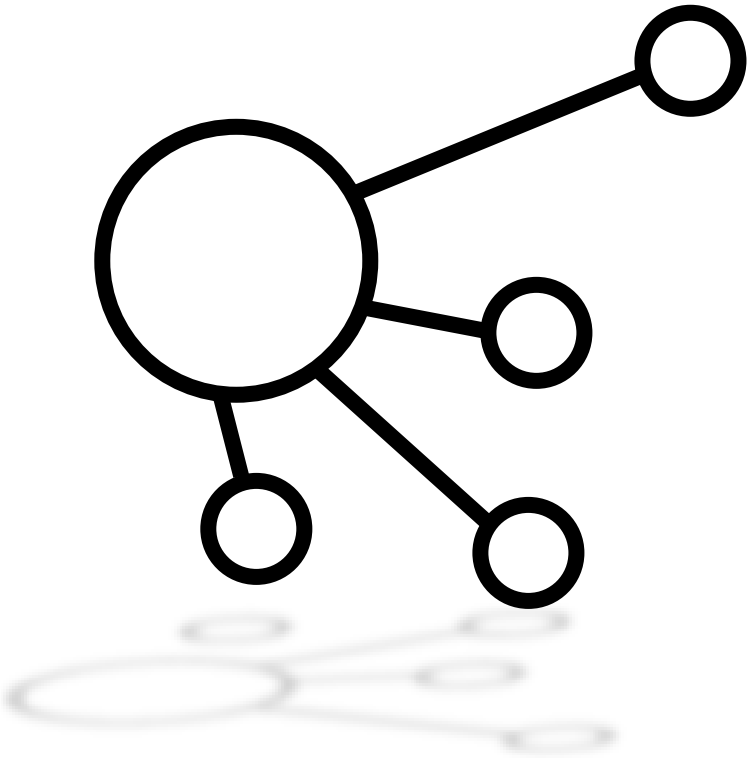
Mandantenfähige Website

Autor: Ruanin Khankah
Datum: 24. Mai 2022



Inhaltsverzeichnis

Version	Datum	Bemerkungen	Bearbeiter
1.0	09.05.2022	Dokument erstellt	Ruanin Khankah
1.1	23.05.2022	Bericht abgeschlossen	Ruanin Khankah

Glossar

Begriff	Definition
AJAX	Asynchronous Javascript an XML, bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server.
API	Eine API (Application Programming Interface) ist ein Satz von Befehlen, Funktionen, Protokollen und Objekten, die Programmierer verwenden können, um eine Software zu erstellen oder mit einem externen System zu interagieren.
Controller	Der Controller verwaltet das View und das Model. Er wird vom View über Benutzerinteraktionen informiert, wertet diese aus und nimmt daraufhin Anpassungen an der View sowie Änderungen an den Daten im Model vor.
IPERKA	Projektplanungsmethode
Model	Das Model enthält Daten, die vom View dargestellt werden. Es ist von View und Controller unabhängig.
MVC	Model View Controller ist ein Muster zur Unterteilung einer Software in drei Komponenten, Datenmodell (model), Präsentation (view) und Programmsteuerung (controller).
Repository	Verwaltetes Verzeichnis, zur Speicherung und Beschreibung digitaler Objekte.
Sessionvariable	Dies ist eine 'Superglobale' oder automatisch globale Variable. Dies bedeutet, dass sie innerhalb des Skripts in jedem Geltungsbereich sichtbar ist. Es ist nicht nötig, sie mit global $variable bekannt zu machen, um aus Funktionen oder Methoden darauf zuzugreifen.
View	Das View ist für die Darstellung der Daten des Models und die Realisierung der Benutzerinteraktionen zuständig. Sie kennt das Model, dessen Daten sie präsentiert, ist aber nicht für die Verarbeitung dieser Daten zuständig. Des Weiteren ist das View vom Controller unabhängig.

1 IPA-Bericht Teil 1

1.1 Titel der Arbeit

Mandantenfähige Website

1.2 Arbeitsbereiche

- PHP
- JavaScript
- SQL

1.3 Vorwort

Diese Dokumentation gehört zur IPA von Ruanin Khankah, welcher bei der Firma Confidis AG angestellt ist. In diesem Dokument sind die Arbeit und das Vorgehen dokumentiert. Die Dokumentation ist in 2 Teile gegliedert Im ersten Teil werden die detaillierte Aufgabenstellung und der Ablauf der Arbeit aufgezeigt. Hier werden die vorhandenen Vorkenntnisse, die benutzten Mittel und Methoden aufgezählt. Im zweiten Teil wird das Projekt auch wirklich dokumentiert und beschrieben, es wird auch erwähnt welche Probleme aufgetreten sind. Durch die ganze Arbeit hinweg wurde die Projektplanungsmethode IPERKA eingesetzt

1.4 Detaillierte Aufgabenstellung

Dazu wird ein neues Formular entwickelt, mit dem die Anfragen über die Website entgegengenommen und per E-Mail an die Zweigstelle übermittelt werden.

- Felder:
Vorname (Es sind nur Buchstaben erlaubt. Pflichtfeld)
Name (Es sind nur Buchstaben erlaubt. Pflichtfeld)
E-Mail (Validierung-Mail, Pflichtfeld)
Telefon (Validierung mit PhoneLibrary, Pflichtfeld)
Firma
Funktion
Strasse /Nr.
PLZ (Pflichtfeld)
Ort (Pflichtfeld, Automatischer Vorschlag nach PLZ falls unter PLZ Tabelle vorhanden ist)
Land (Dropdown von Länder Tabelle)
Beruf (Dropdown von Berufstabelle aus Datenbank, Pflichtfeld)
Ich suche (Textfeld, Pflichtfeld)

- Die Datenschutzerklärung muss vor der Übermittlung angenommen werden. Muss sichergestellt werden bis es nach unten gescrollt werden.
- die Webapplikation soll in Responsive Design programmiert sein. Das Formular wird auf der Webseite von Planova eingebunden.
- die im „Look und Feel“ der bestehenden Planova Website ist (www.planova.ch). Die Webapplikation soll in den neuesten Versionen IE, Firefox und Chrome sauber dargestellt sein.

- SQL-Injection muss verhindert werden.
- Die eingegebenen Felder sollten gespeichert werden (SESSION), falls mehrere Vakanzen zu besetzen sind.
- Die Kunden sollten informiert werden, falls der Beruf der Stellenmeldepflicht unterliegt. Sollte direkt nach Berufsauswahl informiert werden, ob es Stellenmeldepflichtig ist oder nicht.
- Nach erfolgreicher Übermittlung wird der Kunde darüber informiert, dass ein Personalverantwortlicher innerhalb von 48 Stunden auf seine Anfrage antworten wird. Andernfalls kann er direkt unter 0848 752 668 anrufen.
- Spamschutz für Formular ist zwingend.
- Konnte eine E-Mail nicht übermittelt werden, muss eine Logdatei erstellt werden.
- Zusätzlich sollte das Kontaktformular in der Datenbank gespeichert werden.
- Es ist ein Projektjournal zu führen, da noch mit dem Vertrieb die notwendigen Felder abgestimmt werden und über den Ablauf informiert werden sollen.
- Formular müsste Funktionsweise getestet werden (Blackbox Test).
- MVC gemäss Jobcircle Framework verwenden.

Wenn das Formular ausgefüllt wurde, muss der Antrag an info@planova.ch gesendet werden. Gleichzeitig würde der Besucher benachrichtigt, dass sein Formular abgeschickt wurde und er ein Benutzerkonto anlegen kann.
Dabei ist es wichtig, dass die vorhandenen Daten der Kunden in der Datenbank gespeichert werden. Ist der Kunde bereits eingeloggt, wird er zur Übersichtsseite mit dem eingereichten Personalbedarf weitergeleitet.

Liste Felder:
Erstellungsdatum, Firma, Kontakt, Beruf, Meldepflichtig Status.

1.5 Mittel und Methoden

Windows Computer
Netbeans
PHP, JAVASCRIPT, MySQL, HTML

1.6 Vorkenntnisse

Arbeiten mit PHP, MySQL, Bootstrap. Der Kandidat durfte drei Monate mit den bestehenden Frameworks arbeiten.

1.7 Vorarbeiten

Der Kandidat durfte auf die bestehenden Frameworks das Weiterempfehlungs-System programmieren.

1.8 Neue Lerninhalte

Regex wäre für den Kandidaten neu. Der Kandidat kann es unter https://www.php.net/ nachschlagen.

1.9 Arbeiten in den letzten 6 Monaten

Weiterempfehlungsprogramm

1.10 Mittel und Methoden

I 1 Autocomplete

Bezeichnung
Autocomplete

Definition (Leitfrage)
Der Ort wurde eingegeben und die Website wird nicht neu geladen. Bei der Eingabe des Ortes sollte die Postleitzahl und das Land ausgewählt werden.

Gütestufe 3
Der Ort Zürich wird eingegeben und es sollten mehrere Postleitzahlen vorgeschlagen werden.

Gütestufe 2
Der Ort, PLZ und Land wird korrekt übernommen.

Gütestufe 1
Ort und Land wird ohne neuladen angezeigt.

Gütestufe 0
Daten werden nicht übernommen.

I 2 Kommentare im Quellcode

Bezeichnung
Kommentare im Quellcode

Definition (Leitfrage)
Wurde der Sourcecode der Applikation ausreichend kommentiert?

Gütestufe 3
Der Sourcecode der Applikation ist vollumfänglich kommentiert:
1. Funktionen, Parameter, Rückgabewerte,
2. Wichtige Stellen im Sourcecode,
3. weitere zusätzliche/nützliche Kommentare.

Gütestufe 2
Der Sourcecode der Applikation ist im Grossen und Ganzen kommentiert. Einer der genannten Punkte könnte präziser sein.

Gütestufe 1
Der Sourcecode der Applikation ist nur teilweise kommentiert.

Gütestufe 0
Der Sourcecode der Applikation ist unzureichend kommentiert.

I 3 Implementierung von Lösungen (Programmieren)

Bezeichnung
Implementierung von Lösungen (Programmieren)

Definition (Leitfrage)
Ist der Kandidat in der Lage die vorgeschlagenen Lösungen zu implementieren?

Gütestufe 3
Der Code entspricht den Erwartungen und kann bedenkenlos verwendet werden. Die Sprachmittel wurden richtig eingesetzt.

Gütestufe 2
Der Code weist einzelne Mängel auf. Das Resultat muss vor dem Einsatz überarbeitet werden. Die Sprachmittel sind nicht immer richtig gewählt.

Gütestufe 1
Der Code weist klare Mängel auf. Das Resultat muss vor dem Einsatz gründlich überarbeitet werden. Die Sprachmittel sind nicht richtig gewählt. Oder: der Kandidat versteht den Zweck der Sprachelemente nicht.

Gütestufe 0
Der Code ist deutlich unter den Erwartungen und kann nicht wirklich gebraucht werden.

I 4 Plausibilisierung der Benutzer-Eingaben

Bezeichnung
Plausibilisierung der Benutzer-Eingaben

Definition (Leitfrage)
Werden die Eingaben des Benutzers überprüft?

Gütestufe 3
Alle Eingabefelder werden überprüft. Es ist eindeutig gekennzeichnet, welche Felder Pflichtfelder sind. Für den Benutzer ist ersichtlich, welche Wertebereiche zulässig sind. Findet die Plausibilisierung eine Fehleingabe, so wird der Benutzer mit konkreten Hinweisen geführt, das entsprechende Feld wird aktiviert.

Gütestufe 2
Plausibilisierung findet statt, Feedback an Benutzer ist mangelhaft/nicht eindeutig/unvollständig. Nur korrekte Daten werden übermittelt.

Gütestufe 1
Eingaben werden plausibilisiert, aber bei Fehlern oder fehlenden Eingaben sind die bisher gemachten Eingaben verloren oder die fehlerhaften Eingaben werden trotzdem übermittelt. Oder: es werden nicht alle Eingaben ueberprueft, welche ueberprueft werden sollten.

Gütestufe 0
Es findet keine Plausibilisierung statt.

I 5 Richtlinien und Standards

Bezeichnung
Richtlinien und Standards

Definition (Leitfrage)
Wurden die Programmierrichtlinien eingehalten? Unternehmensrichtlinien erweitern die sprachbezogenen Basisstandards. Tabellen werden CamelCase erstellt. Array mit arr begonnen. Objekte werden mit Obj begonnen.

Gütestufe 3
Die Basisstandards und Unternehmensrichtlinien wurden komplett eingehalten.

Gütestufe 2
Die Basisstandards und Unternehmensrichtlinien wurden weitgehend eingehalten.

Gütestufe 1
Nur die Basisstandards wurden eingehalten.

Gütestufe 0
Weder Basisstandards noch Unternehmensrichtlinien wurden eingehalten.

I 6 Benutzerfreundlichkeit: GUI, Bedienung

Bezeichnung
Benutzerfreundlichkeit: GUI, Bedienung

Definition (Leitfrage)
Ist das Produkt benutzerfreundlich?

Gütestufe 3
Die Bedienung ist dem Problem/dem Prozess angepasst und intuitiv. Alle GUI-Elemente sind sinnvoll gewählt. Parameter-Felder sind aussagekräftig angeschrieben. Menu-, Befehls- und Masken-Hierarchie oder Masken-Abfolge sind sinnvoll aufgebaut.

Gütestufe 2
Die Bedienung ist zum Teil intuitiv, zum Teil aber nur mit Erklärung verständlich. Menu-Punkte und Befehlsknopf-Beschriftungen sind unklar oder missverständlich.

Gütestufe 1
Die Bedienung ist nur mit Hilfe einer Beschreibung möglich. Die Logik hat mit dem Prozess wenig zu tun oder ist kaum ersichtlich. Parameter werden zusammenhangslos eingefordert oder können zu Unzeiten gesetzt werden (wenn deren Änderung stört).

Gütestufe 0
Selbst mit Online-Help oder Beschreibung ist die Bedienung ein Buch mit sieben Siegeln.

I7 Sicherheitsanalyse (Web-Applikation):

Die Web-Applikation muss sicherstellen, dass sie beliebten Angriffsmethoden wie JavaScript-Injection und SQL-Injection widersteht.	
GS 3: Es wurde eine gründliche Sicherheitsanalyse durchgeführt. Die notwendigen Massnahmen wurden ergriffen und entsprechen dem "state of the art". Mindestens drei verschiedene, typische Injectionsversuche werden erfolgreich abgewehrt und sind dokumentiert.	GS 2: Ein Aspekt ist mangelhaft erfüllt
GS 1: Zwei Aspekt ist mangelhaft erfüllt	GS 0: Mehr als zwei Aspekte sind mangelhaft erfüllt.

Dieses Kriterium wurde mit dem ursprünglichen Punkt aus der Aufgabenstellung ersetzt.

1.11 Projektorganisation

1.11.1 Durchführung

Die IPA wird während zehn Arbeitstagen am Stück mit der Ausnahme von Schul- oder Feiertagen durchgeführt.

Startdatum: 09.05.2022

Enddatum: 24.05.2022

Ausführungsort: Confidis AG
Josefstrasse 218
8005 Zürich

Beteiligte

Vorname Name	E-Mail	Telefon	Funktion
André Lichtsteiner	andre.lichtsteiner@gmx.ch	+41 79 469 33 67	Hauptexperte
Christian Walder	christian.walder@benedict.ch	+41 44 298 17 83	Berufsbildner
Thiru Siva	thiru.siva@vinosol.ch	+41 58 255 27 17	Verantwortliche Fachkraft
Hansruedi Menzi	It-support@bewegdi.com	+41 79 242 08 37	Nebenexperte
Ruanin Khankah	ruanin_18@hotmail.com	+41 76 580 93 93	Kandidat

1.11.2 Arbeitsmethode

Für die Umsetzung der IPA wurde die Projektmethode IPERKA verwendet.

Dabei stehen jeder der Buchstaben für eine Projektphase:

I – Informieren
P – Planen
E – Entscheiden
R – Realisieren
K – Kontrollieren
A – Auswerten

Die Anwendung von IPERKA in einem Projektablauf, verlangt eine strukturierte Vorgehensweise. Ich habe mich für diese Arbeitsmethode entschieden, weil es für den Umfang dieses Projekt geeignet ist. Diese Methode hilft sehr bei der Planung sowie beim Verstehen. Hilfreich ist die Methode ebenfalls, um ein Projekt systematisch zu bearbeiten oder ein Problem gezielt zu lösen. Die Planung hat hierbei eine grosse Bedeutung. Erst wenn ein Konzept ausgearbeitet ist und die Entscheidung feststeht, sollte mit der Realisierung begonnen werden.

1.11.3 Ablage der Daten

Der Zeitplan und der IPA-Bericht werden im folgenden Ordner abgelegt:

C:\xampp4\htdocs\jobcircle_new\Dokumente

Der Code wir im folgenden Ordner abgelegt:
C:\xampp4\htdocs\jobcircle_new\Code

In diesem Ordner wird für jeden Tag der IPA ein Unterordner erstellt, in welchem am Ende des Tages der Stand der Dokumente abgelegt wird.

Das Format für die Unterordner: YYYYMMDD

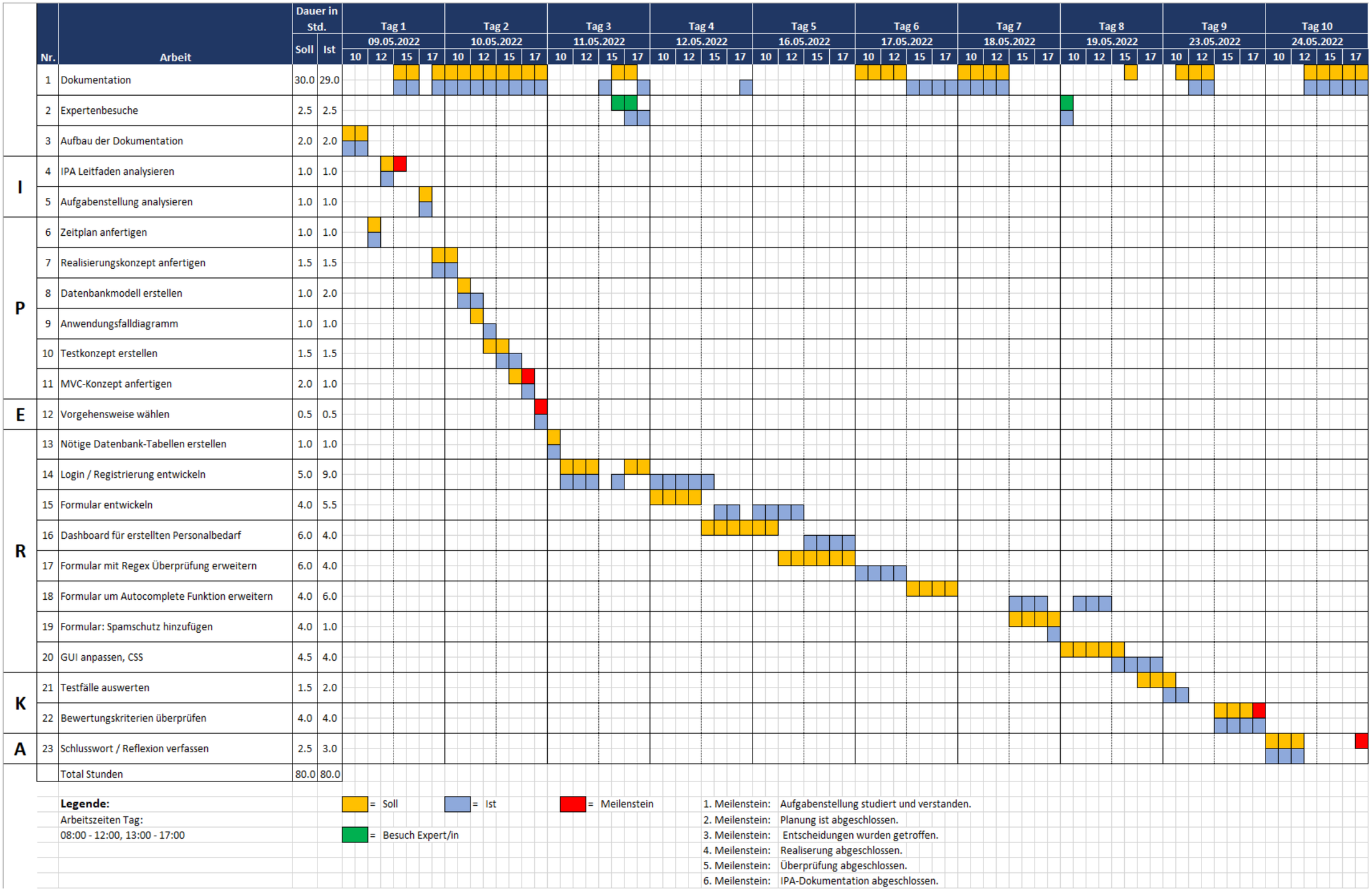
Also wenn ich einen Ordner am 10.05.2022 erstellen sollte, würde der Ordner „20220510“ heissen.

Die Implementierung der Oberfläche wird im zur Verfügung gestellten Git-Repository abgelegt und versioniert. Diese ist öffentlich und unter folgendem Link abrufbar:

https://github.com/ruanin/jobcircle_origin

1.12 Zeitplan

	Nr.	Arbeit	Dauer in Std. Soll	Ist
I	1	Dokumentation	30.0	29.0
	2	Expertenbesuche	2.5	2.5
	3	Aufbau der Dokumentation	2.0	2.0
	4	IPA Leitfaden analysieren	1.0	1.0
	5	Aufgabenstellung analysieren	1.0	1.0
P	6	Zeitplan anfertigen	1.0	1.0
	7	Realisierungskonzept anfertigen	1.5	1.5
	8	Datenbankmodell erstellen	1.0	2.0
	9	Anwendungsfalldiagramm	1.0	1.0
	10	Testkonzept erstellen	1.5	1.5
	11	MVC-Konzept anfertigen	2.0	1.0
E	12	Vorgehensweise wählen	0.5	0.5
R	13	Nötige Datenbank-Tabellen erstellen	1.0	1.0
	14	Login / Registrierung entwickeln	5.0	9.0
	15	Formular entwickeln	4.0	5.5
	16	Dashboard für erstellten Personalbedarf	6.0	4.0
	17	Formular mit Regex Überprüfung erweitern	6.0	4.0
	18	Formular um Autocomplete Funktion erweitern	4.0	6.0
	19	Formular: Spamschutz hinzufügen	4.0	1.0
	20	GUI anpassen, CSS	4.5	4.0
K	21	Testfälle auswerten	1.5	2.0
	22	Bewertungskriterien überprüfen	4.0	4.0
A	23	Schlusswort / Reflexion verfassen	2.5	3.0
		Total Stunden	80.0	80.0



Legende:
Arbeitszeiten Tag:
08:00 - 12:00, 13:00 - 17:00

= Soll = Ist = Meilenstein
= Besuch Expert/in

1. Meilenstein:	Aufgabenstellung studiert und verstanden.
2. Meilenstein:	Planung ist abgeschlossen.
3. Meilenstein:	Entscheidungen wurden getroffen.
4. Meilenstein:	Realiserung abgeschlossen.
5. Meilenstein:	Überprüfung abgeschlossen.
6. Meilenstein:	IPA-Dokumentation abgeschlossen.

Abbildung 1: Zeitplan

1.12.1 Meilensteine

09.05.2022 – Informieren

Aufgabenstellung durchlesen & verstehen

10.05.2022 – Planung

Bis zu diesem Meilenstein müssen in der Dokumentation die Phasen Informieren & Planen abgeschlossen sein. Dazu gehören der Zeitplan, das Anwendungsfalldiagramm, das Datenbankmodell und die ganzen Konzepte.

19.5.2022 – Entscheiden

Entscheidung wurde getroffen, Begründung mit Lösungsvariante wurde entsprechend dokumentiert.

19.05.2022 – Realisieren

Dieser Meilenstein beinhaltet die Fertigstellung des Validierung- und Analyse-Portals. Es kann mit dem Testen begonnen werden.

23.05.2022 – Kontrollieren

Testprotokoll durchgeführt, und Nachbesserungen sind erledigt. Kurzfassung ist verfasst. Als nächstes wird der IPA-Bericht fertiggestellt.

24.05.2022 – Auswerten

Abgabe des Projekts.

1.13 Arbeitsjournal

1.13.1 Tag 1 – 09.05.2022

Tagesablauf

Anfangs habe ich angefangen meine Dokumentation aufzubauen und noch meinen Soll/Ist-Zeitplan fertig gestellt. Zeitgleich habe ich den IPA-Leitfaden als auch meine Aufgabenstellung studiert. Da waren mir einige Sachen noch etwas unklar, weshalb ich diese bei meiner vertretenden Fachkraft nachgefragt habe. Nur reine Verständnissachen.

Die Fragen:

- Ein MVC-Konzept ist klar was das ist, aber wo sehe ich, was die „Jobcircle Framework“ für spezifische Vorschriften hat an die ich mich halten müsste (Geschäftslogik)?
- Liste Felder: „Erstellungsdatum, Firma, Kontakt, Beruf, Meldepflichtig Status“ für die Übersichtsseite mit dem eingereichten Personalbedarf gemeint, richtig? Steht kein Kommentar dazu.
- In der Aufgabenstellung steht, dass beim übermitteln des Formulars der Kunde ein Benutzerkonto anlegen KANN. Ist es in diesem Fall also nicht obligatorisch, dass er ein Konto anlegen muss?

Zum Schluss habe ich angefangen mein Realisierungskonzept zu schreiben

Ausgeführte Arbeiten

- Soll/Ist-Zeitplan erstellt
- Dokumentation aufgebaut.
- IPA-Leitfaden & Aufgabenstellung analysiert.
- Angefangen ein Realisierungskonzept zu schreiben

Ungeplante Arbeiten

Erreichte Ziele

- Der Zeitplan steht.
- Die Dokumentation ist strukturiert und wurde erweitert.

Aufgetretene Probleme

- Keine

Benötigte Hilfestellungen

- Keine

Reflexion

Heute war der erste IPA-Tag, ich war anfangs sehr nervös, aber als ich dann rein kam und die ersten Meilensteine erreichte, war das dann kein Problem mehr. Ich bin gut gestartet und konnte viel erledigen. Ich bin gespannt auf die nächsten Tage sowie auf die Treffen mit dem Hauptexperten.

1.13.2 Tag 2 – 10.05.2022

Tagesablauf
Heute habe ich die Planung sowie die Entscheidung von der IPERKA Methode abgeschlossen. Ich konnte das Realisierungskonzept abschliessen, ich habe ein Datenbankmodell erstellt samt Beziehungen, ein Anwendungsfalldiagramm, das Testkonzept definiert und ich habe mein Vorgehen mit dem MVC-Konzept niedergeschrieben.
Ausgeführte Arbeiten
- Realisierungskonzept abgeschlossen - Datenbankmodell erstellt - Anwendungsfalldiagramm erstellt - Testkonzept definiert (Black-Box-Test) - MVC-Konzept definiert - Vorgehensweise gewählt
Ungeplante Arbeiten
Erreichte Ziele
- Realisierungskonzept steht - Ich konnte die Datenbankbeziehungen erstellen - Ich habe das Anwendungsfalldiagramm erstellt - Das Testkonzept definiert - Eine Grafik für das MVC-Konzept erstellt. - Den Punkt Entscheidungen abgeschlossen.
Aufgetretene Probleme
- Keine
Benötigte Hilfestellungen
- Keine
Reflexion
Der zweite Tag der IPA war schon etwas anstrengender, da ich laut dem Zeitplan auch etwas mehr machen musste. Aber an sich ging es gut mit der Zeit und ich bin auch zufrieden gewesen heute. Ich konnte mir schon einen genauen Plan machen und weiss wie ich vorgehen muss.

1.13.3 Tag 3 – 11.05.2022

Tagesablauf
Heute habe ich mir vorgenommen, das Datenbankmodell umzusetzen und die Tabellen zu erstellen samt Beziehungen. Ebenfalls habe ich mir vorgenommen, mit dem Punkt Realisieren anzufangen, heisst gemäss dem Zeitplan mit anzufangen das Login und Registrierungsformular zu programmieren. Auch treffe ich heute den Experten für das erste Treffen. In diesem Treffen besprechen wir den Start, die Bewertungskriterien und die Vorgehensweise.
Ausgeführte Arbeiten
Die Tabellen für die Datenbank habe ich nach dem Datenbankmodell erstellt und die dazugehörigen Beziehungen. Ebenfalls habe ich das Registrierungsformular soweit programmiert, das es schon mal möglich ist sich zu registrieren. Über eine Verlinkung im regulären Anmeldeformular für den Kandidaten, wird man zum View RegisterNewClient weitergeleitet welche über den Controller Client Daten vermittelt bekommt. Von dort aus habe ich das Formular mit den nötigen Feldern entwickelt und es ist möglich sich per Mail oder Telefon zu registrieren.
Ungeplante Arbeiten
- Das erste Treffen wurde auf eine Stunde nachgeschoben,
Erreichte Ziele
- Das Registrieren als Kunde ist möglich. - Die Datenbanktabellen stehen samt Beziehungen.
Aufgetretene Probleme
- Ich musste für kleinere Syntaxfehler länger Reverse Engineering betreiben und das hat mich etwas Zeit gekostet.
Benötigte Hilfestellungen
- Ich habe meine VF die Frage gestellt, ob ich das Registrierungsformular am besten separat einbetten sollt in der Navigation oder ob ich das mit einem Input Radio bestätigen kann im vorhanden Formular für Kandidaten. Er verwies mich darauf, das es am besten ist eine Verlinkung im Registrierungsformular für Kandidaten einzuprogrammieren.
Reflexion
Am 3. Tag fing das Realisieren an. Ich hatte anfangs keine Schwierigkeiten. Im Prozess habe ich etwas Zeit verloren durch einige Syntaxerror. Der Experte kam um 16:30 und hat mich über einige Sachen aufgeklärt und wir haben zusammen mit meinem VF die individuellen Bewertungskriterien besprochen und etwas abgeändert. Ich zeigte dem Experten auch meine bisherige Doku und Zeitplan und er verwies mich auf Verbesserungsvorschläge. Im grundlegenden bin ich auch hier zufrieden und motiviert.

1.13.4 Tag 4 – 12.05.2022

Tagesablauf
Ich nehme mir heute vor, das Registrierungs- und Login Formular weiter auszufeilen. Ich möchte heute noch das Dashboard fertig stellen, danach nehme ich wie beim Zeitplan vorgegeben das Personalbedarfsformular in Angriff.
Ausgeführte Arbeiten
Heute habe ich das Registrierungsformular fertig gestellt. Es ist möglich sich mit allen nötigen Feldern zu registrieren. Man kann sich ebenfalls ganz gemütlich und übersichtlich ein und aus loggen, als Kandidat als auch als Kunde. Der registrierte Kunde hat ein Dashboard, in welchem er seine persönlichen Angaben wie Firmenbezeichnung, Name etc. ändern kann. Ebenfalls kann er vom Dashboard sein Passwort ändern. Unnötige Ansichten wie Spontanbewerbungen oder Stellenangebote wurde der Gruppe Kunde entzogen. Für das Personalbedarfsformular blieb nur wenig Zeit übrig am Ende vom Tag, ich habe mich zu sehr auf das Login fokussiert.
Ungeplante Arbeiten
- Einfügung der Funktion die persönlichen Daten zu ändern (Name, Passwort…)
Erreichte Ziele
- Das Login Formular funktioniert einwandfrei.
Aufgetretene Probleme
- Ich habe zu viel Zeit in das Login investiert und kam nur knapp dazu mit dem Personalbedarfsformular zu beginnen.
Benötigte Hilfestellungen
- Keine
Reflexion
Man könnte meinen ich wäre jetzt fast schon in der Halbzeit angelangt. Ich war heute sehr produktiv, ich habe das Registrierungsformular sprich das gesamte Login-System soweit es geht fertig entwickelt. Das Formular für den Personalbedarf konnte ich heute auch schon anfangen aber ich merke da, das es zu Asynchronität kommt mit dem Zeitplan.

1.13.5 Tag 5 – 16.05.2022

Tagesablauf
Ich nehme mir heute vor zuerst nochmals vor einen Überblick über den bisherigen Verlauf zu verschaffen. Passend zum neuen Wochenstart wäre es hilfreich festzustellen, was bisher abgeschlossen wurde und was noch zu erledigen ist. Ich bin zuversichtlich heute das Formular fertig zu stellen und für das übermittelte Formular die Übersicht an das Dashboard einzuführen.
Ausgeführte Arbeiten
- Formular für registrierte und nicht registrierte Kunden erstellt. - Angefangen das Dashboard für übermittelte Personalbedarfe zu entwickeln, beinahe fertig.
Ungeplante Arbeiten
- Keine
Erreichte Ziele
- Formular „Personalbedarf“ erstellt
Aufgetretene Probleme
- Ich musste mit Reverse Engineering einige Syntaxfehler beseitigen. Es gab Probleme bei der save query.
Benötigte Hilfestellungen
- Keine
Reflexion
Der heutige Tag ist eigentlich prima verlaufen. Ich kam am Personalbedarf Formular sowie an der Übersichtsseite essenziell voran. Das einzige Problem ist, dass ich am Zeitplan etwas hinterherschleife. Für den morgigen Tag ist geplant, meine bisherigen Ziele vom Punkt „Realisieren“ zu dokumentieren und am Formular die Überprüfungsfunktion mit Regex zu integrieren.

1.13.6 Tag 6 – 17.2022

Tagesablauf
Anfangs werde ich erstmal die Dokumentation auf den neusten Stand bringen und meinen bisherigen Fortschritt dokumentieren. Das mache ich den halben Tag. Danach werde ich schauen, dass die Formulare die ich erstellt habe mit der Regex Funktion erweitert werden. Im Zeitplan habe ich mir ursprünglich vorgenommen, schon das Formular mit der Autocomplete Funktion zu erweitern, aber ich werde das voraussichtlich verschieben müssen, da dafür keine Zeit bleibt nach meiner Einschätzung.
Ausgeführte Arbeiten
- Dokumentation bearbeitet. - Alle Formulare mit Regex erweitert. - Übersichtsseiten für beide Gruppen fertig gestellt.
Ungeplante Arbeiten
- Keine
Erreichte Ziele
- Mit der Dokumentation voran geschritten. - Regex Überprüfung implementiert. - Übersichtsseiten für übermitteltes Personalbedarfsformular implementiert.
Aufgetretene Probleme
- Zeitplan Verschiebung.
Benötigte Hilfestellungen
- https://www.php.net/manual/de/function.preg-match.php
Reflexion
Bisher kam ich gut mit dem Projekt an sich voran, heute war mir wichtig, dass ich meine Fortschritte auch gut dokumentieren kann. Das habe ich ganz gut hingekriegt. Beim Reflektieren auf die Arbeit, bin ich alles auch nochmal für mich durchgegangen und habe mir selbst auch die Frage gestellt: „Wie habe ich das genau gemacht?“. Ebenfalls habe ich da auch meine Kommentare im Quellcode bearbeiten können. Ich habe danach mir über das Internet Informationen beschafft, wie ich ein Regex-Muster zur Überprüfung der Feldeingaben entwickeln kann und konnte dieses Wissen gut auf das Projekt einsetzen. Felder wie E-Mail, Telefon, Name etc. werden absofort gekennzeichnet, sollte das Feld falsch oder nicht ausgefüllt sein.

1.13.7 Tag 7 – 18.05.2022

Tagesablauf
Heute nehme ich mir vor die Autocomplete Funktion in Angriff zu nehmen. Über eine Tabelle in der Datenbank welche alle Postleitzahlen samt Ortsnamen der Schweiz enthält, sollte es möglich sein, bei der Angabe der Postleitzahl das Feld Ort mit dem dazugehörigen Namen zu füllen. Zum anderen werde ich versuchen, sollte es nicht zu knapp kommen die Spammschutzfunktion integrieren. Dafür benutze ich dann die Google API reCAPTCHA.
Ausgeführte Arbeiten
- Dokumentation voran geschritten - Autocomplete angefangen - Spammschutz implementiert - Sendmail-Funktion abgeschlossen - Mehrere Verbesserungen an den Formularen durchgeführt wie z.B Definierung der Log Files vorgenommen, gegen SQL und Script Injections Regular Expression definiert und eine Datenschutzerklärung dem Formular beigelegt.
Ungeplante Arbeiten
-Sendmail-Funktion, ich habe über den SMTP der Firma die Sendmail Funktion integriert.
Erreichte Ziele
- Dokumentation voran gebracht. - Spammschutz gegen Bots und andere Malware Softwares - Sendmail-Funktion implementiert.
Aufgetretene Probleme
- Ich bin stecken geblieben bei der Autocomplete Funktion. Ich habe mich im Internet schlau gemacht wie ich am besten vorgehe, ich habe zuerst einen Autocomplete Controller erstellt, über eine fetch function die Query definiert und dann mit einer Ajax Library ein Script geschrieben. Ich schätze der Knackpunkt liegt an der Library Version.
Benötigte Hilfestellungen
- Keine
Reflexion
Heute war ein etwas strenger Tag. Ich konnte alles erledigen bis auf die Autocomplete Funktion. Im Grunde kann das aber nicht allzu schwer sein. Immerhin konnte ich das heutige Ziel mit dem Spammschutz erreichen. Das ging einwandfrei und war nicht schwer. Ich musste dafür nur eine Recaptcha Abfrage im Controller definieren und eine function mit JS definieren. Zudem habe ich mit Regex das Pattern für die Felder so bearbeitet, dass eine Injection nicht mehr möglich ist. Sprich Zeichen, womit eine Manipulation innerhalb der Abfrage möglich sei blockiert.

1.13.8 Tag 8 – 19.05.2022

Tagesablauf
Geplant ist für Heute das Gespräch mit Herr Lichsteiner, das Ausarbeiten der Autocomplete Funktion und die Webseite mit den hinzugewonnen Erweiterungen benutzerfreundlicher zu machen. Auf dem Zeitplan war noch geplant mit den Testkonzepten zu beginnen, aber das werde ich dann verschieben, da ich dazu nicht mehr kommen kann.
Ausgeführte Arbeiten
- Gespräch geführt - Webseite benutzerfreundlicher gestaltet - Einige Buggs behoben (Fehlermeldungen besser definiert)
Ungeplante Arbeiten
- Die Autocomplete Funktion hat mir enorm Zeit gekostet
Erreichte Ziele
-Webseite ist benutzerfreundlich gestaltet. -Fehlermeldungen für Spamschutz definiert,
Aufgetretene Probleme
- Autocomplete nicht abgeschlossen.
Benötigte Hilfestellungen
- Keine
Reflexion
Heute hatte ich das zweite Gespräch mit Herr Lichtsteiner. Ich war aus irgendeinem Grund sehr nervös im Verlauf vom Gespräch. Ich zeigte ihn meinen bisherigen Fortschirtt und er gab mir auch hier Verbesserungsvorschläge und klärte mich über das dritte Treffen, also das Fachgespräch auf. Nach dem Gespräch habe ich mich an die Autocomplete Funktion dran gesetzt. Ich habe mit Ajax eine Funktion geschrieben gehabt in einem Testbereich der Seite welche Tastenschläge sofort erkennen soll. Es scheiterte aber leider daran, dass die von mir angegebene URL nicht erkannt wurde und ich somit keine Abfrage angeben konnte. Ich habe im Internet recherchiert gehabt, die Versionen geprüft aber ich konnte das Problem nicht ausfindig machen. Die restliche Zeit habe ich die GUI anpassen gesetzt. Die Seite war allgemein an einigen Stellen nicht responsiv oder es waren gar Bilder ganz abseits verschoben etc. Diese habe ich mit Hilfe von Bootstrap behoben gehabt.

1.13.9 Tag 9 – 23.05.2022

Tagesablauf
Geplant für Heute ist das Auswerten der Testfälle und dann die Bewertungskriterien noch einmal durchgehen für die Dokumentation. Die Testfälle werde ich dann dokumentieren und das Dokument so bearbeiten, dass sie der Bewertungskriterien ausreichend entsprechen.
Ausgeführte Arbeiten
- Testfälle durchgeführt - Dokumentation erweitert um „Kontrollieren“. - Dokumentation angepasst.
Ungeplante Arbeiten
- Keine
Erreichte Ziele
- Testfälle beendet
Aufgetretene Probleme
- Testfall Autocomplete hat nicht geklappt.
Benötigte Hilfestellungen
- Keine
Reflexion
Es naht der Abschluss der IPA und ich habe mittlerweile ein gutes Gefühl. Bei der Auswertung der Testfälle ging eine Sache nicht, und das ist die Funktion Autocomplete. Ich brauchte die letzten Tage dafür zu lange und musste dann einsehen, dass ich die Webseite nicht um diese Funktion erweitern konnte. Danach habe ich mich ans dokumentieren gesetzt und habe es nochmals überarbeitet. Das Dokument wurde ebenfalls um den Punkt „Kontrollieren“ erweitert. Die Bewertungskriterien habe ich dann nochmals mit meinem Dokument abgeglichen und ich habe für morgen vor, noch ein paar Änderungen durchzuführen.

1.13.10 Tag 10 -24.05.2022

Tagesablauf
Heute ist der letzte Tag der IPA. Ich habe vor die restliche Zeit weiter am Bericht zu arbeiten, darunter einige Formulierungen zu ändern, ein Schlusswort / Reflexion verfassen und ich werde den Anhang mit dem Quellcode vorbereiten. Wenn dann noch genug Zeit übrig bleibt noch einmal durch die Kriterien gehen und vergleichen ob der Bericht alles soweit erfüllt.
Ausgeführte Arbeiten
- Dokument überarbeitet
Ungeplante Arbeiten
- Keine
Erreichte Ziele
- IPA-Bericht Abgabebereit gemacht.
Aufgetretene Probleme
- Keine
Benötigte Hilfestellungen
- Keine
Reflexion
ch bin gespannt und zuversichtlich auf die Ergebnisse meiner Projektarbeit. Meine Zeiteinteilung für die Dokumentation hat sich gut bewährt. In der letzten Stunde vor der Abgabe habe ich ein sicheres Gefühl. Ich freue mich auf die Präsentation und Demonstration.

2 IPA Bericht Teil 2

2.1 Kurzfassung

Diese Kurzfassung richtet sich an die fachlich kompetenten Leser mit Fachwissen in der Informatik. Sie soll den Einstieg für das Verständnis dieser Arbeit erleichtern und einen ersten Überblick liefern.

2.1.1 Ausgangslage

Bei der Planova erhält man die Stellenangebote aktuell via E-Mail oder auch telefonisch. Viele Mail-Anfragen sind nicht vollständig. Das bedeutet für den Personalberater einen zusätzlichen Arbeitsaufwand, da er telefonisch den Kunden nach fehlenden Informationen erbitten muss.

Zur Entlastung der Personalberater wäre es hilfreich, wenn die Kunden die offenen Stellen online über die Website übermitteln können.

2.1.2 Umsetzung

Zu Beginn des Projekts wurde ein Zeitplan passend zu den Phasen der IPERKA Projektmethode erstellt.

Das Projekt wird in drei Teile gegliedert. In das Modell (Datenlogik und Anbindung der Datenbank), der View (Ergebnispräsentation & UI) und der Controller(Vermittlung & Steuerung).

Das gewünschte Formular wird mit der Entwicklungsumgebung NetBeans DIE 12.0 entwickelt. Als Framework nutze ich dafür Bootstrap und Laravel.

2.1.3 Ergebnis

Die Mandantenfähige Webseite konnte erfolgreich umgesetzt werden, die Tests verliefen grösstenteils bis auf eine Ausnahme positiv.

Die Funktionalität stimmt soweit und die erweiterten Funktionen wie Formular, Dashboard etc. werden deutlich dargestellt.

2.2 Informieren

Beim erste Teil der IPERKA-Methode geht es darum sich zu informieren. In dieser Phase wird der Auftrag genauestens analysiert und die Erkenntnisse ausgewertet. Die wesentlichen Punkte werden so früh wie möglich erkannt und allfällige Fragen geklärt.

2.2.1 Anfängliche Unklarheiten

Zu klärende Fragen gehörten hierbei:

- Ein MVC-Konzept ist klar was das ist, aber wo sehe ich, was die „Jobcircle Framework“ für spezifische Vorschriften hat an die ich mich halten müsste (Geschäftslogik)?

- Liste Felder: „Erstellungsdatum, Firma, Kontakt, Beruf, Meldepflichtig Status“ für die Übersichtsseite mit dem eingereichten Personalbedarf gemeint, richtig? Steht kein Kommentar dazu.

- In der Aufgabenstellung steht, dass beim übermitteln des Formulars der Kunde ein Benutzerkonto anlegen KANN. Ist es in diesem Fall also nicht obligatorisch, dass er ein Konto anlegen muss?

Diese Fragen wurden meiner vertretenden Fachkraft per E-Mail übermittelt, ich warte derzeit noch auf eine Antwort. Ich konnte sonst soweit den Auftrag, den Leitfaden als auch die Kriterien gut verstehen.

2.3 Planen

2.3.1 Realisierungskonzept

Ich verwende die Developer Umgebung von planova.ch benutzt. Als DIE benutze ich NetBeans 12.0. Ich programmiere PHP und benutze Elemente aus Laravel und Ajax sowie JavaScript. Als Datenbank brauche ich MySQL.

1. **Login/Registrierungs-Funktion**

   Zunächst sollte es möglich sein sich als Kunde, also als Arbeitsgeber zu registrieren. Da man sich bereits schon als Kandidat, also Arbeitssuchenden registrieren und einloggen kann, muss ich darauf achten, bei jeder Login Abfrage klar definiert zu haben welches Model abgerufen werden soll.

   Bei der Abfrage registrieren kann man sich dann ganz klar dafür entscheiden, ob man sich mit einem Kundenkonto registrieren will, oder mit einem Kandidaten Konto. Für beide Gruppen gibt es ein eigenes Formular. Die Login Abfrage wird dann so ablaufen, dass der Controller Client das Model „**cust_user**“, die Login Daten mit der Datenbank abgleichen lässt. Sollten diese dann eine Übereinstimmung finden, mit dem noch zu erstellenden Model „**cust_user**“, erfolgt das Login.

   Bei der Registrierung wird darauf geachtet, dass keine Mutationen entstehen und ein Mail jeweils für ein Kundenkonto verwendet werden kann. Gleichzeitig sollte man aber mit einer Mail jeweils ein Kunden und Kandidatenkonto eröffnen können. Das wird dann mit Sessions gelöst.

   Die Felder für die Registrierung sind: Vorname, Name, E-Mail, Telefon, Firma, Funktion, Strasse/Nr., PLZ, Ort und Land. Nach der Registrierung wird es im Dashboard einen Bereich geben in welchem man diese Daten updaten kann sollte man es müssen. Für E-Mail und Telefon bezieht sich das hierbei nur auf die Kontaktangabe des Users und nicht auf die Login Methode. Heisst, wenn ein User via Mail eingeloggt ist und unter dem Bereich im Dashboard seine Angaben zum Mail ändert, muss er sich immer noch mit der gleichen Adresse einloggen wie zuvor.

   Die gewünschten Daten werden auch hier geprüft und in per asynchroner Datenübertragung dem User gemeldet, sollten falsche Zeichen verwendet worden sein bevor das Formular abgeschickt wird. Bei der Übermittlung werden per Regex die eingegebenen Felder nochmals überprüft. Die Überprüfung er asynchrone Datenübertragung funktioniert Clientseitig.

2. **Formular zum Personalbedarf**

   Unabhängig davon, ob man jetzt registriert, eingeloggt oder ausgeloggt ist, sollte einem die Ansicht zum Formular gewährt werden. Wenn man die Formular-Übermittlung bestätigt, wird je nach Case abgefragt ob man sich einloggen, sich registrieren, oder sofort zur Übersichtseite gelangen will ohne sich einzuloggen.

   Wenn man sich dazu entscheidet, das Formular zu bestätigen ohne sich einzuloggen oder zu registrieren, wird das Formular keinem Benutzer zugewiesen als Referenz.

   Sollte man bereits eingeloggt sein, wird das gewünschte Rekord auf der Übersichtsseite dem Kundenkonto als Referenz zugewiesen, so kann er auf der Übersichtsseite einsehen, wie viele Personalbedarfe er wann abgeschickt hat. Hat man das Formular als Gast abgeschickt, erhält man eine einmalige Übersicht zu seinem Personalbedarf.

3. **Übersichtsseite**

   Wie beim zweiten Punkt erwähnt, soll der Kunde zu einer Übersichtsseite gelangen sobald dieser das Formular übermittelt hat. Dort sollten ihm all seine offenen Personalbedarfe angezeigt werden. Diese Records werden mit den gewünschten Feldern dargestellt (Erstellungsdatum, Firma, Kontakt, Beruf, Meldepflichtig Status).

   Als Gast ist diese Seite nur einmalig aufrufbar, eingeloggt wird sie im Profil gespeichert.

4. **Übermittlung an E-Mail Adresse**

   Zeitgleich bei der Übermittlung des Personalbedarfs werden die Daten des übermittelten Formulars an die Zweigstelle versandt. Dafür, verwerde ich einen SMTP-Server welcher von der Firma gegeben ist.

   Die Zweigstelle erhält eine Übersicht zum übermittelten Personalbedarf des Kunden.

5. **Felder Überprüfung mit Regex**

   Die erstellten Formulare, sei es jetzt der Personalbedarf oder die Registrierung, sollen um eine Funktion erweitert werden, in welcher die Felder geprüft werden nach falschen Zeichen. Sollte man zum Beispiel im Feld «Name» ein Add-Zeichen eingeben sollte direkt eine Nachricht aufpoppen von wegen man hätte ein unerlaubtes Zeichen verwendet. Das soll auch ohne irgendeine Bestätigung passieren, sprich bevor man das Formular einreicht.

6. **Wiedersteht beliebten Angriffsmethoden wie JavaScript- oder SQL-Injection**

   Dieser Punkt hängt mit dem Punkt 6 halbwegs zusammen. Ich muss dafür sorgen, dass der Benutzer keine unnötigen Sonderzeichen verwenden um irgendwie eine SQL Query auszuführen durch ein Feld im Formular. Vor allem hier wichtig das eine Clientseitige als auch Serverseitige Überprüfung stattfindet.

7. **Autocomplete**

   Bei der Angabe der Postleitzahl sollte das Feld Ort automatisch gefüllt werden mit der dazu passenden eingegeben Postleitzahl. Dies erfolgt beim Registrierungs- als auch beim Übermittlungsformular vom Kunden.

2.3.2 Datenbankmodell

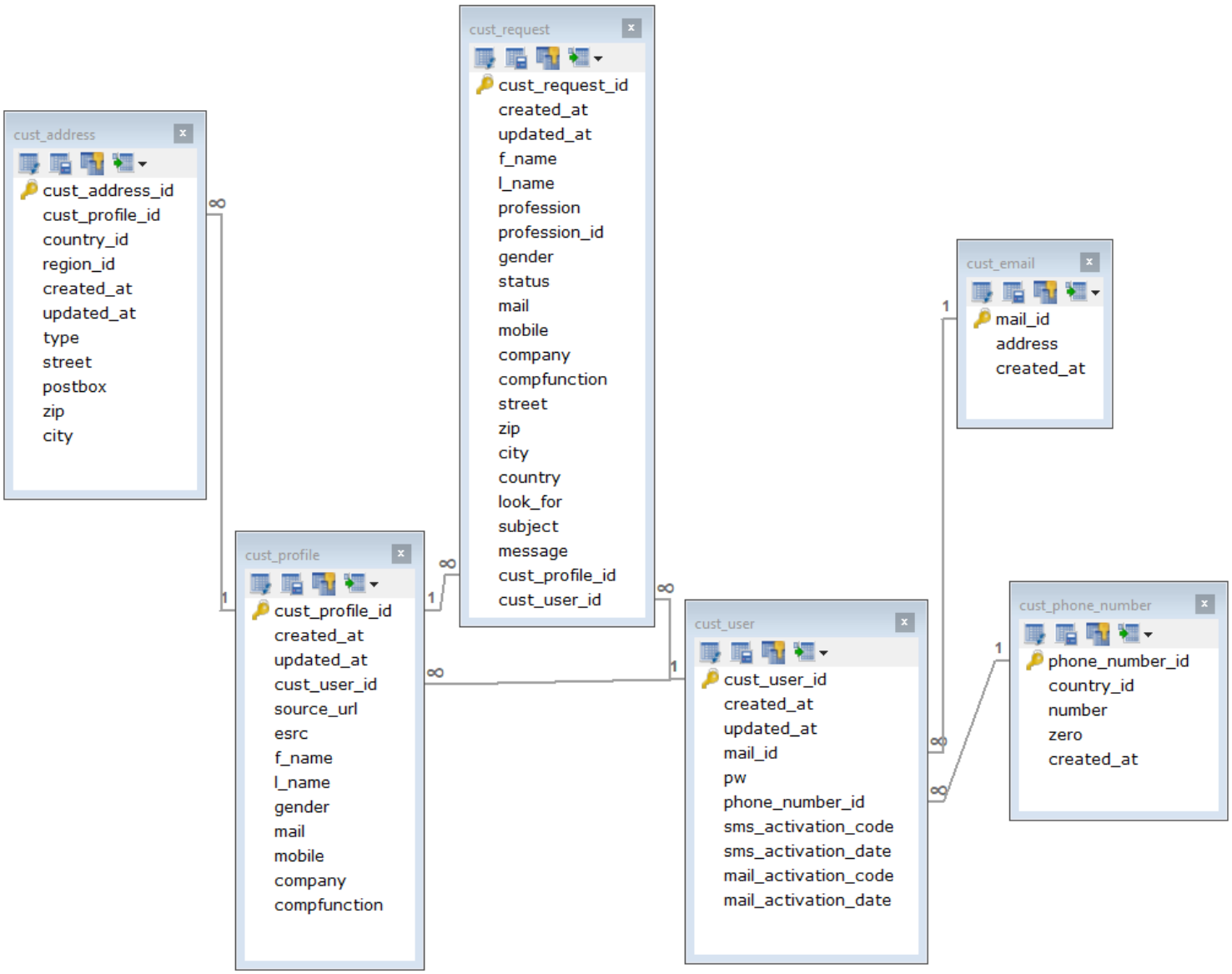


Abbildung 2: Datenbankmodell

2.3.3 Anwendungsfalldiagramm

Um nochmals den Anwendungsfall klarzustellen, ist ein Use Case bzw. Anwendungsfalldiagramm eine sehr hilfreiche Methode. Sie gibt einen Überblick über das Verhalten eines Systems. Es fast alle Interaktionen von Anwendern, die auftreten können, wenn ein Anwender ein fachliches Ziel mithilfe eines geplanten Systems erreichen will. (microTOOL, 2022)

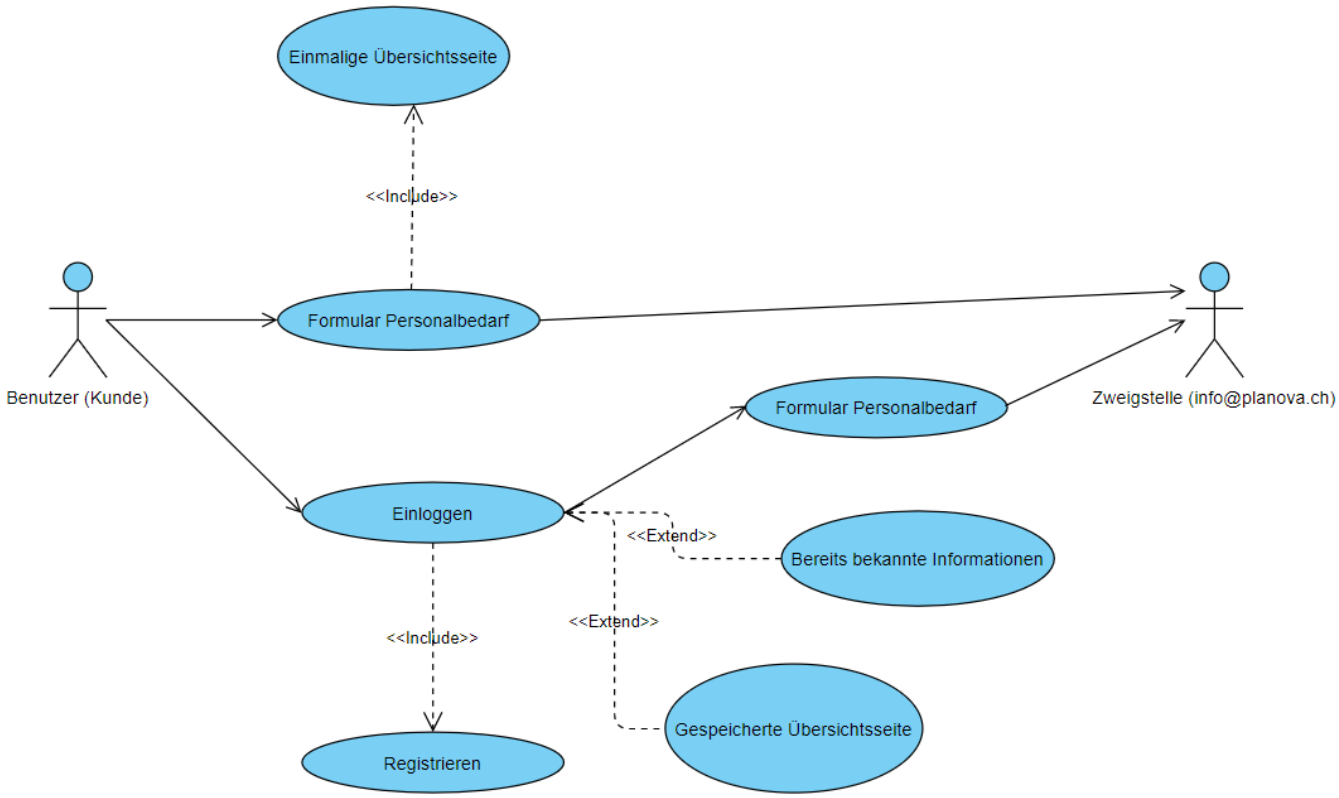


Abbildung 3: Anwendungsfalldiagramm

2.3.4 Testkonzept

Es wird die Black Box Testmethode eingesetzt. Dies ist eine Testmethode welche die Funktionalität einer Anwendung untersucht, ohne in ihre internen Strukturen oder Funktionsweisen einzudringen, also ohne sich den Code anzuschauen. So können wir einen klaren Einblick in die Userexperience schaffen und uns in den Kunden hineinversetzen.

2.3.5 Model-View-Controller-Konzept

Das MVC-Modell ist ein Architektur Muster in Webseiten und Software.

Im Model liegt die Datenlogik und die Anbindung an die Datenbank.

Der View kümmert sich um die Ergebnispräsentation, also um das User Interface. Die hierfür benötigten Daten kommen aus dem Model.

Der Controller wiederrum ist für die Vermittlung & Steuerung zuständig, Model und View kommunizieren nicht direkt miteinander sondern lassen sich vom Controller Anweisungen geben. Gleichzeitig kümmert der Controller sich um die Anfragen vom Benutzer. Auch was bei validen oder invaliden Ergebnissen zu tun ist entscheidet der Controller.

Eine Veranschaulichung zum MVC-Konzept:

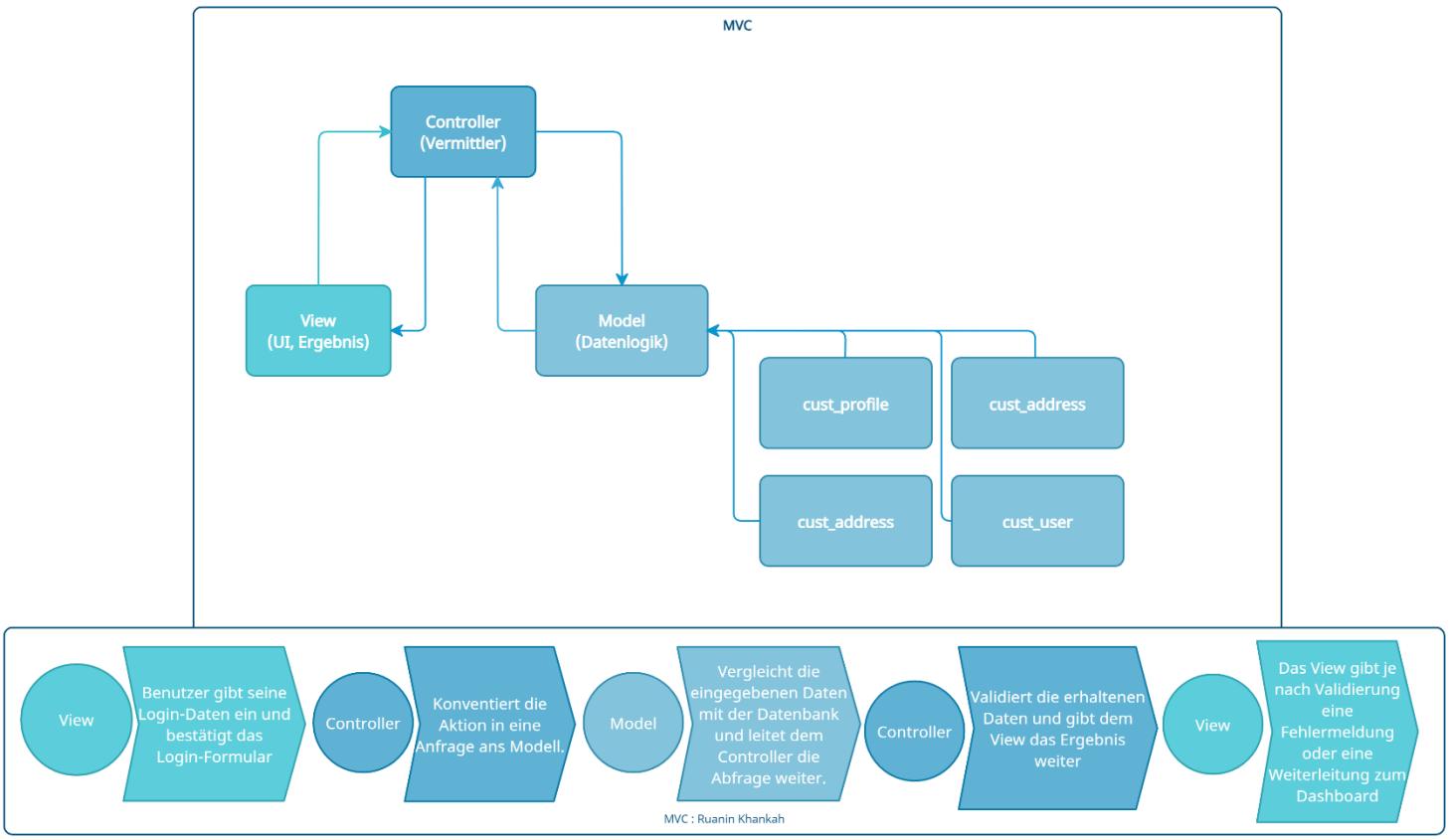


Abbildung 4: MVC-Konzept

2.4 Entscheiden

2.4.1 Login System

Bei dieser Entscheidung geht es darum festzulegen, ob Kandidaten und Kunden die gleiche Ansicht haben sollen und es nur einen kleinen festgelegten Bereich für den Personalbedarf der Kunden geben soll oder beide Gruppen jeweils verschiedene Ansichten haben. Das heisst die Kunden können ihren Lebenslauf ergänzen, während Kunden das nicht können, da sie es auch nicht brauchen.

Hier war mir dann schnell bewusst, dass ich ein eigenständiges Login System für Kunden machen will, wo am Anfang auch ganz klar zu definieren ist, ob man ein Kundenkonto erstellen will oder ein Kandidatenkonto. Das führt auch zu zwei Formularen, einmal für Kandidaten und einemal für Kunden. Dabei führen die Tabellen '**cust_email**' und '**cust_phone_number**' für die Gruppe Kunden jeweils die Abfragen aus um Mutationen zu vermeiden. Bei der Login Abfrage wird geprüft ob das eingegebene Mail und Passwort mit einem Kundenkonto übereinstimmt.

Beim erstmaligen aufrufen der Login Seite, landet man zunächst auf das Formular um sich als Kandidat anzumelden. Der Grund hierfür ist, dass die Nachfrage an Kandidatenkonten anfangs sowie auch in Zukunft höher ist und sein wird als die von Kunden. Der Kunde kann über eine Verlinkung zu seinem Formular springen.

2.4.2 Anzeige der Auswertung

Sobald man das Personalbedarfs-Formular eingeloggt übermittelt hat wird man auf die Übersichtsseite weitergeleitet in welchem man seinen eben gesendeten Personalbedarf dargestellt bekommt samt Nachricht, dass man in 48 Stunden kontaktiert wird.

Da man das Formular als Gast und eingeloggt übermitteln soll, habe ich mich dazu entschlossen, bei registrierten Benutzer den übermittelten Personalbedarf dem Konto zuzuweisen. Das heisst, er kann stetig auf sein Profil und seinen übermittelten Personalbedarf nachverfolgen.

Bei Benutzer die ausgeloggt sind und kein Konto besitzen, wird die Übersichtsseite mit dem erstellten Record einmalig angezeigt. Der Personalbedarf wird wie gewohnt in der Datenbank erfasst nur keinem Benutzer als Referenz zugeteilt.

2.5 Realisieren

2.5.1 Datenbank-Tabellen

Nach dem Datenbankmodell habe ich folgende Tabellen erstellt. „**cust**“ steht hierbei für Customer:

‘**cust_user**’
In dieser Tabelle werden die Login relevanten Informationen gespeichert für den Benutzer.

‘**cust_profile**’
In dieser Tabelle werden alle Kontakt- samt persönlichen Angaben gespeichert. Die Adresse wird als Referenz hier angegeben

‘**cust_email**’
Speichert die Login Methode E-Mail in dieser Tabelle.

‘**cust_phone_number**’
Speichert die Login Methode Mobile in dieser Tabelle.

‘**cust_address**‘
Speichert die Adressen der Kunden seperat.

‘**cust_request** ‘
Speichert die Daten für das Formular Personalbedarf.

2.5.2 Registrierungssystem

Für das Registrierungssystem wurde zuerst die Methode **registerWizard()** erstellt unter dem bereits verfügbaren Controller „**Client**“. Befindet sich der Benutzer auf dem Registrierungsformular für Kunden, also “**…Client/registerWizard()“** werden die Daten an das View von der Methode übermittelt.

Sollte der Kunde dort die Felder zur Registrierung ausfüllen und übermitteln, so wird vom Controller erstmal ein Logverzeichnis und Variablen erstellt für die ganzen Post Variablen aus dem Formular. Ebenfalls wird die Variabel **$time** für einen Aktivierungscode gespeichert. Die Variabel wird mit der Funktion **time()** initialisiert.

```
$logger = new Katzgrau\KLogger\Logger(APPLICATION.'/database/log/registerWizardClient');
$success = '';
$error = array();
$inputTask = Toolkit::getPost('task', '');
$secretInput = Toolkit::getPost('secretInput', '');
$inputLoginMobile = Toolkit::getPost('login_telefon', '');
$inputLoginMail = Toolkit::getPost('login_email', '');
$inputSalutation = Toolkit::getPost('salutation', '');
$inputFirstname = Toolkit::getPost('firstname', '');
$inputLastname = Toolkit::getPost('lastname', '');
$inputCompany = Toolkit::getPost('company', '');
$inputCompfunction = Toolkit::getPost('compfunction', '');

$inputZip = Toolkit::getPost('zip', '');
$inputStreet = Toolkit::getPost('street', '');
$inputCity = Toolkit::getPost('city', '');
$inputCountry = Toolkit::getPost('country', '1');
$inputMobile = Toolkit::getPost('mobile', '');
$inputEmail = Toolkit::getPost('email', '');

$inputLoginPassword = Toolkit::getPost('login_password', '');
$inputLoginPasswordConfirm = Toolkit::getPost('login_password_confirm', '');

$time = time();     //gets the current timestamp for the md5 mail activation code
```

Die Daten werden dann über die Klassenfunktion **checkClientFields(array()** aus der Klasse **ClassToolkit** nochmals auf unerlaubte Zeichen geprüft. Sollte eine Diskrepanz hier auffallen, wird ein Error zurückgegeben, in welchem man angegeben bekommt welches Feld nicht leer sein darf bzw. welches Feld unerlaubte Zeichen hat.

```
$error = Toolkit::checkClientFields(array('inputFirstname' => $inputFirstname, 'inputLastname' => $inputLastname,
    'inputCompany' => $inputCompany, 'inputCompfunction' => $inputCompfunction, 'inputMobile' => $inputMobile,
    'inputLoginMobile' => $inputLoginMobile, 'inputEmail' => $inputEmail, 'inputLoginMail' => $inputLoginMail,
    'inputCountry' => $inputCountry));
```

Wenn soweit alles stimmt, wird geprüft, ob die zwei Felder für die Passwort Eingabe gefüllt sind. In derselben Bedingung wird geprüft, ob der reCAPTCHA secret key gefüllt ist. Wenn nicht wird das Error Array gefüllt mit der Information, Verdacht auf Spam zu haben.

```
if((!empty($inputLoginPassword) && !empty($inputLoginPasswordConfirm)) && (!empty($secretInput) && $secretInput == SECRET_KEY)){
    if($inputLoginPassword == $inputLoginPasswordConfirm){
    }else{
        $error = "Die Passwörter stimmen nicht überein.";
    }
```

```
}else{
    $error[] = 'Daten konnten nicht gespeichert werden.';
    $logger->error('Das Profil konnte nicht gespeichert werden. Verdacht auf SPAM.');
    $logger->error('SPAM IP Adresse: '.$_SERVER['REMOTE_ADDR']);
}
```

Je nachdem, ob man sich jetzt via Mobile oder via Mail registriert, wird erstmal ein Objekt zum Model „**cust_user**“ erstellt. Das eingegebene Mail oder die eingegebene Nummer wird nun validiert. Sollte die eingegebene Login Methode bereits in der Datenbank gespeichert sein, wird ein Error zurückgegeben, welche die Information erhält, dass bereits ein Konto mit derartiger Login Methode existiert.

```
if((((!empty($inputLoginMail) && filter_var($inputLoginMail, FILTER_VALIDATE_EMAIL)) || (isset($inputLoginMobile)))){
    $objUser = $this->model('cust_user');
    $mailCount = $phoneCount = 0;
    if(!empty($inputLoginMail)){
        $mailCount = $objUser->where('address',$inputLoginMail)
                ->join('cust_email', 'cust_user.mail_id', '=', 'cust_email.mail_id')
                ->count();

    }

    if(!empty($inputLoginMobile)){

        require_once '../app/class/helper_phone_lib.php';
        $libPhoneObj = new helper_phone_lib();
        $arrIsPhoneNumberValid = $libPhoneObj->isPhoneNumberValidFromCountryId($inputLoginMobile, $inputCountry);

        if($arrIsPhoneNumberValid['isValidFromCountry'] != 1){
                $error[] = 'Die Telefonnummer ist nicht gültig für die ausgewählte Vorwahl: '.$inputLoginMobile;
        }else{

            $phoneCount = $objUser->where('number',$arrIsPhoneNumberValid['nationalNumber'])
                    ->where('country_id',$arrIsPhoneNumberValid['countryId'])
                    ->join('cust_phone_number', 'cust_user.phone_number_id', '=', 'cust_phone_number.phone_number_id')
                    ->count();
        }
    }
```

```
if($phoneCount > 0){
    $error[] = "Es existiert bereits ein Konto mit dieser Mobile-Nummber";
}else if($mailCount > 0){
    $error[] = "Es existiert bereits ein Konto mit dieser E-Mail";
}else{
```

Jetzt wird ein activation_code festgelegt. Je nachdem ob man die Mail Methode oder die Mobile Methode wählt, fällt dieser anders aus. Für das Mail wird mit der $time Variabel und einer md5 Methode ein Code festgelegt. Der Code für das SMS wird mit einer zufällig generierten Zahl von 100000 – 999999 festgelegt und dem Benutzer auf sein Handy per SMS zugeschickt. Einen richtigen Nutzen haben diese Aktivierungs-Codes allerding noch nicht. Da die Gruppe Kandidat diese Werte in ihrer Tabellen speichern, sollte es die Gruppe Kunden auch tun, für einen vorzeitigen Entscheid, das Login System so zu erweitern, dass man seine Mail oder Nummer bestätigen kann, ist man somit gewappnet. Wird bei der Eingabe die Mobile Nummer als Login Methode verwendet, wird die Nummer formatiert mit einem Plus und die passende Vorwahl. Dies funktioniert über die Utility für internationale Telefonnummer.

```
}else{

    if(isset($inputLoginMail) && !empty($inputLoginMail)){

        $objUser->mail_id = $inputLoginMail;
        $objUser->mail_activation_code = md5($time.$inputLoginMail);
    }else{

        $cleanNumber = str_replace('+','',$arrIsPhoneNumberValid['e164Format']);

        $objUser->phone_number_id = $cleanNumber;

        if(isset($_SESSION['sms_code']) && !empty($_SESSION['sms_code'])){
            $objUser->sms_activation_code = $_SESSION['sms_code'];

        }
    }
```

Nachdem das erledigt wurde und die nötigen Felder für das Objekt von ‚**cust_user**' gefüllt worden, werden vom Model aus mit einer save Funktion gespeichert.
Sollte die Query nicht erfolgen, wird sie aufgefangen und wird im Error Log verzeichnet.

```
try{

    $status = $objUser->save();

} catch (Exception $ex) {

    $logger->error('Beim User speichern ist ein Fehler aufgetreten.');
    $logger->debug('User-Save Exception:', $ex->getMessage());
    $logger->debug('User-Save Error:', $status);
}
```

```
if($status != true || is_array($status)){
    $error[] = 'Bitte versuchen sie es später noch einmal.';
    $logger->error('Beim User speichern ist ein Fehler aufgetreten.');
    $logger->debug('User-Save Error: ',$status);
}else{
```

```
}else{
    if(isset($inputMobile) || isset($inputLoginMobile)){

        if(isset($inputMobile) && !empty($inputMobile)){

            $arrPhoneNumber = $phoneUtil->parse($inputMobile, $arrCountry->code);

            $isValid = $phoneUtil->isValidNumber($arrPhoneNumber);

            if($isValid){
               $inputMobile = $phoneUtil->format($arrPhoneNumber, \libphonenumber\PhoneNumberFormat::E164);
            }
        }else{
            if(isset($inputLoginMobile) && !empty($inputLoginMobile)){
                $arrPhoneNumber = $phoneUtil->parse($inputLoginMobile, $arrCountry->code);
                $isValid = $phoneUtil->isValidNumber($arrPhoneNumber);

                if($isValid){
                   $inputMobile = $phoneUtil->format($arrPhoneNumber, \libphonenumber\PhoneNumberFormat::E164);
                }
            }
        }
    }
```

Es wurde ein Objekt über das '**cust_profile**' Model erstellt und für die Felder die input Variablen eingesetzt und anschliessen gespeichert. Auch hier wird beim Fehlschlag ein Logverzeichnis

```
$_SESSION['cust_user_id'] = $objUser->cust_user_id;
$_SESSION['logged'] = 'true';
$objCustProfile = $this->model('cust_profile');
$objCustProfile->cust_user_id = $_SESSION['cust_user_id'];
$objCustProfile->source_url =  $_SESSION['org_referer'];

$objCustProfile->esrc =  $_SESSION['esrc'];
$objCustProfile->gender = $inputSalutation;
$objCustProfile->f_name = Toolkit::cleanName($inputFirstname);
$objCustProfile->l_name = Toolkit::cleanName($inputLastname);
$objCustProfile->company = $inputCompany;
$objCustProfile->compfunction =  $inputCompfunction;
$objCustProfile->mail = $inputEmail;
$objCustProfile->mobile = $inputMobile;
try {
    $status = $objCustProfile->save();
} catch (Exception $ex) {
    $logger->error('Beim speichern des User-Profils ist ein Fehler aufgetreten.');
    $logger->debug('User-Profil Exception: ',$ex->getMessage());
    $logger->debug('Profil-Save: ',$status);
}
```

Die Adresse wird über das ‚**cust_address**‘ Model gespeichert. Der Grund warum die Adresse abseits der ‚**cust_profile**‘ Tabelle gespeichert wird, ist wenn das Dashboard des Benutzers erweitert werden sollte insofern das man mehrere Adressen angeben könnte, man so mehrere Adressen an der jeweiligen ‚**cust_profile_id**‘ zuweisen könnte. Das kann bei Kunden noch die Möglichkeit geben mehrere Sitze anzugeben und somit das Arbeitsgebiet für die Anfrage auch besser eingrenzen zu können. Vorerst soll es jedoch nur möglich sein, eine Adresse anzugeben und diese gegebenenfalls zu ändern.

```php
if($status == true){
    $_SESSION['cust_profile_id'] = $objCustProfile->cust_profile_id;
    $objCustAdress = $this->model('cust_address');
    $objCustAdress->cust_profile_id = $_SESSION['cust_profile_id'];
    $objCustAdress->type = 'private';
    $objCustAdress->street = $inputStreet;
    $objCustAdress->zip = $inputZip;
    $objCustAdress->city = $inputCity;
    $objCustAdress->country_id = $inputCountry;
    $status = $objCustAdress->save();
    try {
        $status = $objCustAdress->save();
    } catch (Exception $ex) {

        $logger->error('Beim speichern der User Adresse ist ein Fehler aufgetreten.');
        $logger->debug('User Adresse Exception: ',$ex->getMessage());
        $logger->debug('Adresse-Save: ',$status);
    }
    $success = 'Ihre Daten wurden erfolgreich gespeichert.';
    $_SESSION['cust_address_id'] = $objCustAdress->cust_address_id;

}else{
    $error[] = 'Daten konnten nicht gespeichert werden.';
    $logger->error('Beim speichern des User-Profils ist ein Fehler aufgetreten.');
    $logger->debug('Profil-Save: ',$status);
}
```

Sollte alles erfolreich gespeichert sein, so wird man dem View ‚**DashboardClient**‘ verwiesen welches über die Funktion **Dashboard()** über den Controller „**Client**“ vermittelt wird. Von dort aus hat man dann die Übersicht seiner Angaben.

```php
if($success == true){
    $location = WEB_URL.'/Client/Dashboard';
    header("Location: $location");

}
```

2.5.3 Login System

Über die Funktion **login()** kann man sich einloggen. Man übergibt im View E-Mail (oder Mobilnummer) und Passwort das Formular. In der Funktion wird geprüft ob die Felder gefüllt sind und je nachdem ein Error zurückgegeben. Es wird ein Objekt zum Model ‚**cust_user**' erstellt und über dieses mit der **checkLogin($username, $password)** geprüt. Bei der Überprüfung wird validiert, ob ein Datenbankeintrag bereits besteht.

```
public function login(){
if(isset($_SESSION['logged']) && $_SESSION['logged'] == true){
    $location = WEB_URL.'/Client/Dashboard';
    header("Location: $location");
}


$success = true;
$error = array();
$email = Toolkit::getPost('email','');
$pw = Toolkit::getPost('password','');

if(!empty($email) && !empty($pw)){
    $objUser = $this->model('cust_user');
    $status = $objUser->checkLogin($email, $pw);
    if($status == true){
        if(isset($_SESSION['create_Profile']) && $_SESSION['create_Profile'] == true){
            $location = WEB_URL.'/Client/profile';
            header("Location: $location");
        }elseif(isset($_SESSION['logged']) && $_SESSION['logged'] == true){
            $location = WEB_URL.'/Client/Dashboard';
            header("Location: $location");
        }
    }else{
        $error[] = 'Es tut uns leid, wir konnten dich nicht anmelden.';
    }
}

 $this->view('RegisterClient',array('error' => $error,
                            'success' => $success,
                            'email' => $email));
}
```

Sollte die Funktion **checkLogin($username, $password)** dann true zurückgeben, so wird man dem Dashboard verwiesen. Sollte die jeweilige **cust_user_id** keinen bestehenden Eintrag in der '**cust_profile'** Tabelle hat, wird man ebenfalls eingeloggt, jedoch zum Formular «Persönliche Angaben» verwiesen. Dort kann man dann seine persönlichen Angaben angeben.

```php
public function checkLogin($username, $password){
    if(strlen($username) > 6 && strlen($password) > 3){
        $md5Key = $this->getMd5KeyFromString($password);
        if((!empty($username) && filter_var($username, FILTER_VALIDATE_EMAIL))){
            $arrUser = $this->where('address',$username)
                            ->where('pw', $md5Key)
                            ->join('cust_email', 'cust_user.mail_id', '=', 'cust_email.mail_id')
                            ->first();
        }elseif(!empty($username)){
            require_once 'cust_phone_number.php';
            require_once '../app/class/helper_phone_lib.php';
            $libPhoneObj = new helper_phone_lib();
            $arrIsPhoneNumberValid = $libPhoneObj->isPhoneNumberValidFromCountryId($username, 1);
            $arrUser = $this->where('number',$arrIsPhoneNumberValid['nationalNumber'])
                            ->where('country_id', $arrIsPhoneNumberValid['countryId'])
                            ->where('pw', $md5Key)
                            ->join('cust_phone_number', 'cust_user.phone_number_id', '=', 'cust_phone_number.phone_number_id')
                            ->first();
        }
        if(isset($arrUser->cust_user_id) && !empty($arrUser->cust_user_id) && is_numeric($arrUser->cust_user_id)){
            $_SESSION['cust_user_id'] = $arrUser->cust_user_id;
            $arrProfile = $this->where('cust_user.cust_user_id',$arrUser->cust_user_id)
                               ->join('cust_profile', 'cust_user.cust_user_id', '=', 'cust_profile.cust_user_id')
                               ->first();

            if(isset($arrProfile->cust_profile_id) && !empty($arrProfile->cust_profile_id) && is_numeric($arrProfile->cust_profile_id)){
                $_SESSION['cust_profile_id'] = $arrProfile->cust_profile_id;
                $_SESSION['create_Profile'] = false;
            }else{
                $_SESSION['create_Profile'] = true;
            }
            $_SESSION['logged'] = true;
            return true;
        }
        return false;
    }
}
```

2.5.4 Formular zum Personalbedarf

Das alte Kontaktformular für Kunden unter dem Menüpunkt «Mitarbeiter finden» wird mit einem neuem ersetzt. Beim Aufrufen des Menüpunktes wird man über den Controller «**Pages**» der Funktion **candidate_find()** zugewiesen. Diese Funktion verweist lediglich mit den nötigen Daten aus den Models zum View.

```php
<label for="country" class="text-info">Land</label><br>
<select name="country" class="form-control" id="country" >
    <option selected="selected" >Bitte wählen</option>
    <?php foreach ($data['value_country'] as $country) { ?>
        <option value='<?=$country['country_id']?>'><?=$country['title']?></option>
    <?php } ?>
</select>
```

candidate_find(){

```php
    $error = $success = '';

            $objCountry = $this->model('value_country');
            $arrCountry = $objCountry->get();

    $objAvamProfession = $this->model('seco_avam_profession');
    $arrProfessionList = $objAvamProfession->orderBy('seco_bezeichnung_male_de', 'asc')->get()->all();

    $objProfile = $this->model('cust_profile');
    $arrProfile = $objProfile->where('cust_profile_id',$_SESSION['cust_profile_id'])
                             ->where('cust_user_id',$_SESSION['cust_user_id'])
                             ->first();

    $this->view('MitarbeiterFinden', array('error' => $error,
                                           'success' => $success,
                                           'value_country' => $arrCountry,
                                           'seco_avam_profession' => $arrProfessionList,
                                           'profile' => $arrProfile));
}
```

Beim Formular werden die geforderten Felder von der Aufgabenstellung eingesetzt und um die Felder «Nachricht» und «Betreff» erweitert. Für die Felder «Land» und «Beruf» werden jeweils die übermittelten Arrays '**arrCountry'** und '**arrProfessionList'** eingesetzt. Im View wird dann mit einem Dropdown Menu eine Auswahl an Berufen und Ländern zur Verfügung gestellt.

View mit dem Formular wird geprüft, ob die Variabel **$_SESSION['cust_user_id']** gesetzt ist, also sprich, ob der User eingeloggt ist oder nicht. Hat der Kunde sich bereits registriert, muss er natürlich weniger Felder ausfüllen, da er dies bei der Registrierung schon erledigt hat.

```
<?php if(empty($_SESSION['cust_user_id'])) { ?>
<div class="form-row">
    <div class="form-group col-lg-6">
        <label for="gender" class="text-info">Anrede*</label><br>
        <select name="gender" style="width: auto!important" class="form-control" value="
```

Je nachdem ob der User eingeloggt ist oder nicht, wird das Formular auch einer anderen Funktion übermittelt. **client_perso()** und **guest_perso()**.

```
action="<?= WEB_URL ?>/Pages/<?php if(isset($_SESSION['cust_user_id'])){ echo 'client'; }else{ echo 'guest'; } ?>_perso"
```

2.5.4.1 Übermitteln des Formulars (eingeloggt)

Der eingeloggte Benutzer bekommt beim Formular zum Personalbedarf folgende Felder welche auszufüllen sind: E-Mail Adresse, Telefon, Beruf, Ich suche, Betreff und Nachricht. All diese Felder sind Pflichtfelder. All die restlichen Daten die benötigt werden wie Firmenname, Firmenfunktion, Adresse etc. werden von der Tabelle '**cust_profile'** abgerufen.

Theoretisch hätte es auch mit den Feldern «E-Mail-Adresse» und «Telefon» funktioniert, jedoch soll der Benutzer gezielt diese nochmals angeben. Es kam oft vor, dass die Personalberater sich mit Kontaktierungsproblemen an das Helpdesk gewandt haben, da die Kunden öfter mal ihre Kontaktdaten geändert haben und das soll hiermit verhindert werden.

Wenn der User das Formular nun korrekt ausfüllt und abschickt wird es an die Funktion **client_perso()** übermittelt. Von dort aus werden die übermittelten Post Variablen in ein Array gespeichert und die restlichen nötigen Daten von der Datenbank entnommen. Es überprüft zunächst die Anfrage mit einem reCAPTCHA Response auf Spam und gibt je nach Resultat eine Fehlermeldung zurück und erstellt ein Error log File.

```
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['recaptcha_response'])) {
    // Build POST request:
    $recaptcha_url = 'https://www.google.com/recaptcha/api/siteverify';
    $recaptcha_secret = RECAPTCHA_SECRET_KEY;
    $recaptcha_response = $_POST['recaptcha_response'];

    // Make and decode POST request:
    $recaptcha = file_get_contents($recaptcha_url . '?secret=' . $recaptcha_secret . '&response=' . $recaptcha_response);
    $recaptcha = json_decode($recaptcha);

    // Take action based on the score returned:
    if ($recaptcha->score >= 0.7) {
        // Verified - send email
    } else {
        // Not verified - show form error
        $error = 'Es liegt ein Verdacht an Spam vor. Bitte versuchen Sie es nochmals. Bitte versuchen Sie es bitte nochmals.';
        $logger->error('Beim Übermitteln vom Personalbedarf (eingeloggt) ist ein Fehler aufgetreten.');
        $logger->debug('Form-Submission Error: Erneute Formular-Übermittlung (Spammschutz).');
    }
}
```

Die E-Mail wird auf Gültigkeit validiert und sollte dies der Fall sein, werden die übermittelten Felder in der Tabelle '**cust_request'** gespeichert. Ebenfalls wird in jedem Record die '**cust_profile_id**' und '**cust_user_id**' mitgegeben.

```
if(Toolkit::getPost('task',0) == 'sendperso'){

    $valmail = filter_var($arrInputForm['email'], FILTER_VALIDATE_EMAIL);
    if(!empty($arrInputForm['email'])){
        if ($valmail == false) {
            $arrInputForm['email'] =  '';
            $error = 'Bitte Mail Adresse prüfen';
        }
    }
```

```
$objCustRequest = $this->model('cust_request');

$objCustRequest->f_name = $arrInputForm['f_name'];
$objCustRequest->l_name = $arrInputForm['l_name'];
$objCustRequest->gender = $arrInputForm['gender'];
$objCustRequest->status = $arrInputForm['status'];
$objCustRequest->compfunction = $arrInputForm['compfunction'];
$objCustRequest->company = $arrInputForm['company'];

$objCustRequest->country = $arrInputForm['country'];
$objCustRequest->street = $arrInputForm['street'];
$objCustRequest->city = $arrInputForm['city'];
$objCustRequest->zip = $arrInputForm['zip'];

$objCustRequest->profession = $arrInputForm['profession'];
$objCustRequest->profession_id = $arrInputForm['profession_id'];
$objCustRequest->look_for = $arrInputForm['look_for'];

$objCustRequest->mobile = $arrInputForm['mobile'];
$objCustRequest->mail = $arrInputForm['email'];
$objCustRequest->subject = $arrInputForm['subject'];
$objCustRequest->message = $arrInputForm['message'];

$objCustRequest->cust_profile_id = $_SESSION['cust_profile_id'];
$objCustRequest->cust_user_id = $_SESSION['cust_user_id'];


try{
$status = $objCustRequest->save();
}catch (Exception $ex) {
            $error = 'Ihre Nachricht konnte nicht übermittelt werden.';
            $logger->error('Beim User speichern ist ein Fehler aufgetreten.');
            $logger->debug('User-Save Exception:', $ex->getMessage());
            $logger->debug('User-Save Error:', $status);
        }
```

Sollten die Daten des Users nicht gespeichert werden können, wird ein Logfile erstellt und eine Error Nachricht übermittelt. Wenn sie allerdings erfolgt, wird das Formular per E-Mail an die Zweigstellte zugesendet und der Kunde erhält eine Benachrichtigung.

```
$objSwiftMailer = new Swiftmailer();
$arrMail['subject'] = $arrInputForm['subject'].' - Absender:'.$arrInputForm[
$arrMail['to']= DEFAULT_MAIL_RECEIVER;
$arrMail['body']= '<html><body style="font-family: Calibri, Arial, Helvetica,
                        <p>Die Nachricht des Kunden: '.$arrInputForm['
                       Der registrierte Kunde '.$arrInputForm['f_name'
                          hat einen Personalbedarf abgeschickt. Der Ku
                          Vorname:      '.$arrInputForm['f_name'].'
                          Name:          '.$arrInputForm['l_name'].'
                          E-Mail:         '.$arrInputForm['email'].'
                          Telefon:       '.$arrInputForm['mobile'].'
                          Firma:          '.$arrInputForm['company']
                          Funktion:     '.$arrInputForm['compfunctio
                          Strasse/Nr.:  '.$arrInputForm['street'].'
                          PLZ:              '.$arrInputForm['zip'].'
                          Ort:               '.$arrInputForm['city']
```

2.5.4.2 Übermitteln des Formulars (ausgeloggt)

Da der Benutzer sich als Gast auf der Seite befindet, bekommt dieser beim Formular zum Personalbedarf folgende Felder, welche auszufüllen sind: Anrede, Vorname, Name, E-Mail Adresse, Telefon, Firma, Funktion, Strasse/Nr., PLZ, Ort, Land, Beruf, Ich suche, Betreff und Nachricht. All diese Felder sind Pflichtfelder.

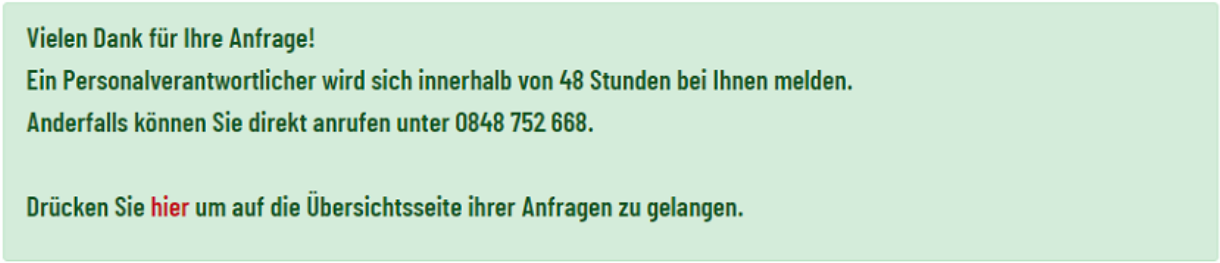
Das Formular übermittelt für den ausgeloggten User die Daten zur Funktion **guest_perso()**.

Von dort aus verläuft dann alles wie bei der Funktion **client_perso()**. Der Unterschied hier ist, dass keine Sessionvariablen gespeichert werden und somit der Anfrage keinen Benutzer zugeordnet wird.

Im gesendeten Mail wird auch nochmal niedergeschrieben, dass der Kunde das Formular als Gast übermittelt hat.

2.5.5 Übersichtsseite

Beim erfolgreichen übermitteln des Formulars, wird ein **succes** alert von Bootstrap zurückgegeben. Als eingeloggter Benutzer ist es eine Meldung in welcher man die Nachricht nach Aufgabenstellung übermittelt bekommt, samt einer Verlinkung zu seiner Übersichtsseite.



Beim erfolgreichen Übermitteln des Formulars als Gast, wird einem direkt die Übersicht seiner Anfrage einmalig angezeigt und der Gast wird darauf hingewiesen, sich optional registrieren zu können.

2.5.6 Übermittlung an E-Mail Adresse

In der Funktion **client_perso()** als auch **guest_perso()** wird bei erfolgreichem speichern des Formulars in die Datenbank eine Instanz der Klasse «**Swiftmailer**» erzeugt. Es wird das Array **$arrMail** erstellt mit den Feldern: «subject», «to», «body» und «message». Für «subject» wird der Betreff eingesetzt und es wird der Absender angehangen. Für das Feld «to» wird die definierte Konstante meiner Developer Umgebung (settings_dev.php), **DEFAULT_MAIL_RECIEVER**, eingesetzt. Die Konstante wurde selbstverständlich mit «info@planova.ch» definiert. In den Feldern «body» und «message» ist die Nachricht für die Personalberater hinterlegt, hierbei sind alle angegebenen Informationen des Formulars enthalten, samt ob der angegebene Beruf meldepflichtig ist oder nicht. Der Unterschied hierbei, im Feld «body» sind die HTML Elemente definiert um die Nachricht strukturiert ausgeben zu können, sollte man dies in der Webseite machen wollen. In unserem Fall geben wir aber noch nirgendwo die Nachricht innerhalb der Webseite aus.

```
$objSwiftMailer = new Swiftmailer();
$arrMail['subject'] = $arrInputForm['subject'].' - Absender:'.$arrInputForm['email'];
$arrMail['to']= DEFAULT_MAIL_RECEIVER;
$arrMail['body']= '<html><body style="font-family: Calibri, Arial, Helvetica, sans-serif; font-size: 12pt;">
                    <p>Die Nachricht des Kunden: '.$arrInputForm['message'].'<br/><br/>
                   Der registrierte Kunde '.$arrInputForm['f_name'].$arrInputForm['l_name'].' von der Firma '.$arrInputForm['company'].'<br/>
                      hat einen Personalbedarf abgeschickt. Der Kunde füllte die Felder wie folgt aus:<br/><br/>
                      Vorname:      '.$arrInputForm['f_name'].'
                      Name:          '.$arrInputForm['l_name'].'
                      E-Mail:          '.$arrInputForm['email'].'
                      Telefon:        '.$arrInputForm['mobile'].'
                      Firma:          '.$arrInputForm['company'].'
                      Funktion:      '.$arrInputForm['compfunction'].'
                      Strasse/Nr.:   '.$arrInputForm['street'].'
                      PLZ:              '.$arrInputForm['zip'].'
                      Ort:               '.$arrInputForm['city'].'
                      Land:            '.$arrInputForm['country'].'
                      Beruf:           '.$arrInputForm['profession'].'
                      Ich suche:     '.$arrInputForm['look_for'].'<br/><br/>
                      Freundliche Grüsse
                      Ihr planova Team</p>
                   '. '</body></html>';
$arrMail['message']= 'Die Nachricht des Kunden: '.$arrInputForm['message'].'

                      Der registrierte Kunde '.$arrInputForm['f_name'].' '.$arrInputForm['l_name'].' von der Firma '.$arrInputForm['company'].'
                      hat einen Personalbedarf abgeschickt. Der Kunde füllte die Felder wie folgt aus:

                      Vorname:      '.$arrInputForm['f_name'].'
                      Name:          '.$arrInputForm['l_name'].'
                      E-Mail:          '.$arrInputForm['email'].'
                      Telefon:        '.$arrInputForm['mobile'].'
                      Firma:          '.$arrInputForm['company'].'
                      Funktion:      '.$arrInputForm['compfunction'].'
                      Strasse/Nr.:   '.$arrInputForm['street'].'
                      PLZ:              '.$arrInputForm['zip'].'
                      Ort:               '.$arrInputForm['city'].'
                      Land:            '.$arrInputForm['country'].'
                      Beruf:           '.$arrInputForm['profession'].'
                      Ich suche:     '.$arrInputForm['look_for'].'

                      Der Beruf ist'.$statusProfession.'meldepflichtig.


                      Freundliche Grüsse
                      Ihr planova Team';
$objSwiftMailer->sendMail($arrMail);

$success = 'Vielen Dank für Ihre Nachricht! <br />Wir melden uns so schnell wie möglich bei Ihnen.'.$arrMail['body'];
```

Über die Funktion **sendmail($arrMail)** wird die Nachricht dann an die Zweigstelle übermittelt und es wird in der Nachricht hinterlegt, dass der Kunde eingeloggt war bei der Übermittlung des Formulars.

2.5.7 Felder Überprüfung (Clientseitig)

Beim Formular zum Registrieren wird im HTML je nachdem ob man auf die «Via Mobile» oder «Via E-Mail» Methode drückt ein onclick Event per JavaScript ausgeführt. Gleichzeitig werden die Attribute des Gegenfeldes deaktiviert, wie zum Beispiel das Attribut «required».

```
<a class="btn btn-link" onclick="$('#login_email').val('');  $('#login_email').attr('disabled', true);
    $('#login_email').removeAttr('required'); $('#login_telefon').removeAttr('disabled'); $('#login_telefon').attr('required', true);"
```

```
<script src="<?= WEB_URL ?>/tmpl_planovavl/js/jquery.validate.js"></script>
```

Bevor man irgendein Formular absendet, ist es wichtig sicherzustellen, dass alle erforderlichen Formularfelder im richtigen Format ausgefüllt sind. Dies wird als clientseitige Formularvalidierung bezeichnet und hilft sicherzustellen, dass die übermittelten Daten den Anforderungen entsprechen, die in den verschiedenen Formularsteuerelementen festgelegt sind.

Das Registrierungsformular ist mit der ID «**register-wizard**» versehen. Über die ID wird auch hier im Footer nach erforderlichen Feldern geprüft und ob die Felder mit keinen Sonderzeichen versehen sind welche unpassend sind.

```
}).validate({
    errorPlacement: function errorPlacement(error, element) {
        element.before(error);
    },
    rules: {
        firstname: {
            required: true,
            cleanNames: '[0-9_.*!?+%@():,#{}=|;<>/]'
        },
        lastname: {
            required: true,
            cleanNames: '[0-9_.*!?+%@():,#{}=|;<>/]'
        },
        company: {
```

```
messages: {
    firstname: {
        cleanNames: 'Vorname enthält nicht erlaubte Zeichen.'
    },
    lastname: {
        cleanNames: 'Nachname enthält nicht erlaubte Zeichen.'
    },
    company: {
```

Dasselbe gillt auch für das Formular zum Personalbedarf:

```
if ($('#staffRequestForm')) {
    $('#staffRequestForm').validate({
        errorPlacement: function errorPlacement(error, element) {
            element.before(error);
        },
        rules: {
            firstname: {
                required: true,
                cleanNames: '[0-9_.*!?+%@():,#{}=|;<>/$]'
            },
            lastname: {
```

2.5.8 Felder Überprüfung mit Regex (Server)

Beim Übermitteln des Formulars zum Registrieren, wird zu Beginn der Übermittlung die Post Variablen in ein Array initialisiert. Die einzelnen Elemente dieses Arrays werden an die Funktion **checkClientFields(array())** aus der Klasse Toolkit.

Von der Funktion aus wird dann jedes übermittelte Feld mit dem passenden Muster und die zu durchsuchende Zeichenkette abgestimmt.

```
public static function checkClientFields($arrInputFields) {
if (empty($arrInputFields) || !is_array($arrInputFields)) {
    return false;
}
$arrError = [];
$phoneUtil = \libphonenumber\PhoneNumberUtil::getInstance();

$regex = '/[0-9\_\.\*\!\?\+\%\@\(\)\,]/';


if (!empty($arrInputFields['inputFirstname'])) {
    preg_match($regex, $arrInputFields['inputFirstname'], $arrMatches);
    if (!empty($arrMatches)) {
        $arrError[] = 'Der Vorname enthält nicht erlaubte Zeichen.';
    }
}
```

Telefonfelder werden mit der **isValidPhoneNumber($phoneNumber) Funktion** validiert aus der Klasse **PhoneNumberUtil**.

```
public static function isValidPhoneNumber($phoneNumber){
    $phoneUtil = \libphonenumber\PhoneNumberUtil::getInstance();
    $arrPhoneNumber = $phoneUtil->parse($phoneNumber, 'CH');
    $isValid = $phoneUtil->isValidNumber($arrPhoneNumber);
    if($isValid === false) {
        return false;
    }
    return true;
}
```

E-Mail-Felder werden mit einem speziellen Pattern abgestimmt.

```
if (!empty($arrInputFields['inputLoginEmail'])) {
    $result = preg_match('/^[äöüÄÖÜa-zA-Z0-9\!\#\$\%\&\'\*\+\-\/\=\?\^\_\`\{\|\}\~]+[\.]*([äöüÄÖÜa-zA-Z0-9\.\!\#\$\%\&\'\*\+\-\/\=\?\^\_\`\{\|\}\~]'
                    . '+)@[äöüÄÖÜa-zA-Z0-9\.\!\#\$\%\&\'\*\+\-\/\=\?\^\_\`\{\|\}\~]+[\.]*'
                    . '([äöüÄÖÜa-zA-Z0-9\.\!\#\$\%\&\'\*\+\-\/\=\?\^\_\`\{\|\}\~]+)\.[a-zA-Z]{2,}$/i', $arrInputFields['inputLoginEmail']);
    if ($result === false) {
        $error[] = 'Die eingegebene E-Mail Adresse ist ungültig.';
    }
```

2.5.9 Wiedersteht beliebten Angriffsmethoden wie JavaScript- oder SQL-Injection

Ein SQL-Injection-Angriff tritt auf, wenn ein Benutzer böswillige SQL-Bits in einer Datenbankabfrage einfügt. Am häufigsten geschieht dies, wenn einem Benutzer ermöglicht wird, Eingaben an eine Datenbankabfrage ohne Validierung zu übergeben, wodurch die ursprünglich beabsichtigte Abfrage geändert werden kann. Durch das Einfügen von eigenem SQL kann der Benutzer Schaden anrichten durch:

- Lesen sensibler Daten
- Verändern sensibler Daten
- Löschen sensibler Daten

Wie man sich wahrscheinlich vorstellen kann, können diese Arten von Angriffen negative Auswirkungen auf die Anwendungen und das Unternehmen haben. Tatsächlich gab es schon einige großen Unternehmen, die in den letzten Jahren in Datenschutzverletzungen verwickelt waren. Dies kann zum Verlust von Kunden, Umsatz, Anwendungsverfügbarkeit und mehr führen.

Bei einer JavaScript-Injection funktioniert das ebenso, nur versucht man meistens auf die Cookie oder Session Daten ran zu kommen, indem man zum Beispiel das onerror Event auslöt.

Beispiele:

```
<img src onerror=“alter(document.cookie)>
```

→ Es wird keine src erkannt, die onerror Funktion wird aufgerufen und die Query gibt Daten aus welche in den Cookies gespeichert sind.

```
Peter“ OR 1=1
```

→ Die obige SQL ist gültig und würde alle Zeilen aus der Tabelle zurückgeben, wenn die Query zum Beispiel eine Suche wäre, da OR 1=1 immer wahr ist.

Um das zu verhindern wurden folgende Methoden verwendet:

-Clientseitige Felder Überprüfung, Sonderzeichen wie „=,.<>“ etc. werden nicht übermittelt.
-Serverseitige Überprüfung, die vorgeprüften Felder werden durch ein regex Pattern abgestimmt.

Zudem wurden Raw Queries vermeiden. Ein Beispiel zu einem Raw Query:

```
$someVariable = Input::get("some_variable");
$results = DB::select( DB::raw("SELECT * FROM some_table WHERE some_col = '$someVariable'") );
```

In der obigen Abfrage fügen wir Benutzereingaben direkt in die Abfrage ein, ohne sie zu bereinigen. Mit so einer Abfrage wäre man zum Beispiel angreifbar.

Durch das vermeiden solcher Abfragen, werden Sicherheitsfunktionen die bereits in das Framework integriert sind genutzt.
(Laravel, 2022)

2.5.10 Autocomplete

Es wurde eine Testumgebung eingerichtet, in welche die Erweiterung „Autocomplete“ getestet werden kann.

