Projektdokumentation

Projekt Repository

PCC verwendet GitHub zur Versionskontrolle. Das Projekt kann hier gefunden werden. Ein Changelog mit den aktuellen Veränderungen des Programs kann hier gefunden werden.

Vorgehensmodell

Für die Entwicklung von PCC wird die agile Methode Extreme Programming (XP) verwendet. Diese Methode wurde gewählt, da sie sich gut für kleine Teams und kurze Entwicklungs-Zyklen eignet. Außerdem soll das Feedback von Usern schnell integriert werden können, was durch XP ermöglicht wird. Letztendlich wurde XP gewählt, um trotz kurzer Release-Zyklen und des Team Umfangs, eine Testbare und Fehlerfreie Software bereit zu stellen.

Libraries und Frameworks

PCC wird in der Programmiersprache Python umgesetzt. Es folgt eine tabellarische Übersicht über verwendete Libraries und Frameworks innerhalb des Projekts.

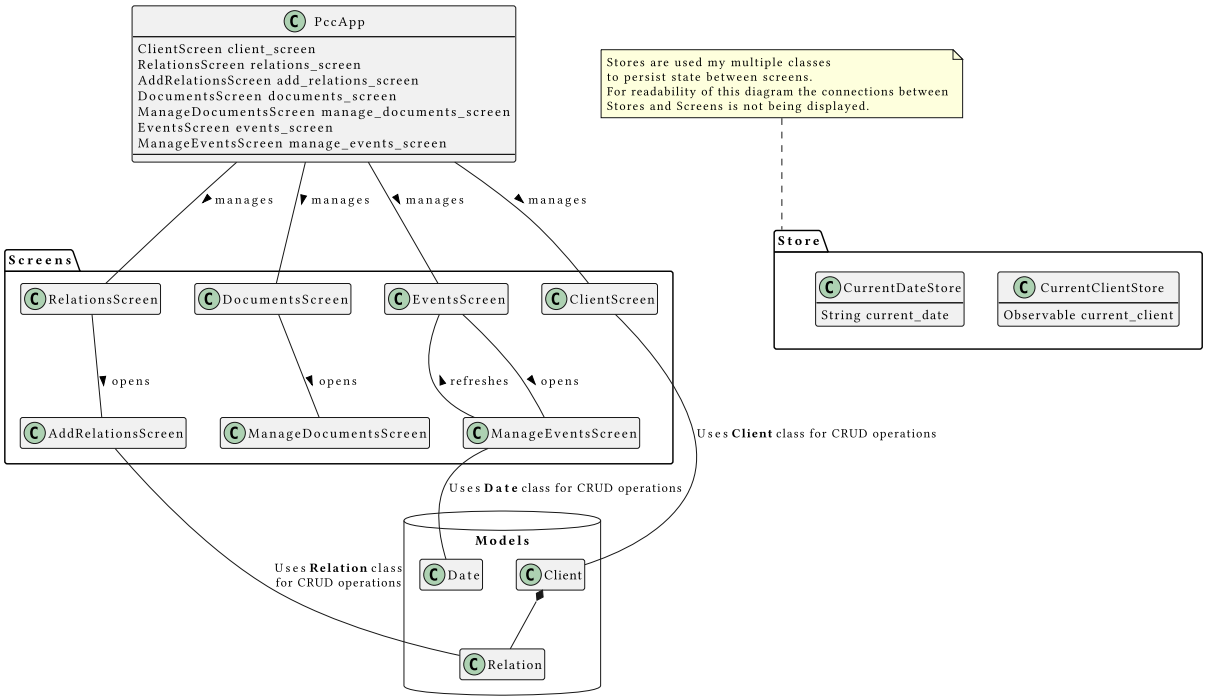
Library / Framework	Begründung	Link	Version
Kivy	Modernes UI Framework in Python für Platformübergreifende Anwendungen	https://pypi.org/project/Kivy/	2.2.1
KivyMD	GUI Library welche Material Design 3 implementiert	https://github.com/liquidiert/KivyMD.git	1.2.0dev1
plyer	Tooling Library welche einen nativen Dateiauswähl Dialog implementiert	https://pypi.org/project/plyer/	2.1.0
objectbox	Effiziente NoSQL Datenbank zur Persistierung von Client Daten	https://pypi.org/project/objectbox/	0.5.0
dateutil	Library für das Rechnen mit datetime Objekten	https://pypi.org/project/python-dateutil/	2.8.2

Exkurs: Deklarativer Programmierstil in KivyMD

KivyMD erlaubt es Komponenten deklarativ zu erstellen. Hierbei wird der Endzustand des Komponenten Baumes (Widget Tree) beschrieben, dadurch kann einfach nachvollzogen werden wie dieser aufgebaut ist. Der deklarative Programmierstil wird unter anderen auch im Cross Platform Framework Flutter benutzt. Genutzt wird dieser Stil weil er zu einem übersichtlichen Widget Tree führt und Abstraktion zwischen GUI Code und Logik erlaubt.

Programmstruktur

PCC ist modular aufgebaut. Es existiert eine Haupt-Klasse `PccApp` welche für den GUI intialisierungs Code verantwortlich ist und sämtliche Screen Klassen verwaltet, wie in folgendem UML-Diagramm zu sehen ist.



Zusätzlich zu den vier Haupt-Screens (`ClientScreen`, `RelationsScreen`, `DocumentsScreen` und `EventsScreen`) gibt es noch drei Sub-Screens auf die aus den Haupt-Screens verwiesen wird.

