TAU INF202 Software Engineering
Individuelles Projekt
Pflichtenheft

Projektdokumentation
Version: 2023.v1.0
Status: Entwurf

Klinikverwaltungssystem

Verantwortliche/r:
İpek Yılmaz, e200503052@stud.tau.edu.tr
Haluk Harun Gündoğan, e200503031@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v1.0	Entwurf	25.02.2023	İpek Yılmaz Haluk Harun Gündoğan	Use Cases wurden auf Wunsch der Beraterin erweitert.

**) Sofern im Projekt nicht anders vereinbart, sind folgende Statusbezeichnungen zu verwenden (in obiger Tabelle und am Deckblatt):*
Dokument-Status: Entwurf / in Review / freigegeben (abgegeben)

Dokument wurde mit folgenden Tools erstellt.

Google Documentation

Lucidchart

Inhaltsverzeichnis

1. Einleitung

In diesem Dokument werden die Anforderungen während der Softwareentwicklungsphasen von der Applikation ,,Klinikverwaltungssystem” vollständig und konsistent erklärt.

Dieses System wird bei den Sekretärinnen in der Klinik benutzt, um den Patienten die passenden Termine zu setzen und die Patienten einzufügen.

Im Kapitel 2 „Ausgangssituation und Ziele“ sind die Ausgangssituation und der Grund zur Wahl von Klinikverwaltungssystem anschaulich dargestellt.

Im Kapitel 3 „Gesamtarchitektur“ sind die physikalische und die konzeptionelle Architektur des Systems und die wichtigsten Subsysteme (Komponenten), die Anwender und die notwendigen Kommunikationsschnittstellen dargestellt. Hier sind auch zusätzliche Anforderungen an die Architektur oder Komponenten zu finden.

Im Kapitel 4 „Funktionale Anforderungen“ beinhaltet die Beschreibung der funktionalen Anforderungen durch die Ablaufbeschreibungen (User Stories), die Anwendungsfällen (Use Cases), und technischen und fachlichen Anforderungen (Requirements). Alle betriebsrelevanten Daten werden durch die Datenmodellen definiert.

Im Kapitel 5 „Nichtfunktionale Anforderungen“ sind die funktionalen Anforderungen durch diejenigen Anforderungen erweitert, die keine funktionalen Anforderungen sind.

Im Kapitel 6 „Abnahmekriterien“ sind die Abgabeartefakten festgelegt, die ohne Abstimmung des Stakeholders nicht zu manipulieren sind.

Im Kapitel 7 „Projekt Meilensteine“ sind die wichtigsten Termine aufgelistet, die den Fortschritt der Teilergebnisse des Projektes definieren.

2. Ausgangssituation und Ziele

Einleitung

Das Klinikverwaltungssystem dient dazu, die Unstimmigkeiten und Komplikationen bei der Terminerstellung zu verhindern, bevor sie sich ergeben. Das System wird vereinfacht, die Patienten- und Termininformationen zu verwalten und sicher zu speichern.

Problemstellung (Funktionalität)

Die Kompliziertheit und Herausforderungen sind nicht vermeidbar, wenn die Patienten und Termininformationen in einer Klinik separat und nicht digital organisiert sind. Diese Daten auf physikalische Quelle wie Heften zu speichern, ist in einer Zeitperiode kostenträchtig, schwierig und nicht nachhaltig.

Mit der Implementierung dieses Klinikverwaltungssystems wird die Verwaltung von Daten in einer Klinik simplifiziert. Die Komplikationen und Kollisionen werden vermieden, die während der Erstellung eines Termins entstehen.

Stakeholder (Anwender)

Die am meisten profitierte Zielgruppe von diesem System sind die Kliniken. Das Klinikverwaltungssystem wird in den Computern in einer Klinik implementiert und von den Sekretärinnen in den Kliniken benutzt.

Systemumfeld (Einsatzumgebung)

Am Mittelpunkt von diesem System steht die Verwaltung und Speicherung von Daten digital, deshalb wird das ganze System per Software realisiert und soll auf einem Computer laufen.

Rahmenbedingung (Einschränkungen)

Es gibt einige Einschränkungen in der Wahl der Software Tools, die während der Erstellung vom System benutzt werden:

- Als Software-Entwicklungstool soll entweder Eclipse oder IntelliJ verwendet werden,
- die Backend Applikationen sollen mit Java Framework und Spring Framework,
- die Frontend Applikationen mit React realisiert werden, und
- die persistenten Daten sollen in einer SQL-Datenbank abgespeichert werden.

Ziele (Lösung)

Es wird ein System entwickelt, das den Termin setzt, Patienten einfügt, Termine und Patienten listet.

Es soll auch möglich sein für Benutzer, diese Aktionen über eine Schnittstelle zu ergreifen und zu sehen.

Auf diese Weiße werden Patientendaten und Terminen digital gespeichert und die Verwaltung von diesen Daten erleichtert.

Außerdem wird es nicht möglich sein, dass der gleiche Termin bei den verschiedenen Patienten genommen wird. Damit werden die Unstimmigkeiten und Komplikationen verhindert.

Die Dienste sollen mit Hilfe von RESTful Services (API) erstellt werden. Die Verwaltung und Speicherung von Patienten- und Termininformationen sollen mit einem Datenbanksystem erfolgen.

3. Gesamtarchitektur

Einleitung

In diesem Kapitel werden sowohl die Systemarchitektur detailliert mit allen ihren Bestandteilen belegt als auch die Gesamtarchitektur mit der Unterstützung von Visualisierungen dargestellt. Um die Grenzen genauer definieren zu können, wird dieses Kapitel abgeleitet.

Gesamtarchitektur

Die Gesamtarchitektur in den folgenden Aspekten in den Betracht gezogen:

- Konzeptualisierung der Klinikverwaltungssystem mit der Kontextdefinierungen
- Systemarchitektur um eine Realisierung für die Kundenanforderungen beschaffen zu können.
- Komponentenarchitektur mit der Spezifikationen der Einzelkomponenten

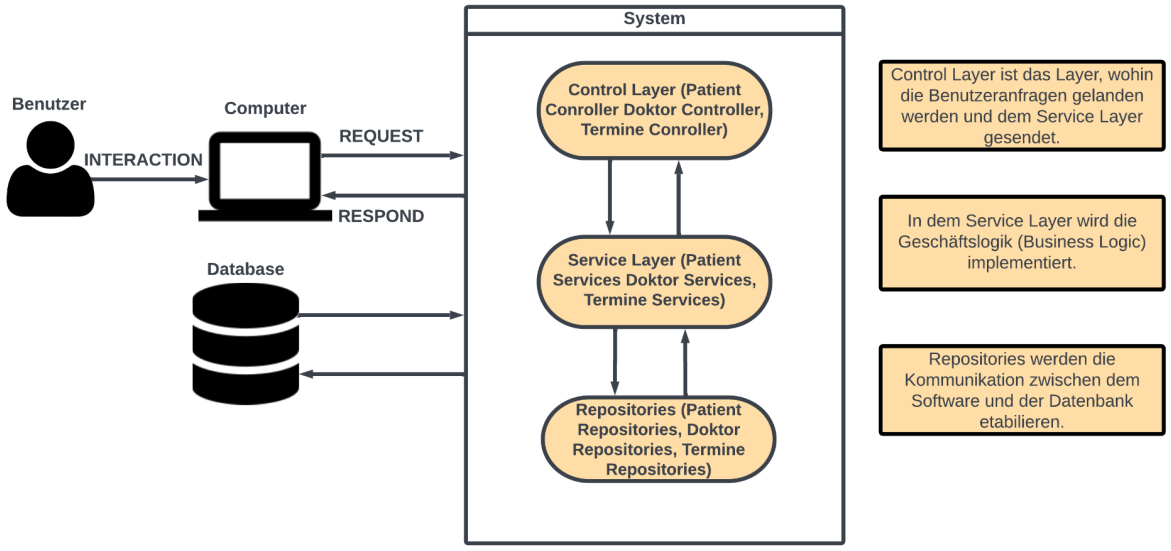

Abbildung 1: Klinikverwaltungssystem konzeptuelle Architektur

Es ist essentiell für uns, die Systemgrenzen zu definieren mit dem Umfang von Kundenanforderungen. In der Abbildung ist definiert, wie sich das System verhält, wenn es den Benutzeranfragen begegnet. Es wurden sowohl die externen Schnittstellen als auch die internen Komponenten dargestellt.

Komponente <Patient>

Komponente Patient trägt die Verantwortung über die folgenden Anforderungen:

- Die Registrierung von Patienten und die persistente Speicherung der Patienteauskünfte
- Auflisten der registrierten Patienten.

Komponente <Doktor>

Komponente Doktor ist die Systemeinheit, wo sich die Doktorinformationen befinden.

Komponente <Termine>

Feste Verantwortung des Termin Komponentes:

- Das Setzen von dem Termin soll ausschließlich über das System erfolgen.
- Aktuelle Termine mit den registrierten Doktoren sollen aufgelistet und dargestellt werden.

Externe Schnittstellen

Die externe Schnittstellen sind:

- **Home-Schnittstelle:** Der Benutzer sieht diese Schnittstelle am Anfang.
- **Patiente Einfügen-Schnittstelle:** Die Schnittstelle zum Einfügen des Patienten.
- **Termine Erstellungs-Schnittstelle:** Die Schnittstelle zum Erstellen des Termins.
- **Termine listen-Schnittstelle:** Die Schnittstelle zum Listen der Termine.

Für die Definition der Schnittstelle siehe die Abbildung 2 “UI Home Page”, Abbildung 3 “Patiente hinzufügen”, Abbildung 4 “ Termine erstellen”, Abbildung 5 “Termine listen” und das Kapitel 4 ,,Funktionale Anforderungen”.

4. Funktionale Anforderungen

Einleitung

In diesem Teil werden die Systemanforderungen, Use Cases und einzelne Systemkomponenten definiert.

UI Use Cases

- **/UI-1/** Auf der Oberfläche dieses Systems sollen die Systemfunktionen in einem Menü entstehen. Am Anfang steht ein Home Page.

Abbildung 2: UI Home Page

- **/UI-2/** Auf der Oberfläche sollen die Patienteninformationen angenommen und in einem Bereich gelistet werden.

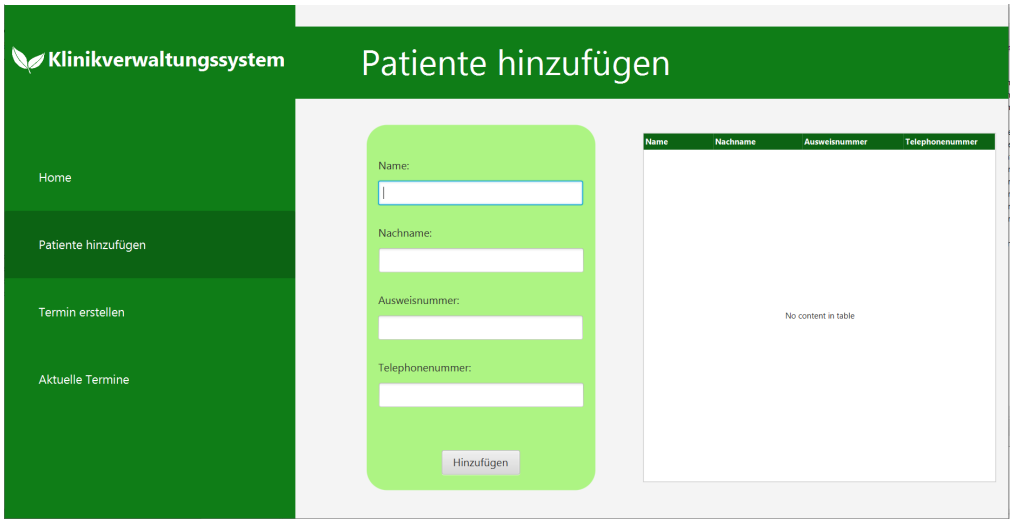

Abbildung 3: Patiente hinzufügen

- **/UI-3/** Das System soll fähig sein, Termine zu setzen, indem man auf der Oberfläche die gewünschten Informationen auswählt.

Abbildung 4: Termine erstellen

- **/UI-4/** Über die Oberfläche soll jeder Termin gelistet werden.

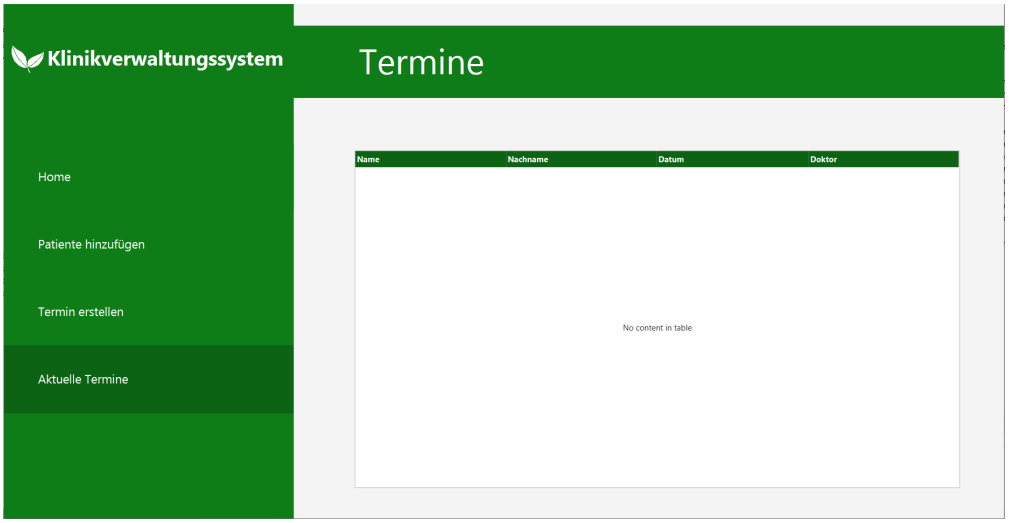

Abbildung 5: Termine listen

API Use Cases

/API-1/ Bei der Registrierung eines neuen Patienten wird diese Schnittstelle benutzt. In dieser Schnittstelle soll der Endbenutzer die Patienteninformationen wie Ausweisnummer, Name, Nachname und Telefonnummer eingeben. (Diese Informationen können im Laufe der Zeit aufgrund der Diskussionen mit den Stakeholdern geändert oder aktualisiert werden.)

Use Case	**Patiente einfügen**	
Geschichte: Dieses Use Case wird benutzt, wenn der Benutzer einen neuen Patient erstellen möchte. Die Informationen werden eingegeben, dann wird der Patient erstellt.		
Schritt	**Aktor**	**Aktion**
1	Benutzer	Gibt den Informationen von Patienten ein. Gibt den Befehl ,,Patiente einfügen”
2	System	Kontrolliert ob der Patient schon im System gespeichert ist
3b	System	Sendet eine Nachricht zu Benutzer über die Existenz von Patient falls der Patient schon existiert
3b	System	Erstellt einen neuen Patient falls der Patient nicht existiert

/API-2/ Um das Auflisten der Patienten visualisieren zu können, wird diese Schnittstelle benutzt. Der Benutzer bekommt Informationen über die Patienten.

Use Case	**Patiente auflisten**	
Geschichte: Dieses Use Case wird benutzt, wenn der Benutzer die existierenden Patienten sehen möchte.		
Schritt	**Aktor**	**Aktion**
1	Benutzer	Gibt den Befehl ,,Patiente auflisten”.
2	System	Gibt die alle existierenden Patienten zurück.

/API-3/ Der Terminerstellungprozess wird mit der Nutzung von dieser Schnittstelle realisiert. Bei der Terminerstellung wird der Benutzer einen Doktor und Termindatum mit der gegebenen Zeit auswählen und für den gewünschten Patient den Termin erstellen.

Use Case	**Termine erstellen**	
Geschichte: Dieses Use Case wird benutzt, wenn der Benutzer einen neuen Termin erstellen möchte. Die Informationen werden eingegeben, dann wird der Termin erstellt.		
Schritt	**Aktor**	**Aktion**
1	Benutzer	Gibt den Informationen von dem Termin ein. Gibt den Befehl ,,Termine erstellen”
2	System	Nimmt die Informationen. Kontrolliert die Verfügbarkeit vom Datum.
3a	System	Sendet Nachricht, falls das Datum nicht verfügbar ist.
3b	System	Falls das Datum verfügbar ist, kontrolliert ob der Patient existiert
3b.1	System	Sendet Nachricht, falls der Patient nicht existiert.
3b.2	System	Erstellt den Termin, falls der Patient existiert.

/API-4/ Um das Auflisten der Terminen visualisieren zu können, wird diese Schnittstelle benutzt. Der Benutzer bekommt Informationen über die Termine.

Use Case	**Termine Auflisten**	
Geschichte: Dieses Use Case wird benutzt, wenn der Benutzer einen neuen Patient erstellen möchte. Die Informationen werden eingegeben, dann wird der Patient erstellt.		
Schritt	**Aktor**	**Aktion**
1	Benutzer	Gibt den Befehl ,,Termine Auflisten”.
2	System	Gibt alle aktuellen Termine zurück.

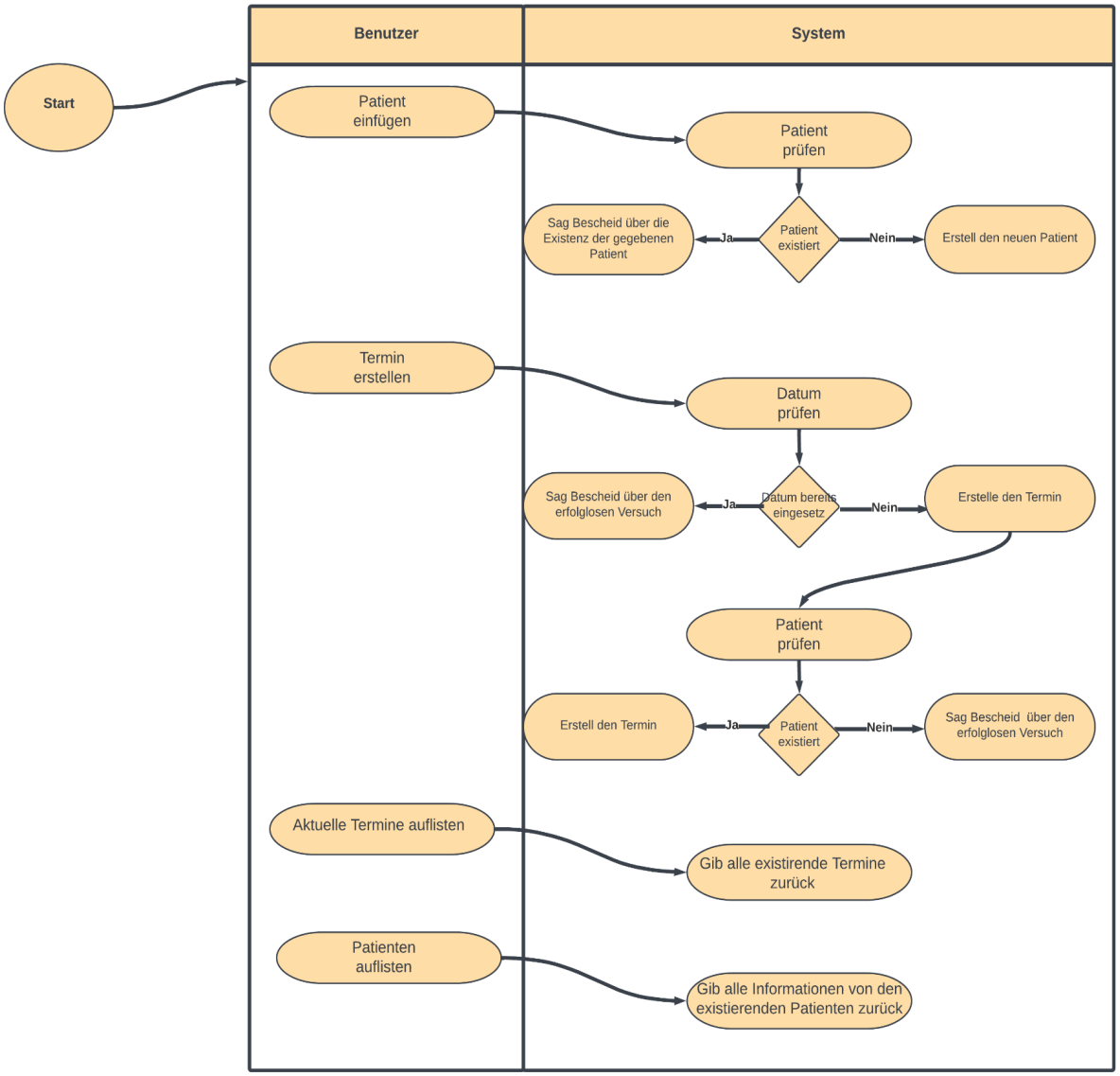

Abbildung 6: Systemverhaltensdiagramm des Klinikverwaltungssystems

Technischen und fachliche Anforderungen

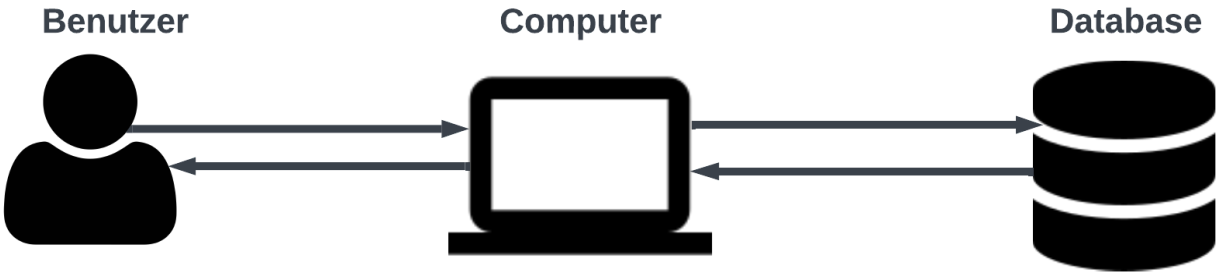

Abbildung 7: Physikalische Systemarchitektur

- **/SYS-1/** Das System soll eine dreiteilige Architektur besitzen. Diese drei Bausteinen sind Benutzer, Computer und Datenbank.

- **/SYS-2/** Der Benutzer gibt Befehle aus und gibt Informationen ein.

- **/SYS-3/** Der Computer beschafft eine Schnittstelle zwischen dem Benutzer und der Datenbank. Das macht das systematische Zugreifen der Benutzer über die Daten möglich.

- **/SYS-4/** Der Datenbank kommuniziert mit dem Computer und soll die Informationen speichern.

- **/SYS-5/** Jede Dienste soll für die persistenten Daten die DBMS nutzen.

Datenmodell

/DAT-1/ Die relevanten Systemparameter sind in einer externen Datenbank abzuspeichern.
- Die Liste der Parameter wird später durch den Stakeholder zur Verfügung gestellt.

/DAT-2/ Ein externes DBMS (eine SQL-Datenbank) ist für die Persistenz der Daten notwendig. Es kann ein Entity/Relationship Diagramm erstellt werden.

/DAT-3/ Das DBMS soll zum Speichern der Daten per Komponente gewählt und eingerichtet sein.

/DAT-4/ Das Datenmodell soll die Parameter der API-Schnittstellen berücksichtigen.

/DAT-5/ Das Datenmodell soll im UML Klassendiagramm modelliert werden.

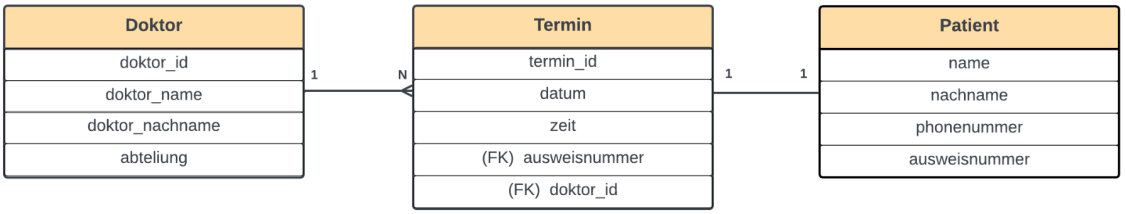

Abbildung 8: Datenmodell UML

5. Nichtfunktionale Anforderungen

Einleitung

Dieses Kapitel macht die nicht funktionalen Anforderungen an das Gesamtsystem bekanntlich sowie an die einzelnen Komponenten des Systems.

Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)

- **/SYS-1/** Die Klinikverwaltungsystem wird auf dem lokalen Host in einem Fenster ausgeführt. Sobald das System bereitgestellt ist, kann jede einzelne Komponente des Systems unter Berücksichtigung der Kundenanforderungen miteinander kommunizieren.

- **/SYS-2/** Diese Kommunikation basiert auf der REST-Architektur und der Interaktion den RESTful Webservices.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung

- **/DEV-1/** Die Entwicklungsumgebung ist frei wählbar!

Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)

- **/TOL-1/** Die ganze Backend des Systems wird mit Spring Boot implementiert.

- **/TOL-2/** Die Frontend Applikation wird mit React realisiert.

- **/TOL-3/** Die Datenpersistenz wird mit einer SQL-basierten RDBMS verwirklicht.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

- **/TEST-1/** Jeder System-Service soll mit der Black-Box-Strategie getestet werden.

- **/TEST-2/** Alle Use Cases, User Stories und Anforderungen sollen getestet und berichtet werden.

6. Abnahmekriterien

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: INF202-Klinikverwaltungssystem-2023.v1.0.docx
- Software
  - Link zu GitHub Projekt: https://github.com/orgs/Gruppe1-Fulya/teams/h-i
- Evidenz:
  - System/Software-Demo via Videoclip: https://drive.google.com/file/d/149-dtkGKo09fpYV2G7SjBUdFtBFxbe1N/view?usp=share_link

Anm.: Die Abgabetermine der Projetartefakts werden durch den Stakeholder festgelegt!

7. Projekt Meilensteine

Meilenstein e#1 (Zwischenabgabe):
- Der Lastenheft-Entwurf ist fertiggestellt, um mit dem Stakeholder abzustimmen.

Meilenstein M#1:
- Das Lastenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.

Meilenstein e#2 (Zwischenabgabe):
- Der Pflichtenheft-Entwurf ist fertiggestellt, um mit dem Stakeholder abzustimmen.

Meilenstein M#2:
- Das Pflichtenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.

Meilenstein M#3:
- In diesem Meilenstein ist die Architektur im Vordergrund.
- Komponenten-basierte Architektur ist entworfen und komponentenweise implementiert.
- Die externen Schnittstellen (Webservices) wurden entworfen/implementiert.
- Die GUI Komponente ist ansatzweise fertig.

Meilenstein M#4:
- In diesem Meilenstein ist das Testen im Vordergrund.
- Die Test-Cases wurden aus den Use Cases abgeleitet und ansatzweise beschrieben.
- Die Testumgebung ist vorbereitet und ein Smoke Test ist durchgeführt.

Meilenstein M#5:
- Das Projekt ist per Vereinbarung abgegeben.