

*TAU INF202 Software Engineering*  
*Individuelles Projekt*  
**Pflichtenheft**

Projektdokumentation

Version: 2023.v1.0

Status: freigegeben

Krankenhaus-Terminsystem

Verantwortliche/r:

Kaan İmamoğlu, e200503011@stud.tau.edu.tr

Ahmet Oğuz Örsler, e200503035@stud.tau.edu.tr

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

## Dokumentenverwaltung

### Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v1.0	Entwurf	07.04.2023	Kaan İmamoğlu, Ahmet Oğuz Örsler	Vorlage wurde für die Studentenprojekte freigegeben
v1.0	freigegeben	17.04.2023	Kaan İmamoğlu, Ahmet Oğuz Örsler	Vorlage wurde für die Studentenprojekte freigegeben

*\*) 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:

Microsoft Office Word

Google Docs

LucidChart



## Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>4</b>
<b>2. Ausgangssituation und Ziele</b>	<b>5</b>
<b>3. Gesamtarchitektur</b>	<b>7</b>
<b>4. Funktionale Anforderungen</b>	<b>10</b>
<b>5. Nichtfunktionale Anforderungen</b>	<b>14</b>
<b>6. Abnahmekriterien</b>	<b>15</b>
<b>7. Projekt Meilensteine</b>	<b>17</b>
<b>8. Referenzen</b>	<b>18</b>

## 1. Einleitung

Dieses Dokument dient dazu, die verbindlichen Anforderungen an das Terminsystem eines Krankenhauses zu definieren und vollständig und einheitlich zu erläutern und zu beschreiben. Dieses Projekt ist ein nützliches Terminsystem, das sowohl von Ärzten als auch von Patienten verwendet werden kann.

Die Use Cases und Anforderungen sind aus der Sicht des Stakeholders beschrieben.

Die graphische Oberfläche zur Überwachung des Krankenhaus Terminsystem ist aus der Sicht eines Patienten und eines Arztes beschrieben.

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

Kapitel 3 „Gesamtarchitektur“ werden die physikalische und konzeptionelle Architektur des Systems, die wichtigsten Teilsysteme (Komponenten), die Benutzer und die notwendigen Kommunikationsschnittstellen beschrieben. Auch weitere Anforderungen an die Architektur oder Komponenten finden Sie hier.

Im Kapitel 4 „Funktionale Anforderungen“ werden die funktionalen Anforderungen durch die Prozess- und User Stories, die Use Cases und die fachlichen Anforderungen beschrieben. Auch das Datenmodell definiert alle betriebsrelevanten Daten.

Im Kapitel 5 „Nichtfunktionale Anforderungen“ haben auch die funktionalen Anforderungen diejenigen Anforderungen, die keine funktionalen Anforderungen sind.

Im Kapitel 6 „Abnahmekriterien“ beschreibt die Abgabeartefakten, die nur mit Zustimmung des Stakeholders manipuliert werden können.

Im Kapitel 7 „Projekt Meilensteine“ sind die wichtigen Termine zum Fortschritt und Teilergebnisse des Projekts enthalten.

Im Kapitel 8 „Referenzen“ sind die wichtigsten Referenzen zum Projekt aufgeführt.

## 2. Ausgangssituation und Ziele

### Einleitung

Das gewählte Thema und Projekt zielen darauf ab, die Belastung des Krankenhauses und der Mitarbeiter im Krankenhaus zu minimieren und einen Online-Termin mit dem für die im System registrierten Patienten geeigneten Arzt zu vereinbaren.

Das Pflichtenheft umfasst den Prozess der Kontaktaufnahme mit den Ärzten in der Datenbank der im System registrierten Patienten und die Vereinbarung eines Termins mit einer Art Programm innerhalb der entsprechenden Zeitzone.

Dieses Pflichtenheft basiert auf dem ersten Entwurf der Kundenvorstellungen, die im „Krankenhaus\_Terminsystem.pptx“ präsentiert wurden. Dieses Präsentationsmaterial soll als das **Lastenheft** verstanden werden.

### Problemstellung (Funktionalität)

Krankenhäuser sind komplexe Unternehmen, in denen mehrere Funktionen zusammengeführt werden. Das Krankenhaus ist zunächst ein Unternehmen, dann eine Einrichtung, die medizinische Dienstleistungen erbringt.

In diesem komplexen Gefüge kann der Einsatz von Computern viele Probleme lösen und nicht nur Unternehmer, sondern auch Arbeitnehmer entlasten.

Elektronische Krankenhausinformationssysteme wurden in den letzten 20 Jahren parallel zur Entwicklung in anderen Sektoren weit verbreitet eingesetzt. In Abhängigkeit von den Fortschritten in den Informationstechnologien haben sich in dieser Zeit auch die Informationssysteme stark weiterentwickelt.

Der Weg zu einer besseren Gesundheitsversorgung besteht darin, Informationen über den Zustand des Patienten mit Informationen über die Ressourcen und das Potenzial des Krankenhauses zu integrieren. Es hängt von der Gestaltung des Informationssystems mit der bereitzustellenden Architektur ab.



Abbildung 1 Traditionelle Terminsystem

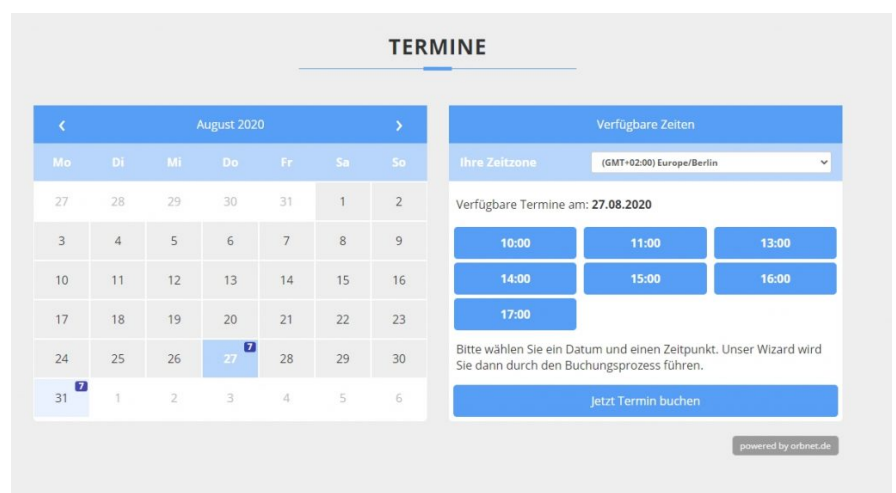


Abbildung 2 Online Terminsystem

Einer der größten und wichtigsten Vorteile des Online-Terminsystem ist, dass der Patient bekommt, was er will, bevor er das Krankenhaus aufsucht oder mit dem Arzt spricht, ohne Zeit zu verlieren. Darüber hinaus kommuniziert der Arzt nicht mit den Patienten und konzentriert sich nur auf seine eigene Arbeit.

Um ein intelligentes Terminsystem zu bilden, braucht es intelligente Funktionseinheiten, die alle Systemteilnehmer kontinuierlich und bedarfsgerecht mit präzisen Echtzeit-Informationen versorgen.

### **Stakeholder (Anwender):**

Das neue System richtet sich an alle Patienten, die eine Termin nehmen und ins Krankenhaus gehen, und alle Ärzte, die in diesem Krankenhaus arbeiten, als Anwender.

### **Systemumfeld (Einsatzumgebung)**

Dies ist ein Terminsystem, aus diesem Grund muss die Software für beide Seiten online zugänglich sein. Es wird auf dem Computer ausgeführt.

### **Rahmenbedingung (Einschränkungen)**

Die Hauptpräferenzen für die Technologie des Projekts:

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

### **Ziele (Lösung)**

Mit dem neuen System wird es für Patienten einfacher, einen Termin im Krankenhaus zu vereinbaren und sie zu sehen, und es wird auch für Ärzte einfacher sein, die Termine zu sehen, die sie haben.

Use Cases werden mithilfe von Webdiensten behandelt, um eine effiziente und skalierbare Kommunikation zwischen Systemkomponenten sicherzustellen. Durch die Verwendung von Webdiensten können Anwendungsfälle unabhängig voneinander entwickelt, getestet und bereitgestellt werden, wodurch das System modularer und wartbarer wird.

Die Persistenz soll mit einem Datenbanksystem erfolgen, um eine sichere und zuverlässige Speicherung der Daten zu gewährleisten. Durch die Verwendung eines Datenbanksystems können die Daten effizient gespeichert, abgerufen und aktualisiert werden.

### 3. Gesamtarchitektur

#### Einleitung

In diesem Kapitel werden die Systemgrenzen definiert, die mit entsprechenden Illustrationen spezifiziert werden. Die externen Schnittstellen (grafische Oberflächen und API) werden vom Benutzer unter Berücksichtigung der Systemgrenzen definiert. Die notwendigen Systemkomponenten und Datenstrukturen werden definiert und modelliert.

Dieses Kapitel führt zu einem besseren Verständnis des geforderten Systems und damit zu einer genauen Definition der funktionalen und nicht-funktionalen Anforderungen.

#### Gesamtarchitektur

Die Gesamtarchitektur sind in den folgenden Aspekten betrachtet:

- Identische Zugangsdaten von Patienten und Ärzte, in das System einzuloggen,
- Verschiedene Schnittstellen und Zugriffsberechtigungen für Ärzte, Patienten und den Chefarzt,
- Ein Diagramm, das die Hauptfunktionen (die Beziehung zwischen Benutzer und Datenbank) des Systems zeigt und wie das System funktioniert.

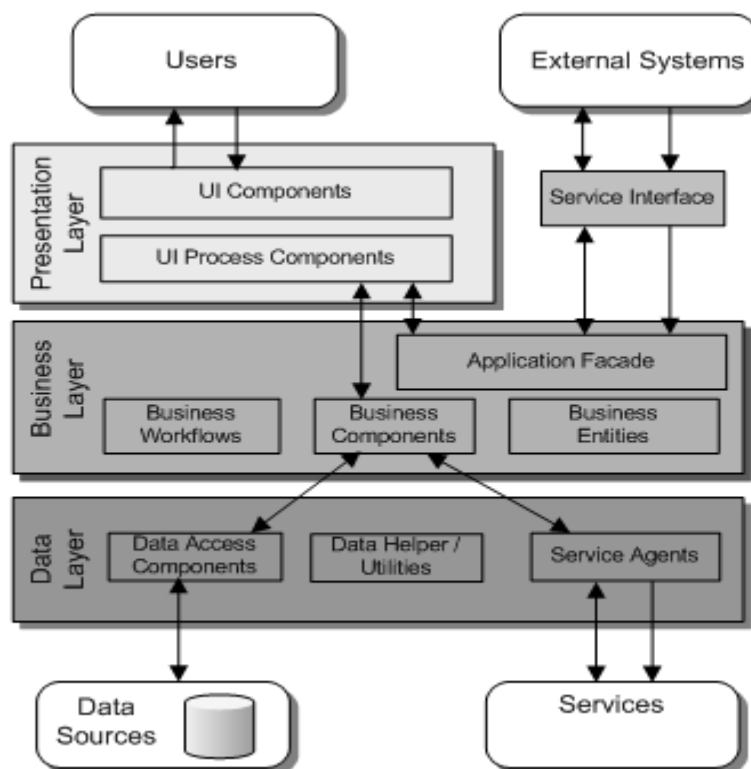


Abbildung 3 User-System Verbindung

Um die Funktionsweise des neuen Systems zu verstehen, ist es wichtig, die Grenzen zu kennen, die das Verhalten verschiedener Benutzer einschränken und die Benutzer durch das System führen. Abbildung - 8 UML-Diagramm in Kapitel 4 zeigt die Benutzergrenzen und welche Aktionen sie mit verschiedenen Schnittstellen ausführen können, die vom Typ des Benutzers bestimmt werden.

Die Abbildung 8 - UML-Diagramm in Kapitel 4 zeigt die Komponenten des Systems, die für die Anforderungen zu berücksichtigen sind.



### **Komponente (Login GUI - Register GUI)**

Nach dem Ausführen des Programms wird eine Anmeldeseite angezeigt. Diese Login-Seite hat zwei verschiedene Logins:

- Patienten-Login
- Arzt-Login.

Auf dieser Anmeldeseite werden bestimmte Informationen für Patienten und Ärzte abgefragt und ein Login angefordert. Personen, die sich anmelden, können die Orte sehen, die mit ihrer Schnittstelle (GUI) verbunden sind.

Willkommen beim  
Krankenhausverwaltungssystem

Patient Login    Arzt Login

ID-Nummer:

Password:

Registrierung    Log in

Abbildung 4 Login GUI

Patienten müssen sich zunächst im System registrieren. Dazu müssen sie auf die " für Registrierung " klicken und die erforderlichen Informationen eingeben. Die Patienten werden nach ihrem Namen, Nachnamen und ihrer ID-Nummer gefragt, wonach die Person ein eindeutiges Passwort festlegen muss. Die Daten und das Passwort der Person werden in unserer Datenbank gespeichert.

Wenn sich der Patient anmelden möchte, kann er sich einfach mit seinen eigenen Identitätsinformationen und dem von ihm festgelegten Passwort beim System anmelden und von nun an die gewünschte Aktivität ausführen.

Ärzte können auch ihren eigenen Eintrag von einem anderen Einstiegspunkt aus vornehmen als Patienten. Ärzte werden wie Patienten nach ihren eigenen Zugangsdaten und einem Passwort gefragt. Ärzte, die dies tun, können sich auch einfach in ihre eigenen Schnittstellen einloggen.

### **Komponente (Arzt GUI - Patient GUI - Chefarzt GUI)**

Da es den Patienten gesundheitlich nicht gut geht und sie vieles nicht mehr können, soll das angebotene System einfach, verständlich und leicht sein. Patienten müssen in maximal 4 Tasten das bekommen, was sie wollen, damit sie die Arbeit ohne großen Aufwand erledigen können.

In diesem Abschnitt können Patienten bestimmte Verfahren nach ihren Wünschen durchführen. Diese Prozesse werden in Kapitel 4 „Funktionale Anforderungen“ ausführlicher und übersichtlicher erläutert.



Abbildung 5 Online Terminvereinbarung

Ärzte ermüden tagsüber nicht nur geistig, sondern auch körperlich. Jeden Tag müssen sie Menschenleben retten, helfen und mit Menschen umgehen. Zusätzlich zu diesen Schwierigkeiten ist es für Ärzte eine zusätzliche Belastung, die Termine ihrer eigenen Patienten zu planen. Um dies zu minimieren, sollten Ärzte nur während der Untersuchung mit Patienten kommunizieren. Ärzte sollten auch am Tag vor der Ankunft der Patienten über ihre Termine informiert werden.

In diesem Abschnitt können Ärzte ihren Zeitplan leicht einhalten. Diese Prozesse werden in Kapitel 4 „Funktionale Anforderungen“ ausführlicher und übersichtlicher erläutert.

Wie in jeder Branche gibt es auch im Gesundheitswesen eine hierarchische Situation. An der Spitze der Ärztehierarchie steht der Chefarzt. Der Chefarzt kümmert sich um die interne Koordination des Krankenhauses und erkennt die Ärzte im Krankenhaus an.

Der Chefarzt kann die Arzt- und Polikliniklisten einsehen und organisieren. Es wird das System gewöhnlicher machen und mögliche Fehler minimieren. Diese Prozesse werden auch in Kapitel 4 „Funktionale Anforderungen“ ausführlicher und übersichtlicher erläutert.

### **Externe Schnittstellen**

Die externen Schnittstellen sind:

1. Arzt-Interface: Ärzte können ihre Termine einsehen.
2. Patient-Interface: Patienten können einen Termin annehmen/löschen und ihn sehen.
3. Chefarzt-Interface: Der Chefarzt kann Ärzte und Polikliniken hinzufügen und die Listen sehen.

Für die Spezifikationen der Schnittstellen siehe Abbildung 3 - UML-Diagramm und Kapitel 4 „Funktionale Anforderungen“.

## 4. Funktionale Anforderungen

### Einleitung

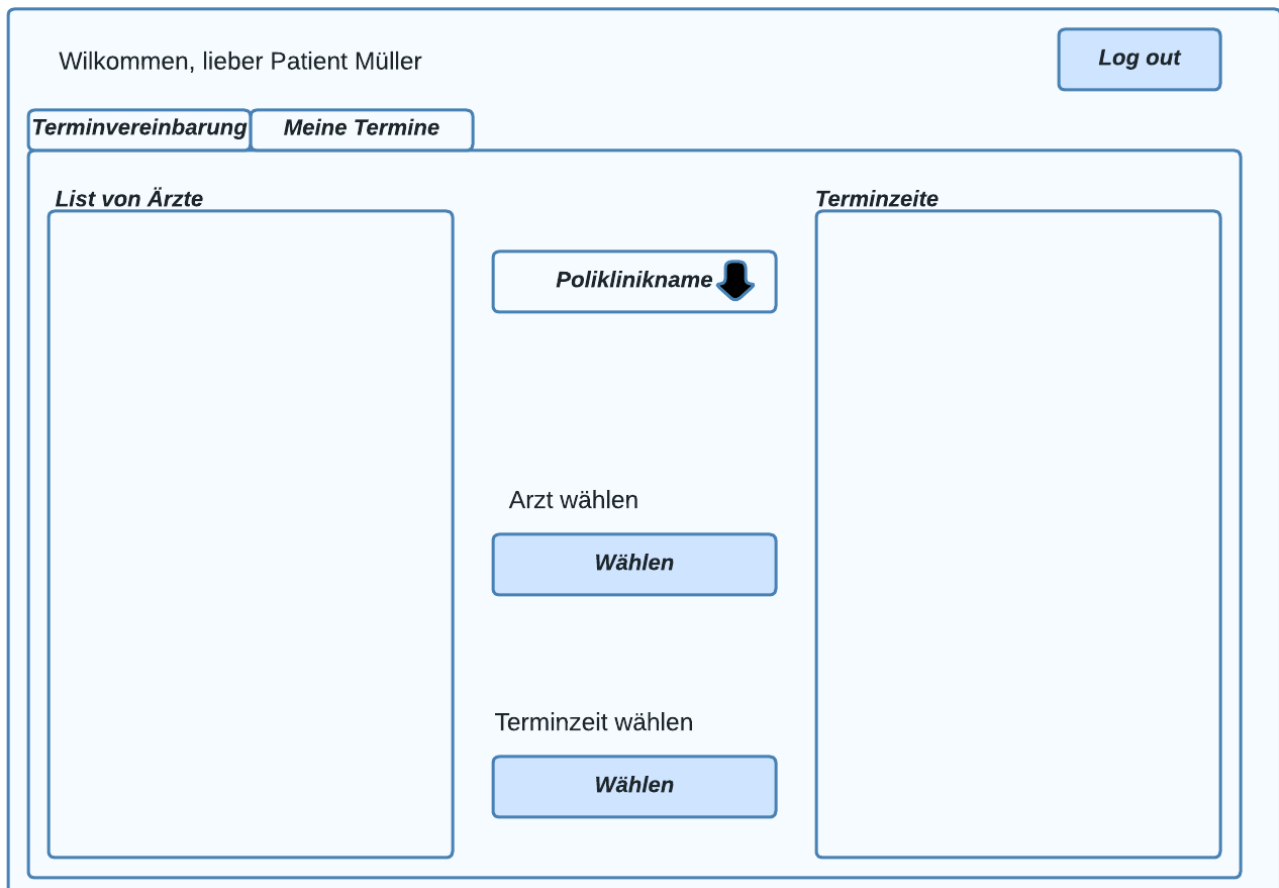
In diesem Kapitel werden Anforderungen (inkl. User Stories und Use Cases) an das Gesamtsystem, aber auch an die einzelnen Systemkomponenten definiert.

### Use Cases / User Stories

#### 1. Patient Use Cases

Dieser Abschnitt beschreibt, was Patienten tun können.

- **Step1: Register-UI**
  - Mit der Register UI können Benutzer ein Passwort erstellen, das mit ihrer Identitätsnummer übereinstimmt, und von nun an werden sie dieses Passwort verwenden, um das System zu betreten.
- **Step2: Login-UI**
  - In diesem Schritt melden sich die Benutzer mit ihrer Identitätsnummer und ihrem eigenen Passwort beim System an.
- **Step3: Patient-UI**
  - In diesem Schritt wird den Patienten mehr als eine Wahl angeboten.
    1. Patienten wählen zunächst eine Poliklinik, um sich ihren Termin aussuchen zu können.
    2. Danach gibt es einen Prozess, um die gewünschten Ärzte auszuwählen, und sie wählen von dort aus einen Arzt aus.
    3. Nach der Wahl des Arztes wählen sie die Uhrzeit und vereinbaren einen Termin.
  - Nachdem der Termin erstellt wurde, wird dieser Vorgang in die Datenbank übertragen und Patienten, die ihren Termin festgelegt haben, können dann unter "Meine Termine" sehen, wann und bei welchem Arzt ihr Termin ist.



The screenshot shows a patient interface with a light blue background. At the top left, it says 'Willkommen, lieber Patient Müller'. At the top right is a 'Log out' button. Below the welcome message are two tabs: 'Terminvereinbarung' (selected) and 'Meine Termine'. The main content area is divided into three sections. On the left is a large empty box labeled 'List von Ärzte'. In the center, there is a dropdown menu labeled 'Poliklinikname' with a downward arrow, followed by the text 'Arzt wählen' and a 'Wählen' button. Below that is the text 'Terminzeit wählen' and another 'Wählen' button. On the right is a large empty box labeled 'Terminzeit'.

Abbildung 6 Patient GUI

## 2. Arzt Use Cases

Dieser Abschnitt beschreibt, was Ärzte tun können.

- **Step1: Register UI**
  - Benutzer, die Ärzte sind, können keine eigenen Passwörter erstellen, Chefarzt erstellt für sie. Sie betreten das System mit ihrer Identitätsnummer und dem für sie erstellten Passwort.
- **Step2: Login UI**
  - In diesem Schritt melden sich die Benutzer mit ihrer Identitätsnummer und ihrem eigenen Passwort beim System an.
- **Step3: Arzt UI**
  - Wenn sich der Arzt beim System anmeldet, wird er mit der Arzt-UI konfrontiert. Hier sehen Sie die Termine mit dem Zeit-, Patientenname und Termin-ID.

## 3. Chefarzt Use Cases

Dieser Abschnitt beschreibt, was der Chefarzt tun könnte.

- **Chefarzt UI**
  - Der Chefarzt des Krankenhauses fungiert hier als Administrator. Er fügt den Ärzten des Krankenhauses in dem System hinzu und definiert für sie ein Passwort. An dieser Oberfläche können Sie die von Ihnen hinzugefügten Ärzte mit ihren Funktionen anzeigen.

- Darüber hinaus kann der Chefarzt dem Krankenhaussystem Kliniken hinzufügen und dieser Poliklinik Ärzte zuweisen. können die Polikliniken im System einsehen.

## Technischen und fachliche Anforderungen

Der Chefarzt trägt mit dem Register-UI Ärzte in die Datenbank ein und Patienten tragen sich mit dem Register-UI in die Datenbank ein. Daher melden sich Patienten und Arzt gemäß den Informationen in der Datenbank im System mit der Login-Benutzeroberfläche an und führen ihre Operationen durch.

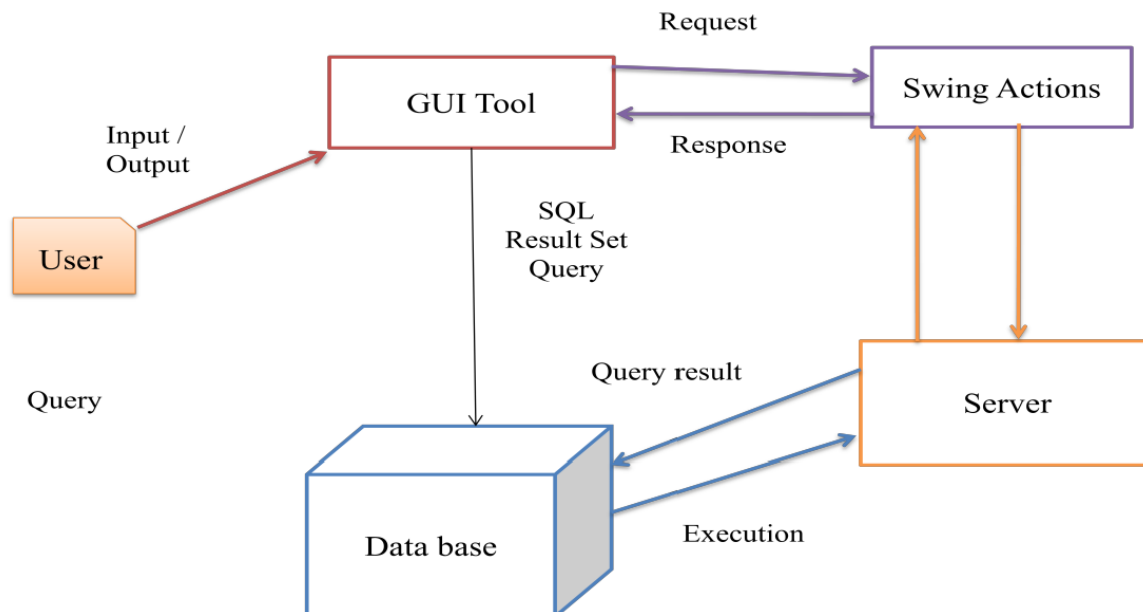


Abbildung 7 Datenverarbeitung Schema

Der Chefarzt fügt dem System Ärzte und Polikliniken hinzu, und diese Informationen werden übertragen und in der Datenbank gespeichert. Wenn Patienten sich anmelden und einen Termin vereinbaren möchten, werden Ärzte, Polikliniken und Zeiten aus der Datenbank gezogen und ihnen präsentiert. Nach dem Erstellen eines Termins werden die Termininformationen auch in der Datenbank gespeichert.

## Datenmodell

- Relevante Systemparameter sollten zur einfachen und effizienten Verwaltung in einer externen Datenbank gespeichert werden. Eine Liste mit Parametern wird vom Stakeholder bereitgestellt und kann nach Bedarf angepasst werden. Die Verwendung einer externen Datenbank ermöglicht auch die gemeinsame Nutzung von Parametern zwischen verschiedenen Instanzen des Systems, was die Skalierbarkeit und Flexibilität erhöht.
- Für die Datenpersistenz ist ein externes DBMS (SQL-Datenbank) erforderlich. Speichern, abrufen und aktualisieren Sie Daten sicher und effizient mit einer SQL-Datenbank. Es bietet außerdem eine hervorragende Skalierbarkeit und Zuverlässigkeit und ist eine bewährte Lösung zum Speichern von Daten.

- Sie sollten ein DBMS auswählen und es so einrichten, dass Daten in separaten Komponenten für eine gute modulare Architektur gespeichert werden. Eine komponentenbasierte Architektur bietet Vorteile wie erhöhte Wiederverwendbarkeit, Wartungs- und Testfreundlichkeit sowie erhöhte Flexibilität und Skalierbarkeit.
- Das Datenmodell wird mit UML-Klassendiagrammen modelliert, um eine klare und konsistente Dokumentation zu gewährleisten. UML-Klassendiagramme bieten eine standardisierte Möglichkeit, Objekte und ihre Beziehungen zu modellieren und die Kommunikation zwischen Entwicklern, Designern und anderen Beteiligten zu erleichtern. UML-Klassendiagramme können auch Datenstrukturen und Beziehungen innerhalb eines Systems klar und deutlich darstellen und so zu einer verbesserten Codequalität und einer effizienten Entwicklung beitragen.

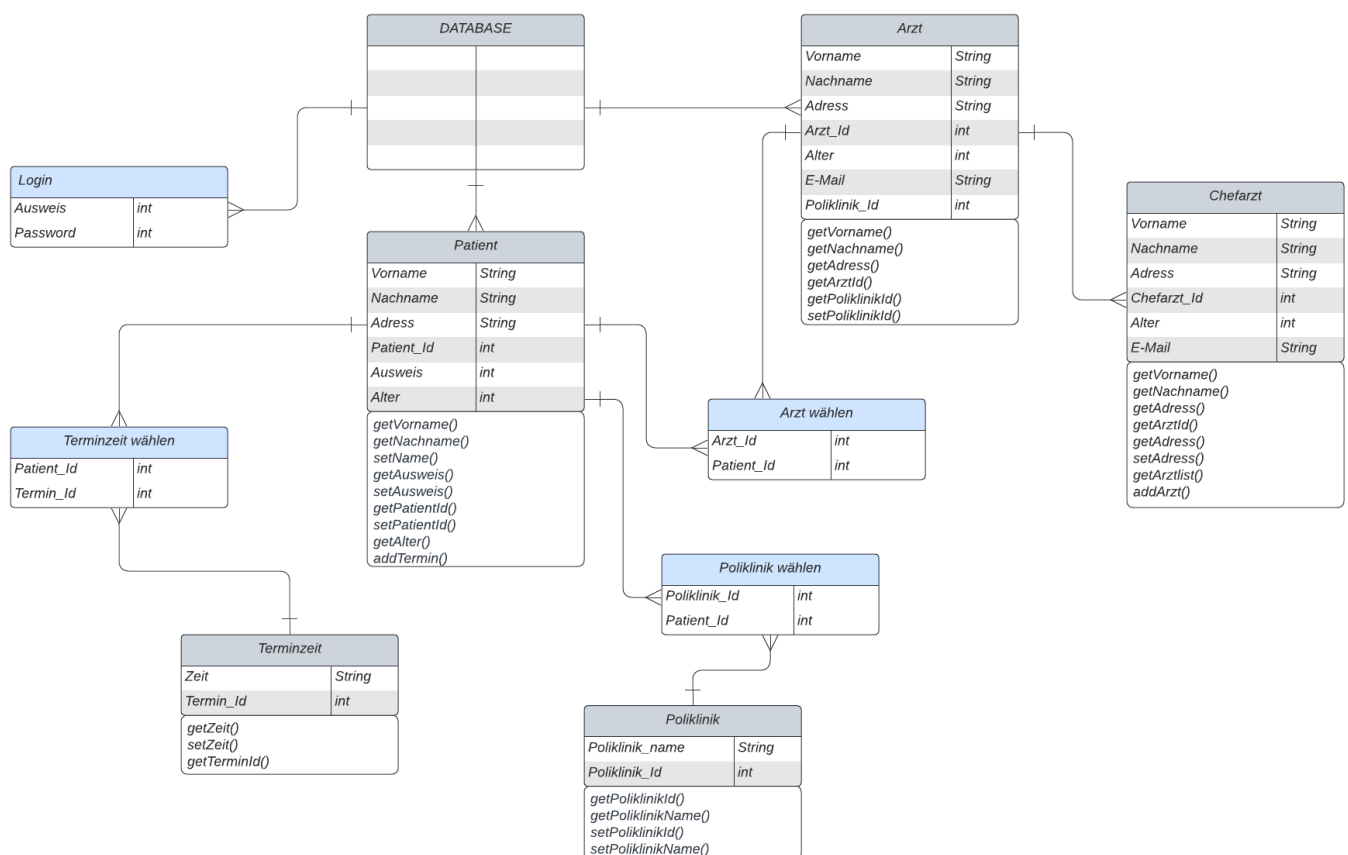


Abbildung 8 UML-Diagramm

## 5. Nichtfunktionale Anforderungen

### Einleitung

In diesem Kapitel sind die nicht-funktionalen Anforderungen ans Gesamtsystem aber auch an die einzelnen Systemkomponenten definiert. Es wird besonders auf die Software Qualität Wert gelegt (Testing).

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

- Architekturmuster: Zur sauberen Trennung von Datenzugriff, Geschäftslogik und Darstellungsschichten empfiehlt sich die Verwendung einer Schichtenarchitektur zur Umsetzung des Anmeldeprozesses. Dies erleichtert die Skalierbarkeit des Systems und ermöglicht Änderungen an einzelnen Schichten ohne Auswirkungen auf andere Schichten.
- Deployment: Das System sollte auf einem Webserver bereitgestellt werden, um die Verfügbarkeit für Benutzer zu gewährleisten. Im Login-Prozess werden die Daten von Webservices aus der Datenbank kontrolliert und entsprechend dem Ergebnis wird der Prozess fortgesetzt.
- Die Kommunikation der Systemkomponenten hängt vom Benutzertyp ab. Es wird erwartet, dass das System je nach Benutzertyp unterschiedliche Daten und Funktionen anzeigt.

### Nicht-funktionale Anforderungen an die Entwicklungsumgebung

- Als Entwicklungsumgebung verwenden wir Eclipse. Entwicklungsteams sollten sich darauf einigen, Eclipse als ihre Haupt-IDE zu verwenden, um eine konsistente Arbeitsumgebung zu gewährleisten und effektiv zusammenzuarbeiten.

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

- Sprache: Die Backend-Anwendung sollte in einem Java-Framework implementiert werden. Dies ermöglicht eine robuste und effiziente Entwicklung unter Verwendung einer großen Anzahl bestehender Frameworks und Bibliotheken.
- IDE: Die Verwendung von Eclipse als Haupt-IDE bietet eine bessere Integration der verwendeten Frameworks und eine einheitliche Arbeitsumgebung für Ihr Entwicklungsteam.
- Die Frontend-Anwendung ist in Java Swing implementiert. Java Swing ist ein etabliertes Framework zur Entwicklung von Desktop-Anwendungen in Java. Es bietet verschiedene Standard-GUI-Komponenten und eine einfache Integration mit anderen Java-basierten Systemkomponenten.
- Persistente Daten werden in einer SQL-Datenbank gespeichert. SQL-Datenbanken bieten eine zuverlässige Möglichkeit, strukturierte Daten zu speichern, was das Abfragen und Bearbeiten von Daten erleichtert. Allerdings muss darauf geachtet werden, dass die Datenbank effizient skalieren kann und Datenschutz- und Sicherheitsanforderungen erfüllt werden.

### **Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)**

- Black-Box-Tests: Alle Systemdienste sollten Black-Box-Tests unterzogen werden, um sicherzustellen, dass die Schnittstelle korrekt funktioniert und die erwarteten Ergebnisse liefert.
- Vollständigkeitstest: Alle Use Cases und User Stories sollten getestet werden, um sicherzustellen, dass das System alle funktionalen Anforderungen erfüllt und korrekt funktioniert.
- GUI-Tests: GUI-Komponenten können separat vom System getestet werden, um sicherzustellen, dass die Benutzeroberfläche intuitiv ist und eine positive Benutzererfahrung bietet.



## 6. Abnahmekriterien

*Die Abnahmekriterien sind durch den Stakeholder definiert und sie dürfen nur mit Zustimmung des Stakeholders neu definiert, geändert oder erweitert werden.*

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: **INF 202-Krankenhaus-Terminsystem-Pflichtenheft-2023.v1.0.docx**
- Software
  - Link zu GitHub Projekt:  
<https://github.com/orgs/Gruppe1-Fulya/teams/40-komando-alayi>
- Evidenz:
  - System/Software-Demo via Videoclip:

Anm.: Die Abgabetermine der Projectartefakts 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 steht 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 steht 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 worden.

### ***Meilenstein M#5:***

Das Projekt ist per Vereinbarung abgegeben worden.

## 8. Referenzen

- <https://docs.oracle.com/javase/8/docs/api/overview-summary.html>
- <https://www.eclipse.org/projects/handbook/#starting>
- <https://docs.oracle.com/en-us/iaas/mysql-database/doc/getting-started.html>
- <https://docs.oracle.com/javase/tutorial/uiswing/index.html>
- <https://www.geeksforgeeks.org/introduction-to-java-swing/>
- <https://www.javatpoint.com/example-to-connect-to-the-mysql-database>
- <https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database>

## Abbildungen

1. <https://www.alamy.com/stock-photo-operation-krankenhaus-termin-im-kalender-173209543.html>
2. <https://www.orbnet.de/mit-online-terminvereinbarung-den-umsatz-immens-steigern/>
3. [https://www.guidanceshare.com/wiki/Application\\_Architecture\\_Guide\\_-\\_Chapter\\_9\\_-\\_Layers\\_and\\_Tiers](https://www.guidanceshare.com/wiki/Application_Architecture_Guide_-_Chapter_9_-_Layers_and_Tiers)
4. Originalabbildung
5. <https://stock.adobe.com/tr/search?k=book%20doctor%20appointment>
6. Originalabbildung
7. <https://www.semanticscholar.org/paper/Design-of-a-Simple-Graphical-User-Interface-to-the-Vijayprasath-Rajan/28d98001eaa3c892f63f2989db7913de0a941100>
8. Originalabbildung