Dafür wurde ein Test-Controller namens „AutocompleteController“ erstellt welche auf Controller vererbt wird:

```php
class AutocompleteController extends Controller
{

    function index()
    {
        $this->view('autocompleteTest');
    }


       public function autocomplete(){

        $input = $_POST['geozip_zip'];
        $objValueZip = $this->model('value_zip');


       // var_dump($objValueZip);

        $query = $objValueZip->where('geozip_zip', $input)->pluck('geozip_city')->first();

        var_dump($query);

    }


}
```

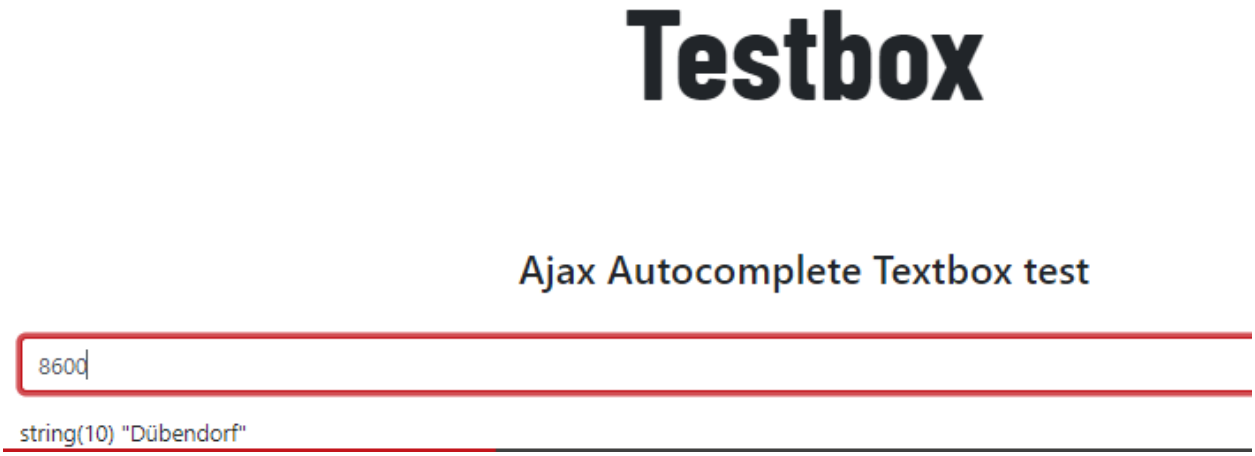
Das aufgerufene View:

```php
<div class="pl-cmmn-cnt-section">
    <div class="pl-ueber">
        <h1 class="bs-title">Testbox</h1>

    </div>
  <div class="container box">
   <h3 align="center">Ajax Autocomplete Textbox test</h3><br/>
   <div class="form-group">
       <input type="text" name="geozip_zip" id="geozip_zip" class="form-control input-lg"
              placeholder="Enter Country Name"/>


  </div>

   <div id="zipList">
   </div>


  </div>
 </body>
</html>



<?php

include "footer.php"
?>
```

In der View, wird durch den Footer ein jQuery **keyup()** aufgerufen. Das **keyup()** Event wird hierbei ausgelöst, sobald, das Feld mit der ID „**geozip_zip**“ befüllt wird. Die Funktion sieht dann so aus:

```
$('#geozip_zip').keyup(function(){
    var geozip_zip = $(this).val();
    if(geozip_zip != '')
    {
        $.ajax({
            url: "<?= WEB_URL ?>/AutocompleteController/autocomplete",
            method:"POST",
            data:{geozip_zip:geozip_zip},
            success:function(data)
            {

                $('#zipList').fadeIn();
            $('#zipList').html(data);

            }
        });
        return data;

   }

});
```

Soltle die Variabel, also das Feld nicht leer sein, so wird über Ajax die Funktion **autcomplete()** aufgerufen. Das Feld wird hier als „**data**“ und per POST Methode übermittelt. Mit einer einfachen Query wird dann am Schluss der jeweilige Ort passend zur Postleitzahl ausgegeben ohne dass die Seite neugeladen wird und ist dabei flexibel.



Zum Schluss war leider keine Zeit mehr übrig, um die Formulare um diese Funktion zu erweitern.

2.5.11 Ändern der Persönlichen Angaben

Wenn der Benutzer eingeloggt ist, hat er von seinem „my Planova“ Bereich (Profil) unter dem Reiter „Persönliche Angaben“ die Möglichkeit seine persönlichen Angaben samt Kontaktdaten zu ändern. Hier werden ausschliesslich Daten über die Tabelle ‚**cust_profile**‘ und ‚**cust_address**‘ geändert. Das heisst, die Login Methode würde sich nicht ändern, wenn man ein anderes E-Mail angibt.

Bei der Übermittlung werden die einzelnen Felder validiert mit der **checkClientFields(array())** Funktion. Zunächst wird ein Objekt über ‚**cust_profile**‘ mit dem Input des Users gefüllt und anschliessend über das Model gespeichert.

```
if ($task == 'edit') {
    $objCustProfile = $this->model('cust_profile');
    $profile = $objCustProfile::find($_SESSION['cust_profile_id']);
    $profile->cust_user_id = $_SESSION['cust_user_id'];
    $error = Toolkit::checkClientFields(array('inputFirstname' => $arrInputForm['f_name'], 'inputLastname' => $arrInputForm['l_name'],
    'inputCompany' => $arrInputForm['company'], 'inputCompfunction' => $arrInputForm['compfunction']));

    if(empty($error)){

            $profile->gender = $arrInputForm['salutation'];
            $profile->f_name = $arrInputForm['f_name'];
            $profile->l_name = $arrInputForm['l_name'];
            $profile->company = $arrInputForm['company'];
            $profile->compfunction = $arrInputForm['compfunction'];


        $status = $profile->save();
```

Bei der Adresse funktioniert das ebenso, nur wird hier zusätzlich noch geprüft ob bereits ein Eintrag existiert welcher überschrieben werden kann, wenn nicht wird einer erstellt:

```
    if ($status == true) {
        if(isset($_SESSION['cust_address_id']) && !empty($_SESSION['cust_address_id'])){
            $objCustAddress = $this->model('cust_address');
            $address = $objCustAddress->find($_SESSION['cust_address_id']);
        }else{
            $address = $this->model('cust_address');
            $address->type = 'private';
            $address->cust_profile_id = $_SESSION['cust_profile_id'];
        }

        $address->street = $arrInputForm['street'];
        $address->zip = $arrInputForm['zip'];
        $address->city = $arrInputForm['city'];
        $address->country_id = $arrInputForm['country'];
        $status = $address->save();

        $success = 'Ihre Daten wurden erfolgreich gespeichert.';
    }
}
```

Anschliessend wird bei der Bedingung, dass die **cust_profile_id** gesetzt nochmal ein Objekt für ‚**cust_profile**' und ‚**cust_address**' erstellt. Mit einer Abfrage wird aus diesen Objekten ein Array erstellt mit dem jeweiligen Eintrag des Users. Die Abfrage vergleicht die Session Variable **cust_profile_id** jeweils.

```
    if(isset($_SESSION['cust_profile_id']) && !empty($_SESSION['cust_profile_id'])){
        $objProfile = $this->model('cust_profile');
        $arrProfile = $objProfile->where('cust_profile_id',$_SESSION['cust_profile_id'])
                                 ->where('cust_user_id',$_SESSION['cust_user_id'])
                                 ->first();
        $objCustAdress = $this->model('cust_address');
        $arrAddress = $objCustAdress->where('cust_profile_id',$_SESSION['cust_profile_id'])
                                    ->where('type','private')
                                    ->get();
        $_SESSION['cust_address_id'] = $arrAddress[0]->cust_address_id;
        $arrInputForm = array();
        $arrInputForm['salutation'] = $arrProfile->gender;
        $arrInputForm['f_name'] = $arrProfile->f_name;
        $arrInputForm['l_name'] = $arrProfile->l_name;
        $arrInputForm['company'] = $arrProfile->company;
        $arrInputForm['compfunction'] = $arrProfile->compfunction;

        if(!empty($arrProfile->mobile)){
            $arrInputForm['mobile'] = $phoneUtil->format($phoneUtil->parse($arrProfile->mobile, 'CH'), \libphonenumber\PhoneNumberFormat::INTERNATIONAL);
        }else{
            $arrInputForm['mobile'] = '';
        }

        $arrInputForm['mail'] = $arrProfile->mail;
        $arrInputForm['street'] = $arrAddress[0]->street;
        $arrInputForm['zip'] = $arrAddress[0]->zip;
        $arrInputForm['city'] = $arrAddress[0]->city;
        $arrInputForm['country'] = $arrAddress[0]->country_id;

    }

}
$this->view('ProfileClient', array('value_country' => $arrCountry,
                                   'profile_form' => $arrInputForm,
                                   'success' => $success,
                                   'error' => $error));
```

Das Array wird mit den Attributen des Records gefüllt und dem View übergeben. Im View wird dann für jedes Input als Value das übermittelte Array eingesetzt.

```
'profile_form' => $arrInputForm,
```

2.5.12 Passwort ändern

Wenn der Benutzer eingeloggt ist, hat er von seinem „my Planova“ Bereich (Profil) unter dem Reiter „Passwort ändern“ die Möglichkeit sein Passwort zu ändern. Hier wird ein einfaches Formular gebraucht, in welchem man dazu aufgerufen wird, sein aktuelles Passwort zu bestätigen und zweimal sein neues.

Hier wird geprüft ob das neue Passwort, welches zweimal ausgefüllt wird, und ob das aktuelle Passwort übereinstimmt.

```
public function changePassword(){

    if(isset($_SESSION['cust_profile_id']) && !empty($_SESSION['cust_profile_id'])){
        $task  = Toolkit::getPost('task','');
        $error = '';
        $success = '';
        if(!empty($task) && $task == 'changePassword'){
            $objCandUser = $this->model('cust_user');
            $userData = $objCandUser::find($_SESSION['cust_user_id']);
            $oldPassword  = Toolkit::getPost('oldpassword','');
            $newPassword  = Toolkit::getPost('newpassword','');
            $newPassword2 = Toolkit::getPost('newpassword2','');

            if($userData['pw'] == $oldPassword && $newPassword == $newPassword2){
                $success = 'Passwort wurde erfolgreich gespeichert.';
                $candUserObj = $objCandUser::find($_SESSION['cust_user_id']);
                $arrSaveData = ['pw' => $newPassword];
                $result = $candUserObj->fill($arrSaveData)->save();
            }elseif($userData['pw'] != $oldPassword){
                $error = 'Aktuelles Passwort ist falsch.';
            }elseif($newPassword != $newPassword2){
                $error = 'Die Passwörter stimmen nicht überein. Erneut versuchen?';
            }
        }

        $this->view('ChangePassword', ['success' => $success, 'error' => $error]);
    }else{
        $location = WEB_URL.'/Client/login';
        header("Location: $location");
    }
}
```

Diese Funktion wird auch für die Gruppe Kandidat genutzt, indem man prüft ob die **$_SESSION[‚cust_user_id‘]** gesetzt ist, wird man zum richtigen Controller mit dieser Funktion weitergeleitet.

```
action='<?=WEB_URL?>/<?= isset($_SESSION['cust_user_id']) ? 'Client' : 'Candidate' ?>/ChangePassword'>
```

2.5.13 Oberfläche

Die Oberfläche wurde mit Bootstrap zu realisiert. Der grosse Vorteil von Bootstrap sind vordefinierte Komponenten. Diese Komponenten sind in CSS-Klassen gegliedert und eine Kombination dieser Klassen erlaubt eine sehr einfache Entwicklung und grobe Individualisierung ohne viel Aufwand. Damit ich mich nicht um die Mobile-Ansicht gross kümmern muss, da Bootstrap dies in Ihren CSS-Klassen schon prima handhabt, erspart mir das viel Arbeit im Design. (Bootstrap, 2022)

Bei der Ansicht wurde darauf geachtet, dass der allgemeine Still der Webseite eingehalten wird und alles etwas einheitlich aussieht.

Zusatz: Die Home Seite wurde der Banner sowie die Verlinkung „Arbeit finden“ zunächst mit der CSS-Klasse „**container**“ mittig dargestellt. Zuvor waren diese linksbündig und liessen die Seite unfertig wirken.
Home.php:

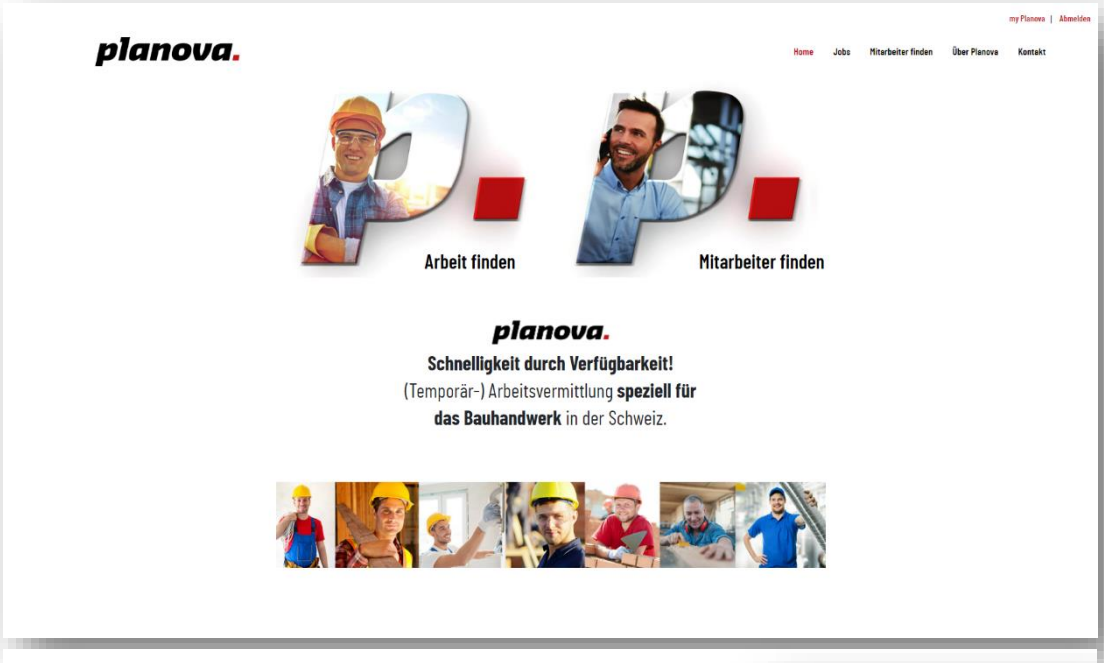


Abbildung 5: Startseite

Abbildung 6: Startseite (Smartphone)

RegisterClient.php:

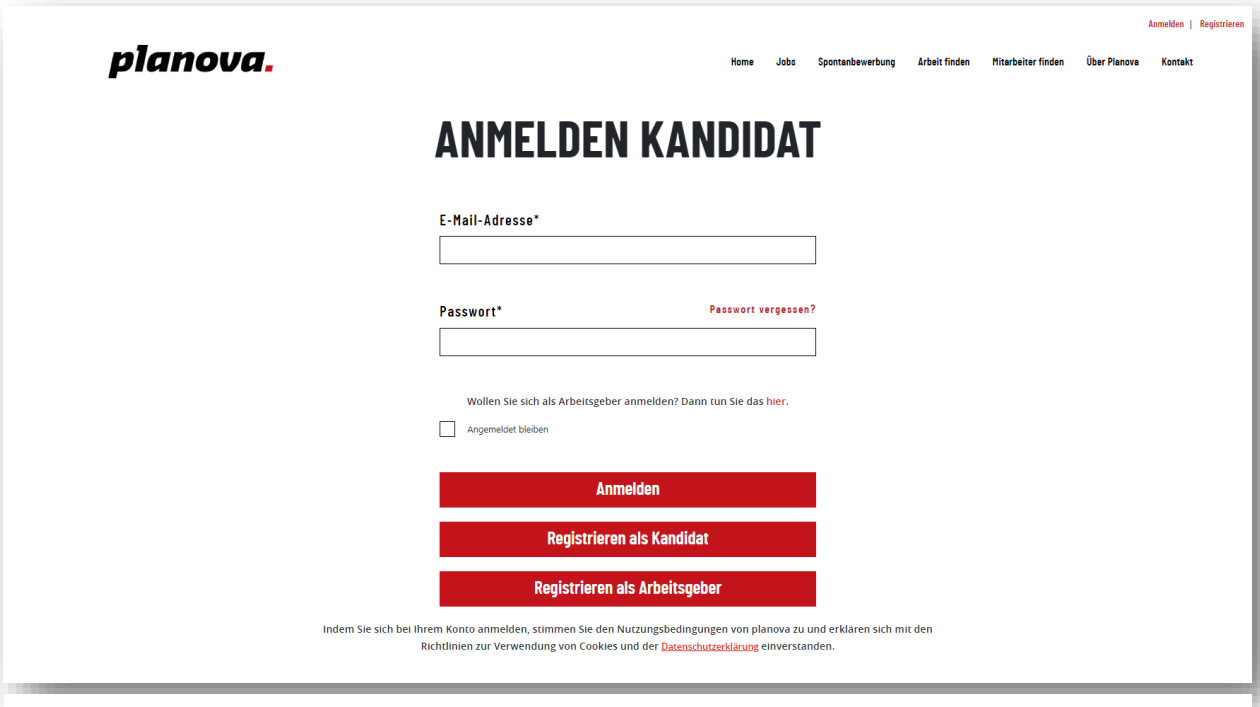


Abbildung 7: Login Formular

Abbildung 8: Login Formular (Smartphone)

RegisterNewClient.php (Fieldset 1):

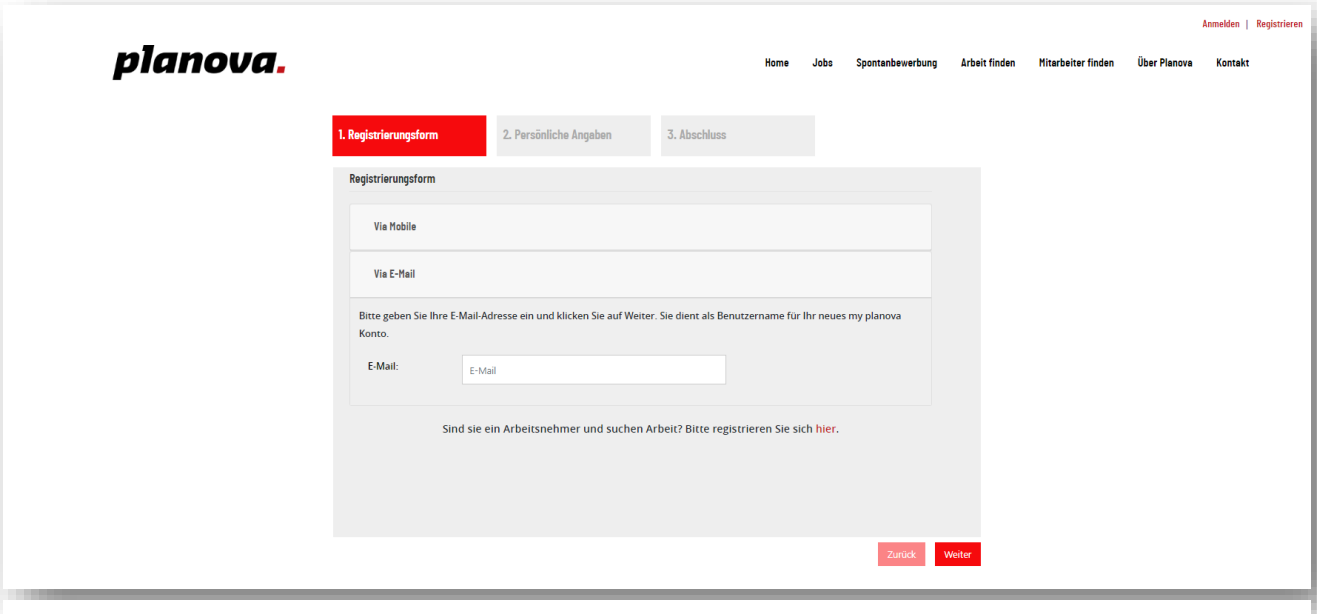


Abbildung 9: Registrierungsformular 1

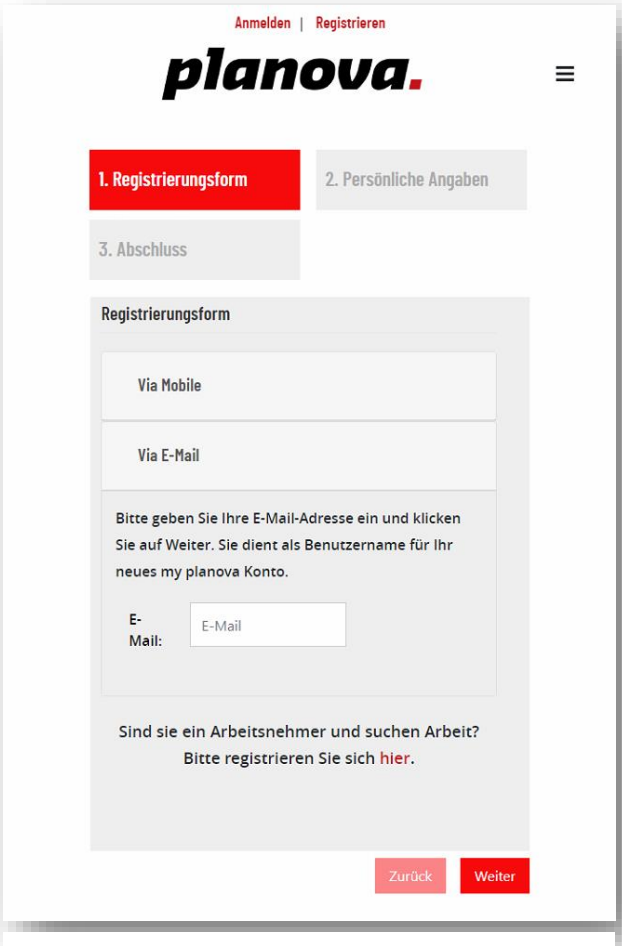


Abbildung 10: Registrierungsformular 1 (Smartphone)

RegisterNewClient.php (Fieldset 2):

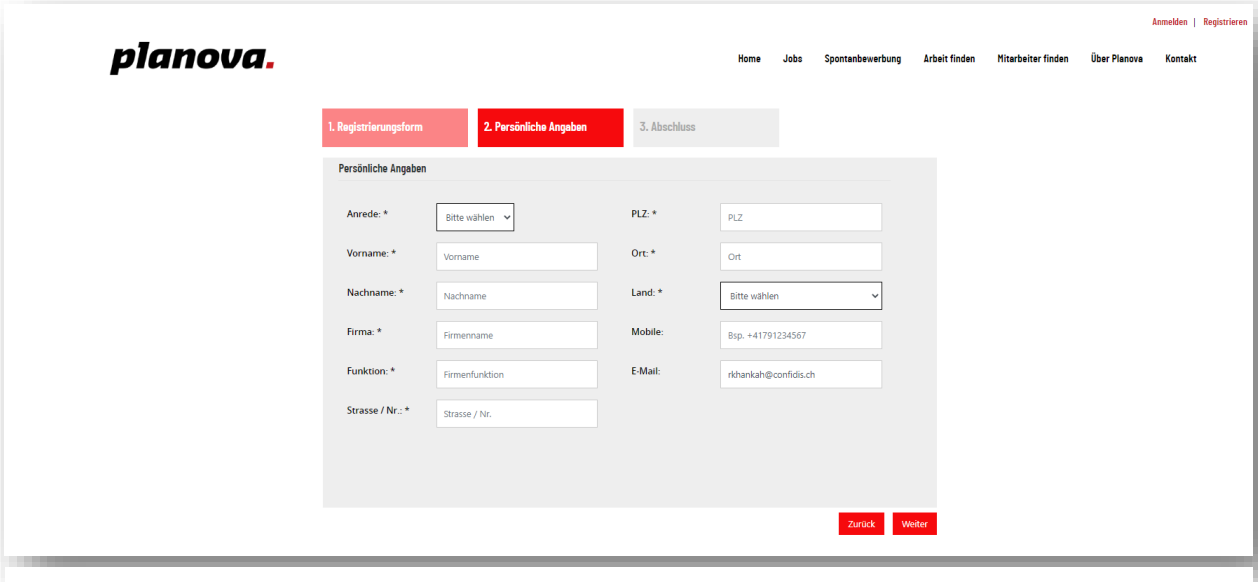


Abbildung 11: Registrierungsformular 2

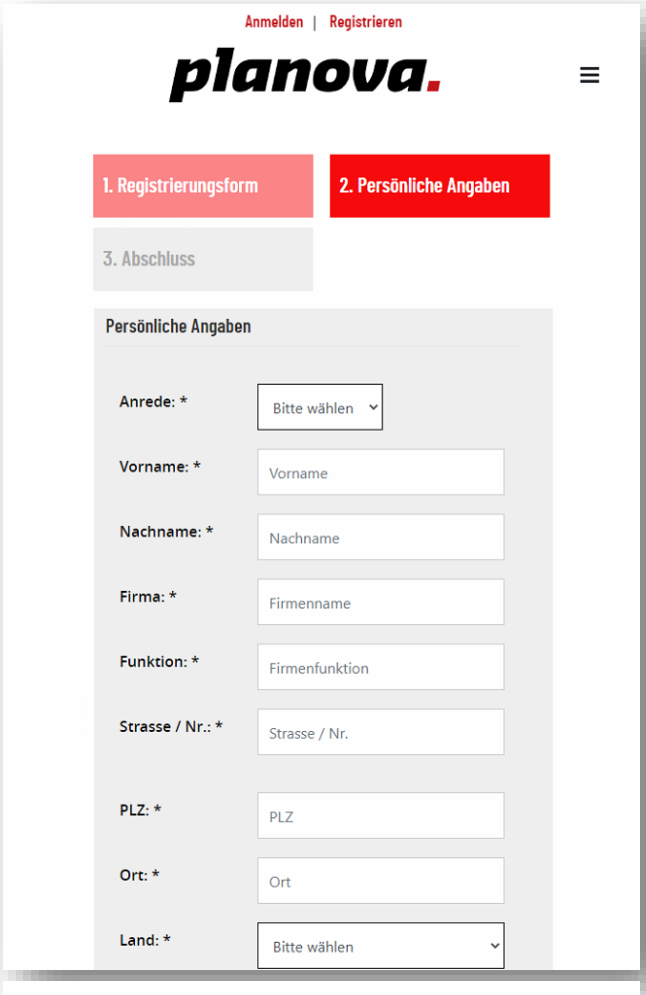


Abbildung 12: Registrierungsformular 2 (Smartphone)

RegisterNewClient.php (Fieldset 3):

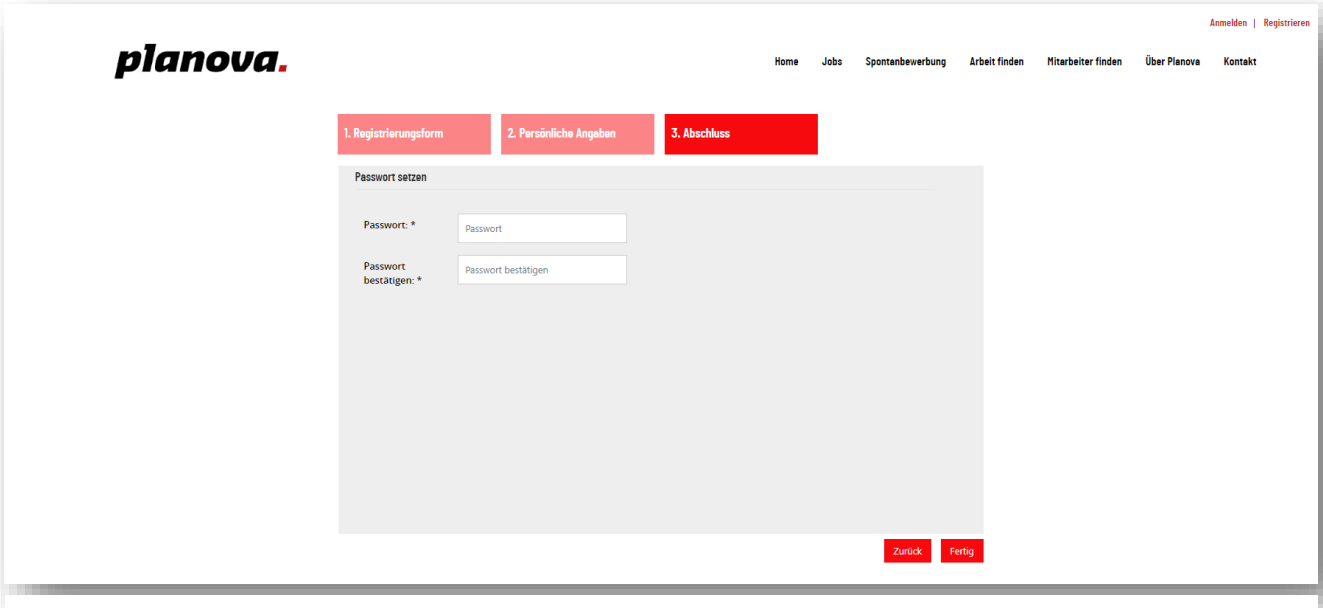


Abbildung 13: Registrierungsformular 3

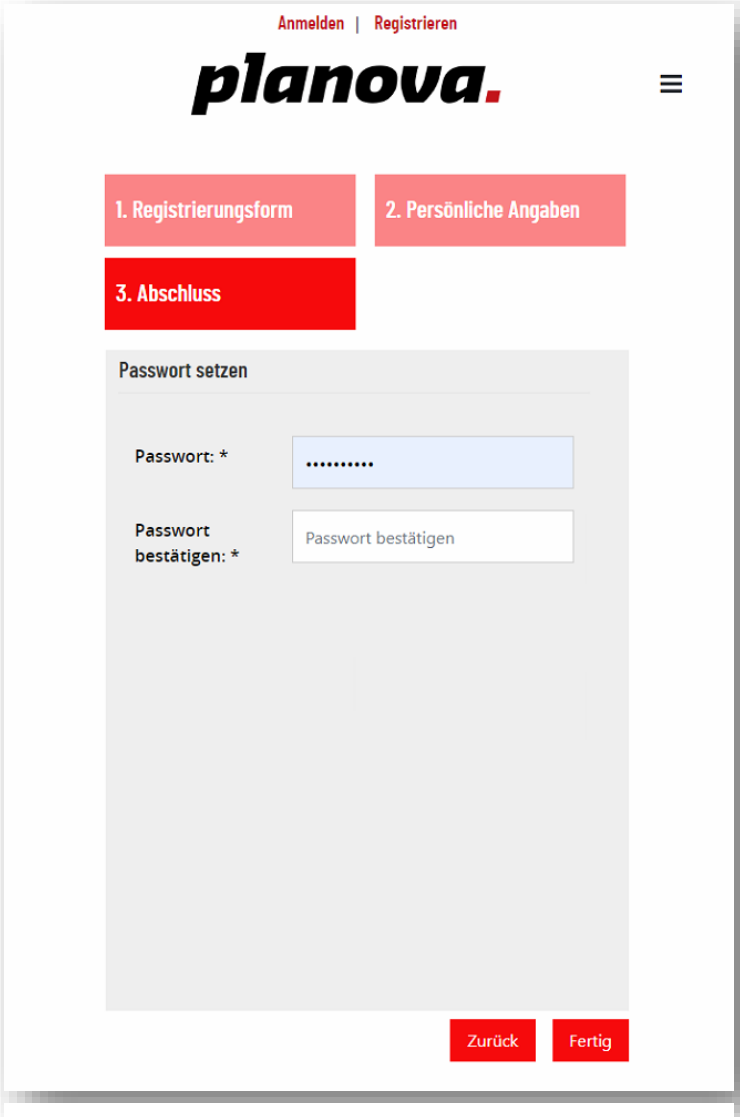


Abbildung 14: Registrierungsformular 3 (Smartphone)

DashboardClient.php:

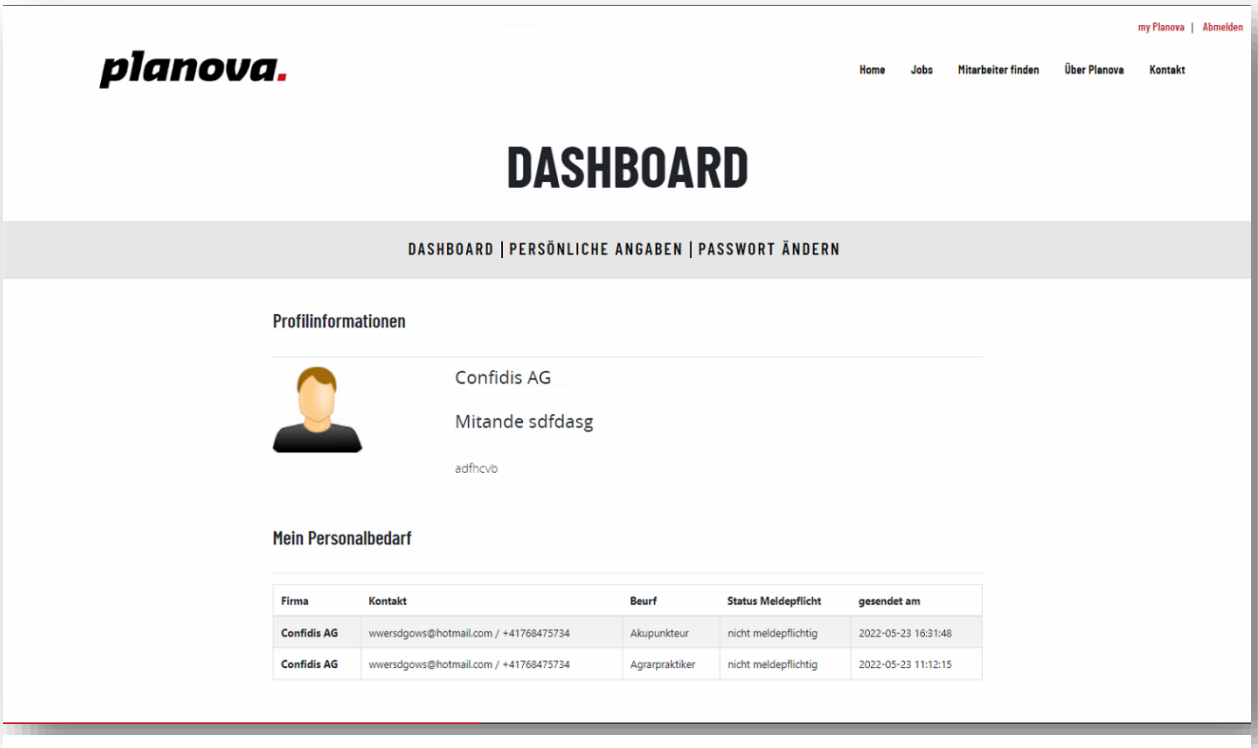


Abbildung 15: Dashboard

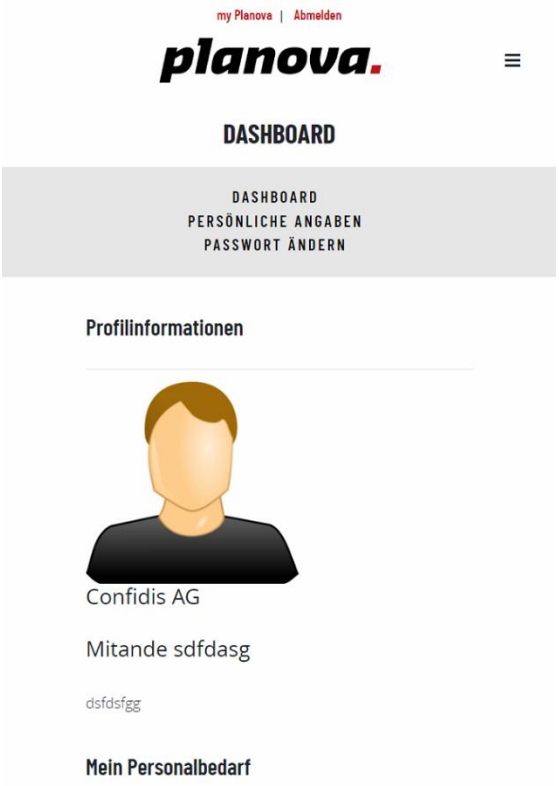


Abbildung 16: Dashboard (Smartphone)

ProfileClient.php:

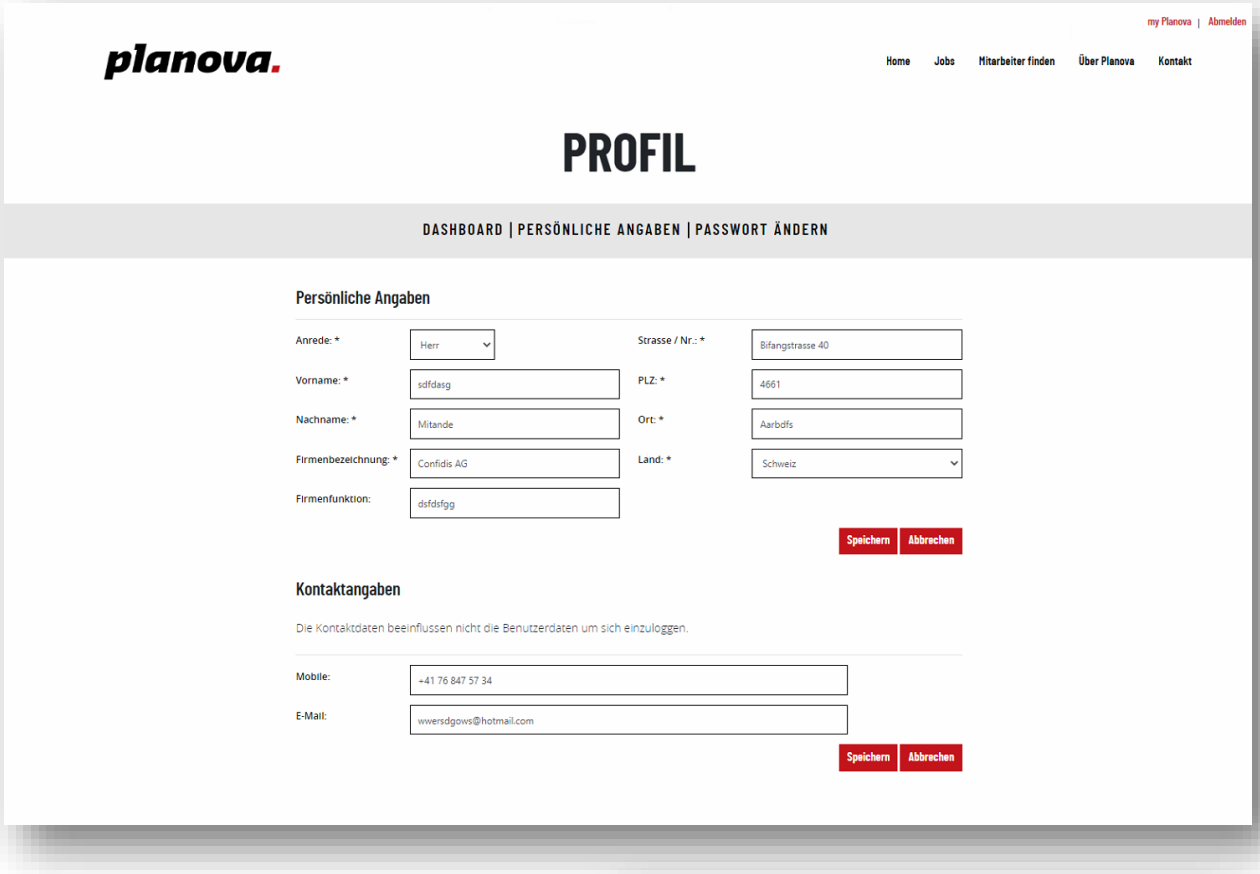


Abbildung 18: Profil

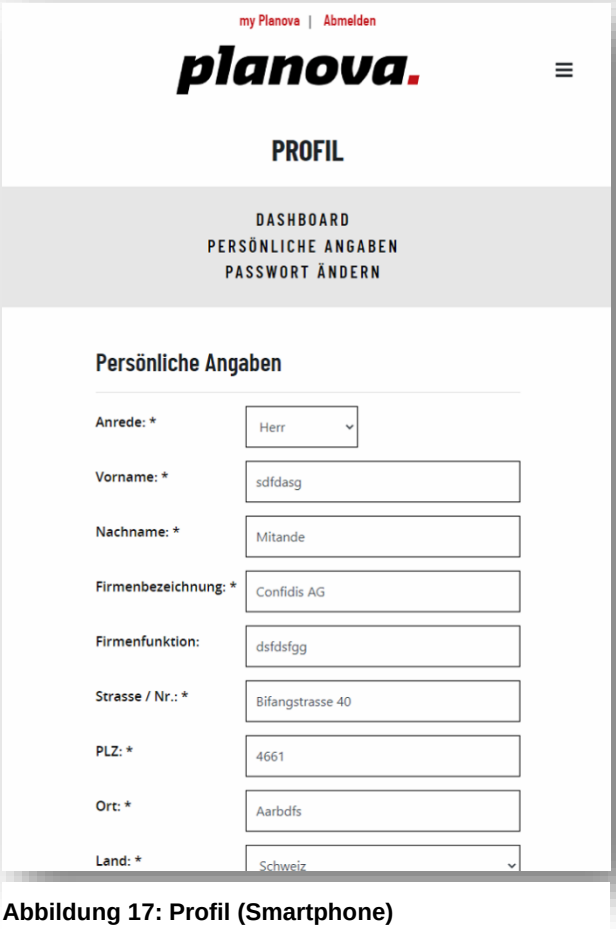


Abbildung 17: Profil (Smartphone)

ChangePassword.php:

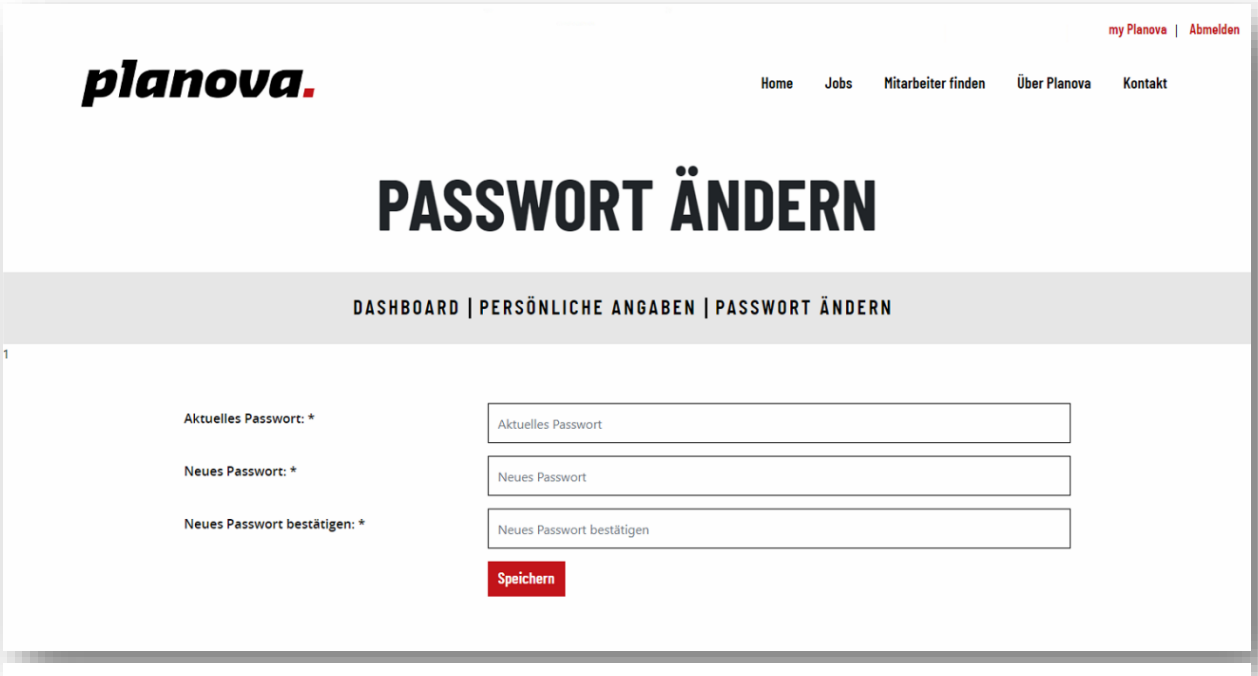


Abbildung 19: Passwort ändern

Abbildung 20: Passwort ändern (Smartphone)

MitarbeiterFinden.php (eingeloggt):

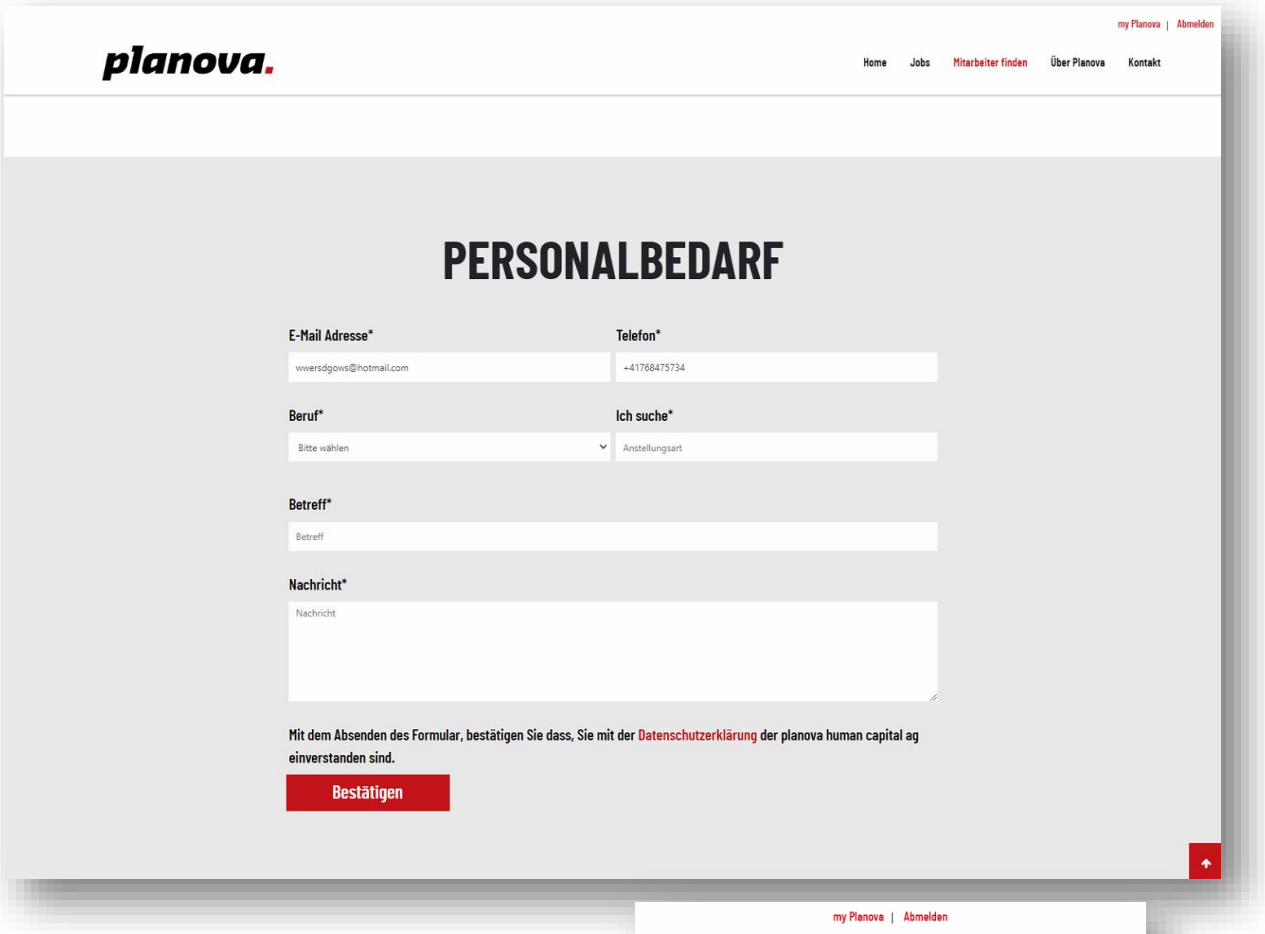


Abbildung 21: Personalbedarf, eingeloggt

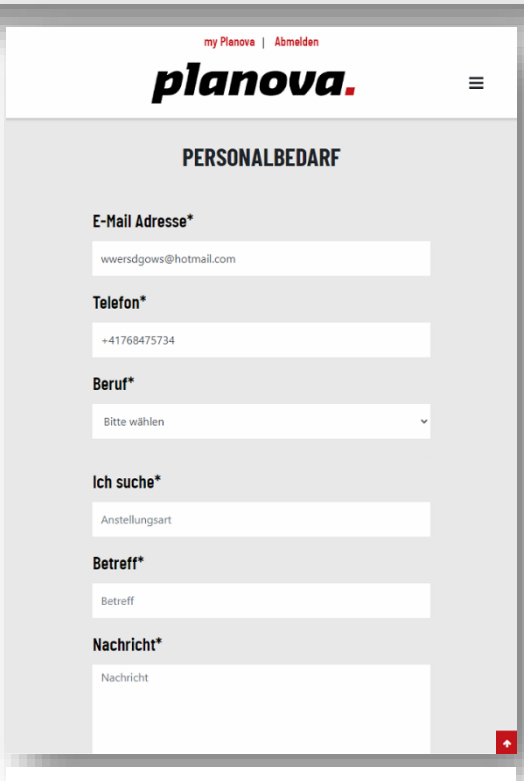


Abbildung 22: Personalbedarf, eingeloggt (Smartphone)

MitarbeiterFinden.php (ausgeloggt):

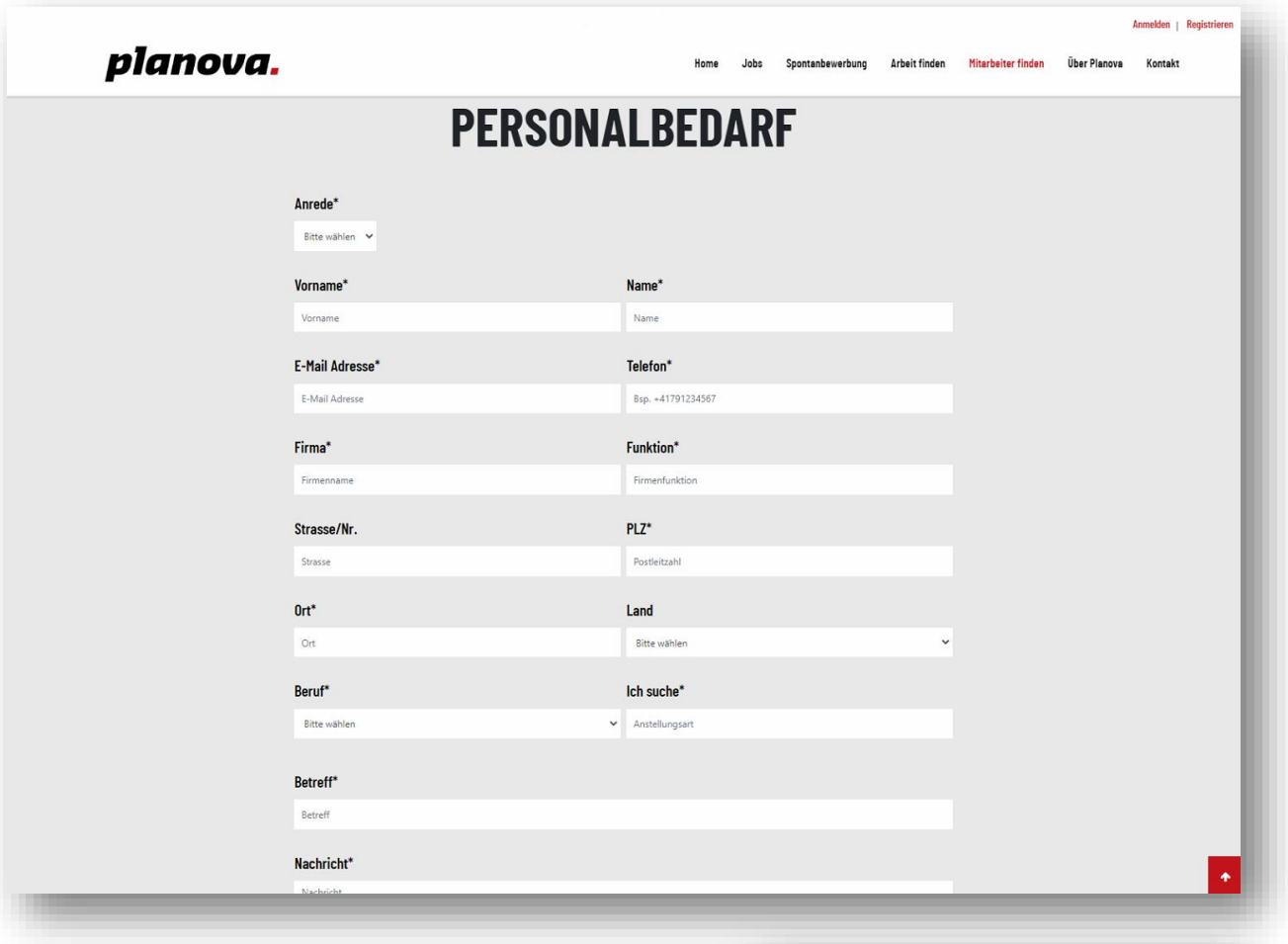


Abbildung 23: Personalbedarf, ausgeloggt

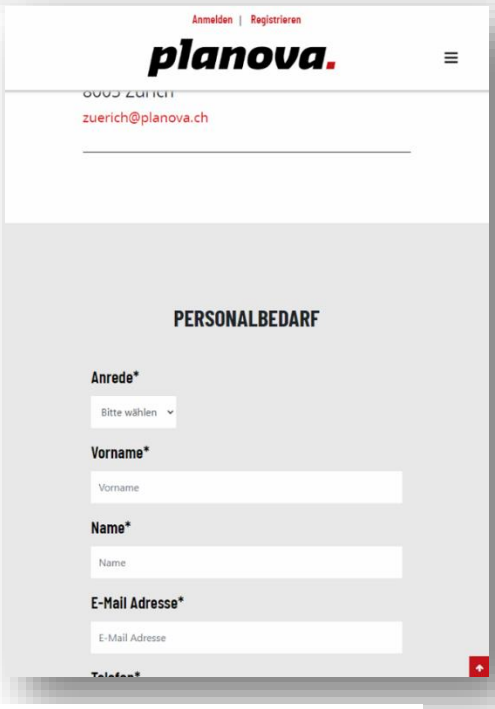


Abbildung 24: Personalbedarf, ausgeloggt (Smartphone)

2.1 Kontrollieren

2.1.1 Testumgebung

Die Hardware, welche für die manuellen Tests verwendet werden, ist ein PC mit einer aktuellen Version von Windows 10. Die Tests werden auf einem lokalen durchgeführt und in der Testdokumentation festgehalten.

2.1.2 Testmethode

Für die Tests verwende ich das Black-Box-Test Prinzip gewählt. Der Sinn dahinter ist, dass der Aufbau des Codes in diesem Fall irrelevant ist. Es wird nur geprüft, ob sich die Oberfläche richtig verhält.

2.1.3 Testdokumentation

In diesen Tests wird die Funktionalität der Mandantenfähigen Webseite geprüft.

Testfall-Nr:	**1**
Anforderung:	Der User kann sich per Mail registrieren
Beschreibung:	Er sollte in der Lage sein, sich von der Verlinkung „Registrieren | Anmelden“ per E-Mail registrieren zu können. Sobald er dies getan hat sollte er automatisch eingeloggt sein.
Voraussetzung:	- E-Mail besitzen.
Erwartetes Resultat:	Der User ist registriert und eingeloggt, es sollte ein Datenbankverzeichnis erstellt.
Ergebnis:	Erwartetes Resultat.

Testfall-Nr:	**2**
Anforderung:	Der User kann sich per Telefon registrieren
Beschreibung:	Er sollte in der Lage sein, sich von der Verlinkung „Registrieren | Anmelden“ per Telefon registrieren zu können. Sobald er dies getan hat sollte er automatisch eingeloggt sein.
Voraussetzung:	- Telefonnummer besitzen
Erwartetes Resultat:	Der User ist registriert und eingeloggt, es sollte ein Datenbankverzeichnis erstellt.
Ergebnis:	Erwartetes Resultat.

Testfall-Nr:	3
Anforderung:	Der User kann sich ein- und ausloggen per Mail
Beschreibung:	Er sollte in der Lage sein, sich von der Verlinkung „Registrieren | Anmelden“ einloggen zu können.
Voraussetzung:	- Der User ist registriert via E-Mail.
Erwartetes Resultat:	Der User soll sich mit den korrekten Daten einloggen können und zum Dashboard gelangen.
Ergebnis:	Erwartetes Resultat

Testfall-Nr:	4
Anforderung:	Der Benutzer kann sich ein- und ausloggen per Telefon
Beschreibung:	Er sollte in der Lage sein, sich von der Verlinkung „Registrieren | Anmelden“ einloggen zu können.
Voraussetzung:	- Der User ist registriert via Mobile.
Erwartetes Resultat:	Der User soll sich mit den korrekten Daten einloggen können und zum Dashboard gelangen.
Ergebnis:	Erwartetes Resultat

Testfall-Nr:	5
Anforderung:	Der Benutzer kann das Personalbedarf-Formular als Gast abschicken und wird anschliessend aufgerufen sich optional zu registrieren.
Beschreibung:	Der Benutzer sollte in der Lage sein das Formular auszufüllen und abzusenden als Gast.
Voraussetzung:	- Der User ist nicht eingeloggt
Erwartetes Resultat:	Das Formular des Users sollte weitergeleitet werden an die Zweigstelle, die Nachricht erwähnt, dass das Formular von einem Gast zugesendet wurde und es sollte ein Datenbankverzeichnis erstellt werden. Der Benutzer bekommt eine Übersicht seiner Übermittlung angezeigt samt der Option sich zu registrieren.
Ergebnis:	Erwartetes Resultat

Testfall-Nr:	6
Anforderung:	Der User kann das Personalbedarf-Formular abschicken wenn er eingeloggt ist.
Beschreibung:	Das Formular erhält weniger Textfelder als die von dem Gast. Das übermittelte Formular sollte weitergeleitet werden an die Zweigstelle, die Nachricht erwähnt, dass das Formular von einem registrierten Kunden zugesendet wurde und es sollte ein Datenbankverzeichnis erstellt werden. Der Benutzer bekommt eine Übersicht seiner Übermittlung angezeigt samt der Option sich zu registrieren.
Voraussetzung:	- Der User ist bereits registriert und eingeloggt.
Erwartetes Resultat:	Das Formular des Users sollte weitergeleitet werden an die Zweigstelle und ein Datenbankverzeichnis soll erstellt werden. Er wird dazu aufgerufen sich optional zu registrieren.
Ergebnis:	Erwartetes Resultat

Testfall-Nr:	5
Anforderung:	Der eingeloggte User soll in seinem „my Planova“ Bereich die Möglichkeit haben seine Kontakt- und persönlichen Angaben zu ändern.
Beschreibung:	Der Benutzer sollte in der Lage sein in seinem Profil (my Planova Bereich) seine persönlichen Angaben samt Kontaktdaten zu ändern. Die Kontaktangaben beziehen sich hierbei nicht auf die Daten um sich anzumelden. Sobald er alle nötigen Felder geändert hat drückt er auf „Speichern“.
Voraussetzung:	- Der User ist registriert und eingeloggt.
Erwartetes Resultat:	Die geänderten Kontakt- und persönlichen Angaben werden übernommen und in der Datenbank verzeichnet. Die Login Methode wird hierbei keineswegs geändert.
Ergebnis:	Erwartetes Resultat.

Testfall-Nr:	6
Anforderung:	Der User sollte nicht in der Lage sein, nach der Übermittlung des Personalbedarfs per Ctrl + R das Formular dauerhaft zu übermitteln.
Beschreibung:	Nach einer Übermittlung des Personalbedarfs sollte der Benutzer bei einem Versuch, das Formular nochmals zu bestätigen, eine Spamschutz Warnung erhalten. Das nochmals bestätigte. Formular sollte dann nicht an die Zweigstellte übermittelt werden.
Voraussetzung:	- User übermittelt schnell und mehrmals das Personalbedarfsformular
Erwartetes Resultat:	Der Benutzer bekommt eine Warnung für Verdacht auf Spam und kann die Seite neu laden um anschliessend noch ein Personalbedarf abzusenden.
Ergebnis:	Erwartetes Resultat.

Testfall-Nr:	7
Anforderung:	Der User soll in seinem „my Planova“ Bereich die Möglichkeit haben sein Passwort zu ändern.
Beschreibung:	Der Benutzer sollte in der Lage sein in seinem Profil (my Planova Bereich) sein Passwort ändern zu können. Dazu sollte er unter dem Reiter „Passwort ändern“ sein altes Passwort eingeben und ein neues mit jeweils zwei Feldern bestätigen.
Voraussetzung:	- User gibt neue Angaben im „my Planova“ Bereich an und drückt auf speichern.
Erwartetes Resultat:	Das alte Passwort wird mit dem neu eingegebenen ersetzt. Es sollte nicht mehr möglich sein sich mit dem alten Passwort einzuloggen, nur noch mit dem neuen.
Ergebnis:	Erwartetes Resultat.

Testfall-Nr:	8
Anforderung:	Der User wird beim Ausfüllen des Formulars darauf hingewiesen, dass er falsche Zeichen bei Feldern verwendet
Beschreibung:	User gibt Felder falsch ein.
Voraussetzung:	- User füllt Felder falsch aus
Erwartetes Resultat:	Wenn unerlaubte Zeichen eingesetzt werden muss der User, bevor er das Formular übermitteln kann, benachrichtigt werden, dass unerlaubte Zeichen vorhanden sind, welche es zu bereinigen gibt.
Ergebnis:	Erwartetes Resultat

Testfall-Nr:	9
Anforderung:	User gibt die PLZ im Formular an und das Feld Ort wird automatisch ausgefüllt.
Beschreibung:	Von der PLZ sollte das System nachvollziehen können, welcher Ort einzusetzen ist. Dieser wird dann eingesetzt, aber soll noch änderbar sein.
Voraussetzung:	- User füllt das Feld PLZ aus.
Erwartetes Resultat:	Der User soll der passende Ort automatisch ausgefüllt werden sobald die Postleitzahl eingegeben wurde.
Ergebnis:	Fehlgeschlagen, siehe Punkt „Autocomplete“ unter Realisieren.

2.2 Auswerten

2.2.1 Fazit

Das Projekt konnte erfolgreich durchgeführt werden. Alle Punkte, welche in der Aufgabenstellung definiert waren, konnten umgesetzt werden bis auf den Punkt „Autocomplete“.

In den Phasen: Informieren, Planen und Entscheiden, verlief alles wie geplant und es gab keine grossen Abweichungen.

In der Realisierungs-Phase fanden etwas grössere Abweichungen im Zeitplan statt. Trotz dessen lief die Realisierung in Ordnung und es kam bis auf den Punkt „Autocomplete“ zu keinen sonderlichen Hindernissen.

Die Planung hatte eine sehr wichtige Bedeutung. Durch die Teilaufträge in der Planung wie das Anwendungsfalldiagramm, Datenbankmodell und das MVC-Konzept konnte ich mir einen Überblick über das Projekt verschaffen. Somit konnte ich mir viele Überlegungen und daraufhin ebenfalls viel Zeit sparen.

Beim Testprotokoll habe ich mir genug Zeit eingeplant. Mir sind während der Umsetzung viele weitere Testfälle nach und nach eingefallen. Hätte ich mehr Zeit darin geplant und investiert, hätte die Zeit anderer Tätigkeiten hierfür nicht verwendet werden müssen.

Es hat sich gelohnt, so viel Zeit in der Erarbeitung des Berichts einzuplanen. Mir war bewusst, dass die Erarbeitung viel Zeit kosten würde. Dies spricht für den endgültigen Stand des Berichts.

Positiv war ebenfalls das ich die IPA Phase mit selbstständigem arbeiten absolvieren konnte.

Somit war keine sonderliche Hilfe notwendig.

Ich bin zufrieden und stolz auf die Arbeit. Mit dem Projekt konnte ich wieder viele Erfahrungen sammeln. Ich konnte mein Wissen nochmals durchaus erweitern und kann behaupten eine individuelle Praktische Arbeit durchgeführt zu haben.

Ich danke Thiru Siva für die Unterstützung während und vor der Arbeit und ich danke Herr André Lichtsteiner für die Tipps und Besuche.

3 Quellenverzeichnis

Medium	Information (Autor, Titel)	Quelle (Link oder ISBN)	Datum
Internet	Laravel	https://laravel.com/docs/9.x/queries#raw-expressions	18.05.2022
Internet	microTOOL	https://www.microtool.de/wissen-online/was-sind-use-cases/	10.05.2022
Internet	php.net	https://www.php.net/manual/de/function.preg-match.php	17.05.2022
Internet	PkOrg	https://www.pkorg.ch	09.05.2022
Internet	w3schools	https://www.w3schools.com/	11.05.2022
Internet	Wikipedia	https://de.wikipedia.org/wiki/Model_View_Controller#:~:text=Model%20View%20Controller%20(MVC%2C%20englisch,und%20Programmsteuerung%20(englisch%20controller).	10.05.2022

3.1 Abbildungsverzeichnis