Erstellt durch: Seraphin Rihm

Probe-IPA Bericht

Inhaltsverzeichnis

## **Teil 1**

[Aufgabenstellung 1](#_Toc498098947)

[Projektbeschreibung 1](#_Toc498098948)

[Plattformen 1](#_Toc498098949)

[Funktionale Anforderungen 1](#_Toc498098950)

[Umfrage erstellen 1](#_Toc498098951)

[Umfrage ausfüllen 2](#_Toc498098952)

[Umfrage auswerten 2](#_Toc498098953)

[Registrierung und Login 2](#_Toc498098954)

[Benutzerrollen 2](#_Toc498098955)

[Projektorganisation 3](#_Toc498098956)

[Planungs-Methode 3](#_Toc498098957)

[Vorkenntnisse 3](#_Toc498098958)

[Vorbereitungen 3](#_Toc498098959)

[Firmenstandards 3](#_Toc498098960)

[Zeitplanung 4](#_Toc498098961)

[Meilensteine 5](#_Toc498098962)

[Kurzbeschrieb 5](#_Toc498098963)

[Projektauftrag 5](#_Toc498098964)

[Umsetzung 5](#_Toc498098965)

[Ergebnis 5](#_Toc498098966)

[Arbeitsprotokoll 6](#_Toc498098967)

[Mittwoch 01.11.17 6](#_Toc498098968)

[Donnerstag 02.11.17 7](#_Toc498098969)

[Freitag 03.11.17 8](#_Toc498098970)

[Mittwoch 08.11.17 9](#_Toc498098971)

[Freitag 10.11.17 10](#_Toc498098972)

## Teil 2

[Informieren 11](#_Toc498098923)

[Ist-Analyse 11](#_Toc498098924)

[Soll-Analyse 11](#_Toc498098925)

[Planen 12](#_Toc498098926)

[Benutzerkonzept 12](#_Toc498098927)

[Use-Cases 12](#_Toc498098928)

[Testkonzept 17](#_Toc498098929)

[Software-Architektur 21](#_Toc498098930)

[Datenbank 22](#_Toc498098931)

[Entscheiden 24](#_Toc498098932)

[Versionsverwaltung 24](#_Toc498098933)

[Angular Version 24](#_Toc498098934)

[Projekt Durchführbarkeit 24](#_Toc498098935)

[Realisieren 25](#_Toc498098936)

[Frontend 25](#_Toc498098937)

[Backend 29](#_Toc498098938)

[Kontrollieren 31](#_Toc498098939)

[Testprotokoll 31](#_Toc498098940)

[Testfazit 32](#_Toc498098941)

[Auswerten 33](#_Toc498098942)

[Reflexion 33](#_Toc498098943)

[Verbesserungen/Erfragungen 33](#_Toc498098944)

[Einsetzbarkeit 33](#_Toc498098945)

[Fazit 33](#_Toc498098946)

Anhang

[Abbildungsverzeichnis 34](#_Toc498098918)

[Tabellenverzeichniss 34](#_Toc498098919)

[Glossar 35](#_Toc498098920)

[Frontend 36](#_Toc498098921)

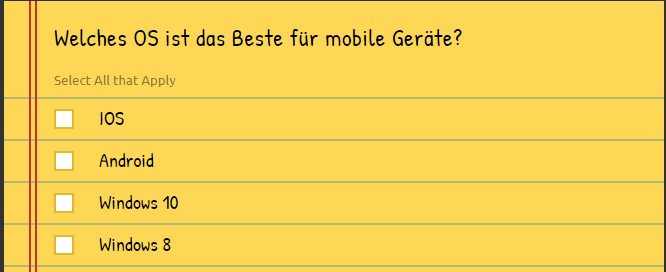
[Backend 50](#_Toc498098922)

# Aufgabenstellung

## Projektbeschreibung

Herr Max Muster möchte eine Umfrage-Software, welche er intern in seiner Firma einsetzten möchte. Hierbei geht es um simple und schnell auszufüllende Umfragen, beispielsweise «Welche dieser vier Optionen würdest Du wählen?» Auch kann es pro Umfrage nicht mehr als eine Frage geben. Eine Umfrage kann von mehreren Leuten gleichzeitig ausgefüllt werden und es kann mehrere Umfragen gleichzeitig geben.

Beispiel:



## Plattformen

|  |  |
| --- | --- |
| **Browser:** | Chrome, Firefox |
| **Devices:** | Tablets, Desktop-PC und Handys |

## Funktionale Anforderungen

## Umfrage erstellen

Als Angemeldeter Benutzer möchte ich neue Umfragen erstellen können. Eine Umfrage besteht hierbei aus zwei Elementen. Einmal aus der Frage welche maximal 255 Zeichen lang sein darf und den Antwortmöglichkeiten. Die Frage muss hierbei minimal aus einem Zeichen bestehen. Die Antwortmöglichkeiten dürfen maximal 140 Zeichen lang sein und es darf maximal nur 10 Antwortmöglichkeiten und minimal 2 Antwortmöglichkeiten geben. Frage und Antworten werden vom Ersteller definiert und können im Nachhinein nicht mehr bearbeitet werden. Man kann jedoch einstellen ob Mehrfachantworten möglich sind. Die Umfrage kann nach der Erstellung über die URL verteilt werden. Achtung: Umfragen sind nie privat, wenn man die URL weiss kann man auch daran teilnehmen!

## Umfrage ausfüllen

Umfragen sollen nach der Erstellung auch ausgefüllt werden können. Dafür muss man keinen Account besitzen, man benötigt nur die URL. Man kann nur mit den vom Ersteller vorgegebenen

Antwortmöglichkeiten antworten und man muss mindestens eine davon auswählen. Man kann eine Umfrage mehrfach ausfüllen.

## Umfrage auswerten

Als angemeldeter User möchte ich zu meinen erstellten Umfragen, eine Auswertung aller Antworten zur jeweiligen Umfrage einsehen können. Hierbei wird in einem Kuchendiagramm dargestellt welche Antwortmöglichkeit am Meisten geklickt wurde (in %). Man kann einstellen, dass Antworten vom gleichen Absender nicht doppelt angegeben werden. Es besteht die Möglichkeit alte Umfragen zu löschen.

## Registrierung und Login

Als Benutzer möchte ich mich in dem Tool registrieren können, dafür wird ein Benutzername und

Passwort benötigt. Nach der Registration kann man sich erneut einloggen mit der Angabe der selben Daten. Man kann sein Passwort nicht zurücksetzen da der Weg über einen Mail-Server den Ramen sprengen würde. Auch wird es keinen Admin-Account geben da der Benutzer alles selber einstellen kann.

**Technologie**

Für das Frontend wird Angular verwendet und der Server wird mit Golang implementiert.

## Benutzerrollen

Es gibt genau zwei Typen von Benutzern: angemeldet und nicht angemeldet. Angemeldete Benutzer können Umfragen Auswerten und erstellen. Nicht angemeldete Benutzer können nur Umfragen ausfüllen.

# Projektorganisation

**Auftraggeber:** Max Muster  
**Auftragnehmer:** Seraphin Rihm  
**Experte:** Remo Steinmann

## Planungs-Methode

Als Planungsmethode verwende ich IPERKA, ein Wasserfallmodel welches in vielen Bereichen für die Planung eingesetzt wird.

### Warum IPERKA

Ich habe IPERKA gewählt, da es sich gut für kleine Projekte eignet, welche sich nicht Über einen grossen Zeitraum spannen. Generell würde ich SCRUM für Informatikprojekte verwenden, jedoch ist dieses Projekt viel zu klein dafür und mit zwei Teilnehmer ist es nicht möglich eine korrekte SCRUM-Umgebung aufsetzen.

## Vorkenntnisse

* Angular Kenntnisse durch BMS-Projekt
* Golang Kenntnisse durch Projekt in der Abteilung
* Java Kenntnisse durch Schule und Betrieb
* C# Kenntnisse durch Schule
* C++ rudimentäre Kenntnisse durch Kleinprojekte
* Web Architektur (REST und Client-Server Architektur) Kenntnisse durch Projekt im Betrieb und der Schule
* Erfahrung mit relationalen Datenbanken (MySQL, MSSQL und Access) durch Projekte im Betrieb

## Vorbereitungen

* Aufsetzung der Entwicklungsumgebung
  + Golang installiert
  + Angular installiert
  + Visual-Studio Code installiert (IDE)
    - Plugins für Golang und Angular installiert
* Dokumentationsvorlage erstellt
* Aufgabenstellung

## Firmenstandards

Da das Projekt keine Verbindung zu meinem Arbeitsort hat verzichte ich auf Firmenstandards, einzig das Template für die Dokumentation ist von einem alten Projekt übernommen (Kopf und Fusszeile)

# Zeitplanung



## Meilensteine

Tabelle 1:Meilensteine

|  |  |  |
| --- | --- | --- |
| Meilenstein | Zeitpunkt | Status |
| Projektstart | 01.11.2017 08:15 | OK |
| Planung | 02.11.2017 15:00 | OK |
| Realisierung | 08.11.2017 17:00 | OK |
| Testing | 09.11.2017 15:00 | OK |
| Projektende | 09.11.2017 23:59 | OK |

# Kurzbeschrieb

## Projektauftrag

Es Soll ein Umfrage System erstellt werden, in wlechem Umfragen erstellt, ausgefüllt und ausgewertet werden sollen. Ein normaler User kann jedoch nur Umfragen ausfüllen. Um Umfragen erstellen und auswerten zu können muss man sich registrieren/eingeloggt sein. Das Frontend muss im Firefox und Chrom funktinieren und auf Tablets, Smarphones und Desktops ansehlich dargestellt werden können.

## Umsetzung

Der Server wird als REST-Schnitstelle in Golang realisiert. Um eine einfache erweiterung und Wartung zu garantieren wird das MVC-Pattern angewant und der Code ist objektorientiert umgesetzt. Für das Frontend wird das Angular-FrameWork verwendet welches ebenfalls eine objektorientierte Programiereung mit TypeScript ermöglicht Dabei wird Bootsrap verwendet um die seite responsive zu gestalten.

## Ergebnis

Aus Zeitlichen Gründen konnte nicht alles Umgesetzt werden, die Hauptfeatures sind jedoch alle vorhanden und Funktionieren. Nur ein paar Nebenfeatures konnten nicht umgesetzt werden.

# Arbeitsprotokoll

## Mittwoch 01.11.17

Tabelle 2:Geplane Arbeiten 01.11.17

|  |  |
| --- | --- |
| Geplante Arbeiten | Zeit |
| Zeitplan erstellen | 8:30-9:45 |
| Dokumentation Teil 1 fertigstellen | 9:45-10:00 |
| Soll/Ist Analyse durchführen | 10:00-10:30 |
| Use-Cases erstellen mit Beschrieb | 10:30 – 14:30 |
| Mit Testkonzept beginnen | 14:30 – 15:30 |
| Arbeitsprotokoll | 8:15-15:30 |

Tabelle 3:Durchgeführte Arbeiten 01.11.17

|  |  |
| --- | --- |
| Durchgeführte Arbeiten | Zeit |
| Zeitplan erstellen | 8:30-9:45 |
| Dokumentation Teil 1 fertigstellen | 9:45-10:00 |
| Soll/Ist Analyse durchführen | 10:00-10:15 |
| Use-Cases erstellen mit Beschrieb | 13:30 – 15:30 |
| Mit Testkonzept beginnen | 15:30 – 16:30 |
| Arbeitsprotokoll | 8:15-16:30 |

### Probleme

Ich hatte ein paar Probleme mit meinem Laptop nachdem ich aus dem Mittag gekommen bin, diese haben sich aber geklärt und ich konnte den Rest des Nachmittages weiterarbeiten. Ich hatte zudem einige Probleme ein geeignetes Konzept für mein Testkonzept zu finden. Anfangs bauten meine Tests aufeinander auf, sodass sie nicht unabhängig voneinander waren.

### Fazit

Diesen Tag konnte ich fast perfekt nach meiner Planung arbeiten einzig der unerwartete Input hat mich ein wenig aus dem Konzept gebracht aber ich konnte mich schnell wieder daraus fangen, da ich mit einem 6h Tag geplant habe. Bei den Use-Cases konnte ich auf ein Template aus der Schule zurückgreifen und konnte so einiges an Zeit einsparen.

