MEETING-FINDER APP

APP ENTWICKLUNG MIT PYTHON, KIVY & DJANGO

VINCENZO VELLONE

Abgabe: 27.10.2021

Klasse: Z-TIN-18-T-a

Diplomcoach: Stephan Kessler

Schweizerische Fachschule Teko

Dipl. Techniker HF Informatik Systemtechnik

Inhaltsverzeichnis

1 Management Summary

Durch die entwickelte Applikation und Website soll effizient und vereinfacht die Suche nach passenden Daten für Meetings ermöglicht werden.

1.1 Problemstellung

Das Team Service-Desk & Monitoring Center der Firma SwissTXT arbeitet im Schichtbetrieb. Dadurch, dass alle Mitarbeiter disponiert werden, ist das Teammeeting maximal vom halben Team besucht. Dies erschwert eine saubere und klare Kommunikation. Die Mitarbeiter, die durch die Schichtplanung nicht am Teammeeting teilnehmen können, müssen entsprechende Informationen aus dem Protokoll entnehmen, welches in der Regel nicht den ganzen Kontext wiedergibt.

1.2 Lösungsansatz

Mit dem Meeting-Finder Tool soll der ideale Termin gefunden werden, welches die Teilnahme einer grösseren Anzahl Team-Mitgliedern an den Meetings ermöglicht.
Das Tool richtet via App und Web eine Umfrage ein, welche zeigen wird, an welchem Termin die meisten Mitarbeiter teilnehmen können. Durch die erhöhte Teilnahme an den Meetings soll die interne Kommunikation verbessert werden.

1.3 Ziele und Rahmenbedingungen

Der Meeting-Organisator kann mithilfe der App oder Web einfach und schnell mehrere Termine auswählen. Die App erstellt mit den gesammelten Daten eine Umfrage. Die eingeladenen Personen können anschliessend via Link an der Umfrage teilnehmen und die Termine auswählen, an denen sie sich beteiligen können. Anhand dieser Angaben kann der Organisator eine Aussage dazu machen, an welchem Datum die meisten Mitarbeiter teilnehmen können.

- Der Benutzer kann Datumsvorschläge auswählen
- Meeting-Name und Dauer können bestimmt werden
- Ein Link zur Umfrage wird generiert
- Multiple-Choice Auswahl
- Resultat zeigt wie viele Mitarbeiter wann teilnehmen können

Optional:

- Benutzer Login
- App ist im PlayStore verfügbar
- Kalender einbinden (Google Kalender etc.)

1.4 Prognose

Durch das Meeting-Finder Tool werden mehr Mitarbeiter an den Meetings teilnehmen können. Die Kommunikation von aktuellen Ereignissen wird die Gesamtkoordination und Sicherstellung der Service und Support Prozesse verbessern.

Das Tool kann auch in anderen Teams implementiert werden. Dadurch können lange E-Mail-Konversationen, um einen passenden Termin zu finden, verhindert werden.

2 Beruflicher Lebenslauf

BERUFLICHE ERFAHRUNG

Seit 05/2020	SwissTXT (SRG SSR) Service Operator ICT und Broadcast (Zentrales Monitoring, Gesamtkoordination und Sicherstellung der Service Support Prozesse)
02/2019 – 02/2020	PricewaterhouseCoopers via iET SA Field Support – Umbau Office (Hardware Installation, Netzwerk Verkabelung und prüfen, Switch Installation & Patchen, Level 1 Support, IT-Planung & Einrichtung von neuen Büros)
11/2018 - 01/2019	Neue Personaldienst Zürich | Steiner+Fäh Elektroinstallateur EFZ - Ausbau der Lüftungszentrale, VBZ Tunnel
05/2018 - 08/2018	Neue Personaldienst Zürich | Elektro Egger Elektroinstallateur EFZ - Umbau Fitnesscenter
07/2017 - 09/2017	Neue Personaldienst Zürich Elektroinstallateur EFZ - Bereich Neu/Umbauten
07/2015 – 12/2015	Neue Personaldienst Zürich | Elektro Burger AG Elektroinstallateur EFZ - Multimedia und UKV Installationen, SWICA, Lombard Odier
08/2014 - 12/2014	Neue Personaldienst Zürich | Steiner+Fäh Elektroinstallateur EFZ - Bauleitung Umbau/Renovation Wohnblock

AUSBILDUNG

Seit 10/2019	HF Informatik Systemtechnik
08/2016 – 06/2017	Berufsmaturität Technik Uster
08/2010 – 08/2014	Ausbildung zum Elektroinstallateur EFZ Elektro Baer Zürich

3 Qualifikationsprofil

Entscheidungen fällen Prozess 2	Mit der Berücksichtigung auf den Prozessablauf ist die Entscheidung getroffen worden eine neue “Skype for Business Reaktionsgruppe” zu integrieren, die das Backoffice von dem Operator trennt.
Projekt planen und leiten Prozess 3	Die Umstellung des Telefonsystems von Agent32 zu Skype for Business wurde erfolgreich geplant und umgesetzt. Die Planung erfolgte mit mehreren Unternehmenseinheiten. Um einen unterbruchfreien Übergang zu gewährleisten, waren beide Systeme während zwei Wochen parallel in Betrieb.
Sich sprachlich verständigen Prozess 4	Nationale Mitarbeit mit den verschiedenen Unternehmenseinheiten. Kommunikation in mehreren Ländersprachen oder English.
Wirkungsvoll präsentieren und kommunizieren Prozess 5	Leitung von wöchentlichen Pikett-Meetings mit allen Pikett-Organisationen. Wichtigste Ereignisse zusammengefasst und präsentiert.
Unternehmensprozesse verstehen und mitgestalten Prozess 6	Anpassung des Change-Prozesses für Unterhaltungsarbeiten von externem Partner. Die Bezeichnungen des externen Partners unterschieden sich von der internen Bezeichnung. Durch ein Tool werden nun die Bezeichnungen „übersetzt“.
Geschäftsziele erreichen Prozess 7	Geschäfts und Team-Ziel im Service Desk und Monitoring Center erreicht. Verfügbarkeit der Haupt-Services muss mindestens 99.99% betragen.

Probleme analysieren und lösen Prozess 9	Zeitaufwendiger Tabellen-Abgleich-Task mit der Hilfe von Scripts automatisiert.
Sich persönlich weiter entwickeln Prozess 10	Selbstständig Python gelernt, mit der Hilfe von Büchern und Tutorials, um monotone Arbeitsschritte zu automatisieren.
Systems Engineering Prozess 11	Erstellung eines neuen Share-Order-Konzeptes, das in Microsoft Teams integriert ist.
System Projektierung Prozess 12	Ablaufs-Struktur für Teamanruf-Gruppe konzipiert und umgesetzt.
IT-Qualität sichern Prozess 13	Immer wiederkehrende Vorfälle erkennen und via Problem-Ticket analysieren lassen. Dies entlastet den User und den Support -Service.
Systeme aufbauen und in Betrieb setzen Prozess 14	Tool erstellt, welches die Kommunikation durch die einheitliche Bezeichnung mit externen Technikern erleichterte.
Systeme warten und erneuern Prozess 15	Gesamtkoordination von nationalen Wartungsarbeiten der Netzwerk-Infrastruktur.
Systeme vernetzen Prozess 16	Alte Jump-Host Clients in ein neues virtuelles Environment migriert.

4 Tools

4.1 Einführung

In diesem Abschnitt werden die Tools vorgestellt, die für die Erstellung des Projektes genutzt wurden.

4.1.1 Libraries

Um die Applikation zu programmieren, werden die zwei Python Libraries Kivy und KivyMD verwendet. Libraries, auch genannt Programmbibliotheken, sind eine Sammlung von Funktionen. Diese sind Code Blöcke, die durchgeführt werden, sobald man sie abruft.
Ein simples Beispiel: Eine Funktion wird benötigt, die jeden User begrüsst. Folglich würde man eine «begruessung» Funktion erstellen. Funktionen definiert man mit «def».

```
def begruessung(name):
  print("Hallo " + name)
```

Nun kann die Funktion aufgerufen werden. Der «name» ist in diesem Fall die Variable.

```
begruessung("Vincenzo")
begruessung("Matthias")
```

Output:

```
Hallo Vincenzo
Hallo Matthias
```

Mithilfe der Funktionen kann viel Zeit gespart werden. Die beiden Libraries Kivy und KivyMD sind eine Sammlung von Funktionen, die man zur App Entwicklung nutzt.

4.1.2 Framework

Für die Webseite wird das Framework Django verwendet. Frameworks haben Ähnlichkeiten zu Libraries, jedoch bieten Frameworks ein Basis-Gerüst. Mit der Installation eines Frameworks werden automatisch die benötigten Funktionen hinzugefügt, um das Grundgerüst zu erstellen. So kann man nach der Installation von Django schon direkt die Webseite starten. Auf der Startseite der Webseite steht dann, dass Django erfolgreich installiert wurde.

4.1.3 Datenbank

Als Datenbank wird Sqlite3 verwendet. Diese Datenbank ist auch Teil des Django Frameworks. In der Datenbank werden die erstellten Meeting-Umfragen gespeichert.

4.2 Python

4.2.1 Vorteile

Python hat den Anspruch gut lesbar zu sein und fördert einen schlichten Programmierstil. Dank der grossen Anzahl an Libraries, die in Python zur Verfügung stehen, ist es in kürzester Zeit möglich ein Skript oder ein kleines Programm zu erstellen.
Im Vergleich zu einigen anderen gängigen Sprachen ist Python zugänglicher. Ein Grund dafür ist der einfache Syntax. So werden zusammengehörige Blöcke durch eine Einrückung gekennzeichnet.
Die Befehle sind auch für Laien verständlicher und sind oft intuitiver.
Beispiel: Dem User soll es möglich gemacht werden seinen Username zu wählen und soll anschliessend begrüsst werde. Der Python code würde so aussehen:

```
name = input("Username: ")
print(f"Welcome {name}")
```

Output:

```
Username: Vincenzo #Eingabe des Users
Welcome Vincenzo
```

Im «name» wird die Eingabe des Users gespeichert. Der Text, der angezeigt werden soll, kann direkt im «input» Befehl hinzugefügt werden. Die Funktion «print» druckt in der Konsole Texte und Variablen aus.
Im Vergleich dazu dieselbe Funktion in Java:

```
import java.util.Scanner; // import the Scanner class
class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    String userName;

    System.out.println("Username: ");
    userName = myObj.nextLine();

    System.out.println("Welcome " + userName);
  }}
```

Diese Übersicht macht den Einstieg in Python einfacher. Wer bereits eine andere Programmiersprache beherrscht, findet sich schnell zurecht.
Python gibt es schon seit den Neunziger und gewann in den letzten Jahren an Bedeutung, dank den Bereichen Data-Science und Machine-Learning, in denen Python sehr gefragt ist.

4.2.2 Nachteile

Python bringt nicht nur Vorteile mit sich. Die Ausführungs-geschwindigkeit ist bei Python im Vergleich zu anderen gängigen Programmiersprachen eher niedrig. Dies hat mehrere Ursachen. Eine davon ist die Art wie die Sprachen kompiliert werden.

In diesem Test wurden Matrix-Multiplikationen gemacht. Über acht Milliarden Rechnungen machte man mit jeder Programmiersprache.