In `models` sind die Datenbank Modelle aufbewahrt. Sie dienen zur einfachen Interaktion mit der objectbox Datenbank. Es besteht eine Composition zwischen `Client` und `Relation`, da eine `Relation` nur mit einem entsprechenden `Client` existieren kann. Dies widerspiegelt sich auch in der PCC UI. Man muss zuerst einen Klient auswählen bevor man eine Beziehung anlegen kann.

Die Klassen des Moduls `store` sind Utility Klassen welche dem State der Anwendung temporär persitieren können. So kann man mithilfe der `CurrentClientStore` Klasse zum Beispiel einen `Client` im `RelationsScreen` zwischenspeichern und dann im `AddRelationsScreen` darauf zugreifen.

Pflicht Design Pattern

Als Design Pattern wurde das Model View Controller (MVC) Entwurfsmuster gewählt.

MVC ist ein häufig verwendetes Entwurfsmuster zur Entwicklung von Benutzeroberflächen, das die damit verbundene Programmlogik in drei miteinander verbundene Elemente unterteilt. Diese Elemente sind die interne Repräsentation von Informationen (das Model), die Benutzeroberfläche (die View), die Informationen dem Benutzer präsentiert und von ihm annimmt, und die Controller-Software, die die beiden verknüpft.

Zusammengefasst: Das MVC-Muster trennt die Programmlogik in Model, View und Controller, um eine klare Struktur und Trennung der Verantwortlichkeiten in der Benutzeroberflächenentwicklung zu schaffen.

Ein Anwendungsbeispiel aus PCC wäre zum Beispiel das Anlegen eines Clients. Es folgt der Code zur Erstelleung der View Komponente:

```
MDScreen(
     MDBoxLayout(
         MDBoxLayout(
             MDLabel(text="Clients", font_style="H3"),
             MDFloatingActionButton(
               icon="plus",
               on_press=self.open_add_dialog
             ),
             adaptive_height=True,
         ),
         self.data_table,
         padding=[12, 10],
         pos_hint={"top": 1},
         spacing="12dp",
         orientation="vertical",
         id="client_screen_box",
     ),
     name="clients",
)
```

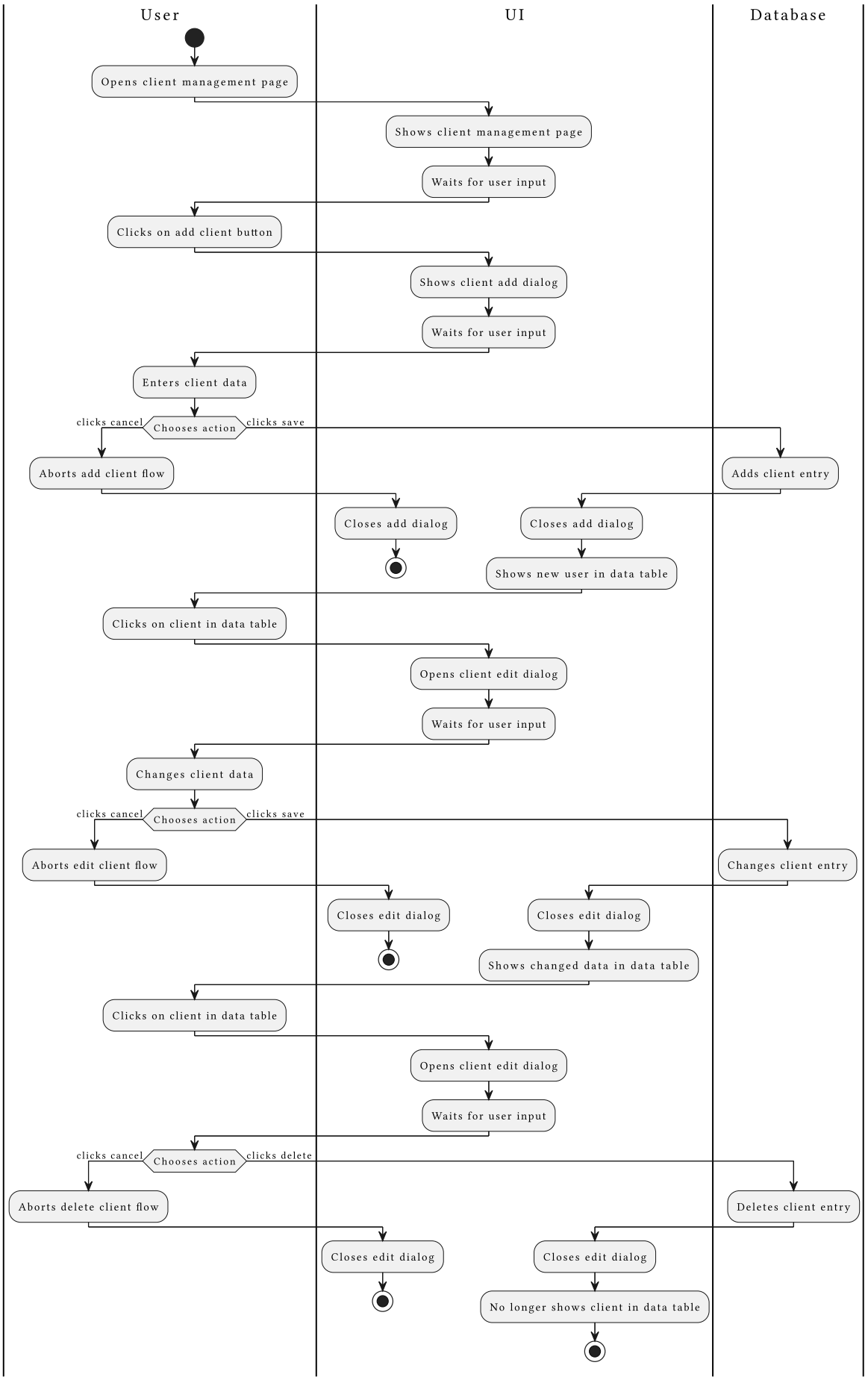
Obiger Code-Block erstellt einen Screen und einige UI Komponenten. Wir betrachten den `MDFloatingActionButton`, welcher eine Referenz zur Methode `open_add_dialog` enthält. Der Button agiert nicht nur als Element der View sondern zugleich auch als Controller. Folgender Code-Block stellt die Methode `open_add_dialog` dar:

```
def open_add_dialog(self, _):
    date_dialog = MDDatePicker()
    date_dialog.bind(on_save=self.save_date)

    if not self.add_dialog:
        self.add_dialog = self.get_dialog_template(
            "Add new client",
            date_dialog,
            self.close_client_add,
            self.save_client,
            lambda x: x,
        )

    self.add_dialog.open()
```

