anforderungen

* Frontend als Website
  + Unterstützt werden folgende Browser
    - Muss: Google Chrome
    - Muss: Mozilla Firefox
    - Optional: Microsoft Internet Explorer 11
  + Läuft unter Windows 7
* Backend-Server
  + Windows-basierter Server
  + Datenspeicherung in einer relationalen Datenbank
* Admin Login
  + Passwort sicher gespeichert (nicht Klartext)
  + Standard Benutzer ‘admin’ mit dem Passwort ‘admin’

## Erweiterungen

Die Installation auf dem Testsystem ist nicht teil der IPA. Falls die Rückmeldungen nach den ersten Tests positive ausfallen wird das System um eine geografische Gruppierung (Bsp. Stockwerke, Gebäude) erweitert. Diese Funktionalität sollte beim Erstellen der Datenstruktur berücksichtigt werden. Während der Testphase kann dieser Ort für alle Einträge auf einen Test-Wert gesetzt werden.

## Wahlkriterien

* 125 Redundanzfreie Gliederung des Programms
* 130 Vollständiges ERM
* 164 Error-Handling
* 166 Lesbarer Code
* 225 Versionsverwaltung
* 235 Use Cases & Activity Diagram korrektes UML
* 250 Schichtentrennung (MVC Backend | MVVM Frontend)

# Projektorganisation

## Beteiligte Personen

|  |  |  |  |
| --- | --- | --- | --- |
| Position | Name | E-Mail | Telefonnummer |
| Auftraggeber | Christoph Fauti | [christoph.fauti@atos.net](mailto:christoph.fauti@atos.net) | - |
| Fachexperte | Rolf Fux | [rolf.fux@atos.net](mailto:rolf.fux@atos.net) | - |
| Kandidat | Pascal Honegger | [pascal.honegger@siemens.com](mailto:pascal.honegger@siemens.com) | 077 421 51 28 |
| Hauptexperte | Remo Steinmann | [remo.steinmann@siemens.com](mailto:remo.steinmann@siemens.com) | - |
| Zweit-Experte | - | - | - |

Tabelle 1: Projektteilnehmer

## Projektmanagementmethode

Ich verwende die Projektmanagementmethode IPERKA. Diese Wasserfallmethode hat den Vorteil, dass die Struktur gut mit den Vorgaben der IPA übereinstimmen. Als Alternative wurde Scrum verworfen, da der Overhead mit den formalen Scrum-Meetings zu gross wäre.

## Backup-Konzept

Um die Chance eines Datenverlusts zu minimieren sind regelmässige Backups essenziell. Es werden regelmässig Commits erstellt und online auf einem privaten GitHub-Repository gesichert. Im Falle eines Laptopfehlers kann eine relative aktuelle Version geladen werden. Dies wird sowohl für Word-Dokumente wie Quellcode gemacht. Zusätzlich wird täglich ein Backup auf einen USB-Stick geladen, für den Fall, dass GitHub und der Laptop zur gleichen Zeit nicht mehr erreichbar sind.

# Ausgangslage

Beschreibung Umfeld / Projekt.

## Vorkenntnisse

* Vertieftes C#-Wissen dank zwei Jahren Projekterfahrung
* Verständnis von Angular (2+) dank einem Jahr Projekterfahrung
* Erfahrungen in der Webentwicklung mit HTML, CSS, JavaScript unter Verwendung von Bootstrap und dem Google Material Design
* Erfahrungen mit Java und Python

## Vorarbeit

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

## Firmenstandards

### OR-Mapper

Ich setze das Entity Framework 6 von Microsoft ein. Dieser direkt in die DIE integrierte OR-Mapper ist Atos-Firmenstandard für C#-basierte Projekte.

### Frontend

Für das Frontend wird das gleiche Framework (Angular 4.4.6) wie im Produktivprojekt verwendet. Ausserdem werden die selben DIE-Konfigurationen verwendet, damit der korrekte Coding-Style eingehalten wird.

# Zeitplanung

## Meilensteine

|  |  |  |
| --- | --- | --- |
| Meilenstein | Beschreibung | Datum |
| Projektstart | Start des Projektes | 01.11.2017 09:00 |
| Ende Informieren | Informieren Phase abgeschlossen | 01.11.2017 11:00 |
| Ende Planen | Planung abgeschlossen | 02.11.2017 16:00 |
| Ende Entscheiden | Entscheiden Phase abgeschlossen | 02.11.2017 17:00 |
| Ende Realisieren | Realisierung abgeschlossen | 08.11.2017 17:00 |
| Ende Kontrollieren | Kontrolle durch Tests abgeschlossen | 10.11.2017 11:00 |
| Projektabschluss | Erfolgreicher Projektabschluss | 10.11.2017 17:00 |

Tabelle 2: Meilensteine

## Verhalten im Falle von Verzug

Falls Meilensteine nicht erreicht werden können müssen Aktionen getroffen werden. Primär werden nicht kritische Teilaufgaben vernachlässigt. Die Teilaufgaben, welche vernachlässigt werden sind das Administrator-Design, Emojis und Ort-Erweiterungen in dieser Reihenfolge.

## Gantt-Diagramm



Abbildung 1: Zeitplan

# Arbeitsprotokoll

## Tag 1 (01.11.2017)

### Planung

Heute möchte ich mit dem Projekt beginnen und die erste IPERKA-Phase abschliessen. Zusätzlich wird mit der Planung begonnen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Projektbeginn | 15min | 15min | ✔ |
| Zeitplanung | 1h 45min | 2h | ✔ |
| Projektorganisation dokumentieren | 30min | 15min | ✔ |
| Ausgangslage dokumentieren | 30min | 15min | ✔ |
| Ist- und Soll-Analyse | 30min | 30min | ✔ |
| Use-Cases | 1h 30min | 2h 30min | ✔ |
| Software-Architektur | 30min | 10min | ✘ |
| Arbeitsprotokoll | 30min | 20min | ✔ |

Tabelle 3: Konkrete Planung Tag 1

### Getane Arbeit

Heute war der erste Tag des Projektes. Nach dem Ankommen begann die Arbeit flott und ohne Probleme. Beim Erstellen des Zeitplans habe ich festgestellt, dass die total fünf Tage à sechs Stunden knapp bemessen sind. Der Grossteil der Dokumentation muss morgen fertig gestellt werden, damit ich am Freitag und nächsten Mittwoch die Implementation abschliessen kann. Danach habe ich lediglich einen Tag für das Testen und Finalisieren des Projekts, solange nichts schiefläuft. Im Falle unerwarteter Probleme habe ich keinen wirklichen Platz für einen Puffer und werden wohl an anderen Funktionen kürzen müssen, damit das Projekt abgeschlossen wird. Also hoffe ich, dass es dazu nicht kommen wird.

### Probleme

Nach der ersten Schnell-Analyse des Auftrages bin ich davon ausgegangen, dass ich lediglich ~5 Use-Cases haben werden. Nach der Umsetzung hat sich jedoch gezeigt, dass ich ganze 11 Use-Cases zu beschreiben habe. Deshalb habe ich auch etwas länger als die initial geplanten 2 Stunden benötigt, um diese auszuformulieren.

### Reflexion

Meine Stimmung wurde direkt zu Beginn zerschmettert. Beim Erstellen des Zeitplans wurde mir bewusst, wie viel Arbeit noch vor mir liegt. Zum Glück ist diese Angst mit jeder Stunde geschrumpft, da ich gute Fortschritte machte. Allgemein bin ich zufrieden mit diesem Projektstart und habe zum ersten Mal in einem Projekt nicht das Gefühl, die Planung zu vernachlässigen.

## Tag 2 (02.11.2017)

### Planung

Heute möchte ich die zwei IPERKA-Phasen Planen und Entscheiden abschliessen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Software-Architektur – Komponentendiagramm | 30min | 40min | ✔ |
| Software-Architektur – Aktivitätsdiagramm | 30min | 40min | ✔ |
| Software-Architektur – Beschreibung | 15min | 20min | ✔ |
| ERM – Schema in Workbench erstellen | 45min | 1h | ✔ |
| ERM – Sinnvolle Feld-Typen definieren und dokumentieren | 45min | 40min | ✔ |
| Testfälle erstellen | 2h 15min | 2h 40min | ✔ |
| Entscheidungen dokumentieren | 30min | 30min | ✔ |
| Expertenbesuch | - | 15min | ✔ |
| Emoji-Recherche | 0min | 10min | **~** |
| Arbeitsprotokoll | 30min | 30min | ✔ |

Tabelle 4: Konkrete Planung Tag 2

### Getane Arbeit

Heute habe ich die beiden IPERKA-Phasen Planen und Entscheiden abgeschlossen. Als Teil der Planung habe ich zwei UML-Diagramme mit PlantUML erstellt. Danach habe ich das ERM realisiert und dokumentiert und die Testfälle beschrieben. Der Zeitaufwand beim ERM und den Testfällen war grösser als erwartet. Da wir heute anstatt der geplanten sechs Stunden mehr Zeit zum Arbeitete hatten, bin ich immer noch im Zeitplan.

### Expertenbesuch

Heute wurde der erste Expertenbesuch durchgeführt. Dabei habe ich meine Dokumentation vorgestellt, da ich noch nicht mit der Realisation angefangen habe. Es gab glücklicherweise keine grösseren Probleme, ausser dem knapp bemessenen Zeitplan.

### Probleme

Wie erwartet gab es Probleme mit den Smileys. Diese wollte ich als Charakter in der Datenbank speichern, wobei hier die Kollation geändert werden muss. Nach weiteren Fehlern habe ich versucht via C# Emojis in die Datenbank zu speichern, in der Hoffnung, dass der bisherige Fehler in der MySQL-Workbench liegt. Dabei stiess ich auf einen weiteren Fehler, welcher nach Recherche auf einen MySQL-Fehler zurückzuführen ist. Als Lösung konnte ich schlussendlich zwei Befehle ausführen, welche ich in folgendem Stack Overflow gefunden haben: <https://stackoverflow.com/questions/39071021/vs-error-the-value-for-column-isprimarykey-in-table-tabledetails-is-dbnull>. Leider hat das Speichern von Emojis wie C# auch nicht funktioniert. Schlussendlich habe ich mich dazu entschieden einen Workaround zu benutzen und den HEX-Code der Emoji-Zeichen in der Datenbank zu speichern. Diese sind nach Unicode bis zu sechs Zeichen lange, weshalb ich das Datenbankfeld 6 Zeichen langgemacht habe. Die Limitierung dieser Lösung ist jedoch, dass gewisse «komplexe» Emojis nicht verwendbar sind. Komplexe Emojis sind hier zusammengebaute Emojis. Beispielsweise kann ich einen gelben (farbneutralen) Emoji mit einer Hautfarbe kombinieren. Diese komplexen Fälle habe ich aus Zeitgründen ignoriert, da diese Spezialfälle nicht verlangt sind.

Im Verlaufe vom Tag sind immer mehr Use-Cases und Testfälle aufgefallen, welche ich nicht bedacht habe. So ist die ohnehin grosse Zahl an Use-Cases noch grösser geworden.

Wir mussten unseren Zeitplan als PDF abgeben, was mir Schwierigkeiten bereitet hat. So habe ich es nicht geschafft den gesamten Zeitplan auf eine Seite zu laden. Meine Abgabe beinhaltet nun leider ein PDF mit zwei Seiten, welches nicht sehr anschaulich ist.

### Reflexion

Heute habe ich viel Zeit mit vergolden verschwendet, was ich auf jeden Fall bereue. Ich habe naiv versucht die Emojis als einen Charakter in der Datenbank abzuspeichern. Dazu kamen zuerst Probleme mit der Kollation, gefolgt von zusammengesetzten Emojis und der Zeichenlänge, welche variieren kann. Ich habe erst verschwenden von Zeit und Nerven meine Idee an den Nagel gehängt und einfach die Hexadezimale Darstellung eines Emojis in die Datenbank gespeichert. Zusammengesetzte / Komplexe Emojis funktionieren so zwar nicht, sind aber auch keine Anforderung. Im Allgemeinen hätte ich vor dem Beginn der Recherche meiner ersten Idee weitere Ansätze studieren müssen. Stattdessen bin ich davon ausgegangen, dass meine Lösung die korrekte ist, und habe unnötig Zeit verloren.

Noch gestern hatte ich das Gefühl, nichts vergessen zu haben und genügend Zeit in die Planung zu investieren. Heute ist mir aufgefallen, dass ich, trotz der vielen Überlegungen während der Planung, Anwendungsfälle und Testfälle vergessen habe. Ich hoffe, dass keine weiteren Überraschungen in der Implementation auftauchen. Hätten wir heute nur die geplanten sechs Stunden arbeiten können, wäre ich bereits hinter dem Zeitplan. Falls ich also bei der Implementation auf weitere Probleme stosse, wird der erfolgreiche Abschluss des Projekts gefährdet.

## Tag 3 (03.11.2017)

### Planung

Heute möchte ich mit der Realisierung beginnen, die Projekte aufsetzen und das Backend fertigstellen.

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Backend-Projekt aufsetzen | 15min | 15min | ✔ |
| Frontend-Projekt aufsetzen | 15min | 15min | ✔ |
| Datenbankanbindung | 1h | 1h 15min | ✔ |
| Backend Schnittstellen | 1h 45min | 2h | ✔ |
| Backend Autorisierung | 30min | 30min | ✔ |
| Frontend Serveranbindung | 45min | 1h 15min | ✔ |
| Frontend Emoji-Handling | 45min | 10min | ✘ |
| Arbeitsprotokoll | 45min | 20min | ✔ |
| MySQL-Probleme beheben | 0min | 1h | ✔ |

Tabelle 5: Konkrete Planung Tag 3

### Probleme

Heute begann endlich die Implementierungs- / Realisierungs-Phase. Direkt zu streikte jedoch Visual Studio 2017, meine C#-IDE. Hierbei stürzte die Benutzeroberfläche des OR-Mappers ohne Kommentar ab, sobald man Code generieren wollte. Aus diesem Grund musste ich den generierten Code von Hand schreiben, was mich unnötig Zeit gekostet hat. Zusätzlich tauchten noch MySQL-Verbindungsfehler auf. Nach Internet-Recherche habe ich herausgefunden, dass dies auf die aktuelle Version von .Net/MySQL-Connector zurückzuführen ist. Nach der Installation einer älteren Version gab es bei der Interaktion zwischen C# und MySQL keine weiteren Probleme.

### Reflexion

Nach den initialen Startschwierigkeiten ging die Arbeit schnell voran. Bei der Implementation des Servers hatte ich das Gefühl, ich würde einen menschlichen Generator nachahmen. Auch beim Schreiben der Frontend-Backend-Kommunikation mussten viele DTOs einmal in C# und einmal in TypeScript definieren. Der sichtbare Fortschritt meiner Implementation ist leider noch nicht sichtbar, was frustrierend ist. Dafür sollten jetzt auf Server-Seite alle Spezialitäten wie Autorisierung und Validierung abgeschlossen sein. Auch im Frontend sind alle Daten bereit geladen zu werden, lediglich die Benutzerdarstellung fehlt. Ich hoffe stark, dass die Umsetzung der Darstellung schnell von Statten geht, da ich sonst garantiert Funktionalität streichen muss.

## Tag 4 (08.11.2017)

### Planung

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Frontend Serveranbindung korrigieren | 15min | 10min | ✔ |
| Frontend Gefühlslage darstellen | 30min | 30min | ✔ |
| Frontend Gefühlslage erfassen | 45min | 45min | ✔ |
| Frontend Autorisierung | 15min | 15min | ✔ |
| Frontend Login | 45min | 15min | ✔ |
| Frontend Administration | 2h | 2h 30min | ✔ |
| Klassendokumentation Backend | 30min | 0min | ✘ |
| Arbeitsjournal | 45min | 30min | ✔ |
| Expertenbesuch | - | 15min | ✔ |

Tabelle 6: Konkrete Planung Tag 4

### Probleme

Direkt zu Beginn musste ich mit Schrecken feststellen, dass die Kommunikation zwischen meine Frontend und Backend doch noch nicht ganz reibungslos verlief. Grund dafür das das falsche JSON-Format. Im Backend wurden die Properties bei der Serialisierung in PascalCase übernommen. Das Frontend jedoch erwartet, dass diese in camelCase ankommen. Nach mühseliger Recherche habe ich herausgefunden, dass das verwenden der «JSON()» Methode das falsche Format anwendet, und ich stattdessen einfach «OK()» verwenden muss.

Auch im Frontend gab es kleine Implementierungsprobleme. So habe ich für eine Angular-Binding anstelle der Banane in der Box «[()]» nur eine Box «[]» verwendet. Bis ich diesen trivialen Fehler gefunden habe kam ich an den Rand der Verzweiflung.

### Expertenbesuch

Heute hat der zweite Expertenbesuch stattgefunden. Da ich mit der Applikation im Grossen und Ganzen fertig bin, konnte ich bereits viel Funktionalität zeigen. Dabei habe ich beinahe zu viel gezeigt, weshalb der Expertenbesuch zu einer halben Demonstration wurde. Als Feedback gab es unter anderem, dass ich den Expertenbesuch in dem Projektjournal vernachlässigt habe.

### Reflexion

Aufgrund des Zeitdrucks habe ich heute eine zusätzliche Stunde Arbeit am Abend investiert, damit die Implementation fertig wurde. Die Dokumentation ist leider noch nicht so weit, wie ich sie gerne hätte. Da morgen der letzte Tag ist gibt es nicht kein Zeitpolster mer. Damit die Dokumentation auf jeden Fall abgeschlossen wird verzichte ich bei der Realisierungs-Dokumentation vorerst auf ein Klassendiagramm und beschränke mich auf die Klassenbeschreibungen. Falls ich am Ende doch genügend Zeit habe, ergänze ich das Diagramm.

## Tag 5 (10.11.2017)

### Planung

|  |  |  |  |
| --- | --- | --- | --- |
| Was | Soll-Dauer | Ist-Dauer | Fertig |
| Code abschliessen | 15min | 30min |  |
| Bestehende Dokumentation fertigstellen | 15min | 15min |  |
| Datenbank Export mit Standardbenutzer | 15min | 10min |  |
| Realisierung dokumentieren | 1h 15min | 2h 15min |  |
| Testfälle durchführen | 1h | 1h |  |
| Testfazit | 15min | 15min |  |
| Projektauswertung | 45min |  |  |
| Arbeitsprotokoll | 30min |  |  |
| Programmcode & Anhang vorbereiten | 30min |  |  |
| Projektabgabe | 15min |  |  |

### Probleme

Testfälle teilweise doppelt vorhanden oder nicht gut beschrieben, Formulierung leicht angepasst.

### Reflexion

# Kurzfassung

In diesem Kapitel folgt eine Kurzzusammenfassung über das Projekt.

## Projektbeschreibung

Der Atos Teamleiter der C#-Projekte Christoph Fauti möchte die Gefühlslage der Mitarbeiter besser verstehen. Momentan gibt es noch keine Möglichkeit als Mitarbeiter seinen Gemütszustand festzuhalten. Für diesen Zweck wird eine Website entwickelt, welche das einfache und anonyme Erfassen seiner Gefühlslage anhand von Smileys erlaubt. Optional können die Mitarbeiter auch einen Kommentar erfassen. Jeder Mitarbeiter kann sehen, wie es den Anderen heute geht. Alle verfügbaren Gefühlslageoptionen können durch einen Administrator verwaltet werden. So können je nach Jahreszeit gewisse Smiley aktiviert / deaktiviert werden. Die Auswahl der Smiley beruht auf Unicode-Emojis.

## System-Beschreibung

Das Frontend ist mit Angular 4 und Angular Material 2 umgesetzt. Angular Material erlaubt das Material Design von Google einfach im Web umzusetzen, ähnlich wie Bootstrap. Weiteres Styling ist mit CSS und Flexbox realisiert. Die Logik ist in TypeScript umgesetzt und bietet gegenüber JavaScript Typsicherheit. Der Frontend und Backend kommunizieren über REST miteinander. Das ASP.NET WebAPI 2 Backend akzeptiert und antwortet als RESTFul Webservice in JSON Objekte. Der Server kapselt die MySQL Datenbank und validiert Benutzereingaben und Berechtigungen.

# Informieren

## Ist-Analyse

Eine Analyse des (noch nicht) vorhandenen Systems.

Momentan bietet die Atos kein direktes Tool an, mit welchem die Mitarbeiter ihre Gefühlslage veröffentlichen können. Möchte man eine Beschwerde oder einen Kommentar zu etwas abgeben, muss man eine betroffene Person direkt kontaktieren. Gefällt einem jedoch das warme Wetter nicht, gibt es keine direkte Ansprechperson. Auch fällt es einem Mitarbeiter schwer Kritik an seinem Chef oder Projekt auszuüben, da dies nicht anonym ist.

## Soll-Analyse

Eine genauere Analyse der Anforderungen und des Soll-Zustands

### Frontend

Das Web-Frontend wird mit Angular (2+) umgesetzt und funktioniert mindestens im Firefox und Chome unter Windows 7. Auf der Startseite sieht man eine Auswahl der verschiedenen Emotionen in Form von Smileys. Je öfter ein solcher Smiley an diesem Tag bereits gedrückt wurde, desto Grösser erscheint dieser. Zufälligerweise werden zu den Smileys Kommentare angezeigt, falls es welche der letzten 24 Stunden gibt. Der Benutzer, normalerweise ein Atos-Angestellter, wählt einen zu seiner Gefühlslage passenden Smiley aus und kann diese Emotion absenden. Optional gibt es unter den Smileys ein Textfeld, in welchem ein bis zu 250 Zeichen langer Kommentar verfasst werden kann. Nach dem Absenden wird die Seite direkt aktualisiert.

Zusätzlich kann der Administrator über die URL auf eine Administrator-Ansicht wechseln. Diese ist mit einem Passwort geschützt und erlaubt das verwalten der verfügbaren Smileys (CRUD-Operationen) und einer grafischen Historie der Gefühlslagen.

### Backend

Das Backend ist in C# geschriebener REST-Service. Dieser Service leitet grösstenteils gefilterte Daten aus der Datenbank an das Frontend. Zusätzlich gibt es die Authentifizierung, welche sicherstellt, dass nur Administratoren auf den gesicherten Bereich zugreifen können.

### Datenbank

Die Datenbank ist in MySQL erstellt. Der Zugriff auf diese Erfolgt über das Microsoft Entity Framework 6, der offizielle OR-Mapper von Microsoft. Alle Aufrufe an die Datenbank über diesen OR-Mapper sind transaktionell gesichert.

# Planen

## Use-Cases

### Use-Case Diagramm

Alle Use-Cases des Atos-Happy-Meters. Der Administrator teilt alle Use-Cases mit einem normalen Benutzer und erweitert diesen. Die Use-Cases sind in die drei Hauptteile der Applikation «Administration», «Anmeldung» und «Startseite» getrennt.

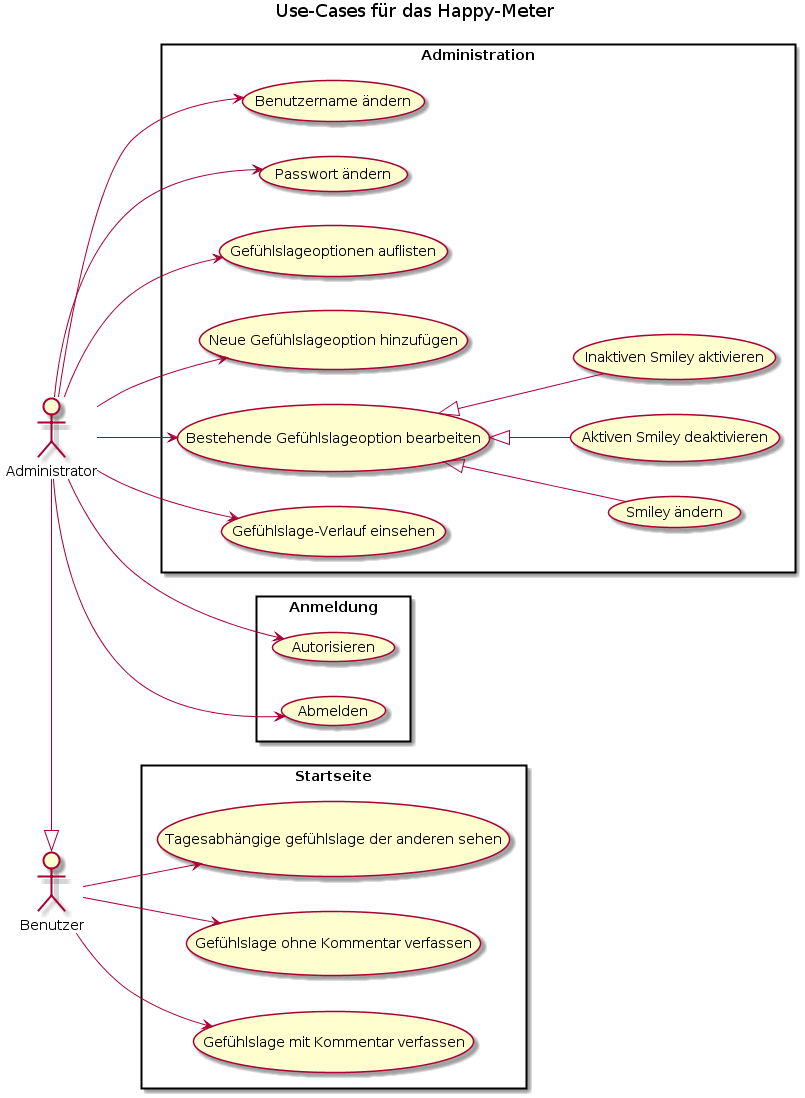


Abbildung 2: Use-Case Diagramm

### Use-Cases 1: Startseite

Alle Use-Cases in diesem Kapitel werden von einem Benutzer oder autorisierten Administrator durchgeführt.

|  |  |
| --- | --- |
| Use-Case 1.1: Tagesabhängige Gefühlslage sehen | |
| Beschreibung | Ein Benutzer kann die allgemeine Gefühlslage sehen. Dadurch kann er einschätze, ob die Stimmung positiv oder negativ ist. Zusätzlich werden die Kommentare anderer Benutzer dargestellt. |
| Ergebnis / Nachbedingung | Die allgemeine Gefühlslage für diesen Tag wird dargestellt |
| Vorbedingung | Mehrere Gefühlslagen sind erfasst, darunter mindestens eine mit einem Kommentar |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Startseite öffnen | |

Tabelle 7: Use-Case 1.1

|  |  |
| --- | --- |
| Use-Case 1.2: Gefühlslage ohne Kommentar erfassen | |
| Beschreibung | Ein Benutzer kann seine jetzige Gefühlslage durch das Auswählen eines Smileys erfassen |
| Ergebnis / Nachbedingung | Die eingetragene Gefühlslage verändert die allgemeine Gefühlslage |
| Vorbedingung | Keine Gefühlslagen sind erfasst, mindestens zwei Gefühlslageoptionen existieren |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Startseite öffnen | | 2 | Einen Smiley auswählen | | 3 | Auf Senden drücken | | 4 | Der ausgewählte Smiley wird in der Übersicht grösser dargestellt | |

Tabelle 8: Use-Case 1.2

|  |  |
| --- | --- |
| Use-Case 1.3: Gefühlslage mit Kommentar erfassen | |
| Beschreibung | Ein Benutzer kann seine jetzige Gefühlslage durch das Auswählen eines Smileys erfassen und zusätzlich einen beschreibenden Kommentar verfassen |
| Ergebnis / Nachbedingung | Die eingetragene Gefühlslage verändert die allgemeine Gefühlslage und der eingetragene Kommentar wird angezeigt |
| Vorbedingung | Keine Gefühlslagen sind erfasst, mindestens zwei Gefühlslageoptionen existieren |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Startseite öffnen | | 2 | Einen Smiley auswählen | | 3 | In das Kommentar-Feld «Mir gefallen diese Smileys» eingeben | | 4 | Auf Senden drücken | | 5 | Der ausgewählte Smiley wird in der Übersicht grösser dargestellt | | 6 | Zusätzlich zum ausgewählten Smiley wird der eingegebene Kommentar «Mir gefallen diese Smileys» angezeigt | |

Tabelle 9: Use-Case 1.3

### Use-Cases 2: Anmeldung

Alle Use-Cases in diesem Kapitel werden von einem noch nicht autorisierten Administrator durchgeführt.

|  |  |
| --- | --- |
| Use-Case 2.1: Anmelden (Direkt) | |
| Beschreibung | Ein Administrator muss sich zuerst als solcher autorisieren, bevor er auf die Administration zugreifen darf. |
| Trigger | Ein noch nicht autorisierter Administrator möchte auf die Administration zugreifen |
| Ergebnis / Nachbedingung | Der Administrator wird auf die Administration weitergeleitet |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Login-Seite öffnen | | 2 | Benutzername «admin» und Passwort «admin» eingeben | | 3 | Knopf «Anmelden» drücken | | 4 | Weiterleitung auf Administrations-Seite | |

Tabelle 10: Use-Case 2.1

|  |  |
| --- | --- |
| Use-Case 2.2: Anmelden (Indirekt) | |
| Beschreibung | Ein Administrator muss sich zuerst als solcher autorisieren, bevor er auf die Administration zugreifen darf. |
| Trigger | Ein noch nicht autorisierter Administrator möchte auf die Administration zugreifen |
| Ergebnis / Nachbedingung | Der Administrator wird auf die Administration weitergeleitet |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | Weiterleitung auf Login-Seite | | 3 | Benutzername «admin» und Passwort «admin» eingeben | | 4 | Knopf Anmelden drücken | | 5 | Weiterleitung auf Administrations-Seite | |

Tabelle 11: Use-Case 2.2

|  |  |
| --- | --- |
| Use-Case 2.3: Abmelden | |
| Beschreibung | Ein Administrator kann sich nach dem Anmelden wieder abmelden |
| Trigger | Ein autorisierter Administrator möchte seine Sitzung beenden |
| Ergebnis / Nachbedingung | Der Administrator wird auf die Startseite weitergeleitet |
| Vorbedingung | Use-Case 2.1 oder 2.2 |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Auf den Knopf «Abmelden» drücken | |

Tabelle 12: Use-Case 2.3

### Use-Cases 3: Administration

Alle Use-Cases in diesem Kapitel werden von einem autorisierten Administrator durchgeführt.

|  |  |
| --- | --- |
| Use-Case 3.1: Gefühlslageoptionen auflisten | |
| Beschreibung | Ein Administrator kann alle (auch deaktivierte) Gefühlslageoptionen einsehen |
| Ergebnis / Nachbedingung | Der Administrator sieht eine Auflistung aller Gefühlslageoptionen |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | |

Tabelle 13: Use-Case 3.1

|  |  |
| --- | --- |
| Use-Case 3.2: Neue Gefühlslageoption hinzufügen | |
| Beschreibung | Ein Administrator kann eine neue Gefühlslageoption hinzufügen |
| Ergebnis / Nachbedingung | Die hinzugefügte Gefühlslageoption ist auf der Startseite und in der Administration sichtbar |
| Vorbedingung | Eine Gefühlslage mit dem Smiley «🎃» existiert noch nicht |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | Auf «Neue Gefühlslageoption hinzufügen» drücken | | 3 | Im Dialog den Smiley «🎃» eingeben | | 4 | Dialog bestätigen | |

Tabelle 14: Use-Case 3.2

|  |  |
| --- | --- |
| Use-Case 3.3: Deaktivierte Gefühlslageoption aktivieren | |
| Beschreibung | Eine deaktivierte Gefühlslageoption wird den Benutzern nicht angezeigt. Der Administrator kann diese Gefühlslage wieder verfügbar machen, indem er diese aktiviert |
| Ergebnis / Nachbedingung | Die neu deaktivierte Gefühlslageoption ist auf der Startseite nicht mehr sichtbar |
| Vorbedingung | Die Gefühlslage mit dem Smiley «🎃» existiert und ist aktiviert |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | In der Übersicht die Gefühlslageoption «🎃» suchen | | 3 | Bei dieser Gefühlslageoption den Knopf «Aktivieren» drücken | |

Tabelle 15: Use-Case 3.3

|  |  |
| --- | --- |
| Use-Case 3.4: Aktivierte Gefühlslageoption deaktivieren | |
| Beschreibung | Eine aktivierte Gefühlslageoption wird den Benutzern angezeigt. Der Administrator kann diese Gefühlslage nicht mehr verfügbar machen, indem er diese deaktiviert |
| Ergebnis / Nachbedingung | Die neu aktivierte Gefühlslageoption ist auf der Startseite sichtbar |
| Vorbedingung | Die Gefühlslage mit dem Smiley «🎃» existiert und ist deaktiviert |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | In der Übersicht die Gefühlslageoption «🎃» suchen | | 3 | Bei dieser Gefühlslageoption den Knopf «Deaktivieren» drücken | |

Tabelle 16: Use-Case 3.4

|  |  |
| --- | --- |
| Use-Case 3.5: Gefühlslage-Verlauf einsehen | |
| Beschreibung | Als Administrator kann ich die Gefühlslage zwischen zwei Daten sehen |
| Ergebnis / Nachbedingung | In Form eines Liniendiagramms wird die tägliche Gefühlslage dargestellt. |
| Vorbedingung | Die Gefühlslage mit dem Smiley «🎃» existiert und hat Gefühlslagen für den jetzigen und vorherigen Tag |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | |

Tabelle 17: Use-Case 3.5

|  |  |
| --- | --- |
| Use-Case 3.6: Benutzername ändern | |
| Beschreibung | Als Administrator kann man seinen Benutzernamen ändern |
| Ergebnis / Nachbedingung | Anmeldung mit dem neuen Benutzernamen ist möglich |
| Vorbedingung | Benutzer «admin2» existiert nicht |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | Benutzername ändern drücken | | 3 | Im Dialog den neuen Benutzernamen «admin» eingeben | | 4 | Benutzername ändern anwählen | |

Tabelle 18: Use-Case 3.6

|  |  |
| --- | --- |
| Use-Case 3.7: Passwort ändern | |
| Beschreibung | Als Administrator kann man sein Passwort ändern |
| Ergebnis / Nachbedingung | Anmeldung mit dem neuen Passwort ist möglich |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | Passwort ändern drücken | | 3 | Im Dialog für die Felder Passwort und Passwort wiederholen das gleiche neue Passwort eingeben | | 4 | Passwort ändern anwählen | |

Tabelle 19: Use-Case 3.7

|  |  |
| --- | --- |
| Use-Case 3.8: Gefühlslage-Smiley ändern | |
| Beschreibung | Als Administrator kann man im Nachhinein den Smiley einer Gefühlslage ändern |
| Ergebnis / Nachbedingung | Der neue Smiley wird für die Gefühlslage angezeigt |
| Vorbedingung | Die Gefühlslage mit dem Smiley «🎃» existiert, die Gefühlslage mit dem Smiley «🎅» existiert nicht |
| Ablauf | |  |  | | --- | --- | | Schritt | Aktion | | 1 | Administrations-Seite öffnen | | 2 | In der Übersicht die Gefühlslageoption «🎃» suchen | | 3 | Bei dieser Gefühlslageoption den Knopf «Smiley ändern» drücken | | 4 | Im Dialog den neuen Smiley «🎅» eingeben | | 5 | Dialog bestätigen | |

Tabelle 20: Use-Case 3.8

## Testkonzept

Das Testkonzept beschreibt wie nach der Realisierung sichergestellt wird, dass die Funktionalität funktioniert. Dazu werden Testfälle definiert und später in der Kontrollieren-Phase durchgeführt.

