Abschlussprüfung Winter 2022/2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur Projektarbeit

Patienten- und Terminverwaltung für eine Gemeinschaftspraxis

Webanwendung zur Erfassung und Verwaltung von Patientendaten mit Terminvergabe

Abgabetermin 15.12.2022

Prüfungsbewerber:

Frank Bartl
Ortsstraße 15
63928 Eichenbühl-Guggenberg

Ausbildungsbetrieb:

Berufsförderungswerk Schömberg gGmbH

Bühlhof 6

75328 Schömberg

Projektbetreuer:

Holger Bube, Reha Ausbilder

Einleitung

Heutzutage ist es kaum vorstellbar, ohne eine digitalisierte und effiziente Patienten- und Terminverwaltung zu arbeiten, da diese den Organisations-Aufwand einer Praxis erheblich verbessert.

Diese Dokumentation dient dazu einen Überblick zu verschaffen, wie eine Patienten- und Terminverwaltungs-Software für eine Arztpraxis erdacht und implementiert werden kann. Es handelt sich um einen fiktiven Kunden, eine Gemeinschaftspraxis mit 4 Ärzten und rund 300 Patienten.

Die Einführung der Software wird dazu beitragen, die Arbeitsauslastung koordinierter zu gestalten und damit die Effizienz zu steigern.

Hinweise zur Lesbarkeit

Um den Lesefluss zu gewährleisten, wird an einigen Stellen bei Personenbezeichnungen und personenbezogenen Hauptwörtern ausschließlich die männlichen Form verwendet. Im Sinne der Gleichbehandlung gelten entsprechende Begriffe grundsätzlich für alle Geschlechter.

Für eine bessere Übersicht sind die Überschriften der einzelnen Kapitel farblich hinterlegt. Auch können die Überschriften im Inhaltsverzeichnis bei der Desktopvariante angeklickt werden, um zum entsprechenden Kapitel zu springen.
Wichtige Sätze oder Schlagwörter sind fett und/oder kursive geschrieben. Für Abkürzungen und Tabellen ist ein separates Verzeichnis vorhanden.

Rechtliche Hinweise

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

E

EPK ..... *Ereignisgesteuerte Prozesskette*
ER-Models ..... *Entity-Relationship-Modell*

O

ORM ..... *Object Relational Mapping*

R

RDBMS ..... *Relational Database Management System*

Tabellenverzeichnis

1 Projektbeschreibung

Die folgende Dokumentation schildert den Ablauf des Projekts für die Abschlussprüfung der IHK, welches der Autor im Rahmen der Umschulung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt hat. Der Ausbildungsbetrieb ist das Berufsförderungswerk Schömberg gGmbH welcher bundesweit vertreten ist. Es unterstützt Menschen dabei nach langer Krankheit oder anderen Defiziten, die ihre alte Arbeit nicht mehr ausüben können, wieder zurück in die Arbeitswelt zu kommen.

Es handelt sich bei diesem Projekt um ein fiktives Szenario. Der Kunde, "Dr. Klein & Kollegen" haben eine Gemeinschaftspraxis für Allgemeinmedizin. Sie praktizieren seit rund 20 Jahren im Stadtzentrum von Miltenberg. Das Praxisteam ist besteht aus 2 Ärzten und 8 Arzthelferinnen. Der Patientenstamm umfasst ca. 300 Patienten und wächst weiter an.

1.1 Ist-Zustand

Die Patienten- und Terminverwaltung erfolgt aktuell nur in handschriftlicher Form. Externe Befunde, die per E-Mail gesendet werden, werden ausgedruckt und in die Karteikarte einsortiert. Des Weiteren werden die Patientenkarten vom Praxispersonal in den Behandlungsraum gebracht. Nach der Behandlung werden die Kateikarten in den Aktenschrank einsortiert.

1.2 Problematik

Die aktuelle Situation in der Arztpraxis kostet wertvolle Arbeitszeit und Ressourcen, wie z.B. Karteikarten und Toner. Die Arbeitsorganisation ist ineffizient und die handschriftlichen Einträge in die Karteikarte schwer nachvollziehbar, da stellenweise sehr unleserlich geschrieben wird.

1.3 Soll-Zustand

Der Kunde wünscht sich eine Webanwendung, die dabei hilft, die Termin- und Patientenverwaltung zu erleichtern.

Mit Hilfe der Anwendung wird die gesamte Verwaltung digitalisiert, Termine können bequem einem Arzt zugeordnet werden, eine versehentliche Doppelbelegung ist nicht mehr möglich. Befunde und Behandlungsnotizen werden der digitalen Patientenkarteikarte zugeordnet.

