

# IPA ABSCHLUSSARBEIT 2022

## BEERES-DOKUMENTATION

Christopher O'Connor

# 1 INHALTSVERZEICHNIS

---

<b>2 AUFGABENSTELLUNG.....</b>	<b>3</b>
2.1 AUSGANGSLAGE .....	3
2.2 DETAILLIERTE AUFGABENSTELLUNG .....	3
<b>3 PROJEKTAUFBAUORGANISATION .....</b>	<b>9</b>
3.1 BETEILIGTE PERSONEN .....	9
3.2 ROLLEN .....	10
3.3 ARBEITSUMGEBUNG.....	10
<b>4 ORGANISATION DER ARBEITSERGEBNISSE .....</b>	<b>10</b>
4.1 QUELLCODE.....	10
4.2 DOKUMENTATION.....	10
<b>5 FIRMENSTANDARDS .....</b>	<b>10</b>
<b>6 MITTEL UND METHODEN .....</b>	<b>11</b>
6.1 TECHNOLOGIEN.....	11
6.2 SOFTWARE .....	11
6.3 LARAVEL LIBRARIES .....	11
6.4 PROJEKTMANAGEMENT METHODE.....	11
<b>7 VORKENNTNISSE.....</b>	<b>11</b>
<b>8 VORARBEITEN .....</b>	<b>12</b>
<b>9 NEUE LERNINHALTE .....</b>	<b>12</b>
<b>10 ARBEITEN IN DEN LETZTEN 6 MONATEN .....</b>	<b>12</b>
<b>11 INDIVIDUELLE KRITERIEN .....</b>	<b>12</b>
<b>12 ZEITPLAN .....</b>	<b>14</b>
<b>13 ARBEITSJOURNAL .....</b>	<b>15</b>
13.1 FREITAG, 11.03.2022 .....	15
13.2 MONTAG, 14.03.2022 .....	16
13.3 03DIENSTAG, 15.03.2022 .....	17
13.4 DONNERSTAG, 17.03.2022 .....	18
13.5 FREITAG, 18.03.2022 .....	20
13.6 MONTAG, 21.03.2022 .....	21
13.7 DIENSTAG, 22.03.2022 .....	23
13.8 DONNERSTAG, 24.03.2022 .....	24
13.9 FREITAG, 25.03.2022 .....	25
13.10 MONTAG, 28.03.2022 .....	26
<b>14 KURZFASSUNG .....</b>	<b>27</b>
14.1 KURZE AUSGANGSSITUATION .....	27
14.2 UMSETZUNG.....	27
14.3 ERGEBNIS.....	27
<b>15 INFORMIEREN .....</b>	<b>28</b>
15.1 EXTERNE SCHNITTSTELLEN .....	28
<b>16 PLANEN.....</b>	<b>30</b>
16.1 AUFTRAGSDIVISION .....	30

16.2	PROJEKTDATENSTRUKTUR.....	30
16.3	MOCKUPS .....	31
16.4	ERD DATENBANKMODELLE .....	35
16.5	SEQUENZDIAGRAMM.....	37
16.6	REALISIERUNGSKONZEPT .....	38
16.7	TESTKONZEPT .....	39
16.8	ERKANNTEN RISIKEN .....	61
<b>17</b>	<b>ENTSCHEIDEN .....</b>	<b>62</b>
17.1	MASSNAHMEN DER RISIKEN.....	62
17.2	DATENBANKMODELL .....	62
17.3	IMKERSUCHE.....	62
<b>18</b>	<b>REALISIEREN .....</b>	<b>63</b>
18.1	SETUP .....	63
18.2	DATENBANK MIGRATION.....	64
18.3	MODELS ERSTELLEN .....	65
18.4	FACTORY & SEEDER .....	65
18.5	USER AUTHENTIFIKATION.....	69
18.6	IMKER REGISTRIERUNG .....	70
18.7	MIDDLEWARE .....	77
18.8	PROFILE .....	79
18.9	ZUSTÄNDIGKEIT (IMKER PLZ).....	80
18.10	S&R AUFTRAGSERSTELLUNG .....	88
18.11	AUFTRAGS-SMS NOTIFIKATION .....	94
18.12	IMKER ZUSAGE.....	97
18.13	AUFTRAGS-SMS BESTÄTIGUNG .....	99
18.14	IMKER-SEARCH .....	100
<b>19</b>	<b>KONTROLIEREN .....</b>	<b>106</b>
19.1	TESTPROTOKOLL 01 .....	106
19.2	TESTPROTOKOLL 02 .....	108
<b>20</b>	<b>AUSWERTEN .....</b>	<b>109</b>
<b>21</b>	<b>SCHLUSSWORT.....</b>	<b>109</b>
21.1	ALLGEMEIN .....	109
21.2	INFORMIEREN .....	109
21.3	PLANUNG.....	109
21.4	REALISIERUNG.....	109
21.5	TESTEN.....	110
21.6	PERSÖNLICHE BILANZ .....	110
<b>22</b>	<b>GLOSSAR.....</b>	<b>111</b>
<b>23</b>	<b>LITERATURVERZEICHNIS.....</b>	<b>112</b>
<b>24</b>	<b>ABBILDUNGSVERZEICHNIS .....</b>	<b>113</b>
<b>25</b>	<b>ANHANG .....</b>	<b>116</b>
25.1	RESULTAT .....	116
25.2	QUELLCODE.....	130

# Teil 1: Umfeld und Ablauf

## 2 AUFGABENSTELLUNG

---

### 2.1 AUSGANGSLAGE

Die Einsatzzentrale der Schutz & Rettung erhält pro Jahr mehrere hundert Anrufe bezüglich Bienenschwarm-Sichtungen. Derzeit bestehen in den meisten Regionen im Kanton Zürich keine Prozesse, welche bei einem solchen Anruf ausgelöst werden können, um den Schwarm fachmännisch entfernen zu lassen und damit die Gefahr für den Menschen zu minimieren.

### 2.2 DETAILLIERTE AUFGABENSTELLUNG

Imker sollen sich bei einer zentralen Web-Plattform als freiwillige Helfer für das Einsammeln von Bienen-Schwärmen registrieren und anmelden können. Die Imker können Regionen angeben, für die sie bereit sind, die Zuständig zu übernehmen. Mehrere Imker können sich dabei für die gleichen Regionen eintragen. Wird ein Schwarm der Einsatzleitzentrale von Schutz & Rettung gemeldet, werden über eine generische SMS alle Imker für die betroffene Region alarmiert. Via Link kann nach dem Prinzip «first come, first served» ein Imker eine bindende Zusage für die Abholung des Schwarms machen. Daraufhin wird dem Imker die genaue Adresse des Schwarms automatisiert übermittelt und damit implizit der Auftrag zur Beseitigung des Schwarms übertragen. Sollte sich innerhalb einer Stunde kein Imker für das Einfangen des Schwarms gemeldet haben, werden ein Erinnerungs-SMS ausgelöst und zusätzliche Imker in angrenzenden Bezirken alarmiert. Beliebige Personen können über dieselbe Web-Plattform einen Imker in Ihrer Region suchen und finden (Funktion "Imker-Search" als Guest). Die Plattform muss "selbst-wartend" sein und soll keines manuellen Aufwands bedürfen (alle Prozesse laufen voll automatisiert).

#### 2.2.1 Das Wichtigste

Das Onlineportal für die Einsammlung von Bienen-Schwärmen ist als Web-Applikation zu erstellen. Die Web-Applikation soll einfach und benutzerfreundlich aufgebaut sein, d.h. dass:

1. Die Benutzer die Applikation ohne Instruktion bedienen können
2. Die GUIs / die Seiten übersichtlich gestaltet sind (= eindeutig beschriftete und klar angeordnete Buttons, gut lesbare Schrift)
3. Die Applikation in sämtlichen gängigen Browsern (mindestens Chrome und Firefox) gleich funktioniert und gleich/ähnlich aussieht
4. Falsche Benutzer-Eingaben abgefangen, eine aussagekräftige Fehlermeldung angezeigt und/bzw. die Eingabefelder rot markiert werden (siehe auch "Eingabeprüfungen" weiter unten)
5. Die jeweils vorgesehene Aktion erst ausgeführt werden kann, wenn keine fehlerhaften Benutzer-Eingaben mehr vorhanden sind (insbesondere bei der Registrierung und dem Login). Anmerkung: Eingabeprüfungen müssen immer sowohl client- als auch serverseitig erfolgen.

## 2.2.2 Rollen

### 2.2.2.1 Imker

Interessierte Imker können sich am System registrieren. Registrierte Imker können Regionen (PLZ) auswählen, sowie ihre Personalien mutieren. Sie werden via SMS von der Applikation über einen möglichen Auftrag informiert (Übereinstimmung der Zuständigkeit für die betreffende Region), welche sie annehmen können - nach dem Prinzip «first come, first served» (siehe auch oben). Wichtig: Einmal angenommen, ist der Auftrag bindend.

### 2.2.2.2 Personal von Schutz & Rettung

Die Mitarbeitenden von Schutz & Rettung besitzen ein Login/Konto mit speziellen Rechten, welches bereits im System vorab erfasst ist (ihr übliches Firmen-Login). Die Schutz & Rettung Mitarbeitenden lösen die Imker-Aufträge aus (S-5).

### 2.2.2.3 Gast

Ein beliebiger, nicht registrierter Benutzer kann die Imkersuche (Funktion "Imker-Search") verwenden, um Imker anhand einer einzugebenden PLZ in der betreffenden Region aufzufinden zu machen (vgl. oben). Ein Gast muss sich nicht einloggen.

## 2.2.3 User Interface (UI)

Die wesentlichen Applikationsoberflächen sind:

- Login (Imker, MA Schutz & Rettung) / Imker-Search (Gast)
- Registrierung (Imker)
- Konto bearbeiten, Regionen erfassen (Imker)
- Auftragserstellung / Einsammeln des betreffenden Bienen-Schwärms auslösen (MA Schutz & Rettung)

## 2.2.4 User Stories

#nr	Story	Beschreibung
S-0	Imker-Registrierung	Als Imker möchte ich mich registrieren, um Aufträge erhalten zu können.
S-1	Imker-Zuständigkeiten	Als Imker möchte ich Regionen (PLZ) hinterlegen, für die ich bereit bin, die Zuständig zu übernehmen / Aufträge anzunehmen.
S-2a	Imker-Anmeldung	Als Imker möchte ich mich bei meinem Konto anmelden können.
S-2b	S&R-Anmeldung	Als Mitarbeiter/r von Schutz & Rettung möchte ich mich am System anmelden können (mit meinem üblichen Login / Passwort).
S-3	Imker-Daten mutieren	Als Imker möchte ich meine eingegebenen Daten (Profil) verändern können.
S-4	Imker-Registrierung löschen	Als Imker möchte ich mein Konto/Profil löschen, um nicht mehr über neue Aufträge benachrichtigt zu werden.
S-5	S&R - Auftragserstellung	Als Mitarbeiter von Schutz & Rettung übergebe ich alle relevanten Informationen an die Web-Applikation, um das Einsammeln des betreffenden Bienen-Schwärms (= Auftrag) auszulösen (Stories: S-6, S-7, S-8 und nötigenfalls S-9).
S-6	Auftrags-Notifikation	Die Web-Applikation ermittelt (mittels Geoinformationen aus Story S-5) alle eingetragenen Imker der betroffenen Region des Einsatzortes und übermittelt eine Notifikation

		via SMS [= Auftragsanfrage; die genaue Adresse des Schwarms wird erst nach Zusage übermittelt (Stories: S7 und S-8)].
S-7	Imker-Zusage	Als Imker erhalte ich einen Link per SMS über einen möglichen Auftrag in meiner (oder einer benachbarten) Region. Damit kann ich eine bindende Zusage über die Abholung des Schwarms machen.
S-8	Übermittlung der Einsatz-Adresse	Als Imker erhalte ich nach der bindenden Zusage eine Bestätigung per SMS sowie die genaue Adresse des Einsatzortes (gemäss den Geoinformationen), um diesen Auszuführen.
S-9	SMS-Erinnerung	Sollte kein Imker den Auftrag innerhalb einer Stunde annehmen, wird eine "Erinnerungs-SMS" an den ursprünglichen Empfängerkreis sowie an zusätzliche Imker aus angrenzenden Regionen verschickt (vgl. Stories S-7 und S-6).
S-10	Imker-Search	Als Gast möchte ich einen Imker in meiner Region ausfindig machen können.
S-11	Jährliche Updatenotifikation	Als Imker werde ich jährlich per SMS erinnert, meine Personalien sowie meine hinterlegten Regionen zu aktualisieren.

**Farb-Legende:**

Wird im Rahmen der IPA verlangt.
Wird nicht im Rahmen der IPA verlangt, kann fakultativ realisiert werden.
Wird nicht im Rahmen der IPA realisiert.

**2.2.5 Definitionen**

- Einsatz-Regionen: Postleitzahlen (1 Region = 1 PLZ).  
 Angrenzende Regionen: Werden intern ermittelt: PLZ der Einsatzregion +/- 5, bis mindestens 1 Imker gefunden wird.  
 Geoinformationen: Koordinaten des Einsatzortes, Form: „Breite (Latitude), Länge (Longitude)“, Decimal Degree, z.B. "47.255900, 9.319950" = Schwägalp, Parkplatz.

S-5: Hier werden folgende Informationen erfasst und somit im System gespeichert:

- Die Geoinformationen (Koordinaten) des Einsatzortes
- Die Kontaktperson: Name, Vorname, Telefonnummer
- Allfällige Hinweise (z.B. "im Gartenhäuschen")

S-6/7: Mit Hilfe der Geoinformationen ermittelt die Applikation die betreffende Region (PLZ) und die für diese Region eingetragenen Imker. Die SMS-Notifikation enthält nur die Region (PLZ). So soll verhindert werden, dass verschiedene Imker sofort aktiv werden und "in eigener Regie" den Schwarm direkt einsammeln.

S-7/8: Wenn der Imker den Auftrag angenommen hat, werden ihm die Detailinformationen zum Auftrag per SMS zugesendet. Es sind dies (vgl. S-5):

- Die Geoinformationen (Koordinaten) des Einsatzortes in Form eines "Google Maps Pin"
- Seine Kontaktperson: Name, Vorname, Telefonnummer
- Allfällige Hinweise (z.B. "im Gartenhäuschen")

Der Imker wird in der SMS ferner dazu angehalten, sich wenn immer möglich mit der Kontaktperson in Verbindung zu setzen, bevor er den Bienenschwarm abholt (Details können so geklärt/besprochen werden)

S-7/8: Imker, die sich nach bereits erfolgter Auftragserteilung (zu spät) bei der Applikation melden (von der Applikation muss bei S-7 überprüft werden, ob der Auftrag bereits vergeben wurde), werden auf ein Formular weitergeleitet, auf dem der Auftrag als "bereits vergeben" ausgewiesen wird; S-8 wird in diesem Falle nicht ausgelöst. Wie dies sichergestellt werden kann, soll vom Kandidaten während der IPA ausfindig gemacht und dokumentiert werden.

S-7: Da die Applikation während der IPA nicht in das produktive Umfeld übernommen wird (siehe "Abgrenzung" weiter unten), kann der SMS-Antwort-Link nicht direkt getestet werden (kein Zugriff ab dem Handy auf die lokale Entwicklungsumgebung). Zu Testzwecken wird der Link daher "von Hand" in einen Browser kopiert und kann so lokal getestet werden.

S-10: Die für eine Region registrierten Imker sollen mittels Eingabefeld (= PLZ) gesucht und angezeigt werden.

## 2.2.6 Datenbank

Das Datenbank-ERM ist anhand der User Stories zu erstellen und mit "Laravel Migrations" zu implementieren. Die Testdaten sind ebenfalls mit Laravel zu erstellen ("Seeder").

Imker-Daten:

- Vorname
- Nachname
- E-Mail-Adresse (= Login)
- Telefonnummer (vorzugsweise Mobilnummer)
- Passwort

Regionen-Daten:

- PLZ
- Ortsbezeichnung

### 2.2.7 Eingabeprüfung

1. Sämtliche Eingabefelder sind auf Vorhandensein (Pflichtfelder / z.B. E-Mail-Adresse und Passwort bei der Registrierung) und Gültigkeit (z.B. formale Korrektheit der E-Mail-Adresse und der Telefonnummer bei der Registrierung) zu prüfen.
2. Ungültige Benutzereingaben werden pro Eingabefeld eindeutig gekennzeichnet (z.B. farblich hinterlegt) und mit einer Fehlerinfo (z.B. Tooltip) versehen.
3. Ungültige Kombinationen von mehreren Eingabefeldern (z.B. Login und Passwort beim Login): die betreffenden Eingabefelder werden eindeutig gekennzeichnet (siehe oben); zudem wird eine spezifische Fehlerinfo zur fehlerhaften Eingabekombination ausgegeben.

Detailangaben zur Eingabeprüfung:

Feldbezeichnung	Eingabeprüfung
<b>Registrierung / Konto bearbeiten (Imker)</b>	
Vorname	fak. / keine formale Prüfung
Nachname	Pflicht / keine formale Prüfung
E-Mail-Adresse (= Login)	Pflicht / (nur) ein @ muss vorhanden sein
Telefonnummer (vorzugsweise Mobilnummer)	Pflicht / nur Ziffern und Leerschläge
Passwort	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
<b>Login (Imker, MA Schutz &amp; Rettung)</b>	
E-Mail-Adresse (= Login)	Pflicht / (nur) ein @ muss vorhanden sein
Passwort	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
<b>Imker-Search (Gast)</b>	
PLZ (Dropdown oder direkte Eingabe)	Pflicht / aus Auswahlliste oder 4 Ziffern
<b>Auftragserstellung (MA Schutz &amp; Rettung)</b>	
PLZ (Dropdown oder direkte Eingabe)	Pflicht / aus Auswahlliste oder 4 Ziffern

### 2.2.8 Application Flow

Die Benutzer (Imker / MA Schutz & Rettung / Gast) können erst dann weitere Schritte am System ausführen, wenn sie sich erfolgreich registriert und/oder angemeldet haben.

Imker:

Ein(e) neue(r) Imker(in) wird registriert und auf der Datenbank erfasst, wenn keine fehlerhaften Eingaben mehr vorhanden sind. Nach erfolgreicher Registrierung ist der/die Imker(in) eingeloggt und wird zum GUI "Konto bearbeiten, Regionen erfassen" weitergeleitet. Dies geschieht ebenfalls, wenn sich ein(e) bereits im System vorhandene(r) Imker(in) erfolgreich angemeldet hat.

MA von Schutz & Rettung:

Erfolgreich am System angemeldete MA von Schutz & Rettung werden auf das GUI "Auftragserstellung ..." weitergeleitet.

Gast:

Kann ausschliesslich die Funktion "Imker-Search" benutzen.

### 2.2.9 Testing

Durch geeignete Unit-Tests (Blackbox) ist die Funktionalität der gesamten Applikation (sämtliche zu realisierenden User Stories) sicherzustellen. Im Frontend können mit Selenium Webdriver automatisierte Tests durchgeführt werden (wird nicht zwingend verlangt). Zwingend sind eine repräsentative Testfall-Sammlung und ein vollständiges, separates Test-Protokoll zu erstellen. Spezielle Vorkehrungen sind nicht zu treffen, sie auch folgendes Kapitel "Abgrenzung".

### 2.2.10 Abgrenzung

Die Applikation wird auf einem lokalen Entwicklungssystem erstellt. Die Installation und Inbetriebnahme eines Web-/Mailservers und/oder eines SMS-Dienstes/-APIs sowie das Publizieren der Applikation auf dem Zielsystem des Kunden ist nicht Bestandteil der IPA.

### 2.2.11 Spezialitäten / Fokus

Bei der Entwicklung der Applikation liegt das Schwergewicht bei den "SMS Notifications" (S-6 bis S8 (S-9)), der Ermittlung der betreffenden Imker (S-6, (S-9,) S-10) sowie dem Handling der ImkerZusagen (S-7/8). Aus Sicht des Auftraggebers wird deren Arbeitsprozess massiv automatisiert (entlastet).

### 2.2.12 Vorbedingungen

1. Der Lernende hat eine vollständig aufgesetzte, lokale Entwicklungsumgebung.
2. Die Entwicklungsumgebung beinhaltet im Wesentlichen das Framework Laravel, GitLab (Projekt auf SANTIS Training Server), MariaDB sowie die weiteren, unter "Mittel und Methoden" genannten Technologien und Software.
3. Der Lernende hat Kenntnisse in PHP, SQL, JavaScript, HTML, CSS (inkl. Bootstrap), Git.

## 3 PROJEKTAUFBAUORGANISATION

---

### 3.1 BETEILIGTE PERSONEN

**Berufsbildner/ Lehrfirma**

**Verantwortliche Fachkraft**

**Hauptexperte**

**Nebenexperte**

**Kandidat**

## 3.2 ROLLEN

Rolle	Aufgaben / Verantwortung
<b>Hauptexperte</b>	Der Hauptexperte (HEX) ist verantwortlich für die Festlegung der Termine (Besuch kurz nach Start der Arbeit, zweiter Besuch gegen Ende der Arbeit sowie Präsentation und Fachgespräch). Er entscheidet bei auftretenden Problemen (Lieferverzögerung, Krankheit, ...) über Massnahmen (Verlängerung der Arbeit, Änderung der Aufgabenstellung, ...) und meldet diese unverzüglich dem Chefexperten. Er studiert den Bericht und leitet das Fachgespräch.
<b>Nebenexperte</b>	Wird ein Nebenexperte (NEX) aufgeboten, ist dieser nur bei der Präsentation, dem Fachgespräch und bei der Bewertung dabei. Er sorgt für ausführliche Notizen zum Ablauf dieses Prüfungsteils und unterstützt den Hauptexperten beim Bewerten der Arbeit.
<b>Verantwortliche Fachkraft</b>	Die Verantwortliche Fachkraft erstellt die Aufgabenstellung und begleitet den Kandidaten während der IPA. Er beurteilt das Arbeitsverhalten sowie die Arbeit des Kandidaten.
<b>Kandidat</b>	Der Kandidat führt die IPA anhand der detaillierten Aufgabenstellung aus und wird anhand der Beurteilungskriterien bewertet.

(Prüfungskommision 19, 2022)

## 3.3 ARBEITSUMGEBUNG

Als primärer Arbeitsplatz dient der normale Arbeitsplatz bei der Santis Training AG. Vereinzelte Tage im Home-Office können möglich sein.

# 4 ORGANISATION DER ARBEITSERGEBNISSE

---

## 4.1 QUELLCODE

Der Quellcode wird in einem Git-Repository auf dem Santis Training AG verwendeten Gitlab nach Erreichung von Meilensteinen / Teilzielen jeweils hochgeladen und somit versioniert.

## 4.2 DOKUMENTATION

Sämtliche Dokumente, welche nicht im Git abgelegt sind (Dokumentation, Zeitplan, etc.), werden im OneDrive in der Cloud gespeichert. Zusätzlich wird am Tagesende jeweils die Dokumentation als Backup abgelegt. Beim Abschluss einer IPERKA Stufe wird jeweils eine Versionierung erstellt.

# 5 FIRMENSTANDARDS

---

Die Santis Training AG hat keine Standards, welche in dieser Arbeit verwendet werden.

## 6 MITTEL UND METHODEN

---

### 6.1 TECHNOLOGIEN

TECHNOLOGIE	BESCHREIBUNG
Programmiersprachen	PHP (7.4.15), JS, (HTML5, CSS3)
Framework	Laravel (8.x)
Datenbank	MariaDB (10.4.17)
Object-Relational-Mapper (ORM)	Eloquent
Libraries	Bootstrap, jQuery (3.6)
Templating-Engine	Blade Templates
SMS-API	Vonage (Nexmo)
Geocoding API	Nominatim

### 6.2 SOFTWARE

NAME	VERWENDUNGSZWECK
XAMPP (3.2.4)	Webserver & Datenbank
PhpStorm (2021.3.1)	IDE
Composer (2.2.5)	Laravel Packet-Manager
Git	Versionskontrollsystem
OneDrive	Dokumentversionskontrollsystem und Backup
MySQL Workbench	ERM Erstellung
phpMyAdmin	Datenbankmanagement
Microsoft Word	Dokumentation

### 6.3 LARAVEL LIBRARIES

NAME	VERWENDUNGSZWECK
Laravel Debugbar	Debugging
Eloquent Model Generator	Basis Models generieren
Nexmo SMS-Client	SMS-API

### 6.4 PROJEKTMANAGEMENT METHODE

IPERKA ist die für diese IPA verwendete Projektmanagement Methode.

## 7 VORKENNTNISSE

---

- Grund bis gute Kenntnisse in SQL, PHP, HTML, CSS (inkl. Bootstrap), JavaScript und Git.
- Vertiefte Kenntnisse in der objektorientierten Programmierung (Java) sowie Laravel
- Überbetriebliche Kurse und Informatikmodule an der Berufsschule (TBZ & BBB) besucht
- ECDL Advanced Zertifikat (Word)

## 8 VORARBEITEN

---

- Entwürfe für mögliche Wireframes / Mockups
- User Stories

## 9 NEUE LERNINHALTE

---

- Vonage (Nexmo) SMS-API
- Laravel externe API abfrage
- Geoinformationen von S&R
- Mailtrap
- Nominatim

## 10 ARBEITEN IN DEN LETZTEN 6 MONATEN

---

Vertiefungsarbeit Berufsschule:

Arduino Drohne selber bauen & programmieren (C/C++) inkl. Autonomes Schweben (IMU, Komplementärfilter & PID-Regler).

Projekt SantisQuiz:

Laravel basierte Quiz-Webapplikation inkl. Teilnehmerverwaltung, Fragebogenstatistiken und Auswertungen sowie E-Mail-Benachrichtigungen.

## 11 INDIVIDUELLE KRITERIEN

---

<b>Leitfrage 1</b>	<b>MVC (Programmierung)</b> Ist die Auftrennung nach dem MVC-Pattern konsequent durchgeführt und sind Abweichungen vom Pattern beschrieben und begründet?
Gütestufe 3	M: konsequent nur Datenaufbereitung für GUI V: konsequent nur Darstellung (GUI) C: konsequent nur Ablaufsteuerung und Validierung Realisierung eingehalten? Sind die Schnittstellen im Code klar ersichtlich? Alle 4 Aspekte erfüllt

<b>Leitfrage 2</b>	<b>Plausibilisierung der Benutzer-Eingaben</b> 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.

<b>Leitfrage 3</b>	<b>Codingstyle - lesbarer Code</b> Ist der Code lesbar geschrieben, gut gegliedert und ist die Namensgebung gut gewählt?
Gütestufe 3	Die Namensgebung entspricht den Vorgaben oder ist einfach gut gewählt. Die Struktur des Codes ist ebenfalls gemäss möglicher Richtlinien oder einfach übersichtlich gemacht. Es ist eine gewisse Einheit zu sehen in der Art und Weise, wie der Code strukturiert ist (d.h. es ist überall etwa gleich gemacht).

<b>Leitfrage 4</b>	<b>Software-Ergonomie</b> Ist die nötige Benutzerfreundlichkeit/SW-Ergonomie implementiert?
Gütestufe 3	GUI-Elemente sind <ul style="list-style-type: none"> <li>- der Funktion entsprechend gewählt</li> <li>- verständlich beschriftet</li> <li>- sinnvoll gruppiert und</li> <li>- korrekt angewendet.</li> </ul>

<b>Leitfrage 5</b>	<b>Vollständiges ERM bzw. Datenmodell</b> Ist das ERM bzw. Datenmodell vollständig dargestellt?
Gütestufe 3	1. Alle Entitäten und Beziehungen sind korrekt dargestellt. 2. Alle Assoziationstypen (1, c, m, mc) sind korrekt eingetragen. 3. Alle Primärschlüssel und Fremdschlüssel sind als solche erkenntlich, also entsprechend markiert bzw. bezeichnet. 4. Alle Attributlisten sind vollständig, die Datentypen aller Attribute sind angegeben.

<b>Leitfrage 6</b>	<b>Erfüllung der Funktionen (Applikation)</b> Sind alle geforderten Funktionen realisiert?
Gütestufe 3	Speziell zu erwähnen (siehe Aufgabenstellung): <ul style="list-style-type: none"> <li>- User Stories (S1 bis S11)</li> <li>- Rollentrennung (Imker, MA Schutz &amp; Rettung, Gast)</li> <li>- Application Flow</li> </ul> Alle Funktionen sind komplett und korrekt implementiert und gut benutzbar. Alle Fehleingaben für vom Benutzer eingegebene Daten, vor allem für solche, welche dann auf Schnittstellen ausgetauscht werden müssen, sind abgefangen.

<b>Leitfrage 7</b>	<b>Kommentare im Quellcode</b> Wurde der Sourcecode der Applikation ausreichend kommentiert?
Gütestufe 3	Der Sourcecode der Applikation ist vollumfänglich kommentiert: <ol style="list-style-type: none"> <li>1. Funktionen, Parameter, Rückgabewerte,</li> <li>2. Wichtige Stellen im Sourcecode,</li> <li>3. weitere zusätzliche/nützliche Kommentare.</li> </ol>

## 12 ZEITPLAN

Mithilfe der detaillierten Aufgabenstellung und der Auftragsdivision wurde der Zeitplan erstellt. Er ist in 2-Stundenblöcke unterteilt und nach IPERKA orientiert. Die Soll-Zeit ist als blau und die Ist-Zeit als violett eingefärbt. Die externen Schnittstellen sind bewusst im Abschnitt informieren eingeplant, da diese als neue Lerninhalte in der Aufgabenstellung definiert sind.

Abbildung 1 Zeitplan

Arbeiten	Aufwand (h)		Fr 11/03/2022	Mb 14/03/2022	Di 15/03/2022	Do 17/03/2022	Fr 18/03/2022	Mb 21/03/2022	Di 22/03/2022	Do 24/03/2022	Fr 25/03/2022	Mb 26/03/2022
	Soll	Ist										
Aufgabenstellung & Kriterienkatalog	1.0	1.0										
Externe Schnittstellen	1.5	1.5										
Auftragsdivision	1.0	1.0										
Zeitplan erstellen	1.5	2.0										
Risiken analysieren	1.0	1.0										
Sequenzdiagramm	1.0	1.0										
Mockups	1.5	1.5										
Datenmodell ERD	2.0	2.5										
Testkonzept	2.0	4.0										
Lösungsvarianten & Massnahmen	1.0	0.5										
Laravel Setup	1.5	1.0										
Datenbank erstellen	1.5	0.5										
Models & Seeder	2.0	2.0										
Auth (Login/Logout)	1.0	1.5										
Imker Registrierung	2.0	1.5										
Zuständigkeit (Imker PLZ)	3.0	4.0										
S&R Auftragserstellung	2.0	4.0										
Auftrags-Notifikation	2.5	4.0										
Imker Zusage	2.0	3.0										
Auftragsbestätigung	1.0	1.0										
Imker Search	2.0	3.0										
Automatisierte Tests erstellen	3.0	0.0										
Testfälle Testen	3.0	1.5										
Reflexion & Fazit	4.0	2.0										
Dokumentation	30.0	33.5										
Arbeitsjournal	4.0	6.0										
Zeitresevren	4.0	0.0										
Expertenbesuche	2.0	2.0										
Total	84.0	86.5										
			Sollzeit		Istzeit		2 Stundenblöcke					

## 13 ARBEITSJOURNAL

### 13.1 FREITAG, 11.03.2022

#### **Arbeiten**

Analyse der Aufgabenstellung & Kriterienkatalog  
 Externe Schnittstellen analysiert / getestet  
 IPA Dokumentation Teil 1 erstellt  
 Auftragsdivision  
 Zeitplan erstellt

#### **Erfolge**

Die neuen Lerninhalte (externe Schnittstellen) sind sehr gut dokumentiert und könnten sehr rasch ausprobiert werden. Für die Implementierung sehe ich keine Hindernisse.  
 Der erste Teil der Dokumentation wurde erstellt sowie das Informieren abgeschlossen.

#### **Misserfolge**

Für die aufgetretenen Probleme müssen alternativen analysiert sowie über das weitere Vorgehen entschieden werden.

#### **Aufgetretene Probleme**

Die Reverse Geocoding Request von Nominatim wird als «deprecated» ausgewiesen.  
 Mailtrap: Russland / Ukraine Situation.

#### **Ungeplante Arbeiten**

Git Repository und neues Laravel Projekt erstellt inkl. Bootstrap UI / Auth via Composer: ~10min

#### **Tests:**

##### **Nominatim**

Die Reverse Lookup Request wurde getestet. Dabei wurde die URL mit Beispiel lon/lat werten ergänzt und das JSON return Objekt betrachtet sowie die enthaltende PLZ validiert.

##### **Google Maps Pin**

Das Erstellen eines Pins wurde durch manuelle Anpassung der URL getestet. Die Pins wurden erfolgreich am richtigen Ort der lon/lat location gesetzt.

##### **Vonage**

In einem Testprojekt wurde Vonage überprüft. Dabei wurde nur eine SMS als Test versendet.

##### **Mailtrap**

In einem Testprojekt wurde Mailtrap überprüft. Dabei wurde nur eine E-Mail versendet.

#### **Hilfestellung**

Beim Informieren der Schnittstellen / APIs habe ich mich an die entsprechenden Dokumentationen gerichtet.

#### **Überzeit**

-

#### **Vergleich mit Zeitplan**

Der Zeitplan wurde nach der Auftragsdivision erstellt. Somit bin ich im Zeitplan.

Die externen Schnittstellen befinden sich im Zeitplan unter Informieren, weil sie als neue Lerninhalte definiert sind.

#### **Reflektion**

Die Vorgehensweise des ersten Tags war klar. Zuerst mussten alle notwendigen Informationen betrachtet sowie zusammengetragen werden (Teil 1 Doku). Damit für den Zeitplan die benötigte Zeit, welche für die Implementierung der APIs gebraucht wird, abgeschätzt werden kann, musste ich mich auch über diese zuerst genauer informieren. Dabei habe ich auch gleich grob analysiert, wie und ob sie verwendet werden. Das Testkonzept wurde im Zeitplan provisorisch unter Kontrollieren eingeplant, damit ein ganzer Tag für das Testen reserviert werden kann. Ich bin allerdings nicht so zufrieden damit. Da dies eigentlich dem Planen angehört. Die aufgetretenen Probleme sowie Lösungsvarianten werden am Tag 2 in der Dokumentation beschrieben.

13.2 MONTAG, 14.03.2022

### Arbeiten

Risiken analysieren  
Sequenzdiagramm  
Mockups  
Datenbankmodell  
Erster Expertenbesuch

### Erfolge

Die Risiken konnten analysiert und lösungsvarianten gefunden werden.  
Das ERD wurde nach Überarbeitung fertiggestellt.  
Der Hauptexperte konnte Unklarheiten klären und die Wichtigkeit des «Kriterien-Driven-Developments» näherbringen.

### Misserfolge

Datenbank Model v01  
Zeitplan  
Arbeitsjournal

### Aufgetretene Probleme

Ich hatte kein UML-Zeichnungstool per se vorgesehen, diesbezüglich musste ich noch eins aussuchen von den vielen Möglichkeiten.

### Ungeplante Arbeiten

Zeitplan überarbeiten

### Tests

-

### Hilfestellung

-

### Überzeit

Zeitplan überarbeiten: ~30min

### Vergleich mit Zeitplan

Die vom Zeitplan vorgenommenen Arbeiten konnten Zeitnahe erledigt werden. Durch das Überarbeiten des ERDs mussten '30 min länger in Anspruch genommen werden. Dies konnte ich während dem Dokumentieren kompensieren.

### Reflektion

Durch die bereits vorhandenen sehr detaillierten Userstories habe ich auf ein UseCase Diagramm verzichtet, woraus die Userstories abgeleitet worden wären. Die Schnittstellen wurden diesbezüglich auch bereits dokumentiert.

Ich war mit dem ERD v01 nicht zufrieden. Das Versenden von SMS ist zentral wichtig. Dadurch, dass S&R Mitarbeiter und Imker in derselben Tabelle waren, mussten die Attribute Phone, Firstname und Lastname auf nullable gesetzt werden. Bei der Registrierung kann zwar durch Validierungen sichergestellt werden, dass ein Imker eine Telefonnummer hat, jedoch bei bspw. Seedern ist das nicht der Fall. Sollte auch das Projekt übernommen werden, wäre anhand des ERD nicht erkennlich, dass diese Informationen required sind.

Beim ERD v02 könnte der Primary Key von Beekeeper der Foreign-Key von User direkt sein. Im Laravel würde dies aber nicht den Namingconventions entsprechen. Um zu verhindern, dass ein User mehrere Beekeepers haben könnte, wurde in der Beekeepers Table der Fremdschlüsse auf unique gesetzt.

Das Gespräch mit dem Hauptexperten hat mir aufgezeigt, dass mein Zeitplan sowie die Tagesjournale noch Optimierungsbedarf haben. Zudem sollte das Testkonzept in der Planung erstellt werden. Den Zeitplan habe ich gleich nach dem Gespräch angepasst.

13.3 03DIENSTAG, 15.03.2022

### **Arbeiten**

Testkonzept  
Lösungsvarianten & Massnahmen  
Laravel Setup  
Datenbank & Migrations erstellen

### **Erfolge**

Planung konnte abgeschlossen werden.  
Entscheidung konnte abgeschlossen werden.

### **Misserfolge**

Testkonzept:  
Die geforderten Anforderungen des Kriterienkatalogs erwiesen sich als schwierig zu beschreiben.

### **Aufgetretene Probleme**

Die kürzlich erschienene neue Bootstrap Version (v5.1) enthält einige unerwartete Änderungen.  
Beispielsweise margin-left wurde von ml zu ms (margin-start) geändert.

### **Ungeplante Arbeiten**

Bootstrap Änderungen nachlesen (Bootstrap, 2022)  
Journal den Kriterien anpassen

### **Tests**

#### **Migrations:**

Die Migrations wurden ausgeführt und anschliessend die Datenbank mittels phpMyAdmin überprüft nach Vorgabe des ERD v02.

#### **Hilfestellung**

-

#### **Überzeit**

Journal überarbeiten: ~30min

### **Vergleich mit Zeitplan**

Die Erstellung des Testkonzepts dauerte länger als eingeplant. Alle einzelnen Testfälle detailliert zu beschreiben, erwies sich als sehr aufwendig. Am Abend musste noch durch Rückmeldung des Hauptexperten die Arbeitsjournals den Kriterien angepasst und überarbeitet werden.

### **Reflektion**

Ein sehr ausführliches Testkonzept zu erstellen mit einzelnen Testfällen erwies sich in der Planung als schwieriger als gedacht. Dennoch konnten etliche Testfälle gut beschrieben werden.  
Die Massnahmen der Lösungsvarianten wurden genauer in Betracht gezogen. Sollte ein Problem auftreten, kann innert kürzester Zeit auf eine Lösungsvariante umgestiegen werden.  
Die Datenbank könnte man durch das «Forward Engineering» der MySQLWorkbench erstellen. Ich habe mich dazu entschieden, eigene Migrations in Laravel zu erstellen. Dadurch könnte beim Testen die komplette Datenbank vor jedem Unitest neu erstellt werden. Zudem ist es einfacher, die Datenbank inkl. Migrations neu zu erstellen direkt via Konsolenbefehl, was während der Entwicklung immer wieder mal der Fall sein wird. Die unerwarteten Bootstrap Änderungen haben kurz für Verwirrung gesorgt. Dadurch, dass ich Laravel v8 verwende und erst seit ca. zwei Wochen v9 erhältlich ist, bin ich davon ausgegangen, dass auch die erst kürzlich neue Bootstrap Version nicht im Laravel v8 enthalten ist. In der Bootstrap Dokumentation musste ich mich kurz in die neusten Änderungen einlesen.

13.4 DONNERSTAG, 17.03.2022

### **Arbeiten**

Models & Seeder  
Auth (Login / Logout)  
Imker Registrierung

### **Erfolge**

Alle zu implementierenden Arbeiten konnten erfolgreich umgesetzt werden.

### **Misserfolge**

Default Wert in Migration bei is\_admin vergessen.  
Beekeeper Model relation zu Contract vertauscht.

### **Aufgetretene Probleme**

Password Validation  
Bootstrap neue Syntax dismissable alert  
Rolebased Login redirect

### **Ungeplante Arbeiten**

Journal wurde aus der Dokumentation entfernt und wird als eigenständiges Dokument geführt.

### **Tests:**

#### **Seeders**

Die Seeders wurden ausgeführt und in der Datenbank mit phpMyAdmin auf die Richtigkeit sowie Vollständigkeit überprüft.

#### **Modelrelations**

Mit dem Befehl «php artisan tinker» kann in einer Console php Code ausgeführt werden. Dadurch, dass in der Datenbank Testaten enthalten sind, konnten alle Relations getestet werden. Ein Imker hat zwei Relations zu einem Contract. Es konnte festgestellt werden, dass die beiden Relations vertauscht wurden. Dies wurde angepasst.

#### **Login/Logout**

Das Login sowie der Logout wurde für Imker und für Mitarbeiter von S&R erfolgreich getestet.

#### **Register**

Das Registrieren eines Imkers wurde getestet. Dabei wurde vor allem die Validierung des Backends überprüft.

#### **Telefonnummer-Formater**

Mittels «php artisan tinker» wurde die Funktion zum Formatieren der Telefonnummer manuell überprüft.

#### **Regex-validation**

Die erstellten Regex-Validierungen wurden mit Hilfe von Testfällen unter «<https://regexr.com/>» getestet.