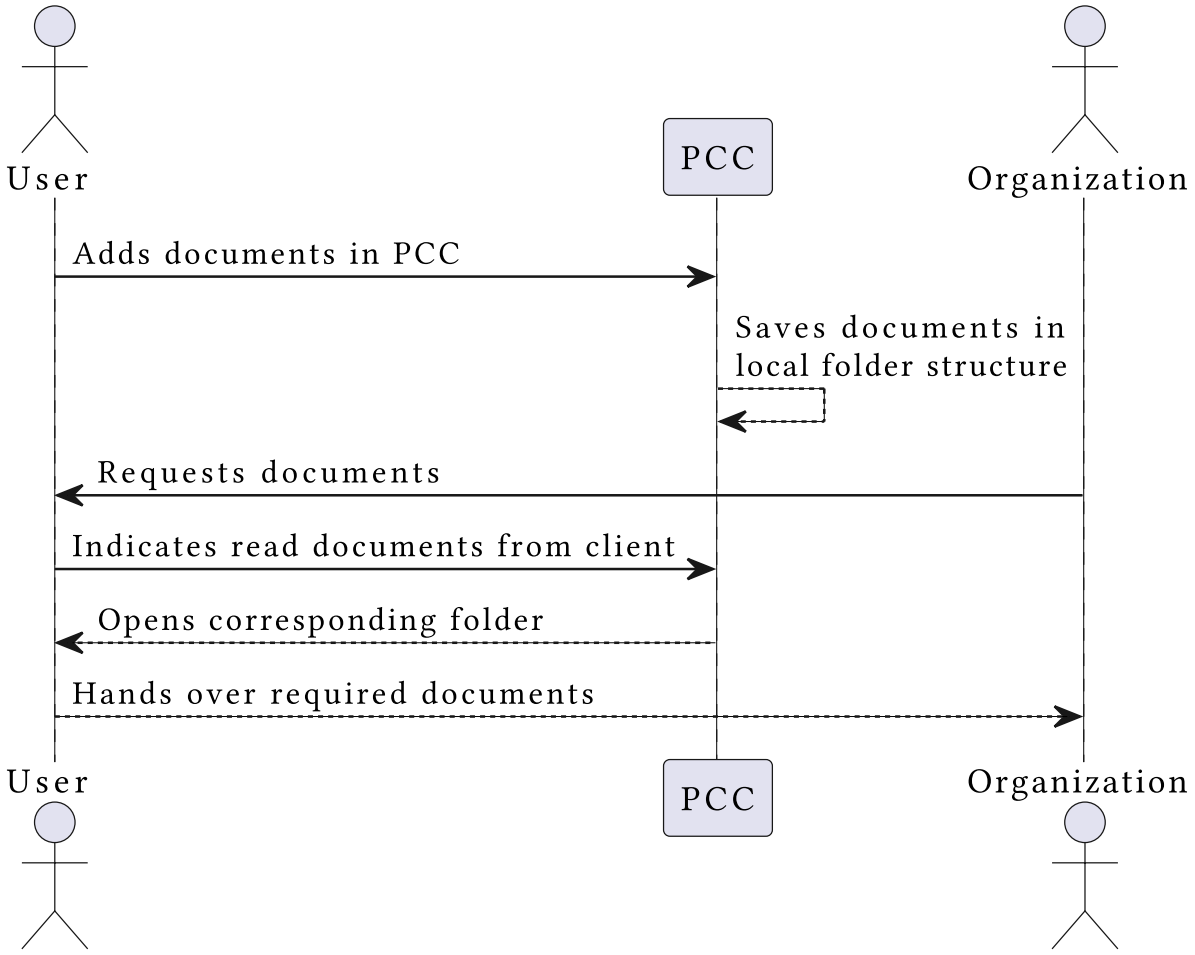
Im Wesentlichen ist diese Methode verantwortlich die View zu aktualisieren und einen Dialog darzustellen in welchem der User die Client Daten eingeben kann. Die Aktualisieren der View erfolgt mit dem letzen Call der Methode `self.dialog.open()`. Die `open` Methode wird durch Kivy zur Verfügung gestellt und kümmert sich automatisch um die Aktualisierung. Das Model das verändert wird ist der Screen welcher zuvor erstellt wurde.