### Ziele

Morgen möchte ich meinen Jetzigen lauf weiterfahren und mich möglichst wenig vom Zeitplan abheben. Grundsätzlich möchte ich bis morgen Mittag mit der Planung fertig sein und dann am Nachmittag mit der Implementierung beginnen.

## Donnerstag 02.11.17

Tabelle 4:Geplante Arbeiten 02.11.17

|  |  |
| --- | --- |
| Geplante Arbeiten | Zeit |
| Testkonzept | 8:30-10:00 |
| Software-Architektur | 10:00-11:00 |
| Datenbank-Beschrieb inklusive ERM | 11:00-13:15 |
| Server Projekt aufsetzen | 13:15 – 13:45 |
| Implementation Server beginnen | 13:45 – 14:30 |
| Arbeitsprotokoll | 8:15-14:30 |

Tabelle 5:Durchgeführte Arbeiten

|  |  |
| --- | --- |
| Durchgeführte Arbeiten | Zeit |
| Testkonzept | 8:30-10:30 |
| Software-Architektur | 10:30-14:30 |
| Datenbank-Beschrieb inklusive ERM | 15:15-16:30 |
|  |  |
| Zeitplan Formatierung überarbeiten | 14:30 – 15:00 |
| Arbeitsprotokoll | 8:15-16:30 |

### Probleme

Heute ist meine Planung nicht ganz aufgegangen, eigentlich wollte ich bereits am Nachmittag mit der Implementierung beginnen, jedoch hatte ich für die Software-Architektur und für das Erstellen des Testkonzeptes länger gebraucht als gedacht. Vor allem beim Testkonzept war ich etwas überfordert mit der schieren Masse an Testfällen. Beim Architekturteil war es wohl eher das Problem, dass ich mir noch viele Gedanken dazu machen musste. Auch habe ich seit einiger Zeit kein einziges Komponenten-Diagramm geschweige denn ein Aktivitäts-Diagramm erstellt. Bei der Datenbank hat ich nur ein klein wenig Probleme mit dem erstellen des Schemas, da ich schon lange nichts mehr mit Mysql-Workbench getan habe

### Fazit

Heute bin ich relativ weit hinter den Zeitplan gerutscht. Was sich auf die ganze weiter Planung auswirken werden wird. Weitere Verschiebungen können nicht mehr so einfach verkraftet werden, da sonst die Implantation zu kurz kommt und mein Programm somit niemals fertig werden wird. Aber ich bin ganz zufrieden mit der Geschwindigkeit mit der ich voran komme das Problem waren mehr das Anfallen von mehr Arbeit als geplant

### Ziele

Morgen kann ich zum Glück mit der Implementation beginnen, dort kann ich wie bereits erwähnt aber keine weiteren Verzögerungen hinnehmen. Da ich aber bereits alle Entwicklungsumgebungen aufgesetzt habe werde gleich mit arbeiten beginnen können.

## Freitag 03.11.17

Tabelle 6:Geplante Arbeiten 03.11.17

|  |  |
| --- | --- |
| Geplante Arbeiten | Zeit |
| Datenbank Anbindung | 8:45-9:30 |
| Autorisierungssystem | 9:30-10:45 |
| Umfrage erstellen Server | 10:45-13:00 |
| Umfrage anzeigen Server | 13:00-13:45 |
| Umfrage beantworten Server/auswerten | 13:45 – 14:30 |
| Forntend -Login | 14:30 – 15:00 |

Tabelle 7: Durchgeführte Arbeiten 03.11.17

|  |  |
| --- | --- |
| Durchgeführte Arbeiten | Zeit |
| Datenbank Anbindung | 8:45-9:30 |
| Autorisierungssystem | 9:30-10:45 |
| Umfrage erstellen Server | 10:45-13:00 |
| Umfrage anzeigen Server | 13:00-13:45 |
| Umfrage beantworten Server/auswerten | 13:45 – 15:00 |

### Probleme

Ich benötigte einiges länger um meine Serverarchitektur umzusetzen als gedacht dies lag vor allem daran, dass ich Probleme mit dem Mysql-Drivern hatte. Als die Datenbankanbindung jedoch endlich stand konnte ich mit der Implementation beginnen. Dabei taten sich wiederum weitere Probleme auf, ich hatte einige Mängel in meiner Datenbank die ich zuerst beheben musste. Ich hatte die AUTO\_INCREMENTD vergessen und einen generellen Fehler im ERM weshalb ich dieses dann auch nochmals überarbeiten musste. Dann musste ich auch noch herausfinden wie ich sinnvoll SQL-Transactions durchführe, was sich dann aber als ziemlich einfach herausstellte. Nachdem meine Backendfunktionen standen wollte ich den Server auch über den Client aufrufbar machen. Probleme hierbei waren, dass ich nicht bedacht habe, dass Angular vor jedem Request noch einen Pre-Request zum Server schickt. Diesen Pre-Request hatte ich im Backend natürlich nicht bedacht, ich konnte das Problem aber lösen indem ich diese Einfach ignorieren lies in meinem Handler.

### Fazit

Ich bin zwar etwas weiter im Zeitplan zurückgefallen, aber ich konnte immerhin den ganzen Server implementieren. Ich hätte die Fehler jedoch vermeiden können indem ich die Mysql-Driver und die Einbettung des Servers in das Frontend bereits im Vorhinein einmal getestet hätte. Auch hat mein Laptop wieder einmal Probleme mit dem Ruhemodus und ich verlor dadurch wertvolle Zeit. Also alles in allem muss ich am Mittwoch Vollgas geben .

### Ziele

Frontend bis Mittwochabend fertig zu stellen.

## Mittwoch 08.11.17

Tabelle 8: Geplante Arbeiten 08.11.17

|  |  |
| --- | --- |
| Geplante Arbeiten | Zeit |
| Login/Register | 8:45-9:30 |
| Surveys anzeigen | 9:30-10:45 |
| Surveys ausfüllen | 10:45-13:00 |
| Surveys erstellen | 13:00-13:45 |
| Surveys auswerten | 13:45 – 14:30 |
| Berechtigungen/ Doku | 14:30 – 15:00 |

Tabelle 9: Durchgeführte Arbeiten

|  |  |
| --- | --- |
| Durchgeführte Arbeiten | Zeit |
| Login/Register | 8:45-9:30 |
| Surveys anzeigen | 9:30-10:45 |
| Surveys ausfüllen | 10:45-13:00 |
| Surveys erstellen | 13:00-13:45 |
| Experten-Gespräch | 13:45 – 14:00 |
| Surveys auswerten | 14:00 – 14:30 |
| Berechtigungen/ Doku | 14:30 – 15:00 |

### Probleme

Ich hatte einige Probleme mit der Input-Validierung, da ich nicht genau wusste wie ich diese genau Umsetzen sollte, deshalb habe ich diese bis aufs weiteres ignoriert. Bootstrap hat auch noch ein paar Probleme verursacht vor allem mit den Formularen und der Navigation. Zudem sind mir noch Serverprobleme aufgetreten welche ich dann noch parallel lösen musste. Das grösste Problem zurzeit ist jedoch, dass ich mit meiner Planung komplett im Verzug bin. Ich muss den ganten Realisierungsprozess noch dokumentieren, dann die Testfälle durchführen und am Schluss noch das Projekt reflektieren.

### Experten-Gespräch

Heute hatte ich das zweite Expertengespräch mit Remo. Ich musste einfach den aktuellen Stand des Produktes und der Dokumentation zeigen. Kritikpunkte an meiner Doku waren, dass ich das Expertengespräch nicht in meinem Arbeitsprotokoll aufgeführt habe. Eigentlich sollte man das gesamte Gespräch in seinem Journal reflektieren, was ich beim Letzen mal nicht getan habe.

### Fazit

Ich konnte das Front-End funktional fertigstellen. Jedoch fehlt die Inputvalidierung noch komplett, dass heisst alle meine Negativ-Testfälle werden mit dem aktuellen Frontend fehlschlagen. Der Server reagiert zwar korrekt auf falsche eingaben es wird dem User einfach nicht angezeigt. Wie beim Problempart bereits angesprochen ist meine Dokumentation zurzeit in einem kritischen Zustand, was sich als ziemlich problematisch herausstellt, da sie doch einen grossen Teil der Endnote darstellt.

## Freitag 10.11.17

Tabelle 10: Geplante Arbeiten 10.11.17

|  |  |
| --- | --- |
| Geplante Arbeiten | Zeit |
| Dokumentation Server | 8:15-10:00 |
| Dokumentation Frontend | 10:00-13:00 |
| Testing | 13:00-15:00 |
| Auswerten | 15:00-16:00 |
| Anhang | 16:00-17:15 |

Tabelle 11: Durchgeführte Arbeiten 10.11.17

|  |  |
| --- | --- |
| Durchgeführte antworten | Zeit |
| Dokumentation Server | 8:15-10:00 |
| Dokumentation Frontend | 10:00-13:00 |
| Testing | 13:00-15:00 |
| Auswerten | 15:00-16:00 |
| Anhang | 16:00-17:15 |

### Probleme

Ich war heute komplett im Verzug mit meiner Dokumentation, da ich die letzten zwei Tage nur implementiert hatte. Zudem musste ich noch neue Anforderungen in die Doku übernehmen, welche neu mit den PK19 Änderungen aufgetaucht sind. Zudem hatte ich den Anhang unterschätzt, ich musste noch den ganzen Code vom Server und vom Frontend ins Dokument kopieren und diesen dann noch so anpassen, dass der Syntax korrekt dargestellt wird. Da Visualstudio-Code aber den Syntax korrekt nach Word exportiert konnte ich einfach alles von dort übernehmen.

### Fazit

Generell bin ich zufrieden mit meiner Leistung heute. Ich konnte die Doku komplett fertigstellen, auch wenn ich einige Probleme mit dem Anhang hatte. Mit dem Projekt ansich bin ich auch ganz zufrieden, obwohl ein paar Features fehlen. Das einzige was mich richtig stört ist, dass die Inputvalidation eingentlich komplett fehlt. Aber ich denke alles in allem konnte ich viele Erfahrungen samelln welche mir auch bei der richtigen IPA helfen.

# Informieren

## Ist-Analyse

Das Projekt wird von Grund auf neu erstellt, das heisst keinerlei Struktur oder Infrastruktur (ausser Entwicklungsumgebung) ist bereits vorhanden. Sowohl Server als auch Client werden komplett selber erstellt.

**Zur Entwicklung verwendet werden:**

* Laptop mit Windows 10
* Golang 1.9.2
* Angular 4 (benötigt Node JS)
* Visual-Studio Code
  + Golang Pugin
  + Angular Plugin
* Node JS 8.9
* Git für Versionskontrolle
* Word für die Dokumentation
* Plant-UML für die Erstellung von Diagrammen

## Soll-Analyse

Der Sollzustand wird durch die Aufgabenstellung definiert

# Planen

## Benutzerkonzept

|  |  |
| --- | --- |
| **Benutzer** | **Rechte** |
| User | Kann Umfragen ausfüllen, sich einloggen, sich registrieren |
| Eingeloggter-User | Kann Umfragen erstellen, auswerten und deaktivieren |

## Use-Cases

### Use-Case Diagramm

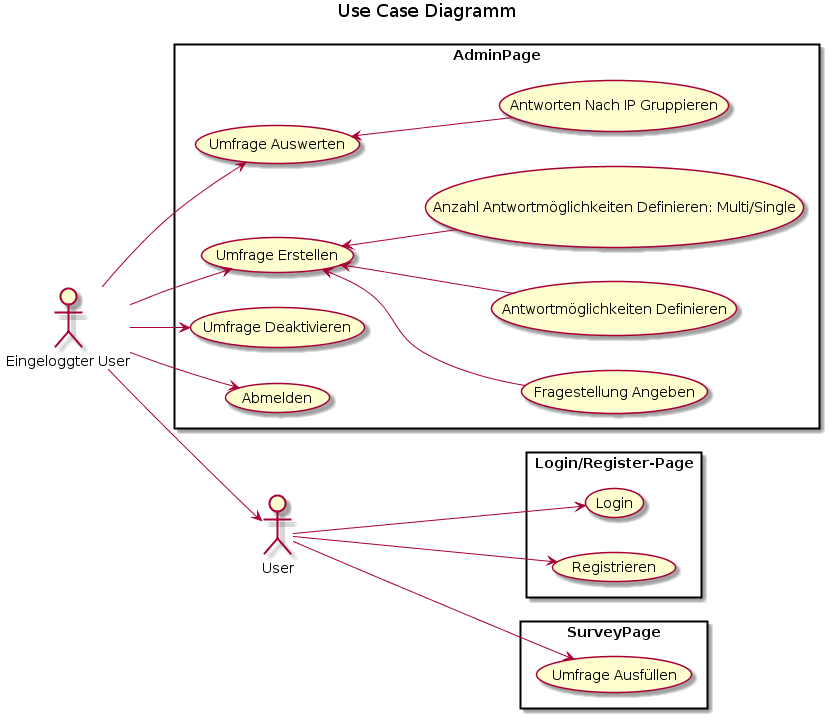


Abbildung 1:UseCases

### Use-Case Registrieren

**Beschreibung**Als User möchte ich mich auf der Seite registrieren können um auf den grösseren Funktionsumfang zugreifen zu können. **Vorbedingungen**System läuft  
**Aktion**