Sprache	C++	Java	Python
Performance	Sehr hoch	Hoch	Mittel
Geschwindigkeit	56.47	33.34	1
Compiling	Compiled zu Maschienen Code	Complied zu byte Code	Interpretiert den Code

Das Resultat zeigt, dass C++ im Vergleich zu Python 56-mal schneller ist. Trotz diesem enormen Geschwindigkeitsunterschied überwiegen die Vorteile von Python dieses Defizit. Heutzutage haben die meisten Geräte so viel Rechenleistung, dass es keine Rolle spielt, ob das Programm nun 0.001 Sekunden oder 0.056 Sekunden lädt.
C++ wird aber heute noch in vielen Bereichen angewendet wo genau dieser Performance-Vorteil gebraucht wird, wie beispielsweise in der Game-Entwicklung. Computerspiele brauchen ohnehin schon viel Rechenleistung. Um die Computer Anforderungen tief zu halten ist daher C++ ideal. Ebenfalls in IoT Geräten, in denen meistens Hardware mit schwacher Leistung verbaut ist, wird oft C++ verwendet. (Brihadiswaran 2020)

4.3 Kivy

Kivy ist eine Open Source Library für Python zum Erstellen von Multi-Platform Applikationen. Kivy stellt dem Programmierer ein Gerüst mit den üblichen Elementen zur Verfügung, die es für die Entwicklung einer App benötigt. So muss man sich zum Beispiel nicht mit der Touchscreen-Bedienung auseinandersetzen, da diese in Kivy bereits beinhaltet ist. Dies erleichtert die Erstellung von Applikationen.

4.3.1 Dokumentation

Zu jeder Library gehört auch eine Dokumentation. Darin ist beschrieben, wie man die Library installiert, nutzt und welche Funktionen zur Verfügung stehen.
Kivy basiert zwar auf Python, benutzt aber für das Design der Applikation die eigene Sprache KV-Language. Diese verhilft zu einer besseren Übersicht, die mittels Trennung von Design und der Programm Logik erreicht wird. Diese Trennung wird auch bei der Web-Entwicklung verwendet.

4.3.2 Hello World

Bei den meisten Programmiersprachen ist es üblich mit einem «Hello World» zu beginnen. Hierzu eine Veranschaulichung zu Kivy:

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text='Hello World')