Weitere UML Diagramme

Userverwaltungs Aktivitätsdiagram


User
UI
Database
Opens client management page
Shows client management page
Waits for user input
Clicks on add client button
Shows client add dialog
Waits for user input
Enters client data
clicks cancel
Chooses action
clicks save
Aborts add client flow
Adds client entry
Closes add dialog
Closes add dialog
Shows new user in data table
Clicks on client in data table
Opens client edit dialog
Waits for user input
Changes client data
clicks cancel
Chooses action
clicks save
Aborts edit client flow
Changes client entry
Closes edit dialog
Closes edit dialog
Shows changed data in data table
Clicks on client in data table
Opens client edit dialog
Waits for user input
clicks cancel
Chooses action
clicks delete
Aborts delete client flow
Deletes client entry
Closes edit dialog
Closes edit dialog
No longer shows client in data table

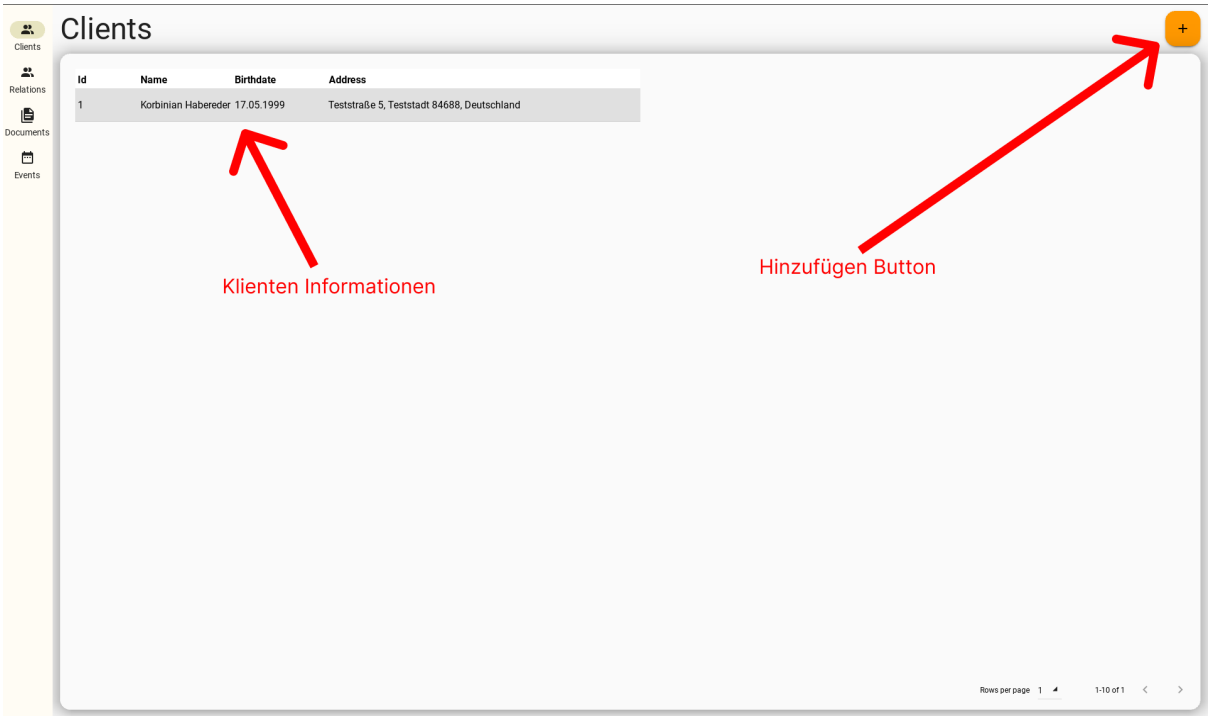
Dokumentenverwaltungs Sequenzdiagram



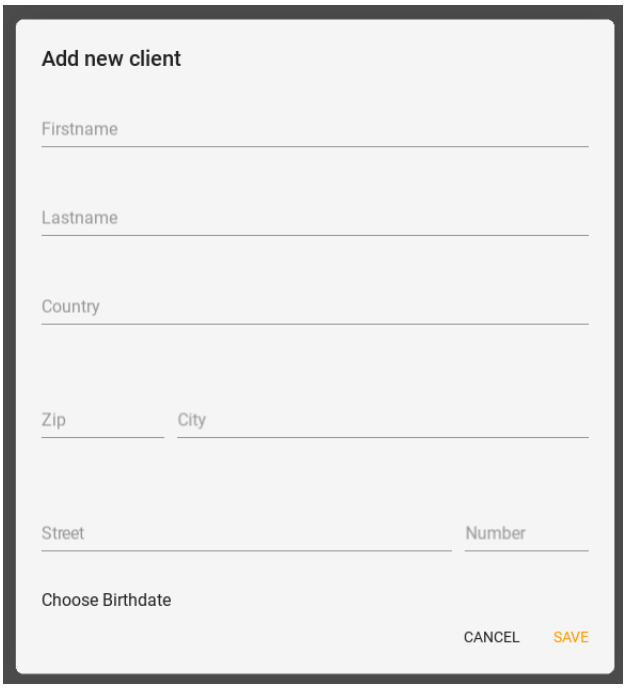
Benutzeranleitung PCC

Klientenverwaltung

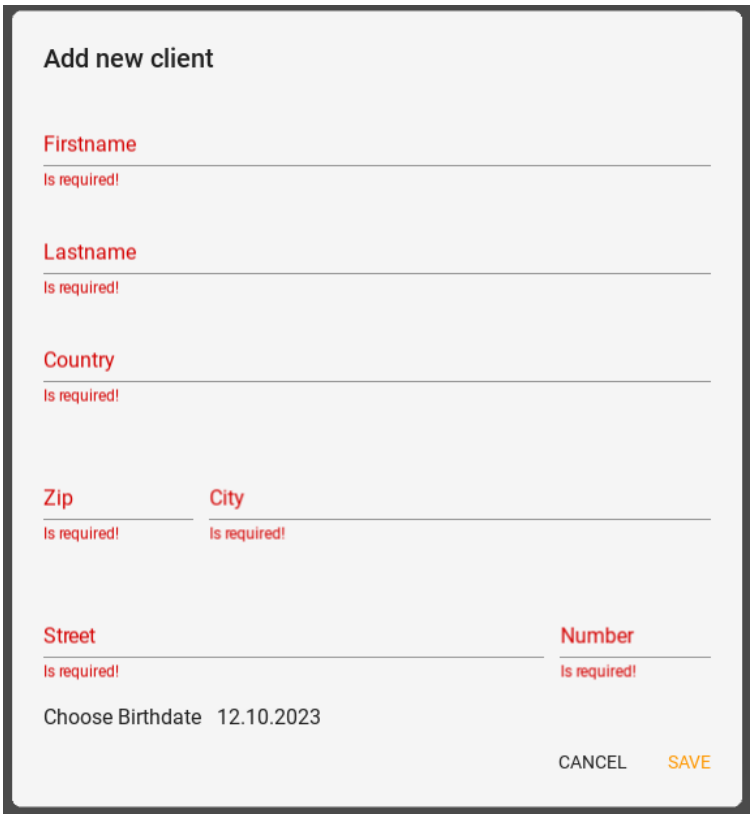
Initialer Bildschirm der Anwendung:



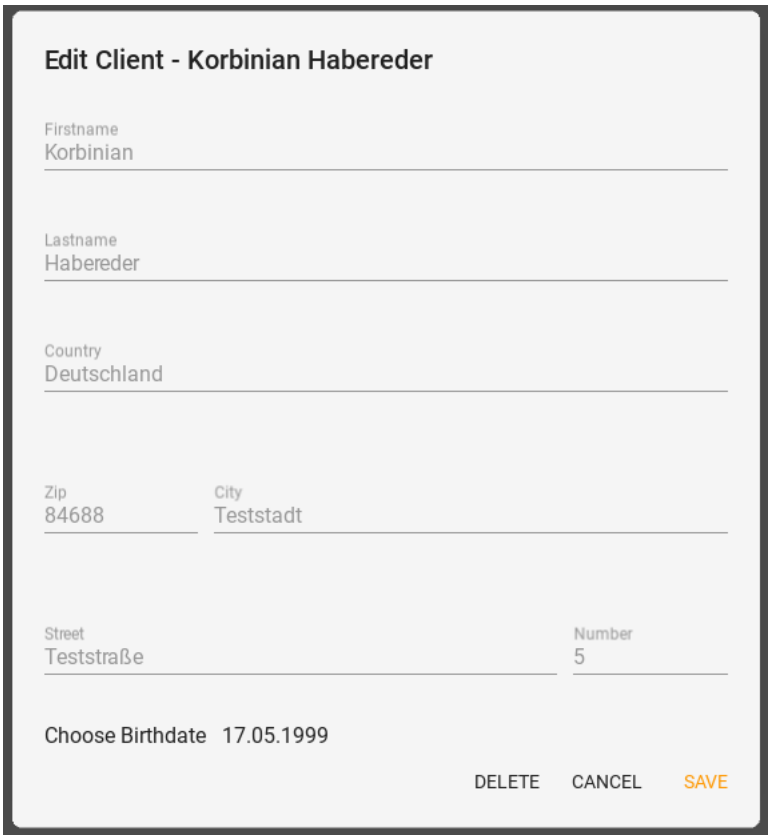
Hier können Klienten der Anwendung hinzugefügt werden. Durch klicken des Hinzufügen Buttons in der rechten oberen Ecke öffnet sich folgender Dialog:



Um einen Klienten hinzuzufügen, müssen alle Felder des Dialogs ausgefüllt werden. Werden Felder nicht ausgefüllt so werden diese entsprechend markiert:

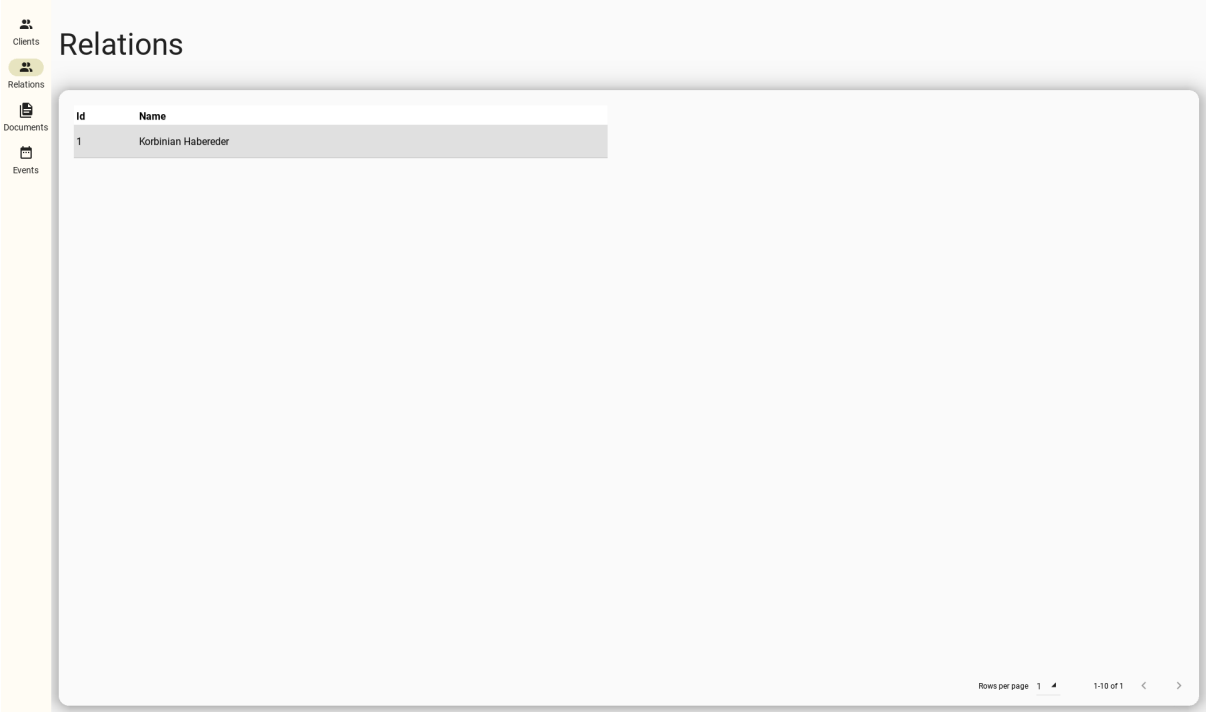


Um einen Klienten zu editieren oder zu löschen, muss dieser in der Übersichtsseite ausgewählt werden. Nach Auswahl des entsprechenden Klienten erscheint folgender Dialog:

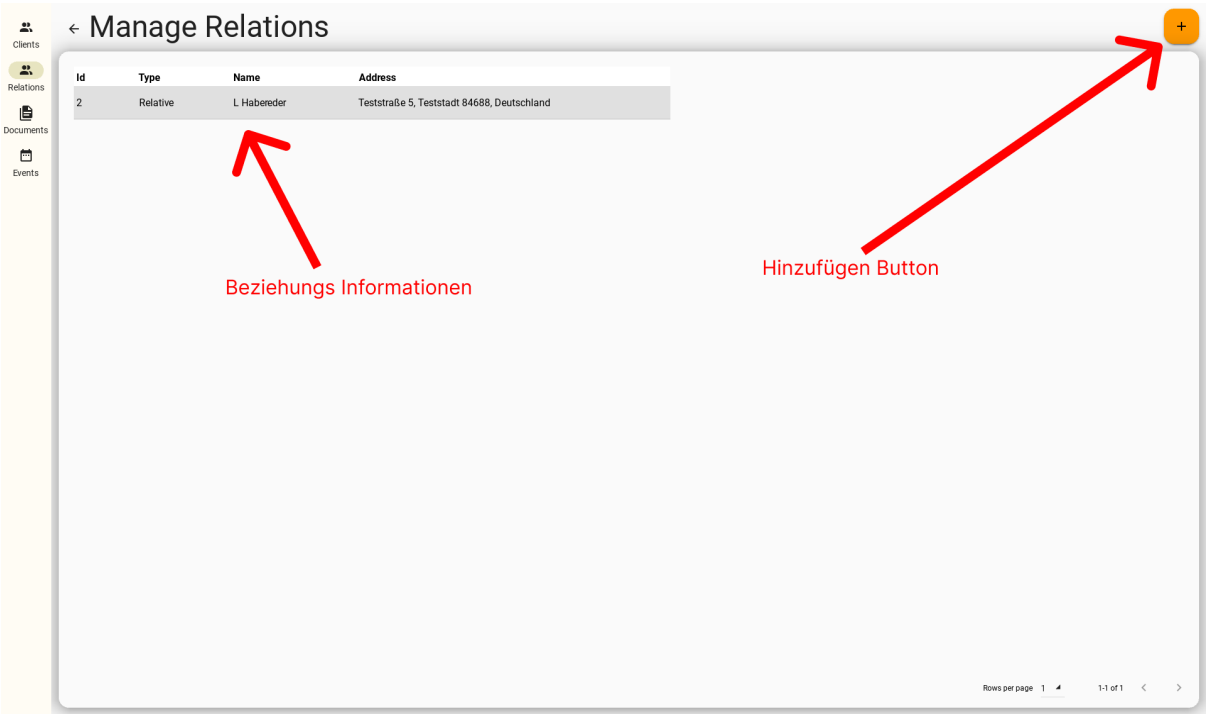


In diesem Dialog kann nun entweder der Datensatz den Klienten angepasst oder der Klient gelöscht werden. Für letzteres muss man den “DELETE” Button betätigen.