Bei der Einführung wird es eine Schulung für die Mitarbeiter geben, damit diese sicher mit dem Programm umgehen können. Für weitere Unterstützung wird ein Support vertrag erstellt.

1.4 Projektbegründung

Durch die aktuelle Situation ist das Praxispersonal, die meiste Zeit mit organisatorischen Tätigkeiten beschäftigt, Karteikarten für den Termin herauszusuchen und dem Arzt ins Behandlungszimmer zu legen. Nach dem Termin bringen die Ärzte die Karten zurück an den Empfang, damit diese wieder einsortiert werden können. Dadurch haben die Arzthelferinnen weniger Zeit für Behandlungen wie Blutdruck messen, Blutabnehmen EKG und dergleichen, was für die Patienten deutlich längere Wartezeiten bedeutet.

Mit Einführung der Verwaltungssoftware, wird der organisatorische Aufwand erheblich reduziert. Die Arzthelferinnen brauchen lediglich neue Patienten über die Software anlegen. Der Arzt kann sich selbstständig über die Suchfunktion bequem den Patienten des jeweiligen Termins anzeigen lassen und weitere Behandlungen oder neue Diagnosen zu dem Patienten hinzufügen.
Allgemein wird ein effizienteres Arbeiten ermöglicht, was eine höhere Anzahl von Patienten bedeutet, die an einem Tag behandelt werden können. Weiter können massiv Ressourcen eigespart werden, was nicht nur die Kosten reduziert, es ist ein großer Beitrag zum Umweltschutz.

1.5 Abweichungen gegenüber dem Projektantrag

Im Vergleich zum Projektantrag ist der Punkt Code Review hinzugekommen, dieser dient zur ersten Überprüfung des Quellcodes durch einen weiteren Mitarbeiter.

2 Projektplanung

Im Kapitel Projektplanung wird im Wesentlichen erklärt, wie die Planung im Allgemeinen stattgefunden hat. Es werden die Phasen der Planung und die Kosten Berechnung dargestellt.

2.1 Projektphasen

Die Projektphasen sind in folgende Punkte unterteilt:

- Kundenwünsche aufnehmen
- Oberflächen Design
- Datenbank erstellen
- Code schreiben Java
- Dokumentation
- Vorhandene Daten einpflegen
- Tests und Fehlerbehebung
- Inbetriebnahme beim Kunden inklusive Mitarbeiterschulung

Projektphasen	Geplante Zeit
Planung	8 h
Design	13 h
Datenbank erstellen	10 h
Code schreiben mit Java	20 h
Funktionstest und Fehlerbehebung	13 h
Vorhandene Daten einpflegen	4 h
Dokumentation	9 h
Inbetriebnahme beim Kunden	2 h
Gesamt	**79 h**

Tabelle 1 Grobe Zeitplanung

Die *Tabelle 1* Grobe Zeitplanung zeigt eine Grobe Zeitplanung. Was von dieser groben Zeiteiteilung wirklich eintrifft bzw. wie sich die Zeiten im Verlaufe des Projekts verschieben wird im Kapitel *Fazit* erläutert.

2.2 Personalplanung

Für die Umsetzung des Projekts wird eine Person eingesetzt. Lediglich für die Qualitätssicherung wird eine zusätzliche Person eingesetzt, diese wird ein Code Review durchführen.
Der Stundensatz liegt bei 95 €. Bei der Planung wird von einer Arbeitszeit von 79 Std ausgegangen. Daraus ergibt sich Personaleinsatz ohne MwSt. in € von 7.505,00 €. Die Gesamtkosten können im Kapitel *2.3 Gesamtkosten* eingesehen werden.

2.3 Gesamtkosten

In der Nachfolgenden Tabelle werden die Gesamtkosten aufgeführt, diese setzen sich aus Kosten für die Arbeitszeit und der Mehrwertsteuer zusammen.

Gesamtkosten

Stundenlohn	95,00 €
Arbeitszeit	79 h
Kosten o MwSt.	7.505,00 €
MwSt. 19%	1.425,95 €
Kosten inkl. MwSt.	**8.930,95 €**

Tabelle 2 Gesamtkosten

2.4 Projektschnittstellen

Das Projekt wird von dem Autor zum größten Teil allein umgesetzt. Um die Wünsche des Kunden zu erfüllen, kommt es in regelmäßigen Abständen zu Austausch mit dem Arzt Dr Klein. Des Weiteren gibt es einen Kollegen, der für das Code Review eingesetzt wird.

2.5 Abgrenzung

Da während des gesamten Entwicklungsprozess ein Austausch mit dem Kunden stattfindet, sind evtl. Änderungen vorbehalten.
Der Auftrag für den Autor besteht darin die App, bis zur Inbetriebnahme zu Programmieren. Die fertige Anwendung wird einer Externen IT-Firma zur Verfügung gestellt, diese stellt eine Instanz auf ihren Server. Die Mitarbeiter Schulung wird wieder vom Autor übernommen.