### Eingesetzte Testmittel und -Methoden

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

* Dell-Laptop mit 16Gb Arbeitsspeicher, einem Intel i7 Prozessor und fungierender Internetverbindung
* Google Chrome 62.0.3202.75
* Windows 7 (Systemsprache: Deutsch)

### Testfälle 1: Startseite

Diese Tests können von Benutzern und Administratoren durchgeführt werden.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 1.1 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃  🎅 x 2 Gefühlslagen  🎃 x 1 Gefühlslagen | |  |  | | --- | --- | | 1 | Startseite öffnen | | Die Gefühlslage 🎅 ist doppelt so gross wie 🎃 |
| 1.2 | Es existieren keine aktiven Gefühlslageoptionen | |  |  | | --- | --- | | 1 | Startseite öffnen | | Eine leere Seite wird angezeigt |
| 1.3 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 und die deaktivierte Gefühlslageoption 🍄 | |  |  | | --- | --- | | 1 | Startseite öffnen | | 🍄 wird nicht angezeigt |
| 1.4 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃  🎅 x 1 Gefühlslagen  🎃 x 1 Gefühlslagen | |  |  | | --- | --- | | 1 | Startseite öffnen | | 2 | Die Gefühlslageoption 🎅 auswählen | | 3 | Senden drücken | | Die Gefühlslage 🎅 ist doppelt so gross wie 🎃 |
| 1.5 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃  🎅 x 1 Gefühlslagen  🎃 x 1 Gefühlslagen | |  |  | | --- | --- | | 1 | Startseite öffnen | | 2 | Die Gefühlslageoption 🎅 auswählen | | 3 | Als Kommentar «Weihnachten ist cool» eingeben | | 4 | Senden drücken | | Die Gefühlslage 🎅 ist doppelt so gross wie 🎃  Bei der Gefühlslage 🎅 wird der Kommentar «Weihnachten ist cool» angezeigt |
| 1.6 | Es existieren die aktive Gefühlslageoptionen 🎅 | |  |  | | --- | --- | | 1 | Startseite öffnen | | 2 | Die Gefühlslageoption 🎅 auswählen | | 3 | Als Kommentar mindestens 251-mal das Zeichen «a» eingeben | | 4 | Senden drücken | | Beim Senden wird ein Fehler angezeigt, dass der Kommentar zu lang ist |

Tabelle 21: Testfälle 1 - Startseite

### Testfälle 2: Anmeldung

Diese Tests können ausschliesslich von noch nicht autorisierten Benutzern durchgeführt werden.

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 2.1 | Der Administrator «admin» mit dem Passwort «admin» existiert | |  |  | | --- | --- | | 1 | Login öffnen | | 2 | Benutzername «admin» eingeben | | 3 | Passwort «admin» eingeben | | 4 | Anmelden drücken | | Man wird auf die Administration weitergeleitet |
| 2.2 | Der Administrator «qwer» existiert nicht | |  |  | | --- | --- | | 1 | Login öffnen | | 2 | Benutzername «qwer» eingeben | | 3 | Passwort «123» eingeben | | 4 | Anmelden drücken | | Ein Fehler wird angezeigt, dass der Benutzername oder das Passwort falsch ist |
| 2.3 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | Man wird auf die Login-Seite weitergeleitet |
| 2.4 | Der Administrator «admin» mit dem Passwort «admin» existiert | |  |  | | --- | --- | | 1 | Login öffnen | | 2 | Benutzername «admin» eingeben | | 3 | Passwort «admin» eingeben | | 4 | Anmelden drücken | | 5 | Knopf Abmelden drücken | | Man wird auf die Startseite weitergeleitet |

Tabelle 22: Testfälle 2 - Anmeldung

### Testfälle 3: Administration

Diese Tests können ausschliesslich von autorisierten Administratoren durchgeführt werden

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 3.1 | Der Benutzer «admin2» existiert nicht | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Benutzername ändern klicken | | 3 | Als neuen Benutzername «admin2» eingeben | | 4 | Benutzername ändern drücken | | Login mit dem Benutzernamen «admin2» möglich |
| 3.2 | Der Benutzer «admin2» existiert | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Benutzername ändern klicken | | 3 | Als neuen Benutzername «admin2» eingeben | | 4 | Benutzername ändern drücken | | Ein Fehler wird angezeigt, der neue Benutzername existiert bereits |
| 3.3 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Benutzername ändern klicken | | 3 | Als neuen Benutzername 101-mal das Zeichen «a» eingeben | | 4 | Benutzername ändern drücken | | Ein Fehler wird angezeigt, der neue Benutzername ist zu lange |
| 3.4 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Benutzername ändern klicken | | 3 | Benutzername ändern drücken | | Ein Fehler wird angezeigt, der neue Benutzername kann nicht leer sein |
| 3.5 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Passwort ändern auswählen | | 3 | Als neues Passwort «123» eingeben | | 4 | Als Passwort wiederholen «123» eingeben | | 5 | Passwort ändern drücken | | Login mit dem Passwort «123» möglich |
| 3.6 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Passwort ändern auswählen | | 3 | Als neues Passwort «123» eingeben | | 4 | Als Passwort wiederholen «qwer» eingeben | | 5 | Passwort ändern drücken | | Fehler wird angezeigt, Passwörter stimmen nicht überein |
| 3.7 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Passwort ändern auswählen | | 3 | Passwort ändern drücken | | Fehler wird angezeigt, Passwort kann nicht leer sein |
| 3.8 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 und die deaktivierte Gefühlslageoption 🍄 | |  |  | | --- | --- | | 1 | Administration öffnen | | Die Gefühlslageoptionen 🎅, 🎃 und 🍄 werden aufgelistet. Die ersten beiden werden aktiviert angezeigt, die letzte deaktiviert |
| 3.9 | Es existieren keine Gefühlslageoptionen | |  |  | | --- | --- | | 1 | Administration öffnen | | Eine leere Liste wird angezeigt |
| 3.10 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Gefühlslageoption hinzufügen drücken | | 3 | Im Dialog 🍄 auswählen | | 4 | Dialog bestätigen | | Die Gefühlslageoptionen 🎅, 🎃 und 🍄 werden aufgelistet. Alle werden aktiviert angezeigt. Auf der Startseite werden alle drei angezeigt |
| 3.11 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Gefühlslageoption hinzufügen drücken | | 3 | Dialog bestätigen | | Ein Fehler wird angezeigt, der Smiley kann nicht leer sein |
| 3.12 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Gefühlslageoption hinzufügen drücken | | 3 | Als Smiley «qwer» eingeben | | 4 | Dialog bestätigen | | Ein Fehler wird angezeigt, der Smiley ist nicht valide |
| 3.13 | Es existiert die Gefühlslageoption 🎅 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Gefühlslageoption hinzufügen drücken | | 3 | Als Smiley 🎅 eingeben | | 4 | Dialog bestätigen | | Ein Fehler wird angezeigt, der Smiley existiert bereits |
| 3.14 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Die Gefühlslage 🎅 suchen | | 3 | Smiley ändern anwählen | | 4 | Im Dialog den Smiley 🎃 eingeben | | 5 | Dialog bestätigen | | Ein Fehler wird angezeigt, der Smiley existiert bereits |
| 3.15 | Es existiert die Gefühlslageoption 🎅 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Die Gefühlslage 🎅 suchen | | 3 | Smiley ändern anwählen | | 4 | Im Dialog den Smiley 🎃 eingeben | | 5 | Dialog bestätigen | | Die Gefühlslage besitzt nun einen neuen Smiley |
| 3.16 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 und die deaktivierte Gefühlslageoption 🍄 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Die Gefühlslage 🎅 suchen | | 3 | Gefühlslage deaktivieren drücken | | Die Gefühlslageoption 🎃 wird aktiv angezeigt, die anderen zwei 🎅 und 🍄 werden deaktiviert angezeigt. Auf der Startseite sieht man nur 🎃 |
| 3.17 | Es existieren die zwei aktiven Gefühlslageoptionen 🎅 & 🎃 und die deaktivierte Gefühlslageoption 🍄 | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Die Gefühlslage 🍄 suchen | | 3 | Gefühlslage aktivieren drücken | | Die Gefühlslageoptionen 🎅, 🎃 und 🍄 werden aufgelistet. Alle werden aktiviert angezeigt. Auf der Startseite werden alle drei angezeigt |
| 3.18 | Die Gefühlslagen 🎃 & 🎅 existieren und haben Gefühlslagen für den jetzigen und vorherigen Tag | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Als Von-Datum das Datum des vorherigen Tages eintragen | | 3 | Als Bis-Datum das jetzige Datum eintragen | | Ein Liniendiagramm mit zwei unterschiedlichen Linien für die beiden Smileys wird angezeigt. Auf der X-Achse sind die zwei Tage zu sehen, auf der Y-Achse die Anzahl Gefühlslagen |
| 3.19 | Es sind keine Gefühlslagen vorhanden | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Als Von-Datum das Datum des vorherigen Tages eintragen | | 3 | Als Bis-Datum das jetzige Datum eintragen | | Leeres Liniendiagramm wird dargestellt |
| 3.20 |  | |  |  | | --- | --- | | 1 | Administration öffnen | | 2 | Als Von-Datum das jetzige Datum eintragen | | 3 | Als Bis-Datum das Datum gestrige Datum eintragen | | Ein Fehler wird angezeigt, das Bis-Datum muss grösser oder gleich dem Von-Datum sein |

Tabelle 23: Testfälle 3 - Administration

## Software-Architektur

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

### C:\IPA\HappyMeter\UML\ZLFBKiGW4BpFLr2vxozO-r1FUafvilP00cOf2UWmsIgrykz24f5ioEeHfhjdkWUsZZXIrse0BL-8aMMecXCneL3MfAm1C5P8LjoZDIIDADWNCFPyoucN4i-gbiTmVh2D.pngÜbersicht

Abbildung 3: Komponentenübersicht

Das Backend ist als RESTful Service umgesetzt und bietet eine REST-Schnittstelle an, welche über HTTP & JSON kommuniziert. Alle im Controller aufgeführten Komponenten können von aussen angesprochen werden und bilden die Schnittstelle zum Frontend. Das Frontend spricht diese Schnittstellen via HTTP-Anfragen an und stellt die geladenen Daten im Browser dar.

### Autorisierung

Die Autorisierung ist selbst implementiert und möglichst simpel gehalten. Da die Website ohne HTTPS und firmenintern betrieben wird, ist die Sicherheit der Übermittlung kein kritischer Faktor. Bei der Anmeldung speichert das Frontend die Anmeldedaten im Browser-Sessionstorage, welcher für solch sensitive Daten ausreichend ist. Bei jeder Abfrage werden am Server Benutzername und Passwort mitgeteilt, damit dieser sicherstellen kann, dass man autorisiert ist.

### Frontend

Das Frontend verfügt Angular (2+) über Hilfsmittel, welche das Trennen von Verantwortlichkeiten erleichtern. Das Frontend verfügt neben der Darstellung über drei Dienste, welche mit dem Server kommunizieren. Bei einer Serveranfrage wird ein HTTP-Anfrage erstellt und an den Server übermittelt. Falls der Benutzer angemeldet ist sind in dieser Anfrage auch die Login-Daten zur Autorisierung vorhanden. Sobald der Server antwortet wird je nach Fall anders mit den Daten umgegangen. Einheitlich werden die HTTP-Codes 500 (Internal Server Error) und 401 (Unauthorized) behandelt. Im Falle eines 500 wird ein Dialog angezeigt, dass etwas schiefgelaufen ist. Im Falle einer 401-Antwort wird der Benutzer auf die Login-Seite weitergeleitet, falls er nicht bereits auf dieser ist.

### Backend

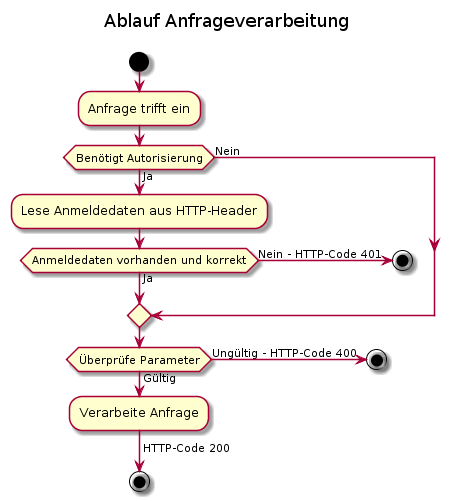
Da REST per Definition Stateless ist (jede Anfrage wird isoliert für sich bearbeitet), werden Autorisierungen über das «AuthorizationAttribute» gelöst. Dieses Attribut wird selbst implementiert und kann an jede Server-Schnittstelle angebracht werden. Sobald das Attribut aktiv ist werden alle Anfragen zuerst auf Autorisierung überprüft und sonst zurückgewiesen.

Abbildung 4: Ablauf Anfrageverarbeitung

## Datenbankdesign

In diesem Kapitel wird die Struktur der Datenbank festgelegt. Die Struktur der MySQL Datenbank wird mit der MySQL-Workbench erstellt. Eine Erweiterung der emotional\_state Tabelle zum Ergänzen eines Ortes sollte keine Schwierigkeiten darstellen.

### ERD

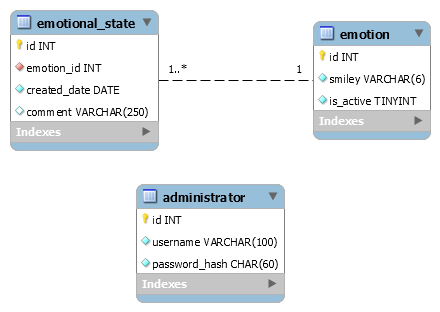


Abbildung 5: ERD

### Entitätsbeschreibungen

|  |  |
| --- | --- |
| emotional\_state (Gefühlslage) | |
| Beschreibung | Eine erfasste Gefühlslage mit einem optionalen Kommentar |
| Verbindungen | Eine Gefühlslage hat immer exakt eine zugewiesene Gefühlslageoption |
| Primärschlüssel | id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | id | INT | Die eindeutige Identifikation | Pflichtfeld  Generiert | | emotion\_id | INT | Die dazugehörige Emotion | Pflichtfeld | | created\_date | DATE | Das Datum, an welchem die Gefühlslage erfasst wurde | Pflichtfeld | | comment | VARCHAR(250) | Ein optionaler Kommentar mit maximal 250 Zeichen |  | |

Tabelle 24: Entitätsbeschreibung emotional\_state

|  |  |
| --- | --- |
| emotion (Gefühlslageoption) | |
| Beschreibung | Eine Gefühlslageoption mit einem Smiley und Aktiv-Zustand |
| Verbindungen | Eine Gefühlslageoption hat mehrere Gefühlslagen |
| Primärschlüssel | id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | id | INT | Die eindeutige Identifikation | Pflichtfeld  Generiert | | smiley | VARCHAR(6) | Der Unicode-Charaktercode als hexadezimaler string  Bsp. Für 1F383 für 🎃 | Pflichtfeld  Unique | | is\_active | TINYINT(1)  (BOOLEAN) | Ob diese Gefühlslageoption ausgewählt werden kann | Pflichtfeld  Default: 1 | |

Tabelle 25: Entitätsbeschreibung emotion

|  |  |
| --- | --- |
| administrator | |
| Beschreibung | Ein Administratorbenutzer |
| Primärschlüssel | id |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feld | Typ | Beschreibung | Sonstiges | | id | INT | Die eindeutige Identifikation | Pflichtfeld  Generiert | | username | VARCHAR(100) | Der Anmeldenamen eines Administrators | Pflichtfeld  Unique | | password\_hash | CHAR(60) | Der sichere bcrypt-Hash eines Passworts | Pflichtfeld | |

Tabelle 26: Entitätsbeschreibung administrator

# Entscheiden

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

## Versionskontrolle

Für regelmässige Backups und risikofreies Programmieren ist eine gute Versionsverwaltung essenziell. Aus diesem Grund habe ich mich für Git entschieden, der De-Facto-Standard in moderner Codeverwaltung. Die grossen Vorteile von Git sind, dass es keinen Server benötigt und man ohne Probleme zu einem funktionierenden Stand zurückgehen kann. Die Arbeit kann also auch ohne Internetverbindung problemlos fortgesetzt werden, falls es einmal Netzwerkschwankungen gibt. Falls man dann mit dem Internet verbunden ist kann man aus diversen Online-Git-Anbietern auswählen oder einen firmeninternen Git-Server benutzen. Da ich gute Erfahrungen mit GitHub gemacht habe und ein privates Repository erstellen kann, werde ich diesen Online-Dienst für meine Backups benutzen.

## Backend-Framework

Microsoft arbeitet seit geraumer Zeit an der plattformunabhängigen C#-Implementation .NET Core. Diese Core Implementation sind übersichtlicher, schneller und moderner. Microsoft hat eine [Offizielle Seite](https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server), auf welcher alle Vergleiche aufgelistet werden. Die Hauptunterschiede belassen sich darauf, dass .NET Core auch unter macOS und Linux funktioniert.

Schlussendlich habe ich mich für das herkömmliche ASP.NET mit Web API 2 entschieden (basierend auf dem Windows .NET Framework). ASP.NET Core wäre zwar sicher die bessere Variante für ein Schulprojekt, jedoch besitze ich damit keine Erfahrungen. Ausserdem arbeitet meine Abteilung exklusiv mit ASP.NET und Web API 2, der Hauptgrund für die Entscheidung dieses Frameworks.

## Frontend-Framework

Als Frontend- Framework wird Angular4 eingesetzt. Sowohl in der Abteilung als in Schulprojekten habe ich Erfahrungen mit diesem von Google entwickelte Single-Page-Application-Framework gesammelt, weshalb ich es weiterhin verwenden möchte. Erst kürzlich ist Angular5 erschienen, eine neuere Version mit diversen Verbesserungen. Diese sind jedoch für mein Projekt nicht relevant, weshalb ich mit der stabilen Angular4 Version arbeiten werde.

## Smiley-Umsetzung

Als Vorbereitung für dieses Projekt habe ich mich grob mit dem Thema Emoji im Web auseinandergesetzt. Dabei auffallend ist vor allem, dass die Darstellung der Emojis stark vom Browser und Betriebssystem abhängig ist, wie man am [Beispiel des Kürbis-Emojis online sehen kann](https://emojipedia.org/jack-o-lantern/). So unterstützen alle Browser unter Windows 10 Emojis, unter Windows 7 hingegen nur der Firefox. Die anderen Browser stellen Emojis zwar dar, jedoch als Schwarz-Weiss-Skizze und nicht als farbenfrohes Emoji. Deshalb habe ich mich dazu entschieden alle Emojis plattformunabhängig darzustellen. Ein Open-Source-Standard von Twitter namens [Twemoji](https://twitter.github.io/twemoji/preview.html) hat schnell meine Aufmerksamkeit auf sich gezogen. Die verschiedenen Emojis können über eine URL angesprochen werden, welches einfach aus dem Unicode-Zeichen zusammengesetzt wird. Ich habe mich daher dafür entschieden die Eingabe über das Emoji-Zeichen zu gestalten (aus einer Auswahl) und diese danach als Twemoji auf allen Plattformen gleich darzustellen.

## Diagramm-Visualisierung

Wie in allen Schulprojekten werde ich [PlantUML](http://plantuml.com/) beziehungsweise die Web-Version [PlantText](https://www.planttext.com/) für die Darstellung meiner Diagramme verwenden. Im Unterschied zu Visio und ähnlichen Programmen erstellt man die Diagramme nicht in einem GUI-Editor, sondern in einem Text-Editor. Man programmiert sozusagen Diagramme und muss sich nicht um die korrekte UML-Darstellung kümmern. Da es gratis zur Verfügung steht und auch in meiner Abteilung oft verwendet wird werde ich all meine Architektur-Diagramme mithilfe dieses Tools erstellen.

# Realisieren

In diesem Kapitel werden auf die Details der Implementierung eingegangen.

## Backend-Klassenübersicht

Die Klassen wurden gemäss Komponentendiagramm der Planung umgesetzt. Bei einem Fehler der Input-Validierung wird der HTTP-Code «400: Bad Request» zurückgegeben. Bei einem unerwarteten Fehler wird der HTTP-Code «500: Internal Server Error» zurückgegeben.

### WebApiConfig (Routing)

Das Routing wird zuerst im Backend umgesetzt. Dabei werden alle Routen nach dem Schema http://server/api/CONTROLLER/METHODE. Der Controller ist dabei Standardmässig «Home» und die Methode «Index». Beim Platzhalter CONTROLLER wird der Suffix Controller weggelassen (Beispielsweise User anstelle von UserController). Beim Zugriff auf die Index Methode des Home-Controllers wird auf die index.html Datei verwiesen. Diese beinhaltet die Angular-Applikation, welches eine separates Frontend-Routing vornimmt. Dabei beginnen die Routen im Frontend immer mit eine #. Das Grundkonzept des Routings existiert so im Produktivprojekt.

### HappyMeterDatabaseContext (Datenbankanbindung)

Die Datenbankanbindung ist mit dem Entity Framework 6 als OR-Mapper realisiert. Die Datenbank wurde zuerst erstellt und daraus der Code generiert. Da der Code-Generator nicht funktioniert hat, wurden die Klassen von Hand generiert. Die Speicheroperationen im EF6 sind automatisch transaktionell geschützt. Die Abfragen auf die Datenbank werden in effizientes SQL umgewandelt, weshalb Performance auch bei etwas grösseren Datensätzen kein Problem darstellen wird. Alle Datenbankaufrufe sind mit dem C#-Sprachfeature Async-Await umgesetzt. Dies ermöglicht beim Warten auf IO-Operationen (Aufruf der Datenbank) den Thread nicht zu blockieren. Dadurch bleibt der Webserver auch bei einer langsamen Datenbankverbindung Reaktionsfähig.

### HomeController

Gibt beim Aufruf den HTTP-Code «301: Permanently Moved» zurück und verweist auf die index.html Datei.

### UserController

Bietet die Möglichkeit, seine Anmeldedaten zu überprüfen, einen neuen Benutzernamen zu setzen und sein Passwort zu ändern. Ändert man seinen Benutzernamen zu einem bereits existierenden Benutzernamen wird der HTTP-Code «400: Bad Request» zurückgegeben. Alle Methoden dieses Controllers verfügen über einen Berechtigungs-Check.

### EmotionController

Ermöglicht das Laden und Verwalten der Gefühlslageoptionen. Dabei kann man den Aktive-Zustand verändern, Smiley verändern oder einen neuen Smiley hinzufügen. Existiert eine bearbeitende Gefühlslageoption nicht, wird der HTTP-Code «404: Not Found» zurückgegeben. Alle Methoden dieses Controllers verfügen über einen Berechtigungs-Check, ausser dem Herunterladen der aktiven Gefühlslageoptionen.

### EmotionalStateController

Durch diese Schnittstelle können Benutzer ihre Gefühlslage erfassen und die Gefühlslage der Mitarbeiter beurteilen. So können alle gewählten Gefühlslagen geladen werden, welche momentan aktiv sind. Auch kann eine Gefühlslage mit einer Gefühlslageoption hinzugefügt werden, welche aktiv ist und existiert. Ist dies nicht der Fall, wird der HTTP-Code «400: Bad Request» zurückgegeben. Als Administrator kann man ausserdem eine Gruppierte Version der Gefühlslagen in einem Zeitraum laden.

## Frontend-Klassenübersicht

Das Frontend ist von der Planung abgewichen und verfügt jetzt über eine Trennung, welche dem Server nahekommt. Diese Trennung erschien sinnvoller als die originale Planung.

### CanActivateViaAuthGuard

Verhindert das Aufrufen der Administrations-Seite, solange der Benutzer nicht angemeldet ist. Ohne Anmeldedaten wird auf die Login-Seite weitergeleitet.

### AuthorizeInterceptor

Ein HTTP-Interceptor, welcher für alle Anfragen den Benutzername und das Passwort in den HTTP-Header schreibt, falls man angemeldet ist. Dies ermöglicht dem Server eine Autorisierung durchzuführen, ohne dass der restliche Server-Code dies berücksichtigen muss.

### ErrorHandlerInterceptor

Ein HTTP-Interceptor, welcher alle Fehler abfängt. Im Falle eines 401 wird auf die Login-Seite verwiesen, im Falle 500 der Error-Dialog angezeigt. Zusätzlich wird eine Warnung in die Konsole ausgegeben, damit man als Entwickler den Fehler finden kann.

### AuthService

Verwaltet die gespeicherten Benutzerdaten. Zur Speicherung werden diese in den sessionStorage geschrieben, welcher beim Schliessen des Tabs geleert wird. Verwendet wird dieser Service vom Login, welcher die Daten einträgt, und den beiden Klassen AuthorizeInterceptor und CanActivateViaAuthGuard, welche diese auslesen.

### ServerService

Eine Basisklasse, welche die Server-URL zur Laufzeit ermittelt. Dabei geht es vor allem darum, ob man im Entwickler- oder im Release-Modus ist. Im Release-Modus wird als Server später eine dedizierte Adresse zugewiesen, für die Entwicklung wird die Lokale Maschine als Server verwendet.

### EmotionService & EmotionalStateService & UserService

Ableitungen des ServerServices, welche die Methoden des Servers aufrufen und die Parameter in die korrekte Form bringen.

## Frontend GUI

### AdministrationComponent

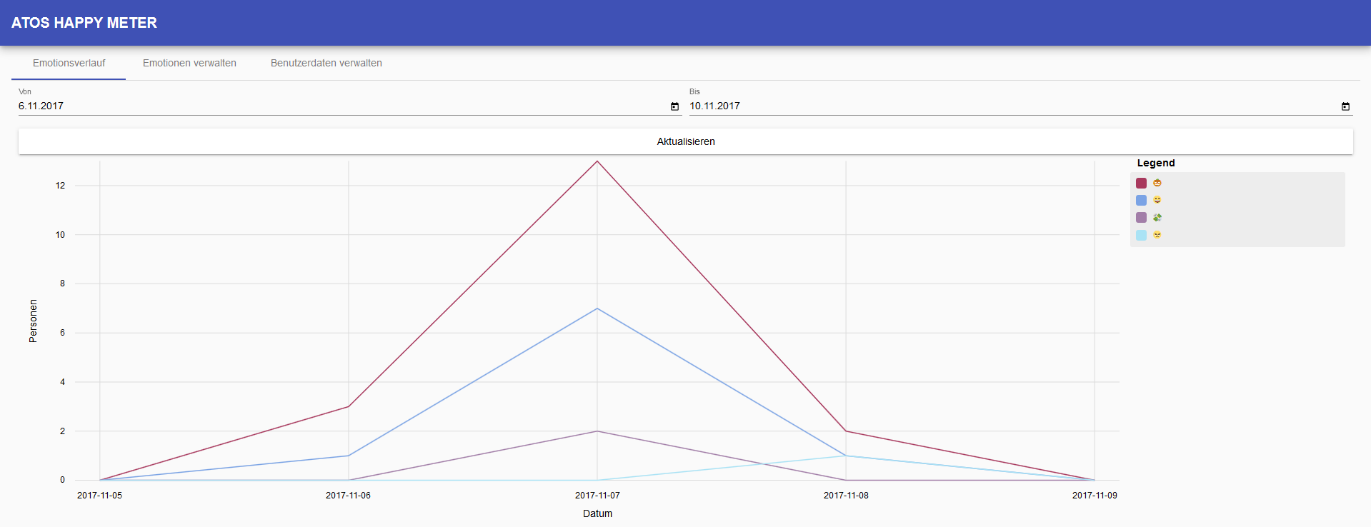
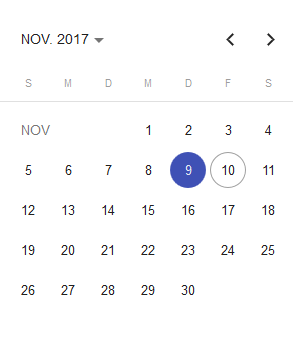
Unter der Route «/Administration» aufzufinden. Zeigt in drei Tabs die Administrationsfunktionen an. Im ersten Tab kann ein Emotionsverlauf angezeigt werden. Mit den zwei Date-Pickern kann ein Von- und Bis-Datum gesetz werden. Durch den Knopf Aktualisieren werden die Daten für diese Zeitspanne vom Server geladen. Das Von-Datum muss kleiner oder gleich dem Bis-Datum sein.

Abbildung 6: Screenshot Administration/Emotionsverlauf

Durch das Wechseln in den «Emotionen Verwalten»-Tab können die Gefühlslageoptionen verwaltet werden. In dieser Ansicht werden alle Gefühlslageoptionen inklusive Emoji angezeigt. Für die Darstellung der Emojis wird die EmojiDisplay-Komponente verwendet. Durch das drücken auf deaktivieren kann eine Gefühlslage deaktiviert werden. Sobald sie deaktiviert ist wechselt der Knopf zu einem Aktivieren Knopf. Beim Drücken auf Smiley ändern oder Emotion hinzufügen wird der SelectEmoji-Dialog geöffnet.

Abbildung 7: Screenshot Date-Picker



Abbildung 8: Screenshot Administration/Emotionen verwalten

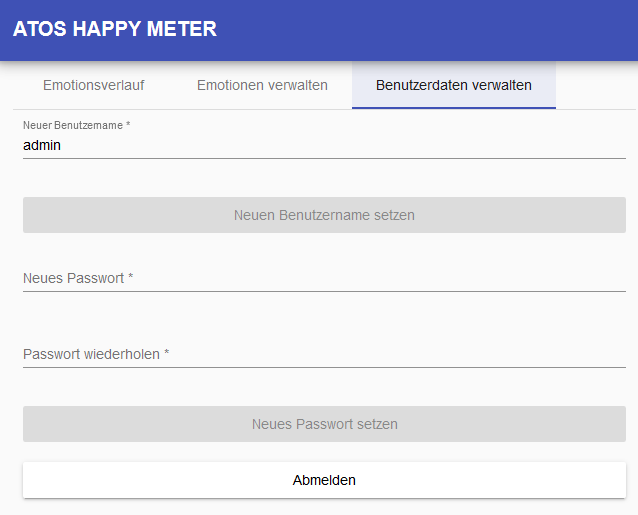
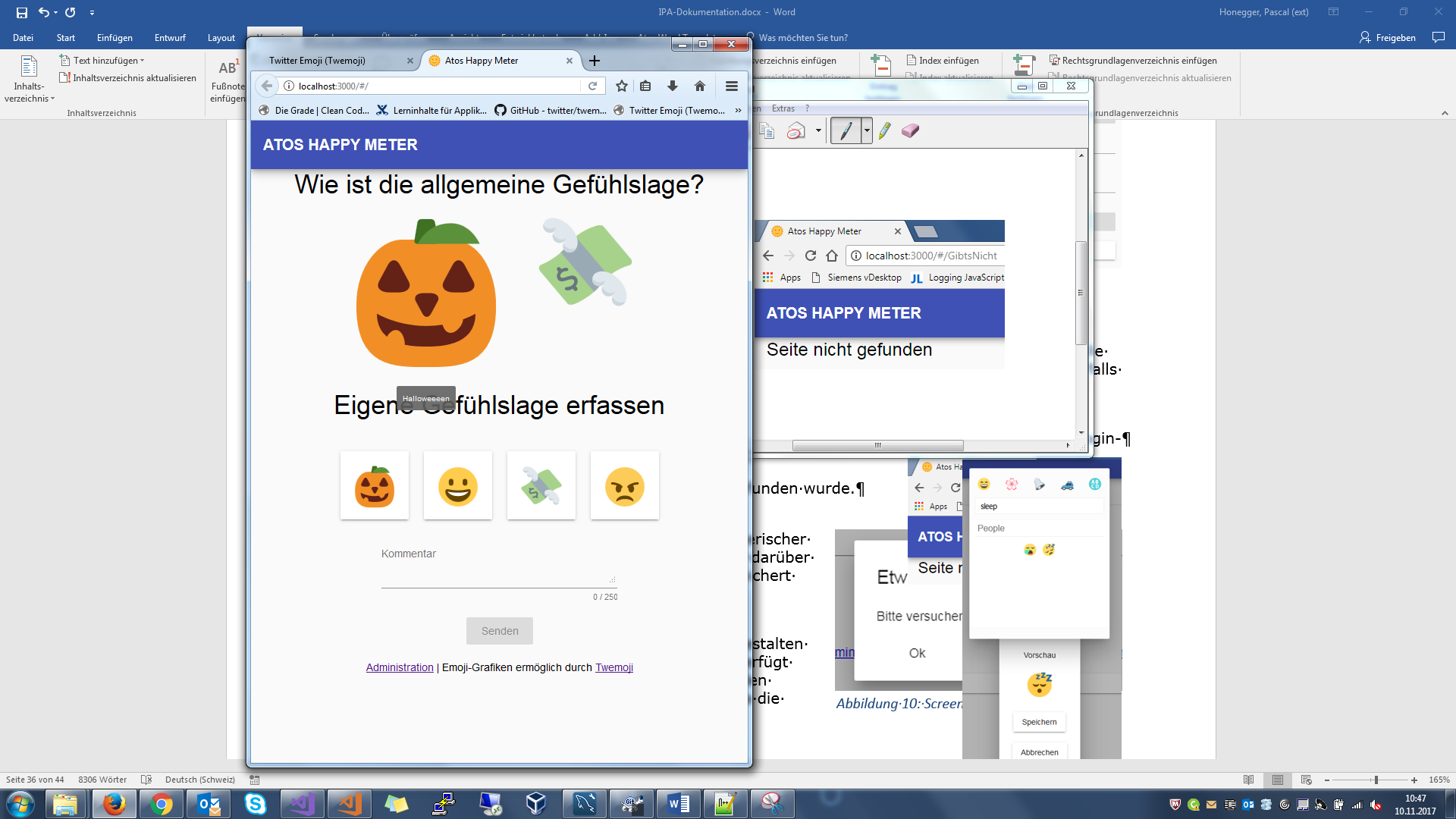
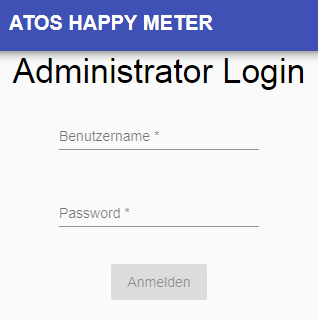
In dem Dritten Tab kann man seine Benutzerdaten verwalten. In einem simplen Formular kann der eigene Benutzername oder Passwort geändert werden. Das neue Passwort muss wiederholt eingegeben werden, um Tippfehler zu verhindern. Zusätzlich gibt es hier den Abmelden-Knopf, welcher einen nach erfolgreichen Abmelden auf die Startseite zurückleitet. Die Knöpfe Benutzername setzen und Neues Passwort setzen sind ausgegraut, solange keine Eingabe getroffen wurde.

Abbildung 9: Screenshot Administration/Benutzerdaten verwalten

### EmojiDisplayComponent

Diese Komponente stellt die Emojis mithilfe von [Twemoji](https://twitter.github.io/twemoji/$) dar. Um dies zu bewerkstelligen wird der Unicode-CodePoint in Hex konvertiert und in eine URL-Vorlage abgefüllt. Dadurch sehen die Emojis auf allen Plattformen gleich aussehen.

### HomeComponent

Die Startseite für alle Benutzer und unter der Route «/» aufzufinden. Stellt am oberen Rand alle vom Server gelieferten Gefühlslagen an. Als Tooltip wird ein zufälliger Kommentar angezeigt, falls einer existiert. Die Grösse der dargestellten Emotionen ist relative zu der Auswahl. In der Grafik links existieren beispielsweise zwei Gefühlslagen zum Kürbis und eine für das Geld. Daher macht der Kürbis zwei Drittel der Grösse aus und das Geld lediglich einen Drittel. In der Auswahl unten kann jeweils nur ein Knopf aktiv sein. Der Kommentar darf höchstens 250 Zeichen lang sein und zeigt dieses Längen-Limit direkt im Feld an. Zur Darstellung der Emojis (Übersicht & Auswahl) wird die EmojiDisplay-Komponente verwendet. Nach dem erfolgreichen Senden wird der Knopf für eine Minute blockiert, um Spam zu verhindern.

### LoginComponent

Die Login-Seite. Bei erfolgreichem Anmelden wird auf die Administration weitergeleitet.

### NoContentComponent

Diese Seite wird angezeigt, falls die URL nicht gefunden wurde.

Abbildung 10: Screenshot Login

### ErrorDialogComponent

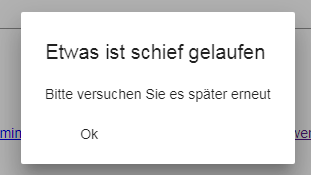
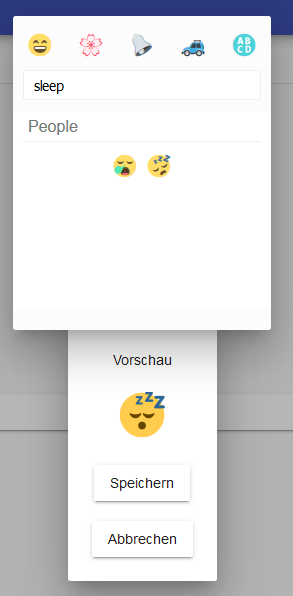
Im Falle eines unerwarteten Fehlers wird ein generischer Fehlerdialog angezeigt. Dadurch ist der Benutzer darüber informiert, dass etwas beispielsweise nicht gespeichert wurde.

Abbildung 12: Screenshot Error Dialog

### SelectEmojiDialogComponent

Um die Auswahl der Emojis möglich einfach zu gestalten habe ich einen Emoji-Picker verwendet. Dieser verfügt sogar über eine Englische Suche, welche das finden gewünschter Gefühlslageoptionen vereinfacht. Da die Darstellung der Emojis innerhalb des Emoji-Pickers Browserabhängig ist gibt es unterhalb eine Vorschau mithilfe der EmojiDisplay-Komponente.

### Sequenzdiagramm??

Ablauf inklusive Interceptors & Attributes

Abbildung 13: Screenshot Emoji-Picker Dialog

# Kontrollieren

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

## Zeitmanagement

|  |  |  |
| --- | --- | --- |
| Meilenstein | Gesetzes Datum | Effektives Datum |
| Projektstart | 01.11.2017 09:00 | 01.11.2017 09:00 |
| Ende Informieren | 01.11.2017 11:00 | 01.11.2017 11:00 |
| Ende Planen | 02.11.2017 16:00 | 02.11.2017 16:00 |
| Ende Entscheiden | 02.11.2017 17:00 | 02.11.2017 17:00 |
| Ende Realisieren | 08.11.2017 17:00 | 09.11.2017 11:00 |
| Ende Kontrollieren | 10.11.2017 11:00 | 10.11.2017 14:00 |
| Projektabschluss | 10.11.2017 17:00 | 10.11.2017 18:00 |

Leider war der Aufwand für die Realisierungsphase grösser wie erwartet. Aus diesem Grund haben sich die Meilensteine nach hinten verschoben.

## Tests

✔ = Bestanden

✔ (**~**) = Bestanden mit Abweichung

✘ = Fehlgeschlagen

### Testresultate 1: Startseite

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 1.1 | ✔ |  |
| 1.2 | ✔ (**~**) | Es wird Angezeigt, dass keine Gefühlslagen erfasst sind |
| 1.3 | ✔ |  |
| 1.4 | ✔ |  |
| 1.5 | ✔ |  |
| 1.6 | ✔ (**~**) | Eingabe von mehr als 250 Zeichen nicht möglich |

Tabelle 27: Testresultate 1

### Testresultate 2: Anmeldung

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 2.1 | ✔ |  |
| 2.2 | ✔ |  |
| 2.3 | ✔ |  |
| 2.4 | ✔ |  |

Tabelle 28: Testresultate 2

### Testresultate 3: Administration

|  |  |  |
| --- | --- | --- |
| Nr. | Bestanden | Abweichungen |
| 3.1 | ✔ |  |
| 3.2 | ✔ |  |
| 3.3 | ✔ | Eingabe von mehr als 100 Zeichen nicht möglich |
| 3.4 | ✔ |  |
| 3.5 | ✔ |  |
| 3.6 | ✔ |  |
| 3.7 | ✔ |  |
| 3.8 | ✔ (**~**) | Man erkennt ob aktiviert oder deaktiviert anhand des Knopfes |
| 3.9 | ✔ |  |
| 3.10 | ✔ |  |
| 3.11 | ✘ | Smiley wird nicht hinzugefügt, es wird kein Fehler angezeigt |
| 3.12 | ✔ (**~**) | Eingabe nicht möglich |
| 3.13 | ✔ |  |
| 3.14 | ✔ |  |
| 3.15 | ✔ |  |
| 3.16 | ✔ |  |
| 3.17 | ✔ |  |
| 3.18 | ✔ |  |
| 3.19 | ✔ (**~**) | Linien werde mit jeweils 0 erfassten Gefühlslagen pro Tag dargestellt |
| 3.20 | ✔ |  |

Tabelle 29: Testresultate 3

## Testfazit

Von den 30 Testfällen haben 24 bestanden. 5 weitere haben bestanden, hatten jedoch Abweichungen. Ein Testfall ist fehlgeschlagen und weist das falsche Verhalten auf.

Sind die Tests bestanden? Kann das Produkt produktiv verwendet werden?

# Auswerten

## Reflexion

Wurden alle Ziele erreicht?

## Verbesserungen / Erfahrungen

Welche Verbesserungen am Projekt gibt es? Was würde ich anders machen? Was habe ich gelernt?

## Einsetzbarkeit

Kann es Produktiv eingesetzt werden?

## Fazit

Wie verlief das Projekt?

# Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
|  |  |
|  |  |

# Verzeichnisse

## Quellenverzeichnis

**Im aktuellen Dokument sind keine Quellen vorhanden.**

## Abbildverzeichnis

[Abbildung 1: Zeitplan 9](file:///C:\Users\Pascal\Desktop\IPA-Dokumentation.docx#_Toc497423219)

[Abbildung 2: Use-Case Diagramm 18](file:///C:\Users\Pascal\Desktop\IPA-Dokumentation.docx#_Toc497423220)

[Abbildung 3: Komponentenübersicht 28](file:///C:\Users\Pascal\Desktop\IPA-Dokumentation.docx#_Toc497423221)

[Abbildung 4: Ablauf Anfrageverarbeitung 29](file:///C:\Users\Pascal\Desktop\IPA-Dokumentation.docx#_Toc497423222)

[Abbildung 5: ERD 30](#_Toc497423223)

## Tabellenverzeichnis

[Tabelle 1: Projektteilnehmer 6](#_Toc497423194)

[Tabelle 2: Meilensteine 8](#_Toc497423195)

[Tabelle 3: Konkrete Planung Tag 1 10](#_Toc497423196)

[Tabelle 4: Konkrete Planung Tag 2 11](#_Toc497423197)

[Tabelle 5: Konkrete Planung Tag 3 13](#_Toc497423198)

[Tabelle 6: Use-Case 1.1 19](#_Toc497423199)

[Tabelle 7: Use-Case 1.2 19](#_Toc497423200)

[Tabelle 8: Use-Case 1.3 20](#_Toc497423201)

[Tabelle 9: Use-Case 2.1 20](#_Toc497423202)

[Tabelle 10: Use-Case 2.2 21](#_Toc497423203)

[Tabelle 11: Use-Case 2.3 21](#_Toc497423204)

[Tabelle 12: Use-Case 3.1 21](#_Toc497423205)

[Tabelle 13: Use-Case 3.2 22](#_Toc497423206)

[Tabelle 14: Use-Case 3.3 22](#_Toc497423207)

[Tabelle 15: Use-Case 3.4 22](#_Toc497423208)

[Tabelle 16: Use-Case 3.5 23](#_Toc497423209)

[Tabelle 17: Use-Case 3.6 23](#_Toc497423210)

[Tabelle 18: Use-Case 3.7 23](#_Toc497423211)

[Tabelle 19: Use-Case 3.8 24](#_Toc497423212)

[Tabelle 20: Testfälle 1 - Startseite 25](#_Toc497423213)

[Tabelle 21: Testfälle 2 - Anmeldung 25](#_Toc497423214)

[Tabelle 22: Testfälle 3 - Administration 28](#_Toc497423215)

[Tabelle 23: Entitätsbeschreibung emotional\_state 31](#_Toc497423216)

[Tabelle 24: Entitätsbeschreibung emotion 31](#_Toc497423217)

[Tabelle 25: Entitätsbeschreibung administrator 32](#_Toc497423218)

# Quellcode

## Frontend

### administration.component.css

.edit-emoji-display {

    width: 50px;

}

.mat-tab-content-div {

    margin: 10px;

}

.input-fill-width {

    width: 100%;

}

### administration.component.html

<mat-tab-group>

<mat-tab label="Emotionsverlauf">

<div fxLayout="column" class="mat-tab-content-div">

<div fxLayout="row" fxLayoutGap="10px">

<mat-form-field fxFlex>

<input matInput [matDatepicker]="fromDatePicker" placeholder="Von"

[(ngModel)]="fromDate">

<mat-datepicker-toggle matSuffix [for]="fromDatePicker"></mat-datepicker-toggle>

<mat-datepicker #fromDatePicker></mat-datepicker>

</mat-form-field>

<mat-form-field fxFlex>

<input matInput [matDatepicker]="toDatePicker" placeholder="Bis"

[(ngModel)]="toDate">

<mat-datepicker-toggle matSuffix [for]="toDatePicker"></mat-datepicker-toggle>

<mat-datepicker #toDatePicker></mat-datepicker>

</mat-form-field>

</div>

<button mat-raised-button (click)="tryRefreshChart()">Aktualisieren</button>

<ngx-charts-line-chart \*ngIf="chartData"

[results]="chartData"

[xAxis]="true"

[yAxis]="true"

[legend]="true"

[showXAxisLabel]="true"

[showYAxisLabel]="true"

xAxisLabel="Datum"

yAxisLabel="Personen">

</ngx-charts-line-chart>

</div>

</mat-tab>

<mat-tab label="Emotionen verwalten">

<div fxLayout="column" fxLayoutGap="20px" class="mat-tab-content-div">

<div \*ngFor="let emotion of allEmotions" fxLayout="row" fxLayoutGap="20px">

<emoji-display class="edit-emoji-display" [emojiCodePoint]="emotion.smiley"></emoji-display>

<button mat-raised-button (click)="invertActive(emotion)">{{ emotion.isActive ? 'Deaktivieren' : 'Aktivieren' }}</button>

<button mat-raised-button (click)="openSetEmojiDialog(emotion.id)">Smiley ändern</button>

</div>

<button mat-raised-button (click)="openNewEmojiDialog()">Emotion hinzufügen</button>

</div>

</mat-tab>

<mat-tab label="Benutzerdaten verwalten">

<div fxLayout="column" fxLayoutGap="20px" class="mat-tab-content-div">

<mat-form-field class="input-fill-width">

<input matInput placeholder="Neuer Benutzername" [formControl]="usernameFormControl" [(ngModel)]="username" required minLength="1" maxLength="100" pattern="[A-Za-z0-9 \-\+\\*%&/\\()=?!:;.,üöäÄÖÜÊêéèà<>^]\*">

<mat-error \*ngIf="usernameFormControl.hasError('pattern')">

Bitte keine ungewöhnlichen Zeichen verwenden

</mat-error>

<mat-error \*ngIf="usernameFormControl.hasError('required') || usernameFormControl.hasError('minLength') || usernameFormControl.hasError('maxLength')">

Bitte einen Benutzernamen zwischen 1 und 100 Zeichen eingeben

</mat-error>

</mat-form-field>

<button mat-raised-button [disabled]="usernameFormControl.invalid || authService.username === username" (click)="setNewUsername()">

Neuen Benutzername setzen

</button>

<mat-form-field class="input-fill-width">

<input matInput placeholder="Neues Passwort" type="password" [formControl]="passwordFormControl" [(ngModel)]="password" required minLength="3" maxLength="100">

<mat-error \*ngIf="passwordFormControl.hasError('required') || passwordFormControl.hasError('minLength') || passwordFormControl.hasError('maxLength')">

Bitte ein Passwort zwischen 3 und 100 Zeichen eingeben

</mat-error>

</mat-form-field>

<mat-form-field class="input-fill-width">

<input matInput placeholder="Passwort wiederholen" type="password" [(ngModel)]="repeatedPassword" required minLength="3" maxLength="100">

</mat-form-field>

<button mat-raised-button [disabled]="passwordFormControl.invalid" (click)="setNewPassword()">

Neues Passwort setzen

</button>

<button mat-raised-button (click)="logout()">

Abmelden

</button>

</div>

</mat-tab>

</mat-tab-group>

### administration.component.ts

import { Component } from '@angular/core';

import { EmotionalStateService } from './../services/emotional-state.service';

import { EmotionService } from './../services/emotion.service';

import { FormControl } from '@angular/forms';

import { MatSnackBar, MatDialog } from '@angular/material';

import { UserService } from './../services/user.service';

import { AuthService } from './../services/auth.service';

import { HttpErrorResponse } from '@angular/common/http';

import { Router } from '@angular/router';

import { FullEmotion } from './../model/full-emotion.model';

import { SelectEmojiDialogComponent } from './../select-emoji-dialog/select-emoji-dialog.component';

const oneDayInMiliseconds = 1 \* 24 \* 3600 \* 1000;

@Component({

selector: 'administration',

styleUrls: ['./administration.component.css'],

templateUrl: './administration.component.html'

})

export class AdministrationComponent {

public allEmotions: FullEmotion[];

public fromDate: Date;

public toDate: Date;

/\* Example:

{

name: '🎃',

series: [

{

name: '2017-11-07',

value: 12

},

{

name: '2017-11-08',

value: 4

}

]

}

\*/

public chartData: Array<

{

name: string,

emotionId: number,

series: Array<{

name: string,

value: number

}>

}>;

public username: string = '';

public password: string = '';

public repeatedPassword: string = '';

public usernameFormControl: FormControl = new FormControl('');

public passwordFormControl: FormControl = new FormControl('');

constructor(public authService: AuthService,

private userServer: UserService,

private emotionServer: EmotionService,

private emotionalStateServer: EmotionalStateService,

private snackBar: MatSnackBar,

private router: Router,

private dialog: MatDialog) {

this.fromDate = new Date(Date.now() - oneDayInMiliseconds);

this.toDate = new Date(Date.now());

this.fromDate.setHours(0, 0, 0, 0);

this.toDate.setHours(0, 0, 0, 0);

this.loadEmotions();

this.username = this.authService.username;

}

public loadEmotions() {

this.emotionServer.allEmotions().subscribe((data) => {

this.allEmotions = data;

});

}

public invertActive(emotion: FullEmotion) {

this.emotionServer.setActiveForEmotion(emotion.id, !emotion.isActive).subscribe(() => {

emotion.isActive = !emotion.isActive;

});

}

public setNewUsername() {

this.userServer.setNewUsername(this.username).subscribe(() => {

this.authService.username = this.username;

this.snackBar.open('Benutzername geändert', 'Ok');

}, (error: HttpErrorResponse) => {

if (error.status === 400) {

this.snackBar.open('Der neu gewählte Benutzername existiert bereits', 'Ok');

}

});

}

public setNewPassword() {

if (this.password !== this.repeatedPassword) {

this.snackBar.open('Die Passwörter stimmen nicht überein', 'Ok');

return;

}

this.userServer.setNewPassword(this.password).subscribe(() => {

this.authService.password = this.password;

this.password = '';

this.repeatedPassword = '';

this.snackBar.open('Passwort geändert', 'Ok');

});

}

public logout() {

this.authService.clearCredentials();

this.snackBar.open('Erfolgreich abgemeldet', null, { duration: 3000 });

this.router.navigate(['']);

}

public openSetEmojiDialog(emotionId: number) {

this.dialog

.open(SelectEmojiDialogComponent)

.afterClosed()

.subscribe((emojiCode) => {

if (emojiCode) {

this.emotionServer

.setSmileyForEmotion(emotionId, emojiCode).subscribe(

() => this.loadEmotions(),

(error: HttpErrorResponse) => {

if (error.status === 400) {

this.snackBar.open('Das neu gewählte Emoji wird bereits verwendet', 'Ok');

}

});

} else {

// Cancelled

}

});

}

public openNewEmojiDialog() {

this.dialog

.open(SelectEmojiDialogComponent)

.afterClosed()

.subscribe((emojiCode) => {

if (emojiCode) {

this.emotionServer

.addNewEmotion(emojiCode).subscribe(

() => this.loadEmotions(),

(error: HttpErrorResponse) => {

if (error.status === 400) {

this.snackBar.open('Das gewählte Emoji wird bereits verwendet', 'Ok');

}

});

} else {

// Cancelled

}

});

}

public tryRefreshChart() {

// Ensure dates are valid

if (this.fromDate > this.toDate) {

this.snackBar.open('Das Start-Datum muss kleiner als das Bis-Datum sein', 'Ok');

} else {

this.emotionalStateServer

.groupedEmotionalStatesWithinRange(this.fromDate, this.toDate)

.subscribe((data) => {

// Create an entry for each emotion / emoji

this.chartData = this.allEmotions.map((emotion) => (

{

name: String.fromCodePoint(parseInt(emotion.smiley, 16)),

emotionId: emotion.id,

series: []

}));

// Iterate through all dates between the start and end date

let incrementingDate = new Date(this.fromDate.getTime());

incrementingDate.setHours(0, 0, 0, 0);

while (incrementingDate <= this.toDate) {

// Load all emotion-groups which belong to this date

const emotionsAtDate = data.filter((d) => {

const parsed = new Date(d.createdDate);

parsed.setHours(0, 0, 0, 0);

return parsed.getTime() === incrementingDate.getTime();

});

// For each emotion which has no emotion at the date, add one with a count of 0

for (const emotion of this.allEmotions

.filter((e) => !emotionsAtDate.find((emo) => emo.emotionId === e.id))) {

emotionsAtDate.push({

emotionId: emotion.id,

createdDate: incrementingDate,

count: 0});

}

const formattedDate = incrementingDate.toISOString().substring(0, 10);

// Add the count to the emoji for each date

for (const dailyEmotion of emotionsAtDate) {

this.chartData

.find((c) => c.emotionId === dailyEmotion.emotionId)

.series.push({

name: formattedDate,

value: dailyEmotion.count

});

}

// Increase the current date

incrementingDate = new Date(incrementingDate.getTime() + oneDayInMiliseconds);

}

});

}

}

}

### emoji-display.component.css

img {

    width: 100%;

}

### emoji-display.component.html

<img [alt]="emojiFromCodePoint" src="https://twemoji.maxcdn.com/svg/{{emojiCodePoint}}.svg">

### emoji-display.component.ts

import { Component, Input, ChangeDetectionStrategy } from '@angular/core';

@Component({

selector: 'emoji-display',

styleUrls: ['./emoji-display.component.css'],

templateUrl: './emoji-display.component.html',

changeDetection: ChangeDetectionStrategy.OnPush

})

export class EmojiDisplayComponent {

@Input()

public emojiCodePoint: string;

public get emojiFromCodePoint(): string {

return String.fromCodePoint(parseInt(this.emojiCodePoint, 16));

}

}

### error-dialog.component.ts

<h2 mat-dialog-title>Etwas ist schief gelaufen</h2>

<mat-dialog-content>Bitte versuchen Sie es später erneut</mat-dialog-content>

<mat-dialog-actions>

<button mat-button mat-dialog-close>Ok</button>

</mat-dialog-actions>

### error-dialog.component.ts

import { Component } from '@angular/core';

@Component({

styleUrls: ['./error-dialog.component.css'],

templateUrl: './error-dialog.component.html'

})

export class ErrorDialogComponent {

}

### can-activate-via-auth.guard.ts

import { Injectable } from '@angular/core';

import { CanActivate, Router } from '@angular/router';

import { AuthService } from './../services/auth.service';

@Injectable()

export class CanActivateViaAuthGuard implements CanActivate {

constructor(private router: Router, private authService: AuthService) {}

public canActivate() {

const usernameDefined = this.authService.username != null && this.authService.username !== '';

if (usernameDefined) {

return true;

}

this.router.navigate(['login']);

return false;

}

}

### home.component.css

.emotion-view-container {

    width: 400px;

}

.comment-input {

    width: 50%;

}

.select-emotion-button {

    margin-top: 20px;

    padding-top: 16px;

    width: 90px;

    height: 90px;

}

### home.component.html

<div style="align-items: center" \*ngIf="activeEmotions && dailyEmotionalStates; else noDataTemplate" fxLayout="column" fxLayoutGap="20px">

<h2 class="mat-display-1">Wie ist die allgemeine Gefühlslage?</h2>

<ng-container \*ngIf="dailyEmotionalStates.length === 0; else hasEmotionalStatesTemplate">

<h3>Heute wurden noch keine Gefühlslagen erfasst</h3>

</ng-container>

<ng-template #hasEmotionalStatesTemplate>

<div class="emotion-view-container" fxLayoutAlign="center" fxLayout="row" fxLayoutGap="20px">

<div fxFlex="{{relativeSize(emotion.id)}}%" \*ngFor="let emotion of activeEmotions" [matTooltip]="randomComment(emotion.id)">

<emoji-display [emojiCodePoint]="emotion.smileyCode"></emoji-display>

</div>

</div>

</ng-template>

<h2 class="mat-display-1">Eigene Gefühlslage erfassen</h2>

<div fxfLayout="row wrap" fxLayoutGap="20px">

<button mat-raised-button class="select-emotion-button" [color]="selectedEmotionId == emotion.id ? 'primary' : 'default'" \*ngFor="let emotion of activeEmotions" (click)="selectedEmotionId = emotion.id">

<emoji-display [emojiCodePoint]="emotion.smileyCode"></emoji-display>

</button>

</div>

<mat-form-field class="comment-input">

<textarea matInput placeholder="Kommentar" [formControl]="commentFormControl" [(ngModel)]="comment" maxLength="250" pattern="[A-Za-z0-9 \-\+\\*%&/\\()=?!:;.,üöäÄÖÜÊêéèà<>^]\*"></textarea>

<mat-hint align="end">{{comment.length}} / 250</mat-hint>

<mat-error \*ngIf="commentFormControl.hasError('pattern')">

Bitte keine ungewöhnlichen Zeichen verwenden

</mat-error>

</mat-form-field>

<button mat-raised-button [disabled]="commentFormControl.invalid || selectedEmotionId == null || saveBlocked" (click)="sendEmotion()">

Senden

</button>

<footer>

<a routerLink="administration">Administration</a> | Emoji-Grafiken ermöglich durch <a href="https://twitter.github.io/twemoji/">Twemoji</a>

</footer>

</div>

<ng-template #noDataTemplate>

Lade Daten...

</ng-template>

### home.component.ts

import { Component } from '@angular/core';

import { EmotionalStateService } from './../services/emotional-state.service';

import { EmotionService } from './../services/emotion.service';

import { EmotionalState } from './../model/emotional-state.model';

import { Emotion } from './../model/emotion.model';

import { FormControl } from '@angular/forms';

import { MatSnackBar } from '@angular/material';

import { HttpErrorResponse } from '@angular/common/http';

// 1 Minute

const reloadIntervalInMs = 60000;

// 1 Minute

const reloadTimeBlock = 60000;

@Component({

selector: 'home',

styleUrls: ['./home.component.css'],

templateUrl: './home.component.html'

})

export class HomeComponent {

public activeEmotions: Emotion[];

public dailyEmotionalStates: EmotionalState[];

public selectedEmotionId: number | null = null;

public comment: string = '';

public commentFormControl: FormControl = new FormControl('');

public saveBlocked: boolean = false;

constructor(private emotionServer: EmotionService,

private emotionalStateServer: EmotionalStateService,

private snackBar: MatSnackBar) {

this.loadData();

setInterval(() => this.loadData(), reloadIntervalInMs);

}

public relativeSize(emotionId: number): number {

const totalAmount = this.dailyEmotionalStates.length;

const emotionAmount = this.dailyEmotionalStates.filter((d) => d.emotionId === emotionId).length;

return 100 / totalAmount \* emotionAmount;

}

public randomComment(emotionId: number): string {

const emotion = this.dailyEmotionalStates

.find((de) => de.emotionId === emotionId && de.comment != null);

return emotion !== undefined ? emotion.comment : undefined;

}

public sendEmotion() {

this.emotionalStateServer.addEmotionalState(this.selectedEmotionId, this.comment)

.subscribe(() => {

// Load new data

this.loadData();

// Reset user input

this.comment = '';

this.selectedEmotionId = undefined;

// Inform user

const snackRef = this.snackBar.open(

'Gesendet - Bitte warten Sie, bis Sie weitere Gefühlslagen erfassen können');

this.saveBlocked = true;

setTimeout(() => {

this.saveBlocked = false;

snackRef.dismiss();

}, reloadTimeBlock);

}, (error: HttpErrorResponse) => {

// In case the sent emoji has been disabled in the meantime

if (error.status === 400) {

this.loadData();

this.snackBar.open('Bitte versuchen Sie es erneut', 'Ok');

}

});

}

// Thanks to https://stackoverflow.com/a/12646864

private shuffleArray<T>(array: T[]): T[] {

for (let i = array.length - 1; i > 0; i--) {

const j = Math.floor(Math.random() \* (i + 1));

[array[i], array[j]] = [array[j], array[i]];

}

return array;

}

private loadData() {

this.emotionServer

.activeEmotions().subscribe((e) => this.activeEmotions = e);

this.emotionalStateServer

.dailyEmotionalStates().subscribe((de) => this.dailyEmotionalStates = this.shuffleArray(de));

}

}

### authorize.interceptor.ts

import { Injectable } from '@angular/core';

import {

HttpRequest,

HttpHandler,

HttpEvent,

HttpInterceptor

} from '@angular/common/http';

import { AuthService } from './../services/auth.service';

import { Observable } from 'rxjs/Observable';

@Injectable()

export class AuthorizeInterceptor implements HttpInterceptor {

constructor(private auth: AuthService) {}

public intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

request = request.clone({

setHeaders: {

username: request.headers.get('username') || this.auth.username,

password: request.headers.get('password') || this.auth.password

}

});

return next.handle(request);

}

}

### error-handler.interceptor.ts

import { Injectable } from '@angular/core';

import {

HttpRequest,

HttpHandler,

HttpEvent,

HttpInterceptor,

HttpErrorResponse

} from '@angular/common/http';

import { Router } from '@angular/router';

import { MatDialog } from '@angular/material';

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/do';

import { AuthService } from './../services/auth.service';

import { ErrorDialogComponent } from '../error-dialog/error-dialog.component';

@Injectable()

export class ErrorHandlerInterceptor implements HttpInterceptor {

constructor(private auth: AuthService, private router: Router, private dialog: MatDialog) {}

public intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

return next.handle(request).do(() => undefined, (event) => {

if (event instanceof HttpErrorResponse) {

console.warn('Received not OK response with status ' + event.status, event);

if (event.status === 400) {

// Ignore requests concerning wrong user input

return;

}

if (request.url.endsWith('TestCredentials')) {

// Ignore login request

return;

}

if (event.status === 401) {

this.auth.clearCredentials();

this.router.navigate(['login']);

return;

}

this.dialog.open(ErrorDialogComponent);

}

});

}

}

### login.component.html

<div style="align-items: center" fxLayout="column" fxLayoutGap="20px">

<h2 class="mat-display-1">Administrator Login</h2>

<mat-form-field>

<input matInput placeholder="Benutzername" [formControl]="usernameFormControl" [(ngModel)]="username" required minLength="1" maxLength="100" pattern="[A-Za-z0-9 \-\+\\*%&/\\()=?!:;.,üöäÄÖÜÊêéèà<>^]\*">

<mat-error \*ngIf="usernameFormControl.hasError('pattern')">

Bitte keine ungewöhnlichen Zeichen verwenden

</mat-error>

<mat-error \*ngIf="usernameFormControl.hasError('minLength') || usernameFormControl.hasError('maxLength')">

Bitte einen Benutzernamen zwischen 1 und 100 Zeichen eingeben

</mat-error>

</mat-form-field>

<mat-form-field>

<input matInput placeholder="Password" type="password" [formControl]="passwordFormControl" [(ngModel)]="password" required minLength="1" maxLength="100">

</mat-form-field>

<button mat-raised-button [disabled]="usernameFormControl.invalid || passwordFormControl.invalid" (click)="login()">

Anmelden

</button>

</div>

### login.component.ts

import { Component } from '@angular/core';

import { FormControl } from '@angular/forms';

import { MatSnackBar } from '@angular/material';

import { UserService } from './../services/user.service';

import { AuthService } from './../services/auth.service';

import { HttpErrorResponse } from '@angular/common/http';

import { Router } from '@angular/router';

@Component({

selector: 'login',

styleUrls: ['./login.component.css'],

templateUrl: './login.component.html'

})

export class LoginComponent {

public username: string = '';

public password: string = '';

public usernameFormControl: FormControl = new FormControl('');

public passwordFormControl: FormControl = new FormControl('');

constructor(private userServer: UserService,

private authService: AuthService,

private snackBar: MatSnackBar,

private router: Router) { }

public login() {

this.userServer.testCredentials(this.username, this.password)

.subscribe(() => {

this.authService.username = this.username;

this.authService.password = this.password;

this.router.navigate(['administration']);

}, (error: HttpErrorResponse) => {

if (error.status === 401) {

this.snackBar.open('Invalide Anmeldedaten', 'Ok');

}

});

}

}

### administrator.model.ts

export interface Administrator {

id: number;

username: string;

}

### emotion.model.ts

export interface Emotion {

id: number;

smileyCode: string;

}

### emotional-state.model.ts

export interface EmotionalState {

emotionId: number;

createdDate: Date;

comment: string;

// TODO Ort: Add Location Id

}

### full-emotion.model.ts

export interface FullEmotion {

id: number;

smiley: string;

isActive: boolean;

}

### grouped-emotional-state.model.ts

export interface GroupedEmotionalState {

emotionId: number;

createdDate: Date;

count: number;

}

### link.model.ts

export interface Link {

url: string[];

matIcon: string;

content: string;

}

### no-content.component.html

<div>

<h1>Seite nicht gefunden</h1>

</div>

### no-content.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'no-content',

styleUrls: ['./no-content.component.css'],

templateUrl: './no-content.component.html'

})

export class NoContentComponent {

}

### select-emoji-dialog.component.css

.emoji-preview {

    width: 50px;

}

### select-emoji-dialog.component.ts

<div style="align-items: center" fxLayout="column" fxLayoutGap="20px">

<i

[emojiPickerIf]="true"

emojiPickerDirection="top"

[emojiPickerPreserveSelection]="true"

[emojiPickerAutofocus]="true"

(emojiPickerSelect)="setSelectedEmoji($event.char)"></i>

<p>Vorschau</p>

<emoji-display class="emoji-preview" \*ngIf="selectedEmojiCodePoint" [emojiCodePoint]="selectedEmojiCodePoint"></emoji-display>

<button mat-raised-button (click)="save()">

Speichern

</button>

<button mat-raised-button (click)="close()">

Abbrechen

</button>

</div>

### select-emoji-dialog.component.ts

import { Component } from '@angular/core';

import { MatDialogRef } from '@angular/material';

@Component({

styleUrls: ['./select-emoji-dialog.component.css'],

templateUrl: './select-emoji-dialog.component.html'

})

export class SelectEmojiDialogComponent {

public selectedEmojiCodePoint: string = '';

constructor(public dialogRef: MatDialogRef<SelectEmojiDialogComponent>) { }

public setSelectedEmoji(value: string) {

this.selectedEmojiCodePoint = value.codePointAt(0).toString(16);

}

public save() {

this.dialogRef.close(this.selectedEmojiCodePoint);

}

public close() {

this.dialogRef.close();

}

}

### auth.service.ts

import { Injectable } from '@angular/core';

const usernameStorageKey = 'sessionUsername';

const passwordStorageKey = 'sessionPassword';

@Injectable()

export class AuthService {

private \_username: string;

private \_password: string;

constructor() {

this.\_username = sessionStorage.getItem(usernameStorageKey) || '';

this.\_password = sessionStorage.getItem(passwordStorageKey) || '';

}

public get username(): string {

return this.\_username;

}

public set username(value: string) {

this.\_username = value;

sessionStorage.setItem(usernameStorageKey, value);

}

public get password(): string {

return this.\_password;

}

public set password(value: string) {

this.\_password = value;

sessionStorage.setItem(passwordStorageKey, value);

}

public clearCredentials() {

this.\_username = '';

this.\_password = '';

sessionStorage.removeItem(usernameStorageKey);

sessionStorage.removeItem(passwordStorageKey);

}

}

### emotion.service.ts

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { ServerService } from './server.service';

import { Emotion } from './../model/emotion.model';

import { FullEmotion } from './../model/full-emotion.model';

@Injectable()

export class EmotionService extends ServerService {

constructor(private httpClient: HttpClient) {

super('Emotion');

}

public activeEmotions() {

return this.httpClient.get<Emotion[]>(this.baseUrl + '/ActiveEmotions');

}

public allEmotions() {

return this.httpClient.get<FullEmotion[]>(this.baseUrl + '/AllEmotions');

}

public setActiveForEmotion(emotionId: number, isActive: boolean) {

return this.httpClient

.post<void>(this.baseUrl + '/SetActiveForEmotion', { emotionId, isActive });

}

public setSmileyForEmotion(emotionId: number, newSmileyCode: string) {

return this.httpClient

.post<void>(this.baseUrl + '/SetSmileyForEmotion', { emotionId, newSmileyCode });

}

public addNewEmotion(newSmileyCode: string) {

return this.httpClient

.post<void>(this.baseUrl + '/AddNewEmotion', { newSmileyCode });

}

}

### emotional-state.service.ts

import { Injectable } from '@angular/core';

import { HttpClient, HttpParams } from '@angular/common/http';

import { ServerService } from './server.service';

import { EmotionalState } from './../model/emotional-state.model';

import { GroupedEmotionalState } from './../model/grouped-emotional-state.model';

@Injectable()

export class EmotionalStateService extends ServerService {

constructor(private httpClient: HttpClient) {

super('EmotionalState');

}

public dailyEmotionalStates() {

return this.httpClient.get<EmotionalState[]>(this.baseUrl + '/DailyEmotionalStates');

}

public groupedEmotionalStatesWithinRange(from: Date, to: Date) {

const params = new HttpParams()

.set('from', from.toISOString())

.set('to', to.toISOString());

return this.httpClient.get<GroupedEmotionalState[]>(

this.baseUrl + '/GroupedEmotionalStatesWithinRange', { params });

}

public addEmotionalState(emotionId: number, comment: string) {

const data = comment.length !== 0 ? { emotionId, comment } : { emotionId };

return this.httpClient

.post<void>(this.baseUrl + '/AddEmotionalState', data);

}

}

### server.service.ts

import { isDevMode } from '@angular/core';

const baseUrlDebug: string = 'http://localhost:49484/api/';

// TODO Prod port 80?

const baseUrlProd: string = 'http://localhost:49484/api/';

export abstract class ServerService {

protected readonly baseUrl: string;

constructor(controllerPrefix: string) {

this.baseUrl = (isDevMode() ? baseUrlDebug : baseUrlProd) + controllerPrefix;

}

}

### user.service.ts

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { ServerService } from './server.service';

@Injectable()

export class UserService extends ServerService {

constructor(private httpClient: HttpClient) {

super('User');

}

public testCredentials(username: string, password: string) {

return this.httpClient.get<void>(this.baseUrl + '/TestCredentials', {

headers: new HttpHeaders({

username,

password

})

});

}

public setNewUsername(newUsername: string) {

return this.httpClient.post<void>(this.baseUrl + '/SetNewUsername', { newUsername });

}

public setNewPassword(newPassword: string) {

return this.httpClient.post<void>(this.baseUrl + '/SetNewPassword', { newPassword });

}

}

### app.component.html

<mat-toolbar class="app-toolbar mat-primary mat-elevation-z6" role="navigation">

<div fxFlex class="app-title">Atos Happy Meter</div>

</mat-toolbar>

<main>

<div class="main-content mat-typography" role="main">

<router-outlet></router-outlet>

</div>

</main>

### app.component.scss

$content-padding: 16px;

$title-padding: 20px;

$menu-margin: 6px;

.app-toolbar {

color: white !important;

text-transform: uppercase;

text-align: center;

z-index: 10;

position: relative;

}

.app-title {

text-align: left;

font-weight: bold;

}

.nav-flex-div {

width: 100%;

}

.main-title {

text-align: center;

padding-top: $title-padding;

padding-bottom: $title-padding;

background: #3f51b5;

color: hsla(0,0%,100%,.87);

padding-left: $content-padding;

padding-right: $content-padding;

}

.main-content {

padding-left: $content-padding;

padding-right: $content-padding;

padding-bottom: $content-padding;

}

### app.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app',

styleUrls: [ './app.component.scss' ],

templateUrl: './app.component.html'

})

export class AppComponent {

}

### app.module.ts

import { FlexLayoutModule } from '@angular/flex-layout';

import { BrowserModule, Title } from '@angular/platform-browser';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { NgModule, LOCALE\_ID } from '@angular/core';

import { RouterModule, PreloadAllModules } from '@angular/router';

import { HttpClientModule } from '@angular/common/http';

// External libraries

import { LineChartModule } from '@swimlane/ngx-charts';

import { EmojiPickerModule } from 'angular2-emoji-picker';

/\*

\* Platform and Environment providers/directives/pipes

\*/

import { ENV\_PROVIDERS } from './environment';

import { ROUTES } from './app.routes';

// App is our top level component

import { AppComponent } from './app.component';

// Different sites

import { HomeComponent } from './home/home.component';

import { LoginComponent } from './login/login.component';

import { AdministrationComponent } from './administration/administration.component';

// Dialog contents

import { ErrorDialogComponent } from './error-dialog/error-dialog.component';

import { SelectEmojiDialogComponent } from './select-emoji-dialog/select-emoji-dialog.component';

// Help components

import { EmojiDisplayComponent } from './emoji-display/emoji-display.component';

// 404 not found page

import { NoContentComponent } from './no-content/no-content.component';

// Services

import { AuthService } from './services/auth.service';

import { UserService } from './services/user.service';

import { EmotionalStateService } from './services/emotional-state.service';

import { EmotionService } from './services/emotion.service';

// Interceptors

import { AuthorizeInterceptor } from './interceptors/authorize.interceptor';

import { ErrorHandlerInterceptor } from './interceptors/error-handler.interceptor';

// Guards

import { CanActivateViaAuthGuard } from './guards/can-activate-via-auth.guard';

// Material 2

import {

MatButtonModule,

MatToolbarModule,

MatDialogModule,

MatTooltipModule,

MatInputModule,

MatSnackBarModule,

MatDatepickerModule,

MatNativeDateModule,

MatTabsModule

} from '@angular/material';

import 'hammerjs';

import '../styles/styles.scss';

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

/\*\*

\* `AppModule` is the main entry point into Angular2's bootstraping process

\*/

@NgModule({

bootstrap: [ AppComponent ],

declarations: [

AppComponent,

HomeComponent,

LoginComponent,

AdministrationComponent,

EmojiDisplayComponent,

ErrorDialogComponent,

SelectEmojiDialogComponent,

NoContentComponent

],

entryComponents: [

ErrorDialogComponent,

SelectEmojiDialogComponent

],

/\*\*

\* Import Angular's modules.

\*/

imports: [

BrowserModule,

BrowserAnimationsModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

RouterModule.forRoot(ROUTES, {

useHash: true,

preloadingStrategy: PreloadAllModules

}),

MatInputModule,

MatButtonModule,

MatToolbarModule,

MatDialogModule,

MatTooltipModule,

MatSnackBarModule,

MatDatepickerModule,

MatNativeDateModule,

MatTabsModule,

FlexLayoutModule,

LineChartModule,

EmojiPickerModule.forRoot()

],

/\*\*

\* Expose our Services and Providers into Angular's dependency injection.

\*/

providers: [

ENV\_PROVIDERS,

Title,

UserService,

EmotionalStateService,

EmotionService,

AuthService,

CanActivateViaAuthGuard,

{

provide: LOCALE\_ID,

useValue: 'de-CH'

},

{

provide: HTTP\_INTERCEPTORS,

useClass: AuthorizeInterceptor,

multi: true

},

{

provide: HTTP\_INTERCEPTORS,

useClass: ErrorHandlerInterceptor,

multi: true

}

]

})

export class AppModule {

}

### app.routes.ts

import { Routes } from '@angular/router';

import { HomeComponent } from './home/home.component';

import { NoContentComponent } from './no-content/no-content.component';

import { CanActivateViaAuthGuard } from './guards/can-activate-via-auth.guard';

import { LoginComponent } from './login/login.component';

import { AdministrationComponent } from './administration/administration.component';

export const ROUTES: Routes = [

{ path: '', component: HomeComponent },

{ path: 'login', component: LoginComponent },

{ path: 'administration', component: AdministrationComponent,

canActivate: [CanActivateViaAuthGuard] },

{ path: '\*\*', component: NoContentComponent }

];

### index.html

<!DOCTYPE html>

<html lang="de-CH">

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="theme-color" content="#3f51b5">

<title><%= htmlWebpackPlugin.options.title %></title>

<!-- base url -->

<base href="<%= htmlWebpackPlugin.options.metadata.baseUrl %>">

<% if (webpackConfig.htmlElements.headTags) { %>

<!-- Configured Head Tags -->

<%= webpackConfig.htmlElements.headTags %>

<% } %>

<%= htmlWebpackPlugin.files.webpackManifest %>

<% if (htmlWebpackPlugin.options.metadata.isDevServer && htmlWebpackPlugin.options.metadata.HMR !== true) { %>

<!-- Webpack Dev Server reload -->

<script src="/webpack-dev-server.js"></script>

<% } %>

<!-- CSS will be injected by webpack here -->

<!-- Preload link tags will be injected by webpack here -->

</head>

<body class="mat-typography">

<app>

Lade...

</app>

<!-- Scripts will be injected by webpack here -->

</body>

</html>

### package.json

{

"name": "atos-happy-meter",

"version": "0.0.1",

"private": true,

"description": "Eine Anwendung zum überprüfen der Glücklichkeit seiner Mitarbeiter",

"keywords": [

"angular",

"angular2",

"angular4",

"webpack",

"typescript"

],

"author": "Pascal Honegger <pascal.honegger@bluewin.ch>",

"license": "MIT",

"scripts": {

"build:aot:prod": "npm run clean:dist && npm run clean:aot && cross-env BUILD\_AOT=1 npm run webpack -- --config config/webpack.prod.js --progress --profile --bail",

"build:aot": "npm run build:aot:prod",

"build:dev": "npm run clean:dist && npm run webpack -- --config config/webpack.dev.js --progress --profile",

"build:prod": "npm run clean:dist && npm run webpack -- --config config/webpack.prod.js --progress --profile --bail",

"build": "npm run build:dev",

"clean:dll": "npm run rimraf -- dll",

"clean:aot": "npm run rimraf -- compiled",

"clean:dist": "npm run rimraf -- dist",

"clean:install": "npm set progress=false && npm install",

"clean": "npm cache clean --force && npm run rimraf -- node\_modules doc coverage dist compiled dll",

"lint": "npm run tslint \"src/\*\*/\*.ts\"",

"node": "node",

"preclean:install": "npm run clean",

"rimraf": "rimraf",

"server:dev:hmr": "npm run server:dev -- --hotOnly",

"server:dev": "npm run webpack-dev-server -- --config config/webpack.dev.js --open --progress --profile --watch --content-base src/",

"server:prod": "http-server dist -c-1 --cors",

"server": "npm run server:dev",

"start:hmr": "npm run server:dev:hmr",

"start": "npm run server:dev",

"tslint": "tslint",

"watch:dev:hmr": "npm run watch:dev -- --hot",

"watch:dev": "npm run build:dev -- --watch",

"watch:prod": "npm run build:prod -- --watch",

"watch": "npm run watch:dev",

"webpack-dev-server": "node --max\_old\_space\_size=4096 node\_modules/webpack-dev-server/bin/webpack-dev-server.js",

"webpack": "node --max\_old\_space\_size=4096 node\_modules/webpack/bin/webpack.js"

},

"dependencies": {

"@angular/animations": "4.4.6",

"@angular/cdk": "2.0.0-beta.12",

"@angular/common": "4.4.6",

"@angular/compiler": "4.4.6",

"@angular/core": "4.4.6",

"@angular/flex-layout": "2.0.0-beta.9",

"@angular/forms": "4.4.6",

"@angular/http": "4.4.6",

"@angular/material": "2.0.0-beta.12",

"@angular/platform-browser": "4.4.6",

"@angular/platform-browser-dynamic": "4.4.6",

"@angular/router": "4.4.6",

"@angularclass/hmr": "2.1.3",

"@angularclass/hmr-loader": "3.0.4",

"@swimlane/ngx-charts": "^6.1.0",

"angular2-emoji-picker": "^1.4.6",

"core-js": "2.5.1",

"d3": "^4.11.0",

"hammerjs": "2.0.8",

"http-server": "0.10.0",

"ie-shim": "0.1.0",

"reflect-metadata": "0.1.10",

"rxjs": "5.5.2",

"web-animations-js": "2.3.1",

"zone.js": "0.8.18"

},

"devDependencies": {

"@angular/compiler-cli": "4.4.6",

"@types/hammerjs": "2.0.35",

"@types/node": "8.0.47",

"@types/source-map": "0.5.2",

"@types/uglify-js": "2.6.29",

"@types/webpack": "3.0.14",

"add-asset-html-webpack-plugin": "2.1.2",

"angular2-template-loader": "0.6.2",

"assets-webpack-plugin": "3.5.1",

"awesome-typescript-loader": "3.2.3",

"codelyzer": "3.2.2",

"copy-webpack-plugin": "4.2.0",

"cross-env": "5.1.1",

"css-loader": "0.28.7",

"exports-loader": "0.6.4",

"expose-loader": "0.7.3",

"extract-text-webpack-plugin": "3.0.2",

"file-loader": "1.1.5",

"find-root": "1.1.0",

"html-webpack-plugin": "2.30.1",

"imports-loader": "0.7.1",

"inline-manifest-webpack-plugin": "3.0.1",

"ng-router-loader": "2.1.0",

"ngc-webpack": "3.2.2",

"node-sass": "4.5.3",

"optimize-js-plugin": "0.0.4",

"preload-webpack-plugin": "2.0.0",

"raw-loader": "0.5.1",

"rimraf": "2.6.2",

"sass-loader": "6.0.6",

"script-ext-html-webpack-plugin": "1.8.7",

"source-map-loader": "0.2.3",

"string-replace-loader": "1.3.0",

"style-loader": "0.19.0",

"to-string-loader": "1.1.5",

"ts-node": "3.3.0",

"tslib": "1.8.0",

"tslint": "5.8.0",

"tslint-loader": "3.5.3",

"typescript": "2.5.3",

"uglifyjs-webpack-plugin": "1.0.1",

"url-loader": "0.6.2",

"webpack": "3.8.1",

"webpack-dev-middleware": "1.12.0",

"webpack-dev-server": "2.9.3",

"webpack-dll-bundles-plugin": "1.0.0-beta.5",

"webpack-merge": "4.1.0"

},

"repository": {

"type": "git",

"url": "https://github.com/PascalHonegger/HappyMeter.git"

},

"bugs": {

"url": "https://github.com/PascalHonegger/HappyMeter/issues"

},

"engines": {

"node": ">= 8.9.0",

"npm": ">= 5.5.1"

}

}

## Backend

### WebApiConfig.cs

using System.Web.Http;

using System.Web.Http.Cors;

namespace AtosHappyMeter

{

    public static class WebApiConfig

    {

        public static void Register(HttpConfiguration config)

        {

            // Web API configuration and services

            var enableForAllAttribute = new EnableCorsAttribute("\*", "\*", "\*");

            config.EnableCors(enableForAllAttribute);

            // Web API routes

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(

                "DefaultApi",

                "api/{controller}/{action}",

                new {controller = "Home", action = "Index"}

            );

        }

    }

}

### AuthorizeAdministratorAttribute.cs

using System.Linq;

using System.Web.Http;

using System.Web.Http.Controllers;

using AtosHappyMeter.Constants;

using AtosHappyMeter.Services;

namespace AtosHappyMeter.Attributes

{

    public class AuthorizeAdministratorAttribute : AuthorizeAttribute

    {

        protected override bool IsAuthorized(HttpActionContext actionContext)

        {

            var headers = actionContext.Request.Headers;

            if (!headers.Contains(AuthorizationConstants.UsernameKey) || !headers.Contains(AuthorizationConstants.PasswordKey))

            {

                // No credentials provided

                return false;

            }

            var username = headers.GetValues(AuthorizationConstants.UsernameKey).First();

            var password = headers.GetValues(AuthorizationConstants.PasswordKey).First();

            var (isAuthorized, foundUser) = LoginService.AuthorizeUser(username, password);

            if (!isAuthorized)

            {

                return false;

            }

            actionContext.Request.Properties[AuthorizationConstants.UserInformationKey] = foundUser;

            return true;

        }

    }

}

### AuthorizationConstants.cs

namespace AtosHappyMeter.Constants

{

    public static class AuthorizationConstants

    {

        public const string UsernameKey = "username";

        public const string PasswordKey = "password";

        public const string UserInformationKey = "UserInformation";

    }

}

### RegularExpressions.cs

namespace AtosHappyMeter.Constants

{

    public static class RegularExpressions

    {

        public const string NoFunkyCharactersRegex = @"^[A-Za-z0-9 \-\+\\*%&/\\()=?!:;.,üöäÄÖÜÊêéèà<>^]\*$";

    }

}

### AddEmotionalStateDto.cs

using System.ComponentModel.DataAnnotations;

using AtosHappyMeter.Constants;

namespace AtosHappyMeter.ControllerModel

{

    public class AddEmotionalStateDto

    {

        [Required]

        public int EmotionId { get; set; }

        [MaxLength(250)]

        [RegularExpression(RegularExpressions.NoFunkyCharactersRegex)]

        public string Comment { get; set; }

        //TODO Ort: Add Location Id

    }

}

### AddNewEmotionDto.cs

using System.ComponentModel.DataAnnotations;

namespace AtosHappyMeter.ControllerModel

{

    public class AddNewEmotionDto

    {

        [Required]

        [MaxLength(6)]

        public string NewSmileyCode { get; set; }

    }

}

### ChangeEmotionIsActiveDto.cs

using System.ComponentModel.DataAnnotations;

namespace AtosHappyMeter.ControllerModel

{

    public class ChangeEmotionIsActiveDto

    {

        [Required]

        public int EmotionId { get; set; }

        [Required]

        public bool IsActive { get; set; }

    }

}

### ChangeEmotionSmileyDto.cs

using System.ComponentModel.DataAnnotations;

namespace AtosHappyMeter.ControllerModel

{

    public class ChangeEmotionSmileyDto

    {

        [Required]

        public int EmotionId { get; set; }

        [Required]

        [MaxLength(6)]

        public string NewSmileyCode { get; set; }

    }

}

### GroupedEmotionalState.cs

using System;

namespace AtosHappyMeter.ControllerModel

{

    public class GroupedEmotionalState

    {

        public int EmotionId { get; set; }

        public DateTime CreatedDate { get; set; }

        public int Count { get; set; }

    }

}

### ReducedEmotion.cs

namespace AtosHappyMeter.ControllerModel

{

    public class ReducedEmotion

    {

        public int Id { get; set; }

        public string SmileyCode { get; set; }

    }

}

### ReducedEmotionalState.cs

using System;

namespace AtosHappyMeter.ControllerModel

{

    public class ReducedEmotionalState

    {

        public int EmotionId { get; set; }

        public DateTime CreatedDate { get; set; }

        public string Comment { get; set; }

        //TODO Ort: Add Location Id

    }

}

### SetPasswordDto.cs

using System.ComponentModel.DataAnnotations;

namespace AtosHappyMeter.ControllerModel

{

    public class SetPasswordDto

    {

        [MinLength(3)]

        [MaxLength(100)]

        public string NewPassword { get; set; }

    }

}

### SetUsernameDto.cs

using System.ComponentModel.DataAnnotations;

using AtosHappyMeter.Constants;

namespace AtosHappyMeter.ControllerModel

{

    public class SetUsernameDto

    {

        [MinLength(1)]

        [MaxLength(100)]

        [RegularExpression(RegularExpressions.NoFunkyCharactersRegex)]

        public string NewUsername { get; set; }

    }

}

### EmotionalStateController.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using AtosHappyMeter.Attributes;

using AtosHappyMeter.ControllerModel;

using AtosHappyMeter.Models;

namespace AtosHappyMeter.Controllers

{

    public class EmotionalStateController : ApiController

    {

        [HttpGet]

        [ResponseType(typeof(List<ReducedEmotionalState>))]

        public async Task<IHttpActionResult> DailyEmotionalStates()

        {

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var currentDate = DateTime.Now.Date;

                var data = await dbContext.EmotionalStates

                    .Where(e => e.Emotion.IsActive && e.CreatedDate == currentDate)

                    .Select(e => new ReducedEmotionalState

                    {

                        Comment = e.Comment,

                        CreatedDate = e.CreatedDate,

                        EmotionId = e.EmotionId

                    })

                    .ToListAsync();

                return Ok(data);

            }

        }

        [HttpGet]

        [ResponseType(typeof(List<GroupedEmotionalState>))]

        [AuthorizeAdministrator]

        public async Task<IHttpActionResult> GroupedEmotionalStatesWithinRange([Required] DateTime from, [Required] DateTime to)

        {

            // Validate parameters

            if (!ModelState.IsValid || from > to)

            {

                return BadRequest();

            }

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var data = await dbContext.EmotionalStates

                    .Where(e => e.CreatedDate >= from.Date && e.CreatedDate <= to.Date)

                    .GroupBy(e => new { e.EmotionId, e.CreatedDate })

                    .Select(group => new GroupedEmotionalState

                    {

                        CreatedDate = group.Key.CreatedDate,

                        EmotionId = group.Key.EmotionId,

                        Count = group.Count()

                    })

                    .ToListAsync();

                return Ok(data);

            }

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> AddEmotionalState([FromBody] AddEmotionalStateDto addEmotionalStateDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                // The provided Emotion has to exist and be active

                if (!await dbContext.Emotions.AnyAsync(e => e.Id == addEmotionalStateDto.EmotionId && e.IsActive))

                {

                    return BadRequest();

                }

                dbContext.EmotionalStates.Add(new EmotionalState

                {

                    EmotionId = addEmotionalStateDto.EmotionId,

                    Comment = addEmotionalStateDto.Comment,

                    CreatedDate = DateTime.Now.Date

                });

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

    }

}

### EmotionController.cs

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using AtosHappyMeter.Attributes;

using AtosHappyMeter.ControllerModel;

using AtosHappyMeter.Models;

namespace AtosHappyMeter.Controllers

{

    public class EmotionController : ApiController

    {

        [HttpGet]

        [ResponseType(typeof(List<ReducedEmotion>))]

        public async Task<IHttpActionResult> ActiveEmotions()

        {

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var data = await dbContext.Emotions

                    .Where(e => e.IsActive)

                    .Select(e => new ReducedEmotion

                    {

                        Id = e.Id,

                        SmileyCode = e.Smiley

                    })

                    .ToListAsync();

                return Ok(data);

            }

        }

        [HttpGet]

        [ResponseType(typeof(List<Emotion>))]

        [AuthorizeAdministrator]

        public async Task<IHttpActionResult> AllEmotions()

        {

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var data = await dbContext.Emotions.ToListAsync();

                return Ok(data);

            }

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthorizeAdministrator]

        public async Task<IHttpActionResult> SetActiveForEmotion([FromBody] ChangeEmotionIsActiveDto changeEmotionIsActiveDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var data = await dbContext.Emotions.FindAsync(changeEmotionIsActiveDto.EmotionId);

                if (data == null)

                {

                    return NotFound();

                }

                data.IsActive = changeEmotionIsActiveDto.IsActive;

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthorizeAdministrator]

        public async Task<IHttpActionResult> SetSmileyForEmotion([FromBody] ChangeEmotionSmileyDto changeEmotionSmileyDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                if (await dbContext.Emotions.AnyAsync(e => e.Smiley == changeEmotionSmileyDto.NewSmileyCode))

                {

                    return BadRequest();

                }

                var data = await dbContext.Emotions.FindAsync(changeEmotionSmileyDto.EmotionId);

                if (data == null)

                {

                    return NotFound();

                }

                data.Smiley = changeEmotionSmileyDto.NewSmileyCode;

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        [AuthorizeAdministrator]

        public async Task<IHttpActionResult> AddNewEmotion([FromBody] AddNewEmotionDto addNewEmotionDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                if (await dbContext.Emotions.AnyAsync(e => e.Smiley == addNewEmotionDto.NewSmileyCode))

                {

                    return BadRequest();

                }

                dbContext.Emotions.Add(new Emotion

                {

                    Smiley = addNewEmotionDto.NewSmileyCode,

                    IsActive = true

                });

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

    }

}

### HomeController.cs

using System;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace AtosHappyMeter.Controllers

{

    public class HomeController : ApiController

    {

        [HttpGet]

        [Route("")]

        public HttpResponseMessage Index()

        {

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

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

            return response;

        }

    }

}

### UserController.cs

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using AtosHappyMeter.Attributes;

using AtosHappyMeter.Constants;

using AtosHappyMeter.ControllerModel;

using AtosHappyMeter.Models;

using AtosHappyMeter.Services;

namespace AtosHappyMeter.Controllers

{

    [AuthorizeAdministrator]

    public class UserController : ApiController

    {

        [HttpGet]

        [ResponseType(typeof(void))]

        public IHttpActionResult TestCredentials()

        {

            // Authorization is done through the Attribute

            return StatusCode(HttpStatusCode.NoContent);

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> SetNewUsername([FromBody] SetUsernameDto setUsernameDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            var autorisedAdministrator =

                (Administrator) ActionContext.Request.Properties[AuthorizationConstants.UserInformationKey];

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                if (await dbContext.Administrators.AnyAsync(a => a.Username == setUsernameDto.NewUsername))

                {

                    return BadRequest();

                }

                dbContext.Administrators.Attach(autorisedAdministrator);

                autorisedAdministrator.UpdateUsername(setUsernameDto.NewUsername);

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

        [HttpPost]

        [ResponseType(typeof(void))]

        public async Task<IHttpActionResult> SetNewPassword([FromBody] SetPasswordDto setPasswordDto)

        {

            // Validate parameters

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

            {

                return BadRequest();

            }

            var autorisedAdministrator =

                (Administrator) ActionContext.Request.Properties[AuthorizationConstants.UserInformationKey];

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                dbContext.Administrators.Attach(autorisedAdministrator);

                autorisedAdministrator.UpdatePassword(setPasswordDto.NewPassword);

                await dbContext.SaveChangesAsync();

                return StatusCode(HttpStatusCode.NoContent);

            }

        }

    }

}

### HappyMeterDatabaseContrex.cs

using System;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity;

using MySql.Data.Entity;

// ReSharper disable UnusedAutoPropertyAccessor.Global

namespace AtosHappyMeter.Models

{

    [DbConfigurationType(typeof(MySqlEFConfiguration))]

    public class HappyMeterDatabaseContext : DbContext

    {

        public HappyMeterDatabaseContext() : base("name=AtosHappyMeterDatabase")

        {

        }

        public virtual DbSet<EmotionalState> EmotionalStates { get; set; }

        public virtual DbSet<Emotion> Emotions { get; set; }

        public virtual DbSet<Administrator> Administrators { get; set; }

    }

    [Table("emotional\_state")]

    public class EmotionalState

    {

        [Column("id")]

        public int Id { get; set; }

        [Column("emotion\_id")]

        public int EmotionId { get; set; }

        [ForeignKey(nameof(EmotionId))]

        public virtual Emotion Emotion { get; set; }

        [Column("created\_date")]

        public DateTime CreatedDate { get; set; }

        [Column("comment")]

        public string Comment { get; set; }

    }

    [Table("emotion")]

    public class Emotion

    {

        [Column("id")]

        public int Id { get; set; }

        [Column("smiley")]

        public string Smiley { get; set; }

        [Column("is\_active")]

        public bool IsActive { get; set; }

    }

    [Table("administrator")]

    public class Administrator

    {

        [Column("id")]

        public int Id { get; set; }

        [Column("username")]

        public string Username { get; set; }

        [Column("password\_hash")]

        public string PasswordHash { get; set; }

    }

}

### LoginService.cs

using System.Linq;

using AtosHappyMeter.Models;

namespace AtosHappyMeter.Services

{

    public static class LoginService

    {

        private static bool VerifyPassword(string text, string hash)

        {

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

        }

        private static string HashPassword(string password)

        {

            return BCrypt.Net.BCrypt.HashPassword(password);

        }

        public static (bool isAuthorized, Administrator foundAdministrator) AuthorizeUser(string username, string password)

        {

            using (var dbContext = new HappyMeterDatabaseContext())

            {

                var foundAdministrator = dbContext.Administrators.FirstOrDefault(a => a.Username == username);

                if (foundAdministrator == null)

                {

                    // Wrong username

                    return (false, null);

                }

                if (!VerifyPassword(password, foundAdministrator.PasswordHash))

                {

                    return (false, null);

                }

                return (true, foundAdministrator);

            }

        }

        public static Administrator CreateUser(string username, string password)

        {

            return new Administrator

            {

                Username = username,

                PasswordHash = HashPassword(password)

            };

        }

        public static void UpdateUsername(this Administrator administrator, string newUsername)

        {

            administrator.Username = newUsername;

        }

        public static void UpdatePassword(this Administrator administrator, string newPassword)

        {

            administrator.PasswordHash = HashPassword(newPassword);

        }

    }

}

### Global.asax.cs

using System.Net.Http.Formatting;

using System.Web;

using System.Web.Http;

using Newtonsoft.Json;

using Newtonsoft.Json.Serialization;

namespace AtosHappyMeter

{

    public class WebApiApplication : HttpApplication

    {

        protected void Application\_Start()

        {

            GlobalConfiguration.Configure(WebApiConfig.Register);

            var config = GlobalConfiguration.Configuration;

            config.Formatters.Clear();

            config.Formatters.Add(new JsonMediaTypeFormatter());

            config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();

            config.Formatters.JsonFormatter.SerializerSettings.Formatting = Formatting.Indented;

            config.Formatters.JsonFormatter.SerializerSettings.NullValueHandling = NullValueHandling.Ignore;

            config.Formatters.JsonFormatter.UseDataContractJsonSerializer = false;

        }

    }

}

### packages.config

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

<packages>

<package id="BCrypt-Official" version="0.1.109" targetFramework="net47" />

<package id="EntityFramework" version="6.2.0" targetFramework="net47" />

<package id="Microsoft.AspNet.Cors" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.AspNet.WebApi" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.AspNet.WebApi.Client" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.AspNet.WebApi.Core" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.AspNet.WebApi.Cors" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.AspNet.WebApi.WebHost" version="5.2.3" targetFramework="net47" />

<package id="Microsoft.CodeDom.Providers.DotNetCompilerPlatform" version="1.0.8" targetFramework="net47" />

<package id="Microsoft.Net.Compilers" version="2.4.0" targetFramework="net47" developmentDependency="true" />

<package id="MySql.Data" version="6.9.10" targetFramework="net47" />

<package id="MySql.Data.Entity" version="6.9.10" targetFramework="net47" />

<package id="Newtonsoft.Json" version="10.0.3" targetFramework="net47" />

</packages>

### 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>

    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

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

  </configSections>

  <appSettings></appSettings>

  <system.web>

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

    <httpRuntime targetFramework="4.7" />

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

  </system.web>

  <system.webServer>

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

        <bindingRedirect oldVersion="0.0.0.0-10.0.0.0" newVersion="10.0.0.0" />

      </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>

    </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="AtosHappyMeterDatabase" providerName="MySql.Data.MySqlClient" connectionString="server=localhost;port=3306;database=atoshappymeter;uid=root;password=root" />

  </connectionStrings>

  <entityFramework>

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

    <providers>

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

      <provider invariantName="MySql.Data.MySqlClient" type="MySql.Data.MySqlClient.MySqlProviderServices, MySql.Data.Entity.EF6"></provider></providers>

  </entityFramework>

<system.data>

<DbProviderFactories>

<remove invariant="MySql.Data.MySqlClient" />

<add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient" description=".Net Framework Data Provider for MySQL" type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data, Version=6.9.10.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d" />

</DbProviderFactories>

</system.data></configuration>

## Datenbank

### AtosHappyMeterExport.sql

-- MySQL Script generated by MySQL Workbench

-- Fri Nov 10 08:00:05 2017

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema AtosHappyMeter

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema AtosHappyMeter

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `AtosHappyMeter` ;

USE `AtosHappyMeter` ;

-- -----------------------------------------------------

-- Table `AtosHappyMeter`.`emotion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `AtosHappyMeter`.`emotion` (

`id` INT NOT NULL AUTO\_INCREMENT,

`smiley` VARCHAR(6) NOT NULL,

`is\_active` TINYINT NOT NULL DEFAULT 1,

PRIMARY KEY (`id`),

UNIQUE INDEX `Smiley\_UNIQUE` (`smiley` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `AtosHappyMeter`.`emotional\_state`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `AtosHappyMeter`.`emotional\_state` (

`id` INT NOT NULL AUTO\_INCREMENT,

`emotion\_id` INT NOT NULL,

`created\_date` DATE NOT NULL,

`comment` VARCHAR(250) NULL,

PRIMARY KEY (`id`),

INDEX `fk\_table1\_table2\_idx` (`emotion\_id` ASC),

CONSTRAINT `fk\_table1\_table2`

FOREIGN KEY (`emotion\_id`)

REFERENCES `AtosHappyMeter`.`emotion` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `AtosHappyMeter`.`administrator`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `AtosHappyMeter`.`administrator` (

`id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(100) NOT NULL,

`password\_hash` CHAR(60) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `Username\_UNIQUE` (`username` ASC))

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

-- -----------------------------------------------------

-- Data for table `AtosHappyMeter`.`administrator`

-- -----------------------------------------------------

START TRANSACTION;

USE `AtosHappyMeter`;

INSERT INTO `AtosHappyMeter`.`administrator` (`id`, `username`, `password\_hash`) VALUES (1, 'admin', '$10$8I3rMrPvn1VmkTV5.4iGi.QDeLpGw1fk.bn9./npHiAdlZE1e87iS');

COMMIT;