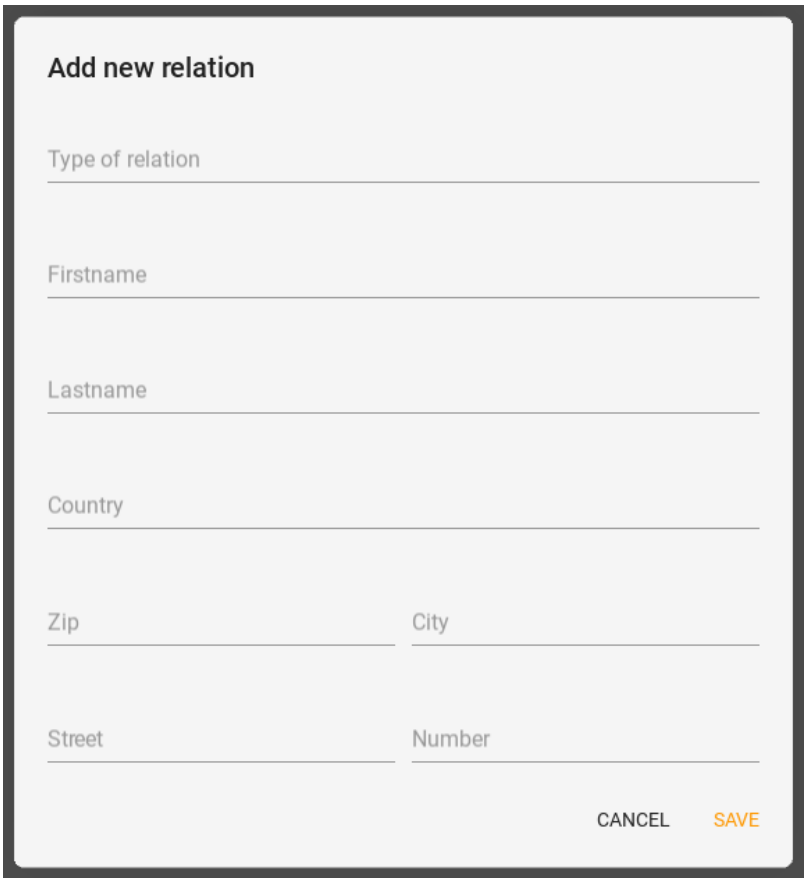
Beziehungs Verwaltung



Die obere Abbildung zeigt den Initialen Bildschirm zur Beziehungs Verwaltung. Um einem Klienten eine Beziehung hinzuzufügen, muss dieser zunächst ausgewählt werden. Folgender Bildschirm wird dann anschließend angezeigt:



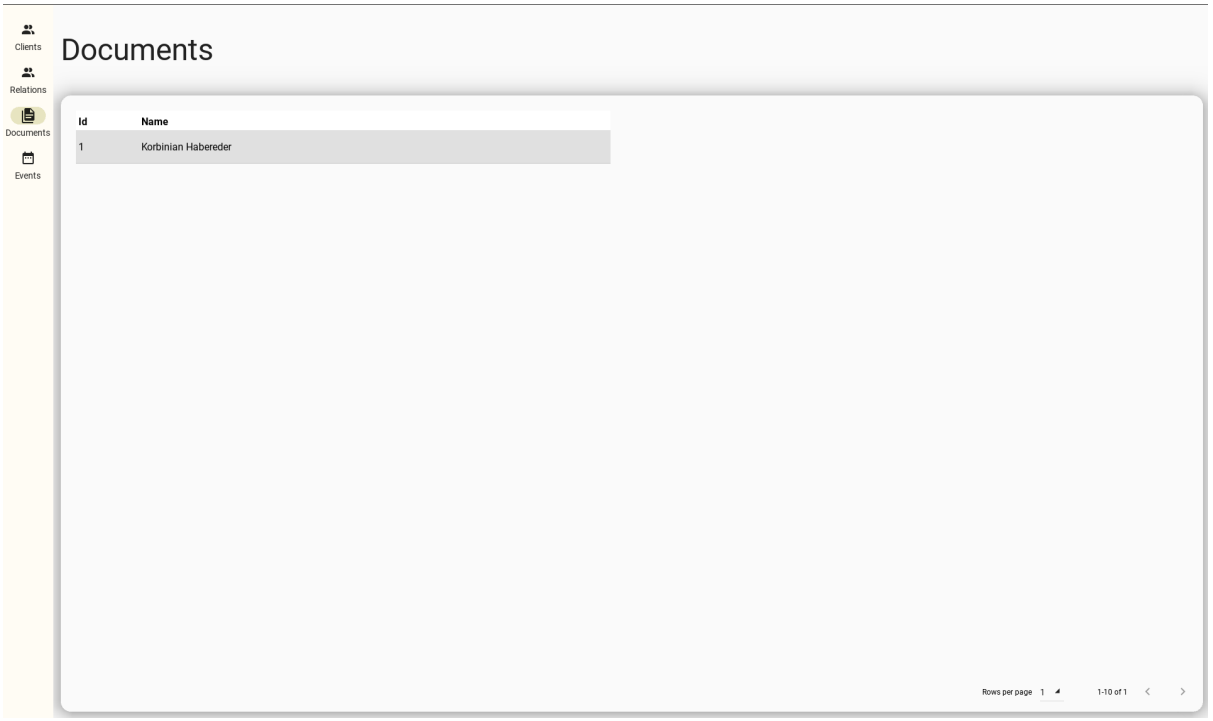
Hier kann, ähnlich wie bei der Klientenverwaltung, eine Beziehung über den Hinzufügen Button angelegt werden. Folgender Dialog ermöglicht die Eingabe der Beziehungsdaten:



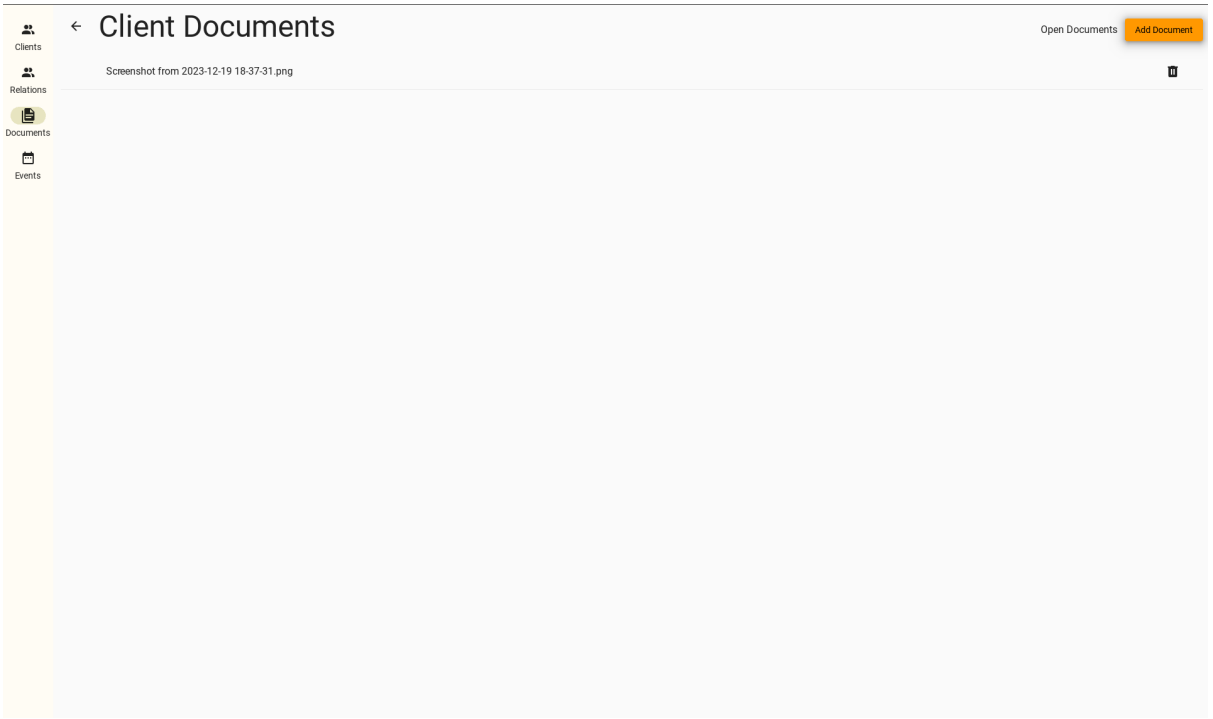
Um eine Beziehung zu verändern / zu löschen, muss diese wieder ausgewählt werden und ein Dialog zur Bearbeitung öffnet sich

Dokumentenverwaltung

Initialer Bildschirm:



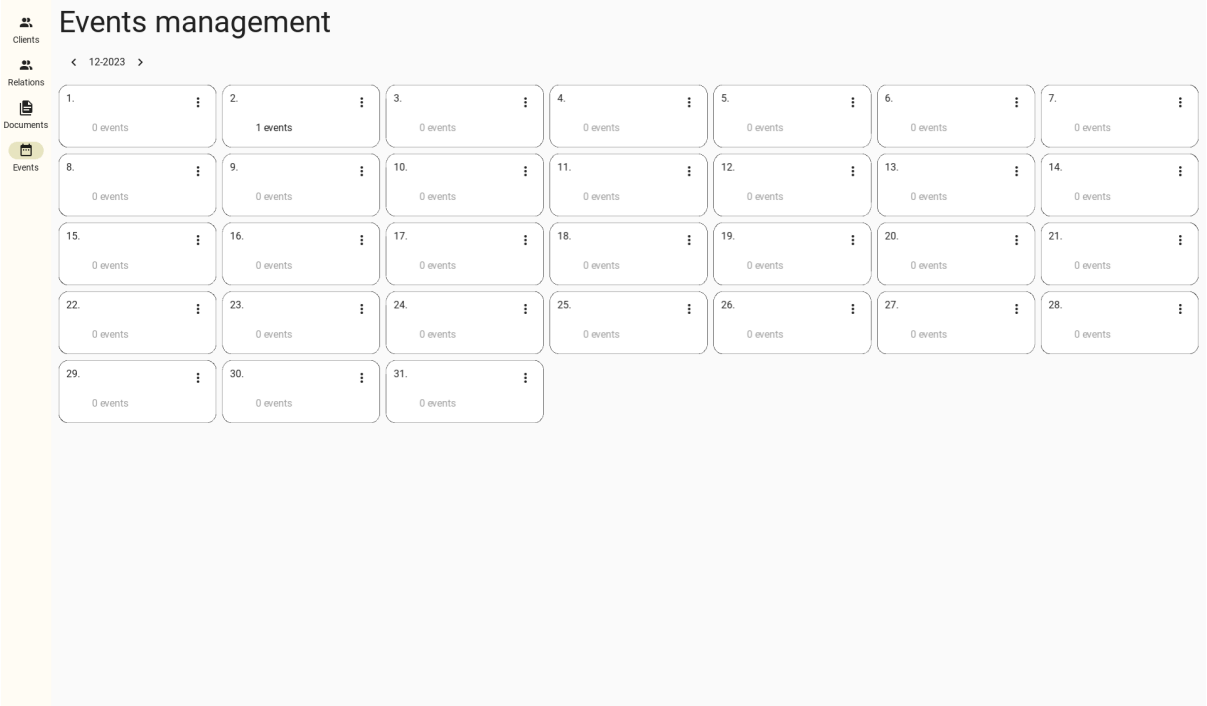
Nach Auswahl eines Klienten, für den Dokumente hinzugefügt werden sollen, erscheint folgender Bildschirm:



Dieser Bildschirm zeigt eine Übersicht der bereits hinzugefügten Dokumente, eine Löschen Funktion dieser sowie eine Hinzufügen Option. Zudem existiert eine Funktion “Open Documents", unter der Dokumente im lokalen Verzeichnis eingesehen werden können.

Terminplanung

Übersichtsseite der Terminplanung. Hier sieht man auf einen Blick, wann Termine anstehen. Durch die Buttons im linken oberen Eck kann durch Monate gewechselt werden.



Um einen Termin an einem bestimmten Datum hinzuzufügen, muss man die drei Punkte eines Tages anklicken. Man wird dann auf folgende Seite weitergeleitet:

Auch hier kann wieder durch Betätigung des Hinzufügen Buttons ein Termin angelegt werden. Um einen Termin zu bearbeiten oder zu löschen, muss dieser angeklickt werden.

Was folgenden Dialog öffnet: