MedPlanner

Web-Anwendungsentwicklung Sommersemester 2021

Egidia Cenko	Madina Kamalova	Matthias Schön	Christoph Schuster	Andrei Trukhin
Medieninformatik	*Medieninformatik*	*Medieninformatik*	*Medieninformatik*	*Medieninformatik*
e.cenko@oth-aw.de	m.kamalova@oth-aw.de	m.schoen@oth-aw.de	c.schuster1@oth-aw.de	a.trukhin@oth-aw.de

***Abstract*—Beschreibung der Software-Architektur für das Projekt *MedPlanner*.**

***Index Terms*—Termin, Selbstmanagement, Verwaltung, Web-Anwendung**

I. Einleitung

A. Motivation

In der heutigen Zeit ist alles schnelllebig und wir als Menschen müssen so flexibel wie möglich bleiben, um nicht den Überblick zu verlieren. Hierbei ist es wichtig, dass man den Fokus auf die zentralen Aufgaben legen kann ohne dadurch einen Nachteil für sich selbst zu schaffen. Wir als Menschen sind nicht unverwundbar und so kann es sein, dass durch chronische Krankheiten oder gesundheitliche Einschränkungen der Bedarf besteht, gehäuft verschiedene Arztbesuche wahrzunehmen. Aber auch Menschen ohne gesundheitliche Beschwerden sollten die Wichtigkeit der Vorsorge nicht aus den Augen verlieren: Routineuntersuchungen, Check-Ups oder einfach nur spontan auftretende Beschwerden sind Lebensaspekte, die auch trotz Schnelllebigkeit der heutigen Zeit nicht in den Hintergrund gerückt werden sollten.

B. Überblick

MedPlanner bietet die Möglichkeit, ärztliche Termine übersichtlich zu verwalten. Es handelt sich hierbei um eine Web-Anwendung, die gezielt auf das Selbstmanagement von Arztterminen abgestimmt ist, da in einem standardmäßigen Terminkalender auch weitere themenunabhängige Termine enthalten sind.

Mithilfe von *MedPlanner* können geplante Arzttermine eingetragen werden. Dies soll vor allem dazu dienen, einen Überblick über die ärztlichen Untersuchungen zu behalten, die man bereits in der Vergangenheit hatte bzw. die zukünftig noch anstehen. Dadurch soll gewährleistet werden, dass man auch als Privatperson ohne Konsultierung der verschiedenen Arztpraxen weiß, wann welche Untersuchungen durchgeführt wurden, um diese bei Bedarf erwähnen zu können.

C. Alleinstellungsmerkmal

Innerhalb der Web-Anwendung ist es möglich, Kontaktinformationen für die eigenen Ärzte abzuspeichern. So kann man zum Beispiel immer die Telefonnummer, Adresse und, falls vorhanden, die Webseite der Arztpraxis einsehen, ohne extra vor einer Terminvereinbarung immer wieder nach den nötigen Informationen zu suchen.

Je nach Projektfortschritt soll *MedPlanner* zusätzlich noch die Funktionalität bieten, Erinnerungen für das Vereinbaren von Terminen zu erhalten, sprich Reminder. So kann man sich beispielhaft vorsorglich eintragen, wann die Auffrischung einer Impfung wieder fällig ist oder in welchem Zeitintervall man eine Routineuntersuchung machen möchte.

D. technische Schlüsselbausteine

MedPlanner wird als Frontend das Framework *Angular* [2] in Kombination mit TypeScript verwenden. Für das Backend wird das Python-Framework *Django* [3] mit REST-API genutzt für die Verknüpfung mit der Datenbank. Hierzu wird die dokumentbasierte Datenbank *MongoDB* [4] genutzt.

II. Verwandte Arbeiten

MedPlanner fokussiert sich im Allgemeinen auf Folgendes: Das Selbstmanagement von Arztterminen sowie die übersichtliche Verwaltung der eigenen Ärzte.

Die Smartphone-App *DoctorBox* [5] bietet hierbei aus Patientensicht an, die eigenen Gesundheitsdaten wie z.B. Befunde ebenfalls einsehen zu können und somit Arzttermine bzw. die ärztliche Betreuung zu optimieren. Aus Ärztesicht kann DoctorBox als Desktop-App verwendet werden, um sowohl digitale Befunde zu versenden als auch vom Patienten zu erhalten. Da in diesem Zusammenhang Datenschutz nicht wegzudenken ist, wird *MedPlanner* wegen der beschränkten Projektdauer die Optimierung von Arztterminen in einer anderen Art erreichen.

In Bezug auf übersichtliche Verwaltung kann *Notion* [6] erwähnt werden: Dieses vielseitige Tool stellt Komponenten wie Notizen, Datenbanken, Kanban-Boards, Kalender und Erinnerungen bereit und kann dabei nicht nur von einem User, sondern auch im Team verwendet werden.

III. Anforderungen

Die Web-Anwendung soll für dieses Projekt nur aus Sicht des Patienten realisiert werden. In Zukunft könnte man auch eine Erweiterung aus Ärztesicht in Erwägung ziehen.

A. User Story 1

Ich als *Patient* möchte meine Termine filtern können, um einen besseren Überblick zu haben.

Akzeptanzkriterien:

Clientseitige Filterung durch:

- Fachrichtung von Arzt
- Zeitraum
- Priorität
- standardmäßig nach Fälligkeit[1] sortiert
- Ort
- Tags*

Zunächst sollen Mock-Daten im JSON-Format erzeugt werden, um die ersten Frontend-Funktionalitäten zu testen.

B. User Story 2

Ich als *Patient* möchte mithilfe des User Interfaces die Termine unterscheiden können, um einen schnellen Überblick ohne Filterung zu bekommen.

Akzeptanzkriterien:

- visuelle Unterscheidung je nach Terminart (durch Fachrichtung des Arztes)
- farblich oder mithilfe von Icons

C. User Story 3

Ich als *Patient* möchte über ein User Interface zwischen meinen ärztlichen Terminen wählen können, um die ganze Information über den Termin zu bekommen.

Akzeptanzkriterien:

- Detailansicht von einem gewählten Termin
- Termininformation ändern und löschen

D. User Story 4

Ich als *Patient* möchte meine Termine abspeichern können, um in Zukunft den Termin nicht zu verpassen.

Akzeptanzkriterien:

- Neue Termine einfügen
- Termin-Information angeben: Datum, Arzt, Adresse, Tags*

Anmerkungen:

Die Möglichkeit, eine Adresse manuell im Termin einzutragen, soll gegeben sein. Damit ist gewährleistet, dass der Patient dennoch eine Adresse im Termin nachlesen kann, auch wenn der Arzt nicht abgespeichert ist.

E. User Story 5

Ich als *Patient* möchte in Zukunft Erinnerungen bezüglich meiner Termine bekommen, um meinen Alltag besser zu planen.

Akzeptanzkriterien:

- Meldungen über nächsten Termin **24 Stunden vorher** per EMail erhalten
- Festlegung der Zeitzone: Europe/Berlin

[1]Termin, der als Nächstes ansteht, wird zuerst angezeigt

F. User Story 6

Ich als *Patient* möchte die Kontaktinformation meiner bereits eingetragenen Ärzte abrufen können, um nicht immer wieder danach suchen zu müssen.

Akzeptanzkriterien:

- Arztinformation: Name, Adresse, Telefonnummer, Fachrichtung des Arztes, ggf. Website
- Übersicht der Ärzte in der Web-Anwendung
- In der Detail-Ansicht eines Termins abrufbar*
- standardmäßig alphabetisch sortiert
- Filterung nach Fachrichtung

G. User Story 7* (optional)

Ich als *Patient* möchte mir Reminder setzen können, um daran erinnert zu werden, bei einem Arzt wieder einen Termin für eine Routineuntersuchung o.Ä. auszumachen.

Akzeptanzkriterien:

- Erinnerungs-Periode kann eingestellt werden
- Meldungen für Reminder ggf. auch per EMail verschickbar (sofern von User gewollt)
- Reminder in der App anzeigen
- Unterscheidung von einem Reminder zu einem tatsächlichen Termin

IV. Methoden

A. geplante Architektur

Es wird eine Microservice-Architektur genutzt unter Verwendung von Docker [1]. Dabei soll jede Komponente (Frontend, Backend und Database) in einem separaten Container laufen. Mithilfe von Docker Compose können alle Container in einem Netzwerk verbunden werden und so miteinander kommunizieren.

B. Mechanik der Anwendung

Der Benutzer loggt sich mit seiner EMail-Adresse und einem Passwort in MedPlanner ein. Dabei werden die Passwörter als Hashes in der Datenbank abgelegt. Dadurch erhält jeder Nutzer eine eineindeutige ID, wodurch data leaks anderer Nutzerdaten verhindert werden sollen.
Der eingeloggte Benutzer kann dann seine Termine verwalten:

- neue Termine hinzufügen
- bestehende Termine bearbeiten
- ungewollte Termine löschen

Das Hinzufügen bzw. Bearbeiten erfolgt über ein Dialogfenster. Man erhält hierbei per E-Mail die Benachrichtigung für den festgelegten Termin. Die Web-Anwendung ermöglicht zudem die Terminfilterung z.B. nach Zeitraum, Fachrichtung des Arztes. Aber auch ohne Filterung sollen Termine visuell zuordenbar sein. Ein angewählter Termin präsentiert, falls vorhanden, nähere Informationen und die Bearbeitungsmöglichkeit.
Außerdem kann der Nutzer seine eigenen Ärzte hinzufügen, indem er selbst die Kontaktinformationen einträgt sowie die Fachrichtung des Arztes.

References

[1] https://www.docker.com/
[2] https://angular.io/
[3] https://www.django-rest-framework.org/
[4] https://www.mongodb.com/de
[5] https://www.doctorbox.de/patienten.jsp
[6] https://www.notion.so/