#### **Hilfestellung**

Die Regex Passwortvalidierung wurde durch eine schnelle Googlesearch suche überprüft und übernommen, um nicht zu viel Zeit in die Regex-Validierung zu investieren. (Maytham, 2020)

Bootstrap v5.1 verwendet eine leicht angepasste Syntax für dismissable-alerts als mir bekannt aus v4.x. Dies konnte in der Doku ausfindig gemacht werden. (Bootstrap, 2022)

Es war mir bekannt, wie ein normaler Redirect nach dem Login geändert werden kann, jedoch wusste ich nicht, dass die Variable durch eine gleichnamige Funktion ersetzt werden kann. (Rahman, 2020)

#### **Überzeit**

-

**Vergleich mit Zeitplan**

Das Login hat etwas länger Zeit in Anspruch genommen als eingeplant, jedoch war die Registrierung durch die Verwendung von Components wesentlich schneller als erwartet. Die Dokumentation zum heutigen realisierten Teil ist noch nicht vollständig. Morgen (Tag-5) sind jedoch noch zwei Stunden für die Doku von heute eingeteilt. Dies sollte zuversichtlich ausreichen. Diesbezüglich befindet sich mich innerhalb des Zeitplans.

**Reflektion**

Basic Models konnten aus der Datenbank generiert werden. Mithilfe des Packages «krlove/eloquent-model-generator» wird ein Model erstellt mit den Attributen und Datentypen aus der Datenbank. Die generierten Relations zu den anderen Models müssen angepasst werden. Beispielsweise der Name der Funktion bei mehreren Relations vom Model Beekeeper zu Contract sowie die Relation selbst von User zu Beekeeper. Die Seeder wurden nach dem Schema aus vorherigen Projekten erstellt. Zu jedem Model gibt es einen eigenen Seeder. Alle werden vom Main-Seeder «DatabaseSeeder» der Reihenfolge nach ausgeführt.

Bei der Authentifizierung musste vorwiegend das Frontend erstellt werden. Die Logik dahinter ist bereits von Laravel erstellt lediglich der Redirect musste angepasst werden.

Die Registrierung ist auch vom Laravel bereits vordefiniert. Die Methoden aus dem RegisterController mussten entsprechend angepasst werden. Dies beinhaltet die Validierung sowie dass ein User und Imker erstellt wird. Das Frontend dazu wurde komplett neu nach Mockups erstellt mit Hilfe von Components. Die Components dienen dazu, dass CopyPaste Teilelemente ausgelagert werden können. Dadurch muss bei Änderungswünschen lediglich im Component selbst eine Änderung vorgenommen werden und das komplette Design im Frontend ändert sich an allen Stellen wo das Component verwendet wird. Zudem ist die View, welche mehrere Components verwendet, viel übersichtlicher, da die ganze Fehlermeldung sowie HTML-Inputs, Labels, CSS-Klassen etc. nicht mehrfach in der View selbst enthalten sind.

13.5 FREITAG, 18.03.2022

### **Arbeiten**

Dokumentation Tag-4 fertigstellen

Middleware

Zuständigkeit (Imker PLZ)

LiveSearch

### **Erfolge**

Die LiveSearch nach PLZ oder Ortsname funktioniert und Postleitzahlen können vom Imker erfolgreich hinzu respektive entfernt werden.

### **Misserfolge**

Die Middleware wurde nicht im Zeitplan eingeplant.

### **Aufgetretene Probleme**

Rückgabe vom Laravel Controller in der LiveSearch war inkonsistent. (beschrieben unter Tests).

Diesbezüglich konnte teilweise nicht mit JS über die Rückgabe iteriert werden mittels for of.

### **Ungeplante Arbeiten**

Kleine Verbesserung an Register Validation

Profil Frontend

Middleware

### **Tests:**

#### **Middleware**

Die Middleware wurde getestet in dem mit allen Rollen (Gast, Mitarbeiter S&R, Imker) die jeweiligen URLs aufgerufen wurden Anhand der Rolle wurde überprüft, ob die URL verfügbar ist oder ob es zu einem Redirect kommt. Alle Tests verliefen erfolgreich.

#### **LiveSearch Backend**

Das Backend wurde via URL und Übermittlung der Post variable ausgiebig getestet. Erkannt konnte werden, dass wenn vom Laravel eine Collection erstellt wird, wird das Collection-Object zurückgegeben, aber bei einem return ohne Collection ein Array verwendet wird. Beim Umwandeln in ein JSON wird aus der Collection ein assoziatives Array.

#### **PLZ zuteilen / entfernen**

Die PLZ suche, hinzufügen und entfernen wurde ausgiebig getestet. Dabei wurden alle möglichen Kombinationen von nur hinzufügen bis hin zu mehrfach Hinzufügung / Löschung getestet.

#### **Hilfestellung**

-

#### **Überzeit**

Dokumentation Tag (5)  
beendet ~45min.

### **Vergleich mit Zeitplan**

Es waren mehr Teilschritte bis zur fertigen Implementation der Zuständigkeit nötig als angenommen. Vorab war mir bewusst, dass eine Middleware benötigt wird für Imker, diese wurde aber nicht als eigenständige Arbeit eingeplant.

### **Reflektion**

Während dem Vervollständigen der Dokumentation ist mir noch aufgefallen, dass keine Unterscheidung der ErrorMsg im RegisterController der beiden unterschiedlichen Regexes gemacht wurde. Dies wurde überarbeitet.

Anstatt einen Link zur Zuständigkeit in der Navbar zu hinterlegen, habe ich einen Link dem Profil hinzugefügt. Als ich dabei war, habe ich gleich das Frontend implementiert mittels der Components. Diese mussten allerdings angepasst werden. Dies hat mehr Zeit in Anspruch genommen als erwartet und wahr im Rahmen der IPA unnötige Arbeit.

Der Block für die Implementierung der Zuständigkeit der Imker hätte vorab durch Teilschritte geplant werden sollen. Mit kleineren Teilschritten hätten besser abgeschätzt werden können, wie lange die Implementierung/Doku dauert. Mir war von Anfang an bewusst, die Rollenunterscheidung mittels Middlewares in den Routes zu handhaben. Allerdings ging dies beim Planen vergessen und musste vor der heutigen geplanten Realisierung implementiert werden.

13.6 MONTAG, 21.03.2022

### **Arbeiten**

S&R Auftragserstellung  
Input Formular Components  
Nominatim PLZ-Ermittlung  
Nominatim Validierung  
Auftrags-SMS Notifikation

### **Erfolge**

Components Rückwärtskompatibel angepasst mit Default Values  
SMS-Notifikationen erfolgreich aus der Applikation versendet

### **Misserfolge**

2. Implementation von Notifications falsche Laravel Doku gelesen. (v9.x)
3. Implementation Versionskonflikt

### **Aufgetretene Probleme**

Versionskonflikt mit dem package nexmo/laravel

Im Gespräch mit VF konnte ich feststellen, dass beim Testen keine automatisierten Tests vorgesehen sind. Unter Unit-Tests sind keine automatisierten phpUnitests angedacht, sondern normale Testfälle. Diesbezüglich entfällt der für die automatisierten Testfälle vorgesehene eingetragene Zeit im Zeitplan und wird der Reserve zugewiesen. Dem Testkonzept wurde das automatisierte Testen entfernt. Zudem ist entschieden worden, dass der Vorname eines Imkers auch ein Pflichtfeld ist wie alle restlichen Attribute eines Imkers.

### **Ungeplante Arbeiten**

Components überarbeiten/neue erstellen

Komplexere Validierung als gedacht durch Berücksichtigung eines möglichen API-Fehlers (Nominatim Validierung)

Informieren über Laravel Notifikationen

SMS-Notifikation überarbeiten

### **Tests:**

#### **Auftragserstellung Validierung**

#### **PLZ-Reverse Search**

Die Reverse-Search wurde ausgiebig überprüft, um sicherzustellen, wenn keine PLZ zugewiesen werden konnte, dies sicher abgedeckt ist.

#### **SMS-Notifikation**

Das Versenden einer SMS wurde ausgiebig getestet. Insbesondere die Textdarstellung wurde mehrfach überarbeitet.

### **Hilfestellung**

Versionskonflikt GitHub Repository von nexmo/laravel

Dokumentation (Nexmo Github, 2022)

Laravel Dokumentation: Notifications (Laravel LLC., 2022)

### **Überzeit**

-

### **Vergleich mit Zeitplan**

Es waren wesentlich mehr Teilschritte und versteckte Arbeiten für die beiden geplanten Implementierungen zu realisieren. Diesbezüglich wurde der ganze Tag programmiert. Die Dokumentation wird morgen nachgetragen. Dafür sind bereits 2h vorab im Zeitplan reserviert worden. Dies wird vermutlich nicht ganz ausreichen. Entsprechend wird es zu einem Verzug kommen. Durch die zusätzlichen 4h Reserve kann dem jedoch entgegengewirkt werden.

**Reflektion**

Beim Informieren, wie Vonage verwendet wird im Laravel war ich nicht gründlich und habe mich auf die Vonage Dokumentation abgestützt, welche die Verwendung in php basierten Applikationen dokumentiert hat, aber nicht genauer auf Frameworks eingeht. Dies war zwar die simpelste Implementierung und hat auch funktioniert, jedoch durch die Verwendung von Notifikationen im Laravel, kann das Senden einer SMS an einen Queue-Workers übergeben werden. Dadurch muss bei der Auftragserstellung nicht gewartet werden, bis alle SMS versendet wurden, um die Erfolgsmeldung «Auftrag erstellt» erscheinen zu lassen. Zudem kann für jede spezifische Nachricht eine neue Notifikation erstellt werden und der Text in dieser ausgelagert werden. Dadurch können alle Nachrichten zentral an einem Ort verwaltet werden. Beim Fehlschlag einer Notifikation bleibt diese zusätzlich einfach in der Queue und wird periodisch abgearbeitet, bis sie erfolgreich versendet wurde.

Der Versionskonflikt konnte durch das Lesen der Dokumentation im Git-Repository von nexmo/laravel behoben werden durch die manuelle Hinzufügung im composer.json File einer früheren Version.

Die Auftragsnotifikation könnte durch einen Observer beim Erstellen eines neuen Contracts aufgerufen werden. Dies ist jedoch später beim Zuweisen eines Imkers nicht möglich.

Diesbezüglich wird wegen Consistency und Lesbarkeit für nicht Laravel-Kenner nach dem erfolgreichen Erstellen eines Auftrags die Auftragsnotifikation direkt aufgerufen.

Der Zeitplan hätte mit einzelnen Teilzielen befüllt werden sollen, um besser abschätzen zu können, wie lange die Implementierung dauern wird.

13.7 DIENSTAG, 22.03.2022

### **Arbeiten**

Dokumentation (Tag 06)

Imker Zusage

Auftrags-SMS Bestätigung

### **Erfolge**

Notifikationen erfolgreich abgeschlossen.

Realisierung bis auf ImkerSearch abgeschlossen.

### **Misserfolge**

Durch die länger als geplante Realisierung des Vortags, musste heute mehr Zeit als geplant für die Dokumentation des Vortags investiert werden.

### **Aufgetretene Probleme**

Beim Testen der Bestätigungs-SMS ist mir aufgefallen, dass bei der Formatierung der Telefonnummer der Case «+0» mehrfach formatiert, sodass +41 zu 4141 formatiert und entsprechend falsch in der Datenbank gespeichert wird.

### **Ungeplante Arbeiten**

Wesentlich mehr Validierung der Berechtigungen war notwendig als gedacht für Auftrag bereits vergeben, Auftrag noch verfügbar, Imker darf (nicht) Auftrag annehmen usw.

### **Tests:**

#### **SMS-Bestätigung**

Der Text der SMS-Bestätigung wurde mehrfach überprüft und verbessert.

#### **Imker Zusage**

Die Imker Zusage sowie alle Möglichkeiten, dass ein Auftrag bereits vergeben wurde, wurden überprüft.

### **Hilfestellung**

-

### **Überzeit**

-

### **Vergleich mit Zeitplan**

Die Dokumentation von Tag 06 dauerte länger als geplant. Die geplante Implementierung der Imker Zusage und der Auftragsbestätigung konnte heute abgeschlossen werden, jedoch muss auch hier die Dokumentation auf den nächsten Tag verlängert werden. Diesbezüglich bin ich im Zeitplan ca. 2h hinterher. Durch die eingeplanten Zeitreserven ist dies aktuell noch kein Problem.

### **Reflektion**

Die Dokumentation vom Vortag musste zuerst abgeschlossen werden, bis die eigentlichen Arbeiten von heute angefangen werden konnten. Anscheinend wird in einem php Case der Vergleichsoperator == verwendet und die beiden Cases «+4» und «04» werden in einen int umgewandelt, bevor sie miteinander verglichen werden und resultiert in «4==4». Durch das Entfernen des + mit str\_replace vor dem Switch wird der Case «+41» behandelt. Ich bin sehr überrascht, dass mir dies beim Testen der Funktion nicht direkt aufgefallen ist.

Für die Auftragsbestätigung konnte einfach eine neue Notifikation erstellt werden, gleich wie die Auftragsnotifikation, aber in welcher der Text der Nachricht entsprechend geändert wurde.

13.8 DONNERSTAG, 24.03.2022

### **Arbeiten**

Dokumentation (Tag 7)  
 Imker Search  
 Auftragsnotifikation erweitern  
 Zweiter Expertenbesuch

### **Erfolge**

Das eher Komplexe DB-Query der Imker Proximity-Search funktioniert auf Anhieb.  
 Die Realisierung konnte abgeschlossen werden. Die Applikation funktioniert so weit. Nach dem Testen kann ein aussagekräftiges Fazit gemacht werden, ob alle Erwartungen erfüllt worden sind.

### **Misserfolge**

Das Testkonzept ist noch nicht detailliert genug beschrieben und muss überarbeitet werden.  
 Die Dokumentation ist noch nicht auf dem aktuellen Stand der Realisierung.

### **Aufgetretene Probleme**

Fehlender Margin Ende des Contents: Footer überlappt mit Content beim Scrollen.  
 Validierung der Telefonnummer auf unique war nicht möglich, da eine nicht formatierte Telefonnummer mit den formatierten Nummern in der Datenbank verglichen wurde. Die Formatierung wurde vor der Validierung hinzugefügt.

### **Ungeplante Arbeiten**

Aufgetretene Probleme lösen

### **Tests**

Imker Proximity Search

### **Hilfestellung**

-

### **Überzeit**

-

### **Vergleich mit Zeitplan**

Die Dokumentation vom Vortag konnte nicht innert der geplanten Zeit abgeschlossen werden.  
 Damit aber die Applikation vollständig ist, wird die Dokumentation auf den nächsten Tag verlegt und mit der geplanten Imker Search fortgefahrene.

### **Reflektion**

Die Auftragsnotifikation konnte wie geplant erweitert werden mit der Proximity Search der Imker, falls kein Imker einer Auftragsregion direkt zugewiesen ist. Dies wurde vorab bereits während der Entwicklung der Auftragsnotifikation in Betracht gezogen und vorbereitet, dass nur noch die Proximity Search aufgerufen werden muss. Der fehlende Margin am Ende des Contents für den Footer war mir bereits seit Längerem bekannt. Die Anpassung wurde jedoch hinausgeschoben, bis die Realisierung der Applikation funktional abgeschlossen ist und wesentlich eine höhere Priorität hat. Beim zufälligen Testen durch das manuelle Erstellen neuer Imker für die Proximity Search konnte festgestellt werden, dass die Validierung nicht funktioniert für Imker Telefonnummern auf unique. Die Formatierung musste vor der Validierung hinzugefügt werden. Ich bin sehr erleichtert, dass das Proximity Search Query auf Anhieb funktioniert hat, so wie ich es mir vorgestellt habe. Die PLZ Suche für die Imker Search konnte fast komplett von der Imker Zuständigkeit übernommen werden und für die Imker Search erweitert werden. Die Dokumentation ist noch nicht vollständig und wird morgen definitiv im Zentrum stehen. Durch den leichten Verzug der Vortage und dass das Testkonzept noch überarbeitet werden muss, wird morgen versucht, die Zeit wieder gut zu machen während dem Dokumentieren.

13.9 FREITAG, 25.03.2022

**Arbeiten**

Testkonzept detaillierter beschreiben

Code dokumentieren

Kriterien der IPA Durcharbeiten

**Erfolge**

Der noch fehlende Sourcecode Kommentar konnte hinzugefügt werden. Somit ist die Programmierung bis auf mögliche Fehler der Testfälle komplett abgeschlossen.

**Misserfolge**

Die Dokumentation war weniger ausgereift als angenommen anhand des Genauen betrachten der Kriterien.

Durch den Verzug konnten die Testfälle nicht mehr heute getestet werden. Diese müssen am letzten Tag als Erstes am Morgen durchgeführt werden.

**Aufgetretene Probleme**

-

**Ungeplante Arbeiten**

Testkonzept detaillierter beschreiben

**Tests**

-

**Hilfestellung**

-

**Überzeit**

Dokumentation für Testen  
vorbereiten: ~30min

**Vergleich mit Zeitplan**

Die geplante Testdurchführung musste um einen Tag verschoben werden, um die Dokumentation bis hin zur Realisierung abzuschliessen. Dies war ein grösserer als geplanter Aufwand.

**Reflektion**

Ich bin nicht mehr so zufrieden wie am Anfang mit der Wahl der Projektmethode. Das im Zentrum stehende Kontrollieren kommt erst viel zu spät zum Zug. Natürlich wurden während allen Projektschritten jeweils die Kriterien beachtet. Aber mehr im Hinterkopf nebenbei als sich tatsächlich richtig damit auseinandergesetzt. In Bezug auf die IPA, welche sehr klar vorgegebene Kriterien aufweist, hätte ich nach jedem Projektschritt die jeweiligen Kriterien begutachtet, evaluiert, überarbeitet und komplett abgeschlossen haben.

Es ist sehr riskant, die Testung erst am letzten Tag eines Projektes durchzuführen. Ich bin jedoch sehr zuversichtlich durch das ständige ausgiebige Testen während der Realisierung, dass keine funktional gravierenden Fehler auftreten werden und die Applikation alle Erwartungen erfüllt. Es könnten fehlende Fehlermeldungen oder kleine, nicht funktionale Verbesserungen bemängelt werden. Diesbezüglich sollten maximal zwei Testläufe benötigt werden innert den ersten beiden Stunden und das Projekt kann anschliessend vollendet werden mit der Reflexion und dem Zusammensetzen der Dokumentation.

13.10 MONTAG, 28.03.2022

**Arbeiten**

Testfälle testen

Reflexion & Fazit

Dokumentation vervollständigen

**Erfolge**

Die IPA wurde aus meiner Sicht vollständig innert der zeitlichen Frist vollendet.

**Misserfolge**

-

**Aufgetretene Probleme**

Drei Testfälle waren nicht komplett erfüllt wie erwartet. Entsprechende Änderungen mussten geplant und implementiert werden.

Mir war nicht klar, ob im Glossar alle verwendeten Fachbegriffe oder nur welche die einer aussenstehenden Fachperson unklar sind, enthalten sein sollen.

**Ungeplante Arbeiten**

Testfälle testen musste vom Vortag auf heute verschoben werden.

**Tests**

Die ganze Applikation wurde mithilfe der Testfälle getestet.

**Hilfestellung**

VF hat zum Glossar bestätigt, es sollen nur Fachbegriffe, welche einer aussenstehenden Person unklar sind, enthalten.

**Nacht/Wochenarbeit**

-

**Vergleich mit Zeitplan**

Die geplante Reflexion & Fazit wurden durch das Durchführen der Testfälle um 1.5h verschoben und wurde auf das Wesentliche verkürzt.

**Reflektion**

Das Risiko mit dem Testen der Testfälle am letzten Tag hat sich meiner Meinung voll ausgezahlt. Die gewonnene Zeit konnte am Freitag in die Dokumentation investiert werden und war zentral für eine pünktliche und aus meiner Sicht vollständige Abgabe. Die eingeplanten Zeitreserven konnten vollkommen in der Dokumentation ausgeschöpft werden, um den letzten Feinschliff zu tätigen.

# Teil 2: Projekt

## 14 KURZFASSUNG

---

### 14.1 KURZE AUSGANGSSITUATION

Die Einsatzzentrale der Schutz und Rettung erhält jährlich fälschlicherweise mehrere Hundert Anrufe bezüglich Bienenschwarmentfernungen. Diese werden mit einem grossen manuellen Mehraufwand an lokale Imker übergeben. Es ist eine Applikation gewünscht, mit welcher sich Imker als freiwillige Helfer registrieren können. Bei einem künftigen Einsatz soll mit der Übermittlung der Geodatenlokalisierung autonom die nächstgelegenen Imker ausfindig gemacht und aufgeboten werden via SMS.

### 14.2 UMSETZUNG

Bei der Applikation handelt es sich um eine PHP-Webapplikation, welche das Framework Laravel verwendet. Als relationale Datenbank wurde MariaDB vom Programm XAMPP verwendet sowie dessen Apache Web Server. Das von Laravel verwendete ORM Eloquent dient als Schnittstelle zwischen den Models und der Datenbank. Das Projekt ist nach MVC gegliedert. Die Views werden von der Template-Engine Blade zusammengesetzt und fürs Frontend gerendert. Jegliche Kontrollstrukturen sowie Benutzereingaben werden in den entsprechenden Controllern gehandhabt und Daten an die Views übergeben. Die rollenbasierte Berechtigung wurde mittels mehrerer Middlewares umgesetzt in den Routes. Formulareingabefelder wurden mittels Components erstellt. Dadurch sind die Formulare sehr übersichtlich und jegliche HTML- und CSS Eigenschaften sowie die Fehleranzeige im Component ausgelagert. SMS-Notifikationen werden mittels Vonage dem Imker versendet. Notifikationen werden beim Erstellen einer Queue hinzugefügt, diese wird von einem Queue-Worker Parallel zur Applikation abarbeitet. Beim Google Maps Pin handelt es sich um eine URL, welche ein Pin an der Longitude und Latitude des Auftragsort setzt. Die gesuchte PLZ wird mittels der API von Nominatim aus der Geodatenlokalisierung des Auftrags ausfindig gemacht.

### 14.3 ERGEBNIS

Imker können sich mit ihren Kontaktinformationen Registrieren und ihre zuständigen Regionen auswählen. Die Mitarbeiter von S&R können einen neuen Auftrag erstellen. Dabei geben sie die Geolokalisierung in Form von Latitude und Longitude an und allfällige Zusatz Informationen wie Kontaktangaben und Details zum Auftrag. Die Applikation ermittelt die repräsentative PLZ des Auftrags durch eine API-Abfrage von Nominatim. Sollte diese nicht funktioniert haben, wird der Mitarbeiter aufgefordert, die PLZ manuell anzugeben. Anschliessend werden Imker falls möglich, welche direkt zuständig sind für diese Region, ansonsten die Nächstegelegenen per SMS benachrichtigt mit der PLZ sowie einer URL zur Auftragsannahme. Dem Imker, welcher als erstes den Auftrag annimmt, erhält die zusätzlichen Infos in der Auftragsübersicht. Die zusätzlichen Infos sowie ein Google Maps Pin des genauen Auftragortes werden zusätzlich ihm per SMS übermittelt.

## 15 INFORMIEREN

Zusätzlich zum ersten Teil dieser Dokumentation wurden nachfolgende Informationen ausfindig gemacht.

### 15.1 EXTERNE SCHNITTSTELLEN

#### 15.1.1 Nominatim

Um die PLZ eines Einsatzes ausfindig machen zu können, müssen die Geokoordinaten von Latitude und Longitude der entsprechenden PLZ zugewiesen werden. Die API Nominatim besitzt eine Reverse Geocoding API Request.

Nominatim sucht das am nächsten gelegene Objekt der Geokoordinaten und gibt die Adressinformationen zurück. Diese umgekehrte Suche ist eher ungenau für spezifische Adressen. In diesem Projekt werden jedoch nur die PLZ benötigt und ist somit absolut im Rahmen des Fehlerwerts.

Eine Suche kann mit folgender URL getätigter werden:

```
https://nominatim.openstreetmap.org/reverse?lat=<value>&lon=<value>&format=json
```

(Nominatim, 2022)

**Beispiel Output als JSON:**

```
// 20220311090248
// https://nominatim.openstreetmap.org/reverse?lat=47.39036&lon=8.49087&format=json

{
  "place_id": 110234953,
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
  "osm_type": "way",
  "osm_id": 31859337,
  "lat": "47.39061755",
  "lon": "8.4909205",
  "display_name": "550, Hohlstrasse, Werdwies, Altstetten, Kreis 9, Zürich, Bezirk Zürich, Zürich, 8048, Schweiz",
  "address": {
    "house_number": "550",
    "road": "Hohlstrasse",
    "neighbourhood": "Werdwies",
    "suburb": "Altstetten",
    "city": "Zürich",
    "county": "Bezirk Zürich",
    "state": "Zürich",
    "postcode": "8048",
    "country": "Schweiz",
    "country_code": "ch"
  },
  "boundingbox": [
    "47.3903833",
    "47.3908518",
    "8.4906472",
    "8.4911938"
  ]
}
```

Abbildung 2 Nominatim JSON Response

### 15.1.2 Google Maps Pin

Um einen Google Maps Pin mit den Geokoordinaten zu teilen via SMS wird folgender Link generiert:

```
https://www.google.com/maps/search/?api=1&query=<lat>%2C<lon>
```

(Google, 2022)

### 15.1.3 Vonage (Nexmo)

Vonage bietet die Möglichkeit SMS aus Applikationen zu verschicken. Um Vonage verwenden zu können wird ein Konto vorausgesetzt.

#### 15.1.3.1 Laravel Setup

Als ersten Schritt muss der Nexmo Client installiert werden via Composer:

```
composer require nexmo/client
```

#### 15.1.3.2 SMS senden

Um eine SMS zu versenden, wird der Nexmo API-Key und das API-Passwort vom Registrierten Konto benötigt:

```
public function sendSmsNotification()
{
    $basic = new \Vonage\Client\Credentials\Basic('key', 'secret');
    $client = new \Vonage\Client($basic);

    $response = $client->sms()->send(
        new \Vonage\SMS\Message\SMS(['to' => "NUMBER", 'from' => 'COC', 'message' => 'Message sent using the Nexmo SMS API'])
    );
}
```

Abbildung 3 Vonage Send Basic SMS

(Vonage, 2022)

Dieses Beispiel dient der Verständlichkeit. Der API-Key und das Passwort werden in der «.env» Datei im IPA Projekt hinterlegt.

### 15.1.4 Mailtrap

Mailtrap ist ein E-Mail-Sandbox Service mit welchem, ausgehende E-Mails getestet werden können. Ein Konto wird benötigt um ein API-Username sowie Passwort zu erhalten.

Um Mailtrap verwenden zu können muss lediglich die «.env» Datei mit folgenden Einträgen ergänzt werden:

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=<API_USERNAME>
MAIL_PASSWORD=<API_PASSWORD>
MAIL_ENCRYPTION=tls
```

Anschliessend können Mails wie für Laravel üblich versendet werden und auf mailtrap.io in der Inbox analysiert werden. (Railsware, 2022)

## 16 PLANEN

---

### 16.1 AUFTRAGSDIVISION

Die zu realisierenden User Stories sowie notwendigen Tasks werden für die Planung und insbesondere für den Zeitplan Stichwortartig chronologisch aufgelistet:

- 1) Laravel Setup
  - a) Projekt-Konfigurationen
  - b) Blade Templates
- 2) Datenbank erstellen
- 3) Models und Seeder erstellen
- 4) User Authentifikation (Login / Logout)
  - a) (S-2a) Imker
  - b) (S-2b) Personal S&R
- 5) (S-0) Imker Registrierung
- 6) (S-1) Zuständigkeit (Imker-PLZ)
- 7) (S-5) S&R Auftragserstellung
- 8) (S-6) Auftrags-Notifikation
  - a) Nächstgelegene Imker suchen
  - b) SMS-Notifikation
- 9) (S-7) Imker Zusage
- 10) (S-8) Auftragsbestätigung (Übermittlung der Einsatz-Adresse)
- 11) (S-10) Imker-Search (Gast)

Die Auflistung erleichtert den Überblick der zu erledigenden Aufgaben im teil Realisierung und werden im Zeitplan übernommen. Beachtet wurde dabei eine Sinnvolle Reihenfolge einzuhalten durch mögliche Abhängigkeiten. Die Authentifikation kann vor der Registrierung erstellt sowie getestet werden durch die Verwendung von Mock-Daten (Datenbank Seeder).

### 16.2 PROJEKTDATENSTRUKTUR

Dieses Projekt wird mit dem Framework Laravel umgesetzt. Entsprechend werden die Normen und Strukturen von Laravel eingehalten.

## 16.3 MOCKUPS

Die aus der Vorarbeit deklarierten Entwürfe für mögliche Mockups der aller ersten Aufgabenstellung (stand 10.01.2021) wurden auf Papier überarbeitet. Sie dienen in erster Linie nicht als definitive Design Vorlage, sondern als Überprüfung, welche Seiten benötigt werden und ob alle Notwendigen Elemente auf den entsprechenden Seiten entsprechend detaillierten Aufgabenstellung enthalten sind. Die Zahlenkennzeichnungen dienen als Navigationshinweis der einzelnen Seiten.

### 16.3.1 Index ①

Die Indexseite besteht einerseits aus der Imker-Search und andererseits aus dem Login. Dazu wird die View halbiert. Für mobile User werden die halbierten Views untereinander dargestellt. Als zusätzliche Überlegung könnte bei genügend Zeitreserven der Login zusätzlich mit der Telefonnummer zur Verfügung gestellt werden. Fehlermeldungen bezüglich des Logins werden zuoberst des Formulars ausgewiesen. Aus Sicherheitsgründen wird nur die Meldung «Login nicht erfolgreich» ausgewiesen und nicht weiter spezifiziert, ob nur das Passwort falsch war.

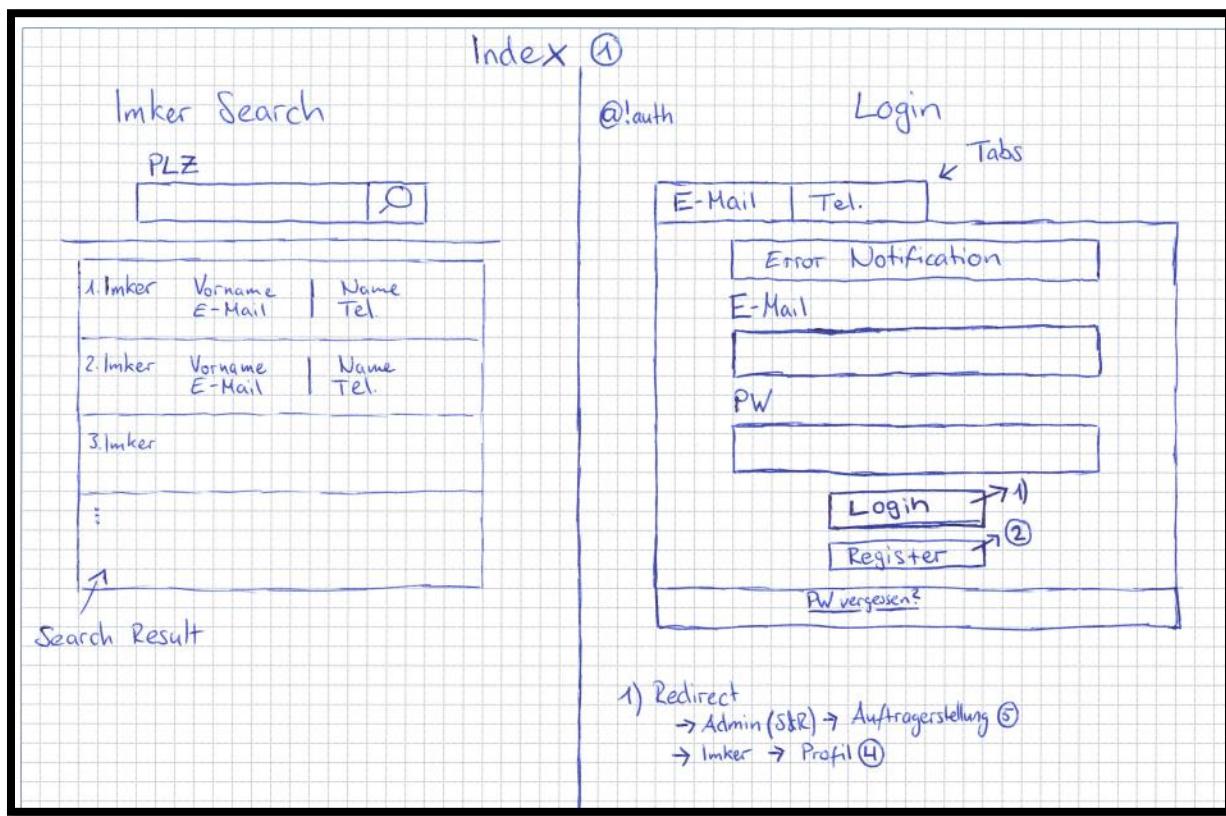


Abbildung 4 Mockup: Index Seite

### 16.3.2 Register ②

Das Registrierformular enthält alle notwendigen Informationen, welcher ein Imker hinterlegen soll. Dadurch, dass die Informationen der Imker bei der «Imker Search» öffentlich zugänglich sind, müssen diese bei der Registrierung ihr Einverständnis dafür geben, aus Datenschutz rechtlichen Gründen.

### 16.3.3 Imker PLZ Zugehörigkeit ③

Die View zur Zugehörigkeit wird halbiert. Auf der linken Seite sind die aktuell zugewiesenen Regionen ersichtlich sowie bei jeder Region ein Delete Button, um die Zugehörigkeit zu entfernen. Auf der rechten Hälfte befindet sich ein Eingabefeld. Mit Beginnen einer Eingabe der PLZ wird unterhalb einer Liste mit Ergebnissen in einer LiveSearch angezeigt. Bei der Auswahl einer Region aus der Liste wird diese zugewiesen.

Register ②

Vorname

Name

E-Mail

Tel

Password

PW Wiederholung

AGBs

Register

Abbildung 5 Mockup: Imker Registrierung

Imker PLZ Zugehörigkeit ③

Zugewiesene PLZ

PLZ   Ort	X	▲
PLZ   Ort	X	
PLZ   Ort	X	
	X	
	X	

PLZ Zuweisen / Search

Search

PLZ

PLZ   Ort	+	▲
PLZ   Ort	+	
PLZ   Ort	+	
	+	
	+	

Abbildung 6 Mockup: Imker PLZ Zuweisung

#### 16.3.4 Profil ④

Das sogenannte Profil, damit die Imker ihre Daten mutieren zu können, ist identisch zur Registrierung mit den Erweiterungen: Altes Passwort Eingabefeld, Benutzerinformationen Löschen Button und Zugehörigkeitslink auf die Imker PLZ Zugehörigkeit.

Ergänzungen:

Old PW

UPDATE ↑④

Löschen

Zugehörigkeit ↑③

Abbildung 7 Mockup: Profil Ergänzung

#### 16.3.5 S&R Auftragserstellung ⑤

Die Auftragserstellung enthält alle notwendigen Eingabefelder nach Aufgabenstellung. Bei einer erfolgreichen Auftragserstellung wird der Benutzer (S&R Personal) wieder auf ein leeres Formular zurückgeleitet mit einer Erfolgsmeldung.

#### 16.3.6 S&R Auftragserstellung PLZ Error ⑥

Sollte es zu einem Fehler kommen mit der automatischen PLZ Erkennung aus den Werten zu lat und lon von der Nominatim Reverse API wird der Benutzer (S&R Personal) zurück auf das Formular geleitet und muss die PLZ von Hand eintragen. Somit kann gewährleistet werden, dass auch Aufträge ohne Abhängigkeit einer externen Schnittstelle eröffnet werden können.

S&R Auftragserstellung Error ⑥

Fehlermeldung

PLZ Manuell eintragen

Speichern ↑⑤

Abbildung 9 Mockup: Nominatim PLZ Fehler

S&R Auftragserstellung ⑤

lat

lon

Kontakt Person

Vorname Name

Tel

Allfällige Hinweise

Success Meldung

Erstellen ↑⑤/⑥

Abbildung 8 Mockup: Auftragserstellung

### 16.3.7 Auftrag Akzeptieren ⑦

Nachdem alle passenden Imker für einen Auftrag gefunden worden sind, werden diese mit einem SMS kontaktiert. Dieses SMS enthält eine URL, um den dazugehörigen Auftrag anzunehmen. Der Imker muss eine Bestätigung abgeben, dass er den Auftrag übernehmen möchte und dass dieser dann bindend ist.

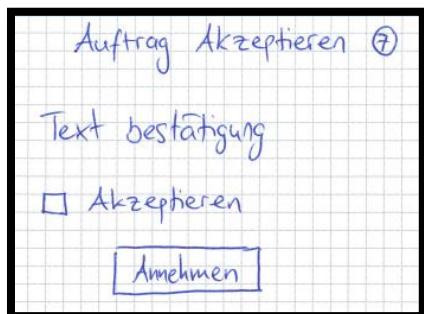


Abbildung 10 Mockup: Imker Auftrag akzeptieren

### 16.3.8 Auftrag bereits Vergeben ⑧

Sollte der Auftrag bereits vergeben sein, wird der Imker auf eine Seite weitergeleitet, in welcher der Auftrag als bereits vergeben ausgewiesen wird.

## 16.4 ERD DATENBANKMODELLE

Nachfolgend wurden zwei Datenbankmodelle mit MySQL Workbench erstellt. Der wesentliche Unterschied besteht darin, wie die Rolle dargestellt werden soll. Die verwendeten Naming-Conventions entsprechen den von Laravel vorgegebenen Konventionen. (webdevetc, 2020)

### 16.4.1 Datenbankmodell v01

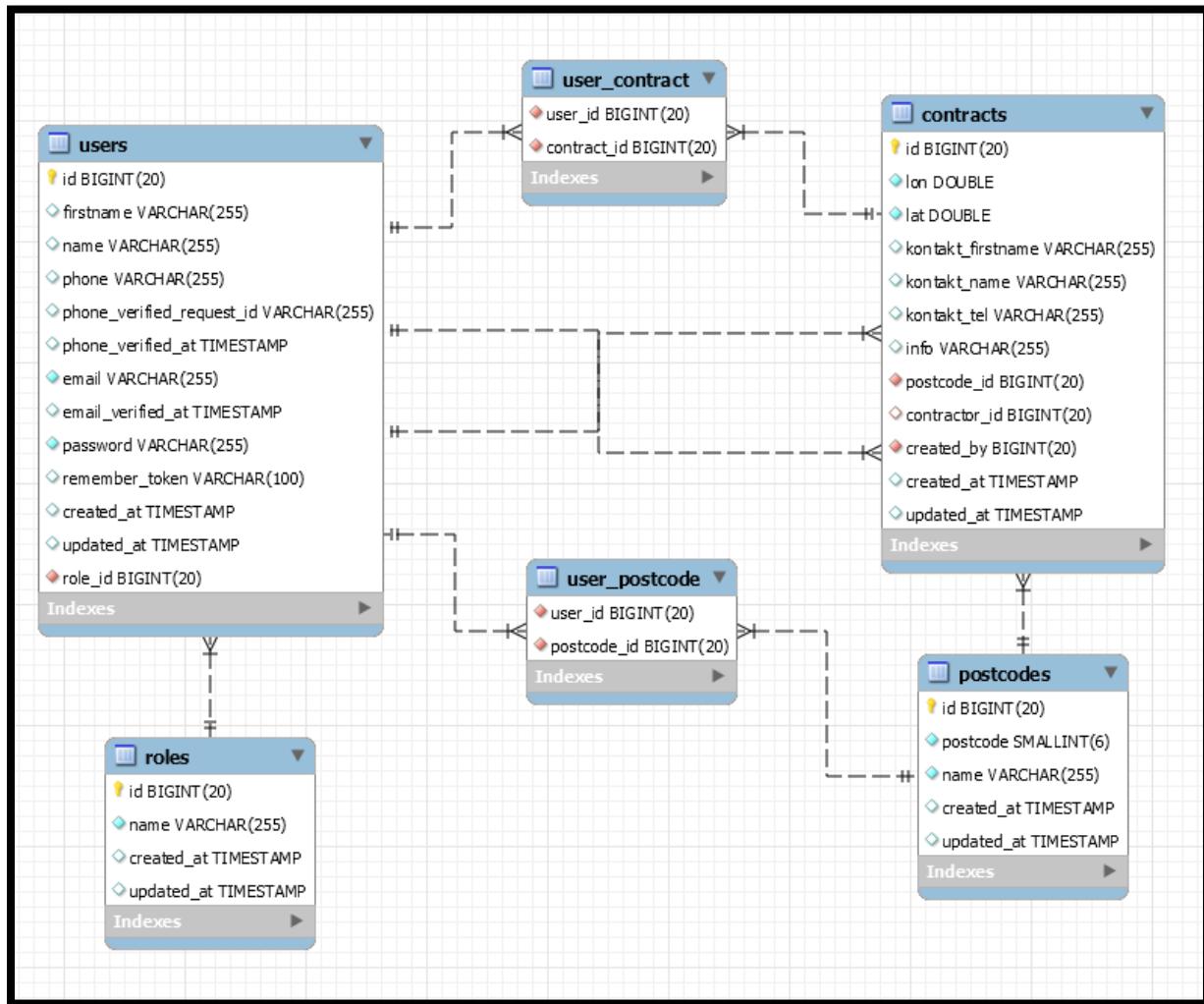


Abbildung 11 Datenbankmodell v01

Beim ursprünglichen Datenmodell v01 hat der User alle Entitäten für einen Imker sowie für einen Mitarbeiter von S&R. Die Rollenunterteilung erfolgt in der Tabelle «roles», dadurch kommt es bei allen Mitarbeitern von S&R, welche nur über eine E-Mail-Adresse sowie ein Passwort verfügen zu vielen leeren Attributen. Die Unterscheidung eines Users als Imker oder Personal von S&R ist zudem auf den ersten Blick nicht ersichtlich. Die SMS-Notifikation ist für Imker zudem essenziell, diesbezüglich darf das Attribut «phone» nicht nullable für Imker in der Datenbank sein.

### 16.4.2 Datenbankmodell v02

Das überarbeitete Datenmodell löst die mangelnden Punkte von v01.

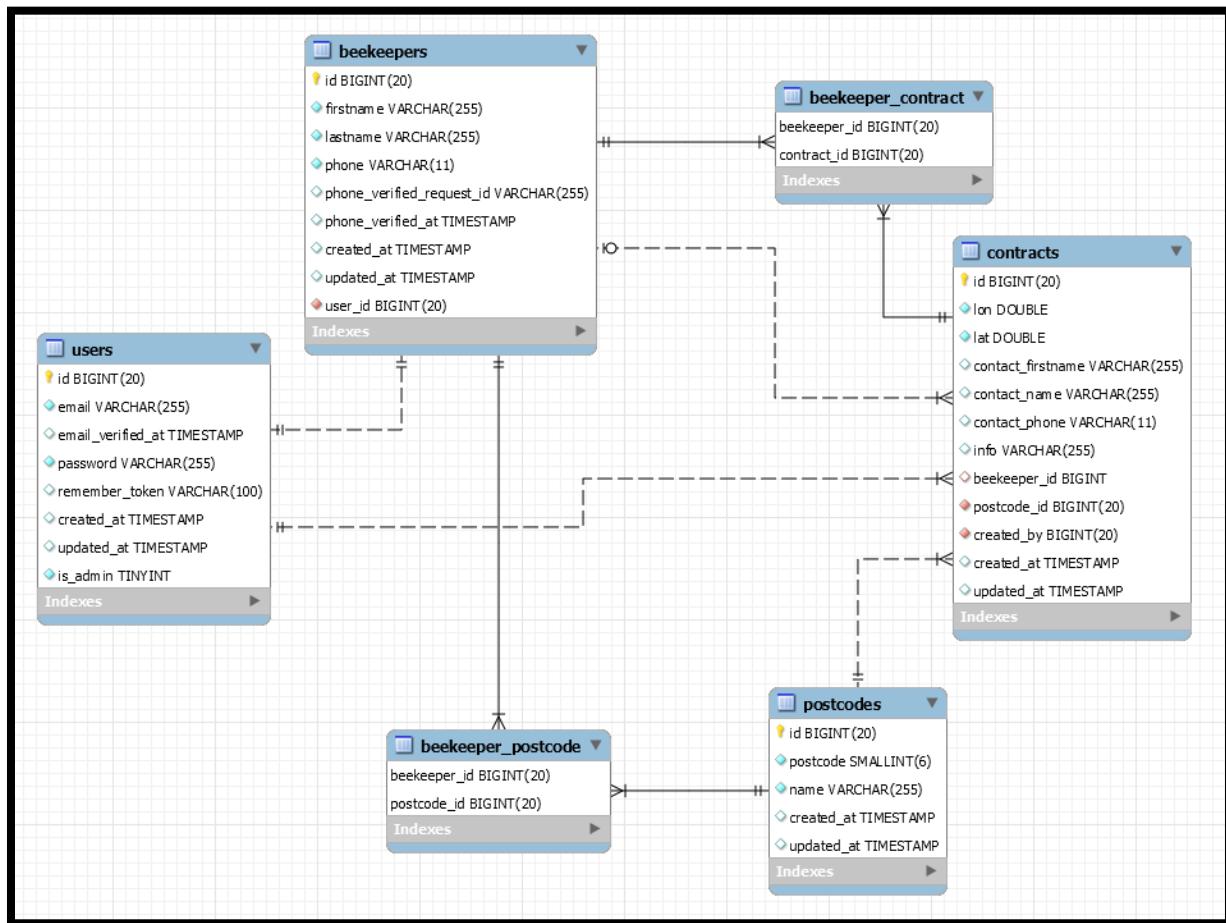


Abbildung 12 Datenbankmodell v02

#### 16.4.2.1 Users

Die Tabelle **Users** enthält alle Login Informationen (E-Mail & Passwort) für Personal S&R sowie für Imker.

#### 16.4.2.2 Beekeepers

Imker erhalten zusätzlich zum User noch eine 1:1 Datenbankbeziehung aus der Tabelle «beekeepers» mit den notwendigen Zusatzinformationen. Sollte zukünftig der Wunsch angebracht werden nicht nur Bienenschwärme, sondern auch andere Objekte einzusammeln mit der gleichen Applikation, kann die Tabelle «beekeepers» zu «contractors» umbenannt werden und eine Tabelle «roles» addiert werden mit einem Fremdschlüssel in «contractors».

#### 16.4.2.3 Postcodes

In der Tabelle «postcodes» werden alle PLZ sowie Ortsbezeichnungen vom Kanton Zürich von der Datei «Liste-der-PLZ-in-der-Schweiz.xlsx» aus dem Dokumentenpool übernommen.

#### 16.4.2.4 Contracts

Diese Tabelle enthält alle Informationen für einen Auftrag. Der Fremdschlüssel «beekeeper\_id» ist bewusst als nullable gewählt, da ein Auftrag zuerst erstellt und zu einem späteren Zeitpunkt einem Imker zugewiesen wird. Die Kontaktangaben sind ebenfalls direkt in dieser Tabelle enthalten, da duplizierte Einträge sehr unwahrscheinlich sind.

#### 16.4.2.5 beekeeper\_contract

Die mc Beziehung zwischen Beekeepers und Contracts gibt Auskunft, welche Imker bereits Aufgeboten wurden für einen Auftrag. Dies ist zukünftig nötig für die Story S-9. Zusätzlich kann damit bei der Imkerzusage überprüft werden, ob ein Imker für den anzunehmenden Auftrag zugewiesen wurde.

#### 16.4.2.6 beekeeper\_postcode

Die mc Beziehung zwischen Beekeepers und Postcodes speichert die vom Imker ausgewählten Zuständigkeitsregionen.

### 16.5 SEQUENZDIAGRAMM

Das folgende Sequenzdiagramm soll den Applikationsablauf darstellen vom Erstellen eines Auftrags durch einen Mitarbeiter der S&R bis zur Auftragsbindung eines Imkers.

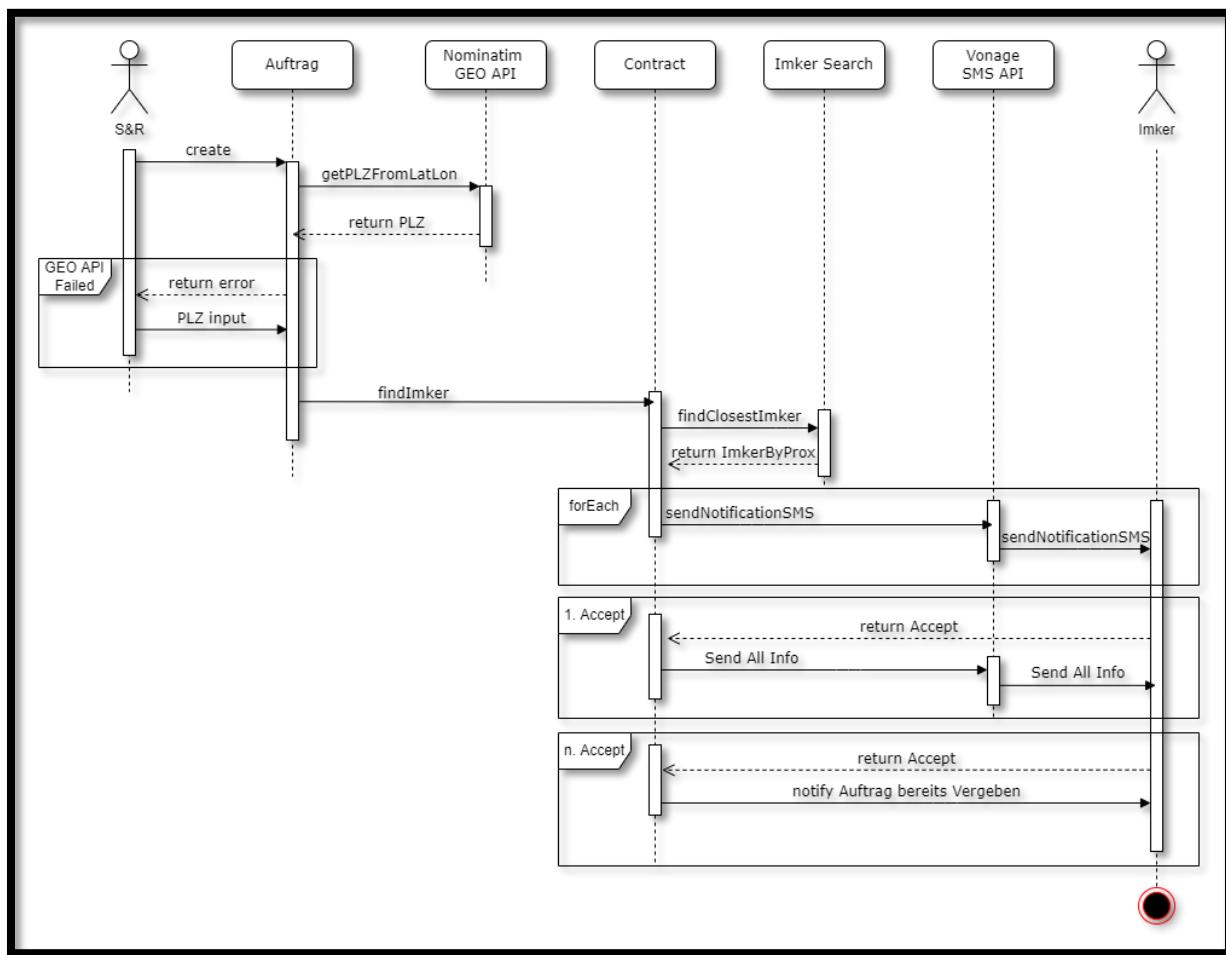


Abbildung 13 Sequenzdiagramm

## 16.6 REALISIERUNGSKONZEPT

### 16.6.1 Ermittlung der zuständigen Imker einer Region

Die Imker einer Region können durch die Verbindungstabelle «v01: user\_postcode» respektive «v02: imker\_postcode» ausfindig gemacht werden. Sollten keine Imker direkt einer Region zugewiesen sein, werden die nächstgelegenen Imker durch die Imker Search ausfindig gemacht.

### 16.6.2 Imker Search

Angrenzende Regionen sind in der detaillierten Aufgabenstellung als PLZ +/- 5 definiert. Daraus kann geschlossen werden, dass ihm Rahmen der IPA alle Postleitzahlen im Kanton Zürich angrenzend verteilt sind. Beispiel: 8700 ist neben 8701. Die Imker können demzufolge nach der Differenz der ihrer am nächsten zugewiesenen Region zu der gesuchten Region sortiert werden.

### 16.6.3 Imker-Zusage

Nach der Ermittlung der zuständigen Imker werden alle via SMS kontaktiert. Die SMS beinhaltet im Wesentlichen die PLZ des Einsatzortes sowie eine URL, welche zum Auftragsusageformular führt. Der zuständige Controller überprüft bei der get-request, ob der Imker für diesen Auftrag zugeteilt wurde, sowie ob der Auftrag bereits vergeben wurde. Wurde der Auftrag bereits vergeben, wird der Imker auf das Formular: «Auftrag bereits vergeben» weitergeleitet. Sind beide Bedingungen erfüllt, kann der Imker den Auftrag annehmen. Im Controller, welcher zuständig für die Annahme ist, wird ebenfalls überprüft, ob der Auftrag bereits vergeben ist. Dies könnte passieren, wenn zwei oder mehrere Imker die URL gleichzeitig öffnen, aber noch keiner die Bestätigung definitiv abgegeben hat. Dadurch kann eine Doppelbuchung verhindert werden.

### 16.6.4 Berechtigungen

Die rollenbasierten Berechtigungen, welche Rolle darf welche URL aufrufen, wird mittels Middleware gelöst. Dazu wird spezifisch eine Middleware für nur Imker und eine für nur Mitarbeiter von S&R erstellt.

### 16.6.5 Components

Für Eingabefelder in Formularen werden Components verwendet. Dadurch wird das Formular viel übersichtlicher und bei Änderungsbedarf muss nur im Component etwas angepasst werden.

### 16.6.6 PLZ Live Search

Bei der PLZ Live Search wird mittels einer Ajax request der Wert aus dem Eingabefeld an einen Controller übermittelt. Dieser returniert die ersten fünf Ergebnisse aus der Suche in der Datenbank und wird mittels JavaScript angezeigt. Dies ist eine wesentliche Verbesserung der Usability. Da kein ständiger Page-Refresh notwendig ist, um das Formular zu übermitteln.

### 16.6.7 MVC Pattern

Die Auftrennung nach dem MVC-Pattern wird konsequent wie auch vom Laravel sehr stark vorgegeben durchgeführt. Die Models dienen zur Datenaufbereitung, die Blade-Views dienen der Darstellung und die Kontroller dienen der Ablaufsteuerung und Validierung.

## 16.7 TESTKONZEPT

Um die Funktionalität der Applikation gewährleisten zu können, muss diese getestet werden. Werden Mängel festgestellt, werden diese behoben. Bei grösseren Fehlern wird eine Lösungsmöglichkeit dokumentiert.

Das zu testende System ist eine PHP-Webapplikation, welche das Framework Laravel verwendet und befindet sich auf der lokalen Entwicklungsumgebung des Entwicklers. Die Applikation automatisiert für S&R den Arbeitsablauf, um lokale Imker aufzubieten bei einer Bienenschwärmsichtung. Imker können sich daher auf der Applikation registrieren und ihre zuständigen Regionen wählen.

Mitarbeiter der S&R können einen neuen Auftrag erstellen. Dieser beinhaltet im Wesentlichen die Geokoordinaten des Einsatzortes und allfällige zusätzliche Informationen wie Kontaktinformationen des Auftraggebers und weitere Infos. Die Applikation ermittelt die PLZ des Einsatzortes und informiert die zuständigen Imker der betroffenen Region per SMS. Nach Annahme eines Auftrags werden alle zusätzlichen Informationen sowie ein Google Maps Pin der Lokalisation per SMS dem zuständigen Imker übermittelt.

Als relationale Datenbank wird MariaDB vom Programm XAMPP verwendet sowie dessen Apache Web Server. SMS-Notifikationen werden mittels Vonage dem Imker versendet. Beim Google Maps Pin handelt es sich um eine URL, welche ein Pin an der Longitude und Latitude des Auftragsort setzt. Die gesuchte PLZ wird mittels der API von Nominatim ausfindig gemacht.

### 16.7.1 Nicht getestet

Die SMS werden von der Applikation als «fired and forget» ausgeführt. Diesbezüglich wird nicht der Fall getestet, was passiert, wenn Vonage nicht verfügbar sein sollte. Für das Testen der Testfälle wird davon ausgegangen, dass der SMS-Dienst von Vonage verfügbar ist. Sollte dies nicht der Fall sein, können einzelne Testfälle, welche auf das Versenden der SMS angewiesen sind, nicht getestet werden.

### 16.7.2 Überprüfung

Alle Testfälle werden manuell vom Entwickler der Reihe nach durchgeführt.

### 16.7.3 Testmittel

Das relevante Testmittel besteht aus der lokalen Entwicklungsumgebung mit einer bestehenden Internetverbindung und einer neu erstellter sowie geseedeter MariaDB Datenbank. Der Apache Webserver sowie die Datenbank werden vom Programm XAMPP zur Verfügung gestellt. Zusätzlich wird ein Telefon mit einer gültigen Telefonnummer benötigt, um SMS empfangen zu können. Getestet wird mit dem Google Chrome Web-Browser v.99.

### 16.7.4 Teststufen

Die nachfolgenden Teststufen werden als Black-Box getestet:

1. Unittests:  
Testen kleinst-möglicher testbarer Funktionalitäten isoliert von anderen.
2. Integrationstests:  
Test der Funktionalität bei der Zusammenarbeit voneinander abhängiger Komponenten.

### 16.7.5 Testmethoden

Folgende Black-Box Testmethoden werden dabei für das Erstellen der Testfallspezifikation verwendet:

1. Äquivalenzklassenbildung
2. Randwertanalyse

### 16.7.6 Äquivalenzklassenanalyse Contract

Eingaben	Gültige Klassen		Ungültige Klassen	
	Nr.	Wertebereich	Nr.	Wertebereich
<b>Latitude</b>	<b>1</b>	47.00000 – 48.00000	<b>3</b>	< 47.00000
			<b>4</b>	> 48.00000
			<b>5</b>	Nicht numerisch
<b>Longitude</b>	<b>2</b>	8.00000 – 9.00000	<b>6</b>	< 8.00000
			<b>7</b>	> 9.00000
			<b>8</b>	Nicht numerisch

#### 16.7.6.1 Testfälle

Testfall Nr.	Eingaben		Erwartetes Ergebnis	Äquivalenzklassen
	Latitude	Longitude		
4.0	47.31901	8.58059	Input plausibel	1, 2
4.3.0, 4.4.0	46.00505	7.00505	Input nicht plausibel	3, 6
4.3.1, 4.4.1	48.39036	9.49087	Input nicht plausibel	4, 7
4.3.2, 4.4.2	lat	lon	Input nicht plausibel	5, 8

#### 16.7.6.2 Randwertanalyse

Testfall Nr.	Eingaben		Erwartetes Ergebnis	Äquivalenzklassen
	Latitude	Longitude		
4.3.3, 4.4.3	46.99999	7.99999	Input nicht plausibel	3, 6
4.3.4, 4.4.4	47.00000	8.00000	Input plausibel	1, 2
4.3.5, 4.4.5	47.00001	8.00001	Input plausibel	1, 2
4.3.6, 4.4.6	47.99999	8.99999	Input plausibel	1, 2
4.3.7, 4.4.7	48.00000	9.00000	Input plausibel	1, 2
4.3.8, 4.4.8	48.00001	9.00001	Input nicht plausibel	4, 7

### 16.7.7 Allgemeine Testvoraussetzungen

- Der Source code der Applikation befindet sich im Verzeichnis «C:\xampp\htdocs»
- XAMPP ist gestartet sowie der Apache und MySQL Dienst sind am Laufen.
- Gültiges Vonage Konto mit genügend Guthaben (> 0.20CHF.-)
- Vonage Key und Secret sind in der Datei «.env» korrekt angegeben.

### 16.7.8 Testfallspezifikation

Die Testfälle sind nicht nach den Teststufen gegliedert, sondern nach dem Applikationsdurchlauf und bauen aufeinander auf. Diesbezüglich müssen die Testfälle der Reihe nach durchgegangen werden. Kommt es zu einem schwerwiegenden Fehler, muss der Testdurchlauf abgebrochen, evaluiert und allenfalls nötige Änderungen vorgenommen werden. Anschliessend muss der Testdurchlauf von vorne neu begonnen werden.

Testfall-Nr.		#1.0
<b>Anforderung</b>		Imker-Registrierung: Erfolgreich
<b>Beschreibung</b>		Ein Imker soll sich erfolgreich Registrieren können.
<b>Testvoraussetzung</b>		1. Nicht angemeldet sein 2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen
<b>Eingabe</b>		1. Firstname: Chris 2. Lastname: OConnor 3. Phone: 076 581 35 96 4. E-Mail: <a href="mailto:chris***@***.ch">chris***@***.ch</a> 5. Password: 9^xGakA+ 6. Confirm Password: 9^xGakA+ 7. AGB akzeptieren 8. Button «Register» klicken
<b>Erwartetes Resultat</b>		Der Imker wird angemeldet und auf <a href="http://localhost/ipa/public/profile">http://localhost/ipa/public/profile</a> weitergeleitet. Rechts in der Navigationsleiste wird die verwendete E-Mail Adresse angezeigt.

Testfall-Nr.		#1.1
<b>Anforderung</b>		Imker-Logout
<b>Beschreibung</b>		Ein Imker kann sich von der Applikation abmelden.
<b>Testvoraussetzung</b>		1. Angemeldet sein
<b>Eingabe</b>		1. Button «Logout» in der Navigationsleiste klicken
<b>Erwartetes Resultat</b>		User wird Ausgeloggt und auf <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> weitergeleitet.

Testfall-Nr.		#1.2
<b>Anforderung</b>		Imker-Registrierung: Leer Frontend
<b>Beschreibung</b>		Frontend required Validierung der Registrierung
<b>Testvoraussetzung</b>		1. Nicht angemeldet sein 2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen
<b>Eingabe</b>		1. Button «Register» klicken
<b>Erwartetes Resultat</b>		Browserfehlermeldung «Füllen Sie dieses Feld aus».

Testfall-Nr.	#1.3
Anforderung	Imker-Registrierung: Leer Backend
Beschreibung	Backend required Validierung der Registrierung.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> <li>3. Bei jedem Eingabefeld:           <ol style="list-style-type: none"> <li>a. Rechtsklick</li> <li>b. Element Untersuchen</li> <li>c. «required» löschen</li> </ol> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Sämtliche Eingabefelder werden rot umrandet und mit einer Fehlermeldung «Muss ausgefüllt sein» gekennzeichnet.

Testfall-Nr.	#1.4
Anforderung	Imker-Registrierung: Keine duplizierte E-Mail / Telefonnummern
Beschreibung	Mehrere User können nicht die selbe E-Mail oder Telefonnummer haben.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Testfall #01 erfolgreich durchgeführt</li> <li>2. Nicht angemeldet sein</li> <li>3. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Die Eingabefelder E-Mail und Tel. werden rot umrandet und die Fehlermeldung «[E-Mail/Tel.] wird bereits verwendet» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.5
Anforderung	Imker-Registrierung: E-Mail Validierung Frontend
Beschreibung	Als Imker muss ich eine gültige E-Mail-Adresse angeben
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: ajdgfs</li> <li>3. Phone: 076 581 35 97</li> <li>4. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Browserfehlermeldung «E-Mail-Adresse muss ein @ enthalten.»

Testfall-Nr.	#1.6
Anforderung	Imker-Registrierung: E-Mail Validierung Backend
Beschreibung	Als Imker muss ich eine gültige E-Mail-Adresse angeben
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> <li>3. Beim Eingabefeld E-Mail: <ul style="list-style-type: none"> <li>a. Rechtsklick</li> <li>b. Element Untersuchen</li> <li>c. type von «email» zu «text» ändern</li> </ul> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: ajdgfs</li> <li>3. Phone: 076 581 35 97</li> <li>4. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Eingabefeld E-Mail wird rot umrandet und die Fehlermeldung «E-Mail-Adresse ungültig» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.0
Anforderung	Imker-Registrierung: Passwort Validierung
Beschreibung	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Password: asd</li> <li>5. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Das Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort muss mind. 8 Stellen lang sein» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.1
Anforderung	Imker-Registrierung: Passwort Validierung
Beschreibung	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Passwort: hallohallo</li> <li>5. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Das Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort muss mind. 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer und mind. 1 Sonderzeichen enthalten» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.2
Anforderung	Imker-Registrierung: Passwort Validierung
Beschreibung	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Passwort: Hallohallo</li> <li>5. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Das Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort muss mind. 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer und mind. 1 Sonderzeichen enthalten» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.3
<b>Anforderung</b>	Imker-Registrierung: Passwort Validierung
<b>Beschreibung</b>	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
<b>Testvoraussetzung</b>	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
<b>Eingabe</b>	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Passwort: Hallohallo1</li> <li>5. Button «Register» klicken</li> </ol>
<b>Erwartetes Resultat</b>	Das Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort muss mind. 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer und mind. 1 Sonderzeichen enthalten» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.4
<b>Anforderung</b>	Imker-Registrierung: Passwort Validierung
<b>Beschreibung</b>	Pflicht / >= 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer, mind. 1 Sonderzeichen
<b>Testvoraussetzung</b>	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
<b>Eingabe</b>	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Passwort: Hallohallo\$</li> <li>5. Button «Register» klicken</li> </ol>
<b>Erwartetes Resultat</b>	Das Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort muss mind. 8 Stellen, mind. 1 Kleinbuchstabe, mind. 1 Grossbuchstabe, mind. 1 Ziffer und mind. 1 Sonderzeichen enthalten» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#1.7.5
Anforderung	Imker-Registrierung: Passwort Validierung Confirm Password
Beschreibung	Passwort und Passwort Bestätigung müssen übereinstimmen
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/register">http://localhost/ipa/public/register</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Alle Informationen identisch zu Testfall #1.0 eintragen</li> <li>2. E-Mail: <a href="mailto:test@test.ch">test@test.ch</a></li> <li>3. Phone: 076 581 35 97</li> <li>4. Password: Hallohallo\$1</li> <li>5. Password Confirm: TEST</li> <li>6. Button «Register» klicken</li> </ol>
Erwartetes Resultat	Das Confirm Passwortfeld wird rot umrandet und die Fehlermeldung «Das Passwort müssen übereinstimmen» erscheint. Alle Informationen vor der Formularübermittlung sind noch in den einzelnen Eingabefeldern vorhanden ausser die beiden Passwortfelder.

Testfall-Nr.	#2.0
Anforderung	Login: Falsche Angaben
Beschreibung	Login mit falschen Angaben darf nicht möglich sein.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Registriert sein als Imker (Testfall #1.0)</li> <li>2. Nicht angemeldet sein</li> <li>3. URL: <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Anmeldeformular Eingabe:             <ol style="list-style-type: none"> <li>a. E-Mail: <a href="mailto:chris***@***.ch">chris***@***.ch</a></li> <li>b. Password: 123</li> </ol> </li> <li>2. Button «Login» klicken</li> </ol>
Erwartetes Resultat	Fehlermeldung «E-Mail oder Passwort falsch» erscheint.

Testfall-Nr.	#2.1
Anforderung	Login: Mitarbeiter S&R
Beschreibung	Als Mitarbeiter von S&R möchte ich mich mit meinem üblichen Login anmelden können.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Nicht angemeldet sein</li> <li>2. URL: <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. E-Mail: admin@admin.ch</li> <li>2. Passwort: password</li> <li>3. Button «Login» klicken</li> </ol>
Erwartetes Resultat	Der Admin wird eingeloggt und auf die URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> weitergeleitet.

Testfall-Nr.	#2.2
Anforderung	Login: Imker
Beschreibung	Imker müssen sich anmelden können.
Testvoraussetzung	1. Registriert sein als Imker (Testfall #1.0) 2. Nicht angemeldet sein 3. URL: <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> aufrufen
Eingabe	1. Anmeldeformular Eingabe: a. E-Mail: <a href="mailto:chris***@***.ch">chris***@***.ch</a> b. Password: 9^xGakA+ 2. Button «Login» klicken
Erwartetes Resultat	Der Imker wird angemeldet und auf <a href="http://localhost/ipa/public/profile">http://localhost/ipa/public/profile</a> weitergeleitet.

Testfall-Nr.	#3.0
Anforderung	Imker-Zuständigkeiten: Temporär Hinzufügen nach PLZ
Beschreibung	Imker müssen ihre zuständigen Regionen auswählen können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. PLZ-Search Eingabe: 8700 2. 8700   Küsnacht anklicken 3. PLZ-Search Eingabe: 8610 4. 8610   User anklicken
Erwartetes Resultat	Ausgewählte PLZ 8700 Küsnacht & 8610 Uster sind in der linken Liste grün umrandet aufgeführt.

Testfall-Nr.	#3.1
Anforderung	Imker-Zuständigkeiten: Temporär Hinzufügen nach PLZ-Name
Beschreibung	Imker müssen ihre zuständigen Regionen auswählen können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. PLZ-Search Eingabe: Gossau 2. 8614   Gossau anklicken 3. PLZ-Search Eingabe: Meilen 4. 8706   Meilen anklicken
Erwartetes Resultat	Ausgewählte PLZ 8614 Gossau & 8706 Meilen sind in der linken Liste grün umrandet aufgeführt.

Testfall-Nr.	#3.2
Anforderung	Imker-Zuständigkeiten: Temporär Hinzufügen Rückgängig machen
Beschreibung	Imker müssen ihre zuständigen Regionen mutieren können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. PLZ-Search Eingabe: Gossau 2. 8614   Gossau anklicken 3. Delete Button in Linker Liste von 8614   Gossau klicken
Erwartetes Resultat	8614 Gossau wird nicht mehr aufgeführt.

Testfall-Nr.	#3.3
Anforderung	Imker-Zuständigkeiten: Suche nach nicht vorhandener PLZ
Beschreibung	Imker suche nach nicht vorhandener PLZ
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. Search Eingabe: «asdf»
Erwartetes Resultat	Keine Postleitzahl wird aufgeführt.

Testfall-Nr.	#3.4
Anforderung	Imker-Zuständigkeiten: Hinzufügen & Speichern
Beschreibung	Imker müssen ihre zuständigen Regionen auswählen & speichern können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. PLZ-Search Eingabe: 8700 2. 8700   Küsnacht anklicken 3. PLZ-Search Eingabe: Uster 4. 8610   User anklicken 5. PLZ-Search Eingabe: Gossau 6. 8614   Gossau anklicken 7. Speichern im Linken Formular klicken
Erwartetes Resultat	Ausgewählte PLZ 8700 Küsnacht, 8610 Uster & 8614 Gossau sind in der linken Liste aufgeführt.

Testfall-Nr.	#3.5
Anforderung	Imker-Zuständigkeiten: Temporär Löschen
Beschreibung	Imker müssen ihre zuständigen Regionen löschen können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. Delete Button von 8614 Gossau klicken
Erwartetes Resultat	Ausgewählte PLZ 8614 Gossau wird rot umrandet.

Testfall-Nr.	#3.6
Anforderung	Imker-Zuständigkeiten: Temporär Löschen Rückgängig machen
Beschreibung	Imker müssen ihre zuständigen Regionen mutieren können.
Testvoraussetzung	1. Als Imker angemeldet sein 2. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen
Eingabe	1. Delete Button von 8614 Gossau klicken 2. Delete Button von 8614 Gossau klicken
Erwartetes Resultat	Ausgewählte PLZ 8614 Gossau wird rot umrandet und anschliessend wieder in den Originalzustand gesetzt.

Testfall-Nr.	#3.7
Anforderung	Imker-Zuständigkeiten: Löschen
Beschreibung	Imker müssen ihre zuständigen Regionen löschen & speichern können.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Imker angemeldet sein</li> <li>URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Delete Button von 8614 Gossau klicken</li> <li>Speichern klicken</li> </ol>
Erwartetes Resultat	Ausgewählte PLZ 8614 Gossau wird nicht mehr in der Liste aufgeführt.

Testfall-Nr.	#3.8
Anforderung	Imker-Zuständigkeiten: Löschen & Hinzufügen gleichzeitig
Beschreibung	Imker müssen ihre zuständigen Regionen mutieren & speichern können.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Imker angemeldet sein</li> <li>URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Delete Button von 8610 Uster klicken</li> <li>PLZ-Search Eingabe: 870</li> <li>8702   Zollikon anklicken</li> <li>8708   Männedorf anklicken</li> <li>Speichern klicken</li> </ol>
Erwartetes Resultat	PLZ 8610 Uster wird nicht mehr in der Liste aufgeführt. PLZ 8702 Zollikon und 8708 Männedorf werden zusätzlich aufgeführt.

Testfall-Nr.	#3.9
Anforderung	Imker-Zuständigkeiten: Duplikat Suchen
Beschreibung	Imker können keine Region mehrfach auswählen.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Imker angemeldet sein</li> <li>URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>PLZ-Search Eingabe: 8702</li> </ol>
Erwartetes Resultat	Es wird keine PLZ unter den Suchergebnissen aufgelistet.

Testfall-Nr.	#4.0.0
Anforderung	S&R – Auftragserstellung: Erfolgreich
Beschreibung	Als Mitarbeiter von S&R möchte ich einen Auftrag erstellen können.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein (Testfall #2.1)</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 47.31901</li> <li>Longitude: 8.58059</li> <li>Firstname: Ueli</li> <li>Lastname: Meier</li> <li>Phone: 044 912 22 35</li> <li>Info: Unter dem Dach</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Übersicht des Auftrags wird angezeigt mit allen getätigten Eingaben. PLZ = 8700 Imker wird per SMS kontaktiert. (Testfall #01 Angegebene Nummer)

Testfall-Nr.	#4.0.1
Anforderung	S&R – Auftragserstellung: Tel Validierung
Beschreibung	Bei der Auftragserstellung wird die Telefonnummer überprüft auf die Richtigkeit damit Imker die Kontaktperson auch tatsächlich kontaktieren können.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein (Testfall #2.1)</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 47.31901</li> <li>Longitude: 8.58059</li> <li>Firstname: Ueli</li> <li>Lastname: Meier</li> <li>Phone: 111</li> <li>Info: Unter dem Dach</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Fehlermeldung Tel: «Bitte gültiges Format Eingeben: +41, 0041, 07x or 04x»

Testfall-Nr.	#4.1
Anforderung	S&R – Auftragserstellung: Frontend Required
Beschreibung	Auftragserstellung required Validierung.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	1. Button Create Contract klicken
Erwartetes Resultat	Browserfehlermeldung Eingabefeld Latitude und Longitude werden benötigt.

Testfall-Nr.	#4.2
Anforderung	S&R – Auftragserstellung: Backend Required
Beschreibung	Auftragserstellung required Validierung.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Beim Eingabefeld Latitude und Longitude:             <ol style="list-style-type: none"> <li>Rechtsklick</li> <li>Element Untersuchen</li> <li>required löschen</li> </ol> </li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Eingabefelder Latitude und Longitude werden rot umrandet und als required gekennzeichnet.

<b>Testfall-Nr.</b>	<b>#4.3.0</b>
<b>Anforderung</b>	S&R – Auftragserstellung: Input Validation Frontend
<b>Beschreibung</b>	Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
<b>Eingabe</b>	<ol style="list-style-type: none"> <li>1. Latitude: 46.00505</li> <li>2. Longitude: 7.00505</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>	<ol style="list-style-type: none"> <li>1. Latitude: Browser Fehlermeldung «Wert muss grösser als oder gleich 47 sein.»</li> <li>2. Longitude: Browser Fehlermeldung «Wert muss grösser als oder gleich 8 sein.»</li> </ol>

<b>Testfall-Nr.</b>	<b>#4.3.1</b>
<b>Anforderung</b>	S&R – Auftragserstellung: Input Validation Frontend
<b>Beschreibung</b>	Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
<b>Eingabe</b>	<ol style="list-style-type: none"> <li>1. Latitude: 48.39036</li> <li>2. Longitude: 9.49087</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>	<ol style="list-style-type: none"> <li>1. Latitude: Browser Fehlermeldung «Wert muss kleiner als oder gleich 47 sein.»</li> <li>2. Longitude: Browser Fehlermeldung «Wert muss kleiner als oder gleich 8 sein.»</li> </ol>

<b>Testfall-Nr.</b>	<b>#4.3.2</b>
<b>Anforderung</b>	S&R – Auftragserstellung: Input Validation Frontend
<b>Beschreibung</b>	Eingabeüberprüfung Latitude und Longitude..
<b>Testvoraussetzung</b>	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
<b>Eingabe</b>	<ol style="list-style-type: none"> <li>1. Latitude: lat</li> <li>2. Longitude: lon</li> </ol>
<b>Erwartetes Resultat</b>	Eingabe nicht möglich.

Testfall-Nr.	#4.3.3
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 46.99999</li> <li>Longitude: 7.99999</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>Latitude: Browser Fehlermeldung «Wert muss grösser als oder gleich 47 sein.»</li> <li>Longitude: Browser Fehlermeldung «Wert muss grösser als oder gleich 8 sein.»</li> </ol>

Testfall-Nr.	#4.3.4
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 47.00000</li> <li>Longitude: 8.00000</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Keine Browserfehlermeldung. Formular wird abgesendet.

Testfall-Nr.	#4.3.5
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 47.00001</li> <li>Longitude: 8.00001</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Keine Browserfehlermeldung. Formular wird abgesendet.

Testfall-Nr.	#4.3.6
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 47.99999</li> <li>Longitude: 8.99999</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Keine Browserfehlermeldung. Formular wird abgesendet.

Testfall-Nr.	#4.3.7
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 48.00000</li> <li>Longitude: 9.00000</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Keine Browserfehlermeldung. Formular wird abgesendet.

Testfall-Nr.	#4.3.8
Anforderung	S&R – Auftragserstellung: Input Validation Frontend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 48.00001</li> <li>Longitude: 9.00001</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>Latitude: Browser Fehlermeldung «Wert muss kleiner als oder gleich 48 sein.»</li> <li>Longitude: Browser Fehlermeldung «Wert muss kleiner als oder gleich 9 sein.»</li> </ol>

Testfall-Nr.	#4.4.0
Anforderung	S&R – Auftragserstellung: Input Validation Backend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>Untersuchen</li> <li>Min und Max Attribut löschen</li> </ol> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>Latitude: 46.00505</li> <li>Longitude: 7.00505</li> <li>Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>Latitude: Fehlermeldung «Wert muss mindestens 47 sein.»</li> <li>Longitude: Fehlermeldung «Wert muss mindestens 8 sein.»</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.4.1</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Input Validation Backend
<b>Beschreibung</b>		Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: 48.39036</li> <li>2. Longitude: 9.49087</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Latitude: Fehlermeldung «Wert darf nicht grösser sein als 48.»</li> <li>2. Longitude: Fehlermeldung «Wert darf nicht grösser sein als 9.»</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.4.2</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Input Validation Backend
<b>Beschreibung</b>		Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Type Attribut ändern zu text</li> </ol> </li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: lat</li> <li>2. Longitude: lon</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Latitude: Fehlermeldung «Muss eine Nummer sein.»</li> <li>2. Longitude: Fehlermeldung «Muss eine Nummer sein.»</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.4.3</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Input Validation Backend
<b>Beschreibung</b>		Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: 46.99999</li> <li>2. Longitude: 7.99999</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Latitude: Fehlermeldung «Wert muss mindestens 47 sein.»</li> <li>2. Longitude: Fehlermeldung «Wert muss mindestens 8 sein.»</li> </ol>

Testfall-Nr.	#4.4.4
Anforderung	S&R – Auftragserstellung: Input Validation Backend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:<ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Latitude: 47.00000</li> <li>2. Longitude: 8.00000</li> <li>3. Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>1. Latitude: Zulässig</li> <li>2. Longitude: Zulässig</li> <li>3. Fehlermeldung keine PLZ konnte zugewiesen werden.</li> </ol>

Testfall-Nr.	#4.4.5
Anforderung	S&R – Auftragserstellung: Input Validation Backend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:<ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Latitude: 47.00001</li> <li>2. Longitude: 8.00001</li> <li>3. Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>1. Latitude: Zulässig</li> <li>2. Longitude: Zulässig</li> <li>3. Fehlermeldung keine PLZ konnte zugewiesen werden.</li> </ol>

Testfall-Nr.	#4.4.6
Anforderung	S&R – Auftragserstellung: Input Validation Backend
Beschreibung	Eingabeüberprüfung Latitude und Longitude.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:<ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Latitude: 47.99999</li> <li>2. Longitude: 8.99999</li> <li>3. Button Create Contract klicken</li> </ol>
Erwartetes Resultat	<ol style="list-style-type: none"> <li>1. Latitude: Zulässig</li> <li>2. Longitude: Zulässig</li> <li>3. Fehlermeldung keine PLZ konnte zugewiesen werden.</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.4.7</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Input Validation Backend
<b>Beschreibung</b>		Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: 48.00000</li> <li>2. Longitude: 9.00000</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Latitude: Zulässig</li> <li>2. Longitude: Zulässig</li> <li>3. Fehlermeldung keine PLZ konnte zugewiesen werden.</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.4.8</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Input Validation Backend
<b>Beschreibung</b>		Eingabeüberprüfung Latitude und Longitude.
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> <li>3. Latitude und Longitude Element jeweils rechtsklick:             <ol style="list-style-type: none"> <li>a. Untersuchen</li> <li>b. Min und Max Attribut löschen</li> </ol> </li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: 48.00001</li> <li>2. Longitude: 9.00001</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Latitude: Fehlermeldung «Wert darf nicht grösser sein als 48.»</li> <li>2. Longitude: Fehlermeldung «Wert darf nicht grösser sein als 9.»</li> </ol>

<b>Testfall-Nr.</b>		<b>#4.5.0</b>
<b>Anforderung</b>		S&R – Auftragserstellung: Nominatim keine PLZ gefunden Fehler
<b>Beschreibung</b>		Nominatim Validierung
<b>Testvoraussetzung</b>		<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
<b>Eingabe</b>		<ol style="list-style-type: none"> <li>1. Latitude: 48</li> <li>2. Longitude: 8</li> <li>3. Button Create Contract klicken</li> </ol>
<b>Erwartetes Resultat</b>		<ol style="list-style-type: none"> <li>1. Fehlermeldung Keine PLZ konnte gefunden werden.</li> <li>2. Lat/Lon wird rot umrandet.</li> <li>3. PLZ Eingabefeld erscheint</li> </ol>

Testfall-Nr.	#4.5.1
Anforderung	S&R – Auftragserstellung: Manuelle PLZ Eingabe: Falsch
Beschreibung	Manuelle Eingabe der PLZ Validierung
Testvoraussetzung	1. Als Mitarbeiter von S&R eingeloggt sein 2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen
Eingabe	1. Latitude: 48 2. Longitude: 8 3. Button Create Contract klicken 4. PLZ: 8999 5. Button Create Contract klicken
Erwartetes Resultat	1. Fehlermeldung PLZ nicht valid. 2. PLZ Eingabefeld wird rot umrandet

Testfall-Nr.	#4.5.2
Anforderung	S&R – Auftragserstellung: Manuelle PLZ Eingabe: korrekt
Beschreibung	Manuelle Eingabe der PLZ Validierung
Testvoraussetzung	1. Als Mitarbeiter von S&R eingeloggt sein 2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen
Eingabe	1. Latitude: 48 2. Longitude: 8 3. Button Create Contract klicken 4. PLZ: 8700 5. Button Create Contract klicken
Erwartetes Resultat	Übersicht des Auftrags wird angezeigt. Ort = 8700 Küsnacht Imker wird per SMS kontaktiert. (Testfall #1.0 Angegebene Nummer)

Testfall-Nr.	#4.5.3
Anforderung	S&R – Auftragserstellung: Manuelle PLZ Eingabe überschreiben mit richtigen Lat/Lon Werten.
Beschreibung	Überschreibung der manuellen Eingabemöglichkeit der PLZ mit neuen Lat/Lon Werten.
Testvoraussetzung	1. Als Mitarbeiter von S&R eingeloggt sein 2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen
Eingabe	1. Latitude: 48 2. Longitude: 8 3. Button Create Contract klicken 4. Latitude: 47.31901 5. Longitude: 8.58059 6. Button Create Contract klicken
Erwartetes Resultat	Übersicht des Auftrags wird angezeigt. Ort = 8700 Küsnacht Imker wird per SMS kontaktiert. (Testfall #1.0 Angegebene Nummer)

Testfall-Nr.	#4.5.4
Anforderung	S&R – Auftragserstellung: Manuelle PLZ Eingabe überschreiben mit richtigen Lat/Lon Werten & PLZ.
Beschreibung	Überschreibung der manuellen Eingabemöglichkeit der PLZ mit neuen Lat/Lon Werten und gültiger PLZ.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Als Mitarbeiter von S&amp;R eingeloggt sein</li> <li>2. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Latitude: 48</li> <li>2. Longitude: 8</li> <li>3. Button Create Contract klicken</li> <li>4. Latitude: 47.31901</li> <li>5. Longitude: 8.58059</li> <li>6. PLZ: 8048</li> <li>7. Button Create Contract klicken</li> </ol>
Erwartetes Resultat	Übersicht des Auftrags wird angezeigt. Ort = 8048 Zürich

Testfall-Nr.	#5.0
Anforderung	Imker-Zusage: AGB nicht akzeptiert
Beschreibung	Die AGB muss zwingend akzeptiert werden vor der Annahme eines neuen Auftrags.
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Testfall #4.0 erfolgreich</li> <li>2. Contract von keinem anderen Imker bereits akzeptiert</li> <li>3. Angemeldet sein mit Testfall #1.0 Imker Konto</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Link aus SMS am Lokalen Computer eingeben</li> <li>2. AGB <b>nicht</b> akzeptieren</li> <li>3. Auftrag annehmen klicken</li> </ol>
Erwartetes Resultat	Fehlermeldung AGB muss akzeptiert sein.

Testfall-Nr.	#5.1
Anforderung	Imker-Zusage: Erfolgreich
Beschreibung	Erfolgreiche Imker Auftragsannahme
Testvoraussetzung	<ol style="list-style-type: none"> <li>1. Testfall #4.0 erfolgreich</li> <li>2. Contract von keinem anderen Imker bereits akzeptiert</li> <li>3. Angemeldet sein mit Testfall #1.0 Imker Konto</li> </ol>
Eingabe	<ol style="list-style-type: none"> <li>1. Link aus SMS am Lokalen Computer eingeben</li> <li>2. AGB akzeptieren</li> <li>3. Auftrag annehmen klicken</li> </ol>
Erwartetes Resultat	Alle Eingetragenen Informationen aus Testfall #4.0 werden angezeigt Übermittlung der Informationen per SMS + GoogleMapsPin in 8700 Küsnacht Bhf.

Testfall-Nr.	#5.2
Anforderung	Imker-Zusage: Nicht zugewiesener Imker ruft Auftrag auf
Beschreibung	Ein nicht für den Auftrag zugewiesener Imker ruft eine Auftragsannahme auf.
Testvoraussetzung	1. Testfall #5.1 erfolgreich 2. Anmelden mit anderem Imker Konto a. E-Mail: <a href="mailto:jon@doe.ch">jon@doe.ch</a> b. Passwort: password
Eingabe	1. Link aus SMS (Testfall #5.1) am Lokalen Computer eingeben
Erwartetes Resultat	Weiterleitung zur Index Seite.

Testfall-Nr.	#5.3
Anforderung	Imker-Zusage: Auftrag bereits vergeben
Beschreibung	Ein anderer Imker war schneller und hat den Auftrag bereits angenommen. Diesbezüglich ist dieser nicht mehr verfügbar.
Testvoraussetzung	1. Anmelden mit 2. Imker Konto a. E-Mail: jon@doe.ch b. Passwort: = password 2. PLZ 8700 zuweisen unter «Jurisdictions» 3. Testfall #4.0 ausführen 4. Testfall #5.1 ausführen 5. Anmelden mit Konto aus Schritt 1.
Eingabe	1. Neuer Möglicher Auftrag Link aus SMS am Lokalen Computer eingeben
Erwartetes Resultat	Weiterleitung zu Auftrag ist bereits vergeben.

Testfall-Nr.	#6.1
Anforderung	Imker-Search: PLZ
Beschreibung	Suche nach Imker der Region oder naheliegende anhand der Eingabe der PLZ
Testvoraussetzung	URL: <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> aufrufen
Eingabe	1. PLZ-Suche Eingabe: 8700 2. 8700   Küsnacht klicken
Erwartetes Resultat	8700   Küsnacht erscheint in der Sucheingabe. Imker werden der Reihe nach aufgrund der kürzesten Distanz aufgelistet (8700-Zugewiesene Region). Dabei sind Imker aus (Testfall #1.0, Testfall #5.3) zuoberst.

<b>Testfall-Nr.</b>		<b>#6.2</b>
<b>Anforderung</b>	Imker-Search	
<b>Beschreibung</b>	Suche nach Imker der Region oder naheliegende anhand der Eingabe des Namens des Ortes	
<b>Testvoraussetzung</b>	URL: <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a> aufrufen	
<b>Eingabe</b>	1. In der linken PLZ-Suche Eingabe: Küsnacht 2. 8700   Küsnacht klicken	
<b>Erwartetes Resultat</b>	8700   Küsnacht erscheint in der Sucheingabe. Imker werden der Reihe nach aufgrund der kürzesten Distanz aufgelistet (8700-Zugewiesene Region). Dabei sind Imker aus (Testfall #1.0, Testfall #5.3) zuoberst.	

<b>Testfall-Nr.</b>		<b>#7.0</b>
<b>Anforderung</b>	Middleware Imker: Profil	
<b>Beschreibung</b>	Nur ein registrierter Imker kann das Profil einsehen.	
<b>Testvoraussetzung</b>	1. Nicht angemeldet sein	
<b>Eingabe</b>	1. URL: <a href="http://localhost/ipa/public/profil">http://localhost/ipa/public/profil</a> aufrufen	
<b>Erwartetes Resultat</b>	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

<b>Testfall-Nr.</b>		<b>#7.1</b>
<b>Anforderung</b>	Middleware Admin	
<b>Beschreibung</b>	Nur ein registrierter Mitarbeiter von S&R (Admin) kann einen Auftrag erstellen.	
<b>Testvoraussetzung</b>	1. Nicht angemeldet sein	
<b>Eingabe</b>	1. URL: <a href="http://localhost/ipa/public/contract/create">http://localhost/ipa/public/contract/create</a> aufrufen	
<b>Erwartetes Resultat</b>	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

<b>Testfall-Nr.</b>		<b>#7.2</b>
<b>Anforderung</b>	Middleware Imker	
<b>Beschreibung</b>	Nur ein registrierter Imker kann seine zuständigen Regionen mutieren.	
<b>Testvoraussetzung</b>	1. Nicht angemeldet sein	
<b>Eingabe</b>	1. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen	
<b>Erwartetes Resultat</b>	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

<b>Testfall-Nr.</b>		<b>#7.3</b>
<b>Anforderung</b>	Middleware Imker	
<b>Beschreibung</b>	Nur ein registrierter Imker kann seine zuständigen Regionen mutieren.	
<b>Testvoraussetzung</b>	1. Angemeldet als S&R Mitarbeiter	
<b>Eingabe</b>	1. URL: <a href="http://localhost/ipa/public/jurisdiction">http://localhost/ipa/public/jurisdiction</a> aufrufen	
<b>Erwartetes Resultat</b>	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

Testfall-Nr.		#7.4
Anforderung	Middleware Imker	
Beschreibung	Nur ein registrierter Imker kann das Profil einsehen.	
Testvoraussetzung	1. Angemeldet als S&R Mitarbeiter	
Eingabe	1. URL: <a href="http://localhost/ipa/public/profil">http://localhost/ipa/public/profil</a> aufrufen	
Erwartetes Resultat	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

Testfall-Nr.		#7.5
Anforderung	Middleware Admin	
Beschreibung	Nur ein registrierter Mitarbeiter von S&R (Admin) kann einen Auftrag erstellen.	
Testvoraussetzung	1. Angemeldet als Imker (Testfall #01 Konto)	
Eingabe	1. URL: <a href="http://localhost/ipa/public/_contract/create">http://localhost/ipa/public/_contract/create</a> aufrufen	
Erwartetes Resultat	Weiterleitung zu <a href="http://localhost/ipa/public/">http://localhost/ipa/public/</a>	

## 16.8 ERKANNTE RISIKEN

Die erkannten Risiken werden nachfolgend aufgeführt mit entsprechenden Lösungsvorschlägen.

### 16.8.1 Nominatim

Nominatim wurde als Geo-API in der Aufgabenstellung deklariert. Die Reverse Geocoding Request API gilt als veraltet und wird womöglich beim nächsten grösseren Update entfernt.

#### 16.8.1.1 Mögliche Alternativen

##### 16.8.1.1.1 Google API

Als mögliche kostenpflichtige alternative, kann die Reverse Geocoding Request API von Google verwendet werden. Diese wird folgendermassen aufgerufen:

```
https://maps.googleapis.com/maps/api/geocode/json?latlng=<lat>,<lon>&key=API_KEY
```

(Google, 2022)

##### 16.8.1.1.2 Manuelle Eingabe PLZ

Die Eingabe der PLZ erfolgt als zusätzliche manuelle Aufgabe und ersetzt Nominatim.

### 16.8.2 Mailtrap

Mailtrap ist ein ukrainisches Unternehmen. Mit der aktuellen russischen Invasion könnte der Dienst allenfalls ausfallen und nicht mehr verfügbar sein.

#### 16.8.2.1 Mögliche Alternative

Ein aus einem anderen Projekt verwendeter Mailserver kann nötigenfalls für Testzwecke verwendet werden. Dazu muss lediglich in der «.env» Datei die Maileinstellungen angepasst werden.

## 17 ENTSCHEIDEN

---

Einige Entscheidungen wurden durch Vorgaben der Aufgabenstellung definiert und werden nicht reevaluiert.

### 17.1 MASSNAHMEN DER RISIKEN

#### 17.1.1 Nominativ

Im Rahmen der IPA wird Nominativ nach Definition in der Aufgabenstellung verwendet. Der Rückgabewert wird evaluiert. Sollte keine PLZ vorhanden sein, wird der Benutzer (S&R Personal) aufgefordert, die PLZ manuell zu ergänzen, um einen Auftrag zu erstellen.

#### 17.1.2 Mailtrap

Mailtrap wird weiterführend verwendet. Der Maildienst wird nur während und kurz nach der IPA benötigt, um den Mailverkehr zu Testen und Demonstrieren.

### 17.2 DATENBANKMODELL

Das Datenbankmodell v02 gilt als Update von v01 und wird für die Realisierung verwendet.

### 17.3 IMKERSUCHE

Die nächstgelegenen Imker werden nicht mit PLZ +/- 5 gesucht, bis ein passender Imker gefunden wurde. Die Imker werden sortiert nach Differenz aus nächstgelegener Region zu gesuchter PLZ.

## 18 REALISIEREN

---

### 18.1 SETUP

#### 18.1.1 Laravel

Ein neues Laravel Projekt wurde mit Hilfe von Composer in der der Git Bash Console erstellt:

```
composer create-project laravel/laravel:^8.0 ipa
cd ipa
composer require laravel/ui
php artisan ui bootstrap
php artisan ui bootstrap --auth
npm install
npm run dev
```

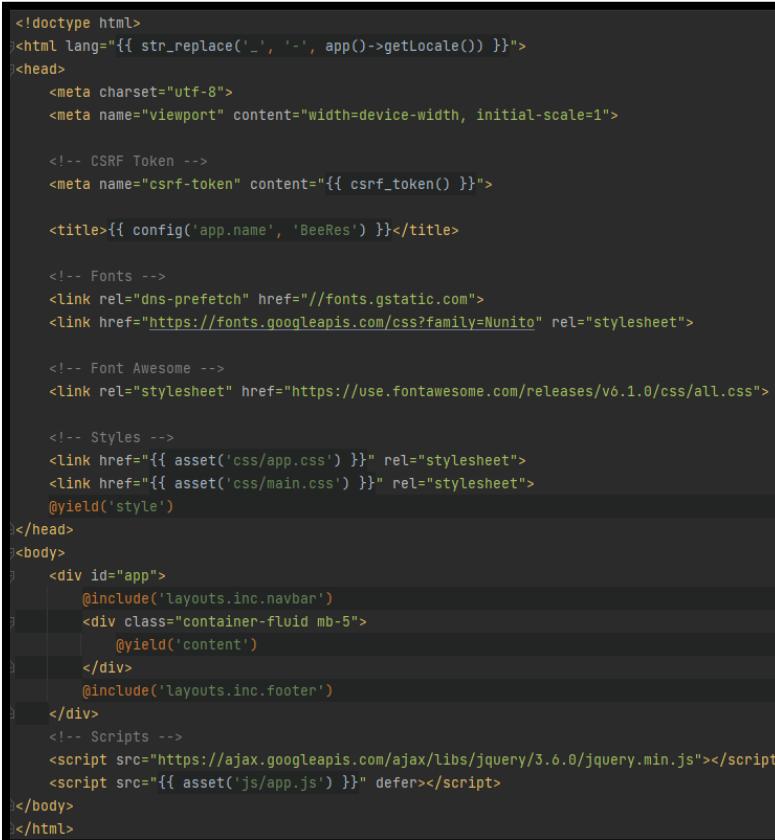
Dabei wurde auch gleich Bootstrap installiert und eine bereits bestehende Authentifikation.

#### 18.1.2 Datenbank erstellen

Eine neue leere Datenbank wurde via phpMyAdmin erstellt.

#### 18.1.3 Blade Template Layout

Das Blade Template wird für alle Views der Applikation verwendet. Die Navbar und der Footer werden dazu inkludiert und der Content von den einzelnen Views wird im «@yield('content')» hinzugefügt.



```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()>getLocale()) }}>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}>

    <title>{{ config('app.name', 'BeeRes') }}</title>

    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.13/css/all.css">

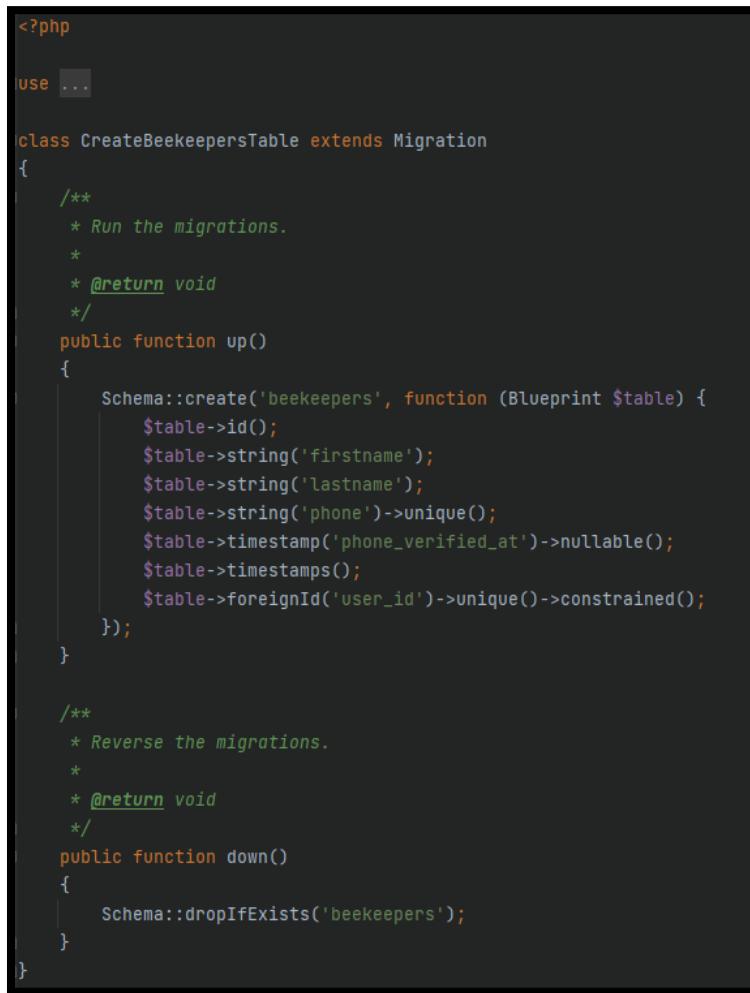
    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('css/main.css') }}" rel="stylesheet">
    @yield('style')
</head>
<body>
    <div id="app">
        @include('layouts.inc.navbar')
        <div class="container-fluid mb-5">
            @yield('content')
        </div>
        @include('layouts.inc.footer')
    </div>
    <!-- Scripts -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="{{ asset('js/app.js') }}" defer></script>
</body>
</html>
```

Abbildung 14 Blade Templat Layout

## 18.2 DATENBANK MIGRATIONS

Das ERD v02 welches in der Planung kreiert wurde, gab vor, wie die Datenbank zu erstellen ist. Für jede Tabelle wurde eine create Migration erstellt und die Attribute der Table definiert:

```
php artisan make:migration create_beekeepers_table
```



```
<?php

use ...

class CreateBeekeepersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('beekeepers', function (Blueprint $table) {
            $table->id();
            $table->string('firstname');
            $table->string('lastname');
            $table->string('phone')->unique();
            $table->timestamp('phone_verified_at')->nullable();
            $table->timestamps();
            $table->foreignId('user_id')->unique()->constrained();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('beekeepers');
    }
}
```

Abbildung 15 Imker Datenbank Migration

### 18.3 MODELS ERSTELLEN

Die Models können direkt aus der Datenbank generiert werden. Dazu wird eine externe Library benötigt. Diese kann mit dem nachfolgenden Befehl via Composer hinzugefügt werden:

```
composer require krlove/eloquent-model-generator
```

Um die Models zu generieren, wird folgender Befehl für jedes Model ausgeführt:

```
php artisan krlove:generate:model <Modelname> --table-name=<Tablename>
```

Reine n:m Verbindungstabellen werden im Laravel nicht als eigenständiges Model benötigt.

Die generierten Models müssen angepasst sowie überprüft werden. Vor allem bei den Relations müssen Anpassungen vorgenommen werden:

```
/**
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function beekeeper_old()
{
    return $this->hasMany( related: 'App\Models\Beekeeper' );
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\HasOne
 */
public function beekeeper()
{
    return $this->hasOne( related: Beekeeper::class );
}
```

Abbildung 16 Model-Relation Anpassung

### 18.4 FACTORY & SEEDER

Um die Datenbank mit Mockdaten zu füllen, benötigt Laravel Factories und Seeders. Diese können mit nachkommendem Befehl ausgeführt werden:

```
php artisan migrate:fresh --seed
```

### 18.4.1 Factory

In der Factory wird ein Blueprint erstellt. Dieser gibt vor, was für einen zufälligen Wert in welchem Attribut generiert werden soll. Die Factory für das Model Contract sieht wie folgt aus:

```
class ContractFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = Contract::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'lon'           => rand(470000, 480000) / 10000,
            'lat'           => rand( 8000, 9000) / 10000,
            'contact_firstname' => $this->faker->firstName,
            'contact_lastname' => $this->faker->lastName,
            'contact_phone'   => rand(41760000000, 41799999999),
            'info'          => $this->faker->realText( maxNbChars: 60 ),
            'postcode_id'   => Postcode::all()->random()->id,
            'created_by'    => User::all()->where( key: 'is_admin' )->random()->id,
            'beekeeper_id'  => rand(0,3) == 1 ? null : Beekeeper::all()->random()->id,
        ];
    }
}
```

Abbildung 17 Contract factory

Definition	Erklärung
Lon/Lat	Die Lon/Lat Werte für die Applikation liegen zwischen 47.0000 - 48.0000 und 8.0000 – 9.0000
Contact_phone	Laravel Faker kennt zwar die Möglichkeit, eine Telefonnummer zufällig zu generieren, diese ist aber nicht wie gewünscht formatiert. Diesbezüglich wird eine zufällige Mobilnummer mit Schweizer Vorwahl aus einem Zahlenbereich erstellt.
Beekeeper_id	Contracts können als noch nicht zugewiesen gelten. Dazu wird beekeeper_id auf null gesetzt. Um dies beim Zufälligen generieren auch zu haben, wird zufällig ca. jede dritte Beekeeper_id im Contract auf null gesetzt. Ansonsten wird zufällig ein Beekeeper ausgesucht.

Damit ein Model weißt, dass es eine Factory hat, muss dies im Model explizit als Trait gekennzeichnet werden:

```
class Contract extends Model
{
    use HasFactory;
```

Abbildung 18 HasFactory Model Trait

### 18.4.2 Seeder

Der Seeder ist dafür zuständig, um Objekte zu erstellen anhand der Definitionen der Factory. Der DatabaseSeeder ruft dabei alle untergeordneten Seeder auf:

```
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call([
            UserSeeder::class,
            BeekeeperSeeder::class,
            PostcodeSeeder::class,
            ContractSeeder::class,
            BeekeeperPostcodeSeeder::class,
            BeekeeperContractSeeder::class,
        ]);
    }
}
```

Abbildung 19 Main Database Seeder

Bei den Beekeepers muss zuerst ein User erstellt und dann zugewiesen werden. Dies kann mit dem Seeder folgend umgesetzt werden:

```
class BeekeeperSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        User::factory()->count( count: 20)->has(Beekeeper::factory())->create();
    }
}
```

Abbildung 20 Beekeeper Account Seeder

Die beekeeper\_contract Table wird folgend gefüllt:

```

class BeekeeperContractSeeder extends Seeder
{
    /**
     * Add random 1-5 applicable contracts to each beekeeper
     * Add all random assigned Contracts to also be applicable
     *
     * @return void
     */
    public function run()
    {
        foreach (Beekeeper::all() as $beekeeper) {

            $j = rand(1, 5);

            for($i = 0; $i < $j; $i++) {
                $beekeeper->contracts_applicable()->attach(Contract::all()->random());
            }

            $beekeeper->contracts_applicable()->attach($beekeeper->contracts);
        }
    }
}

```

Abbildung 21 Beekeeper Contract Seeder

Dabei werden alle Imker iteriert und für jeden zufällig 1-5 Aufträge als applicable zugewiesen. Applicable bedeutet, der erstellte Auftrag ist in einer vom Imker selektierten Region und er wurde per SMS, kontaktiert aber noch nicht definitiv zugeteilt. Am Ende der Iteration werden zusätzlich alle definitiv zugewiesene Aufträge dem Imker als applicable hinzugefügt.

Mitarbeiter von S&R werden als User erstellt und erhalten das Attribut is\_admin als true:

```

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        User::factory(['is_admin' => 1])->count( count: 3)->create();
    }
}

```

Abbildung 22 S&R Admin Account Seeder

Die Postleitzahlen vom Dokument «Liste-der-PLZ-in-der-Schweiz.xlsx» wurden in eine neue Datenbank importiert. Danach wurden alle PLZ des Kantons Zürich selektiert und exportiert in den folgenden Seeder:

Dadurch werden gleich auch alle PLZ des Kanton Zürichs für die Datenbank jeweils statisch miterstellt.

```

class PostcodeSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('postcodes')->insert([
            ["postcode"=>8001, "name"=>"Zürich"],
            ["postcode"=>8002, "name"=>"Zürich"],
            ["postcode"=>8003, "name"=>"Zürich"],
            ["postcode"=>8004, "name"=>"Zürich"],
            ...
        ]);
    }
}

```

Abbildung 23 Postcode Seeder

## 18.5 USER AUTHENTIFIKATION

### 18.5.1 Laravel User Authentifikation

Laravel verfügt bereits über eine User Authentifikation. Dabei wird im Trait des Auth/LoginControllers beim Login request in der Datenbank überprüft, ob die Login credentials in der Datenbank existieren. Wenn dies der Fall ist, wird in der Session der Zeitpunkt, wann das Passwort confirmed wurde, gespeichert.

### 18.5.2 Login Redirect

Ein Mitarbeiter von S&R soll auf die Seite «contract/create» und ein Imker auf «/profile» weitergeleitet werden. Dies kann im Auth/LoginController definiert werden, in dem die vordefinierte variable «\$redirectTo» mit einer gleichnamigen Methode ersetzt wird: (Rahman, 2020)

```
public function redirectTo()
{
    return Auth::user()->is_admin ? '/contract/create':'/profile';
}
```

Abbildung 24 Login Rolebased redirect

## 18.6 IMKER REGISTRIERUNG

Die Registrierung wurde mittels folgender Schritte implementiert.

### 18.6.1 Components

Die Formular Eingabe Felder weisen immer die gleichen Eigenschaften auf diesbezüglich wurde ein Component erstellt für ein normales Eingabefeld sowie ein Component für das Passwortfeld.

Ein Component kann mittels folgenden Befehls erstellt werden:

```
php artisan make:component Forms/Input
```

Dabei wird ein Blade File erstellt sowie ein Model:

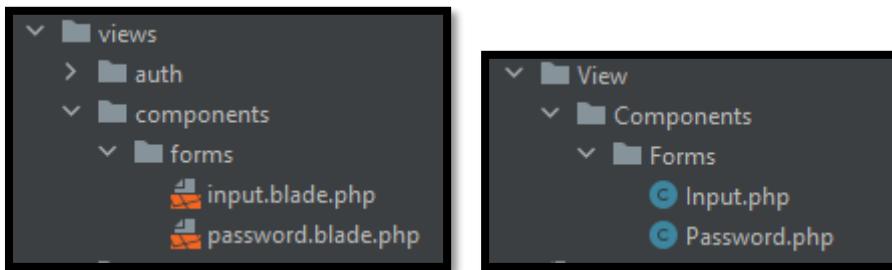


Abbildung 25 Component Blade und Model

Im Model werden die benötigten Variablen definiert und im Konstruktor zugewiesen:

```
class Input extends Component
{
    public $type;
    public $name;
    public $title;

    /**
     * Create a new component instance.
     *
     * @return void
     */
    public function __construct($type, $name, $title)
    {
        $this->type = $type;
        $this->name = $name;
        $this->title = $title;
    }

    /**
     * Get the view / contents that represent the component.
     *
     * @return \Illuminate\Contracts\View\View|\Closure|string
     */
    public function render()
    {
        return view('components.forms.input');
    }
}
```

Abbildung 26 Component Model

In der Blade Datei können diese dann ausgegeben werden:

```
<div class="form-group">

    <label class="form-label mt-4" for="{{ $name }}>{{ $title }}<span class="text-danger">*</span></label>

    <input class="form-control @if($errors->any()) @error($name) is-invalid @else is-valid @enderror @endif"
        type="{{ $type }}" name="{{ $name }}" id="{{ $name }}" placeholder="{{ $title }}" value="{{ old($name) }}>

    @error($name)
        <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>
```

Abbildung 27 Component Blade

Sollte zukünftig ein Änderungsbedarf entstehen, muss lediglich eine Änderung im Component gemacht werden. Zudem ist das Formular für die Registrierung wesentlich verständlicher:

```
<form method="POST" action="{{ route('register') }}">
    @csrf
    @method('POST')

    <x-forms.input type="text" name="firstname" title="Firstname" />
    <x-forms.input type="text" name="lastname" title="Lastname" />
    <x-forms.input type="text" name="phone" title="Phone" />
    <x-forms.input type="email" name="email" title="E-Mail" />
    <x-forms.password confirm="true" />

    <input class="btn btn-block btn-primary mt-3" type="submit" value="Register" />

</form>
```

Abbildung 28 Formular mit Components

### 18.6.2 Input Validation

Das Formular übermittelt die eingegebenen Informationen an den Auth/RegisterController.php. Bei der Telefonnummer muss vor der Validierung alle Leerschläge sowie Trennzeichen (-) entfernt werden und ins richtige Format formatiert werden.

```
protected function validator(array $data)
{
    $data['phone'] = formatPhoneNum($data['phone']);

    return Validator::make($data, [
        'firstname' => ['required', 'string', 'max:255'],
        'lastname' => ['required', 'string', 'max:255'],
        'phone' => ['required', 'regex:/^41\d{9}$/, 'unique:beekeepers'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:8', 'regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[\$\#%^&*-]).{8,}$/', 'confirmed'],
        'agb' => ['required']
    ], [
        'phone.regex' => 'Not a valid Format, please use: +41, 0041, 07x or 044',
        'password.regex' => 'Password must contain at least 1x Uppercase, 1x Lowercase, a number and a special character!'
    ]);
}
```

Abbildung 29 Registrierung Inputvalidierung

Für die Telefonnummer sowie das Passwort wurde eine Regex Validation erstellt:

#### 18.6.2.1 Telefonnummer

Die Eingabe der Telefonnummer kann möglicherweise auf verschiedene Arten geschehen. Zum einen können Leerschläge oder das Trennzeichen «-» Vorkommen. Diese werden vorab entfernt. Zum anderen gibt es verschiedene mögliche Eingabevarianten mit und ohne CH-Vorzeichen etc. In dieser Applikation sind folgende Formate möglich: 0765813596, +41765813596, 0041765813596. Die SMS-API Vonage erwartet als gültige Telefonnummer folgendes Format: 41765813596. Diesbezüglich werden die Telefonnummern formatiert, bevor sie validiert werden.

Nachfolgend die erstellte Regex Überprüfung mit Testfällen:

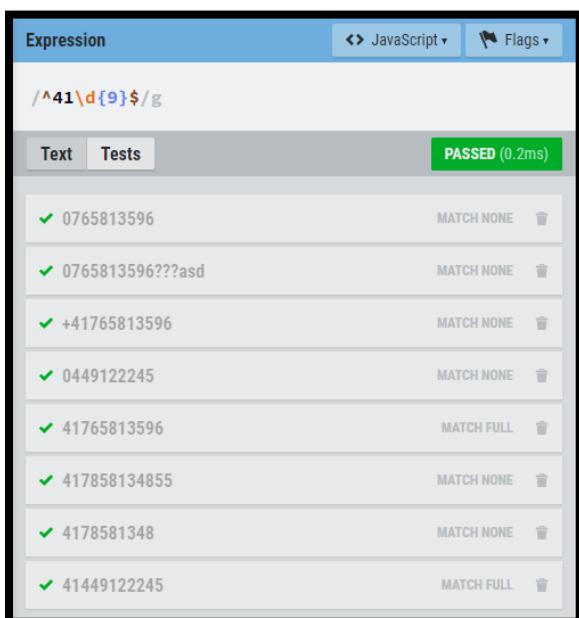


Abbildung 30 Telefonnummer Regex Überprüfung

### 18.6.2.2 Passwort

Das Passwort muss mindestens einen Grossbuchstaben, ein Kleinbuchstabe, eine Zahl sowie ein Sonderzeichen enthalten und muss mindestens acht Zeichen lang sein. Dies kann auch mittels Regex überprüft werden:

The screenshot shows a regex testing interface with the following details:

- Expression:** `/^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*[#?!.@$/%^&*-]).{8,}$/.g`
- Status:** PASSED (0.5ms)
- Text Tests:**
  - 123: MATCH NONE
  - Hallo: MATCH NONE
  - HALLO: MATCH NONE
  - hallo\$: MATCH NONE
  - Hallo\$: MATCH NONE
  - Haloo\$1: MATCH ANY
  - Hallo!1: MATCH ANY
  - Haloo\$1: MATCH NONE

Abbildung 31 Passwort Regex Überprüfung

(Tgugnani, 2018)

### 18.6.2.3 Validation Failed

Beim Fehlschlag einer oder mehrerer Validierungsregeln wird ein Errorbag erstellt und zurückgegeben. Aus dem Errorbag können die fehlgeschlagenen Felder ausgelesen werden sowie die definierte Error-Nachricht. Die vordefinierten Error-Nachrichten können im zweiten Index der Validierung überschrieben werden, wie beispielsweise «phone.regex»

The screenshot shows a registration form titled "Beekeeper-Register" with the following validation errors:

- Phone\***: +1 (677) 756-8127 (highlighted in red) - Not a valid Format, please use: +41, 0041, 07x or 044
- Password\***: (highlighted in red) - Password must contain at least 1x Uppercase, 1x Lowercase, a number and a special character!

The other fields (Firstname, Lastname, E-Mail, Confirm Password, AGB Accepted) appear to be valid.

Abbildung 32 Register Error Nachrichten

### 18.6.3 Telefonnummer-Formater

Die Eingabe der Telefonnummer kann möglicherweise auf verschiedene Arten geschehen. Die Funktion, um die Telefonnummer zu Formatieren könnte möglicherweise an mehreren Stellen in der Applikation verwendet werden, diesbezüglich wurde ein «app/helpers.php» File erstellt. Dieses wird im «composer.json» hinzugefügt.

```
"autoload": {  
    "psr-4": {  
        "App\\": "app/",  
        "Database\\Factories\\": "database/factories/",  
        "Database\\Seeders\\": "database/seeders/"  
    },  
    "files": [  
        "app/helpers.php"  
    ]  
}
```

Abbildung 33 Composer.json

Dadurch stehen sämtliche Funktionen innerhalb dieses Files jederzeit der ganzen Applikation zur Verfügung. Leer und Trennzeichen werden mit str\_replace entfernt:

```
function replaceEmptyChars($phone): string  
{  
    $phone = str_replace( search: ' ', replace: '', $phone);  
    $phone = str_replace( search: '-', replace: '', $phone);  
  
    return $phone;  
}
```

Abbildung 34 Telefonnummer Entfernen von Leer und Trennzeichen

Die Telefonnummer selbst muss beim Speichern in der Datenbank auf das einheitliche Format: 41765813596 formatiert werden. In einem Switch werden die ersten beiden Zeichen analysiert und anhand des entsprechenden Cases formatiert. Beim Case '07' und '04' muss dasselbe Geschehen diesbezüglich ist der Case '07' leer. Dadurch, dass kein Break enthalten ist, wird der nächste Case auch ausgeführt:

```
function formatPhoneNum($phone): string
{
    $phone = replaceEmptyChars($phone);

    $phone = str_replace( search: '+', replace: '', $phone);

    switch (substr($phone, offset: 0, length: 2)) {

        case "00" : $phone = substr($phone, offset: 2); break;
        case "07" :
        case "04" : $phone = '41' . substr($phone, offset: 1); break;

    }
    return $phone;
}
```

Abbildung 35 Telefonnummer Formatierung

```
PS C:\xampp\htdocs\ipa> php artisan tinker
Psy Shell v0.11.2 (PHP 7.4.11 - cli) by Justin Hileman
>>> formatPhoneNum('+41765813596')
=> "41765813596"
>>> formatPhoneNum('0041765813596')
=> "41765813596"
>>> formatPhoneNum('0765813596')
=> "41765813596"
>>> formatPhoneNum('0445813596')
=> "41445813596"
```

Abbildung 36 Telefonnummer Formatierung Überprüfung

#### 18.6.4 Create Beekeeper/User

Nachdem die Validation erfolgreich war, wird im gleichen Controller «Auth/RegisterController» der Beekeeper erstellt. Damit ein Beekeeper erstellt werden kann, muss jedoch zuerst ein User erstellt und anschliessend dem Beekeeper zugewiesen werden. Beim Erstellen des Users wird zudem auch das Passwort gehasht.

```
protected function create(array $data)
{
    $user = User::create([
        'email'      => $data['email'],
        'password'   => Hash::make($data['password']),
    ]);

    Beekeeper::create([
        'firstname'  => $data['firstname'],
        'lastname'   => $data['lastname'],
        'phone'      => formatPhoneNum($data['phone']),
        'user_id'    => $user->id,
    ]);

    return $user;
}
```

Abbildung 37 Create Beekeeper und User

Der User wird für hierbei zurückgegeben, um gleich als dessen eingeloggt zu werden.

## 18.7 MIDDLEWARE

Middlewares bieten die Möglichkeit Routes mit einer Validierung zu schützen. Für diese Applikation wird einerseits eine Middleware benötigt, um zu überprüfen, ob der Benutzer eingeloggt ist. Eine spezifisch, um zu überprüfen, ob es sich um einen Imker handelt sowie eine, um zu überprüfen, ob es sich um einen Mitarbeiter von S&R handelt.

Eine neue Middleware kann erstellt werden mit folgendem Befehl:

```
php artisan make:Middleware BeekeeperOnly
```

Dabei wird ein neues File im Verzeichnis App\Http\Middleware erstellt.

Bei der Middleware «BeekeeperOnly» wird überprüft, ob der bereits Authentifizierter User ein Beekeeper ist:

```
public function handle(Request $request, Closure $next)
{
    if(auth()->user()->beekeeper) {
        return $next($request);
    }

    return redirect(route('index'));
}
```

Abbildung 38 BeekeeperOnly Middleware

Diese Middleware muss zusätzlich unter «app\Http\Kernel.php» Registriert werden als Route-Middleware:

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    'beekeeperOnly' => \App\Http\Middleware\BeekeeperOnly::class,
];
```

Abbildung 39 Middleware Kernel Registrierung

Nach dem gleichen Prinzip wurde auch die Middleware für «AdminOnly» erstellt.

In den Routes können nun die Middlewares verwendet werden:

```
Route::group(['middleware' => 'auth'], function() {
    Route::group(['middleware' => 'beekeeperOnly'], function() {
        Route::get('profile', [App\Http\Controllers\BeekeeperController::class, 'profile'])->name('profile');
    });
    Route::group(['middleware' => 'adminOnly'], function() {
        //
    });
});
```

Abbildung 40 Middleware Routes

Die Middleware «BeekeeperOnly» sowie «AdminOnly» setzt einen authentifizierten User voraus für die Validierung, diesbezüglich muss die Middleware umschlossen werden von der Auth-Middleware.

Alle zukünftigen Routes können nun entweder nur unter Auth für S&R Mitarbeiter und Imker hinterlegt werden oder unter BeekeeperOnly für nur Imker respektive unter AdminOnly für nur Mitarbeiter S&R.

Die Route Auth muss noch abgeändert werden, da das Login nicht standardmäßig unter /login sondern auf der Index Seite ist. Bei einem Fehlschlag der Middleware soll der User diesbezüglich auf die Indexseite redirected werden:

```
class Authenticate extends Middleware
{
    /**
     * Get the path the user should be redirected to when they are not authenticated.
     *
     * @param \Illuminate\Http\Request $request
     * @return string|null
     */
    protected function redirectTo($request)
    {
        if (! $request->expectsJson()) {
            return route('index');
        }
    }
}
```

Abbildung 41 Auth Middleware redirect Anpassung

## 18.8 PROFILE

Um zur Zuständigkeit zu gelangen, muss der Imker via Profil auf den Button «PLZ - Jurisdiction» klicken. Diesbezüglich wurde ein Profil-Seite erstellt.

Damit im Profil die bekannten Informationen ausgewiesen werden, musste das Component angepasst werden, um im Profil sowie bei der Registrierung verwendet werden zu können. Dabei wurde der vorhin im Component deklarierte value=>old()</value> jetzt neu als zusätzliche variable übergeben:

Register:

```
<x-forms.input type="text" name="firstname" title="Firstname" value="{{ old('firstname') }}" />
<x-forms.input type="text" name="lastname" title="Lastname" value="{{ old('lastname') }}" />
<x-forms.input type="text" name="phone" title="Phone" value="{{ old('phone') }}" />
<x-forms.input type="email" name="email" title="E-Mail" value="{{ old('email') }}" />
```

Abbildung 43 Register Components anpassungen

Profil:

```
<x-forms.input type="text" name="firstname" title="Firstname" value="{{ $beekeeper->firstname }}" />
<x-forms.input type="text" name="lastname" title="Lastname" value="{{ $beekeeper->lastname }}" />
<x-forms.input type="text" name="phone" title="Phone" value="{{ $beekeeper->phone }}" />
<x-forms.input type="email" name="email" title="E-Mail" value="{{ $beekeeper->user->email }}" />
```

Abbildung 44 Profil Components

Das Backend vom Profil wird nicht während der IPA realisiert.

The screenshot shows a registration form titled 'PROFILE'. It includes fields for Firstname, Email, Lastname, Phone, and E-Mail, all populated with placeholder values. Below these are fields for Old Password, Password, and Confirm Password. At the bottom are two buttons: a green 'Update' button and a red 'Delete' button. A note at the top right says 'Wird nicht im Rahmen der IPA realisiert.'

Abbildung 42 Profile

## 18.9 ZUSTÄNDIGKEIT (IMKER PLZ)

Für die Zuständigkeit wurde eine neue View unter «/resources/views/models/beekeeper/jurisdiction.blade.php» sowie ein neuer Controller «BeekeeperController» erstellt.

Die View wird mittels Bootstrap Cols in zwei Hälften unterteilt:

```
<div class="row" style="...">
    <!-- LEFT SIDE -->
    <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
        <div class="h-100 w-100">
            </div>
    </div>

    <!-- RIGHT SIDE -->

    <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
        <div class="h-100 w-100">
            </div>
    </div>
```

Abbildung 45 Bootstrap View Teilung

### 18.9.1 Zugewiesene PLZ-Liste

Die Liste der aktuell zugewiesenen PLZ wurde im linken Teil implementiert. Dazu wird vom Controller alle aktuell zugewiesenen Postcodes überwiesen und in einer Schleife dargestellt:

```
<div class="list-group" id="currentJur">
    @foreach($postcodes as $postcode)
        <div class="list-group-item list-group-item-action">
            {{ $postcode->postcode . ' | ' . $postcode->name }}</div>
    @endforeach
</div>
```

Abbildung 46 Zugewiesene Postcodes im Blade



Abbildung 47 Zugewiesene Postcodes Darstellung

### 18.9.2 Zugewiesene PLZ-Liste mutieren

Die Liste wurde einem Formular hinzugefügt sowie zwei hidden «div»:

```
<form action="{{ route('jurisdiction.update') }}" method="POST" id="jurisdictionForm">
    @csrf
    @method('POST')

    <div class="d-none" id="delJur"></div>
    <div class="d-none" id="addJur"></div>
```

Abbildung 48 Hidden <div> für Postcodes im Formular

Den einzelnen Listeneinträgen wurde jeweils ein Delete Button hinzugefügt. Wird dieser getätigert, wird eine JavaScript Funktion ausgeführt:

```
@foreach($postcodes as $postcode)
    <div class="list-group-item list-group-item-action">
        {{ $postcode->postcode . ' | ' . $postcode->name }}
        <button type="button" class="btn btn-danger float-end" onclick="del({{ $postcode->id }}, this.parentElement)">X</button>
    </div>
@endforeach
```

Abbildung 49 Listenelemente Hinzufügung eines Deletebuttons

Mittels der JS Funktion wird ein neues Input-Element erstellt und dem div «delJur» angehängt sowie das zu Löschende Listenelement Rot umrandet:

```
el.style.border = '1px solid red';

let input = document.createElement('input');
input.setAttribute('type', 'hidden');
input.setAttribute('name', 'delJur[]');
input.setAttribute('value', id);

delJur.append(input);
```

Abbildung 50 JS neues Input Element erstellen

```

function del(id, el) {
    const delJur = document.getElementById('delJur');

    let duplicate = false;

    document.getElementsByName('delJur[]').forEach(element => {
        if(element.value == id) {
            duplicate = true;
            element.remove();
        }
    });

    if(duplicate) {
        el.style.border = '1px solid rgba(0, 0, 0, 0.125)';
    } else {
        el.style.border = '1px solid red';
    }

    let input = document.createElement('input');
    input.setAttribute('type', 'hidden');
    input.setAttribute('name', 'delJur[]');
    input.setAttribute('value', id);

    delJur.append(input);
}

}

```

Abbildung 51 JS Delete Funktion

Damit die Löschung rückgängig gemacht werden kann, wird vorab validiert, ob ein zu löschenes Input-Element mit derselben Postcode id bereits existiert. Ist dies der Fall, möchte der User seine Löschung der PLZ Rückgängig machen. Dazu wird das vorherige erstellte hidden Input Element entfernt und der Postcode in der visuellen Liste der rote Rahmen zurückgesetzt.

Beim Übermitteln des Formulars werden nun alle zu löschenen Postcodes IDs im Array «delJur[]» übergeben.

### 18.9.3 PLZ hinzufügen

Im rechten Teil der View wird eine LiveSearch implementiert. Mit dieser soll es möglich sein, Postleitzahlen zu suchen und dem linken Formular hinzuzufügen.

#### 18.9.3.1 LiveSearch

Für die LiveSearch wird zuerst ein Eingabefeld erstellt sowie ein <div>, um die Suchresultate hinzuzufügen:

```

<input type="text" class="form-control" id="searchInput" onkeyup="search(this)" placeholder="PLZ Search" autocomplete="off">

<div class="list-group overflow-auto" style="max-height: 15.5rem" id="searchOutput"></div>

```

Abbildung 52 LiveSearch HTML Input und Output

Die LiveSearch wird mittels einer Ajax Request implementiert:

```

function search (e) {
    $.ajax({
        type : 'get',
        url : '{{ route('search.plz') }}',
        data : {search: e.value},
        success: function (data) {
            //
        }
    });
}

```

Abbildung 53 Ajax Request

Bei jedem KeyUp des Inputfeldes wird eine get-Request an die Route «search.plz» getätigert und der Eingabewert übermittelt.

Ein neuer Controller wurde erstellt unter «http/Controller/SearchController». Dieser empfängt den Eingabewert und returniert das Suchergebnis der übereinstimmenden Postleitzahlen:

```
class SearchController extends Controller
{
    public function searchPLZ(Request $request)
    {
        if($request->input('key: 'search')) {

            $data = DB::table('table: 'postcodes')
                ->where('column: 'postcode', 'operator: 'LIKE', 'value: %' . $request->input('key: 'search') . '%')
                ->limit('value: 10)->get();

            return Response($data);
        }

        return Response(Postcode::all()->take('limit: 10));
    }
}
```

Abbildung 54 SearchController für Ajax Request

In der Ajax Funktion kann anschliessend im «success» die zurückgelieferten Daten iteriert werden und dem «Search-Result div» angehängt werden:

```
success: function (data) {

    for(let postcode of data) {
        insert(postcode);
    }

}
```

Abbildung 55 Ajax Request Erfolg

```
function insert(postcode) {

    let el = document.createElement('div');
    el.setAttribute('class', 'list-group-item list-group-item-action oldSearch');
    el.setAttribute('onclick', 'add(this, '+ postcode.id +')');
    el.innerHTML = postcode.postcode + ' | ' + postcode.name;

    searchRes.appendChild(el);

}
```

Abbildung 56 Suchresultate in der View anzeigen

### 18.9.3.2 Old Search Delete

Damit bei einer erneuten Suche die alten Suchresultate entfernt werden, wird jedem Listenelement die CSS-Klasse oldSearch hinzugefügt. Anschliessend kann bei einer erneuten Request im «success» zuerst die alten Daten entfernt werden:

```
success: function (data) {
    $('.oldSearch').remove();

    for(let postcode of data) {
        insert(postcode);
    }

}
```

Abbildung 57 Alte Suchergebnisse löschen

### 18.9.3.3 Add Result to Formular

Den Suchresultaten wurde, ein «onclick Event» hinzugefügt. Damit ist es möglich, wenn ein Suchresultat angeklickt wurde die Funktion «add(this, postcode.id) aufzurufen.

Dadurch, dass die Frontend Liste mit den aktuellen PLZ und die Liste mit den Suchresultaten identisch von den HTML und CCS Attributen ist, wird das entsprechende Searchresult-Element einfach der Liste der aktuellen PLZ angehängt und das onclick Event sowie die CSS klasse oldSearch entfernt und ein grüner Rahmen gesetzt.

```
function add(el, id) {
    el.removeAttribute('onclick');
    el.classList.remove('oldSearch');
    el.style.border = '1px solid green';

    document.getElementById('currentJur').append(el);

    let input = document.createElement('input');
    input.setAttribute('type', 'hidden');
    input.setAttribute('name', 'addJur[]');
    input.setAttribute('value', id);

    document.getElementById('addJur').append(input);
}
```

Abbildung 58 Suchergebniss dem Formular hinzufügen

Zusätzlich wird wie beim Delete im Formular ein neues Input-Element erstellt mit der Postleitzahl id als value und dem div «addJur» angehängt.

#### 18.9.3.4 Hinzufügen Rückgängig machen

Damit das Hinzufügen auch wieder rückgängig gemacht werden kann, wird dem neuen PLZ-Listenelement ein Button hinzugefügt.

```
el.removeAttribute('onclick');
el.classList.remove('oldSearch');

el.style.border = '1px solid green';
el.innerHTML = el.innerHTML + '<button type="button" class="btn btn-danger float-end" onclick="reverseAdd('+id+', this.parentElement)">X</button>';
```

Abbildung 59 Suchergebnis Delete Button Hinzufügen

Bei Tätigung dieses Buttons wird das entsprechende Inputelement aus dem Formular entfernt sowie die hinzugefügte PLZ aus der Liste

```
function reverseAdd(id, el) {
    document.getElementsByName('addJur[]').forEach(element => {
        if(element.value == id) {
            element.remove();
        }
    });
    el.remove();
}
```

Abbildung 60 Suchergebnis im Formular löschen

#### 18.9.4 Resultat

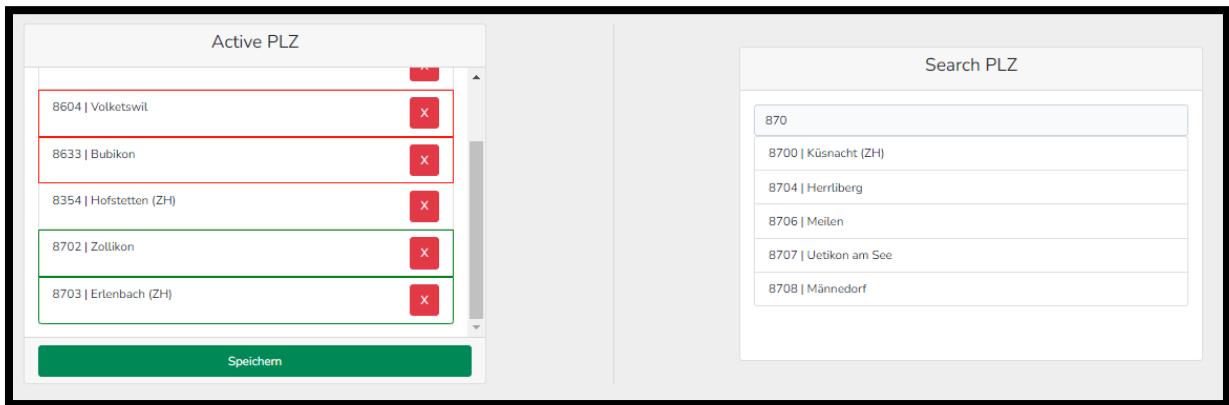


Abbildung 61 LiveSearch Ergebnis Frontend

### 18.9.5 MultiSearch

Damit es auch möglich ist, nicht nur via PLZ, sondern auch durch den Namen einer Ortschaft eine Suche zu betätigen, wird im Backend der Search String validiert. Ist dieser numeric, wird die Suche in der Spalte «postcode» durchgeführt, ansonsten über die Spalte «name».

```
class SearchController extends Controller
{
    public function searchPLZ(Request $request)
    {
        if($request->input('key:search')) {

            $column = is_numeric($request->input('key:search')) ? 'postcode' : 'name';

            $data = DB::table('postcodes')
                ->where($column, 'operator', 'LIKE', 'value', '%' . $request->input('key:search') . "%")
                ->whereNotIn('id', auth()->user()->beekeeper->postcodes->pluck('id'))
                ->limit('value', 10)->get();

            return Response($data);
        }

        return Response(Postcode::all()->take('limit', 10));
    }
}
```

Abbildung 62 MultiSearch

Damit dem Imker bei der Suche nicht bereits selektierte PLZs zur Auswahl gegeben werden, werden alle aktuell bereits Zugewiesenen aussortiert mittels «whereNotIn()». Das Suchresultat wird ebenfalls limitiert, damit nur maximal 10 Suchresultate in der View zur Auswahl stehen. Der Return Wert bleibt unverändert, diesbezüglich müssen keine Änderungen im Frontend getätigter werden.

### 18.9.6 PLZ definitiv Zuteilen

Durch die Übermittlung des Formulars mit den beiden Arrays «addJur[]» und «delJur[]» werden alle id's der Postleitzahlen übermittelt.

Im Controller BeekeeperController wurde eine neue Methode erstellt, um die Post request zu empfangen sowie die entsprechende Route im «web.php»:

```
Route::group(['middleware' => 'auth'], function() {

    Route::group(['middleware' => 'beekeeperOnly'], function() {

        Route::get('profile', [App\Http\Controllers\BeekeeperController::class, 'profile'])->name('profile');
        Route::get('jurisdiction', [App\Http\Controllers\BeekeeperController::class, 'jurisdiction'])->name('jurisdiction');
        Route::get('search/plz', [App\Http\Controllers\SearchController::class, 'searchPLZ'])->name('search.plz');

        Route::post('jurisdiction/update', [App\Http\Controllers\BeekeeperController::class, 'updateJurisdiction'])->name('jurisdiction.update');
    });
});
```

Abbildung 63 Zuteilung der PLZ Routes

Im ersten Schritt müssen die übermittelten ids validiert werden:

```
public function updateJurisdiction(Request $request)
{
    $validator = Validator::make($request->all(), [
        'delJur' => ['array'],
        'addJur' => ['array'],
        'delJur.*' => ['integer', 'exists:App\Models\Postcode,id'], // Validate each content of Array
        'addJur.*' => ['integer', 'exists:App\Models\Postcode,id'], // Validate each content of Array
    ]);
}
```

Abbildung 64 PLZ Zuteilung Validierung

Dabei wird zuerst überprüft, ob es sich bei den beiden Variablen um Arrays handelt. Danach wird jedes einzelne enthaltene Element validiert, ob es sich tatsächlich um einen integer handelt (id) und ob es diesen auch entsprechend in der Datenbank gibt.

War die Validierung erfolgreich, können die ausgewählten Postcodes entsprechend attached oder detached werden:

```
if(! $validator->fails()) {
    $postcodes = auth()->user()->beekeeper->postcodes();
    if(isset($validator->validated()['addJur'])) $postcodes->attach($validator->validated()['addJur']);
    if(isset($validator->validated()['delJur'])) $postcodes->detach($validator->validated()['delJur']);
}
return redirect(route('jurisdiction'));
```

Abbildung 65 PLZ Zuteilung

Anschliessend wird der Imker auf die bereits aufgerufene Seite Jurisdictions zurückgeleitet mit den aktuellen zugewiesenen Postcodes.

## 18.10 S&R AUFTRAGSERSTELLUNG

Für die Auftragserstellung wurde eine neue View unter «resources/views/models/contract/create.blade.php» erstellt sowie ein neuer Controller: «ContractController».

### 18.10.1 Components

Die Eingabefelder für das Formular der Auftragserstellung benötigen zusätzliche Components sowie die bereits bestehenden Components müssen angepasst werden.

#### 18.10.1.1 Alert Component

Um Alerts einfach zu kreieren, wurde ein Alert-Component erstellt. Dies können Fehler- sowie Erfolgsmeldungen sein, entsprechend den übergebenen Werten:

```
<x-forms.alert title="Error" message="{{ $message }}" type="danger" />
```

Abbildung 66 Alert Component im Formular

```
<div class="alert alert-{{ $type }} alert-dismissible fade show" role="alert">
    <strong>{{ $title }}</strong> {{ $message }}
    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
```

Abbildung 67 Alert Component Blade

#### 18.10.1.2 Textarea

Ein neues Component für eine Textarea wurde ebenfalls erstellt.

#### 18.10.1.3 Number Component

Ein neues Component für die Eingabe der Latitude/Longitude wurde ebenfalls erstellt. Zusätzlich zum Component Input wurde ein Min- sowie Max-Wert definiert sowie ein step. Mit dem step kann definiert werden, wie viele Nachkommastellen die Zahl der Eingabe enthalten darf. Für die Lon/Lat Werte sind 5 Nachkommastellen zulässig:

```
<x-forms.number name="lat" title="Latitude" min="47" max="48" placeholder="47.39036" value="{{ old('lat') }}" step=".00001" />
<x-forms.number name="lon" title="Longitude" min="8" max="9" placeholder="8.49087" value="{{ old('lon') }}" step=".00001" />
```

Abbildung 68 Number Component im Formular

```
<input class="form-control @if($errors->any()) @error($name) is-invalid @else is-valid @enderror @endif"
       type="number" name="{{ $name }}" id="{{ $name }}" placeholder="{{ $placeholder }}" value="{{ $value }}"
       min="{{ $min }}" max="{{ $max }}" step="{{ $step }}>
```

Abbildung 69 Number Component Blade

#### 18.10.1.4 Default Required

Alle Eingabefelder galten bisher als «requiert». Mit der Erweiterung von optionalen Eingaben bei der Auftragserstellung durch die Kontaktangaben müssen die Components angepasst werden.

PHP bietet keine Möglichkeit, Methoden zu Overloaden. Jedoch ist es möglich, einer variabel einen Default Wert zu setzen, sollte diese nicht übergeben worden sein im Konstruktor:

```
public function __construct($type, $name, $title, $value, $required = 'true')
{
    $this->type = $type;
    $this->name = $name;
    $this->title = $title;
    $this->value = $value;
    $this->required = $required == 'true';
}
```

Abbildung 70 Component Default Required Constructor

Alle zuvor bereits verwendeten Components erhalten required als true, da keine Angabe gemacht wurde. Optionale Felder können nun folgend erstellt werden:

```
<x-forms.input type="text" name="" title="" value="" required="false" />
```

Abbildung 71 Component Default Formular

In den entsprechenden Blade Components wurde folgende Erweiterung hinzugefügt:

```
<label class="form-label mt-4" for="{{ $name }}">{{ $title }} @if($required) <span class="text-danger">*</span> @endif</label>
```

Abbildung 72 Component Default Blade

Sowie für das Input-Element entsprechende optionale «required» für die Frontend-Validierung. Diese Änderungen wurden in allen Components durchgeführt.

#### 18.10.1.5 Optionaler Platzhalter

Damit bei der Eingabe der Telefonnummer das erwünschte Format ersichtlich ist anhand einer Beispielnummer, wurde ein optionaler Platzhalter im Component Input erstellt. Wird kein Platzhalter übergeben, wird der Titel verwendet:

```
public function __construct($type, $name, $title, $value, $required = 'true', $placeholder = '')
{
    $this->type = $type;
    $this->name = $name;
    $this->title = $title;
    $this->value = $value;
    $this->required = $required == 'true';
    $this->placeholder = $placeholder == '' ? $title : $placeholder;
}
```

Abbildung 73 Component optionaler Platzhalter

### 18.10.2 Formular

Das Formular wurde durch die Verwendung der erstellten Components zusammengesetzt mit allen notwendigen Eingabefeldern. Das Eingabefeld für die manuelle Eingabe der PLZ erscheint erst, wenn ein Fehler in der Validierung der API stattgefunden hat. Dabei wird auch zusätzlich eine entsprechende Fehlermeldung angezeigt.

```
<form method="POST" action="{{ route('contract.store') }}>
    @csrf
    @method('POST')
    @error('plz-api')
        <x-forms.alert title="Error" message="{{ $message }}" type="danger" />
    @enderror
    <x-forms.number name="lat" title="Latitude" min="47" max="48" placeholder="47.39036" value="{{ old('lat') }}" step=".00001" />
    <x-forms.number name="lon" title="Longitude" min="-8" max="9" placeholder="8.49087" value="{{ old('lon') }}" step=".00001" />
    @error('plz-api')
        <x-forms.number name="plz" title="Postcode" min="8000" max="9000" placeholder="8048" value="{{ old('plz') }}" step="1" required="false" />
    @enderror
    <hr />
    <h4 class="text-center">Contact Information</h4>
    <x-forms.input type="text" name="contact_firstname" title="Firstname" value="{{ old('contact_firstname') }}" required="false" />
    <x-forms.input type="text" name="contact_lastname" title="Lastname" value="{{ old('contact_lastname') }}" required="false" />
    <x-forms.input type="text" name="contact_phone" title="Phone" value="{{ old('contact_phone') }}" required="false" placeholder="076 581 35 96" />
    <x-forms.textarea name="info" title="Additional Information" value="{{ old('info') }}" required="false" />
    <input class="btn btn-block btn-primary mt-3" type="submit" value="Create Contract" />
</form>
```

Abbildung 74 Auftrag erstellen Formular mit Components

### 18.10.3 Validierung

#### 18.10.3.1 Datenvorbereitung

Damit die Validierung durchgeführt werden kann, muss die Telefonnummer mittels der helper Funktion in das richtige Format formatiert werden. Im gleichen Schritt wird auch der «created\_by» Eintrag hinzugefügt:

```
public function store(Request $request)
{
    // Prepare Request for Validation: Format phone and add created_by
    $request->merge([
        'contact_phone' => replaceEmptyChars($request->all()['contact_phone']),
        'created_by'     => auth()->user()->id,
    ]);
}
```

Abbildung 75 Auftrag erstellen Datenvorbereitung für Validierung

### 18.10.3.2 Validierung

Bei der Validierung werden alle Eingaben folglich überprüft:

```
$validator = Validator::make($request->all(), [
    'lat'          => ['required', 'numeric', 'min:47', 'max:48'],
    'lon'          => ['required', 'numeric', 'min:8', 'max:9'],
    'plz'          => ['nullable', 'numeric', 'min:8000', 'max:9000', 'exists:App\Models\Postcode,postcode'],
    'contact_firstname' => ['nullable', 'string'],
    'contact_lastname' => ['nullable', 'string'],
    'contact_phone'   => ['nullable', 'regex:/^(?:0041|0|\+41)(\d{9})$/'],
    'info'          => ['nullable', 'string'],
    'created_by'    => ['required', 'exists:App\Models\User,id']
]);
```

Abbildung 76 Auftrag erstellen Validierung

Die Eingabe der PLZ kann erst erfolgen, wenn die API fehlgeschlagen ist und das Formular erneut ausgefüllt wird. Diesbezüglich kann die PLZ null sein. Sollte die PLZ nicht null sein, wird diese überprüft, ob es eine Nummer innerhalb des Min und Max-Werts ist, sowie ob sie in der Datenbank enthalten ist.

Fehlschlägt die Validierung wird überprüft, ob die PLZ mitausgefüllt wurde. Voraussetzung ist dafür, dass beim ersten Versuch die API einen Fehler vorgewiesen hat. Wenn dies der Fall sein sollte, wird die Fehlermeldung «plz-api» manuell hinzugefügt und erscheint wieder im Formular.

```
if($validator->fails()) {
    // If Key exists:
    // 1st Validation Nominatim Failed to find PLZ & input PLZ was shown
    // 2nd Validation either Nominatim Failed again or PLZ failed and input PLZ needs to be shown again
    if(array_key_exists('plz', $request->all())) {
        $validator->errors()->add('plz-api', 'Something went wrong with Nominatim');
    }
    return back()->withErrors($validator)->withInput();
}
```

Abbildung 77 Auftrag erstellen Validierung failed

Wurden alle Angaben korrekt ausgefüllt, wird unterschieden, ob die PLZ manuell übergeben wurde oder ob die Applikation die PLZ selbst ermitteln muss:

```
$validated = $validator->validated();

if(array_key_exists('plz', $validated) && !is_null($validated['plz'])) {

    $validated['postcode_id'] = Postcode::where('postcode', $validated['plz'])->first()->id;

} else {

    // API CALL
    ...
}
```

Abbildung 78 Auftrag erstellen Validierung, ob PLZ manuell erfolgte

Wurde die PLZ korrekt übergeben, wird die Postleitzahl aus der Datenbank gesucht und die entsprechende id im validierten Array gespeichert.

#### 18.10.4 Nominatim PLZ-Ermittlung

Ein neuer Controller «GeoLocationController» wurde erstellt. Darin wird ein externer API-Call getätigkt, um von den übermittelten Latitude- und Longitude-Wert die entsprechende PLZ aufzulösen.

```
class GeoLocationController extends Controller
{
    public static function getPlzFromLatLon($lat, $lon)
    {
        $res = Http::get('https://nominatim.openstreetmap.org/reverse?lat=' . $lat . '&lon=' . $lon . '&format=json');

        $place = json_decode($res->body());

        if(array_key_exists('address', $place) && array_key_exists('postcode', $place->address)) {
            $postcode_nom = $place->address->postcode;

            $postcode = Postcode::where('postcode', $postcode_nom)->first();

            if($postcode) return $postcode;
        }
    }

    return null;
}
```

Abbildung 79 GeoLocationController Nominatim PLZ ermittlung

Mittels einer Get Request kann die URL mit den Latitude und Longitude werten ergänzt und versucht werden die PLZ ausfindig zu machen. Als Return wird ein JSON Objekt erwartet und als JSON decodiert. Dieses wird überprüft, ob der Key «address» enthalten ist sowie der Key «address.postcode». Sind beide Keys enthalten, kann die zurückgegebene PLZ aus der Datenbank gesucht und zurückgegeben werden. Bei einem Fehlschlag der API wird null zurückgegeben.

### 18.10.5 Nominatim Validierung

Wurde die PLZ via Nominatim ermittelt wird validiert, ob dies erfolgreich war.

```
// Try to get PLZ from Nominatim
$postcode = GeoLocationController::getPlzFromLatLon($validated['lat'], $validated['lon']);

// If Nominatim Failed to provide a PLZ
if (is_null($postcode)) {
    // Add validation failed keys + messages.
    $validator->errors()->add('plz-api', 'Please Enter Postcode manually or change Latitude/Longitude.');
    $validator->errors()->add('lat', ' ');
    $validator->errors()->add('lon', ' ');
    $validator->errors()->add('plz', 'Please Enter Postcode manually or change Latitude/Longitude.');

    return back()->withErrors($validator)->withInput();
}

$validated['postcode_id'] = $postcode->id;
```

Abbildung 80 Nominatim Validierung

Wurde die PLZ gefunden, wird die Postleitzahl id im validierten Array gespeichert. Sollte die API fehlschlagen, werden manuell Fehlermeldungen dem Validator hinzugefügt und der User auf das Contract-Create Formular mit Auskunft der Fehlermeldungen zurückgeleitet.

### 18.10.6 Create Contract

Dadurch, dass im validierten Array alle Keys der Attribute des Contract Models entsprechen, kann ein neuer Auftrag direkt erstellt werden aus dem Array mit den validierten Eingaben:

```
$contract = Contract::create($validated);
```

Abbildung 81 Auftrag erstellen

Nach dem erfolgreichen Erstellen eines Contracts werden die Imker der zuständigen Region kontaktiert und dem Mitarbeiter von S&R eine Übersicht des erstellten Contracts angezeigt.

## 18.11 AUFRAGS-SMS NOTIFIKATION

### 18.11.1 Voraussetzungen

#### 18.11.1.1 Composer Packages

Die benötigten Packages für das Versenden von SMS mittels Vonage können mit folgendem Befehl hinzugefügt werden:

```
composer require laravel/nexmo-notification-channel nexmo/laravel
```

#### 18.11.1.2 Config

Um den Nexmo (Vonage) Key, Secret und Von Adresse im «.env» File auszulagern wurde im «config/services.php» folgender Eintrag hinzugefügt:

```
'nexmo' => [
    'key' => env('NEXMO_KEY'),
    'secret' => env('NEXMO_SECRET'),
    'sms_from' => env('NEXMO_FROM'),
],
```

Abbildung 82 Config anpassung für Nexmo

Im «.env» File können nun die Angaben hinterlegt werden:

```
NEXMO_KEY=*****
NEXMO_SECRET=*****
NEXMO_FROM=BeeRes
```

Abbildung 83 Nexmo Key, Secret und From in .env Datei

#### 18.11.1.3 Beekeeper Notifiable

Um eine Notifikation zu erstellen, muss das Model Beekeeper mit dem Trait «Notifiable» erweitert werden sowie die Funktion «routeNotificationForNexmo()» hinzugefügt werden. Diese ist notwendig, um die Telefonnummer des Models Beekeeper der Notifikation zu übermitteln. Dadurch weiss Vonage an welche Nummer das SMS geschickt werden soll.

```
class Beekeeper extends Model
{
    use HasFactory, Notifiable;
```

Abbildung 84 Beekeeper Model Notifiable trait

```
public function routeNotificationForNexmo($notification)
{
    return $this->phone;
}
```

Abbildung 85 Beekeeper Ziel Telefonnummer für Nexmo Notifikation

### 18.11.2 Notifikation

Die neu erstellte Notifikation «ContractApplicableNotification» muss angepasst werden. Per Default sind Notifikationen für E-Mails ausgelegt. Mit dem Implementieren des «ShouldQueue» Interfaces, werden neue Notifikationen einer Queue hinzugefügt. Diese wird dann von einem Queue-Worker abarbeitet.

```
class ContractApplicableNotification extends Notification implements ShouldQueue
{
    use Queueable;

    private $contract;

    /**
     * Create a new notification instance.
     *
     * @param Contract $contract
     */
    public function __construct(Contract $contract)
    {
        $this->contract = $contract;
    }
}
```

Abbildung 86 Contract Applicable Notification

Die via Funktion wird angepasst, damit Vonage (Nexmo) verwendet wird:

```
public function via($notifiable)
{
    return ['nexmo'];
}
```

Abbildung 87 Notifikation via

In der toNexmo Funktion wird die zu sendende Nachricht erfasst:

```
public function toNexmo($notifiable)
{
    return (new NexmoMessage)
        ->content(content: "You have been selected for a new Beekeeper job!"
            ."\\nLocation: ".$this->contract->postcode->codeName
            ."\\n\\nApply here:\\n".
        route(name: 'contract.accept', $this->contract->id));
}
```

Abbildung 88 Notifikation Nachricht

### 18.11.2.1 Imker ausfindig machen

Die direkt zuständigen Imker können vom Postcode des Contracts direkt ausfindig gemacht werden. Sollte jedoch kein Imker direkt zuständig sein, werden die zwei am nächsten befindlichen Imker von der Imker Search gesucht. Anschliessend werden allen gefundenen Imkern der Auftrag als applicable hinzugefügt sowie die SMS-Notifikation erstellt und versendet.

```
class NotificationController extends Controller
{

    /**
     * Find the closest Beekeepers and notify about the newly created contract.
     *
     * @param Contract $contract
     * @return void
     */
    public static function notifyBeekeeperNewContract(Contract $contract)
    {

        // Find all beekeepers from specific region
        $beekeepers = $contract->postcode->beekeepers;

        // If no beekeepers could be found, find the closest Beekeepers
        if(count($beekeepers) == 0) {
            $beekeepers = SearchController::findClosestBeekeeper($contract->postcode->postcode)->take( limit: 2 );
        }

        foreach ($beekeepers as $beekeeper) {

            // Add beekeeper to be applicable to contract
            $beekeeper->contracts_applicable()->attach($contract);
            // Notify beekeeper about the new applicable contract
            $beekeeper->notify(new ContractApplicableNotification($contract));
        }
    }
}
```

Abbildung 89 Zuständige Imker ausfindig machen und kontaktieren per Notifikation

## 18.12 IMKER ZUSAGE

Nach dem Erhalten einer SMS-Notifikation über einen neuen möglichen Auftrag kann der Imker die in der Notifikation enthaltene URL zur Annahme des Auftrags öffnen:

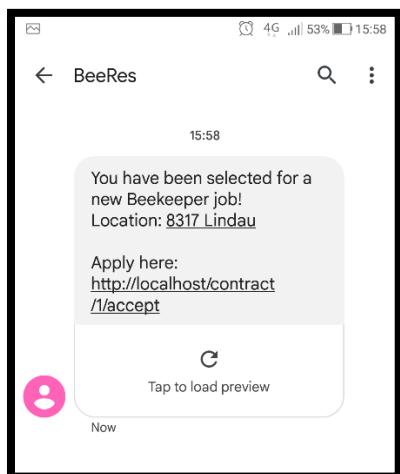


Abbildung 90 Imker Notifikation SMS

### 18.12.1 Auftrag anzeigen

Bevor der mögliche Auftrag angezeigt wird, wird validiert, ob dieser bereits vergeben ist und ob der angemeldete Imkere für diesen Auftrag applicable ist.

```
public function accept($id)
{
    $contract = Contract::findOrFail($id);

    if($contract->beekeeper) {
        if($contract->beekeeper->id == auth()->user()->beekeeper->id) {
            return redirect(route('contract.accept.success', $contract));
        }
        return redirect(route('contract.taken'));
    }

    if(! $contract->beekeepers->pluck('id')->contains(auth()->user()->beekeeper->id)) return redirect(route('index'));

    return view('models.contract.accept', ['contract' => $contract]);
}
```

Abbildung 91 Auftrag applicable Validierung

Beim Fehlschlag der Validierung wird der Imker auf die Seite «Auftrag bereits vergeben» weitergeleitet. Ist der Auftrag noch nicht vergeben und der Imker darf ihn annehmen, wird ihm das Formular zur Annahme angezeigt. Dies beinhaltet die Region sowie die von ihm zu bestätigende «Terms and Conditions».

### 18.12.2 Auftrags-Annahme

Bevor der Auftrag definitiv angenommen wird, wird validiert, ob zwischenzeitlich der Auftrag bereits vergeben wurde und ob der Imker für diesen Auftrag applicable ist. Schlussendlich wird noch überprüft, ob er die «Terms and Conditions» akzeptiert hat.

```
public function acceptContract(Request $request, $id)
{
    $contract = Contract::findOrFail($id);

    // Validate if contract already has a beekeeper
    if($contract->beekeeper) return redirect(route('contract.taken'));

    // Validate if beekeeper is applicable to accept this contract
    if(! $contract->beekeepers->pluck('id')->contains(auth()->user()->beekeeper->id)) return redirect(route('index'));

    $request->validate([
        'acc' => ['required']
    ], [
        'required' => 'The terms and conditions are required!'
    ]);

    // Add beekeeper to contract
    $contract->beekeeper()->associate(auth()->user()->beekeeper)->save();

    // Notify detail Info per SMS.
    NotificationController::notifyBeekeeperContractAssigned($contract);

    return redirect(route('contract.accept.success', $contract));
}
```

Abbildung 92 Auftrags-Annahme

Nach einer erfolgreichen Validierung wird der Auftrag dem Imker zugewiesen und die Notifikation der Details erstellt und versendet vom NotificationsController. Der Imker erhält eine Erfolgsmeldung und zusätzlich werden im Browser die Details des Auftrages ausgewiesen.

## 18.13 AUFTRAGS-SMS BESTÄTIGUNG

Eine neue Notifikation wurde erstellt: «ContractAssignedNotification» welche identisch ist zur «ContractApplicableNotification» ausser dem Inhalt der Nachricht. Die optionalen Felder werden validiert, ob ein Eintrag vorhanden ist. Sollte dies der Fall sein, wird der entsprechende Eintrag der Nachricht hinzugefügt.

```
private function generateTextMessage(): string
{
    $msg = "You have been assigned for a new Beekeeper job! \n\n";

    if ($this->contract->hasOptionalInfo) {
        $msg .= "Additional Information: \n";
        if ($this->contract->contact_firstname) {
            $msg .= 'Firstname: ' . $this->contract->contact_firstname . "\n";
        }
        if ($this->contract->contact_lastname) {
            $msg .= 'Lastname: ' . $this->contract->contact_lastname . "\n";
        }
        if ($this->contract->contact_phone) {
            $msg .= 'Phone: ' . $this->contract->contact_phone . "\n";
        }
        if ($this->contract->info) {
            $msg .= 'Info: ' . $this->contract->info . "\n";
        }
    }

    $msg .= "\nIf possible, please get in touch with the contact person in advance to discuss details.\n\n";
    $msg .= GeoLocationController::generateGoogleMapsPin($this->contract->lat, $this->contract->lon);

    return $msg;
}
```

Abbildung 93 Auftrag Details Notifikationsnachricht

Im GeoLocationController befindet sich die Funktion, um die URL des GoogleMaps-Pins zu erstellen:

```
public static function generateGoogleMapsPin($lat, $lon): string
{
    return 'https://www.google.com/maps/search/?api=1&query=' . $lat . '%2C' . $lon;
}
```

Abbildung 94 GoogleMaps-Pin generieren

Der NotificationController wurde mit folgender Funktion erweitert, um die Details an den Imker zu senden:

```
public static function notifyBeekeeperContractAssigned(Contract $contract)
{
    $contract->beekeeper->notify(new ContractAssignedNotification($contract));
}
```

Abbildung 95 Imker Auftragsdetails zusenden

## 18.14 IMKER-SEARCH

### 18.14.1 Frontend

Das Frontend besteht im Wesentlichen aus einem Input-Element für die Sucheingabe und zwei Divs um die Suchresultate hinzuzufügen.

```
<div class="card col-lg-8 offset-lg-2 mb-2 mb-md-0">
    <div class="card-header text-center"><h4>Imker Search</h4></div>
    <div class="card-body" style="...">
        <div class="input-group mb-1">
            <input type="text" class="form-control" id="searchInput" onkeyup="searchPlz(this)" placeholder="PLZ Search"
                   autocomplete="off" aria-describedby="basic-addon">
            <div class="input-group-append">
                <span class="input-group-text h-100" id="basic-addon">
                    <i class="fa-solid fa-3x fa-magnifying-glass"></i>
                </span>
            </div>
        </div>
        <div class="list-group overflow-auto border" style="..." id="plzSearchOutput"></div>
        <div class="list-group overflow-auto border mt-1" style="..." id="beekeeperSearchOutput"></div>
    </div>
</div>
<script>
    const searchPlzUrl      = '{{ route('guest.search.plz') }}';
    const searchBeekeeperUrl = '{{ route('guest.search.beekeeper') }}';
</script>
<script src="{{ asset('js/imkerSearch.js') }}"></script>
```

Abbildung 96 ImkerSearch: Blade

Die PLZ Suche ist identisch zu der, welche Imker verwenden für die Jurisdictions mit Ausnahme, dass die Postleitzahlen nicht gefiltert werden nach den bereits vorhanden sowie das Suchergebnis sich auf fünf PLZ beschränkt:

```
$data = DB::table( table: 'postcodes' )
    ->where($column, operator: 'LIKE', value: '%' . $request->input( key: 'search' ) . "%")
    ->limit( value: 5)->get();
```

Abbildung 97 ImkerSearch: PLZ

Die Suchergebnisse werden unter dem div id «plzSearchOutput» angehängt. Mit dem onclick Event auf einer PLZ wird dessen Name sowie Postleitzahl in das Suchfeld einfügt, die Suchresultate der Postleitzahlen gelöscht und die eigentliche ImkerSuche aufgerufen:

```
function selectPostcode(e, postcode_id) {
    document.getElementById( elementId: 'searchInput').value = e.innerHTML;
    $('.oldSearch').remove();

    searchBeekeeper(postcode_id);

}
```

Abbildung 98 ImkerSearch aufrufen mit onclick Event auf PLZ

Bei der Ajax Request wird die id der Postleitzahl dem SearchController übergeben. Die zurückgelieferten Imker werden mit der Funktion insertBeekeeper dem div id «beekeeperSearchOutput» angehängt.

```

function searchBeekeeper(postcode_id) {
    $.ajax({
        type : "get",
        url : searchBeekeeperUrl,
        data : {postcode_id: postcode_id},
        success: function (data) {
            $('.oldSearch').remove();

            for(let beekeeper of data) {
                insertBeekeeper(beekeeper);
            }
        }
    });
}

function insertBeekeeper(beekeeper) {
    let jurEl = document.createElement('p');
    jurEl.setAttribute('qualifiedName', 'class', value: 'small');
    jurEl.innerHTML = beekeeper.jurisdictionsToString;

    let el = document.createElement('div');
    el.setAttribute('qualifiedName', 'class', value: 'list-group-item list-group-item-action oldSearch');
    el.innerHTML = beekeeper.fullName + '<br>' + beekeeper.reverseFormattedPhone;
    el.appendChild(jurEl);

    beekeeperSearch.appendChild(el);
}

```

Abbildung 99 Imker suchen und im Frontend anzeigen

## 18.14.2 Backend

### 18.14.2.1 Validierung

Bevor eine Suche in der Datenbank getätigigt wird, wird überprüft, ob die übermittelte id einer PLZ existiert:

```

public static function guestSearchBeekeeper(Request $request)
{
    $validator = Validator::make($request->all(), [
        'postcode_id' => ['required', 'exists:App\Models\Postcode,id'],
    ]);
}

```

Abbildung 100 ImkerSearch: PLZ Validierung

Ist die Validierung erfolgreich, werden die am nächsten gelegenen Imker anhand der PLZ gesucht.

```

if(! $validator->fails()) {

    $validated = $validator->validated();
    $postcode = Postcode::find($validated['postcode_id']);

    $beekeepers = self::findClosestBeekeeper($postcode->postcode);

    return Response($beekeepers->take( limit: 5)->map->only(['fullName', 'formattedPhone', 'jurisdictionsToString']));
}

```

Abbildung 101 Imker suchen und selektierte Daten zurückgeben

### 18.14.2.2 Rückgabewert

Zurückgegeben werden von den Beekeepers jeweils nur der volle Name, die formatierte Telefonnummer und die zuständigen Regionen als String. Dies sind Funktionen auf dem Beekeeper-Model und sehen wie folgt aus:

```
public function getFullNameAttribute(): string
{
    return $this->firstname . ' ' . $this->lastname;
}
```

Abbildung 102 Beekeeper FullName Model Attribute

Um die Telefonnummer aus der Datenbank vom Format mit Ländercode (4176xx) umzuwandeln zu (076xx) wurde in der «app/helpers.php» Datei folgende globale Funktion hinzugefügt:

```
function reverseFormatPhoneNum($phone): string
{
    $num[0] = substr($phone, offset: 2, length: 2);
    $num[1] = substr($phone, offset: 4, length: 3);
    $num[2] = substr($phone, offset: 7, length: 2);
    $num[3] = substr($phone, offset: 9, length: 2);

    return '0' . $num[0] . ' ' . $num[1] . ' ' . $num[2] . ' ' . $num[3];
}
```

Abbildung 103 Formatierung der Telefonnummer Rückgängig machen

Der Aufruf geschieht direkt aus dem Beekeeper-Model:

```
public function getReverseFormattedPhoneAttribute(): string
{
    return reverseFormatPhoneNum($this->phone);
}
```

Abbildung 104 Beekeeper Reverse Format Phone Model Attribute

Die zuständigen Regionen werden iteriert und jeweils die PLZ sowie der Name einem String angehängt. Am Ende wird noch das überschüssige Trennzeichen entfernt.

```
public function getJurisdictionsToStringAttribute() :string
{
    $res = '';

    foreach ($this->postcodes()->pluck('name', 'key: 'postcode') as $postcode => $name) {
        $res .= $postcode.' '.$name.', ';
    }

    return substr($res, 0, strlen($res) - 2);
}
```

Abbildung 105 Beekeeper Zuweisungen als String umformen Model Attribute

```
>>> App\Models\Beekeeper::find([1,2])->map->only(['fullName', 'reverseFormattedPhone', 'jurisdictionsToString']);
=> Illuminate\Support\Collection {#4435
    all: [
        [
            "fullName" => "Howard Hintz",
            "reverseFormattedPhone" => "076 599 94 40",
            "jurisdictionsToString" => "8006 Zürich",
        ],
        [
            "fullName" => "Lue Legros",
            "reverseFormattedPhone" => "079 699 32 66",
            "jurisdictionsToString" => "8634 Hombrechtikon, 8816 Hirzel, 8617 Mönchaltorf",
        ],
    ],
}
```

Abbildung 106 Selektierte Rückgabewerte Überprüfung

### 18.14.3 Imker proximity Search

Die nächstgelegenen Imker können ausfindig gemacht werden durch die Berechnung:

*Gesuchte PLZ – Imker zuständige PLZ*

Dazu müssen in der Datenbank die Imker und ihre zuständigen PLZ zusammen gesetzt werden mittels eines Joins. Anschliessend kann nach dem berechneten Wert der Differenz die Imker sortiert werden. Dabei ist wichtig, dass der absolute Wert jeweils verglichen wird. In diesen Beispielen wird die PLZ 8700 als gesuchte PLZ verwendet:

```

1
2 •  SELECT beekeepers.id, beekeepers.lastname, postcodes.postcode, abs(8700 - postcodes.postcode) AS mindiff FROM beekeepers
3   JOIN beekeeper_postcode ON beekeeper_postcode.beekeeper_id = beekeepers.id
4   JOIN postcodes ON beekeeper_postcode.postcode_id = postcodes.id
5   ORDER BY mindiff;

```

	id	lastname	postcode	mindiff
▶	21	Doe	8700	0
	19	Wyman	8702	2
	6	Toy	8706	6
	3	Wintheiser	8635	65
	2	Legros	8634	66
	2	Legros	8617	83
	11	Pouros	8616	84
	18	Crist	8610	90

Abbildung 107 Proximity Search Query

Mittels der Joins werden die Imker jeweils pro ihre Zuständigkeit aufgeführt. Damit jeder Imker nur einmal enthalten ist, werden die Imker nach ihrer id gruppiert und jeweils nur der kleinste Wert aus der Errechnung der Differenz verwendet:

```

1 •  SELECT beekeepers.id, beekeepers.lastname, min(abs(8700 - postcodes.postcode)) AS mindiff FROM beekeepers
2   JOIN beekeeper_postcode ON beekeeper_postcode.beekeeper_id = beekeepers.id
3   JOIN postcodes ON beekeeper_postcode.postcode_id = postcodes.id
4   GROUP BY beekeepers.id
5   ORDER BY mindiff;

```

	id	lastname	mindiff
▶	21	Doe	0
	19	Wyman	2
	6	Toy	6
	3	Wintheiser	65
	2	Legros	66
	11	Pouros	84
	18	Crist	90

Abbildung 108 Proximity Search Query Erweiterung

Das Ergebnis ist eine sortierte Liste der Imker aufgrund der Differenz ihrer zuständigen Regionen zu der gesuchten PLZ.

Das umgeformte Query für Laravel sieht wie folgt aus:

```
public static function findClosestBeekeeper($postcode)
{
    $data = DB::table('beekeepers')
        ->join('table: beekeeper_postcode', 'first: beekeeper_postcode.beekeeper_id', 'operator: =', 'second: beekeepers.id')
        ->join('table: postcodes', 'first: beekeeper_postcode.postcode_id', 'operator: =', 'second: postcodes.id')
        ->selectRaw('expression: beekeepers.id, min(abs('. $postcode . ' - postcodes.postcode)) AS mindiff')
        ->groupBy('beekeepers.id')
        ->orderBy('mindiff')
        ->pluck('column: beekeepers.id');
```

Abbildung 109 Proximity Search Query in Laravel

Dabei wird am Ende nur ein sortiertes Array der IDs der Imker benötigt (pluck).

Anschliessend werden die Imker Models anhand der IDs aus dem Array sortiert und zurückgegeben:

```
$beekeepers = Beekeeper::all();

return $data->map(function($id) use($beekeepers) {
    return $beekeepers->where('id', $id)->first();
});
```

Abbildung 110 Imker sortieren nach IDs der Proximity

## 19 KONTROLIEREN

---

### 19.1 TESTPROTOKOLL 01

Getestet wurde am 28.03.2022 vom Entwickler Christopher O'Connor.

Testfall Nr.	Resultat	Bemerkung	Unterschrift
1.0	Teilweise erfüllt	Der Imker wird registriert und angemeldet aber auf die Index Seite weitergeleitet.	COC
1.1	Erfüllt		COC
1.2	Erfüllt		COC
1.3	Erfüllt		COC
1.4	Erfüllt		COC
1.5	Erfüllt		COC
1.6	Erfüllt		COC
1.7.0	Erfüllt		COC
1.7.1	Erfüllt		COC
1.7.2	Erfüllt		COC
1.7.3	Erfüllt		COC
1.7.4	Erfüllt		COC
1.7.5	Teilweise erfüllt	Die Fehlermeldung erscheint im Passwortfeld und nicht im Confirm-Passwortfeld.	COC
2.0	Erfüllt		COC
2.1	Erfüllt		COC
2.2	Erfüllt		COC
3.0	Erfüllt		COC
3.1	Erfüllt		COC
3.2	Erfüllt		COC
3.3	Erfüllt		COC
3.4	Erfüllt		COC
3.5	Erfüllt		COC
3.6	Erfüllt		COC
3.7	Erfüllt		COC
3.8	Erfüllt		COC
3.9	Erfüllt		COC
4.0.0	Erfüllt		COC
4.0.1	Teilweise erfüllt	Fehlermeldung gibt keine Auskunft über das spezifisch zu verwendende Format.	COC
4.1	Erfüllt		COC
4.2	Erfüllt		COC
4.3.0	Erfüllt		COC
4.3.1	Erfüllt		COC
4.3.2	Erfüllt		COC
4.3.3	Erfüllt		COC
4.3.4	Erfüllt		COC
4.3.5	Erfüllt		COC
4.3.6	Erfüllt		COC
4.3.7	Erfüllt		COC
4.3.8	Erfüllt		COC
4.4.0	Erfüllt		COC
4.4.1	Erfüllt		COC
4.4.2	Erfüllt		COC

<b>4.4.3</b>	Erfüllt		COC
<b>4.4.4</b>	Erfüllt		COC
<b>4.4.5</b>	Erfüllt		COC
<b>4.4.6</b>	Erfüllt		COC
<b>4.4.7</b>	Erfüllt		COC
<b>4.4.8</b>	Erfüllt		COC
<b>4.5.0</b>	Erfüllt		COC
<b>4.5.1</b>	Erfüllt		COC
<b>4.5.2</b>	Erfüllt		COC
<b>4.5.3</b>	Erfüllt		COC
<b>4.5.4</b>	Erfüllt		COC
<b>5.0</b>	Erfüllt		COC
<b>5.1</b>	Erfüllt		COC
<b>5.2</b>	Erfüllt		COC
<b>5.3</b>	Erfüllt		COC
<b>6.1</b>	Erfüllt		COC
<b>6.2</b>	Erfüllt		COC
<b>7.0</b>	Erfüllt		COC
<b>7.1</b>	Erfüllt		COC
<b>7.2</b>	Erfüllt		COC
<b>7.3</b>	Erfüllt		COC
<b>7.4</b>	Erfüllt		COC
<b>7.5</b>	Erfüllt		COC

### 19.1.1 Massnahmen

Die teilweise erfüllten Testfälle müssen überarbeitet werden. Dadurch, dass nur kleine Änderungen vorgenommen werden müssen, welche keinerlei Einfluss auf das Gesamtsystem haben, werden die Testfälle #1.0, #1.7.5 und #4.0.1 separat nochmals überprüft.

#### 19.1.1.1 Testfall #1.0 Imker Registrierung: Erfolgreich

Die Registrierung und Anmeldung eines Imkers waren erfolgreich. Dieser wurde jedoch nicht korrekt auf sein Profil weitergeleitet.

Im «App\Controllers\Auth\RegisterController» wurde der variable «\$redirectTo» der Wert «profile» zugewiesen.

#### 19.1.1.2 Testfall #1.7.5 Imker Registrierung: Confirm Password Validation

Die Validierung, ob das Passwort und die Passwortbestätigung übereinstimmen, wird auf dem Passwort gemacht. Diesbezüglich wird auch die Fehlermeldung fälschlicherweise unter dem Passwort ausgegeben, anstatt unter dem «Password Confirmed» Eingabefeld.

Bei der Validierung im «App\Controllers\Auth\RegisterController::validator()» wurde für das Eingabefeld «password\_confirmed» die Regel: «same:password» hinzugefügt und die Regel «confirmed» vom «password» entfernt.

#### 19.1.1.3 Testfall #4.0.1 S&R Auftragserstellung Phone Validierung

Die Validierung der optionalen Telefonnummer funktioniert. Bei der Fehlermeldung wird jedoch nicht das erwartete Format spezifiziert, sondern nur «Format is invalid» ausgegeben.

In der Validierung im «App\Controllers\ContractController::store()» wurde die custom Fehlermeldung «Not a valid Format, please use: +41, 0041, 07x or 04x» hinzugefügt für «contact\_phone.regex».

## 19.2 TESTPROTOKOLL 02

Getestet wurde am 28.03.2022 vom Entwickler Christopher O'Connor. Inhalt des Testens sind die überarbeiteten, teilweise erfüllten Testfälle aus Testprotokoll 01.

Testfall Nr.	Resultat	Bemerkung	Unterschrift
1.0	Erfüllt		COC
1.7.5	Erfüllt		COC
4.0.1	Erfüllt		COC

### 19.2.1 Massnahmen

Das erwartete und tatsächliche Resultat der überarbeiteten Testfälle stimmte überein. Die Tests liefen fehlerfrei. Diesbezüglich müssen keine weiteren Massnahmen getroffen werden.

## 20 AUSWERTEN

---

Die Funktionalität der Applikation kann anhand des Testfallprotokolls 01 ausgelesen werden. Die überarbeiteten, teilweise erfüllten Testfälle aus Testprotokoll 02, welche nicht funktional fehlgeschlagen sind, sondern kleine Verbesserungen benötigten, wurden nachverbessert und bei der nochmaligen Überprüfung im Testprotokoll 02 erfolgreich abgenommen.

Aufgrund der erfolgreichen Testfallüberprüfung kann ausgesagt werden, dass die Applikation alle Erwartungen erfüllt und eingesetzt werden kann.

## 21 SCHLUSSWORT

---

### 21.1 ALLGEMEIN

Zu Beginn wollte ich wesentlich mehr implementieren, als von der Aufgabenstellung gefordert worden ist. Mir wurde schnell bewusst, dass ich mich jedoch nur auf das Wesentliche konzentrieren kann.

### 21.2 INFORMIEREN

Beim Informieren war ich ungenau. Beispielsweise habe ich die Schnittstelle Mailtrap beschrieben, obwohl diese gar keine Verwendung in der Applikation gefunden hat. Zusätzlich habe ich mich mit der Dokumentation von Vonage, welche beschreibt, wie diese in einer PHP-Applikation implementiert wird, zu genüge getan. Ich bin schlicht davon ausgegangen, dass dies nicht in der Laravel Dokumentation enthalten ist. Dies war jedoch nicht der Fall. Es ist sogar sehr detailliert beschrieben, wie Vonage mittels Notifikationen verwendet werden kann. Diesbezüglich musste ich mich während dem Realisieren damit auseinandersetzen.

### 21.3 PLANUNG

Der Zeitplan hätte bei grösseren Teilaufgaben weiter verfeinert werden sollen, um den Arbeitsaufwand besser abschätzen zu können. Diesbezüglich habe ich bei einigen Aufgaben den tatsächlichen Aufwand unterschätzt. Zudem wurde in der Realisierung nur die Realisierungszeit eingeplant und das Dokumentieren des Sourcecodes hätte eigenständig geplant werden sollen und nicht einfach nur unter Dokumentieren fallen. Die von mir erstellten Mockups sowie das Sequenzdiagramm waren für die Realisierung unentbehrlich.

### 21.4 REALISIERUNG

Während dem Realisieren kam es immer wieder mal zu unerwarteten Ereignissen oder zu Problemen. Diese konnten allerdings immer innert Kürze ausfindig gemacht und gut gelöst werden. Während dem Realisieren habe ich bereits sehr ausgiebig getestet und konnte Fehler sehr früh beheben sowie auch unerwartete Fehler aufspüren.

## 21.5 TESTEN

Das Testen habe ich aus zeitlichen Gründen, um die Dokumentation zu vervollständigen, auf den letzten Tag versetzt. Diese Entscheidung war meiner Meinung nach die Richtige und hat sich voll ausgezahlt. Das Testen verlief wie erwartet und keine funktionalen Fehler konnten gefunden werden. Diesbezüglich war ich sehr erleichtert und dies konnte mir genügend notwendige Zeit verschaffen, um die Dokumentation ohne zu grossen Zeitdruck zu fertigen.

## 21.6 PERSÖNLICHE BILANZ

In Bezug auf die IPA, welche sehr klar vorgegebene Kriterien aufweist, hätte ich nach jedem Projektschritt die jeweiligen Kriterien begutachtet, evaluiert, überarbeitet und vollständig abgeschlossen haben, anstatt mich mit teilweise erfüllten Kriterien zu Genüge tun und erst im Schritt Kontrollieren mich intensiv damit zu befassen. Andererseits konnte ich so sicherstellen, dass meine IPA alles was abverlangt wird, teilweise enthält und gegen Ende nach Priorisierung vervollständigen konnte.

Für zukünftige Arbeiten nehme ich mir vor, die genau erwarteten Kriterien beim Beginn der Arbeit genauer zu analysieren und die Planung detaillierter zu gestalten mit Unterteilung von grösseren Teilaufgaben.

Insgesamt bin ich sehr zufrieden mit meiner Leistung und mit dem entstandenen Resultat.

## 22 GLOSSAR

---

BEGRIFF	DEFINITION / ERKLÄRUNG
<b>Applicable</b>	Übertragbar / Ein Imker ist für einen Auftrag applicable wenn dieser in seiner Region ist.
<b>Attach</b>	Anfügen
<b>Auftragsdivision</b>	Auftrag in kleinere überschaubarere Teilaufträge dividieren.
<b>Beekeeper</b>	Imker
<b>Cols (Bootstrap)</b>	Spalte
<b>Deprecated</b>	Veraltet
<b>Detach</b>	Ablösen
<b>Errorbag</b>	Schlägt eine Validierung fehl, wird in einem Assoziativen Array als Key der Name des Eingabefelds und als Wert die Fehlermeldung hinzugefügt. Dieser sogenannte Errorbag kann anschliessend dem Formular zurückübermittelt werden, um die Fehlermeldung am entsprechenden Input auszuweisen.
<b>ErrorMsg</b>	Fehlermeldung
<b>Fire and forget</b>	Ausführen und vergessen / Nicht überprüfen, ob die Ausführung tatsächlich funktioniert hat.
<b>Forward Engineering</b>	MySQL Workbench erstellt eine Datenbank anhand eines ERDs.
<b>Jurisdiction</b>	Zuständigkeit, bspw. Imker – Region
<b>Lat</b>	Latitude
<b>Lon</b>	Longitude
<b>Middleware</b>	Middlewares im Laravel Filtern eingehende HTTP-Requests
<b>Notification</b>	Kurze und unkomplizierte Nachricht welche einem Benutzer übermittelt wird mit wichtigen Informationen.
<b>Numeric</b>	Numerischer Wert
<b>Pluck</b>	Alle Werte für einen bestimmten Key aus einer Collection abrufen
<b>Proximity</b>	Räumliche Nähe
<b>Queue-Worker</b>	«Arbeiter» (Prozess) welcher eine Liste/Warteschlange von Aufträgen abarbeitet in einer sogenannten Queue.
<b>Reverse</b>	Umgekehrt
<b>S&amp;R</b>	Schutz und Rettung
<b>Template-Engine</b>	Zur Laufzeit ersetzt die Template-Engine Variablen in einer Vorlagendatei durch tatsächliche Werte und wandelt die Vorlage in eine HTML-Datei um, welche dann an den Client gesendet wird.
<b>Trait</b>	Eigenschaft / Code Teile welcher einer Klasse hinzugefügt werden.

## 23 LITERATURVERZEICHNIS

---

- Bootstrap. (2022). *Bootstrap Docs*, 5.1.3. Abgerufen am 17. März 2022 von <https://getbootstrap.com/docs/5.1/>
- Google. (2022). *Google Maps Dokumentation*. Abgerufen am 11. März 2022 von <https://developers.google.com/maps/documentation/urls/get-started>
- Google. (2022). *Google Maps Dokumentation Reverse geocoding request API*. Abgerufen am 11. März 2022 von <https://developers.google.com/maps/documentation/geocoding/requests-reverse-geocoding>
- Laravel LLC. (2022). *Laravel Docs*, 8.x. Abgerufen am 11. März 2022 von <https://laravel.com/docs/8.x>
- Maytham. (29. Mai 2020). *stackoverflow*. Abgerufen am 17. März 2022 von <https://stackoverflow.com/questions/31539727/laravel-password-validation-rule>
- Nexmo Github. (03. Februar 2022). *github Nexmo Package*. Abgerufen am 21. März 2022 von <https://github.com/Nexmo/nexmo-laravel>
- Nominatim. (2022). *Nominatim Documentation*. Abgerufen am 11. März 2022 von <https://nominatim.org/release-docs/develop/api/Reverse/>
- Prüfungskommision 19. (2022). *pk19*. Abgerufen am 11. März 2022 von [QV-Leitfaden\\_2022.pdf](https://pk19.ch/wp-content/uploads/2021/11/QV-Leitfaden_2022.pdf): [https://pk19.ch/wp-content/uploads/2021/11/QV-Leitfaden\\_2022.pdf](https://pk19.ch/wp-content/uploads/2021/11/QV-Leitfaden_2022.pdf)
- Prüfungskommision 19. (2022). *pk19*. Abgerufen am 11. März 2022 von Kriterienkatalog-Standardkriterien\_2022.pdf: [https://pk19.ch/wp-content/uploads/2021/11/Kriterienkatalog-Standardkriterien\\_2022.pdf](https://pk19.ch/wp-content/uploads/2021/11/Kriterienkatalog-Standardkriterien_2022.pdf)
- Rahman, Z. (05. August 2020). *medium*. Abgerufen am 17. März 2022 von Laravel Redirect Login: <https://medium.com/fabcoding/laravel-redirect-users-according-to-roles-and-protect-routes-bde324fe1823#2303>
- Railware. (2022). *Mailtrap Guide*. Abgerufen am 11. März 2022 von <https://help.mailtrap.io/article/12-getting-started-guide>
- Tgugnani. (2018). *5balloons*. Abgerufen am 17. März 2022 von <https://5balloons.info/password-regex-validation-laravel-authentication/amp/>
- Vonage. (2022). *Vonage Dokumentation*. Abgerufen am 11. März 2022 von <https://developer.vonage.com/documentation>
- webdevetc. (2. Januar 2020). *webdevetc*. Abgerufen am 14. März 2022 von Laravel Naming Conventions: <https://webdevetc.com/blog/laravel-naming-conventions/>
- Wohnert, M. (25. Januar 2021). *Software-Testing-Methoden*. Abgerufen am 15. März 2022 von <https://www.software-testing.academy/software-testing-methoden.html>

## 24 ABBILDUNGSVERZEICHNIS

---

Abbildung 1 Zeitplan .....	14
Abbildung 2 Nominatim JSON Response.....	28
Abbildung 3 Vonage Send Basic SMS .....	29
Abbildung 4 Mockup: Index Seite.....	31
Abbildung 5 Mockup: Imker Registrierung.....	32
Abbildung 6 Mockup: Imker PLZ Zuweisung .....	32
Abbildung 7 Mockup: Profil Ergänzung .....	33
Abbildung 8 Mockup: Auftragserstellung.....	33
Abbildung 9 Mockup: Nominatim PLZ Fehler.....	33
Abbildung 10 Mockup: Imker Auftrag akzeptieren .....	34
Abbildung 11 Datenbankmodell v01 .....	35
Abbildung 12 Datenbankmodell v02 .....	36
Abbildung 13 Sequenzdiagramm .....	37
Abbildung 14 Blade Templat Layout .....	63
Abbildung 15 Imker Datenbank Migration.....	64
Abbildung 16 Model-Relation Anpassung .....	65
Abbildung 17 Contract factory .....	66
Abbildung 18 HasFactory Model Trait.....	66
Abbildung 19 Main Database Seeder .....	67
Abbildung 20 Beekeeper Account Seeder.....	67
Abbildung 21 Beekeeper Contract Seeder .....	68
Abbildung 22 S&R Admin Account Seeder .....	68
Abbildung 23 Postcode Seeder .....	68
Abbildung 24 Login Rolebased redirect.....	69
Abbildung 25 Component Blade und Model .....	70
Abbildung 26 Component Model .....	70
Abbildung 27 Component Blade .....	71
Abbildung 28 Formular mit Components.....	71
Abbildung 29 Registrierung Inputvalidierung .....	72
Abbildung 30 Telefonnummer Regex Überprüfung.....	72
Abbildung 31 Passwort Regex Überprüfung .....	73
Abbildung 32 Register Error Nachrichten.....	73
Abbildung 33 Composer.json .....	74
Abbildung 34 Telefonnummer Entfernen von Leer und Trennzeichen.....	74
Abbildung 35 Telefonnummer Formatierung .....	75
Abbildung 36 Telefonnummer Formatierung Überprüfung.....	75
Abbildung 37 Create Beekeeper und User .....	76
Abbildung 38 BeekeeperOnly Middleware .....	77
Abbildung 39 Middleware Kernel Registrierung .....	77
Abbildung 40 Middleware Routes.....	78
Abbildung 41 Auth Middleware redirect Anpassung .....	78
Abbildung 42 Profile .....	79
Abbildung 43 Register Components anpassungen.....	79
Abbildung 44 Profil Components .....	79
Abbildung 45 Bootstrap View Teilung.....	80
Abbildung 46 Zugewiesene Postcodes im Blade .....	80
Abbildung 47 Zugewiesene Postcodes Darstellung.....	80

Abbildung 48 Hidden <div> für Postcodes im Formular.....	81
Abbildung 49 Listenelemente Hinzufügung eines Deletebuttons.....	81
Abbildung 50 JS neues Input Element erstellen .....	81
Abbildung 51 JS Delete Funktion.....	82
Abbildung 52 LiveSearch HTML Input und Output.....	82
Abbildung 53 Ajax Request.....	82
Abbildung 54 SearchController für Ajax Request.....	83
Abbildung 55 Ajax Request Erfolg .....	83
Abbildung 56 Suchresultate in der View anzeigen.....	83
Abbildung 57 Alte Suchergebnisse löschen.....	84
Abbildung 58 Suchergebniss dem Formular hinzufügen.....	84
Abbildung 59 Suchergebnis Delete Button Hinzufügen .....	85
Abbildung 60 Suchergebnis im Formular löschen.....	85
Abbildung 61 LiveSearch Ergebnis Frontend.....	85
Abbildung 62 MultiSearch .....	86
Abbildung 63 Zuteilung der PLZ Routes .....	86
Abbildung 64 PLZ Zuteilung Validierung.....	87
Abbildung 65 PLZ Zuteilung .....	87
Abbildung 66 Alert Component im Formular.....	88
Abbildung 67 Alert Component Blade .....	88
Abbildung 68 Number Component im Formular .....	88
Abbildung 69 Number Component Blade .....	88
Abbildung 70 Component Default Required Constructor .....	89
Abbildung 71 Component Default Formular .....	89
Abbildung 72 Component Default Blade.....	89
Abbildung 73 Component optionaler Platzhalter .....	89
Abbildung 74 Auftrag erstellen Formular mit Components.....	90
Abbildung 75 Auftrag erstellen Datenvorbereitung für Validierung.....	90
Abbildung 76 Auftrag erstellen Validierung .....	91
Abbildung 77 Auftrag erstellen Validierung failed .....	91
Abbildung 78 Auftrag erstellen Validierung, ob PLZ manuell erfolgte.....	91
Abbildung 79 GeoLocationController Nominatim PLZ ermittlung .....	92
Abbildung 80 Nominatim Validierung .....	93
Abbildung 81 Auftrag erstellen .....	93
Abbildung 82 Config anpassung für Nexmo .....	94
Abbildung 83 Nexmo Key, Secret und From in .env Datei .....	94
Abbildung 84 Beekeeper Model Notifiable trait .....	94
Abbildung 85 Beekeeper Ziel Telefonnummer für Nexmo Notifikation .....	94
Abbildung 86 Contract Applicable Notification .....	95
Abbildung 87 Notifikation via.....	95
Abbildung 88 Notifikation Nachricht.....	95
Abbildung 89 Zuständige Imker ausfindig machen und kontaktieren per Notifikation.....	96
Abbildung 90 Imker Notifikation SMS .....	97
Abbildung 91 Auftrag applicable Validierung.....	97
Abbildung 92 Auftrags-Annahme .....	98
Abbildung 93 Auftrag Details Notifikationsnachricht.....	99
Abbildung 94 GoogleMaps-Pin generieren .....	99
Abbildung 95 Imker Auftragsdetails zusenden .....	99
Abbildung 96 ImkerSearch: Blade .....	100

Abbildung 97 ImkerSearch: PLZ.....	100
Abbildung 98 ImkerSearch aufrufen mit onclick Event auf PLZ .....	100
Abbildung 99 Imker suchen und im Frontend anzeigen .....	101
Abbildung 100 ImkerSearch: PLZ Validierung .....	101
Abbildung 101 Imker suchen und selektierte Daten zurückgeben .....	101
Abbildung 102 Beekeeper FullName Model Attribute.....	102
Abbildung 103 Formatierung der Telefonnummer Rückgängig machen.....	102
Abbildung 104 Beekeeper Reverse Format Phone Model Attribute .....	102
Abbildung 105 Beekeeper Zuweisungen als String umformen Model Attribute .....	103
Abbildung 106 Selektierte Rückgabewerte Überprüfung .....	103
Abbildung 107 Proximity Search Query .....	104
Abbildung 108 Proximity Search Query Erweiterung.....	104
Abbildung 109 Proximity Search Query in Laravel .....	105
Abbildung 110 Imker sortieren nach IDs der Proximity .....	105

## 25 ANHANG

### 25.1 RESULTAT

In den nachkommenden Bildern handelt es sich um per Zufall generierten Personen.

#### 25.1.1 Index

The screenshot shows the IPA BeeRes website interface. At the top, there is a blue header bar with the IPA logo. Below it, the main content area is divided into two sections: 'Imker Search' on the left and 'Login' on the right. The 'Imker Search' section contains a search bar labeled 'PLZ Search' with a magnifying glass icon. The 'Login' section contains fields for 'E-Mail Address\*' and 'Password\*', followed by 'Login' and 'Register' buttons. A small link 'Forgot Your Password?' is also visible. At the bottom of the page, there is a footer bar with the text '©2022 IPA C.O'Connor'.

##### 25.1.1.1 *Imker Search*

###### 25.1.1.1.1 PLZ Search nach Zahl

The screenshot shows the 'Imker Search' results for the input '87'. The results list three entries: '8187 | Weiach', '8487 | Zell (ZH)', and '8700 | Küsnacht (ZH)'. The results are presented in a scrollable list with a vertical scrollbar on the right side of the list area.

## 25.1.1.1.2 PLZ Search nach Namen

The screenshot shows a search interface titled "Imker Search". A search bar at the top contains the text "lang". Below the search bar, two search results are listed in a table-like format:

8135   Langnau am Albis, Horgen
8153   Rümlang

## 25.1.1.1.3 Imker Search

The screenshot shows a search interface titled "Imker Search". A search bar at the top contains the text "8700 | Küschnacht (ZH)". Below the search bar, three search results are listed in a table-like format:

Jon Doe 076 116 65 82 8421 Dättlikon, 8700 Küschnacht (ZH), 8704 Herrliberg, 8706 Meilen
Emie Abernathy 076 803 01 56 8426 Lufingen, 8633 Bubikon, 8708 Männedorf, 8713 Stäfa
Mikayla Kris 078 288 49 93 8193 Eglisau, 8708 Männedorf, 8406 Winterthur, 8057 Zürich

### 25.1.1.2 Login Error

The screenshot shows a login interface with a red error message at the top: "Error E-Mail Address or Password is wrong!". Below it is a "Login" button. The "E-Mail Address\*" field contains "jon@doe.ch". The "Password\*" field is redacted. Below the fields are "Login" and "Register" buttons. A "Forgot Your Password?" link is at the bottom.

Error E-Mail Address or Password is wrong!

Login

E-Mail Address\*

jon@doe.ch

Password\*

Forgot Your Password?

Login

Register

### 25.1.2 Register

The screenshot shows a registration form titled "Beekeeper-Register". It includes fields for Firstname, Lastname, Phone, E-Mail, Password, and Confirm Password. There is also a checkbox for a privacy statement and a "Register" button. The form is set against a blue header and footer.

IPA

Beekeeper-Register

Firstname \*

Lastname \*

Phone \*

E-Mail \*

Confirm Password\*

I hereby confirm that I have taken note that my contact information may be disclosed for the beekeeper search\*

Register

©2022 IPA C.O'Connor

### 25.1.2.1 Register Errors

The image contains two side-by-side screenshots of a web form titled "Beekeeper-Register".

**Left Screenshot:**

- Firstname \***: Field is empty, highlighted in red. Error message: "The firstname field is required."
- Lastname \***: Field is empty, highlighted in red. Error message: "The lastname field is required."
- Phone \***: Field contains "076 581 35 96", highlighted in red. Error message: "The phone field is required."
- E-Mail \***: Field is empty, highlighted in red. Error message: "The email field is required."
- Password\***: Field is empty, highlighted in red.
- Confirm Password\***: Field is empty.
- Agb**: A checkbox labeled "I hereby confirm that I have taken note that my contact information may be disclosed for the beekeeper search\*" is empty.
- Agb Error**: Message: "The agb field is required."

**Right Screenshot:**

- Firstname \***: Field contains "Firstname", highlighted in red. Error message: "The firstname field is required."
- Lastname \***: Field contains "Dettlin", highlighted in green with a checkmark.
- Phone \***: Field contains "(0041) 76 581 35-96", highlighted in red. Error message: "Not a valid Format, please use: +41, 0041, 07x or 044".
- E-Mail \***: Field contains "det", highlighted in red. Error message: "The email must be a valid email address."
- Password\***: Field is empty, highlighted in red. Error message: "Password must contain at least 1x Uppercase, 1x Lowercase, a number and a special character!"
- Confirm Password\***: Field is empty.
- Agb**: A checkbox labeled "I hereby confirm that I have taken note that my contact information may be disclosed for the beekeeper search\*" is empty.
- Agb Error**: Message: "The agb field is required."

**Buttons:**

- Both screenshots show a blue "Register" button at the bottom.

### 25.1.3 Profil

The image shows a screenshot of a profile update form titled "Profile".

**Fields:**

- Firstname \***: Value "Jon"
- Lastname \***: Value "Doe"
- Phone \***: Value "076 116 65 82"
- E-Mail \***: Value "jon@idox.ch"
- Old Password \***: Placeholder "Old Password"
- Password\***: Placeholder "Password"
- Confirm Password\***: Placeholder "Confirm Password"

**Buttons:**

- Update**: Green button
- Delete**: Red button

**Footer:**

- PLZ - Jurisdiction
- ©2022 IPA C.O'Connor

#### 25.1.4 Jurisdictions

The screenshot shows a web-based application interface. At the top, there's a blue header bar with the IPA logo and the word "Jurisdictions". On the right side of the header, there's an email address "joni@iono.ch" and a small profile icon. Below the header, the main content area is divided into two sections: "Active PLZ" on the left and "Search PLZ" on the right.

**Active PLZ:** This section contains a list of four active postal codes (PLZ) with their corresponding locations:

- 8421 | Dättlikon
- 8700 | Küsnacht (ZH)
- 8704 | Hettlingen
- 8706 | Meilen

Each item in the list has a small red square button with a white "X" next to it. At the bottom of this section is a green horizontal button labeled "Speichern".

**Search PLZ:** This section features a search input field with the placeholder "PLZ Search" and a magnifying glass icon to its right. Below the input field is a large, empty white area.

At the very bottom of the page, there's a thin blue footer bar with some small icons and the text "©2022 IPA C.O'Connor".

##### 25.1.4.1 PLZ Search Zahl

This screenshot shows the "Search PLZ" results for the query "870". The search bar at the top contains the number "870". Below the search bar is a list of results:

- 8702 | Zollikon
- 8703 | Erlenbach (ZH)
- 8707 | Uetikon am See
- 8708 | Männedorf

#### 25.1.4.2 PLZ Search Name

Search PLZ

 Q

8122   Maur
8123   Maur
8124   Maur
8162   Steinmaur

#### 25.1.4.3 Hinzufügen / Löschen einer Zuweisung

Active PLZ

8700   Küsnacht (ZH)	X
8704   Herrliberg	X
8706   Meilen	X
8117   Fällanden	X
8122   Maur	X

Speichern

### 25.1.5 Contract Create

The screenshot shows the 'Contract-Create' form. At the top, there are fields for Latitude (47.39036) and Longitude (8.49087). Below these are sections for 'Contact Information' (Firstname, Lastname, Phone), 'Additional Information', and a 'Create Contract' button. The status bar at the bottom indicates '©2022 IPA C.O'Connor'.

#### 25.1.5.1 Nominatim Error

The screenshot shows the 'Contract-Create' form with validation errors. A red error message at the top states 'Error Please Enter Postcode manually or change Latitude/Longitude.' The 'Latitude' field contains '48', 'Longitude' contains '9', and 'Postcode' contains '8048'. Below these fields, another error message says 'Please Enter Postcode manually or change Latitude/Longitude.' The 'Contact Information' section includes fields for Firstname, Lastname, Phone, and Additional Information, all of which are currently empty. The 'Create Contract' button is at the bottom.

### 25.1.5.2 PLZ Error

Contract-Create

Latitude \*

 ✓

Longitude \*

 ✓

Postcode

 ⓘ  
The selected plz is invalid.

Contact Information

Firstname

 ✓

Lastname

 ✓

Phone

 ✓

Additional Information

 ✓

**Create Contract**

### 25.1.5.3 Lat/Long Validierung

Contract-Create

Latitude \*

 ⓘ  
The lat field is required.

Longitude \*

 ⓘ  
The lon must be at least 8.

Contact Information

Firstname

 ✓

Lastname

 ✓

Phone

 ✓

Additional Information

 ✓

**Create Contract**

Contract-Create

Latitude \*

 ⓘ  
The lat must not be greater than 48.

Longitude \*

 ⓘ  
The lon must be at least 8.

Contact Information

Firstname

 ✓

Lastname

 ✓

Phone

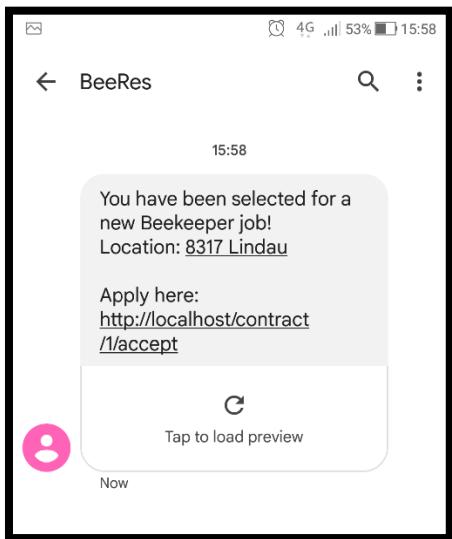
 ✓

Additional Information

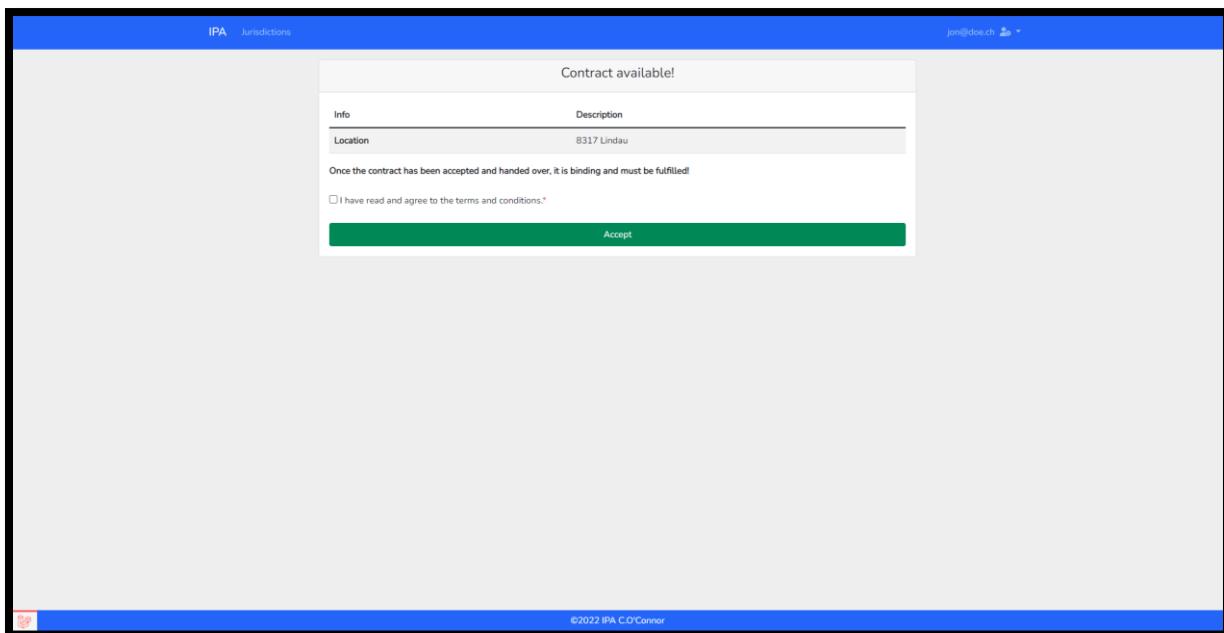
 ✓

**Create Contract**

### 25.1.6 SMS Notification applicable



### 25.1.7 Contract Accept



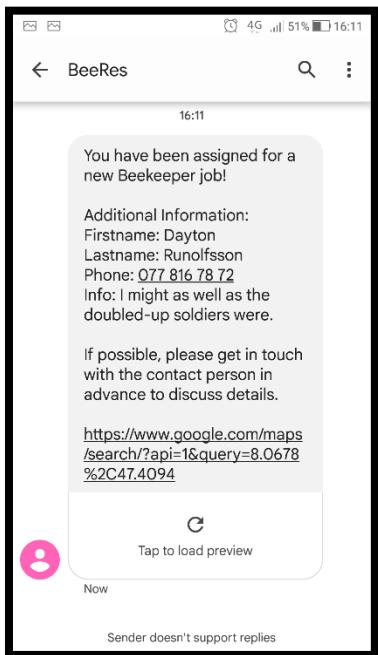
### 25.1.7.1 Error

The screenshot shows a web-based application interface for accepting a contract. At the top, there is a blue header bar with the IPA logo and a user dropdown. The main content area has a white background with a central card. The card title is "Contract available!". Inside the card, there is a table with two columns: "Info" and "Description". Under "Info", there is a single entry "Location" with the value "8317 Lindau". Below the table, a note states: "Once the contract has been accepted and handed over, it is binding and must be fulfilled". A checkbox labeled "I have read and agree to the terms and conditions." is present, followed by the text "The terms and conditions are required!". A large green button at the bottom right of the card is labeled "Accept". At the very bottom of the page, there is a footer bar with the text "©2022 IPA C.O'Connor".

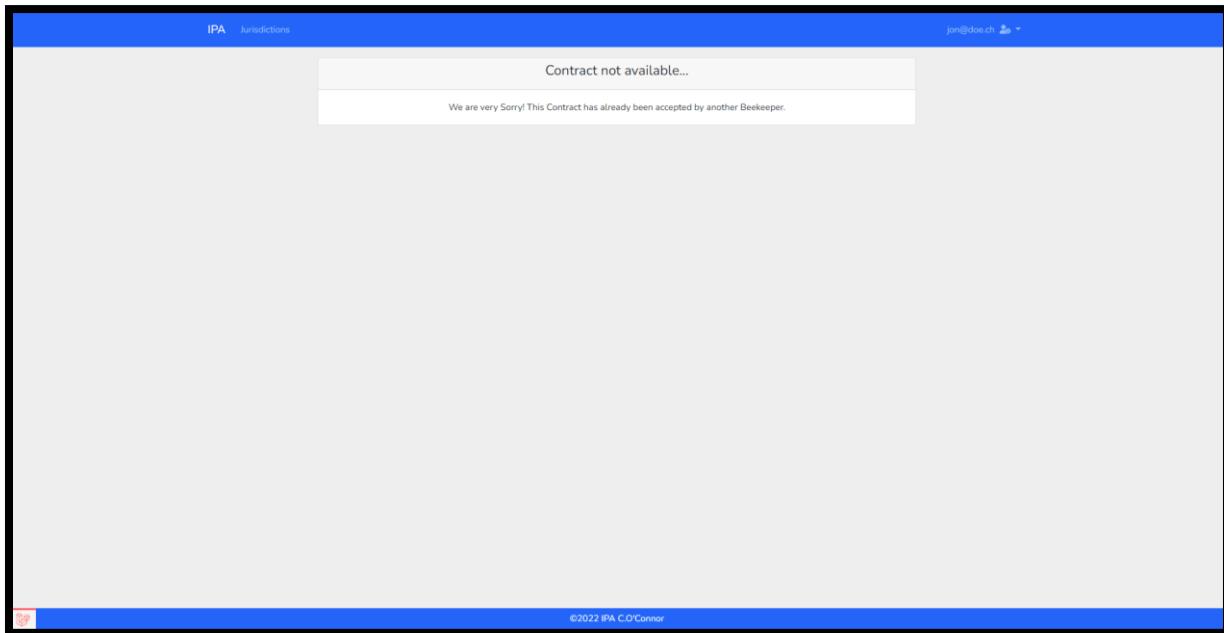
### 25.1.7.2 Accept Success

The screenshot shows the same web-based application interface as the previous one, but now in a successful state. The main content area features a central card with the title "Contract successfully taken". Below this, a message reads: "Congratulations! You have received the Contract. All informations will follow shortly. Please check your SMS-Inbox." The card contains a table with two columns: "Description" and "Value". The entries are: Latitude (8.0678), Longitude (47.4094), Location (8317 Lindau), Firstname (Dayton), Lastname (Runoffson), Phone (077 816 78 72), and Additional information (I might as well as the doubled-up soldiers were.). At the bottom of the page, there is a footer bar with the text "©2022 IPA C.O'Connor".

### 25.1.8 SMS Notification details



### 25.1.9 Contract already taken



## 25.1.10 Mobile Views (Abmessungen: Samsung Galaxy S20 Ultra)

IPA

Imker Search

8700 | Küsnacht (ZH)

Christopher O'Connor  
8700 Küsnacht (ZH), 8702 Zollikon, 8708 Männedorf

Jon Doe  
079 978 76 78  
8340 Hinwil, 8623 Wetzikon (ZH), 8704 Herrliberg,  
8544 Wiesendangen, 8453 Kleinandelfingen

Nasir Wilkinson  
076 036 03 27  
8154 Oberholz, 8805 Richterswil, 8409 Winterthur

Login

E-Mail Address\*

Password\*

[Forgot Your Password?](#)

©2022 IPA C.O'Connor

IPA

Firstname \*

Lastname \*

Phone \*

E-Mail \*

Old Password \*

Confirm Password\*

©2022 IPA C.O'Connor

The image displays two side-by-side screenshots of a mobile application interface for IPA BeeRes.

**Screenshot 1 (Left):**

- Header:** IPA (blue bar), Jurisdictions (text).
- User Profile:** Email (@gmail.com) and Logout button.
- Active PLZ:** A list of saved Postcode-Location-Zipcode (PLZ) entries:
  - 8700 | Küsnacht (ZH)
  - 8702 | Zollikon
  - 8708 | Männedorf
- Buttons:** Speichern (green) and Search PLZ (button).
- Search PLZ:** Input field with placeholder "PLZ Search" and a magnifying glass icon.
- Footer:** ©2022 IPA C.O'Connor.

**Screenshot 2 (Right):**

- Header:** IPA (blue bar).
- Title:** Active PLZ.
- List:** A list of active PLZ entries:
  - 8700 | Küsnacht (ZH) (red X)
  - 8702 | Zollikon (red X)
  - 8708 | Männedorf (red X)
  - 8704 | Herrliberg (green X)
- Buttons:** Speichern (green) and Search PLZ (button).
- Search PLZ:** Input field with placeholder "870" and a magnifying glass icon. Below it is a list of search results:
  - 8703 | Erlenbach (ZH)
  - 8706 | Meilen
  - 8707 | Uetikon am See
- Footer:** ©2022 IPA C.O'Connor.

IPA

Contract successfully taken

Congratulations! You have received the Contract.  
All informations will follow shortly. Please check your SMS-Inbox.

Description	Value
Latitude	47.31901
Longitude	8.58059
Location	8700 Küsnacht (ZH)
Firstname	Ueli
Lastname	Meier
Phone	0449122235
Additional Information	Unter dem Dach

©2022 IPA C.O'Connor

IPA

Contract-Create

Latitude \*

Longitude \*

## 25.2 QUELLCODE

app/Http/Controllers/Auth/LoginController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Illuminate\Support\Facades\Auth;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to the profile screen. The controller uses a trait
    | to conveniently provide its functionality.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect Users to based on role
     * @return string
     */
    public function redirectTo():string
    {
        return Auth::user()->is_admin ? '/contract/create':'/profile';
    }

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

app/Http/Controllers/Auth/RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\Beekeeper;
use App\Providers\RouteServiceProvider;
use App\Models\User;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new beekeepers as well as their
    | validation and creation.
    |
    */

    use RegistersUsers;

    /**
     * Where to redirect beekeepers after registration.
     *
     * @var string
     */
    protected $redirectTo = 'profile';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param array $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {

        $data['phone'] = formatPhoneNum($data['phone']);

        return Validator::make($data, [
            'firstname' => ['required', 'string', 'max:255'],
            'lastname' => ['required', 'string', 'max:255'],
            'phone' => ['required', 'regex:/^41\d{9}$/i', 'unique:beekeepers'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[@$%^&*!]).{8,}$/i'],
            'password_confirmation' => ['required', 'string', 'same:password'],
            'agb' => ['required']
        ], [
            'phone.regex' => 'Not a valid Format, please use: +41, 0041, 07x or 044',
            'password.regex' => 'Password must contain at least 1x Uppercase, 1x Lowercase, a number and a
special character!',
        ]);
    }

    /**
     * Create a new user and beekeeper
     * Create user instance after a valid registration.
     *
     * @param array $data
     * @return User
     */
    protected function create(array $data)
    {
        $user = User::create([
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);

        Beekeeper::create([
            'firstname' => $data['firstname'],
            'lastname' => $data['lastname'],
        ]);
    }
}
```

```

        'phone'      => formatPhoneNum($data['phone']),
        'user_id'    => $user->id,
    ]);

    return $user;
}

}

```

app/Http/Controllers/BeekeeperController.php

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class BeekeeperController extends Controller
{

    /**
     * Show the form to edit Beekeeper (Profile).
     *
     * @return \Illuminate\Http\Response
     */
    public function profile()
    {
        return view('models.beekeeper.profile', [
            'beekeeper' => auth()->user()->beekeeper,
        ]);
    }

    /**
     * Show the form to edit Jurisdictions.
     *
     * @return \Illuminate\Http\Response
     */
    public function jurisdiction()
    {
        return view('models.beekeeper.jurisdiction', [
            'postcodes' => auth()->user()->beekeeper->postcodes
        ]);
    }

    /**
     * Update jurisdictions of Beekeeper in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function updateJurisdiction(Request $request)
    {

        $validator = Validator::make($request->all(), [
            'delJur'   => ['array'],
            'addJur'   => ['array'],
            'delJur.*' => ['integer', 'exists:App\Models\Postcode,id'], // Validate each content of Array
            'addJur.*' => ['integer', 'exists:App\Models\Postcode,id'], // Validate each content of Array
        ]);

        if (!$validator->fails()) {

            // Get Relation of Postcodes from Authenticated Beekeeper
            $postcodes = auth()->user()->beekeeper->postcodes();

            if(isset($validator->validated()['addJur'])) $postcodes->attach($validator->validated()['addJur']);
            if(isset($validator->validated()['delJur'])) $postcodes->detach($validator->validated()['delJur']);

        }

        return redirect(route('jurisdiction'));
    }
}

```

app/Http/Controllers/ContractController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Contract;
use App\Models\Postcode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class ContractController extends Controller
{

    /**
     * Show the form for creating a new contract.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('models.contract.create');
    }

    /**
     * Store a newly created contract in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {

        // Prepare Request for Validation: Format phone and add created_by
        $request->merge([
            'contact_phone' => replaceEmptyChars($request->all()['contact_phone']),
            'created_by'     => auth()->user()->id,
        ]);

        $validator = Validator::make($request->all(), [
            'lat'           => ['required', 'numeric', 'min:47', 'max:48'],
            'lon'           => ['required', 'numeric', 'min:8', 'max:9'],
            'plz'           => ['nullable', 'numeric', 'min:8000', 'max:9000'],
            'exists:App\Models\Postcode,postcode' => [
                'contact_firstname' => ['nullable', 'string'],
                'contact_lastname'  => ['nullable', 'string'],
                'contact_phone'     => ['nullable', 'regex:/^(?:0041|0|\+41)(\d{9})$/'],
                'info'              => ['nullable', 'string'],
                'created_by'        => ['required', 'exists:App\Models\User,id']
            ],
            [
                'contact_phone.regex' => 'Not a valid Format, please use: +41, 0041, 07x or 04x'
            ]
        ]);

        if($validator->fails()) {

            // If Key exists:
            // 1st Validation Nominatim Failed to find PLZ & input PLZ was shown
            // 2nd Validation either Nominatim Failed again or PLZ failed and input PLZ needs to be shown
            again:
            if(array_key_exists('plz', $request->all())) {
                $validator->errors()->add('plz-api', 'Invalid Postcode!');
            }

            return back()->withErrors($validator)->withInput();
        }

        $validated = $validator->validated();

        // If Key exists:
        // 1st Validation Nominatim Failed to find PLZ
        // 2nd Validation PLZ was entered manually and needs to be added to Contract
        if(array_key_exists('plz', $validated) && !is_null($validated['plz'])) {

            $validated['postcode_id'] = Postcode::where('postcode', $validated['plz'])->first()->id;
        } else {

            // Try to get PLZ from Nominatim
            $postcode = GeoLocationController::getPlzFromLatLon($validated['lat'], $validated['lon']);

            // If Nominatim Failed to provide a PLZ
            if(is_null($postcode)) {
                // Add validation failed keys + messages.
                $validator->errors()->add('plz-api', 'Please Enter Postcode manually or change Latitude/Longitude.');
                $validator->errors()->add('lat', ' ');
                $validator->errors()->add('lon', ' ');
            }
        }
    }
}
```

```

        $validator->errors()->add('plz', 'Please Enter Postcode manually or change
Latitude/Longitude.');
        return back()->withErrors($validator)->withInput();
    }
    $validated['postcode_id'] = $postcode->id;
}
// Create new Contract with Validated Data
$contract = Contract::create($validated);
// Notify applicable Beekeeper
NotificationController::notifyBeekeeperNewContract($contract);
return redirect(route('contract.show', $contract->id));
}

/**
 * Display the specified contract for Admins.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $contract = Contract::findOrFail($id);
    return view('models.contract.show', ['contract' => $contract]);
}

/**
 * Display the form to accept a contract as beekeeper.
 *
 * @param int $id
 * @return \Illuminate\Http\Response
 */
public function accept($id)
{
    $contract = Contract::findOrFail($id);

    // Validate if contract has already been taken by a beekeeper
    if($contract->beekeeper) {
        // If contract belongs to this beekeeper show detail information
        if($contract->beekeeper->id == auth()->user()->beekeeper->id) {
            return redirect(route('contract.accept.success', $contract));
        }
        // If contract belongs to another beekeeper show contract taken
        return redirect(route('contract.taken'));
    }

    // Validate if beekeeper is applicable to accept this contract
    if(! $contract->beekeepers->pluck('id')->contains(auth()->user()->beekeeper->id)) return
    redirect(route('index'));

    return view('models.contract.accept', ['contract' => $contract]);
}

/**
 * Display the form if a contract already has a beekeeper.
 *
 * @return \Illuminate\Http\Response
 */
public function taken()
{
    return view('models.contract.taken');
}

/**
 * Display the successfully accepted specified contract for Beekeeper.
 *
 * @param $id
 * @return \Illuminate\Http\Response
 */
public function success($id)
{
    $contract = Contract::findOrFail($id);

    // Validate if contract already has a beekeeper and if it is the authenticated beekeeper
    if($contract->beekeeper && $contract->beekeeper->id == auth()->user()->beekeeper->id) {
        return view('models.contract.success', ['contract' => $contract]);
    }
    return redirect(route('contract.taken'));
}
*/

```

```

    * Update specified contract with Beekeeper.
    *
    * @param Request $request
    * @param $id
    * @return \Illuminate\Http\Response
    */
public function acceptContract(Request $request, $id)
{
    $contract = Contract::findOrFail($id);

    // Validate if contract already has a beekeeper
    if($contract->beekeeper) return redirect(route('contract.taken'));

    // Validate if beekeeper is applicable to accept this contract
    if(! $contract->beekeepers->pluck('id')->contains(auth()->user()->beekeeper->id)) return
    redirect(route('index'));

    $request->validate([
        'acc' => ['required']
    ], [
        'required' => 'The terms and conditions are required!'
    ]);

    // Add beekeeper to contract
    $contract->beekeeper()->associate(auth()->user()->beekeeper)->save();

    // Notify detail Info per SMS.
    NotificationController::notifyBeekeeperContractAssigned($contract);

    return redirect(route('contract.accept.success', $contract));
}
}

app/Http/Controllers/GeoLocationController.php

```

```

<?php

namespace App\Http\Controllers;

use App\Models\Postcode;
use Illuminate\Support\Facades\Http;

class GeoLocationController extends Controller
{

    /**
     * Get PLZ from Latitude & Longitude.
     *
     * @param float $lat
     * @param float $lon
     * @return Postcode|null
     */
    public static function getPlzFromLatLon(float $lat, float $lon): ?Postcode
    {

        $res = Http::get('https://nominatim.openstreetmap.org/reverse?lat=' . $lat . '&lon=' . $lon
        . '&format=json');

        $place = json_decode($res->body());

        // Validate if postcode exists
        if(array_key_exists('address', $place) && array_key_exists('postcode', $place->address)) {

            $postcode_nom = $place->address->postcode;

            $postcode = Postcode::where('postcode', $postcode_nom)->first();

            if($postcode) return $postcode;
        }
        return null;
    }

    /**
     * Generate GoogleMaps URL Pin.
     *
     * @param float $lat
     * @param float $lon
     * @return string
     */
    public static function generateGoogleMapsPin(float $lat, float $lon):string
    {
        return 'https://www.google.com/maps/search/?api=1&query='.$lat.'%2C'.$lon;
    }
}

```

app/Http/Controllers/NotificationController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Contract;
use App\Notifications\ContractApplicableNotification;
use App\Notifications\ContractAssignedNotification;
use Illuminate\Http\Request;

class NotificationController extends Controller
{

    /**
     * Find the closest Beekeepers and notify about the newly created contract.
     *
     * @param Contract $contract
     * @return void
     */
    public static function notifyBeekeeperNewContract(Contract $contract)
    {

        // Find all beekeepers from specific region
        $beekeepers = $contract->postcode->beekeepers;

        // If no beekeepers could be found, find the closest Beekeepers
        if(count($beekeepers) == 0) {
            $beekeepers = SearchController::findClosestBeekeeper($contract->postcode->postcode)->take(2);
        }

        foreach ($beekeepers as $beekeeper) {

            // Add beekeeper to be applicable to contract
            $beekeeper->contracts_applicable()->attach($contract);
            // Notify beekeeper about the new applicable contract
            $beekeeper->notify(new ContractApplicableNotification($contract));
        }
    }

    /**
     * Notify the beekeeper about all the Detailinformation of the accepted contract.
     *
     * @param Contract $contract
     * @return void
     */
    public static function notifyBeekeeperContractAssigned(Contract $contract)
    {
        $contract->beekeeper->notify(new ContractAssignedNotification($contract));
    }
}
```

app/Http/Controllers/SearchController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Beekeeper;
use App\Models\Postcode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Validator;

class SearchController extends Controller
{

    /**
     * LiveSearch Postcodes by postcode or name for beekeepers.
     *
     * @param Request $request
     * @return \Illuminate\Http\Response
     */
    public static function searchPLZ(Request $request)
    {

        if($request->input('search')) {

            $column = is_numeric($request->input('search')) ? 'postcode' : 'name';

            $data = DB::table('postcodes')
                ->where($column, 'LIKE', '%' . $request->input('search') . "%")
                ->whereNotIn('id', auth()->user()->beekeeper->postcodes->pluck('id'))
                ->limit(10)->get();

            return Response($data);
        }

        return Response(Postcode::all()->take(10));
    }

    /**
     * LiveSearch Postcodes by postcode or name for guests.
     *
     * @param Request $request
     * @return \Illuminate\Http\Response
     */
    public static function guestSearchPLZ(Request $request)
    {

        if($request->input('search')) {

            $column = is_numeric($request->input('search')) ? 'postcode' : 'name';

            $data = DB::table('postcodes')
                ->where($column, 'LIKE', '%' . $request->input('search') . "%")
                ->limit(5)->get();

            return Response($data);
        }

        return Response(Postcode::all()->take(5));
    }

    /**
     * LiveSearch the closest proximity of Beekeepers to postcode for guests.
     *
     * @param Request $request
     * @return \Illuminate\Http\Response
     * @throws \Illuminate\Validation\ValidationException
     */
    public static function guestSearchBeekeeper(Request $request)
    {

        // Validate if provided postcode exists
        $validator = Validator::make($request->all(), [
            'postcode_id' => ['required', 'exists:App\Models\Postcode,id'],
        ]);

        if(! $validator->fails()) {

            $validated = $validator->validated();
            $postcode = Postcode::find($validated['postcode_id']);

            $beekeepers = self::findClosestBeekeeper($postcode->postcode);

            // Only return the full Name, formatted Phone Number and Jurisdictions
        }
    }
}
```

```

        return Response($beekeepers->take(5)->map->only(['fullName', 'reverseFormattedPhone',
'jurisdictionsToString']));
    }

    return null;
}

/**
 * Search the closest proximity Beekeepers by postcode.
 *
 * @param $postcode
 * @return \Illuminate\Support\Collection
 */
public static function findClosestBeekeeper($postcode)
{
    $sortedBeekeeperIds = DB::table('beekeepers')
        ->join('beekeeper_postcode', 'beekeeper_postcode.beekeeper_id', '=', 'beekeepers.id')
        ->join('postcodes', 'beekeeper_postcode.postcode_id', '=', 'postcodes.id')
        ->selectRaw('beekeepers.id, min(abs('. $postcode.' - postcodes.postcode)) AS mindiff')
        ->groupBy('beekeepers.id')
        ->orderBy('mindiff')
        ->pluck('beekeepers.id');

    $beekeepers = Beekeeper::all();

    // Sort beekeepers based on sequence of ids in array $sortedBeekeeperIds
    return $sortedBeekeeperIds->map(function($id) use($beekeepers) {
        return $beekeepers->where('id', $id)->first();
    });
}
}

```

## app/Http/Middleware/AdminOnly.php

```

<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class AdminOnly
{
    /**
     * Handle an incoming request and validate if user is an admin.
     *
     * @param \Illuminate\Http\Request $request
     * @param Closure(\Illuminate\Http\Request) $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if(auth()->user()->is_admin) {
            return $next($request);
        }

        return redirect(route('index'));
    }
}

```

app/Http/Middleware/BeekeeperOnly.php

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class BeekeeperOnly
{
    /**
     * Handle an incoming request and validate if user is a beekeeper.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure(\Illuminate\Http\Request) $next
     */
    public function handle(Request $request, Closure $next)
    {
        if(auth()->user()->beekeeper) {
            return $next($request);
        }

        return redirect(route('index'));
    }
}
```

app/Models/Beekeeper.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Notifications\Notifiable;

/**
 * @property integer $id
 * @property integer $user_id
 * @property string $firstname
 * @property string $lastname
 * @property string $fullName
 * @property string $phone
 * @property string $phone_verified_at
 * @property string $reverseFormattedPhone
 * @property string $jurisdictionsToString
 * @property string $created_at
 * @property string $updated_at
 * @property User $user
 * @property Contract[] $contracts_applicable
 * @property Postcode[] $postcodes
 * @property Contract[] $contracts
 */
class Beekeeper extends Model
{
    use HasFactory, Notifiable;

    /**
     * The "type" of the auto-incrementing ID.
     *
     * @var string
     */
    protected $keyType = 'integer';

    /**
     * Model attributes in Database.
     *
     * @var array
     */
    protected $fillable = ['user_id', 'firstname', 'lastname', 'phone', 'phone_verified_at', 'created_at', 'updated_at'];

    /**
     * Concat firstname and lastname.
     *
     * @return string
     */
    public function getFullNameAttribute():string
    {
        return $this->firstname . ' ' . $this->lastname;
    }

    /**
     * Reverse Phone format from with countrycode 41x to 0x for Views.
     */
}
```

```
/*
 * @return string
 */
public function getReverseFormattedPhoneAttribute():string
{
    return reverseFormatPhoneNum($this->phone);
}

/**
 * Concat jurisdictions to string for Views.
 *
 * @return string
 */
public function getJurisdictionsToStringAttribute():string
{
    $res = '';

    foreach ($this->postcodes()->pluck('name', 'postcode') as $postcode => $name) {
        $res .= $postcode.'.'.$name.', ';
    }

    return substr($res, 0, strlen($res) - 2);
}

/**
 * Route notifications for the Nexmo channel.
 *
 * @param \Illuminate\Notifications\Notification $notification
 * @return string
 */
public function routeNotificationForNexmo($notification):string
{
    return $this->phone;
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
 */
public function user()
{
    return $this->belongsTo(User::class);
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\BelongsToMany
 */
public function contracts_applicable()
{
    return $this->belongsToMany(Contract::class);
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\BelongsToMany
 */
public function postcodes()
{
    return $this->belongsToMany(Postcode::class);
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function contracts()
{
    return $this->hasMany(Contract::class);
}
```

app/Models/Contract.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * @property integer $id
 * @property integer $beekeeper_id
 * @property integer $postcode_id
 * @property integer $created_by
 * @property float $lon
 * @property float $lat
 * @property bool $hasOptionalInfo
 * @property string $contact_firstname
 * @property string $contact_lastname
 * @property string $contact_phone
 * @property string $info
 * @property string $created_at
 * @property string $updated_at
 * @property Beekeeper $beekeeper
 * @property Postcode $postcode
 * @property User $created_by_user
 * @property Beekeeper[] $beekeepers
 */
class Contract extends Model
{
    use HasFactory;

    /**
     * The "type" of the auto-incrementing ID.
     *
     * @var string
     */
    protected $keyType = 'integer';

    /**
     * Model attributes in Database.
     *
     * @var array
     */
    protected $fillable = ['beekeeper_id', 'postcode_id', 'created_by', 'lon', 'lat', 'contact_firstname',
    'contact_lastname', 'contact_phone', 'info', 'created_at', 'updated_at'];

    /**
     * Evaluate if optional fields contain data.
     *
     * @return bool
     */
    public function getHasOptionalInfoAttribute() :bool
    {
        return ($this->contact_firstname || $this->contact_lastname || $this->contact_phone || $this-
>info);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsToMany
     */
    public function beekeepers()
    {
        return $this->belongsToMany(Beekeeper::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function beekeeper()
    {
        return $this->belongsTo(Beekeeper::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function postcode()
    {
        return $this->belongsTo(Postcode::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function created_by_user()
    {
        return $this->belongsTo(User::class, 'created_by');
    }
}
```

}

app/Models/Postcode.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
 * @property integer $id
 * @property integer $postcode
 * @property string $name
 * @property string $created_at
 * @property string $updated_at
 * @property string $codeName
 * @property Beekeeper[] $beekeepers
 * @property Contract[] $contracts
 */
class Postcode extends Model
{
    use HasFactory;

    /**
     * The "type" of the auto-incrementing ID.
     *
     * @var string
     */
    protected $keyType = 'integer';

    /**
     * Model attributes in Database.
     *
     * @var array
     */
    protected $fillable = ['postcode', 'name', 'created at', 'updated at'];

    /**
     * Concat postcode and name.
     *
     * @return string
     */
    public function getCodeNameAttribute() :string
    {
        return $this->postcode . ' ' . $this->name;
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsToMany
     */
    public function beekeepers()
    {
        return $this->belongsToMany(Beekeeper::class);
    }

    /**
     * @return \Illuminate\Database\Eloquent\Relations\HasMany
     */
    public function contracts()
    {
        return $this->hasMany(Contract::class);
    }
}
```

app/Models/User.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

/**
 * @property integer $id
 * @property string $email
 * @property string $email_verified_at
 * @property string $password
 * @property boolean $is_admin
 * @property string $remember_token
 * @property string $created_at
 * @property string $updated_at
 * @property Beekeeper $beekeeper
 * @property Contract[] $contractsCreated
 */
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * @var array
     */
    protected $fillable = ['email', 'email_verified_at', 'password', 'is_admin', 'remember_token',
    'created_at', 'updated_at'];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'is_admin' => 'boolean',
    ];
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\HasOne
 */
public function beekeeper()
{
    return $this->hasOne(Beekeeper::class);
}

/**
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function contractsCreated()
{
    return $this->hasMany(Contract::class, 'created_by');
}
}
```

app/Notifications/ContractApplicableNotification.php

```
<?php

namespace App\Notifications;

use App\Models\Contract;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Notifications\Messages\NexmoMessage;
use Illuminate\Notifications\Notification;

class ContractApplicableNotification extends Notification implements ShouldQueue
{
    use Queueable;

    private $contract;

    /**
     * Create a new notification instance.
     *
     * @param Contract $contract
     */
    public function __construct(Contract $contract)
    {
        $this->contract = $contract;
    }

    /**
     * Get the notification's delivery channels.
     *
     * @param mixed $notifiable
     * @return array
     */
    public function via($notifiable)
    {
        return ['nexmo'];
    }

    /**
     * Get the Vonage / SMS representation of the notification.
     *
     * @param mixed $notifiable
     * @return \Illuminate\Notifications\Messages\NexmoMessage
     */
    public function toNexmo($notifiable)
    {
        return (new NexmoMessage)
            ->content("You have been selected for a new Beekeeper job!")
            ."\nLocation: ".$this->contract->postcode->codeName
            ."\n\nApply here:\n".
            route('contract.accept', $this->contract->id));
    }

    /**
     * Get the array representation of the notification.
     *
     * @param mixed $notifiable
     * @return array
     */
    public function toArray($notifiable)
    {
        return [
            //
        ];
    }
}
```

## app/Notifications/ContractAssignedNotification.php

```
<?php

namespace App\Notifications;

use App\Http\Controllers\GeoLocationController;
use App\Models\Contract;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Notifications\Messages\NexmoMessage;
use Illuminate\Notifications\Notification;

class ContractAssignedNotification extends Notification implements ShouldQueue
{
    use Queueable;

    private $contract;

    /**
     * Create a new notification instance.
     *
     * @param Contract $contract
     */
    public function __construct(Contract $contract)
    {
        $this->contract = $contract;
    }

    /**
     * Get the notification's delivery channels.
     *
     * @param mixed $notifiable
     * @return array
     */
    public function via($notifiable)
    {
        return ['nexmo'];
    }

    /**
     * Get the Vonage / SMS representation of the notification.
     *
     * @param mixed $notifiable
     * @return \Illuminate\Notifications\Messages\NexmoMessage
     */
    public function toNexmo($notifiable)
    {
        return (new NexmoMessage)
            ->content($this->generateTextMessage());
    }

    /**
     * Get the array representation of the notification.
     *
     * @param mixed $notifiable
     * @return array
     */
    public function toArray($notifiable)
    {
        return [
            //
        ];
    }

    /**
     * Generate SMS Textmessage based on optional values.
     *
     * @return string
     */
    private function generateTextMessage() :string
    {
        $msg = "You have been assigned for a new Beekeeper job! \n\n";

        if($this->contract->hasOptionalInfo) {
            $msg .= "Additional Information: \n";
        }

        if ($this->contract->contact_firstname) {
            $msg .= 'Firstname: ' . $this->contract->contact_firstname . "\n";
        }
        if ($this->contract->contact_lastname) {
            $msg .= 'Lastname: ' . $this->contract->contact_lastname . "\n";
        }
        if ($this->contract->contact_phone) {
            $msg .= 'Phone: ' . $this->contract->contact_phone . "\n";
        }
        if ($this->contract->info) {
            $msg .= 'Info: ' . $this->contract->info . "\n";
        }
    }
}
```

```

        }

        $msg .= "\nIf possible, please get in touch with the contact person in advance to discuss
details.\n\n";

        $msg .= GeoLocationController::generateGoogleMapsPin($this->contract->lat, $this->contract->lon);

        return $msg;
    }
}

```

app/Providers/AppServiceProvider.php

```

<?php

namespace App\Providers;

use Illuminate\Support\Facades\Blade;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Blade::if('admin', function() {
            return auth()->user() && auth()->user()->is_admin;
        });

        Blade::if('beekeeper', function() {
            return auth()->user() && !is_null(auth()->user()->beekeeper);
        });
    }
}

```

app/View/Components/Forms/Alert.php

```

<?php

namespace App\View\Components\Forms;

use Illuminate\View\Component;

class Alert extends Component
{
    /**
     * The alert title.
     *
     * @var string
     */
    public $title;

    /**
     * The alert message.
     *
     * @var string
     */
    public $message;

    /**
     * The alert type.
     *
     * @var string
     */
    public $type;

    /**
     * Create a new component instance.
     */
}

```

```

    * @param $title
    * @param $message
    * @param $type
    */
    public function construct($title, $message, $type)
    {
        $this->title = $title;
        $this->message = $message;
        $this->type = $type;
    }

    /**
     * Get the view / contents that represent the component.
     *
     * @return \Illuminate\Contracts\View\View|Closure|string
     */
    public function render()
    {
        return view('components.forms.alert');
    }
}

```

app/View/Components/Forms/Input.php

```

<?php

namespace App\View\Components\Forms;

use Illuminate\View\Component;

class Input extends Component
{
    /**
     * The input type.
     *
     * @var string
     */
    public $type;

    /**
     * The input name.
     *
     * @var string
     */
    public $name;

    /**
     * The input title.
     *
     * @var string
     */
    public $title;

    /**
     * The input value.
     *
     * @var string
     */
    public $value;

    /**
     * If the input is required.
     *
     * @var boolean
     */
    public $required;

    /**
     * The input placeholder.
     *
     * @var string
     */
    public $placeholder;

    /**
     * Create a new component instance.
     *
     * @param $type
     * @param $name
     * @param $title
     * @param $value
     * @param $required
     * @param $placeholder
     */
    public function construct($type, $name, $title, $value, $required = 'true', $placeholder = '')
    {
        $this->type = $type;
        $this->name = $name;
    }
}

```

```

        $this->title = $title;
        $this->value = $value;
        $this->required = $required == 'true';
        $this->placeholder = $placeholder == '' ? $title : $placeholder;
    }

    /**
     * Get the view / contents that represent the component.
     */
    public function render()
    {
        return view('components.forms.input');
    }
}

```

app/View/Components/Forms/Number.php

```

<?php

namespace App\View\Components\Forms;

use Illuminate\View\Component;

class Number extends Component
{

    /**
     * The input number name.
     *
     * @var string
     */
    public $name;

    /**
     * The input number title.
     *
     * @var string
     */
    public $title;

    /**
     * The input number value.
     *
     * @var float
     */
    public $value;

    /**
     * The input number min value.
     *
     * @var float
     */
    public $min;

    /**
     * The input number max value.
     *
     * @var float
     */
    public $max;

    /**
     * The input number increment steps.
     *
     * @var string
     */
    public $step;

    /**
     * The input number placeholder.
     *
     * @var float
     */
    public $placeholder;

    /**
     * If the input number is required.
     *
     * @var boolean
     */
    public $required;

    /**
     * Create a new component instance.
     *
     * @param string $name
     */
}

```

```

    * @param $title
    * @param $value
    * @param $min
    * @param $max
    * @param $step
    * @param $placeholder
    * @param $required
    */
    public function __construct($name, $title, $value, $min, $max, $step, $placeholder, $required = 'true')
    {
        $this->name      = $name;
        $this->title     = $title;
        $this->value      = $value;
        $this->min       = $min;
        $this->max       = $max;
        $this->step      = $step;
        $this->placeholder = $placeholder;
        $this->required   = $required == 'true';
    }

    /**
     * Get the view / contents that represent the component.
     *
     * @return \Illuminate\Contracts\View\View|Closure|string
     */
    public function render()
    {
        return view('components.forms.number');
    }
}

```

app/View/Components/Forms/Password.php

```

<?php

namespace App\View\Components\Forms;

use Illuminate\View\Component;

class Password extends Component
{
    /**
     * If the password needs to be confirmed.
     *
     * @var boolean
     */
    public $confirm;

    /**
     * Create a new component instance.
     *
     * @param $confirm
     */
    public function __construct($confirm)
    {
        $this->confirm = $confirm == 'true';
    }

    /**
     * Get the view / contents that represent the component.
     *
     * @return \Illuminate\Contracts\View\View|Closure|string
     */
    public function render()
    {
        return view('components.forms.password');
    }
}

```

app/View/Components/Forms/Textarea.php

```
<?php

namespace App\View\Components\Forms;

use Illuminate\View\Component;

class Textarea extends Component
{
    /**
     * The textarea name.
     *
     * @var string
     */
    public $name;

    /**
     * The textarea title.
     *
     * @var string
     */
    public $title;

    /**
     * The textarea value.
     *
     * @var string
     */
    public $value;

    /**
     * If the textarea is required.
     *
     * @var boolean
     */
    public $required;

    /**
     * Create a new component instance.
     *
     * @param $name
     * @param $title
     * @param $value
     * @param $required
     */
    public function __construct($name, $title, $value, $required = 'true')
    {
        $this->name      = $name;
        $this->title     = $title;
        $this->value     = $value;
        $this->required = $required == 'true';
    }

    /**
     * Get the view / contents that represent the component.
     *
     * @return \Illuminate\Contracts\View\View|Closure|string
     */
    public function render()
    {
        return view('components.forms.textarea');
    }
}
```

app/helpers.php

```
<?php

/**
 * Replace empty chars and - from string.
 *
 * @param $phone
 * @return string
 */
function replaceEmptyChars($phone): string
{
    $phone = str_replace(' ', '', $phone);
    $phone = str_replace('-', '', $phone);

    return $phone;
}

/**
 * Convert Phone number to Nominatim expected Number
 * with added countrycode 41.
 *
 * Input: 0765813596 | 0041765813596 | +41765813596
 * Output: 41765813596
 *
 * @param $phone
 * @return string
 */

function formatPhoneNum($phone): string
{
    $phone = replaceEmptyChars($phone);

    $phone = str_replace('+', ' ', $phone);

    switch (substr($phone, 0, 2)) {
        case "00" : $phone = substr($phone, 2); break;
        case "07" :
        case "04" : $phone = '41' . substr($phone, 1); break;
    }
    return $phone;
}

/**
 * Reverse Phone format from db:
 * 417658135 to 076 581 35 96.
 *
 * @param $phone
 * @return string
 */
function reverseFormatPhoneNum($phone): string
{
    $num[0] = substr($phone, 2, 2);
    $num[1] = substr($phone, 4, 3);
    $num[2] = substr($phone, 7, 2);
    $num[3] = substr($phone, 9, 2);

    return '0' . $num[0] . ' ' . $num[1] . ' ' . $num[2] . ' ' . $num[3];
}
```

database/factories/BeekeeperFactory.php

```
<?php

namespace Database\Factories;

use App\Models\Beekeeper;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

class BeekeeperFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = Beekeeper::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'firstname'      => $this->faker->firstName,
            'lastname'       => $this->faker->lastName,
            'phone'          => rand(4176000000, 4179999999),
            'phone_verified_at' => now(),
        ];
    }
}
```

database/factories/ContractFactory.php

```
<?php

namespace Database\Factories;

use App\Models\Beekeeper;
use App\Models\Contract;
use App\Models\Postcode;
use App\Models\User;
use Illuminate\Database\Eloquent\Factories\Factory;

class ContractFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = Contract::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'lon'           => rand(470000, 480000) / 10000,
            'lat'           => rand( 80000, 90000) / 10000,
            'contact_firstname' => $this->faker->firstName,
            'contact_lastname'  => $this->faker->lastName,
            'contact_phone'    => rand(4176000000, 4179999999),
            'info'           => $this->faker->realText(60),
            'postcode_id'     => Postcode::all()->random()->id,
            'created_by'      => User::all()->where('is_admin')->random()->id,
            'beekeeper_id'    => rand(0,3) == 1 ? null : Beekeeper::all()->random()->id,
        ];
    }
}
```

database/factories/UserFactory.php

```
<?php

namespace Database\Factories;

use App\Models\User;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

class UserFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = User::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'email'            => $this->faker->unique()->safeEmail(),
            'email verified at' => now(),
            'password'          => '$2y$10$92IXUNpkjO0rQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi',
            'password'          => Str::random(10),
            'remember_token'   => Str::random(10),
        ];
    }
}
```

database/migrations/2022\_03\_15\_140000\_create\_users\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('email')->unique();
            $table->timestamp('email verified at')->nullable();
            $table->string('password');
            $table->boolean('is_admin')->default(0);
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

database/migrations/2022\_03\_15\_145208\_create\_beekeepers\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateBeekeepersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('beekeepers', function (Blueprint $table) {
            $table->id();
            $table->string('firstname');
            $table->string('lastname');
            $table->string('phone')->unique();
            $table->timestamp('phone verified at')->nullable();
            $table->timestamps();
            $table->foreignId('user_id')->unique()->constrained();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('beekeepers');
    }
}
```

database/migrations/2022\_03\_15\_145648\_create\_postcodes\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostcodesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('postcodes', function (Blueprint $table) {
            $table->id();
            $table->smallInteger('postcode');
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('postcodes');
    }
}
```

database/migrations/2022\_03\_15\_145748\_create\_contracts\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateContractsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('contracts', function (Blueprint $table) {
            $table->id();
            $table->double('lon');
            $table->double('lat');
            $table->string('contact_firstname')->nullable();
            $table->string('contact_lastname')->nullable();
            $table->string('contact_phone')->nullable();
            $table->string('info')->nullable();
            $table->foreignId('beekeeper_id')->nullable()->constrained();
            $table->foreignId('postcode_id')->constrained();
            $table->foreignId('created_by')->constrained('users');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('contracts');
    }
}
```

database/migrations/2022\_03\_15\_150215\_create\_beekeeper\_contract\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateBeekeeperContractTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('beekeeper_contract', function (Blueprint $table) {
            $table->foreignId('beekeeper_id')->constrained();
            $table->foreignId('contract_id')->constrained();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('beekeeper_contract');
    }
}
```

database/migrations/2022\_03\_15\_150301\_create\_beekeeper\_postcode\_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateBeekeeperPostcodeTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('beekeeper_postcode', function (Blueprint $table) {
            $table->foreignId('beekeeper_id')->constrained();
            $table->foreignId('postcode_id')->constrained();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('beekeeper_postcode');
    }
}
```

database/seeders/BeekeeperContractSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\Beekeeper;
use App\Models\Contract;
use Illuminate\Database\Seeder;

class BeekeeperContractSeeder extends Seeder
{
    /**
     * Add random 1-5 applicable contracts to each beekeeper
     * Add all random assigned Contracts to also be applicable
     *
     * @return void
     */
    public function run()
    {
        foreach (Beekeeper::all() as $beekeeper) {
            $j = rand(1, 5);

            for($i = 0; $i < $j; $i++) {
                $beekeeper->contracts_applicable()->attach(Contract::all()->random());
            }

            $beekeeper->contracts_applicable()->attach($beekeeper->contracts);
        }
    }
}
```

database/seeders/BeekeeperPostcodeSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\Beekeeper;
use App\Models\Postcode;
use Illuminate\Database\Seeder;

class BeekeeperPostcodeSeeder extends Seeder
{
    /**
     * Add random 1-5 jurisdictions to each beekeeper
     *
     * @return void
     */
    public function run()
    {
        foreach (Beekeeper::all() as $beekeeper) {