* Index Seite aufrufen
* Auf «Account erstellen» drücken
* Username und Passwort eingeben
* Auf Registrieren drücken

**Nachbedingung  
Erfolg:**

Weiterleitung auf Adminseite **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler) **Akteure**

User **Auslösendes Ereignis**

User möchte sich registrieren

### Use-Case Login

**Beschreibung**Als User möchte ich mich auf der Seite einloggen können um auf den grösseren Funktionsumfang zugreifen zu können und bereits erstellte Umfragen auswerten zu können. **Vorbedingungen**System läuft  
**Aktion**

* Index Seite aufrufen
* Benutzername und Passwort eingeben
* Auf Login drücken

**Nachbedingung  
Erfolg:**

Weiterleitung auf Adminseite **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler)

**Akteure**

User **Auslösendes Ereignis**:  
User möchte sich mit einem bestehenden Account einloggen

### Use-Case Abmelden

**Beschreibung**Als Eingeloggter-User möchte ich mich wieder ausloggen können **Vorbedingungen**System läuft, User ist eingeloggt  
**Aktion**

* Auf «Logout» Knopf drücken

**Nachbedingung  
Erfolg:**

Weiterleitung auf Startseite **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler)

**Akteure**

Eingeloggter-User **Auslösendes Ereignis**:  
User möchte sich aus der Seite ausloggen

### Use-Case Umfrage ausfüllen

**Beschreibung**Als User möchte ich eine bereits erstellte Umfrage ausfüllen können **Vorbedingungen**System läuft, User hat eine URL zu einer Validen existierenden Umfrage  
**Aktion**

* Umfrage-URL aufrufen
* Umfrage ausfüllen
* Auf Absende-Knopf drücken

**Nachbedingung  
Erfolg:**

Meldung zur erfolgreichen Ausfüllung wird angezeigt **Fehlschlag:**

Fehlermeldung anzeigen **Akteure**

User

**Auslösendes Ereignis**

User möchte eine Umfrage ausfüllen

### Use-Case Umfrage erstellen

**Beschreibung**Als Eingeloggter-User möchte ich neue Umfragen erstellen können **Vorbedingungen**System läuft, User ist eingeloggt  
**Aktion**

* Auf «Neu Umfrage erstellen Knopf» drücken
* Fragestellung eingeben
* Alle Antwortmöglichkeiten eingeben
* Multi oder Single Select auswählen (Bei Multi kann man beim Ausfüllen mehrere Antwortmöglichkeiten auswählen)
* Auf «Umfrage Veröffentlichen» klicken

**Nachbedingung  
Erfolg:**

Weiterleitung auf die neue Umfrage, die URL dieser neuen Seite kann zum verteilen dieser Umfrage verwendet werden. **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler)

**Akteure**

Eingeloggter-User **Auslösendes Ereignis**

Eingeloggter-User möchte eine neue Umfrage erstellen

### Use-Case Umfrage Auswerten

**Beschreibung**Als Eingeloggter-User möchte ich meine erstellten Umfragen auswerten können. **Vorbedingungen**System läuft, User ist eingeloggt, User hat bereits mindestens eine Umfrage erstellt

**Aktion**

* Bereits erstellte Umfrage auswählen
* Auf «Auswerten» drücken
* Auswählen ob gleiche IPs sortiert werden sollen

**Nachbedingung  
Erfolg:**

Es wird eine Auswertungsseite angezeigt **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler) **Akteure**

Eingeloggter-User **Auslösendes Ereignis**

Eingeloggter-User möchte das Resultat seiner Umfrage sehen

### Use-Case Registrieren

**Beschreibung**Als Eingeloggter-User möchte ich eine meiner Umfragen deaktivieren können **Vorbedingungen**System läuft, Eingeloggt, mindestens eine Umfrage existiert

**Aktion**

* Eine Umfrage auswählen
* Auf «Deaktivieren» drücken

**Nachbedingung  
Erfolg:**Umfrage ist nicht mehr über Umfrage-URL erreichbar **Fehlschlag:**

Fehlermeldung wird angezeigt (variiert je nach Fehler) **Akteure**

Eingeloggter-User **Auslösendes Ereignis**

Eingeloggter-User möchte eine Umfrage nicht mehr zur Verfügung stellen.

## Testkonzept

### Rahmenbedingungen

Als generelle Vorbedingung für alle Testfälle gilt, dass das System(Server) läuft.  
Verwendeter Browser:

* Google Chrome Version 62.0.3202.75
* Firefox 56.0.0

Tabelle 12 Testfälle 1-5:

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte sich registrieren** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 1 | User ist nicht eingeloggt | 1. Startseite aufrufen 2. Auf «Account erstellen» drücken 3. «TestUser123» bei Username-Feld eingeben 4. «1324Test» bei Passwort eingeben 5. Auf «Registrieren» drücken | Man wird auf die Adminseite weitergeleitet |
| 3 | User ist nicht eingeloggt | 1. Startseite aufrufen 2. Auf «Account erstellen» drücken 3. «» bei Username-Feld eingeben 4. «1324Test» bei Passwort eingeben 5. Auf «Registrieren» drücken | Es wird direkt neben dem Username-Feld angezeigt, dass dieses Feld nicht leer sein darf und die Registrierung wird nicht durchgeführt |
| 4 | User ist nicht eingeloggt | 1. Startseite aufrufen 2. Auf «Account erstellen» drücken 3. «TestUser123» bei Username-Feld eingeben 4. «» beim Passwort-Feld eingeben 5. Auf «Registrieren» drücken | Es wird direkt neben dem Passwort-Feld angezeigt, dass dieses Feld nicht leer sein darf und die Registrierung wird nicht durchgeführt |
| 5 | User ist nicht eingeloggt und User «TestUser123» existiert. | 1. Startseite aufrufen 2. Auf «Account erstellen» drücken 3. «TestUser123» bei Username-Feld eingeben 4. «1324Test» beim Passwort-Feld eingeben 5. Auf «Registrieren» drücken | Es wird eine Fehlermeldung angezeigt, dass der User bereits existiert |

Tabelle 13:Testfälle: 6-11

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte sich einloggen** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 6 | User ist nicht eingeloggt und «TestUser123» existiert | 1. Startseite aufrufen 2. «TestUser123» bei Username-Feld eingeben 3. «1324Test» bei Passwort eingeben 4. Auf «Login» drücken | Man wird auf die Adminseite weitergeleitet |
| 7 | User ist nicht eingeloggt | 1. Startseite aufrufen 2. «» bei Username-Feld eingeben 3. «1324Test» bei Passwort eingeben 4. Auf «Login» drücken | Es wird direkt neben dem Username-Feld angezeigt, dass dieses Feld nicht leer sein darf und man wird nicht auf die Adminseite weitergeleitet |
| 8 | User ist nicht eingeloggt | 1. Startseite aufrufen 2. «TestUser123» bei Username-Feld eingeben 3. «» beim Passwort-Feld eingeben 4. Auf «Login» drücken | Es wird direkt neben dem Passwort-Feld angezeigt, dass dieses Feld nicht leer sein darf und man wird nicht auf die Adminseite weitergeleitet |
| 9 | User ist nicht eingeloggt und User «NoUser» existiert nicht. | 1. Startseite aufrufen 2. «NoUser» bei Username-Feld eingeben 3. «1324NoUser» beim Passwort-Feld eingeben 4. Auf «Login» drücken | Es wird eine Fehlermeldung angezeigt, dass kein User mit diesen Angaben gefunden werden konnte und man wird nicht auf die Adminseite weitergeleitet |
| 10 | User ist eingeloggt | 1. Startseite aufrufen | Man wird direkt wieder auf die Adminseite weitergeleitet |
| 11 | User ist nicht eingeloggt | 1. Adminseite aufrufen | Man wird wieder auf die Startseite weitergeleitet da man nicht angemeldet ist |

Tabelle 14: Testfalle 11

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte sich ausloggen** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 11 | User ist eingeloggt | 1. Adminseite aufrufen 2. Auf «Logout» drücken | Man wird wieder auf die Startseite weitergeleitet. |

Tabelle 15: Testfälle 12-13

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte eine Umfrage ausfüllen** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 12 | Umfrage existiert | 1. Umfrage-URL aufrufen 2. Folgende Antwortmöglichkeiten auswählen (Check-Boxen): «Salat», «Spinat» und «Orangen» 3. Auf «Absenden» klicken | Man wird auf die Abgesendetseite weitergeleitet |
| 13 | Umfrage existiert | 1. Umfrage-URL aufrufen 2. Keine Antwortmöglichkeit auswählen 3. Auf «Absenden» klicken | Es wird eine Fehlermeldung angezeigt, dass man mindestens eine Antwortmöglichkeit auswählen muss |
| 14 | Umfrage existiert nicht | 1. NotExistingUmfrage-URL aufrufen | Es wird auf eine «Umfrage nicht gefunden» Seite weitergeleitet |
| 15 | Umfrage existiert | 1. SingleAnswerUmfrage-URL aufrufen 2. Folgende Antwortmöglichkeiten auswählen (Radio-Buttons): «Windows 10» auswählen 3. Auf «Absenden» klicken | Man wird auf die Abgesendetseite weitergeleitet |

Tabelle 16:Tesfälle 16-22

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte eine Umfrage erstellen** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 16 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «Was ist das beste Mobile-OS?» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): «IOS», «Android», «Windows 10», «Windows 8» 4. Auf «Umfrage» erstellen klicken | Man wird auf die neu erstellte Umfrage weitergeleitet. Die Antwortmöglichkeiten sind mit Radio-Buttons angegeben und Fragestellung und Antwortmöglichkeiten stimmen mit der vorhergehenden Antwortmöglichkeit überein |
| 17 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «Welches dieser Lebensmittel gilt als gesund?» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): «Salat», «Orange», «Spinat», «Schokolade», «Windows 8» 4. Mehrere «Antwortmöglichkeiten erlauben» selektieren 5. Auf «Umfrage» erstellen klicken | Man wird auf die neu erstellte Umfrage weitergeleitet. Die Antwortmöglichkeiten sind mit Check-Boxen angegeben und Fragestellung und Antwortmöglichkeiten stimmen mit der vorhergehenden Antwortmöglichkeit überein |
| 18 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): «IOS», «Android», «Windows 10», «Windows 8» 4. Auf «Umfrage» erstellen klicken | Es wird eine Fehlermeldung angezeigt, dass das Fragestellungsfeld nicht leer sein darf |
| 19 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld einen Satz mit mehr als 255 Zeichen eingeben | Es ist nicht möglich zu mehr als 255 Zeichen einzugeben |
| 20 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «Was ist das beste Mobile-OS?» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): «IOS» 4. Auf «Umfrage» erstellen klicken | Es wird eine Fehlermeldung angezeigt, dass es mindestens zwei Antwortmöglichkeiten sein müssen. |
| 21 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «Was ist das beste Mobile-OS?» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): eine Antwortmöglichkeit mit mehr als 140 Zeichen | Es ist nicht möglich eine Antwortmöglichkeit mit mehr als 140 Zeichen einzugeben |
| 22 | User ist eingeloggt | 1. Auf «Neue Umfrage» klicken 2. Beim Fragestellungsfeld «Was ist das beste Mobile-OS?» eingeben 3. Folgende Antwortmöglichkeiten erfassen (unterhalb des Fragestellungsfeldes): 11-mal «Test» | Es ist nicht möglich mehr als 10 Antwortmöglichkeiten zu erfassen |

Tabelle 17:Testfälle 23-25

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte eine auswerten** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 23 | User ist eingeloggt und hat bereits eine Umfrage erstellt | 1. Umfrage in Liste selektieren 2. Auf «Auswerten» klicken | Man wird auf die Auswertungsseite weitergeleitet. Auf welcher man ein Kuchendiagramm sieht. Das Kuchendiagramm stellt prozentual dar welche Antwortmöglichkeit wie oft ausgewählt wurde. |
| 24 | User ist eingeloggt und hat bereits eine Umfrage erstellt | 1. Umfrage in Liste selektieren 2. Auf «Auswerten» klicken 3. Auf «Antworten von gleicher IP gruppieren» klicken | Man wird auf die Auswertungsseite weitergeleitet. Auf welcher man ein Kuchendiagramm sieht. Das Kuchendiagramm stellt prozentual dar welche Antwortmöglichkeit wie oft ausgewählt wurde. Antworten von der gleichen IP werden nur noch einmalig beachtet |
| 25 | User ist eingeloggt und hat noch keine Umfrage erstellt |  | Es werden keine Umfragen aufgelistet |

Tabelle 18:Testfall 26

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte eine Umfrage deaktivieren** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 26 | User ist eingeloggt und hat bereits eine Umfrage erstellt | 1. Umfrage in Liste selektieren 2. Auf «Deaktivieren» klicken | Umfrage ist nicht mehr erreichbar und kann von niemanden mehr ausgefüllt werden |

## Software-Architektur

In diesem Kapitel wird das System genauer beschrieben.

### Komponente

Abbildung 2:Komponenten Diagramm

### Front-End

Das Frontend wird mit Angular umgesetzt, dabei wird ein MVVM-Pattern angewandt um auch dort eine Schichtentrennung zu erreichen, da Server und Frontend komplett gekapselt sind haben diese auch eine separate Schichtentrennung. Das Frontend kommuniziert mit dem Server über das HTTP-Protokoll. Auch werden die Berechtigungen für den Seitenzugriff komplett vom Front-End gehandelt. Der Server ist nur für das Login an sich verantwortlich. Das Frontend ist dank Angular als Singel-Page-Application umgesetzt.

### Server

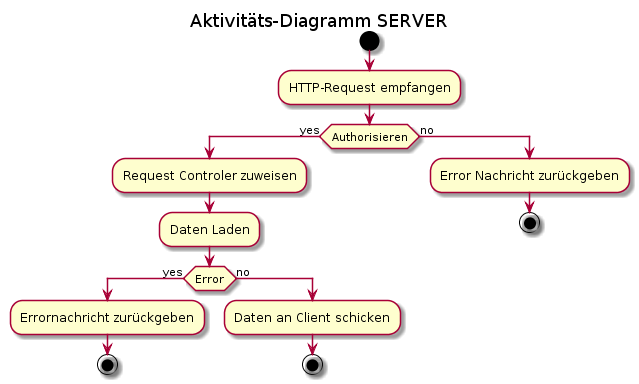
Der Server dient als REST (Representational State Transfer) API über welche das Frontend Daten laden und speichern kann. Grundsätzlich heisst REST, dass der Server komplett State-Less ist. Ein Benutzer ist also nur auf dem Frontend eingeloggt wenn er Serverabfragen tätig muss er sich erneut Autorisieren was aber direkt vom Frontend übernommen wird.

Abbildung 3:Akitvitäts-Diagramm

## Datenbank

### ERD

Abbildung 4:ERD

Tabelle 19:User

|  |  |  |
| --- | --- | --- |
| **User:** In dieser Tabelle werden Benutzer abgespeichert. | | |
| **Feld** | **Propertys** | **Beschreibung** |
| idUser | INT, Primary-Key, Not Null, Unique | - |
| username | varchar(45), Not Null | Benutzername des Benutzers |
| password | varchar(45), Not Null | Passwort des Benutzers |

**User ->(1:n) Survey**Ein Benutzer kann mehrere Surveys haben ein Survey kann aber nur einem Benutzer zugewiesen sein.

Tabelle 20:Survey

|  |  |  |
| --- | --- | --- |
| **Survey: In dieser Tabelle wird eine Umfrage abgebildet** | | |
| **Feld** | **Propertys** | **Beschreibung** |
| idSurvey | INT, Primary-Key, Not Null, Unique |  |
| question | varchar(255), Not Null | Fragestellung der Umfrage |
| User\_idUser | INT, Foreign Key, Not Null |  |

**Survey ->(1:n) Choices**Eine Umfrage kann mehrere Antwortmöglichkeiten haben aber eine Antwortmöglichkeit kann nur zu einer Umfrage gehören

Tabelle 21: Choices

|  |  |  |
| --- | --- | --- |
| **Choices: Hier werden die Antwortmöglichkeiten einer Umfrage abgebildet** | | |
| **Feld** | **Propertys** | **Beschreibung** |
| idChoices | INT, Primary-Key, Not Null, Unique |  |
| choice | varchar(140), Not Null | Antwortmöglichkeiten der Umfrage |
| Survey\_idSurvey | INT, Foreign Key, Not Null |  |

**Choices ->(1:n)R esult**Eine Antwortmöglichkeit kann in mehreren Antworten vorkommen eine Antwort hat aber nur eine Antwortmöglichkeit

Tabelle 22: Result

|  |  |
| --- | --- |
| **Result: Antworten zu einer Umfrage werden hier abgebildet** | |
| **Feld** | **Propertys** |
| Choices\_idChoices | INT, Foreign Key, Not Null |

# Entscheiden

## Versionsverwaltung

Um konstante Back-Ups zu liefern verwende ich ein Versionsverwaltungssystem. Ich habe mich für GIT gehostet von Github entschieden, da ich bereits mit GIT vertraut bin und wir GIT auch Siemensintern verwenden. Da mein Projekt jedoch nichts mit dem Geschäft zu tun hat verwende ich auch nicht den internen Hoster, sondern Github. Durch das Hosting über GitHub ist der gesamte Source-Code auch für andere Leute sichtbar, diese können dann auch Änderungsvorschläge einbringen, sie können den Code jedoch nicht selbständig verändern. Durch konstantes Commit kann ich ein einfaches Back-Tracking ermöglichen, dass heisst wenn ich mit einem Commit etwas Kaput mach, kann ich ganz einfach analysieren welche Änderung schuld ist und eventuell sogar einen Commit zurückspringen und meine Änderungen rückgängig machen. Eine Continuous-Integration hätte diesen Prozess noch weiter unterstützt, da ich aber keine automatisierten Tests umsetze, habe ich mich dagegen entschieden

Um Git einfacher zu bedienen verwende ich TortoiseGit welches git direkt in Windows integriert.  
  
GitHub Projektlink: <https://github.com/StarlordTheCoder/ProbeIPA>

## Angular Version

Ich habe mich für Angular4 entschieden, obwohl es bereits eine neuere Version geben würde (Angular 5). Aus dem einfachen Grund, dass Angular 5 nur Performance Verbesserungen bring, jedoch keine neuen Features welche ich bräuchte. Zudem brache ich ngx-charts und Bootstrap welche bereits gut in Angular 4 integriert sind und erst teilweise in Angualr5. Auch ist Angular5 erst 1 Tag nach der initailaen Planung erschienen, sodass ich mich zu diesem Zeitpunkt bereits für Angular 4 entschieden habe.

### Bootstrap

Ich werde Bootsrap verwenden um einfach ein responsives Frontend zur Verfügung zu stellen. Zudem ist Bootstrap einfach in das Angular-Framework integrierbar und weitverbreitet ungeschützt. Es punktet auch noch mit einer einfachen Implementation.

### Ngx-charts

Ich verwende ngx-charts um meine Umfragen auszuwerten. Vorteile sind, dass es einfach in Angular integrierbar ist und es sehr einfach zu implementieren ist. Man übergibt einfach ein JSON-Object und es wird ein vordefiniertes Diagramm erstellt.

## Projekt Durchführbarkeit

Das Projekt ist mit der jetzigen Planung durchführbar, Verzögerungen können jedoch eine Kettenreaktion auslösen, sodass ich einige Features nicht mehr umgesetzt werden könnten. Aber ansonsten ist alles für die Realisierung vorbereitet: Architektur und Testfälle stehen jetzt müssen diese nur noch umgesetzt werden.

# Realisieren

## Frontend

### Komponenten

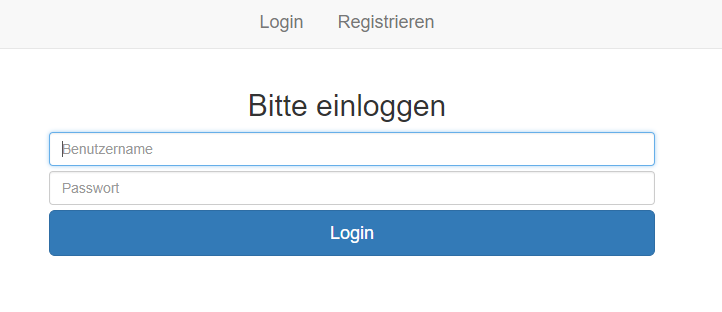
**Login**

Abbildung 5:Login

Die Login-Seite dient als Entrypoint der Applikation. Um sich einloggen zu können muss man seine Benutzerdaten eingeben. Dies ist nur möglich, wenn man sich bereits registriert hat. Von der Login-Seite hat man nur zugriff auf die Registrierungsseite. Nach dem Login wird man auf die Adminseite weitergeleitet

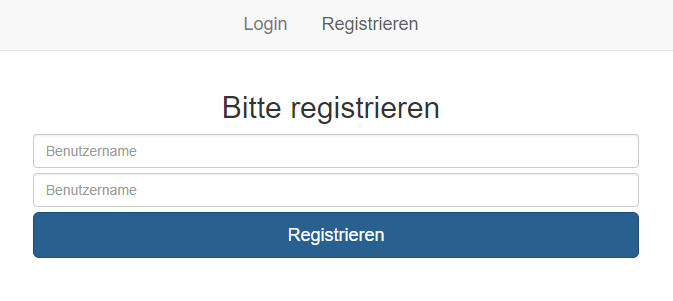
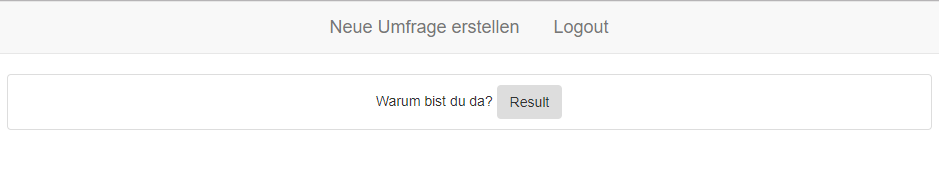
**Registrieren**

Abbildung 6:Registrieren

Auf dieser Seite kann man sich Registrieren wenn man noch keinen Account im System besitzt, hierfür wird ein Benutzername und ein Passwort benötigt. Nach erfolgreicher Registrierung wird man auf den Adminbereich weitergeleitet.

**Adminbereich**

Hier werden in einer Liste alle bereits von dir erstellten Umfragen aufgelistet. Man hat von hieraus Zugriff auf die Resultate der jeweiligen Umfrage (Resultat-Knopf). Zudem kann man sich über die Logoutschaltfläche wieder ausloggen und man wird auf die Login-Seite weitergeleitet. Über die «Neue Umfrage erstellen» Schaltfläche wird man auf die Umfrage-Erstellen-Seite weitergeleitet.

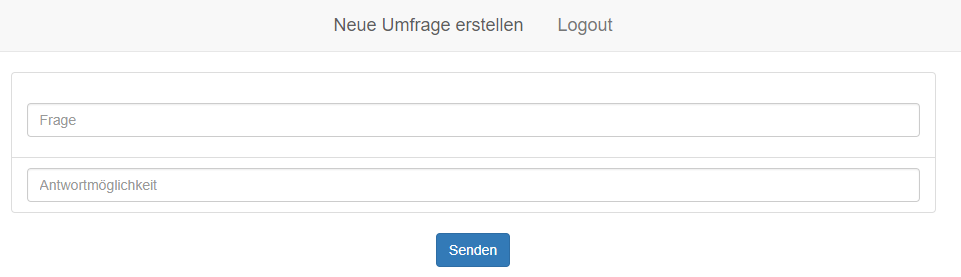
**Umfrage erstellen**

Abbildung 7: Umfrage erstellen

Hier kann man eine neue Umfrage erstellen. Dafür wird mindestens eine Antwortmöglichkeit benötigt und die Umfrage-Frage: z.B. Welche Farbe hat eine Banane? Auf diese Seite hat man nur als eingeloggter User Zugriff. Nach erfolgreicher Erstellung wird man auf die neue Umfrage weitergeleitet

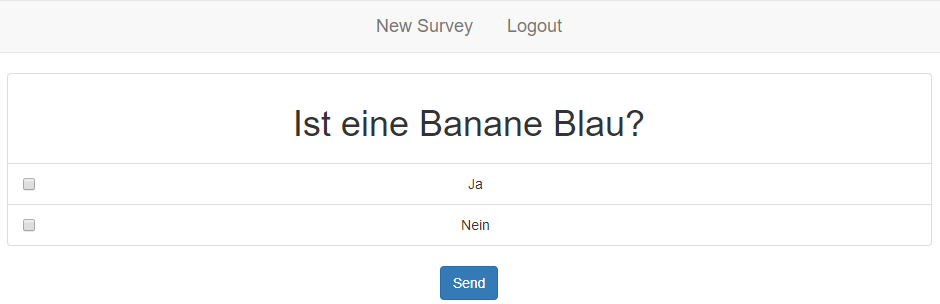
**Umfrage ausfüllen**

Abbildung 8:Umfrage ausfüllen

Wenn man die Umfrage-URL weiss kann man auf diese auch immer zugreifen. Sobald man eine oder mehrere Antwortmöglichkeiten ausgewählt hat kann man diese dann auch absenden. Bei erfolgreichem Abschluss wird man entweder auf die Loginseite weitergeleitet oder auf den Adminbereich weitergeleitet.

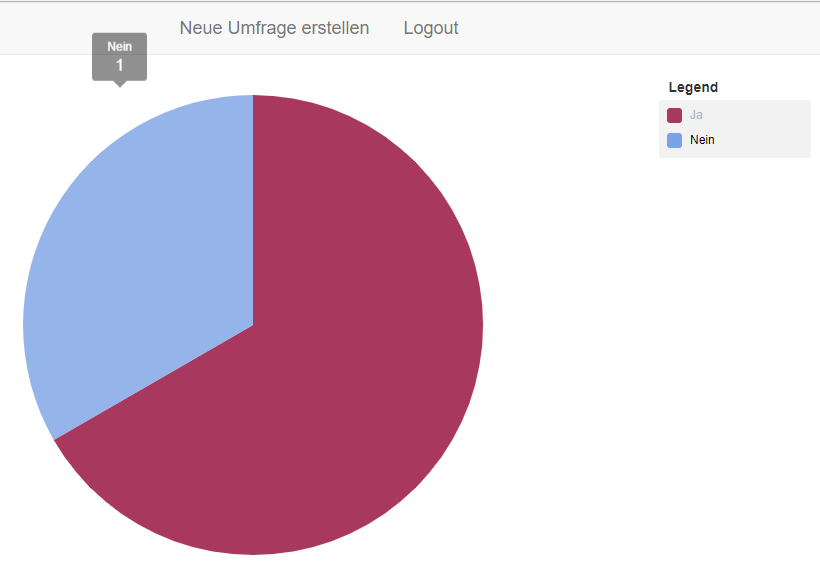
**Resultat**

Abbildung 9:Resultat

Hier wird angezeigt wie oft welche Antwortmöglichkeit angewählt wurde. Man hat nur als eingeloggter User darauf Zugriff.

## Backend

### Klassen Diagramm

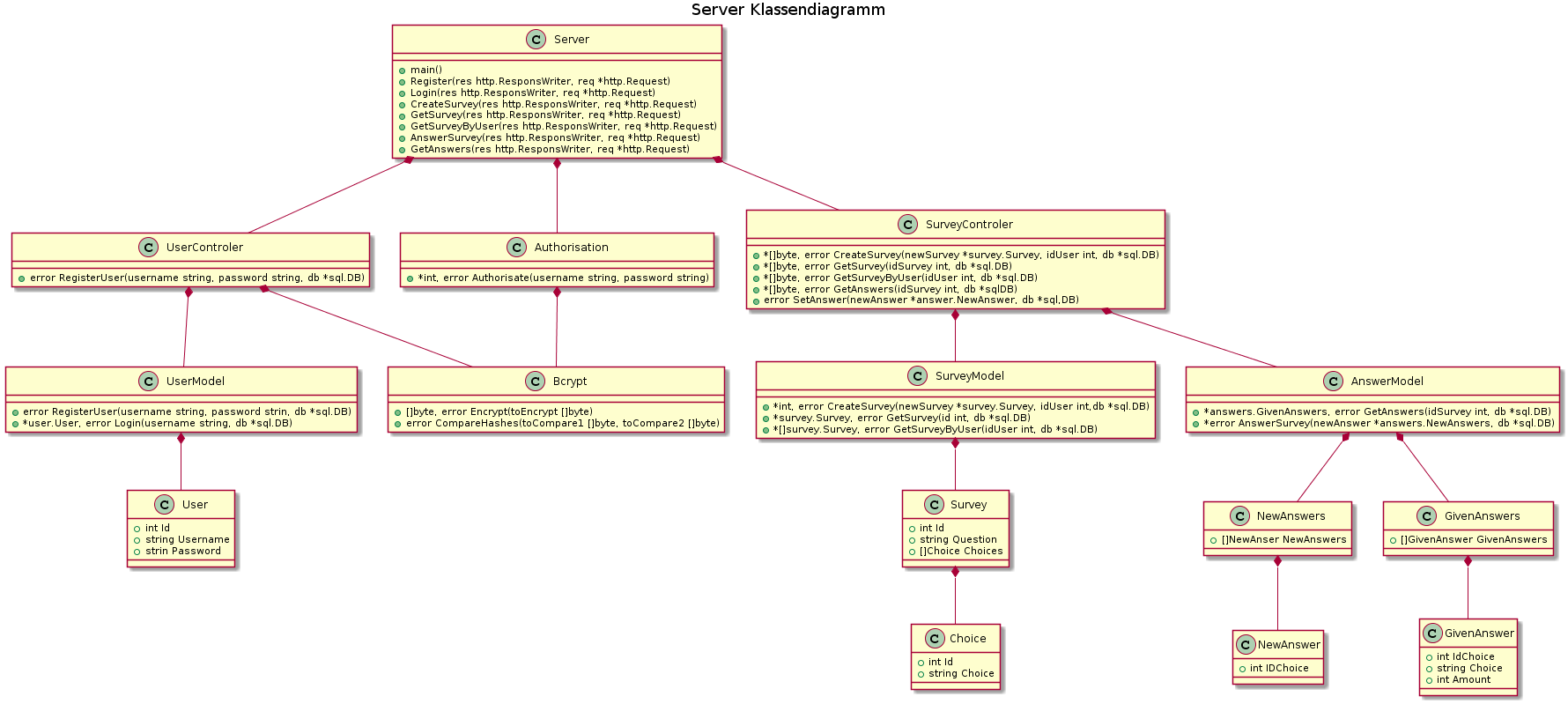


Abbildung 10:Klassendiagramm Server

### Klassenbeschrieb

**Server**Diese Klasse dient als Router welche alle Serveranfragen verarbeitet, dass jede anfrage welche an den Server geht wird hier an die Verschiedenen Controler weitergeleitet. Jede Funktion ist in einer Route abgebildet. Wenn der Server eine Anfrage einfängt wird diese zuerst der entsprechenden Route weitergeleitet und dann werden gegeben falls die Anfrage-Daten aus der Anfrage extrahiert und diese an den Controler weitergeleitet. Je nach dem ob die Anfrage ein Erfolg oder Misserfolg ist werden dann Daten zurückgegeben. Jede Abfrage ist State-Less, dass heisst es gibt auf dem Server keine einzige Statusänderung (z.B. Login), er bleibt immer im gleichen State. Falls eine Abfrage Berechtigungen braucht wird die Authorisation-Klasse gebraucht um dies zu überprüfen.

**UserControler**Dieser Controler ist nur für die Registrierung zuständig. Hierfür wird ein Username und ein Passwort benötigt Das Passwort wird mit der Bycript-Klasse gehashed und dieser Hash wird dann an das Model weitergeleitet

**SurveyControler**Dieser Controler ist für das erstellen, laden, beantworten, und das Zusammenstellen der Resultate von Umfragen verantwortlich.

**Authorisation**Die Authorisations-Klasse wird anstelle einer Login-Methode verwendet, sie überprüft ob der in der Abfrage mitgegeben User existiert und die Passwörter übereinstimmen, dafür werden die Hashes verglichen. Dies wird über die Bcrypt-Klasse getan. Die Klasse wird vom Server verwendet um zu überprüfen ob der Absender dazu berechtigt ist diese Aktion auszuführen.

**Bcrypt**Die Bcrypt-Klasse wird zum erstellen und vergleichen von Passworthashes gebraucht.

**UserModel**Das UserModel kümmert sich um die Userdaten auf der Datenbank, es ist die einzige Klasse welche direkt SQL-Befehle bezüglich des Users ausführt

**SurveyModel**Das UserModel kümmert sich um die Umfragedaten auf der Datenbank, es ist die einzige Klasse welche direkt SQL-Befehle bezüglich der Umfragen ausführt

**AnswerModel**Das AnswerModel kümmert sich um die Resultatdaten der Umfragen auf der Datenbank, es ist die einzige Klasse welche SQL-Befehle bezüglich der Resultate ausführt

**DTOS**Die restlichen Klassen dienen als DTOs um die Daten von der Datenbank im Backend abzubilden

# Kontrollieren

## Testprotokoll

Die Testfälle wurden in den definierten Browsern durchgeführt.

Tabelle 23:Testprotololl 1-5

|  |  |  |
| --- | --- | --- |
| Nr. | Status | Abweichung |
| 1 | OK | - |
| 3 | OK | - |
| 4 | OK | - |
| 5 | OK | - |

Tabelle 24:Testprotokoll 6-11

|  |  |  |
| --- | --- | --- |
| Nr. | Vorbedingung | Abweichung |
| 6 | OK | - |
| 7 | OK | - |
| 8 | OK | - |
| 9 | OK | - |
| 10 | OK | - |
| 11 | OK | - |

Tabelle 25: Testprotokoll 11

|  |  |  |
| --- | --- | --- |
| Nr. | Vorbedingung | Abweichung |
| 11 | OK | - |

Tabelle 26:Testprotokoll 16-22

|  |  |  |
| --- | --- | --- |
| Nr. | Vorbedingung | Abweichung |
| 12 | OK(~) | Man wird auf die Login-Seite weitergeleitet |
| 13 | OK(~) | Es wird einfach nicht abgesendet aber der User wird nicht informiert |
| 14 | OK(~) | Man wird auf die Loginseite weitergeleitet |
| 15 | OK | - |

Tabelle 27:Testprotokoll 16-22

|  |  |  |
| --- | --- | --- |
| Nr. | Vorbedingung | Abweichung |
| 16 | NOK | Man kann keine Umfrage erstellen mit Radio-Buttons |
| 17 | OK | - |
| 18 | OK(~) | User wird nicht informiert |
| 19 | OK |  |
| 20 | NOK | Man kann die Umfrage auch erstellen wenn nur die Frageerstellung angibt |
| 21 | OK | - |
| 22 | OK | - |

Tabelle 28:Testprotokoll 23-25

|  |  |  |
| --- | --- | --- |
| Nr. | Vorbedingung | Abweichung |
| 23 | OK | - |
| 24 | NOK | Das Feature wurde nicht implementiert |
| 25 | OK | - |

Tabelle 29:Testprotokoll 26

|  |  |  |  |
| --- | --- | --- | --- |
| **User möchte eine Umfrage deaktivieren** | | | |
| Nr. | Vorbedingung | Eingabe | Erwartete Ausgabe |
| 26 | NOK |  |  |

## Testfazit

Die Hauptfeatures des Programms funktionieren. Das heisst man kann Umfragen erstellen diese dann Beantworten und Auswerten. Auch das Login und die Registrierung funktionieren wie auch das Berechtigungskonzept. Der Server funktioniert auch Komplett unabhängig vom Frontend. Jedoch ist die inputvalidation zurzeit noch nicht genügend. Gewisse fehler werden gar nicht Abgefangen, oder der User wird gar nicht richtig über einen Fehler oder eine Fehleingabe informiert. Das System ist aber bei keinem der Testfälle komplett abgestürtzt, es kam nur zu fehlverhalten. Der rest funktioniert jedoch konstant. Der Server gibt auch Detailierte Fehlermeldungen zurück wenn etwas schiefgegangen ist. So ist es ganz einfach herauszufinden was genau schiefging

# Auswerten

## Reflexion

Das Projekt war eine gute Erfahrung für mich. Erstens habe ich nun erfahrung wie es ist eine Projekt 1 zu 1 nach Iperka durchzuführen und zu Dokumentieren. Vorallem die Dokumentation habe ich dabei unterschätzt, da es Doch ein Ziemlicher aufwand war jede Iperka-Stuffe ausführlich zu Dokumentieren. Das grösste Problem war jedoch der Projektauftrag, da ich diesen viel zu ausführlich definiert hatte, konnte ich nicht ganz alle Features davon implementieren. Auch, dass ich mit meinen Technologien(GO/Angular) noch nicht alzuviele Erfahrungen hatte, war nicht gerade Produktiv. Dafür konnte ich meine geplante Architektur ohne Problme umsetzen, sodass mein Projekt sehr einfach erweiterbar ist.

## Verbesserungen/Erfragungen

Nächstes Mal würde ich meinen Auftrag so umformulieren, dass gewisse Teile davon als Potential erfasst sind und zwingend Umgsetzt werden müssen. Auch würde ich nächstesmal lieber technologien verwenden welche ich beriets in and out kenne. Am Produkt ansich würde ich vorallem die Inputvalidierung anpassen und verbessern, da sie zurzeit sehr Userunfreundlich ist.

## Einsetzbarkeit

Das Produkt könnte im jetzigen zustand gebraucht werden, es sollte einfach noch die Benutzerfreundlichkeit verbessert werden.

## Fazit

Das Projekt hat mir viele neue Erfahrungen gebracht und ich konnte mir nun ein Bild machen wie stressig eine IPA sein wird. Mit der Umsetzung bin ich generell eigentlich zufrieden, vorallem da alles Objektorientiert umgesetzt ist und ich neuste Technologien verwendet habe.

# Anhang

## Abbildungsverzeichnis

[Abbildung 1:UseCases 12](#_Toc498092301)

[Abbildung 2:Komponenten Diagramm 21](#_Toc498092302)

[Abbildung 3:Akitvitäts-Diagramm 21](file:///C:\Users\Serafin\ProbeIPA\ProbeIPA\Doku\ProbeIPABericht.docx#_Toc498092303)

[Abbildung 4:ERD 22](#_Toc498092304)

[Abbildung 5:Login 25](#_Toc498092305)

[Abbildung 6:Registrieren 25](#_Toc498092306)

[Abbildung 7: Umfrage erstellen 27](#_Toc498092307)

[Abbildung 8:Umfrage ausfüllen 27](#_Toc498092308)

[Abbildung 9:Resultat 28](#_Toc498092309)

[Abbildung 10:Klassendiagramm Server 29](#_Toc498092310)

## Tabellenverzeichniss

[Tabelle 1:Meilensteine 5](#_Toc498098588)

[Tabelle 2:Geplane Arbeiten 01.11.17 6](#_Toc498098589)

[Tabelle 3:Durchgeführte Arbeiten 01.11.17 6](#_Toc498098590)

[Tabelle 4:Geplante Arbeiten 02.11.17 7](#_Toc498098591)

[Tabelle 5:Durchgeführte Arbeiten 7](#_Toc498098592)

[Tabelle 6:Geplante Arbeiten 03.11.17 8](#_Toc498098593)

[Tabelle 7: Durchgeführte Arbeiten 03.11.17 8](#_Toc498098594)

[Tabelle 8: Geplante Arbeiten 08.11.17 9](#_Toc498098595)

[Tabelle 9: Durchgeführte Arbeiten 9](#_Toc498098596)

[Tabelle 10: Geplante Arbeiten 10.11.17 10](#_Toc498098597)

[Tabelle 11: Durchgeführte Arbeiten 10.11.17 10](#_Toc498098598)

[Tabelle 12 Testfälle 1-5: 17](#_Toc498098599)

[Tabelle 13:Testfälle: 6-11 17](#_Toc498098600)

[Tabelle 14: Testfalle 11 18](#_Toc498098601)

[Tabelle 15: Testfälle 12-13 18](#_Toc498098602)

[Tabelle 16:Tesfälle 16-22 19](#_Toc498098603)

[Tabelle 17:Testfälle 23-25 20](#_Toc498098604)

[Tabelle 18:Testfall 26 20](#_Toc498098605)

[Tabelle 19:User 22](#_Toc498098606)

[Tabelle 20:Survey 22](#_Toc498098607)

[Tabelle 21: Choices 23](#_Toc498098608)

[Tabelle 22: Result 23](#_Toc498098609)

[Tabelle 23:Testprotololl 1-5 31](#_Toc498098610)

[Tabelle 24:Testprotokoll 6-11 31](#_Toc498098611)

[Tabelle 25: Testprotokoll 11 31](#_Toc498098612)

[Tabelle 26:Testprotokoll 16-22 31](#_Toc498098613)

[Tabelle 27:Testprotokoll 16-22 31](#_Toc498098614)

[Tabelle 28:Testprotokoll 23-25 32](#_Toc498098615)

[Tabelle 29:Testprotokoll 26 32](#_Toc498098616)

## Glossar

|  |  |
| --- | --- |
| **Begriff** | **Erläuterung** |
| Angular | Ist ein Frame-Work für die Frontendentwicklung im Web, es basiert auf TypeScript welches zu Javascript kompiliert wird. |

## Frontend

### login.component.css

.form-control {

margin: 5px;

margin-right: 5px;

}

.login-button {

margin: 5px;

margin-right: 5px;

}

### login.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row" style="margin: 5px;">

<div class="col-md-4 col-md-offset-4 col-sm-12 text-center">

<form class="form-signin">

<h2 class="form-signin-heading">Bitte einloggen</h2>

<label for="inputEmail" class="sr-only">Benutzername</label>

<input name="username" type="username" id="username" class="form-control" placeholder="Benutzername" required autofocus [(ngModel)]="user.Username" maxlength="20">

<label for="inputPassword" class="sr-only">Passwort</label>

<input name="password" type="password" id="inputPassword" class="form-control" placeholder="Passwort" required [(ngModel)]="user.Password" maxlength="20">

<button class="btn btn-lg btn-primary btn-block login-button" (click)="login()">Login</button>

</form>

</div> <!-- /container -->

</div>

### login.component.ts

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

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

import { User } from '../model/user.model';

import { Const } from '../const';

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

@Component({

selector: 'login',

templateUrl: './login.component.html',

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

})

export class LoginComponent {

public user: User = {Id: 0} as User;

constructor (private http: HttpClient, private router: Router) {

if (sessionStorage.getItem('username') !== null) {

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

}

}

public login() {

console.warn(this.user);

//Request To Server

this.http.post(Const.baseUrl + 'login', this.user).subscribe(data => {

sessionStorage.setItem('username', this.user.Username);

sessionStorage.setItem('password', this.user.Password);

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

}, err => {

console.warn('error');

});

}

}

### register.component.css

.form-control {

margin: 5px;

}

.login-button {

margin: 5px;

}

### register.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row" style="margin:5px;">

<div class="col-md-4 col-md-offset-4 col-sm-12 text-center">

<form class="form-signin">

<h2 class="form-signin-heading">Bitte registrieren</h2>

<label for="inputEmail" class="sr-only">Benutzername</label>

<input name="username" id="username" class="form-control" placeholder="Benutzername" required autofocus [(ngModel)]="user.Username" maxlength="20">

<label for="inputPassword" class="sr-only">Passwort</label>

<input name="password" type="password" id="inputPassword" class="form-control" placeholder="Benutzername" required [(ngModel)]="user.Password" maxlength="20">

<button class="btn btn-lg btn-primary btn-block login-button" (click)="register()">Registrieren</button>

</form>

</div> <!-- /container -->

</div>

### register.component.ts

import { User } from './../model/user.model';

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

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

import { Const } from '../const';

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

@Component({

selector: 'register',

templateUrl: './register.component.html',

styleUrls: ['./register.component.css']

})

export class RegisterComponent {

public user: User = {Id: 0} as User;

constructor (private http: HttpClient, private router: Router) {

if (sessionStorage.getItem('username') !== null) {

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

}

}

public register() {

console.warn(this.user);

//Register user on Server

this.http.post(Const.baseUrl + 'register', this.user).subscribe(data => {

sessionStorage.setItem('username', this.user.Username);

sessionStorage.setItem('password', this.user.Password);

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

}, err => {

console.warn('error');

});

}

}

### admin-main.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row" style="margin:5px;">

<div class="col-md-6 col-md-offset-3 col-sm-12 text-center">

<ul class="list-group">

<li \*ngFor="let survey of surveys" class="list-group-item">

{{survey.Question}}

<button type="button" class="btn" (click)="loadResult(survey.Id)">Result</button>

</li>

</ul>

</div> <!-- /container -->

</div>

### admin-main.compontent.ts

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

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

import { Const } from '../const';

import { Survey } from '../model/survey.model';

import { Header } from '../header';

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

@Component({

selector: 'main-admin',

templateUrl: './main-admin.component.html',

styleUrls: ['./main-admin.component.css']

})

export class MainAdminComponent {

public surveys: Survey[] = [];

private headers: Header;

constructor (private http: HttpClient, private router: Router) {

this.headers = new Header();

//load Surveys from Server

this.http.get<Survey[]>(Const.baseUrl + 'getSurveyByUser', { headers: this.headers.headers }).subscribe(data => {

this.surveys = data;

}, err => {

console.warn('error', err);

});

}

loadResult(id: number) {

this.router.navigate(['/result', id]);

}

}

### survey.component.css

h1{

word-wrap: break-word;

-webkit-hyphens: auto;

-moz-hyphens: auto;

-ms-hyphens: auto;

-o-hyphens: auto;

hyphens: auto;

}

### survey.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row" style="margin:5px;">

<div class="col-md-6 col-md-offset-3 col-sm-12 text-center">

<ul class="list-group">

<li class="list-group-item">

<h1>{{surveys.Question}}</h1>

</li>

<li \*ngFor="let choice of surveys.Choices" class="list-group-item">

<input type="checkbox" style="float: left" (click)="checkBox(choice.Id)">

{{choice.Choice}}

</li>

</ul>

<button type="button" class="btn btn-primary" (click)="answerSurvey()">Send</button>

</div> <!-- /container -->

</div>

### survey.component.ts

import { Component, OnInit, OnDestroy } from '@angular/core';

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

import { Const } from '../const';

import { Survey } from '../model/survey.model';

import { Header } from '../header';

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

import { NewAnswers, NewAnswer } from '../model/new-answers';

@Component({

selector: 'survey',

templateUrl: './survey.component.html',

styleUrls: ['./survey.component.css']

})

export class SurveyComponent implements OnInit, OnDestroy {

private id: number;

private sub: any;

public surveys: Survey = {} as Survey;

private newAnswers: NewAnswer[] = [];

constructor (private http: HttpClient, private route: ActivatedRoute, private router: Router) {}

//Send Answer to Server

public answerSurvey() {

const answers: NewAnswers = {NewAnswers: this.newAnswers};

this.http.post(Const.baseUrl + 'answerSurvey', answers).subscribe(data => {

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

console.warn(answers);

}, err => {

console.warn('error', err);

});

}

//If a Choice gets selected or deselcted it gets stored or deleted

public checkBox(id: number) {

const answer: NewAnswer = {IdChoice: id};

const index = this.newAnswers.indexOf(answer);

if (index === -1) {

this.newAnswers.push(answer);

} else {

this.newAnswers.splice(index, 1);

}

}

public ngOnInit() {

this.sub = this.route.params.subscribe(params => {

this.id = +params['id'];

this.http.get<Survey>(Const.baseUrl + 'getSurvey/' + this.id).subscribe(data => {

this.surveys = data;

}, err => {

console.warn('error', err);

});

});

}

public ngOnDestroy() {

this.sub.unsubscribe();

}

}

### survey-create.component.css

h1{

word-wrap: break-word;

-webkit-hyphens: auto;

-moz-hyphens: auto;

-ms-hyphens: auto;

-o-hyphens: auto;

hyphens: auto;

}

### survey-create.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row" style="margin:5px;">

<div class="col-md-6 col-md-offset-3 col-sm-12 text-center">

<ul class="list-group">

<li class="list-group-item">

<h1><input type="Text" placeholder="Frage" class="form-control" [(ngModel)]="question" maxlength="255"></h1>

</li>

<li class="list-group-item" \*ngFor="let choice of choices">

<input #ChoiceInput="ngModel" type="Text" placeholder="Antwortmöglichkeit" class="form-control" [(ngModel)]="choice.Choice" (focus)="checkList(ChoiceInput.touched)" maxlength="140">

</li>

</ul>

<button type="button" class="btn btn-primary" (click)="createSurvey()">Senden</button>

</div> <!-- /container -->

</div>

### survey-create.component.ts

import { Component, OnInit, OnDestroy } from '@angular/core';

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

import { Const } from '../const';

import { Survey, Choice } from '../model/survey.model';

import { Header } from '../header';

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

import { NewAnswers, NewAnswer } from '../model/new-answers';

@Component({

selector: 'survey',

templateUrl: './survey-create.component.html',

styleUrls: ['./survey-create.component.css']

})

export class SurveyCreateComponent {

public question: string;

public choices: Choice[] = [{Id: 0, Choice: ''}];

private headers: Header;

constructor (private http: HttpClient, private router: Router) {

this.headers = new Header();

}

createSurvey() {

const survey = { Id: 0, Question: this.question, Choices: this.choices.filter(c => c.Choice !== '')} as Survey;

//send new Survey to Server

this.http.post<number>(Const.baseUrl + 'createSurvey', survey, { headers: this.headers.headers }).subscribe(data => {

const newId = data;

this.router.navigate(['/survey', newId]);

}, err => {

console.warn('error', err);

});

}

//check if new Choice input should be created

checkList(touched: boolean) {

if (!touched && this.choices.length <= 10) {

this.choices.push({Id: 0, Choice: ''});

}

}

}

### result.component.css

.charts-wrapper{

min-height: 500px;

}

### result.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div class="row">

<div class="col-md-6 col-md-offset-3 col-sm-12 charts-wrapper">

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

[results]="chartData"

[legend]="true"

[labels]="false"

[doughnut]="false">

</ngx-charts-pie-chart>

</div> <!-- /container -->

</div>

### result.component.ts

import { GivenAnswers } from './../model/new-answers';

import { Component, OnInit, OnDestroy } from '@angular/core';

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

import { Const } from '../const';

import { Survey, Choice } from '../model/survey.model';

import { Header } from '../header';

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

import { NewAnswers, NewAnswer } from '../model/new-answers';

@Component({

selector: 'result',

templateUrl: './result.component.html',

styleUrls: ['./result.component.css']

})

export class ResultComponent implements OnInit, OnDestroy {

private id: number;

private sub: any;

private headers: Header;

public chartData:

{

name: string

value: number

}[];

constructor (private http: HttpClient, private route: ActivatedRoute) {

this.headers = new Header();

}

public ngOnInit() {

this.sub = this.route.params.subscribe(params => {

this.id = +params['id'];

//load Answers from Server and store it for ngx-charts

this.http.get<GivenAnswers>(Const.baseUrl + 'getAnswers/' + this.id, { headers: this.headers.headers } ).subscribe(data => {

this.chartData = data.GivenAnswers.map((ga) => ({ name: ga.Choice, value: ga.Amount }));

}, err => {

console.warn('error', err);

});

});

}

public ngOnDestroy() {

this.sub.unsubscribe();

}

}

### app.component.css

.navbar-header {

float: left;

padding: 15px;

text-align: center;

width: 100%;

}

.navbar-brand {float:none;}

### app.component.html

<!--The content below is only a placeholder and can be replaced.-->

<nav class="navbar navbar-default">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<a class="navbar-brand" routerLink="/login" \*ngIf="!isLogedin">Login</a>

<a class="navbar-brand" routerLink="/register" \*ngIf="!isLogedin">Registrieren</a>

<a class="navbar-brand" routerLink="/createSurvey" \*ngIf="isLogedin">Neue Umfrage erstellen</a>

<a class="navbar-brand" routerLink="/login" \*ngIf="isLogedin" (click)="logout()">Logout</a>

</div>

</nav>

<router-outlet></router-outlet>

### app.component.ts

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'app';

public get isLogedin() {

const savedUsername = sessionStorage.getItem('username');

return savedUsername !== null;

}

logout() {

sessionStorage.removeItem('username');

sessionStorage.removeItem('password');

}

}

### user-model.component.ts

export interface User {

Id: number;

Username: string;

Password: string;

}

### survey-model.component.ts

export interface Survey {

Id: number;

Question: string;

Choices: Choice[];

}

export interface Choice {

Id: number;

Choice: string;

}

### answer-model.component.ts

export interface NewAnswers {

NewAnswers: NewAnswer[];

}

export interface NewAnswer {

IdChoice: number;

}

export interface GivenAnswer {

IdChoice: number;

Choice: string;

Amount: number;

}

export interface GivenAnswers {

GivenAnswers: GivenAnswer[];

}

### app-routing.components.ts

import { SurveyCreateComponent } from './survey-create/survey-create.component';

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

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

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

import { RegisterComponent } from './register/register.component';

import { MainAdminComponent } from './main-admin/main-admin.component';

import { SurveyComponent } from './survey/survey.component';

import { ResultComponent } from './results/result.component';

import { AuthGuardService } from './guard';

//Routen

const routes: Routes = [

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

{ path: 'register', component: RegisterComponent},

{ path: 'admin', component: MainAdminComponent, canActivate: [AuthGuardService]},

{ path: 'survey/:id', component: SurveyComponent},

{ path: 'createSurvey', component: SurveyCreateComponent, canActivate: [AuthGuardService]},

{ path: 'result/:id', component: ResultComponent, canActivate: [AuthGuardService]},

{ path: '', component: LoginComponent}];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

### guard.ts

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

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

@Injectable()

//checks if User ist logged in

export class AuthGuardService implements CanActivate {

constructor( private router: Router) {}

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {

if (sessionStorage.getItem('username') !== null) {

return true;

} else {

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

return false;

}

}

}

### header.ts

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

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

//Header-Object for Server Requests

export class Header {

public headers: HttpHeaders = new HttpHeaders();

constructor() {

this.headers = this.headers

.append('username', sessionStorage.getItem('username'))

.append('password', sessionStorage.getItem('password'));

}

}

### const.ts

export class Const {

public static baseUrl: string;

}

package.json  
{

"name": "my-app",

"version": "0.0.0",

"license": "MIT",

"scripts": {

"ng": "ng",

"start": "ng serve",

"build": "ng build",

"test": "ng test",

"lint": "ng lint",

"e2e": "ng e2e"

},

"private": true,

"dependencies": {

"@angular/animations": "^4.4.6",

"@angular/common": "^4.4.6",

"@angular/compiler": "^4.4.6",

"@angular/core": "^4.4.6",

"@angular/forms": "^4.4.6",

"@angular/http": "^4.4.6",

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

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

"@angular/router": "^4.4.6",

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

"bootstrap": "^3.3.7",

"core-js": "^2.4.1",

"d3": "^4.11.0",

"ngx-bootstrap": "^2.0.0-beta.8",

"reflect-metadata": "^0.1.10",

"rxjs": "^5.4.2",

"zone.js": "^0.8.14"

},

"devDependencies": {

"@angular/cli": "1.4.9",

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

"@angular/language-service": "^4.4.6",

"@types/jasmine": "~2.5.53",

"@types/jasminewd2": "~2.0.2",

"@types/node": "~6.0.60",

"codelyzer": "~3.2.0",

"jasmine-core": "~2.6.2",

"jasmine-spec-reporter": "~4.1.0",

"karma": "~1.7.0",

"karma-chrome-launcher": "~2.1.1",

"karma-cli": "~1.0.1",

"karma-coverage-istanbul-reporter": "^1.2.1",

"karma-jasmine": "~1.1.0",

"karma-jasmine-html-reporter": "^0.2.2",

"protractor": "~5.1.2",

"ts-node": "~3.2.0",

"tslint": "~5.7.0",

"typescript": "~2.3.3"

}

}

## Backend

### server.go

package main

import (

    "encoding/json"

    "fmt"

    "net/http"

    "server/authorisation"

    "server/database"

    "server/survey/controler2"

    "server/survey/model/answers"

    "server/survey/model/survey"

    "server/user/controler"

    "server/user/model/user"

    "strconv"

    "github.com/gorilla/handlers"

    "github.com/gorilla/mux"

)

var db database.Database

func Register(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    decoder := json.NewDecoder(req.Body)

    var user user.User

    err := decoder.Decode(&user)

    if err != nil {

        fmt.Println("c")

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    err = controler.RegisterUser(user.Username, user.Password, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.WriteHeader(http.StatusNoContent)

}

func Login(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    decoder := json.NewDecoder(req.Body)

    var user user.User

    err := decoder.Decode(&user)

    if err != nil {

        fmt.Println("a")

        http.Error(res, http.StatusText(http.StatusUnauthorized), http.StatusUnauthorized)

        return

    }

    \_, err = authorisation.Authorisate(user.Username, user.Password, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusUnauthorized), http.StatusUnauthorized)

        return

    }

    res.WriteHeader(http.StatusNoContent)

}

func CreateSurvey(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    decoder := json.NewDecoder(req.Body)

    var newSurvey survey.Survey

    err := decoder.Decode(&newSurvey)

    if err != nil {

        fmt.Println("a")

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    username := req.Header.Get("username")

    password := req.Header.Get("password")

    id, err := authorisation.Authorisate(username, password, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusUnauthorized), http.StatusUnauthorized)

        return

    }

    result, err := controler2.CreateSurvey(&newSurvey, \*id, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.Header().Set("Content-Type", "application/json")

    res.Write(\*result)

}

func GetSurvey(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    vars := mux.Vars(req)

    id, \_ := strconv.Atoi(vars["id"])

    result, err := controler2.GetSurvey(id, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.Header().Set("Content-Type", "application/json")

    res.Write(\*result)

}

func GetSurveyByUser(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    username := req.Header.Get("username")

    password := req.Header.Get("password")

    id, err := authorisation.Authorisate(username, password, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusUnauthorized), http.StatusUnauthorized)

        return

    }

    fmt.Println(\*id)

    result, err := controler2.GetSurveyByUser(\*id, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.Header().Set("Content-Type", "application/json")

    res.Write(\*result)

}

func AnswerSurvey(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    decoder := json.NewDecoder(req.Body)

    var newAnswers answers.NewAnswers

    err := decoder.Decode(&newAnswers)

    if err != nil {

        fmt.Println("a")

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    err = controler2.SetAnswers(&newAnswers, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.WriteHeader(http.StatusNoContent)

}

func GetAnswers(res http.ResponseWriter, req \*http.Request) {

    fmt.Println("d")

    vars := mux.Vars(req)

    id, \_ := strconv.Atoi(vars["id"])

    username := req.Header.Get("username")

    password := req.Header.Get("password")

    \_, err := authorisation.Authorisate(username, password, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusUnauthorized), http.StatusUnauthorized)

        return

    }

    result, err := controler2.GetAnswers(id, db.Db)

    if err != nil {

        fmt.Println(err)

        http.Error(res, http.StatusText(http.StatusBadRequest), http.StatusBadRequest)

        return

    }

    res.Header().Set("Content-Type", "application/json")

    res.Write(\*result)

}

func main() {

    db = database.Database{}

    err := db.Connect("survey\_db", "root", "")

    if err != nil {

        fmt.Println(err)

    }

    defer db.Close()

    r := mux.NewRouter()

    r.HandleFunc("/api/login", Login).Methods("POST")

    r.HandleFunc("/api/register", Register).Methods("POST")

    r.HandleFunc("/api/createSurvey", CreateSurvey).Methods("POST")

    r.HandleFunc("/api/getSurvey/{id}", GetSurvey).Methods("GET")

    r.HandleFunc("/api/getSurveyByUser", GetSurveyByUser).Methods("GET")

    r.HandleFunc("/api/answerSurvey", AnswerSurvey).Methods("POST")

    r.HandleFunc("/api/getAnswers/{id}", GetAnswers).Methods("GET")

    headersOk := handlers.AllowedHeaders([]string{"content-type", "username", "password"})

    originsOk := handlers.AllowedOrigins([]string{"\*"})

    //methodsOk := handlers.AllowedMethods([]string{"\*"})

    http.ListenAndServe(":8000", handlers.CORS(originsOk, headersOk)(r))

}

### usercontroler.go

package controler

import (

    "database/sql"

    "server/bcrypt"

    "server/user/model"

)

//RegisterUser registers a user

func RegisterUser(username string, password string, db \*sql.DB) error {

    hashedPassword, err := bcrypt.Encrypt([]byte(password))

    if err != nil {

        return err

    }

    err = model.RegisterUser(username, string(hashedPassword), db)

    if err != nil {

        return err

    }

    return nil

}

### surveycontroler.go

package controler2

import (

    "database/sql"

    "encoding/json"

    "server/survey/model"

    "server/survey/model/answers"

    "server/survey/model/survey"

)

func CreateSurvey(newSurvey \*survey.Survey, idUser int, db \*sql.DB) (\*[]byte, error) {

    idSurvey, err := model.CreateSurvey(newSurvey, idUser, db)

    if err != nil {

        return nil, err

    }

    idSurveyJSON, err := json.Marshal(idSurvey)

    if err != nil {

        return nil, err

    }

    return &idSurveyJSON, nil

}

func GetSurvey(idSurvey int, db \*sql.DB) (\*[]byte, error) {

    survey, err := model.GetSurvey(idSurvey, db)

    if err != nil {

        return nil, err

    }

    surveyJSON, err := json.Marshal(survey)

    if err != nil {

        return nil, err

    }

    return &surveyJSON, nil

}

func GetSurveyByUser(idUser int, db \*sql.DB) (\*[]byte, error) {

    surveys, err := model.GetSurveyByUser(idUser, db)

    if err != nil {

        return nil, err

    }

    surveysJSON, err := json.Marshal(surveys)

    if err != nil {

        return nil, err

    }

    return &surveysJSON, nil

}

func GetAnswers(idSurvey int, db \*sql.DB) (\*[]byte, error) {

    answers, err := model.GetAnswers(idSurvey, db)

    if err != nil {

        return nil, err

    }

    answersJSON, err := json.Marshal(answers)

    if err != nil {

        return nil, err

    }

    return &answersJSON, nil

}

func SetAnswers(newAnswer \*answers.NewAnswers, db \*sql.DB) error {

    err := model.AnswerSurvey(newAnswer, db)

    if err != nil {

        return err

    }

    return nil

}

### authorisation.go

package authorisation

import (

    "database/sql"

    "server/bcrypt"

    "server/user/model"

)

//Authorisate checks authorisation

func Authorisate(username string, password string, db \*sql.DB) (\*int, error) {

    user, err := model.Login(username, db)

    if err != nil {

        return nil, err

    }

    err = bcrypt.CompareHashes([]byte(user.Password), []byte(password))

    if err != nil {

        return nil, err

    }

    return &user.Id, nil

}

### bcrypt.go

package bcrypt

import (

    "golang.org/x/crypto/bcrypt"

)

//Encrypt a string

func Encrypt(toEncrypt []byte) ([]byte, error) {

    return bcrypt.GenerateFromPassword(toEncrypt, bcrypt.DefaultCost)

}

//CompareHases returns an error if the hashes arnt equal

func CompareHashes(toCompare1 []byte, toCompare2 []byte) error {

    return bcrypt.CompareHashAndPassword(toCompare1, toCompare2)

}

### usermodel.go

package model

import (

    "database/sql"

    "errors"

    "server/user/model/user"

)

//RegisterUser returns a user-object if succsesfuly created in db

func RegisterUser(username string, password string, db \*sql.DB) error {

    tx, err := db.Begin()

    stmt, err := db.Prepare("INSERT User SET username=?, password=?")

    if err != nil {

        tx.Rollback()

        return err

    }

    \_, err = stmt.Exec(username, password)

    if err != nil {

        tx.Rollback()

        return err

    }

    tx.Commit()

    return nil

}

//Login returns a user-object if found in db

func Login(username string, db \*sql.DB) (\*user.User, error) {

    stmt, err := db.Prepare("SELECT \* FROM User WHERE username=?")

    if err != nil {

        return nil, err

    }

    res, err := stmt.Query(username)

    if err != nil {

        return nil, err

    }

    for res.Next() {

        var uid int

        var username string

        var password string

        err = res.Scan(&uid, &username, &password)

        if err != nil {

            return nil, err

        }

        return &user.User{uid, username, password}, nil

    }

    return nil, errors.New("No User")

}

### surveymodel.go

package model

import (

    "database/sql"

    "server/survey/model/survey"

    "fmt"

)

func CreateSurvey(newSurvey \*survey.Survey, idUser int, db \*sql.DB) (\*int, error) {

    tx, err := db.Begin()

    stmt, err := db.Prepare("INSERT Survey SET question=?, User\_idUser=?")

    if err != nil {

        tx.Rollback()

        return nil, err

    }

    res, err := stmt.Exec(newSurvey.Question, idUser)

    if err != nil {

        tx.Rollback()

        return nil, err

    }

    id, \_ := res.LastInsertId()

    idint := int(id)

    for \_, choice := range newSurvey.Choices {

        stmt, err := db.Prepare("INSERT Choices SET choice=?, Survey\_idSurvey=?")

        if err != nil {

            tx.Rollback()

            return nil, err

        }

        \_, err = stmt.Exec(choice.Choice, idint)

        if err != nil {

            tx.Rollback()

            return nil, err

        }

    }

    tx.Commit()

    return &idint, nil

}

func GetSurvey(id int, db \*sql.DB) (\*survey.Survey, error) {

    stmt, err := db.Prepare("SELECT Survey.IdSurvey, Survey.question, Choices.idChoices, Choices.choice FROM Survey INNER JOIN Choices ON Survey.IdSurvey = Choices.Survey\_idSurvey WHERE Survey.IdSurvey=?")

    if err != nil {

        return nil, err

    }

    res, err := stmt.Query(id)

    if err != nil {

        return nil, err

    }

    var sid int

    var question string

    choices := []survey.Choice{}

    for res.Next() {

        var cid int

        var choice string

        err = res.Scan(&sid, &question, &cid, &choice)

        if err != nil {

            return nil, err

        }

        choices = append(choices, survey.Choice{Id: cid, Choice: choice})

    }

    return &survey.Survey{Id: sid, Question: question, Choices: choices}, nil

}

func GetSurveyByUser(idUser int, db \*sql.DB) (\*[]survey.Survey, error) {

    stmt, err := db.Prepare("SELECT Survey.IdSurvey, Survey.question FROM Survey WHERE Survey.User\_idUser=?")

    if err != nil {

        return nil, err

    }

    fmt.Println(idUser)

    res, err := stmt.Query(idUser)

    if err != nil {

        return nil, err

    }

    surveys := []survey.Survey{}

    for res.Next() {

        var sid int

        var question string

        err = res.Scan(&sid, &question)

        if err != nil {

            return nil, err

        }

        surveys = append(surveys, survey.Survey{Id: sid, Question: question, Choices: []survey.Choice{}})

    }

    return &surveys, nil

}

### answermodel.go

package model

import (

    "database/sql"

    "server/survey/model/answers"

)

func GetAnswers(idSurvey int, db \*sql.DB) (\*answers.GivenAnswers, error) {

    stmt, err := db.Prepare("SELECT Choices.idChoices, Choices.choice FROM Survey INNER JOIN Choices ON Survey.IdSurvey = Choices.Survey\_idSurvey WHERE Survey.IdSurvey=?")

    if err != nil {

        return nil, err

    }

    res, err := stmt.Query(idSurvey)

    if err != nil {

        return nil, err

    }

    newanswers := []answers.GivenAnswer{}

    for res.Next() {

        var amount int

        var idChoice int

        var choice string

        err = res.Scan(&idChoice, &choice)

        if err != nil {

            return nil, err

        }

        stmt, err := db.Prepare("SELECT COUNT(Results.Choices\_idChoices) FROM Results where Results.Choices\_idChoices=?")

        if err != nil {

            return nil, err

        }

        res2, err := stmt.Query(idChoice)

        if err != nil {

            return nil, err

        }

        for res2.Next() {

            err = res2.Scan(&amount)

            if err != nil {

                return nil, err

            }

        }

        newanswers = append(newanswers, answers.GivenAnswer{IdChoice: idChoice, Choice: choice, Amount: amount})

    }

    return &answers.GivenAnswers{newanswers}, nil

}

func AnswerSurvey(newAnswer \*answers.NewAnswers, db \*sql.DB) error {

    tx, \_ := db.Begin()

    for \_, answer := range newAnswer.NewAnswers {

        stmt, err := db.Prepare("INSERT Results SET Choices\_idChoices=?")

        if err != nil {

            tx.Rollback()

            return err

        }

        \_, err = stmt.Exec(answer.IDChoice)

        if err != nil {

            tx.Rollback()

            return err

        }

    }

    tx.Commit()

    return nil

}

### answer.go

package answers

type GivenAnswer struct {

    IdChoice int

    Choice string

    Amount int

}

type GivenAnswers struct {

    GivenAnswers []GivenAnswer

}

type NewAnswer struct {

    IDChoice int

}

type NewAnswers struct {

    NewAnswers []NewAnswer

}

### survey.go

package survey

type Survey struct {

    Id int

    Question string

    Choices []Choice

}

type Choice struct {

    Id int

    Choice string

}

### user.go

package user

type User struct {

    Id int

    Username string

    Password string

}

### database.go

package database

import (

    "database/sql"

    \_ "github.com/go-sql-driver/mysql"

)

type Database struct {

    Db \*sql.DB

}

func (database \*Database) Connect(connection string, username string, password string) error {

    db, err := sql.Open("mysql", username+":"+password+"@/"+connection)

    if err != nil {

        return err

    }

    err = db.Ping()

    if err != nil {

        return err

    }

    database.Db = db

    return nil

}

func (database Database) Close() {

    database.Close()

}

### Datenbank-Export (SQL)

-- MySQL Script generated by MySQL Workbench

-- Fri Nov 3 11:12:31 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 survey\_db

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

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

-- Schema survey\_db

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

CREATE SCHEMA IF NOT EXISTS `survey\_db` DEFAULT CHARACTER SET utf8 ;

USE `survey\_db` ;

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

-- Table `survey\_db`.`User`

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

CREATE TABLE IF NOT EXISTS `survey\_db`.`User` (

`idUser` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(45) NOT NULL,

`password` BINARY(60) NOT NULL,

PRIMARY KEY (`idUser`),

UNIQUE INDEX `idUser\_UNIQUE` (`idUser` ASC),

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

ENGINE = InnoDB;

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

-- Table `survey\_db`.`Survey`

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

CREATE TABLE IF NOT EXISTS `survey\_db`.`Survey` (

`idSurvey` INT NOT NULL AUTO\_INCREMENT,

`question` VARCHAR(255) NOT NULL,

`User\_idUser` INT NOT NULL,

PRIMARY KEY (`idSurvey`),

INDEX `fk\_Survey\_User\_idx` (`User\_idUser` ASC),

UNIQUE INDEX `idSurvey\_UNIQUE` (`idSurvey` ASC),

CONSTRAINT `fk\_Survey\_User`

FOREIGN KEY (`User\_idUser`)

REFERENCES `survey\_db`.`User` (`idUser`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `survey\_db`.`Choices`

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

CREATE TABLE IF NOT EXISTS `survey\_db`.`Choices` (

`idChoices` INT NOT NULL AUTO\_INCREMENT,

`choice` VARCHAR(140) NOT NULL,

`Survey\_idSurvey` INT NOT NULL,

PRIMARY KEY (`idChoices`),

INDEX `fk\_Choices\_Survey1\_idx` (`Survey\_idSurvey` ASC),

UNIQUE INDEX `idChoices\_UNIQUE` (`idChoices` ASC),

CONSTRAINT `fk\_Choices\_Survey1`

FOREIGN KEY (`Survey\_idSurvey`)

REFERENCES `survey\_db`.`Survey` (`idSurvey`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `survey\_db`.`Results`

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

CREATE TABLE IF NOT EXISTS `survey\_db`.`Results` (

`Choices\_idChoices` INT NOT NULL AUTO\_INCREMENT,

INDEX `fk\_results\_Choices1\_idx` (`Choices\_idChoices` ASC),

CONSTRAINT `fk\_results\_Choices1`

FOREIGN KEY (`Choices\_idChoices`)

REFERENCES `survey\_db`.`Choices` (`idChoices`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

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

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