TestApp().run()
```

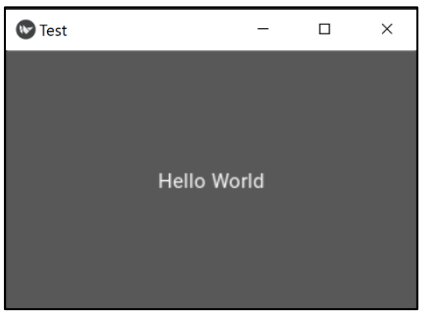


Abb. 1: Hello World in einer Kivy Applikation

Hier wird eine App erstellt. Die App besteht nur aus einem Knopf, mit der Aufschrift «Hello World». Dieser Knopf hat jedoch noch keine Funktion.

4.4 KivyMD

KivyMD ist eine Library, die viele Design Elemente für Kivy beinhaltet. Dies ermöglicht es den Entwicklern schnell eine App mit einem modernen Design auf die Beine zu stellen.

4.4.1 Kivy und KivyMD

Die Kivy Standard Designs für Knöpfe, Textfelder etc. erinnern eher an einer 90er Jahre Webseite als an eine moderne Mobile Applikation. Sie können zwar beliebig angepasst werden, jedoch benötigt dies sehr viel Code.
Alles was man mit KivyMD erstellen kann, ist auch mit Kivy möglich. Dies wäre aber mit einem grösseren Aufwand verbunden.

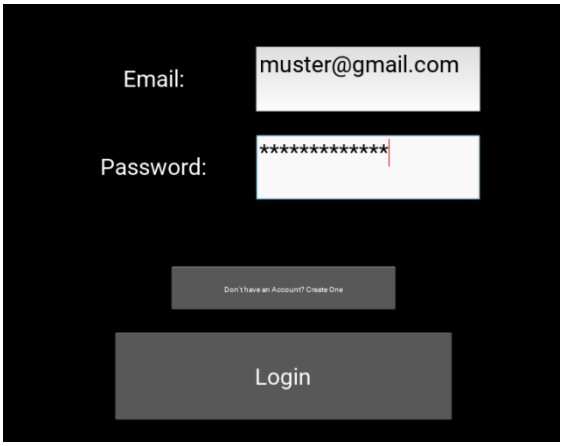


Abb. 2: Kivy Test App mit Login-Fenster

4.5 Django

4.5.1 Was ist Django

Django ist ein Tool mit diversen Funktionen, über das eine eigenständige Arbeit geschrieben werden könnte. Grosse Webseiten wie Instagram, Spotify und YouTube sind mit Django erstellt worden. Dieses auf Python basierende Framework ermöglicht es effizient eine Webseite zu erstellen.
Auf der Webseite von Django ist folgender Satz anzutreffen: «focus on writing your app without needing to reinvent the wheel». Diese Aussage bringt es exakt auf dem Punkt, da das Django Framework viele für eine Webseite relevante Elemente bereits enthält.

5 Meeting-Finder

5.1 Aufgabenstellung

In der Abteilung Service-Desk & Monitoring Center der Firma SwissTXT sind oft nur wenige Mitarbeiter am Team-Meeting anwesend. Dies ist hauptsächlich auf den sich wöchentlich ändernden Schichtbetrieb

zurückzuführen. Mithilfe der Applikation soll die Teilnehmerzahl der Team-Meetings erhöht werden.
Der Team-Meeting-Organisator kann mithilfe der Applikation dem ganzen Team eine Umfrage senden und somit feststellen, welches Datum und welche Uhrzeit für die grösste Anzahl der Teilnehmenden am geeignetsten ist.

5.2 Erfolgskriterien

Mithilfe der Webseite und der Applikation, kann die Anzahl der Teilnehmer in Zukunft erhöht werden. Um dies zu erreichen, müssen die Tools folgende Funktionen beinhalten.

- Zur Auswahl stehende Datumsvorschläge
- Meeting-Name und Dauer des Meetings können bestimmt werden
- Umfrage ist via Link abrufbar
- Multiple-Choice Auswahl für die Umfrage
- Übersicht über die Umfrage-Resultate

5.3 Projektrisiken

Durch die geringe Erfahrung in der App- und Web-Entwicklung ist eine Abschätzung des Zeitaufwandes schwierig. Diese Umstände erschweren eine detaillierte Risikoanalyse. Folgende Punkte erläutern die wichtigsten Risiken:

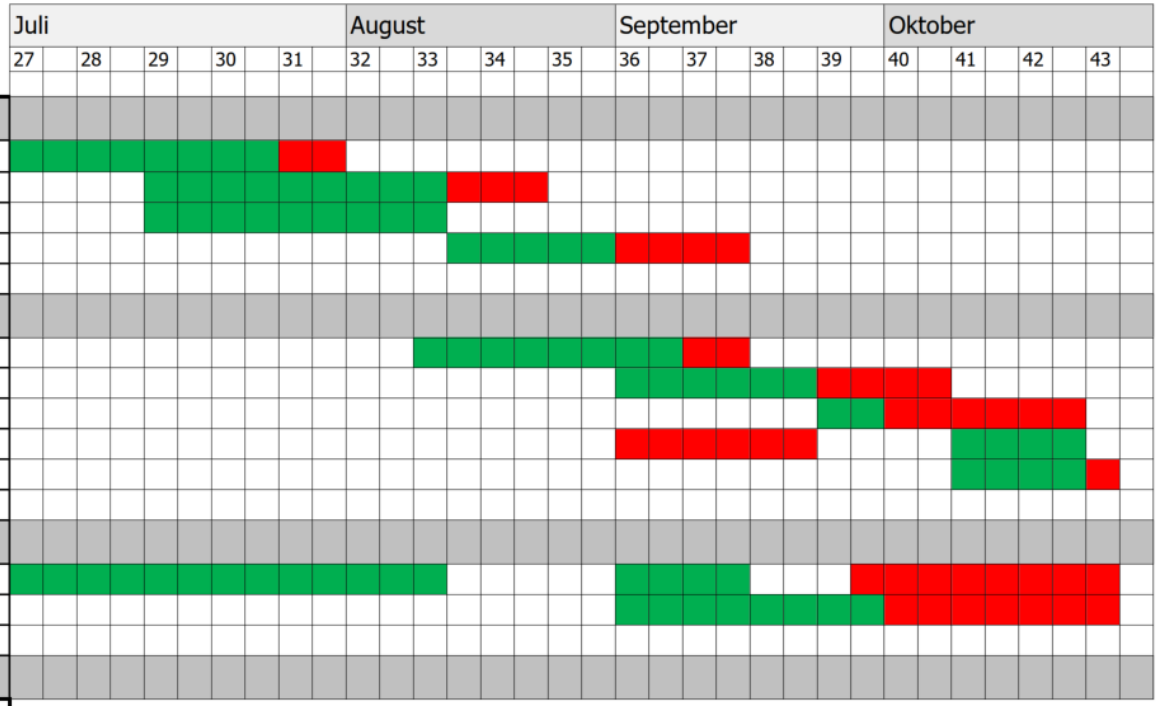
- Erstes grosses Python-Projekt
- Aufwand Abschätzung
- Unerfahrenheit
- Kommunikation zwischen App und Web

5.4 Terminplan

Projektplan - Meeting-Finder App

Projekttitel Meeting-Finder App
Projektstart 11.08.2021
Projektende 27.10.2021

	SOLL-Aufwand in Stunden	IST-Aufwand
Vorstudie	55	80
Python selbststudium (zählt nicht zur Arbeitszeit)	30	40
Einarbeitung Kivy	25	35
App Ideen sammeln	15	5
Einarbeitung Django	15	40
Projektstart	85	130
App erstellen	50	35
Webseite erstellen	25	35
API	5	20
Optimierungen	5	40
Design		
Dokumentation	25	35
Recherche	10	15
Dokumentation	15	20
Gesamtstunden	**165**	**245**



5.5 Erster Entwurf

Nach dem Start der Applikation folgt die Eingabe des Usernamen. Man gelangt zur Hauptseite, wo der Meeting-Name und die Dauer des Meetings bestimmt werden. Auf der dritten Seite wählt man die verschiedenen Daten und Zeiten, die man als Organisator zur Auswahl stellen will.
Alle Daten werden auf die Datenbank gesendet, wo auf der Webseite eine Umfrage generiert wird. Der Link zur Umfrage wird dem User auf der letzten Seite angezeigt.
Die Umfrage kann einfach via Multiple-Choice beantwortet werden. Nach der Umfrage werden die aktuellen Resultate angezeigt.
Die Applikation und Webseite werden jeweils in English sein, um diese für eine grössere Anzahl Personen zugänglich zu machen.

Auf dem nächsten Flussdiagramm wird der erste Ablauf-Entwurf aufgezeigt. Die Applikation besteht aus 4 Seiten.

- Startseite «MenuScreen»: UserName Eingabe
- Hauptseite «MainScreen»: Eingabe Meeting-Name und Dauer
- Erstellungsseite «CreateScreen»: Auswahl der Daten und Zeiten
- Linkseite «LinkScreen»: Anzeige des Links zur Umfrage

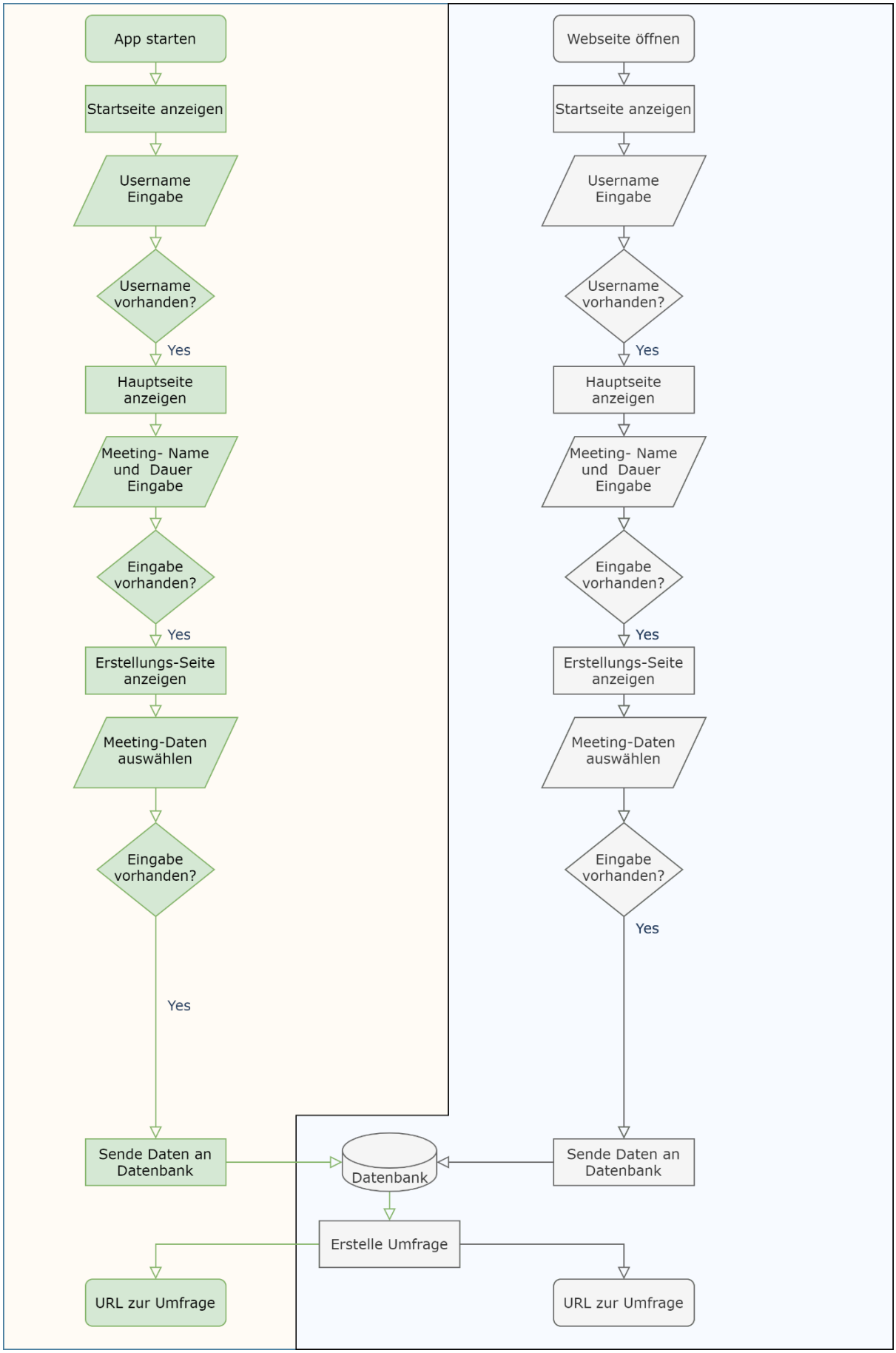


Abb. 3: Erster Ablauf-Entwurf von der App und Webseite

5.6 Kivy Applikation

5.6.1 Erste Schritte mit Kivy

Um den Umgang mit Kivy kennenzulernen, gibt es in der Dokumentation zwei Tutorials. Ersteres ist ein Pong-Spiel und das zweite ist ein Paint ähnliches Programm.
Die Tutorials zeigen, wie man einige Elemente von Kivy nutzen kann. Leider zeigt keiner der beiden die Funktion «Screen Manager» an, welche es ermöglichen würde zwischen mehreren Seiten der Applikation zu navigieren.

Das Pong-Spiel zeigt bereits einige Stärken von Kivy auf. So war es mit wenig Aufwand möglich einen Multi-Touch-Funktion zu integrieren, der die Bewegungen auf der linken und auf der rechten Bildschirmhälfte erkennt, wodurch zu zweit Pong gespielt werden kann.
Eine weitere Eigenschaft von Kivy ist die Anpassung an die verschiedenen Fenstergrössen. Dies ermöglicht der Applikation sich am jeweiligen Display des Smartphones oder Tablets anzupassen.

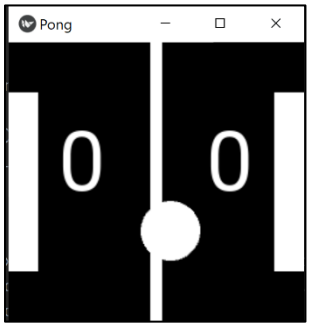


Abb. 5: Pong-App auf einem kleinen Screen

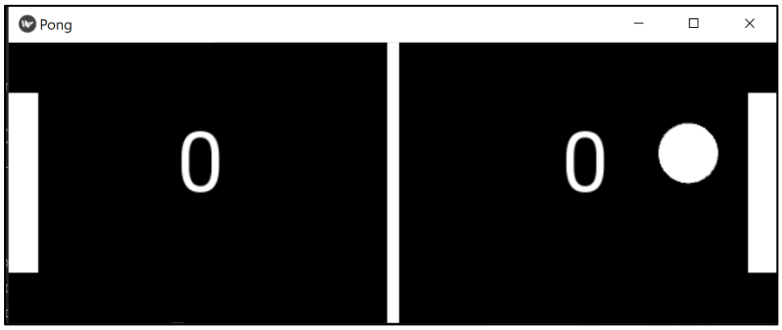


Abb. 4: Kivy-Pong App auf einem grossen Screen

Die Tutorials verhalfen zu einem ersten Einblick. Es wurden jedoch nur wenige Elemente vorgestellt, die für die Meeting-Finder Applikation von Nutzen sein würden.
Durch die Erstellung von vielen kleinen Applikationen konnte man sich mit den Funktionen vertraut machen. So lernte man Buttons, Text-Felder, Listen, Pop-Up-Fenster und vieles mehr zu erstellen. Die Resultate waren zwar funktionsfähig, jedoch entsprach die Darstellung nicht den Erwartungen.
Auf der Suche nach Möglichkeiten die Applikation optisch moderner zu gestalten, wurde man auf KivyMD aufmerksam.

5.6.2 Erste Schritte KivyMD

Obwohl KivyMD spezifisch für Kivy Applikationen entwickelt wurde, ist die Integrierung von KivyMD Elemente in einer Kivy Applikation nicht einfach umzusetzen. Um KivyMD Funktionen zu benutzen, muss man die Applikation von Beginn an mit der KivyMD Library erstellen, was folglich die Einarbeitung in KivyMD forderte.
Mithilfe der KivyMD erinnerten die Test Applikationen an die alltäglichen modernen Applikationen.

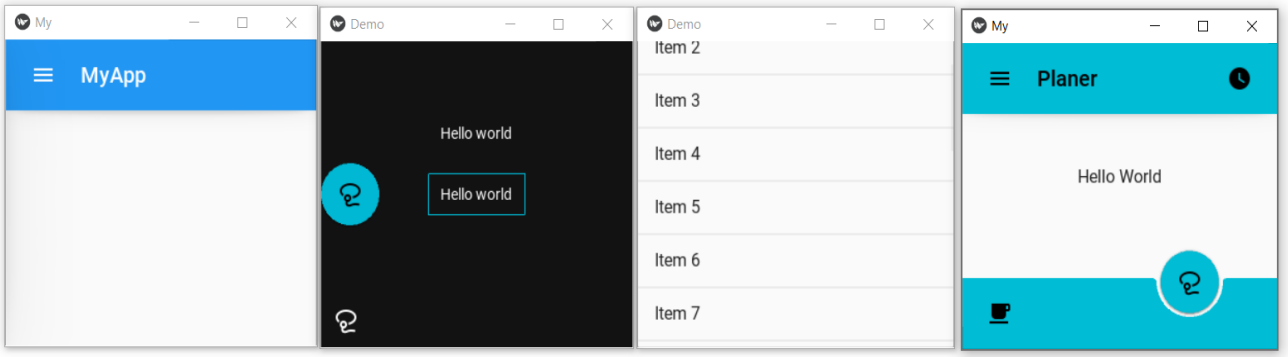


Abb. 6: Verschiedene Test-Applikationen die mit KivyMD erstellt worden sind

5.6.3 Startseite «MenuScreen»

Auf der Startseite kann der User seinen Usernamen eingeben. Da die App keinerlei sensible Daten beinhaltet, war der Nutzen eines Login Fensters gering, womit darauf verzichtet wurde. Die Startseite wurde im Verlauf des Projekts entfernt. Dies könnte zu einem späteren Zeitpunkt wieder integriert werden, falls das Programm erweitert werden soll.

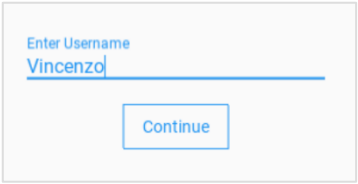


Abb. 7: MenuScreen Username-Eingabe-Fenster

Zurzeit gibt es keine Funktionen, die von einem Login profitieren würden. Das Programm kann dennoch mit dieser Funktion erweitert werden.
Für die aufgelisteten Funktionen würde das Login wieder sinnvoll sein:

- Den eigenen Kalender integrieren, um direkt eine Übersicht zu haben an welchen Daten man Zeit hat
- Die erstellten Umfragen in einem neuen Fenster auflisten
  - Meeting-Name & Link zur Umfrage
- Eine Benachrichtigung, wenn ein User an der Umfrage teilgenommen hat

5.6.4 Hauptseite «MainScreen»

Auf der Hauptseite kann der gewünschte Meeting-Name und die Dauer des Meetings bestimmt werden.
Im Verlaufe des Projektes wurde dies zur ersten Seite, womit beim Starten der Applikation direkt die Meeting-Umfrage erstellt werden kann.
Der Meeting-Name wird später auf der Umfrage Webseite angezeigt. Die Meeting-Dauer wird zu Beginn bestimmt. Der Gedanke dahinter ist, dass die Dauer der Meetings nicht variiert. Statt beim Auswählen der Zeiten immer VON - BIS auszuwählen, kann so einmalig die Dauer bestimmt werden und die Applikation addiert die Dauer zur Start-Zeit des Meetings.
Beide Text-Felder müssen ausgefüllt werden. Wenn diese nicht ausgefüllt sind, erscheint ein Pop-Up Fenster mit dem Hinweis, dass eine Eingabe fehlt.

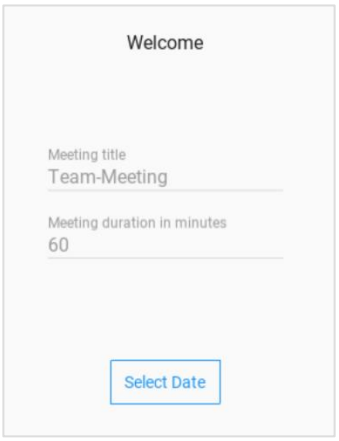


Abb. 8: Ansicht-MainScreen

5.6.5 Erstellungsseite «CreateScreen»

Auf dieser Seite kann das Datum und die Uhrzeit gewählt werden. Am Anfang des Projektes entsprach das Design der Abbildung 10. Mit dem Plus- und dem Minus-Zeichen war es möglich mehrere Datums- und Zeit-Felder hinzuzufügen.
Beim Testen wurde jedoch schnell klar, dass dieses Design nicht ideal war. Bei kleinen Bildschirmen konnten nur wenige Datums-Optionen hinzugefügt werden. Die anderen befanden sich nicht mehr im Sichtfeld des Users.
Anhand dieses Beispiels ist die Wichtigkeit und Einfluss von Design auf die Funktionalität zu erkennen. Grosse Firmen, wie Facebook und YouTube haben ganze Design Teams für Ihre Applikationen. Diese dienten als Orientierung an grundlegende Design Prinzipien. Die meisten grossen Apps haben

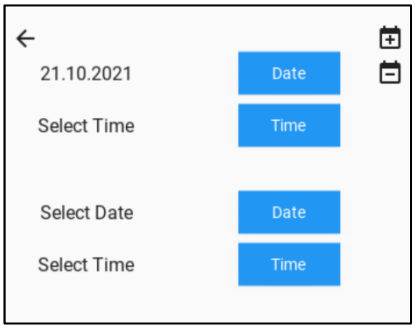


Abb. 9: CreateScreen - Erste Design Variante

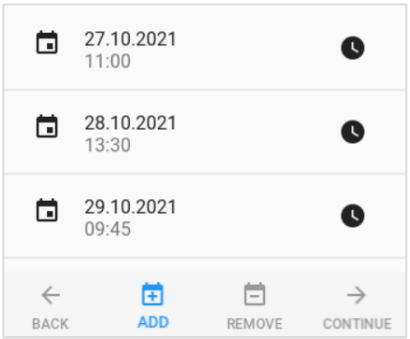


Abb. 10: CreateScreen - Finales Design

einen ähnlichen Aufbau, was ziemlich schnell erkennbar ist. Bei den meisten ist am oberem oder unterm Ende eine Navigations-Leiste. In der Mitte befand sich der Hauptinhalt, welcher meistens in einer «scrollbaren» Liste war.
Nach dem Re-Design waren die Datums- und Zeit-Vorschläge in einer Liste. Beim Anwählen der Liste oder des Kalenders öffnet sich die Kalenderanzeige. Da die Zeitangabe notwendig ist, wurde der Kalender direkt so eingestellt, dass unmittelbar nach der Datumsauswahl die Uhr geöffnet wird. Falls man nur die Zeit anpassen möchte, kann man auf das Uhr-Icon klicken, um die Zeit-Auswahl zu öffnen.
Mit dem ADD-Knopf in der Navigations-Leiste kann man einen Eintrag in der Liste hinzufügen. Mit dem REMOVE-Knopf kann man diese auch wieder entfernen. Auch hier wurde eine Eingabe-Überprüfung eingebaut. Der User muss nämlich mindestens einen Termin ausgewählt haben und die mit ADD hinzugefügte Elemente dürfen nicht leer gelassen werden.
Wenn alles vorhanden ist, kann der User mit CONTINUE fortfahren.

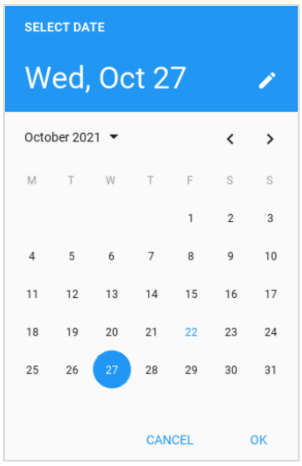


Abb. 11: KivyMD - DatePicker

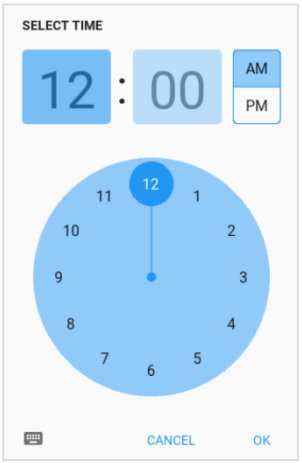


Abb. 12: KivyMD - TimePicker

5.6.6 Kontroll-Seite «ControlScreen»

Erst im Verlauf des Projektes hat sich die Idee einer Kontroll-Seite entwickelt. Der User hat auf dieser Seite einen finalen Überblick über seine ausgewählten Daten. Der Zweck dieser Seite ist erst dann gegeben, wenn man auch die Daten anpassen kann.
Folglich wurde das Zurückkehren auf die vorherige Seite ermöglicht, um die gewünschten Angaben anzupassen.
Der Nutzer kann bei zufriedenstellender Eingabe die Auswahl bestätigen.

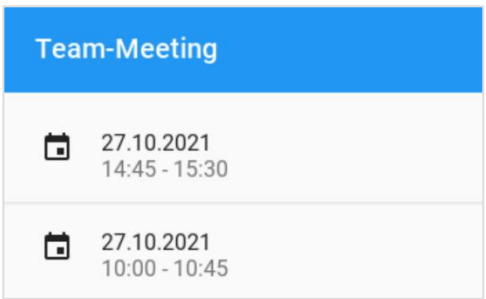


Abb. 13: Ansicht - ControlScreen

5.6.7 Link-Seite «LinkScreen»

Auf der letzten Seite der Applikation bekommt der User den Link zu der Umfrage. Dieser kann einfach mit dem Knopf COPY kopiert werden. Der Link kann danach einfach versendet werden. Da die Umfrage-Teilnehmer während der Arbeitszeit entweder im Büro oder via VPN mit dem Office-Netzwerk verbunden sind, haben sie Zugriff auf die Seite.
Die App kann nach Beendigung der Umfrage geschlossen werden oder es kann eine neue Umfrage erstellt werden.

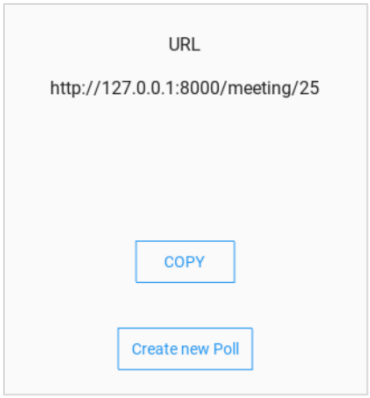


Abb. 14: Ansicht - LinkScreen

5.7 Django Webseite

5.7.1 Erste Schritte mit Django

Das Django Framework bietet ein Basisgerüst, mittels Python eine Webseite zu erstellen. Die ganze Logik der Webseite kann mit Python programmiert werden, während die Benutzeroberfläche wie für Webseiten üblich «html» Code ist.

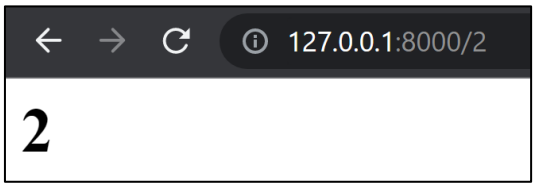


Abb. 15: Django – Erste Versuche mit «urls.py» und «views.py»

In den ersten Versuchen wurde eine Webseite erstellt, welche den gewählten Link auf der Seite wiedergibt. Wenn also, wie im Beispiel «/2» am Ende des Links einfügt, wird dieser auf der Webseite dargestellt. Dies ermöglicht den Zusammenhang zwischen «Frontend» und «Backend» zu verstehen.

5.7.2 Startseite/ Hauptseite

Die Startseite ist das Äquivalent der Kivy-Hauptseite. Der User kann den Meeting-Name und die Meeting-Dauer bestimmen.
Auch hier kommt eine Hinweis-Box, wenn ein Feld leer gelassen wird.

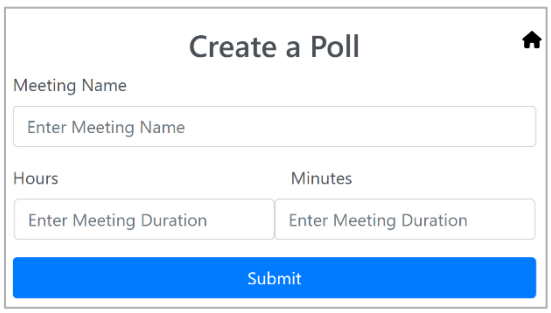


Abb. 16: Django - Startseite

5.7.3 Erstellungs-Seite

Auch hier ist die Funktion gleich, wie in der Kivy Applikation. Der User kann Terminvorschläge hinzufügen.

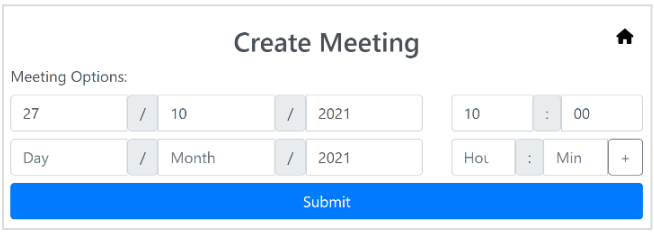


Abb. 17: Django- CreateScreen

5.7.4 Link-Seite

Zwischen der Erstellungs-Seite und der Link-Seite wird im Hintergrund die Umfrage generiert. Der User kann auch hier einfach den Link kopieren und weiterleiten.

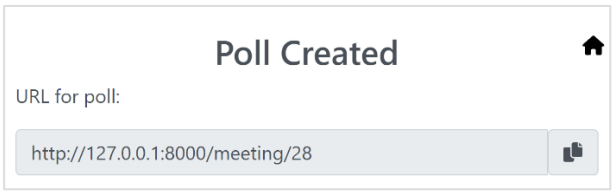


Abb. 18: Django - LinkScreen

5.7.5 Umfrage

Auf der Webseite kann man entweder seine Auswahl treffen oder sich die Resultate ansehen. Wenn man seine Auswahl getroffen hat, wird man direkt auf die Resultat-Seite weitergeleitet.
Mit dem Haus-Icon kann man auf die Startseite der Webseite gehen.

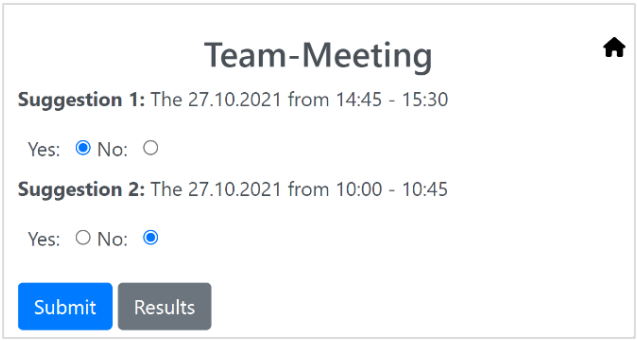


Abb. 19: Django – Umfrage-Seite

5.7.6 Resultat – Webseite

Auf dieser Seite ist ersichtlich an welchen Termin-Vorschlag die meisten Teilnehmer Zeit haben.

Abb. 20: Django – Resultat-Seite

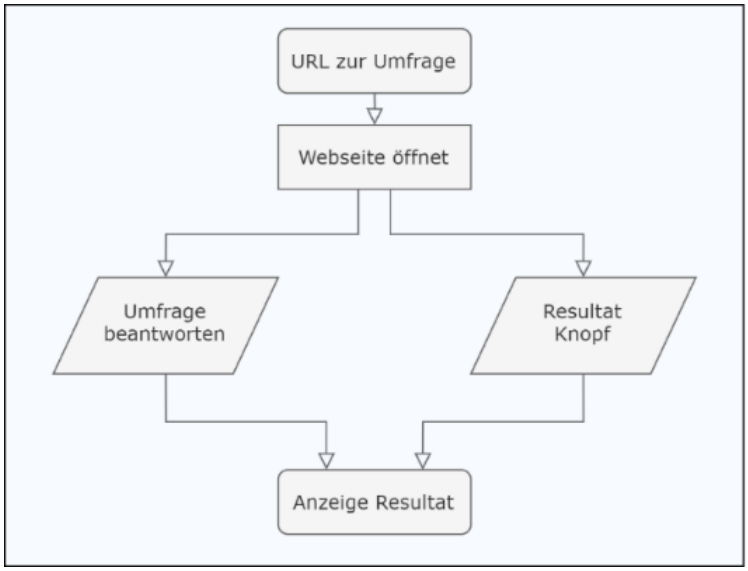


Abb. 21: Ablauf der Umfrage

5.8 Finaler-Aufbau

Während des Projekts sind einige Elemente neu hinzugekommen, wie beispielsweise die Kontroll-Seite. Hingegen haben sich andere als überflüssig erwiesen. Die finale Übersicht des Aufbaus der Webseite und Applikation wird in der folgenden Abbildung grafisch dargestellt.

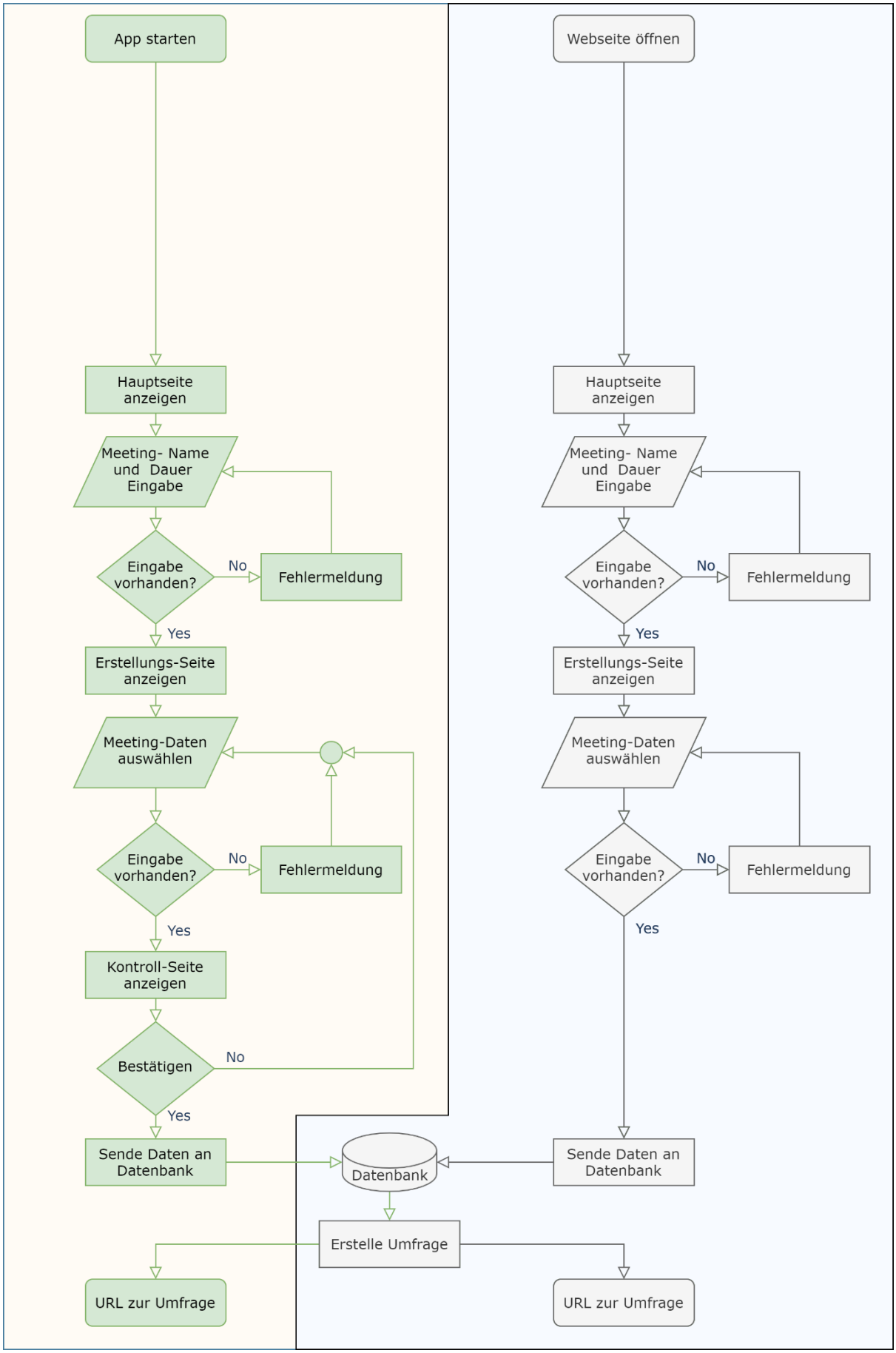


Abb. 22: Finaler Ablauf der App und der Webseite

6 Dokumentation

6.1 Kivy-Applikation

6.1.1 Übersicht

Die Applikation ist in den zwei Files aufgeteilt: «main.py» File und «kvfile.kv» File.
Im Main-File ist der grösste Teil der Logik. Aufgeteilt ist das Ganze in 5 Klassen. Die MyApp-Klasse startet die Applikation und ruft das Kivy-File ab. Alle anderen Klassen werden dann über das Kivy-File und den «ScreenManager» gestartet. Für jeden Screen gibt es eine Klasse.
Das Kivy-File ist ähnlich aufgebaut. Der «ScreenManager» beinhaltet alle 4 Screens. Im Kivy-File sind hauptsächlich Design Elemente vorhanden und simple Logik wie die Screen Reihenfolge.

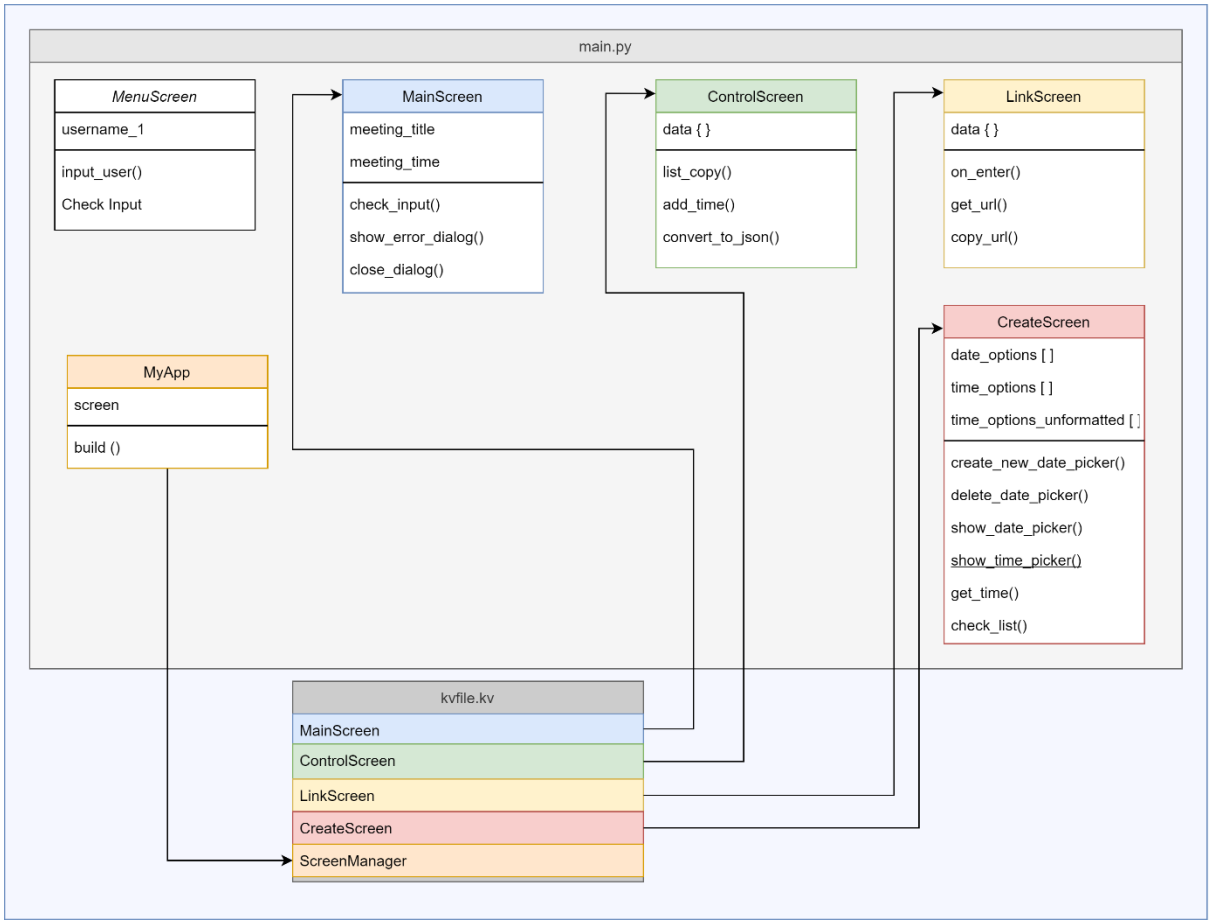


Abb. 23: Interaktion zwischen «main.py» File und KV-File

6.1.1.1 Klassen

Eine Übersicht über die wichtigsten Variablen und Funktionen der Applikation:

- **class MyApp**
  - screen
    - enthält das Kivy-File
  - build()
    - erstellt die Applikation mit dem Kivy-File

- **class MenuScreen**
  - username_1 - String
  - input_user()
    - Funktion, die den eingegebenen Username auf die Variable username_1 speichert

- **class MainScreen**
  - meeting_title - String
  - meeting_time - Integer
    - Auswahl der Länge des Meetings
  - check_input()
    - Überprüft, ob «meeting_title» und «meeting_time» vorhanden sind
  - show_error_dialog()
  - close_dialog()

- **class CreateScreen**
  - data_options [] - List
    - Liste mit gewählten Daten
  - time_options [] - List
    - Liste mit gewählten Zeiten
  - create_new_date_picker()
    - Erstellt ein neues Auswahl-Feld in der Liste
  - delete_date_picker()

        - Entfernt ein Auswahl-Feld aus der Liste und löscht die vorhandenen Daten dieses Feldes
    - show_date_picker()
        - Öffnet den Kalender und speichert die Auswahl in «data_options []»
    - show_time_picker()
        - Öffnet die Uhr und speichert die Auswahl in «time_options []»
    - get_time()
        - Wandelt die Zeit-Variable um: «10:00»

- **class ControlScreen**
    - data {} - Dictionary
        - JSON-Format aller Eingaben
    - add_time()
        - Addiert die «meeting_time» zu den ausgewählten Zeiten
    - list_copy()
        - Sammelt die Angaben von MainScreen und CreateScreen
    - convert_to_json()
        - Wandelt die gesammelten Daten in ein JSON Format um

- **class LinkScreen**
    - data {}
    - on_enter()
        - Funktion, die beim Öffnen der Seite startet, diese ruft die Funktion «get_url()» auf
    - get_url()
        - Sendet die JSON-Datei «data {}» zur Webseite und fordert den Link zur Umfrage an
    - copy_url()
        - Kopiert den Link in den Zwischenspeicher

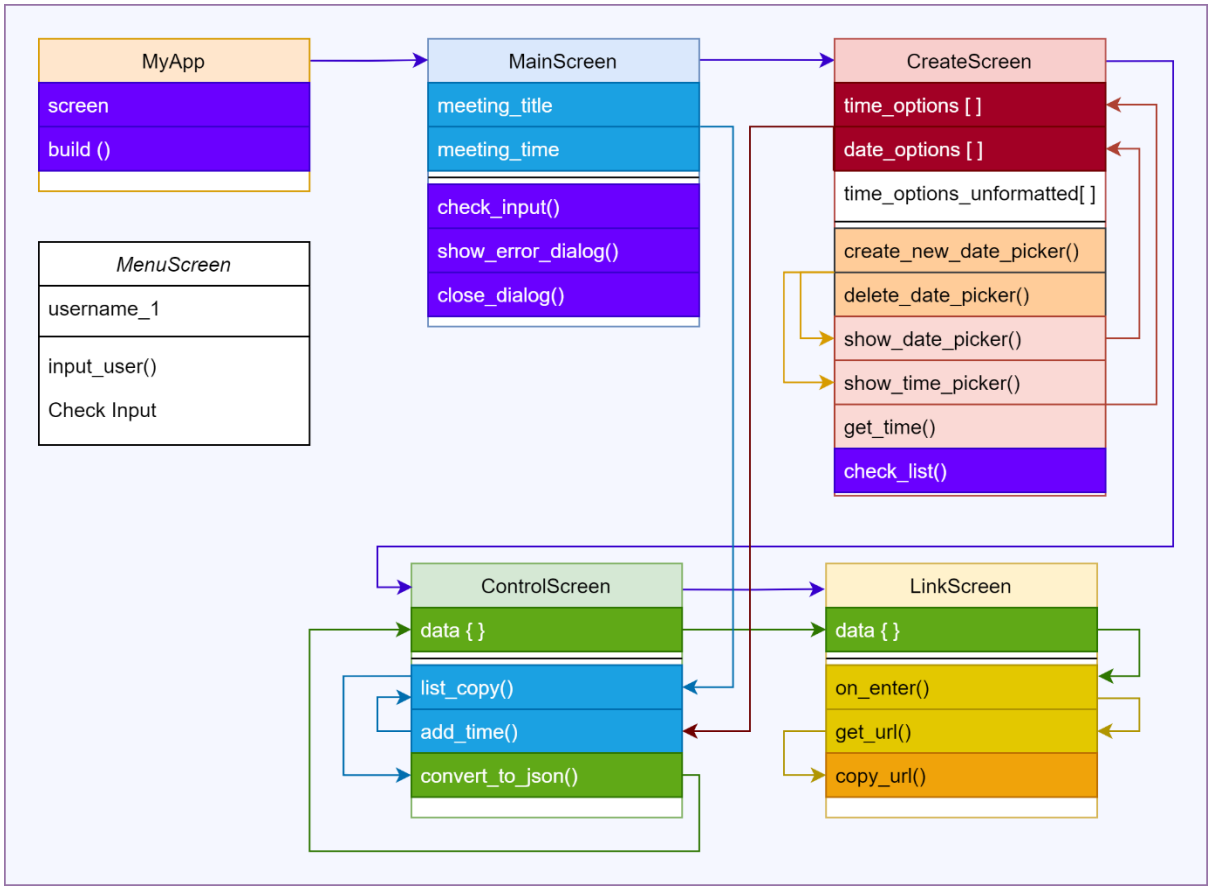


Abb. 24: Klassen-Zusammenhang der Kivy-App

6.1.2 MainScreen

Die KivyMD Text-Felder «MDTextField» brauchen eine ID, damit man sie im Python-Code abrufen kann. Der Text-Input kann folgenderweise abgerufen werden.

```
self.ids.idname.text
```

Wenn man auf den Knopf «Select Date» klickt, wird die Funktion check_input() gestartet. Beide Text-Felder werden in der Funktion überprüft. Wenn einer der beiden Felder leer ist, wird eine Fehlermeldung angezeigt und der Screen wird nicht gewechselt. Falls beide Felder ausgefüllt sind, wird der Bildschirm gewechselt.

6.1.3 CreateScreen

6.1.3.1 MDList

Das MDList Element ermöglicht es eine Liste darzustellen. Dies funktioniert nur in Zusammenspiel mit dem «ScrollView» Element. In der Liste können verschiedene Elemente hinzugefügt werden. Ein «TwoLineAvatarIconListItem» ermöglicht es zwei Icons und zwei Reihen Text hinzuzufügen.

6.1.3.2 MDList Probleme

Die Dokumentation von der MDListe behandelt hauptsächlich die Erstellung mit der KV-Language. Jedoch müssen für eine Variable Liste die Elemente via Python-Code hinzugefügt werden. Dies ist nicht gelungen.
Erst gegen Ende des Projektes wurden StackOverFlow Beiträge gefunden, die sich mit dieser Thematik beschäftigten.
So scheiterten erste Versuche schon beim Aufrufen der Funktion. Diese muss folgenderweise aufgerufen werden.

```
TwoLineAvatarIconListItem(text=f"Single-line item {i}",
                          secondary_text="Test",
                          on_release=
                          lambda x: self.show_date_picker(i)))
```

Durch «lambda x:» wird eine unbenannte Funktion aufgerufen, wenn das «on_release» Ereignis ausgelöst wird. Ohne «lambda x:» würde beim Erstellen des Lists-Items auf ein «return» von der Funktion gewartet werden.

Auch die ID des Elementes musste hier anders eingefügt werden. In der KV-Language kann die ID auf dieselbe Weise, wie die anderen Eigenschaften hinzugefügt werden.

```
MDLabel:
        id: id_name
        text: "Welcome"
        halign: "center"
```

Im Python-Code muss dies in einem separaten Schritt gemacht werden.

```
mdtitle = (
            MDLabel(
                text="Welcome",
                halign="center"
            ))
        self.ids["id_name"] = mdtitle
        self.ids.mlist.add_widget(mdtitle)
```

Mit der Hilfe dieser Änderungen funktionierte nun die variable Erstellung von List-Item. Das Entfernen der Items funktionierte wiederum noch nicht. Dies war gegen Ende des Projektes, womit die Entscheidung getroffen wurde mit dem Workaround fortzufahren. Diese MDList Version wird im Anhang unter dem Namen «List» zu finden sein.

6.1.3.3 MDList Workaround

Um das Problem mit der variablen Liste zu umgehen, wurde ein Workaround erstellt. In diesem werden die List-Items mit der Hilfe des Kivy-Files erstellt.
Die Items sind beim Öffnen der «CreateScreen» Seite noch leer. Durch das Betätigen des Knopfes «ADD» werden die Icons der Items sichtbar. Jedes Mal, wenn «ADD» betätigt wird, werden die Listen «date_options[]» und «time_options[]» um eins erhöht. In diesen Listen werden folglich die gewählten Daten und Zeiten gespeichert.

Durch das Drücken auf dem List-Item oder dem Kalender-Icon wird der DatePicker geöffnet. Um zu verhindern, dass auch die ausgeblendeten Items den DatePicker öffnen, wird vorher durch eine «if» Abfrage geprüft, ob die Nummer des List-Items höher ist als die Elemente in «date_options[]».

```
    def show_date_picker(self, button_nr):
        self.button_nr_atm = button_nr
        if button_nr > (len(self.date_options)):
            pass
```

Wenn die Icons sichtbar sind, können diese auch angeklickt werden. Nach der durch Ok bestätigte Auswahl des Datums wird der TimePicker geöffnet. Im Hintergrund wird das Datum umformatiert von der amerikanischen Schreibweise (2021-1-31) zu der europäischen (31.1.2021).

Die Zeit-Auswahl wird nach der Bestätigung ebenfalls von (13:00:00) auf (13:00) umformatiert.
Beide Angaben werden dann im List-Item sichtbar sein. Durch den REMOVE Knopf kann das letzte List-Item gelöscht werden.

6.1.3.4 List-Elemente hinzufügen

Im KV-File muss unter «CreateScreen» und «ControlScreen» ein List-Item hinzugefügt werden. Diese Items müssen gleich aufgebaut sein, wie die anderen. Die Zahlen müssen jeweils um eins erhöht werden.

```
TwoLineAvatarIconListItem:
                    id: list8
                    text: ""
                    secondary_text: ""
                    IconLeftWidget:
                        id: icon8
                        icon: ""
```

Im Python-File «main.py» muss in der Klasse CreateScreen die Variable «amount_of_list_elements» auf die gewünschte Anzahl erhöht werden.

6.1.3.5 MDBottomNavigation

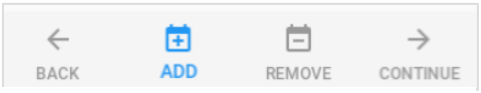


Abb. 25: Bottom-Navigation

Die Bottom-Navigation-Funktion erlaubt es, wie der Name es bereits sagt, zwischen mehreren Inhalten zu navigieren und auf dem gleichen Screen zu bleiben.
Diese Funktion kennt man von vielen Applikationen. Hier wurde jedoch die Bottom-Navigation umfunktioniert. Die zwei Pfeil-Icons werden zum Wechseln der Seite benötigt. Die Kalander-Icons fügen ein List-Item hinzu oder entfernen eines davon.
Die Bottom-Navigation musste angepasst werden, da durch die Änderung nun kein Inhaltswechsel mehr stattfand. Dies gelang durch das Entfernen der Variable «Screen», die im Normalfall den neuen Inhalt wiedergeben würde.

6.1.3.6 check_list()

Bei der Betätigung des ADD Knopfes wird der Zeit- und Datumsliste jeweils das Item «0» hinzugefügt. Wenn man mit CONTINUE fortfahren

will, wird die Funktion «check_list()» gestartet. Diese überprüft, ob sich noch eine «0» in den Listen befindet und kann damit erkennen, ob ein Feld noch unausgefüllt ist. Auch hier wird im Falle eines leeren Feldes ein Pop-Up-Fenster erscheinen, welches den User auf die fehlende Angabe hinweist.

6.1.4 ControlScreen

6.1.4.1 list_copy()

Im KV-File kann durch den Eintrag «on_enter» ein Befehl beim Öffnen des Screens gestartet werden. Die «list_copy()» Funktion wird so gestartet. Diese erstellt eine Liste mit den selektierten Terminen und addiert die Dauer des Meetings zu den Meeting-Startzeiten.

Das Addieren der Zeit war schwieriger als erwartet. Mit der Funktion «combine()» ist es nicht möglich eine Zeit ohne Datum zu addieren. Deshalb musste zur Meeting Zeit ein Datum hinzugefügt werden, welches am Ende der Operation wieder entfernt wurde.

6.1.4.2 convert_to_json()

«JSON» ist ein kompaktes Datenformat in einer einfach lesbaren Textform welches oft für den Datenaustausch zwischen Anwendungen genutzt wird.
Das Application Programming Interface (API) braucht folgendes «JSON» Format, um die Daten zu bearbeiten:

```
json.dumps({
    "name": "Meeting-Name",
    "duration": "60",
    "appointments": [
        {
            "date": "27.10.2021",
            "time": "09:00"
        },
        {
            "date": "28.10.2021",
            "time": "19:00"
        }
    ]
})
```

6.1.5 LinkScreen

6.1.5.1 get_url()

Hier wird die JSON-Datei zum API gesendet. Die URL der Webseite kann unter der String-Variablen «main_url_website» angepasst werden. Hier muss nur die Startseite (index.html) angegeben werden.
Nachdem die Webseite die Daten erhalten hat und eine Umfrage erstellt worden ist, wird der Link zur Umfrage zurückgesendet und auf die Applikation angezeigt.

6.1.5.2 copy_url()

Diese Funktion speichert die URL der Umfrage in den Zwischenspeicher des benutzten Gerätes, welcher anschliessend geteilt werden kann

6.1.6 MenuScreen

Die Seite «MenuScreen» wird vorerst nicht gebraucht und kann zu einem später Zeitpunkt wieder integriert werden. Dazu müssten folgende Schritte durchgeführt werden:

- «main.py»
  - MenuScreen Klasse erstellen
    - Zeile 17-22 entkommentieren
  - ScreenManager Widget hinzufügen Zeile 287
- kvfile.ky
  - MenuScreen zu ScreenManager Liste hinzufügen
  - Vom File MenuScreen.txt den KV-Code in Zeile 8 hinzufügen
  - Back-Knopf hinzufügen
    - Zeile 38-42 entkommentieren

6.2 Django Webseite

6.2.1 Übersicht

Die Webseite besteht aus vier Ordnern:

- PlanerWeb
- main
- api
- templates

Der PlanerWeb-Ordner wurde beim Starten des Django Projektes erstellt. Hier sind alle Komponenten vom Framework zu finden, unter anderem auch die Admin Seite. Der main-Ordner beinhaltet die Web-Applikation. Diese ermöglicht die Erstellung der Umfrage und greift dafür auf die HTML Templates zu.
Der API-Ordner ermöglicht es die Daten von der Applikation zu empfangen und sendet den URL zur Umfrage zurück. Das API wurde mit dem Django REST Framework erstellt.

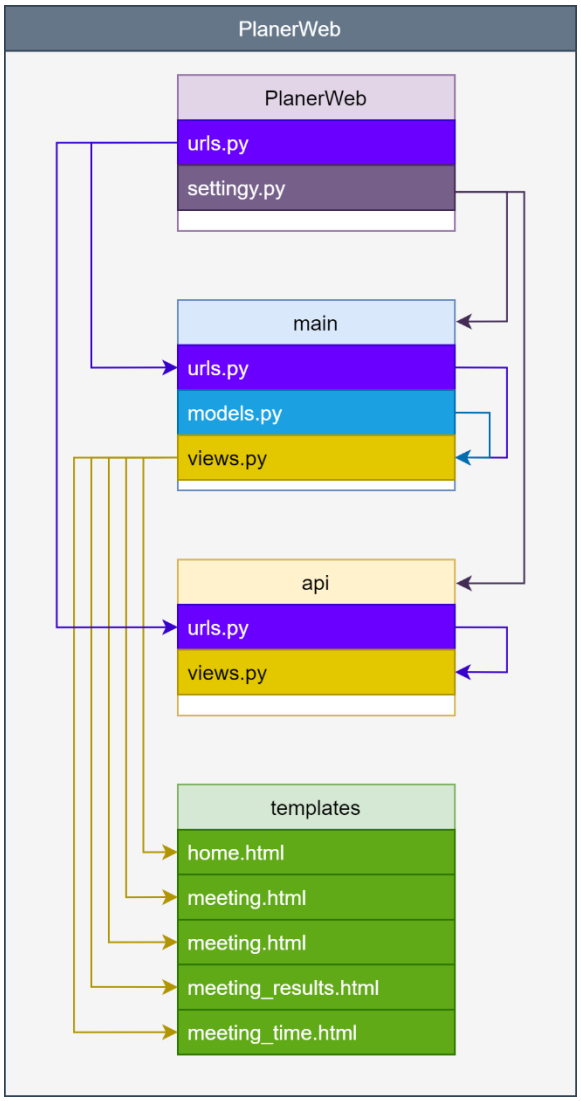


Abb. 26: Interaktion zwischen Django Projekt und Web-Applikationen

6.2.2 PlanerWeb

PlanerWeb ist das Projekt-File. Projekte sind in Django eine Sammlung von Konfigurationen und Applikationen für die spezifische Webseite. Ein Projekt kann mehrere Applikationen beinhalten.

Die meisten Files wurden durch das Django Framework erstellt. Daher wird nicht auf alle Files und jeden Code des Projektes eingegangen, da diese für dieses Projekt nicht benötigt werden oder nichts daran geändert wurde

6.2.2.1 settings.py

Im File «settings.py» können viele Grund-Einstellungen gemacht werden. Für dieses Projekt sind die Abschnitte INSTALLED_APPS und DATABASES wichtig.
Unter «INSTALLED_APPS» müssen die neu erstellten Applikationen «main» und «api» hinzugefügt werden, damit Django diese als Teil des Projektes miteinbezieht.
Unter DATABASES kann eingestellt werden welche Datenbank für die Webseite gebraucht werden soll. Hier wurde die Standard Datenbank SQL gebraucht.

6.2.2.2 urls.py

In diesem Python-File werden die URLs des Projektes festgelegt.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include("main.urls")),
    path("api/", include("api.urls"))
]
```

An dieser Stelle werden die verschiedenen Pfade eingestellt. So wird die URL Endung «admin/» den User zur Admin Seite führen. Wenn nur der Hauptlink eingegeben wurde, öffnet sich das «main.url.py» File in der Applikation. Dieses würde in diesem Fall den «home» Screen mithilfe des gleichnamigen HTML-File erstellen.

6.2.3 main-App

Die «main-App» ist das Herzstück der Webseite. Hier kann eine Umfrage erstellt werden und die Webseite generiert diese anschliessend.

6.2.3.1 urls.py & views.py

Auch die «main» Web-Applikation beinhaltet ein «urls.py» File. Dieses beinhaltet alle Links, die für den User der Webseite nötig sind.
Die beiden Files «urls.py» & «views.py» sind eng miteinander verknüpft. So ruft die URL die vorgegebene Funktion in den «views.py» auf. Diese wiederum öffnet das dazugehörige HTML-File und speichert die ausgewählten Daten in der Datenbank ab.

6.2.3.2 models.py

In diesem File werden die Datenmodelle «models» festgelegt. Ein «model» definiert die Daten, die in der Datenbank gespeichert werden. Es beinhaltet die essenziellen Felder und Eigenschaften der Daten, die man speichern will.
Sobald die Datenmodelle erstellt sind, wird von Django automatisch eine Datenbank-abstraction-API zur Verfügung gestellt. Diese erlaubt es Objekte erstellen, abrufen, aktualisieren oder löschen zu können.

6.2.4 api-App

Das API ist die Programmschnittschnittstelle zwischen der Kivy-App und der Django-Website. Das API ist auf der Django-Webseite als weitere Web-Applikation integriert.
Wenn der Web-Server gestartet wurde, kann man via «/api/get_meeting_link» auf das Django-REST-Framework-Interface zugegriffen werden. Dort kann man diverse Tests durchführen.
Die Applikation ist so aufgebaut, dass keine falsch formatierte Datei versendet werden kann. Trotzdem überprüft das API das Format, um auch Anfragen ausserhalb der App bearbeiten zu können.

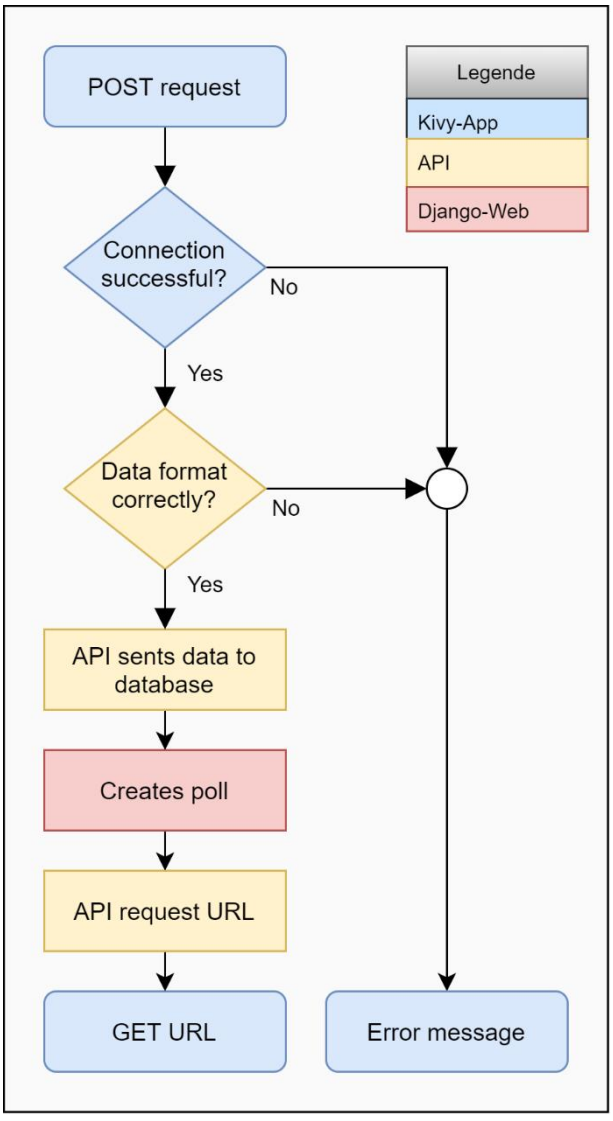


Abb. 27: Ablauf der POST-request

6.2.5 views.py

In diesem File wird die POST Anfrage überprüft. Wenn das Format nicht mit der Vorgabe übereinstimmt, wird eine Fehlermeldung zurückgesendet.
Wenn das JSON-Format richtig ist, wird ein «Meeting» und ein «Appointments» Objekt erstellt.

7 Reflexion

Im Rahmen der Ausbildung als Dipl. Techniker HF Informatik Fachrichtung Systemtechnik habe ich mich mit der Erstellung des Meeting-Finder Tools auseinandergesetzt. Es handelt sich um mein erstes grösseres Programmier-Projekt. Die mangelnde Erfahrung hatte einen grossen Einfluss auf den Terminplan, von dessen Kurs ich schon früh im Projekt abweichen musste. Dank der grosszügigen Einplanung von Zeitreserven wurden die Ziele dennoch erreicht.
Die Grundfunktionen der Applikation konnte ich zeitig integrieren, was mich dazu motivierte schon früh die App zu optimieren. Dieser Prozess dauerte schlussendlich länger als die Integration der Grundfunktionen, was den Pareto-Effekt bestätigt. So war die Erstellung der Liste zur Zeit- und Datumsauswahl einer der aufwendigsten Aufgaben. Das Einsetzen von mehreren Kivy-Tools, die miteinander interagieren mussten, stellte sich als schwierige Aufgabe heraus. Die Dokumentation war zwar übersichtlich aber unvollständig. Erst zu einem späteren Zeitpunkt im Projekt stellte ich fest, dass KivyMD sich noch in der Alpha-Version befindet. Durch die zu hohe Zeitinvestition in die App blieb später weniger Zeit für die Webseite übrig.
Die App wurde in der gleichen Reihenfolge erstellt, wie sie aufgebaut ist. Dies bewirkte, dass ich mich immer durch die ganze Applikation durchklicken musste, um die hinzugefügten Funktionen zu testen. Der Debug-Mode, der in den meisten Integrierten-Entwicklungsumgebung vorhanden ist, sollte genau dieses Problem umgehen. Dies funktionierte dabei leider nur bedingt. Da das KV-File nur beim Starten der App geladen wird, musste nach einer Anpassung die App neugestartet werden. In Zukunft würde ich daher den «Start-Screen» oder «Menu-Screen» erst gegen Ende des Projektes erstellen. Mit der Erfahrung, die ich bei der App Erstellung gewinnen konnte, konzentrierte ich mich bei der Webseite auf die wesentlichen Funktionen.

Für dieses Projekt musste ich mich zum ersten Mal in Libraries einarbeiten. Anhand der Kivy und KivyMD Dokumentation konnte ich schnell die benötigten Tools ausfindig machen. In diesen wurde meistens beschrieben, wie ein Widget via KV-Language eingefügt werden kann. Mir fehlte dabei oft die Beschreibung dazu, wie das gleiche in Python erreicht werden kann.
Bei der Django Dokumentation hatte ich etwas mehr Mühe. Django ist ein grosses Framework und ich war oft überwältigt von der Anzahl an

Tools, die mir zur Verfügung standen. Zusätzlich sind viele Schritte in Django automatisiert. So versuchte ich anfangs noch eine Datenbank in Django zu integrieren, obwohl diese bereits erstellt wurde. Sobald ich aber die richtigen Angaben der Dokumentation gefunden hatte, konnte ich dies schnell integrieren. Durch die verkürzte Zeit, die ich in der Webseite investieren konnte, beliess ich es bei einem Standard-Design, welches sicherlich noch ausbaufähig ist.
Was ein Application-Programming-Interface ist, wusste ich bei der Themeneingabe noch nicht. Erst Im Verlauf des 6. Semester lernten wir im Fach Web-Engineering den Nutzen und Umgang mit APIs. Ich war der Meinung, dass die App direkt mit der Webseite kommunizieren könnte. Das Django Framework verhindert aber aus Sicherheitsgründen diese direkte Kommunikation.
Die Erstellung der API verlief sehr gut, jedoch hatte ich mit der Erstellung des richtigen JSON Format zu kämpfen. Ein anderes Problem verursachte die Verbindung zum API. Diese konnte anfangs nicht erreicht werden, da ein Slash-Zeichen am Ende der URL fehlte. Ich konnte keine Rückschlüsse daraus ziehen, wieso dieses “Slash” die Verbindung verhinderte.
Insgesamt bin ich mit der Arbeit zufrieden, trotz der vielen Schwierigkeiten konnte ich das gewünschte Produkt in der vorgegebenen Zeit erstellen.

8 Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und die hierzu verwendeten Quellen und Hilfsmittel im Literaturverzeichnis angegeben habe.
Die verwendeten Quellen wurden in der Arbeit als Zitate bzw. Paraphrasen kenntlich gemacht.
Diese Diplomarbeit ist noch nicht veröffentlicht worden. Sie ist somit weder anderen Interessenten zugänglich gemacht noch einer anderen Prüfungsbehörde vorgelegt worden.

Ort, Datum

Unterschrift

9 Quellenverzeichnis

Bonacina, Michael. *Python 3 Programmieren für Einsteiger*. BMU Verlag, 2020.

Bootstrap. *Validation*. -. https://getbootstrap.com/docs/5.0/forms/validation/ (Zugriff am 20. 10 2021).

Brihadiswaran, Gunavaran. *Medium*. 2020. https://medium.com/swlh/a-performance-comparison-between-c-java-and-python-df3890545f6d (Zugriff am 25. Oktober 2021).

buildwithpython. *Kivy Tutorial* . 2020. https://www.youtube.com/watch?v=LRXo0juuTrw&list=PLhTjy8cBISEoQQLZ9IBlVlr4WjVoStmy- (Zugriff am 25. Oktober 2021).

Django Documentation. *django-admin and manage.py*. kein Datum. https://docs.djangoproject.com/en/2.0/ref/django-admin/#runserver (Zugriff am 25. Oktober 2021).

—. *django-rest-framework*. kein Datum. https://www.django-rest-framework.org/ (Zugriff am 25. Oktober 2021).

—. *first Django app*. - . https://docs.djangoproject.com/en/3.2/intro/tutorial01/ (Zugriff am 25. Oktober 2021).

—. *Templates*. kein Datum. https://docs.djangoproject.com/en/3.2/topics/templates/ (Zugriff am 25. Oktober 2021).

Hahn, Silke. *heise.de*. 13. Oktober 2021. https://www.heise.de/news/Programmiersprachen-Ranking-Viel-Bewegung-JavaScript-und-Python-konstant-vorne-5067987.html.

Mccarthy, Andrew. *Django Documation*. 2021. https://docs.djangoproject.com/en/3.2/.

Neel, Prakash. *Valuebound*. 2017. https://www.valuebound.com/resources/blog/overview-of-json-api#:~:text=Let%E2%80%99s%20start%20with%20what%20is%20JSON%20API%3F (Zugriff am 25. Oktober 2021).

Rodríguez, Andrés. *KivyMD documentation*. 2021. https://kivymd.readthedocs.io/en/latest/ (Zugriff am 25. Oktober 2021).

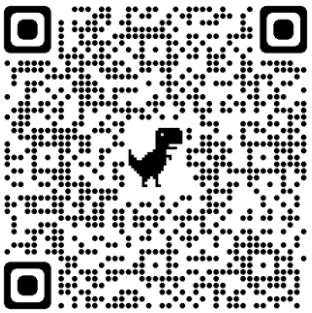
Stackoverflow. *Error when closing DatePicker*. 2021. https://stackoverflow.com/questions/68792587/kivy-error-when-i-close-the-datepicker-multiscreen (Zugriff am 25. Oktober 2021).

—. *How to set id of button*. 2018. https://stackoverflow.com/questions/50099151/python-how-to-set-id-of-button (Zugriff am 25. Oktober 2021).

—. *on_press in Kivy keeps running at start up instead*. 2021. https://stackoverflow.com/questions/12368390/on-press-in-kivy-keeps-running-at-start-up-instead (Zugriff am 25. Oktober 2021).
t3n. *6 Gründe, warum ihr spätestens 2021 Python lernen solltet*. März 2021. https://t3n.de/news/python-lernen-1194593/.
Tech With Tim. *Django For Beginners - Full Tutorial*. 2021. https://www.youtube.com/watch?v=sm1mokevMWk (Zugriff am 25. Oktober 2021).
TheKivy-Authors. *Kivy Documentation*. 2018. https://kivy.org/doc/stable/.

10 Abbildungsverzeichnis

11 Anhang



Share-Order Inhalt:

- PDF-Diplomarbeit
- Kivy-Applikation (PlanerApp)
- Django-Website (PlanerWeb)
- List (von Abschnitt 6.1.3.2)

https://drive.google.com/drive/folders/10JW865KfBBFrFjhfDirA1nnLNf4w5GQU?usp=sharing

Zwischengespräch 1

Ort: Teams

Datum: 18.10.2021

Uhrzeit: 17:00

Teilnehmer: Stephan Kessler, Vincenzo Vellone

Punkte der Agenda

1. Applikation und Webseite demonstrieren.
2. Probleme mit der Kommunikation zwischen App und Datenbank
3. Webseite Lokal oder via Ngrox hosten
4. Allgemeine Fragen zur Abgabe der Arbeit
5. Qualifikationsprofil

Protokoll

1. Applikation und Website funktionieren soweit. Es fehlt noch die Kommunikation zwischen der App und der Datenbank. In der App kann man bisher nur sechs Daten hinzufügen, Stephan hat darauf hingewiesen, dass ich dieses Problem dokumentieren soll, falls ich es nicht lösen kann.
2. Bisherige versuche direkt via App auf die Datenbank zu senden sind fehlgeschlagen. Django blockiert diese aus Sicherheitsgründen. Stephan hat darauf hingewiesen, dass mit Hilfe einer API die Kommunikation hergestellt werden kann. Da in der API programmiert werden kann, welcher Datenaustausch stattfinden kann und somit muss nicht im Source-Code von Django angepasst werden.
3. Wenn die Webseite nicht Lokal gehostet werden kann, soll analysiert werden, wieso dies nicht möglich ist.
4. Der Code der ganzen Arbeit kann zusätzlich abgegeben werden.
5. Das Qualifikationsprofil gibt es dieses Jahr zum ersten Mal und löst die Kompetenzkarten ab. Das Qualifikationsprofil soll aufzeigen, welchen erlernte Wissen vom Studium für die Diplomarbeit benötigt wurde.

Nächster Termin ist am 25.10.21

Zwischengespräch 2

Ort: Teams

Datum: 25.10.2021

Uhrzeit: 17:30

Teilnehmer: Stephan Kessler, Vincenzo Vellone

Punkte der Agenda

1. Frage zu try-except Funktion in Python.
2. Finales App und Webseite testen
3. Kivy variable Liste angeschaut
4. Überblick der Schriftlichen Arbeit

Protokoll

1. Der „except“ Block konnte die Error Meldung nicht auffangen. Nach einer kurzen Analyse der Fehlermeldung ist aufgefallen, dass noch zusätzliche Information fehlte damit der Error aufgefangen werden konnte. Der Error musste statt „except ConnectionError:“ folgerderweise erfasst werden: „except requests.exceptions.ConnectionError:“. Zusätzlich war im der Seitenlange Fehlermeldung noch ein weiterer Error vorhanden: „except urllib3.exceptions.NewConnectionError“.
2. Die App und Webseite wurden schon im letzten Zwischengespräch getestet. Nun musste nur noch die Kommunikation zwischen Webserver und der Applikation geprüft werden. Dies wurde durch die Rest Framework API ermöglicht.
3. Es ist nun möglich neue List-Item Elemente zu erstellen. Jedoch gibt es noch Probleme mit der Entfernung dieser Elemente. Daher wird weiterhin mit dem Workaround gearbeitet, was ein Limit von 8 List-Items hat. Das Problem wird detailliert in der Dokumentation behandelt.
4. Es wurde kurz der Aufbau der Arbeit besprochen. Von Aufbau her scheint alles vorhanden zu sein.