            $j = rand(1, 5);

            for($i = 0; $i < $j; $i++) {
                $beekeeper->postcodes()->attach(Postcode::all()->random());
            }
        }
    }
}
```

database/seeders/BeekeeperSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\Beekeeper;
use App\Models\User;
use Illuminate\Database\Seeder;

class BeekeeperSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // create 20x random beekeepers
        User::factory()->count(20)->has(Beekeeper::factory())->create();
        // create 1x default beekeeper
        User::factory(['email' => 'jon@doe.ch'])->has(Beekeeper::factory(['firstname' => 'Jon', 'lastname' => 'Doe']))->create();
    }
}
```

database/seeders/ContractSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\Contract;
use Illuminate\Database\Seeder;

class ContractSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Contract::factory()->count(10)->create();
    }
}
```

database/seeders/DatabaseSeeder.php

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call([
            UserSeeder::class,
            BeekeeperSeeder::class,
            PostcodeSeeder::class,
            ContractSeeder::class,
            BeekeeperPostcodeSeeder::class,
            BeekeeperContractSeeder::class,
        ]);
    }
}
```

database/seeders/PostcodeSeeder.php

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PostcodeSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('postcodes')->insert([
            ["postcode"=>8001, "name"=>"Zürich"],
            ["postcode"=>8002, "name"=>"Zürich"],
            ["postcode"=>8003, "name"=>"Zürich"],
            ["postcode"=>8004, "name"=>"Zürich"],
            ["postcode"=>8005, "name"=>"Zürich"],
            ["postcode"=>8006, "name"=>"Zürich"],
            ["postcode"=>8008, "name"=>"Zürich"],
            ["postcode"=>8032, "name"=>"Zürich"],
            ["postcode"=>8037, "name"=>"Zürich"],
            ["postcode"=>8038, "name"=>"Zürich"],
            ["postcode"=>8041, "name"=>"Zürich"],
            ["postcode"=>8044, "name"=>"Dübendorf, Zürich"],
            ["postcode"=>8045, "name"=>"Zürich"],
            ["postcode"=>8046, "name"=>"Zürich"],
            ["postcode"=>8047, "name"=>"Zürich"],
            ["postcode"=>8048, "name"=>"Zürich"],
            ["postcode"=>8049, "name"=>"Zürich"],
            ["postcode"=>8050, "name"=>"Zürich"],
            ["postcode"=>8051, "name"=>"Zürich"],
            ["postcode"=>8052, "name"=>"Zürich"],
            ["postcode"=>8053, "name"=>"Zürich"],
            ["postcode"=>8055, "name"=>"Zürich"],
            ["postcode"=>8057, "name"=>"Zürich"],
            ["postcode"=>8064, "name"=>"Zürich"],
            ["postcode"=>8102, "name"=>"Oberengstringen"],
            ["postcode"=>8103, "name"=>"Unterengstringen"],
            ["postcode"=>8104, "name"=>"Weiningen (ZH)"],
            ["postcode"=>8105, "name"=>"Regensdorf"],
            ["postcode"=>8106, "name"=>"Regensdorf"],
            ["postcode"=>8107, "name"=>"Buchs (ZH)"],
            ["postcode"=>8108, "name"=>"Dällikon"],
            ["postcode"=>8112, "name"=>"Oetelfingen"],
            ["postcode"=>8113, "name"=>"Boppelsen"],
            ["postcode"=>8114, "name"=>"Dänikon"],
            ["postcode"=>8115, "name"=>"Hüttikon"],
            ["postcode"=>8117, "name"=>"Fällanden"],
            ["postcode"=>8118, "name"=>"Fällanden"],
            ["postcode"=>8121, "name"=>"Fällanden"],
            ["postcode"=>8122, "name"=>"Maur"],
            ["postcode"=>8123, "name"=>"Maur"],
            ["postcode"=>8124, "name"=>"Maur"],
            ["postcode"=>8125, "name"=>"Zollikon"],
            ["postcode"=>8126, "name"=>"Zumikon"],
```

```

["postcode"=>8127, "name"=>"Küschnacht (ZH)" ],
["postcode"=>8132, "name"=>"Egg"],
["postcode"=>8133, "name"=>"Egg"],
["postcode"=>8134, "name"=>"Adliswil"],
["postcode"=>8135, "name"=>"Langnau am Albis, Horgen"],
["postcode"=>8136, "name"=>"Thalwil"],
["postcode"=>8142, "name"=>"Uitikon"],
["postcode"=>8143, "name"=>"Stallikon"],
["postcode"=>8152, "name"=>"Opfikon"],
["postcode"=>8153, "name"=>"Rümlang"],
["postcode"=>8154, "name"=>"Oberglatt"],
["postcode"=>8155, "name"=>"Niederhasli"],
["postcode"=>8156, "name"=>"Niederhasli"],
["postcode"=>8157, "name"=>"Diebsdorf"],
["postcode"=>8158, "name"=>"Regensberg"],
["postcode"=>8162, "name"=>"Steinmaur"],
["postcode"=>8164, "name"=>"Bachs"],
["postcode"=>8165, "name"=>"Oberweningen, Schleinikon, Schöfflisdorf"],
["postcode"=>8166, "name"=>"Niederweningen"],
["postcode"=>8172, "name"=>"Niederglatt"],
["postcode"=>8173, "name"=>"Neerach"],
["postcode"=>8174, "name"=>"Stadel"],
["postcode"=>8175, "name"=>"Stadel"],
["postcode"=>8180, "name"=>"Bülach"],
["postcode"=>8181, "name"=>"Höri"],
["postcode"=>8182, "name"=>"Hochfelden"],
["postcode"=>8184, "name"=>"Bachenbülach"],
["postcode"=>8185, "name"=>"Winkel"],
["postcode"=>8187, "name"=>"Weizach"],
["postcode"=>8192, "name"=>"Glattfelden"],
["postcode"=>8193, "name"=>"Eglsau"],
["postcode"=>8194, "name"=>"Hünztwangen"],
["postcode"=>8195, "name"=>"Wasterkingen"],
["postcode"=>8196, "name"=>"Wil (ZH)"],
["postcode"=>8197, "name"=>"Rafz"],
["postcode"=>8245, "name"=>"Feuerthalen"],
["postcode"=>8246, "name"=>"Feuerthalen"],
["postcode"=>8247, "name"=>"Flurlingen"],
["postcode"=>8248, "name"=>"Laufen-Uhwiesen"],
["postcode"=>8302, "name"=>"Kloten"],
["postcode"=>8303, "name"=>"Bassersdorf"],
["postcode"=>8304, "name"=>"Wallisellen"],
["postcode"=>8305, "name"=>"Dietlikon"],
["postcode"=>8306, "name"=>"Wangen-Brüttisellen"],
["postcode"=>8307, "name"=>"Illnau-Effretikon"],
["postcode"=>8308, "name"=>"Illnau-Effretikon"],
["postcode"=>8309, "name"=>"Nürensdorf"],
["postcode"=>8310, "name"=>"Lindau"],
["postcode"=>8311, "name"=>"Brütten"],
["postcode"=>8312, "name"=>"Lindau"],
["postcode"=>8314, "name"=>"Illnau-Effretikon"],
["postcode"=>8315, "name"=>"Lindau"],
["postcode"=>8317, "name"=>"Lindau"],
["postcode"=>8320, "name"=>"Fehraltorf"],
["postcode"=>8322, "name"=>"Russikon"],
["postcode"=>8330, "name"=>"Pfäffikon"],
["postcode"=>8331, "name"=>"Pfäffikon"],
["postcode"=>8332, "name"=>"Russikon"],
["postcode"=>8335, "name"=>"Hittnau"],
["postcode"=>8340, "name"=>"Hinwil"],
["postcode"=>8342, "name"=>"Hinwil"],
["postcode"=>8344, "name"=>"Bäretswil"],
["postcode"=>8345, "name"=>"Bäretswil"],
["postcode"=>8352, "name"=>"Elsau, Winterthur"],
["postcode"=>8353, "name"=>"Elgg"],
["postcode"=>8354, "name"=>"Hofstetten (ZH)"],
["postcode"=>8400, "name"=>"Winterthur"],
["postcode"=>8404, "name"=>"Winterthur"],
["postcode"=>8405, "name"=>"Winterthur"],
["postcode"=>8406, "name"=>"Winterthur"],
["postcode"=>8408, "name"=>"Winterthur"],
["postcode"=>8409, "name"=>"Winterthur"],
["postcode"=>8412, "name"=>"Neftenbach"],
["postcode"=>8413, "name"=>"Neftenbach"],
["postcode"=>8414, "name"=>"Buch am Irchel"],
["postcode"=>8415, "name"=>"Berg am Irchel"],
["postcode"=>8416, "name"=>"Flaach"],
["postcode"=>8418, "name"=>"Schlatt (ZH)"],
["postcode"=>8421, "name"=>"Dättlikon"],
["postcode"=>8422, "name"=>"Pfungen"],
["postcode"=>8424, "name"=>"Embrach"],
["postcode"=>8425, "name"=>"Oberembrach"],
["postcode"=>8426, "name"=>"Lufingen"],
["postcode"=>8427, "name"=>"Freienstein-Teufen, Rorbas"],
["postcode"=>8428, "name"=>"Freienstein-Teufen"],
["postcode"=>8442, "name"=>"Hettlingen"],
["postcode"=>8444, "name"=>"Henggart"],
["postcode"=>8447, "name"=>"Dachsen"],
["postcode"=>8450, "name"=>"Andelfingen"],
["postcode"=>8451, "name"=>"Kleinandelfingen"]

```

```
[{"postcode"=>8452, "name"=>"Adlikon"],  
["postcode"=>8453, "name"=>"Kleinandelfingen"],  
["postcode"=>8457, "name"=>"Humlikon"],  
["postcode"=>8458, "name"=>"Dorf"],  
["postcode"=>8459, "name"=>"Volken"],  
["postcode"=>8460, "name"=>"Marthalen"],  
["postcode"=>8461, "name"=>"Kleinandelfingen"],  
["postcode"=>8462, "name"=>"Rheinau"],  
["postcode"=>8463, "name"=>"Benken (ZH)"],  
["postcode"=>8464, "name"=>"Marthalen"],  
["postcode"=>8465, "name"=>"Trüllikon"],  
["postcode"=>8466, "name"=>"Trüllikon"],  
["postcode"=>8467, "name"=>"Truttikon"],  
["postcode"=>8468, "name"=>"Waltalingen"],  
["postcode"=>8471, "name"=>"Dägerlen"],  
["postcode"=>8472, "name"=>"Seuzach"],  
["postcode"=>8474, "name"=>"Dinhard"],  
["postcode"=>8475, "name"=>"Ossingen"],  
["postcode"=>8476, "name"=>"Unterstammheim"],  
["postcode"=>8477, "name"=>"Oberstammheim"],  
["postcode"=>8478, "name"=>"Thalheim an der Thur"],  
["postcode"=>8479, "name"=>"Altikon"],  
["postcode"=>8482, "name"=>"Winterthur"],  
["postcode"=>8483, "name"=>"Zell (ZH)"],  
["postcode"=>8484, "name"=>"Weisslingen"],  
["postcode"=>8486, "name"=>"Zell (ZH)"],  
["postcode"=>8487, "name"=>"Zell (ZH)"],  
["postcode"=>8488, "name"=>"Turbenthal"],  
["postcode"=>8489, "name"=>"Wildberg"],  
["postcode"=>8492, "name"=>"Wila"],  
["postcode"=>8493, "name"=>"Bauma"],  
["postcode"=>8494, "name"=>"Bauma"],  
["postcode"=>8495, "name"=>"Turbenthal"],  
["postcode"=>8496, "name"=>"Fischenthal"],  
["postcode"=>8497, "name"=>"Fischenthal"],  
["postcode"=>8498, "name"=>"Fischenthal"],  
["postcode"=>8499, "name"=>"Bauma"],  
["postcode"=>8523, "name"=>"Hagenbuch"],  
["postcode"=>8542, "name"=>"Wiesendangen"],  
["postcode"=>8543, "name"=>"Wiesendangen"],  
["postcode"=>8544, "name"=>"Wiesendangen"],  
["postcode"=>8545, "name"=>"Rickenbach (ZH)"],  
["postcode"=>8548, "name"=>"Ellikon an der Thur"],  
["postcode"=>8600, "name"=>"Dübendorf"],  
["postcode"=>8602, "name"=>"Wangen-Brüttisellen"],  
["postcode"=>8603, "name"=>"Schwerzenbach"],  
["postcode"=>8604, "name"=>"Volketswil"],  
["postcode"=>8605, "name"=>"Volketswil"],  
["postcode"=>8606, "name"=>"Greifensee, Uster"],  
["postcode"=>8607, "name"=>"Seegräben"],  
["postcode"=>8608, "name"=>"Bubikon"],  
["postcode"=>8610, "name"=>"Uster"],  
["postcode"=>8614, "name"=>"Gossau (ZH), Uster"],  
["postcode"=>8615, "name"=>"Uster"],  
["postcode"=>8616, "name"=>"Uster"],  
["postcode"=>8617, "name"=>"Mönchaltorf"],  
["postcode"=>8618, "name"=>"Oetwil am See"],  
["postcode"=>8620, "name"=>"Wetzikon (ZH)"],  
["postcode"=>8623, "name"=>"Wetzikon (ZH)"],  
["postcode"=>8624, "name"=>"Gossau (ZH)"],  
["postcode"=>8625, "name"=>"Gossau (ZH)"],  
["postcode"=>8626, "name"=>"Gossau (ZH)"],  
["postcode"=>8627, "name"=>"Gründingen"],  
["postcode"=>8630, "name"=>"Rüti (ZH)"],  
["postcode"=>8632, "name"=>"Dürnten"],  
["postcode"=>8633, "name"=>"Bubikon"],  
["postcode"=>8634, "name"=>"Hombrichtikon"],  
["postcode"=>8635, "name"=>"Dürnten"],  
["postcode"=>8636, "name"=>"Wald (ZH)"],  
["postcode"=>8637, "name"=>"Wald (ZH)"],  
["postcode"=>8700, "name"=>"Küschnacht (ZH)"],  
["postcode"=>8702, "name"=>"Zollikon"],  
["postcode"=>8703, "name"=>"Erlenbach (ZH)"],  
["postcode"=>8704, "name"=>"Herrliberg"],  
["postcode"=>8706, "name"=>"Meilen"],  
["postcode"=>8707, "name"=>"Uetikon am See"],  
["postcode"=>8708, "name"=>"Männedorf"],  
["postcode"=>8712, "name"=>"Stäfa"],  
["postcode"=>8713, "name"=>"Stäfa"],  
["postcode"=>8714, "name"=>"Hombrichtikon"],  
["postcode"=>8800, "name"=>"Thalwil"],  
["postcode"=>8802, "name"=>"Kilchberg (ZH)"],  
["postcode"=>8803, "name"=>"Rüschlikon"],  
["postcode"=>8804, "name"=>"Wädenswil"],  
["postcode"=>8805, "name"=>"Richterswil"],  
["postcode"=>8810, "name"=>"Horgen"],  
["postcode"=>8815, "name"=>"Horgen"],  
["postcode"=>8816, "name"=>"Hirzel"],  
["postcode"=>8820, "name"=>"Wädenswil"],  
["postcode"=>8824, "name"=>"Schönenberg (ZH)"]]
```

```

        ["postcode"=>8825, "name"=>"Hütten"],
        ["postcode"=>8833, "name"=>"Richterswil"],
        ["postcode"=>8902, "name"=>"Urdorf"],
        ["postcode"=>8903, "name"=>"Birmensdorf (ZH)"],
        ["postcode"=>8904, "name"=>"Aesch (ZH)"],
        ["postcode"=>8906, "name"=>"Bonstetten"],
        ["postcode"=>8907, "name"=>"Wettswil am Albis"],
        ["postcode"=>8908, "name"=>"Hedingen"],
        ["postcode"=>8909, "name"=>"Affoltern am Albis"],
        ["postcode"=>8910, "name"=>"Affoltern am Albis"],
        ["postcode"=>8911, "name"=>"Rifferswil"],
        ["postcode"=>8912, "name"=>"Obfelden"],
        ["postcode"=>8913, "name"=>"Ottenbach"],
        ["postcode"=>8914, "name"=>"Aeugst am Albis"],
        ["postcode"=>8915, "name"=>"Hausen am Albis"],
        ["postcode"=>8925, "name"=>"Hausen am Albis"],
        ["postcode"=>8926, "name"=>"Kappel am Albis"],
        ["postcode"=>8932, "name"=>"Mettmenstetten"],
        ["postcode"=>8933, "name"=>"Maschwanden"],
        ["postcode"=>8934, "name"=>"Knonau"],
        ["postcode"=>8942, "name"=>"Oberrieden"],
        ["postcode"=>8951, "name"=>"Weiningen (ZH)"],
        ["postcode"=>8952, "name"=>"Schlieren"],
        ["postcode"=>8953, "name"=>"Dietikon"],
        ["postcode"=>8954, "name"=>"Geroldswil"],
        ["postcode"=>8955, "name"=>"Oetwil an der Limmat"]
    ]);
}
}

```

database/seeders/UserSeeder.php

```

<?php

namespace Database\Seeders;

use App\Models\User;
use Illuminate\Database\Seeder;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // create 3x random admin accounts
        User::factory(['is_admin' => 1])->count(3)->create();
        // create default admin account
        User::factory(['is_admin' => 1, 'email' => 'admin@admin.ch'])->create();
    }
}

```

public/css/main.css

```

body {
    margin: 0;
    padding: 0;
    background-color: #eee;
    -webkit-print-color-adjust: exact;
}

.footer {
    position: fixed;
    bottom: 0;
    width: 100%;
    padding-top: 5px;
    padding-bottom: 5px;
    color: white;
    overflow: hidden;
    z-index: 5;
}

.btn-block {
    display: block;
    width: 100%;
}

```

public/js/imkerSearch.js

```
/*
|-----
| Imker Search JS
|-----
|
| Handles all JS for the beekeeper and plz search.
|
*/

```

```
// Define output elements
const plzSearch      = document.getElementById('plzSearchOutput');
const beekeeperSearch = document.getElementById('beekeeperSearchOutput');
```

```
/***
 * Request Search of PLZ
 *
 * @param e
 */
function searchPlz (e) {
    $.ajax({
        type : "get",
        url  : searchPlzUrl,
        data : {search: e.value},
        success: function (data) {
            $('.oldSearch').remove();

            for(let postcode of data) {
                insertPLZ(postcode);
            }
        }
    });
}

/***
 * Append Postcode information in search result.
 *
 * @param postcode
 */
function insertPLZ(postcode) {
    let el = document.createElement('div');
    el.setAttribute('class', 'list-group-item list-group-item-action oldSearch');
    el.innerHTML = postcode.postcode + ' | ' + postcode.name;
    el.setAttribute('onclick', 'selectPostcode(this,'+ postcode.id+')');

    plzSearch.appendChild(el);
}

/***
 * Select PLZ from PLZ search result & search for beekeeper
 *
 * @param e
 * @param postcode_id
 */
function selectPostcode(e, postcode_id) {
    document.getElementById('searchInput').value = e.innerHTML;
    $('.oldSearch').remove();

    searchBeekeeper(postcode_id);
}

/***
 * Request Search of Beekeeper
 *
 * @param postcode_id
 */
function searchBeekeeper(postcode_id) {
    $.ajax({
        type : "get",
        url  : searchBeekeeperUrl,
        data : {postcode id: postcode id},
        success: function (data) {
            $('.oldSearch').remove();

            for(let beekeeper of data) {
                insertBeekeeper(beekeeper);
            }
        }
    });
}
```

```

        }
    });

}

/***
 * Append Beekeeper information in search result.
 *
 * @param beekeeper
 */
function insertBeekeeper(beekeeper) {
    let jurEl = document.createElement('p');
    jurEl.setAttribute('class', 'small');
    jurEl.innerHTML = beekeeper.jurisdictionsToString;

    let el = document.createElement('div');
    el.setAttribute('class', 'list-group-item list-group-item-action oldSearch');
    el.innerHTML = beekeeper.fullName + '<br>' + beekeeper.reverseFormattedPhone;
    el.appendChild(jurEl);

    beekeeperSearch.appendChild(el);
}

```

public/js/jurisdiction.js

```

/*
-----+
| Jurisdiction JS
|-----+
|
| Handles all JS for the beekeeper jurisdiction and plz search.
|
*/
// Define output element
const searchRes = document.getElementById('searchOutput');

/***
 * Add selected PLZ to Delete
 *
 * @param id
 * @param el
 */
function del(id, el) {
    const delJur = document.getElementById('delJur');

    let duplicate = false;

    // Eval is PLZ should be delete or revert de deletion
    document.getElementsByName('delJur[]').forEach(element => {
        if(element.value == id) {
            duplicate = true;
            element.remove();
        }
    });
    // revert if duplicate found else add to delete
    if(duplicate) {
        el.style.border = '1px solid rgba(0, 0, 0, 0.125)';
    } else {
        el.style.border = '1px solid red';
        let input = document.createElement('input');
        input.setAttribute('type', 'hidden');
        input.setAttribute('name', 'delJur[]');
        input.setAttribute('value', id);

        delJur.appendChild(input);
    }
}

/***
 * Add PLZ from Search result to form.
 *
 * @param el
 * @param id
 */
function add(el, id) {
    el.removeAttribute('onclick');
    el.classList.remove('oldSearch');
    el.style.border = '1px solid green';
}

```

```

el.innerHTML      = el.innerHTML + '<button type="button" class="btn btn-danger float-end"
onclick="reverseAdd('+id+', this.parentElement)">X</button>';
document.getElementById('currentJur').append(el);

let input = document.createElement('input');
input.setAttribute('type', 'hidden');
input.setAttribute('name', 'addJur[]');
input.setAttribute('value', id);

document.getElementById('addJur').append(input);
}

/**
 * Reverse a frontend Only added PLZ from form
 *
 * @param id
 * @param el
 */
function reverseAdd(id, el) {
    document.getElementsByName('addJur[]').forEach(element => {
        if(element.value == id) {
            element.remove();
        }
    });
    el.remove();
}

/**
 * Request Search of PLZ
 *
 * @param e
 */
function search (e) {

$.ajax({
    type : "get",
    url : searchURL,
    data : {search: e.value},
    success: function (data) {

        $('.oldSearch').remove();

        for(let postcode of data) {
            insert(postcode);
        }

    }
});
}

/**
 * Append Postcode information in search result.
 *
 * @param postcode
 */
function insert(postcode) {

let el = document.createElement('div');
el.setAttribute('class', 'list-group-item list-group-item-action oldSearch');
el.setAttribute('onclick', 'add(this,'+ postcode.id+')');
el.innerHTML = postcode.postcode + ' | ' + postcode.name;

searchRes.appendChild(el);
}

```

resources/views/auth/login.blade.php

```
<div class="row justify-content-center">
    <div class="col-md-12 col-lg-8">
        @error('email')
            <div class="alert alert-danger alert-dismissible fade show" role="alert">
                <strong>Error</strong> E-Mail Address or Password is wrong!
                <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
            </div>
        @enderror
        <div class="card">
            <div class="card-header text-center"><h4>Login</h4></div>
            <div class="card-body">
                <form method="POST" action="{{ route('login') }}">
                    @csrf

                    <div class="row mb-3">
                        <label for="email" class="col-form-label">E-Mail Address<span class="text-danger">*</span></label>

                        <div class="col">
                            <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old('email') }}" required autocomplete="email" autofocus>
                        </div>
                    </div>

                    <div class="row mb-3">
                        <label for="password" class="col-form-label">Password<span class="text-danger">*</span></label>

                        <div class="col">
                            <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" required autocomplete="current-password">
                                @error('password')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                        </div>
                    </div>

                    <button type="submit" class="btn btn-primary btn-block mb-2">Login</button>
                    <a href="{{ route('register') }}" class="btn btn-secondary btn-block mb-2">Register</a>
                    @if (Route::has('password.request'))
                        <a class="btn btn-link" href="{{ route('password.request') }}>
                            {{ ('Forgot Your Password?') }}
                        </a>
                    @endif
                </form>
            </div>
        </div>
    </div>
</div>
```

resources/views/auth/register.blade.php

```
@extends('layouts.app')

@section('content')


<div class="card col-md-6 offset-md-3">

        <div class="card-header text-center">
            <h4>Beekeeper-Register</h4>
        </div>
        <div class="card-body">
            <form method="POST" action="{{ route('register') }}">
                @csrf
                @method('POST')

                <x-forms.input type="text" name="firstname" title="Firstname" value="{{ old('firstname') }}"/>
                <x-forms.input type="text" name="lastname" title="Lastname" value="{{ old('lastname') }}"/>
                <x-forms.input type="text" name="phone" title="Phone" value="{{ old('phone') }}>
                    placeholder="076 581 35 96"
                <x-forms.input type="email" name="email" title="E-Mail" value="{{ old('email') }}"/>
                <x-forms.password confirm="true" />

                <div class="form-group mt-4">
                    <input class="@error('agb') is-invalid @enderror" type="checkbox" name="agb" id="agb">
                    <span>I hereby confirm that I have taken note that my contact information may be
                </div>
            </form>
        </div>
    </div>


```

```
disclosed for the beekeeper search<span class="text-danger">*</span></span>
    @error('agb')
        <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>

<input class="btn btn-block btn-primary mt-3" type="submit" value="Register" />

    </form>
</div>
</div>
</div>
@endsection
```

resources/views/components/forms/alert.blade.php

```
<div class="alert alert-{{ $type }} alert-dismissible fade show" role="alert">
    <strong>{{ $title }}</strong> {{ $message }}
    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
```

resources/views/components/forms/input.blade.php

```
<div class="form-group">
    <label class="form-label mt-4" for="{{ $name }}">{{ $title }} @if($required) <span class="text-danger">*</span> @endif</label>
    <input class="form-control @if($errors->any()) @error($name) is-invalid @else is-valid @enderror" type="{{ $type }}" name="{{ $name }}" id="{{ $name }}" placeholder="{{ $placeholder }}" value="{{ $value }}" @if($required) required @endif
        @error($name)
            <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
```

resources/views/components/forms/number.blade.php

```
<div class="form-group">
    <label class="form-label mt-4" for="{{ $name }}">{{ $title }} @if($required) <span class="text-danger">*</span> @endif</label>
    <input class="form-control @if($errors->any()) @error($name) is-invalid @else is-valid @enderror" type="number" name="{{ $name }}" id="{{ $name }}" placeholder="{{ $placeholder }}" value="{{ $value }}"
        min="{{ $min }}" max="{{ $max }}" step="{{ $step }}" @if($required) required @endif
        @error($name)
            <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
```

resources/views/components/forms/password.blade.php

```
<div class="form-group">
    <label class="form-label mt-4" for="password">Password<span class="text-danger">*</span></label>
    <input class="form-control @error('password') is-invalid @enderror" type="password" name="password" id="password" required
        @error('password')
            <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
    @if($confirm)
        <div class="form-group">
            <label class="form-label mt-4" for="password-confirm">Confirm Password<span class="text-danger">*</span></label>
            <input class="form-control @error('password_confirmation') is-invalid @enderror" type="password" name="password_confirmation" id="password-confirm" required
                @error('password_confirmation')
                    <div class="invalid-feedback">{{ $message }}</div>
                @enderror
            </div>
        @endif
    </div>
```

resources/views/components/forms/textarea.blade.php

```
<div class="form-group">
    <label class="form-label mt-4" for="{{ $name }}>{{ $title }} @if($required) <span class="text-danger">*</span> @endif</label>
    <textarea class="form-control @if($errors->any()) @error($name) is-invalid @else is-valid @enderror @endif" rows="3" name="{{ $name }}" id="{{ $name }}" placeholder="{{ $title }} @if($required) required @endif{{ $value }}</textarea>
    @error($name)
        <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>
```

resources/views/layouts/inc/footer.blade.php

```
<footer class="footer bg-primary d-flex justify-content-center">
    ©2022 IPA C.O'Connor
</footer>
```

resources/views/layouts/inc/navbar.blade.php

```
<nav class="navbar navbar-expand-md navbar-dark bg-primary sticky-top mb-3">
    <div class="container">
        <a class="navbar-brand" href="{{ route('index') }}>IPA</a>
        <button class="navbar-toggler collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="navbar-collapse collapse" id="navbarCollapse">
            <ul class="navbar-nav me-auto mb-2 mb-md-0">
                @admin
                    <li class="nav-item">
                        <a class="nav-link" href="{{ route('contract.create') }}>Create Contract</a>
                    </li>
                @endadmin
                @beekeeper
                    <li class="nav-item">
                        <a class="nav-link" href="{{ route('jurisdiction') }}>Jurisdictions</a>
                    </li>
                @endbeekeeper
            </ul>
            <ul class="navbar-nav ml-auto">
                @auth
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" data-bs-toggle="dropdown" aria-expanded="false">
                            <span class="me-2">{{ auth() -> user() -> email }}</span><i class="fa-solid fa-user-gear"></i><span class="caret"></span>
                        </a>
                        <ul class="dropdown-menu mt-md-2">
                            @beekeeper
                                <li class="nav-item">
                                    <a class="dropdown-item" href="{{ route('profile') }}>Profile</a>
                                </li>
                            @endbeekeeper
                            <li>
                                <form id="logout-form" action="{{ route('logout') }}" method="POST">
                                    @csrf
                                    <input type="submit" class="dropdown-item" value="{{ __('Logout') }}>
                                </form>
                            </li>
                        </ul>
                    </li>
                @endauth
            </ul>
        </div>
    </div>
</nav>
```

resources/views/layouts/app.blade.php

```
<!doctype html>
<html lang="{{ str_replace(' ', '-', app()>getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'BeeRes') }}</title>

    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v6.1.0/css/all.css">

    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('css/main.css') }}" rel="stylesheet">
    @yield('style')
</head>
<body>
    <div id="app">
        @include('layouts.inc.navbar')
        <div class="container-fluid mb-5">
            @yield('content')
        </div>
        @include('layouts.inc.footer')
    </div>
    <!-- Scripts -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="{{ asset('js/app.js') }}" defer></script>
</body>
</html>
```

resources/views/models/beekeeper/jurisdiction.blade.php

```
@extends('layouts.app')

@section('style')
    <style>
        @media (min-width: 768px) {
            .col-md-border:not(:last-child) {
                border-right: 2px solid #d7d7d7;
            }
        }
    </style>
@endsection

@section('content')
    <div class="row" style="min-height: 70vh;">
        <!-- LEFT SIDE -->
        <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
            <div class="h-100 w-100">
                <div class="card col-lg-8 offset-lg-2 mb-2 mb-md-0">
                    <div class="card-header text-center"><h4>Active PLZ</h4></div>
                    <div class="card-body overflow-auto" style="height: 21rem">
                        <form action="{{ route('jurisdiction.update') }}" method="POST"
id="jurisdictionForm">
                            @csrf
                            @method('POST')
                            <div class="d-none" id="delJur"></div>
                            <div class="d-none" id="addJur"></div>
                            <div class="list-group" id="currentJur">
                                @foreach($postcodes as $postcode)
                                    <div class="list-group-item list-group-item-action">
                                        {{ $postcode->postcode . ' | ' . $postcode->name }}
                                        <button type="button" class="btn btn-danger float-end"
onclick="del('{{ $postcode->id }}', this.parentElement)">X</button>
                                    </div>
                                @endforeach
                            </div>
                            <div class="card-footer">
                                <button form="jurisdictionForm" type="submit" class="btn btn-success btn-block">Speichern</button>
                            </div>
                        </div>
                    </div>
                <!-- RIGHT SIDE -->
                <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
                    <div class="h-100 w-100">
```

```

<div class="card col-lg-8 offset-lg-2 mb-2 mb-lg-0">
    <div class="card-header text-center"><h4>Search PLZ</h4></div>
    <div class="card-body" style="height: 21rem">
        <div class="input-group mb-1">
            <input type="text" class="form-control" id="searchInput" onkeyup="search(this)" placeholder="PLZ Search" autocomplete="off" aria-describedby="basic-addon">
            <div class="input-group-append">
                <span class="input-group-text h-100" id="basic-addon"><i class="fa-solid fa-3 fa-magnifying-glass"></i></span>
            </div>
        </div>
        <div class="list-group overflow-auto border" style="max-height: 16.5rem" id="searchOutput"></div>
    </div>
</div>
<script>const searchURL = '{{ route('search.plz') }}';</script>
<script src="{{ asset('js/jurisdiction.js') }}"></script>
@endsection

```

resources/views/models/beekeeper/profile.blade.php

```

@extends('layouts.app')

@section('content')
    <div class="card col-md-8 col-lg-4 offset-md-2 offset-lg-4">
        <div class="card-header text-center">
            <h4>Profile</h4>
        </div>
        <div class="card-body pt-0">
            <form method="POST" action="">
                @csrf
                @method('PUT')
                <x-forms.input type="text" name="firstname" title="Firstname" value="{{ $beekeeper->firstname }}"/>
                <x-forms.input type="text" name="lastname" title="Lastname" value="{{ $beekeeper->lastname }}"/>
                <x-forms.input type="text" name="phone" title="Phone" value="{{ $beekeeper->reverseFormattedPhone }}"/>
                <x-forms.input type="email" name="email" title="E-Mail" value="{{ $beekeeper->user->email }}"/>
                <x-forms.input type="password" name="old_pw" title="Old Password" value="" />
                <x-forms.password confirm="true" />
                <input class="btn btn-block btn-success mt-3" type="button" value="Update" />
                <button class="btn btn-danger btn-block my-2" type="button">Delete</button>
                <a href="{{ route('jurisdiction') }}" class="btn btn-secondary btn-block">PLZ - Jurisdiction</a>
            </form>
        </div>
    </div>
@endsection

```

resources/views/models/beekeeper/search.blade.php

```
<div class="card col-lg-8 offset-lg-2 mb-2 mb-md-0">
    <div class="card-header text-center"><h4>Imker Search</h4></div>
    <div class="card-body" style="height: 22.5rem">
        <div class="input-group mb-1">
            <input type="text" class="form-control" id="searchInput" onkeyup="searchPlz(this)" placeholder="PLZ Search" autocomplete="off" aria-describedby="basic-addon">
            <div class="input-group-append">
                <span class="input-group-text h-100" id="basic-addon">
                    <i class="fa-solid fa-3 fa-magnifying-glass"></i>
                </span>
            </div>
        </div>
        <div class="list-group overflow-auto border overflow-auto" style="max-height: 7.6rem" id="plzSearchOutput"></div>
        <div class="list-group overflow-auto border overflow-auto mt-1" style="max-height: 18rem" id="beekeeperSearchOutput"></div>
    </div>
</div>
<script>
    const searchPlzUrl      = '{{ route('guest.search.plz') }}';
    const searchBeekeeperUrl = '{{ route('guest.search.beekeeper') }}';
</script>
<script src="{{ asset('js/imkerSearch.js') }}"></script>
```

resources/views/models/contract/accept.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="card col-md-6 offset-md-3">
        <div class="card-header text-center">
            <h4>Contract available!</h4>
        </div>
        <div class="card-body">
            <form method="POST" action="{{ route('contract.accept', $contract->id) }}">
                @csrf
                @method('POST')
                <table class="table table-striped">
                    <thead>
                        <tr>
                            <th scope="col">Info</th>
                            <th scope="col">Description</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr>
                            <th scope="row">Location</th>
                            <td>{{ $contract->postcode->codename }}</td>
                        </tr>
                    </tbody>
                </table>
                <div class="fw-bold">
                    Once the contract has been accepted and handed over, it is binding and must be
                    fulfilled!
                </div>
                <div class="form-group">
                    <input class="@error('acc') is-invalid @enderror" type="checkbox" name="acc" id="acc">
                    <label class="form-label mt-4" for="acc">
                        I have read and agree to the terms and conditions.<span class="text-
                        danger">*</span>
                    </label>
                    @error('acc')
                    <div class="invalid-feedback">{{ $message }}</div>
                    @enderror
                </div>
                <input class="btn btn-block btn-success mt-3" type="submit" value="Accept" />
            </form>
        </div>
    </div>
@endsection
```

resources/views/models/contract/create.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="card col-md-8 col-lg-4 offset-md-2 offset-lg-4">
        <div class="card-header text-center">
            <h4>Contract Create</h4>
        </div>
        <div class="card-body">
            <form method="POST" action="{{ route('contract.store') }}">
                @csrf
                @method('POST')
                @error('plz-api')
                    <x-forms.alert title="Error" message="{{ $message }}" type="danger" />
                @enderror
                <x-forms.number name="lat" title="Latitude" min="47" max="48" placeholder="47.39036" value="{{ old('lat') }}" step=".00001" />
                <x-forms.number name="lon" title="Longitude" min="8" max="9" placeholder="8.49087" value="{{ old('lon') }}" step=".00001" />
                @error('plz-api')
                    <x-forms.number name="plz" title="Postcode" min="8000" max="9000" placeholder="8048" value="{{ old('plz') }}" step="1" required="false" />
                @enderror
                <hr />
                <h4 class="text-center">Contact Information</h4>
                <x-forms.input type="text" name="contact_firstname" title="Firstname" value="{{ old('contact_firstname') }}" required="false" />
                <x-forms.input type="text" name="contact_lastname" title="Lastname" value="{{ old('contact_lastname') }}" required="false" />
                <x-forms.input type="text" name="contact_phone" title="Phone" value="{{ old('contact_phone') }}" required="false" placeholder="076 581 35 96" />
                <x-forms.textarea name="info" title="Additional Information" value="{{ old('info') }}" required="false" />
                <input class="btn btn-block btn-primary mt-3" type="submit" value="Create Contract" />
            </form>
        </div>
    </div>
</div>
@endsection
```

resources/views/models/contract/show.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="card col-md-6 offset-md-3">
        <div class="card-header text-center">
            <h4>Contract Overview #{{ $contract->id }}</h4>
        </div>
        <div class="card-body">
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th scope="col">Description</th>
                        <th scope="col">Value</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <th scope="row">Latitude</th>
                        <td>{{ $contract->lat }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Longitude</th>
                        <td>{{ $contract->lon }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Location</th>
                        <td>{{ $contract->postcode->codename }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Firstname</th>
                        <td>{{ $contract->contact_firstname }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Lastname</th>
                        <td>{{ $contract->contact_lastname }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Phone</th>
                        <td>{{ $contract->contact_phone }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Additional Information</th>
                        <td>{{ $contract->info }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Created by</th>
                        <td>{{ $contract->created_by_user->email }}</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
@endsection
```

resources/views/models/contract/success.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="card col-md-6 offset-md-3">
        <div class="card-header text-center">
            <h4>Contract successfully taken</h4>
        </div>
        <div class="card-body">
            <div><b>Congratulations!</b> You have received the Contract.<br>All informations will follow shortly. Please check your SMS-Inbox.</div>
            <table class="table table-striped mt-2">
                <thead>
                    <tr>
                        <th scope="col">Description</th>
                        <th scope="col">Value</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <th scope="row">Latitude</th>
                        <td>{{ $contract->lat }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Longitude</th>
                        <td>{{ $contract->lon }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Location</th>
                        <td>{{ $contract->postcode->codename }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Firstname</th>
                        <td>{{ $contract->contact_firstname }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Lastname</th>
                        <td>{{ $contract->contact_lastname }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Phone</th>
                        <td>{{ $contract->contact_phone }}</td>
                    </tr>
                    <tr>
                        <th scope="row">Additional Information</th>
                        <td>{{ $contract->info }}</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
@endsection
```

resources/views/models/contract/taken.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="card col-md-6 offset-md-3">
        <div class="card-header text-center">
            <h4>Contract not available...</h4>
        </div>
        <div class="card-body text-center">
            <div>We are very Sorry! This Contract has already been accepted by another Beekeeper.</div>
        </div>
    </div>
@endsection
```

resources/views/index.blade.php

```
@extends('layouts.app')

@section('style')
    @auth @else
        <style>
            @media (min-width: 768px) {
                .col-md-border:not(:last-child) {
                    border-right: 2px solid #d7d7d7;
                }
            }
        </style>
    @endauth
@endsection

@section('content')
    @auth
        <div class="col-md-8 col-md-border m-auto">
            @include('models.beekeeper.search')
        </div>
    @else
        <div class="row" style="min-height: 70vh;">
            <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
                <div class="h-100 w-100">
                    @include('models.beekeeper.search')
                </div>
            </div>
            <div class="col-md-6 col-md-border d-flex justify-content-center m-auto">
                <div class="w-100 w-100">
                    @include('auth.login')
                </div>
            </div>
        </div>
    @endauth
@endsection
```

## routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () { return view('index'); })->name('index');

Auth::routes();

Route::get('guest/search/plz', [App\Http\Controllers\SearchController::class, 'guestSearchPLZ'])
    ->name('guest.search.plz');
Route::get('guest/search/beekeeper', [App\Http\Controllers\SearchController::class,
    'guestSearchBeekeeper'])->name('guest.search.beekeeper');

Route::group(['middleware' => 'auth'], function() {

    Route::group(['middleware' => 'beekeeperOnly'], function() {

        Route::get ('profile', [App\Http\Controllers\BeekeeperController::class, 'profile'])
            ->name('profile');
        Route::get ('jurisdiction', [App\Http\Controllers\BeekeeperController::class,
            'jurisdiction'])->name('jurisdiction');
        Route::post('jurisdiction/update', [App\Http\Controllers\BeekeeperController::class,
            'updateJurisdiction'])->name('jurisdiction.update');
        Route::get ('search/plz', [App\Http\Controllers\SearchController::class, 'searchPLZ'])
            ->name('search.plz');
        Route::get ('contract/{id}/accept', [App\Http\Controllers\ContractController::class, 'accept'])
            ->name('contract.accept');
        Route::get ('contract/taken', [App\Http\Controllers\ContractController::class, 'taken'])
            ->name('contract.taken');
        Route::get ('contract/{id}/success', [App\Http\Controllers\ContractController::class, 'success'])
            ->name('contract.accept.success');
        Route::post('contract/{id}/accept', [App\Http\Controllers\ContractController::class,
            'acceptContract'])->name('contract.accept');

    });

    Route::group(['middleware' => 'adminOnly'], function() {

        Route::get ('contract/create', [App\Http\Controllers\ContractController::class, 'create'])
            ->name('contract.create');
        Route::post('contract/store', [App\Http\Controllers\ContractController::class, 'store'])
            ->name('contract.store');
        Route::get ('contract/{id}/show', [App\Http\Controllers\ContractController::class, 'show'])
            ->name('contract.show');

    });

});

});
```