2.4 Sachmittelplanung

Die benötigten Sachmittel für dieses Projekt sind:

Hardware und Software

- Desktop PC
- Microsoft Visio → zum Erstellen des ER-Models, EPK und RDBMS
- PowerPoint → zum Erstellen des GANTT Diagramms
- IntelliJ IDEA Community Edition → Erarbeiten des Codes und Oberflächen Gestaltung
- HeidiSQL → Bereitstellung der Datenbank MariaDB
- GitHub → Visionsverwaltung

2.5 Zielplattform Planung

Der Wunsch des Kunden ist es, eine Webanwendung zu erhalten. Dies gibt dem Autor ein breites Spektrum an Möglichkeiten bei der Auswahl der Programmiersprache. Von PHP, Java, JavaScript über Ruby bis hin zu Python um nur ein paar zu nenne.

Da der Autor im Rahmen seines Praktikums Einblicke in Java erhalten hat, hat er sich für diese Programmiersprache entschieden. Java bietet auch viele Vorteile. Zu den größten Stärken gehört, seine Plattformunabhängigkeit. Als Ergänzungen werden die Frameworks SpringBoot, Vaadin und Hibernate genutzt.

2.6 Wirtschaftliche Betrachtung

Durch die Einführung der Patientenverwaltungssoftware werden die organisatorischen Strukturen der Praxis verändert. Die Arzthelferinnen können sich mehr auf die Behandlung und Betreuung der Patienten konzentrieren. Im Behandlungsraum gibt es zudem keine Wartezeiten mehr durch evtl. falsche Patientenkarten. Der gesamte Ablauf wird reibungsloser und hat weniger Fehlerquellen.

2.7 Amortisationsrechnung

Die nachfolgenden Tabellen geben einen Überblick über die Material- und Mitarbeiterlohn kosten vor und nach Einführung der Patienten- und Terminverwaltungsapplikation.

Material- und Mitarbeiterlohnkosten ***vor Einführung*** der Verwaltungssoftware

Materialkosten	**Einzelpreis in €**	**Ø Verbrauch pro Druck in €**	**Ausdruck pro Tag in € (x4)**	**Ausdruck pro Monat in € (x20)**	**Ausdruck pro Jahr in € (12)**
Toner für Drucker		0,033	0,132	2,640	31,68
Druckerpapier		0,008	0,032	0,640	7,68
Karteikarten DIN A5, 70 Stück pro Jahr	0,118				8,26
Gesamtkosten pro Jahr:					**47,62**

Tabelle 3 Materialkosten vor der Einführung

Mitarbeiterlohnkosten

Ø Stundenlohn **16,42 €** x Faktor 2,0 Arbeitgeber Brutto	32,84
Beschäftigungszeit Karteikarte: 0,75 h x 3 Arzthelferinnen = 2,25 h/Tag in €	73,89

Arbeitszeit pro Monat in €: 73,89 € x 20 Arbeitstage	1.477,80
Arbeitszeit pro Jahr in €: 1.477,80 € x 12 Monate	17.733,60
Materialkosten pro Jahr in €	47,62
Gesamtkosten pro Jahr in €	**19.365,75**

Tabelle 4 Mitarbeiterlohnkosten vor der Einführung

Material- und Mitarbeiterlohnkosten ***nach Einführung*** der Verwaltungssoftware

Materialkosten	**Einzelpreis in €**	**Ø Verbrauch pro Druck in €**	**Ausdruck pro Tag in € (x4)**	**Ausdruck pro Monat in € (x20)**	**Ausdruck pro Jahr in € (12)**
Toner für Drucker		0,000	0,000	0,000	0,00
Druckerpapier		0,000	0,000	0,000	0,00
Karteikarten DIN A5, 70 Stück pro Jahr	0,000				0,00
Gesamtkosten pro Jahr:					**0,00**

Tabelle 5 Materialkosten nach der Einführung

Mitarbeiterlohnkosten

Ø Stundenlohn **16,42 €** x Faktor 2,0 Arbeitgeber Brutto	32,84
Beschäftigungszeit im Frontend der Verwaltungssoftware: 32,84 € : 60 Minuten = 0,55 €/min. x 10 Minuten x 2 Arzthelferinnen	10,95
Arbeitszeit pro Monat in €: 10,95 € x 20 Arbeitstage	219,00
Arbeitszeit pro Jahr in €: 219,00 € x 12 Monate	2.628,00
Materialkosten pro Jahr in €	0,00
Gesamtkosten pro Jahr in €	**2.890,79**

Tabelle 6 Mitarbeiterlohnkosten nach der Einführung

Ersparnis der Gesamtkosten im Jahr

Gesamtkosten ohne Verwaltungssoftware in €	19.365,75
Gesamtkosten mit Verwaltungssoftware in €	2.890,79
Gesamtkosten in €	**16.474,96**

Tabelle 7 Ersparnis der Gesamtkosten pro Jahr

Wie zu erkennen ist, ist die Investition in eine Verwaltungssoftware durchaus sinnvoll, da die bisherigen Materialkosten für Druckerpapier, Toner und Karteikarten wegfallen und zusätzlich die Mitarbeiterlohnkosten um 85,04% gesenkt werden.

Die Amortisationszeit beträgt (Berechnungsbetragt geteilt durch Ersparnis pro Jahr):

$$\frac{8.930{,}95 \quad €}{16.474{,}96 \frac{€}{Jahr}} \approx 0{,}54\, Jahre \approx 6 \text{ Monate } 16 \text{ Tage} \approx 28\, Wochen$$

Die Amortisierung ist nach ca. 28 erreicht. Das ist der Zeitraum, in der die Verwaltungssoftware mindestens zum Einsatz kommen muss, damit sich das Projekt wirtschaftlich rechnet.

2.8 Ablaufplanung

Bevor mit der Umsetzung des Projekts begonnen werden konnte, mussten sich der Autor für einen geeigneten Entwicklungsprozess entscheiden. Dieser definiert die Vorgehensweise, nach der die Umsetzung erfolgen soll. Da der Autor das Projekt bis auf das Code-Review alleine betreut, wird das Wasserfallmodel zur Projektsteuerung herangezogen.

Die Modelle für die Ablaufplanung sind:

- Projektstrukturplan
- GANTT-Diagramm
- EPK

Projektstrukturplan

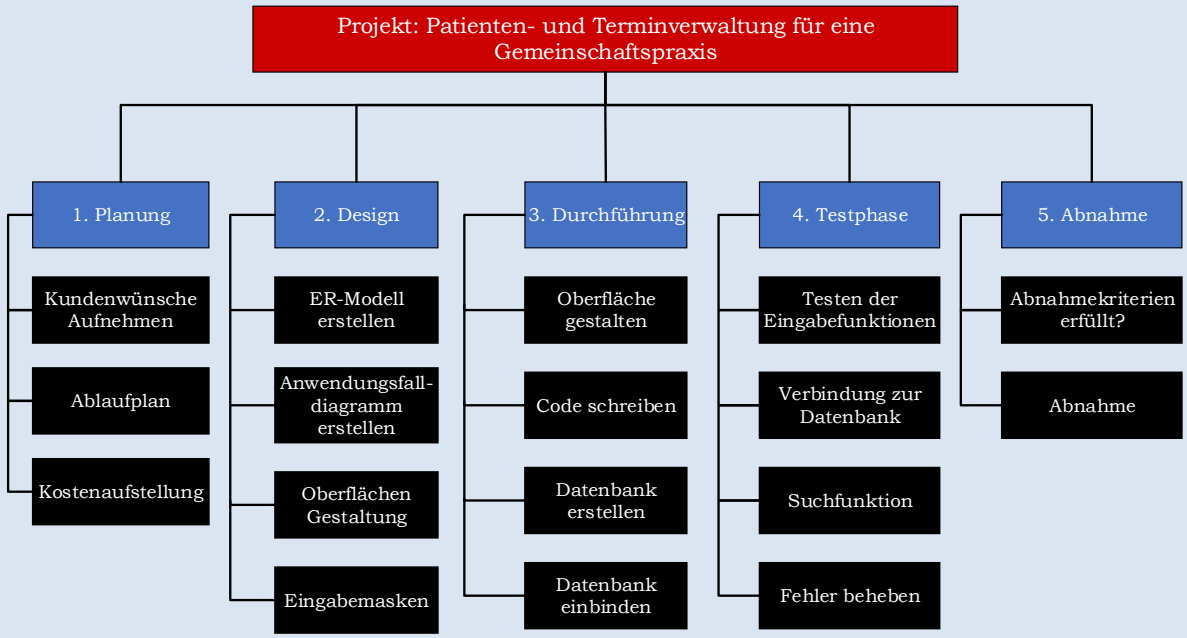


Abbildung 1 Projektstrukturplan

GANTT-Diagramm

Um die Zeitliche Einteilung und die Teilziele darstellen zu können, hat sich der Autor für das GANTT-Diagramm entschieden. Die Teilziele sind als Meilensteine aus dem Diagramm zu entnehmen. Dies finden sie im Anhang *A1 Gantt-Diagramm*

EPK-Modell

Die Ereignisgesteuerte Prozesskette welche dem Anhang *A2 EPK-Modell* zu entnehmen ist, zeigt den Grundsätzlichen Ablauf des Projekts.

3 Entwurfsphase

Das Kapitel *Entwurfsphase* beschäftigt sich mit der Planung, von der Oberfläche bis hin zur Businesslogik und der Datenbank.

3.1 Zielplattform

Die Auswahl der Programmiersprache für das Projekt wurde zu Beginn der Planungsphase festgelegt und ist dem Kapitel *2.5 Zielplattform Planung* einzusehen.

3.2 Layout Design

Der Wunsch des Kunden ist es, eine einfache Webapplikation zu bekommen, die die wesentlichen Aspekte der Patienten- und Terminverwaltung abdeckt. Hierfür wird eine Webanwendung geschaffen welche leicht zu bedienen ist. Die Startseite ist mit Hintergrundbild des Empfangsbereichs der Praxis versehen. Von dort aus sind alle Bereiche bequem erreichbar. Auf der Startseite ist eine Menüleiste vorhanden, diese kann bei Bedarf eingeklappt werden. Über dieses Menü lassen sich die anderen Bereiche, Termin, Behandlung, Patient, Arzt, Mitarbeiter, Krankenkasse und Diagnose der Web-App schnell mit einem Klick erreichen. Das erste Layout wurde skizziert und mit dem Kunden besprochen.

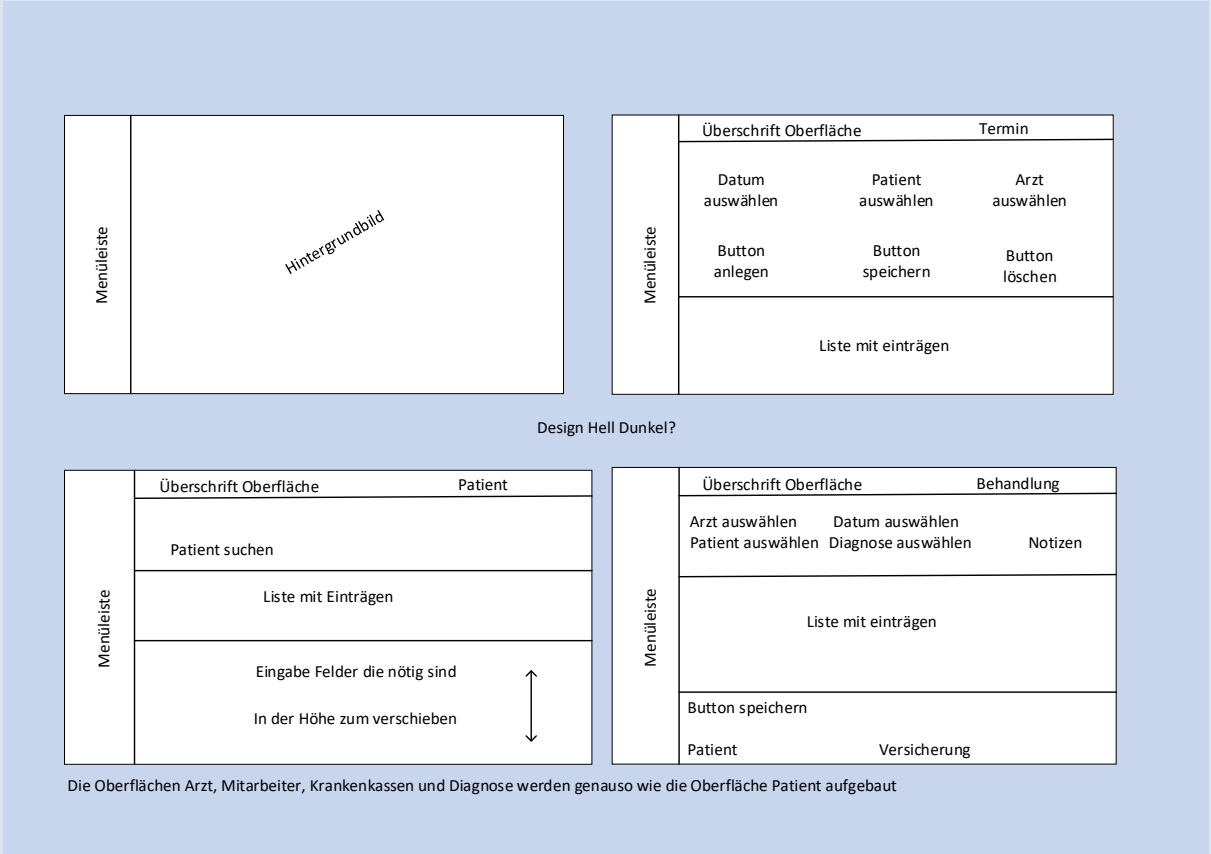


Abbildung 2 Skizze Oberflächen

3.3 Anwendungsfälle

Um eine Übersicht aller Anwendungsfälle zu erhalten, welche in der Web-App möglich sein sollen, wurde in der Entwurfsphase ein Anwendungsfalldiagramm (Use-Case-Diagramm) erstellt. Die dazugehörige Abbildung befindet sich im Anhang und ist unter *A3 Use-Case-Diagramm* einzusehen.

3.4 Entwurf des Datenbankdesigns

Der Datenbankentwurf wurde in Visio in einem ER-Modell logisch aufgebaut und dargestellt, zur bessern Ansicht und um im Anschluss die SQL-Datenbank einfacher zu implementieren.

Nachfolgend werden zwei Entitäten des ER-Modells Arzt und Termin beschrieben. Die Entität Arzt soll die wichtigsten Daten abbilden, welche für den Arzt relevant sind. Die Entität Termin steht in Relation zu dem Arzt. Ein Arzt kann mehrere Termine haben, ein Termin wird von einem Arzt betreut. Die beiden Entitäten stehen in einer 1:n Beziehung.
Für den Entwurf des ER-Modells wurde im Vorfeld mit dem Kunden geklärt, welche Informationen er gespeichert haben möchte, zudem wurde es um relevante Attribute ergänzt.

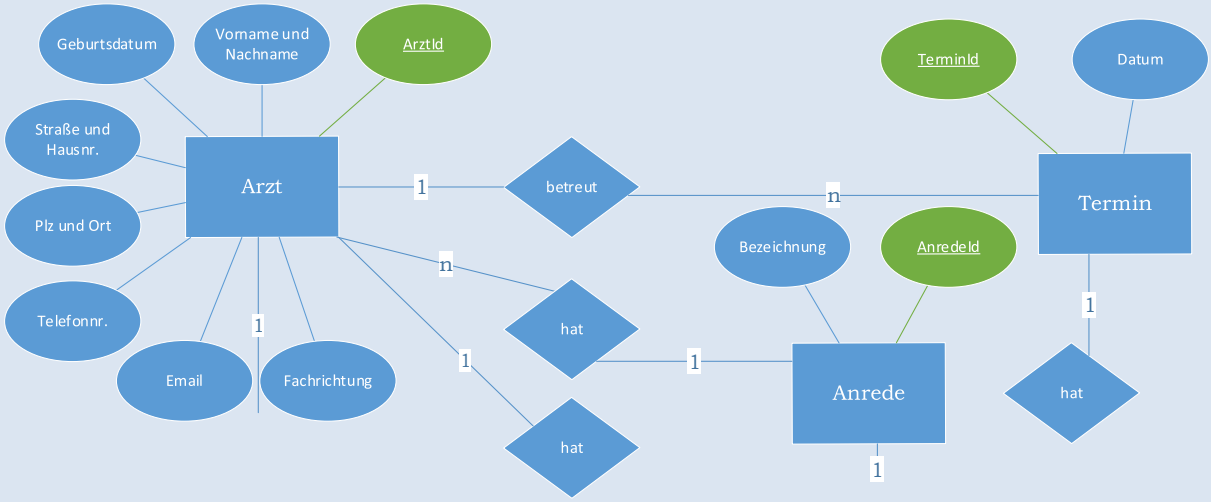


Abbildung 3 Teilausschnitt ER-Modell

Das oben dargestellte ER-Modell zeigt einen Teilausschnitt. Das gesamte ER-Modell ist dem Anhang *A4 ER-Modell* zu entnehmen.

3.5 Relationales Datenmodell

Für das Relationale Datenbankmodell wurde das Er-Modell in ein Tabellarisches Datenmodell umgewandelt. In diesem sind die Entitätstypen und Attribute übersichtlich dargestellt. Es wird auch aufgezeigt welche Tabellen miteinander in Beziehung stehen.

Bei der Umsetzung wurde die dritte Normalform verwendet, um Redundanz zu reduzieren, Datenanomalien zu vermeiden und die referenzielle Integrität sicherzustellen.

Das entsprechende Diagramm kann dem Anhang *A5 Relationales Datenmodell* entnommen werden.

4 Implementierung

Die Eingabe wird über die Entwicklungsumgebung von IntelliJ IDEA Community Edition erarbeitet und mit der Programmiersprache Java eine Web-Applikation programmiert. Um die Daten abzulegen und zu speichern, wurde mithilfe von HeidiSQl und auf Basis von MariaDB eine Datenbank zur Verfügung gestellt, welche mithilfe von Hibernate mit der Web-App kommuniziert.

Bei Hibernate handelt es sich um ein Framework zur Abbildung von Objekten auf relationalen Datenbanken für die Programmiersprache Java. Mann bezeichnet es auch als ORM-Tool.

Für die Implementierung der Oberfläche wurde das Framework Vaadin benutzt, dieses bietet die Möglichkeit vorhandene Komponenten in den Quellcode zu übernehmen. Um das ganze anschaulicher darzustellen dient die folgende *Abbildung 4 Zusammenspiel Spring, Vaadin und Hibernate*. Die Add-Ons Vaadin und Hibernate werden zu Spring hinzugefügt und erleichtern die Frontendgestaltung und den Datenbankzugriff.

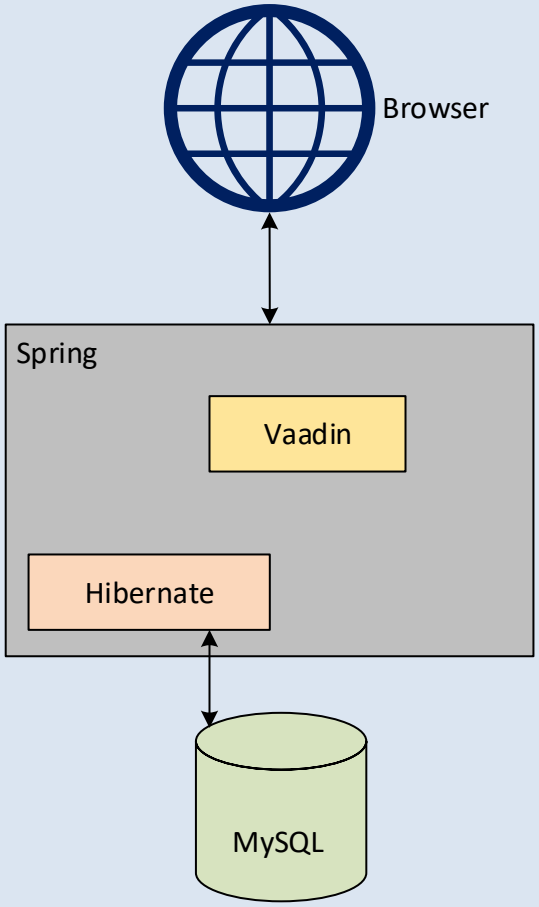


Abbildung 4 Zusammenspiel Spring, Vaadin und Hibernate

4.1 Erstellen der Datenbank

Durch das Framework Hibernate, welches Teil des Spring-Frameworks ist, wurde die Datenbankstruktur automatisch anhand der Klassen, welche in IntelliJ geschrieben wurden, auf HeidiSQl erzeugt.

4.2 Verbindung der Datenbank mit IntelliJ

Das Framework Hibernate übernimmt auch während der Laufzeit die Kommunikation zwischen der App und der Datenbank.

4.3 Implementierung der Benutzeroberfläche

Die Implementierung der Benutzeroberfläche anhand des Designs aus Kapitel *3.2 Layout Design* wurde komplett Java geschrieben. Dafür wurde ein weiteres Framework benutzt. Vaadin stellt verschiedene Komponenten zur Verfügung, die bequem in den Quellcode übernommen und angepasst werden können.

Die Welcome Page wurde mit dem Eingangsbereich der Praxis versehen. Am linken Rand befindet sich die Menüleiste, von wo aus alle Bereiche einfach zu erreichen sind. Die folgende *Abbildung 5 Welcome Page* zeigt sie Welcome Page in der finalen Version. Die weiteren Oberflächen werden im Rahmen der Kundendokumentation ausführlich beschrieben und deren Bedienung erklärt.

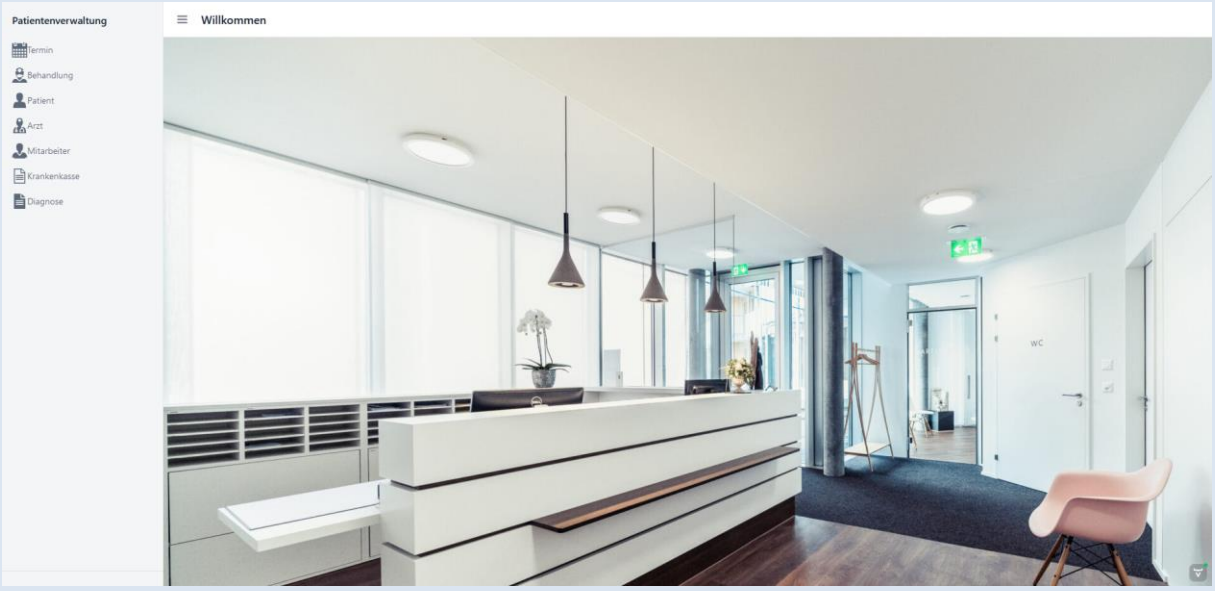


Abbildung 5 Welcome Page

4.4 Implementierung der Features Alle Patienten und Suchen

Ein wesentlicher Bestandteil der Web-Applikation soll es sein, dem Arzt bei einem Termin, die entsprechende Patientenakte anzuzeigen und zu ergänzen. Hierfür wurde ein Grid

genutzt. Um eine Zusammenführung der Klassen Patient, Arzt und Diagnose zu ermöglichen, wurde die Klasse Behandlung implementiert.
Die Klasse Behandlung kann dem Anhang *A6 Klasse Behandlung* entnommen werden.

Damit diese dann sich die benötigten Daten aus den einzelnen Tabellen holen und anzeigen kann, wurde für jede Klasse ein Repository geschrieben. In der Repository sind SQL-Abfragen, welche für die jeweilige Tabelle benötigt werden, definiert. Die einzelnen Oberflächen erhalten je nach Bedarf, welche Tabelle benötigt wird, die Repository in den Konstruktor.

Ein weiteres Feature ist es sich über die Suchfunktion einen bestimmten Datensatz anzeigen zu lassen. Dieses Feature ist bis auf Termin und Behandlung auf jeder Oberfläche vorhanden. Hierfür wurde ein SQL-Statement geschrieben, welches der jeweiligen Repository eingefügt und angepasst wurde.

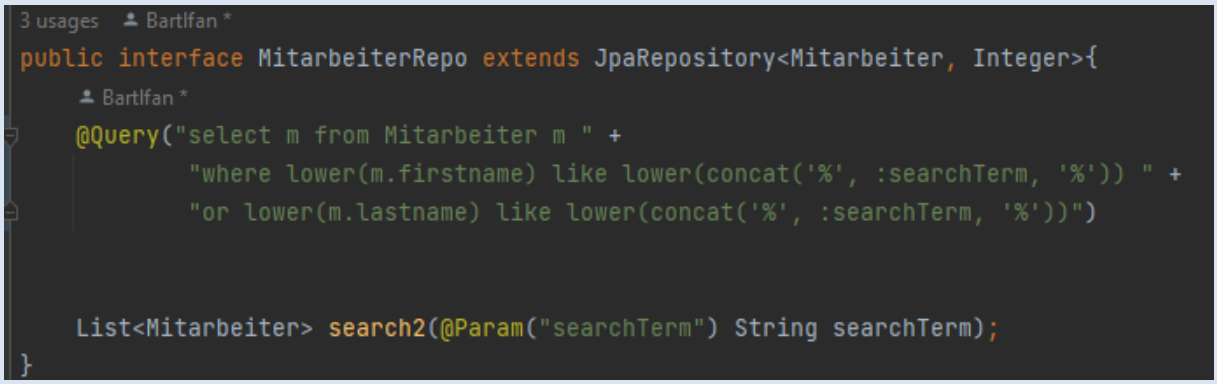


Abbildung 6 Query zum Suchen

Dieses ermöglicht nach einem einzelnen Buchstaben oder einer Buchstabenkette des gewünschten Patienten, Arzt, Diagnose oder Versicherung zu suchen.

Die folgenden Abbildungen zeigten, dass die Suchfunktion ihre Arbeit korrekt ausführt.

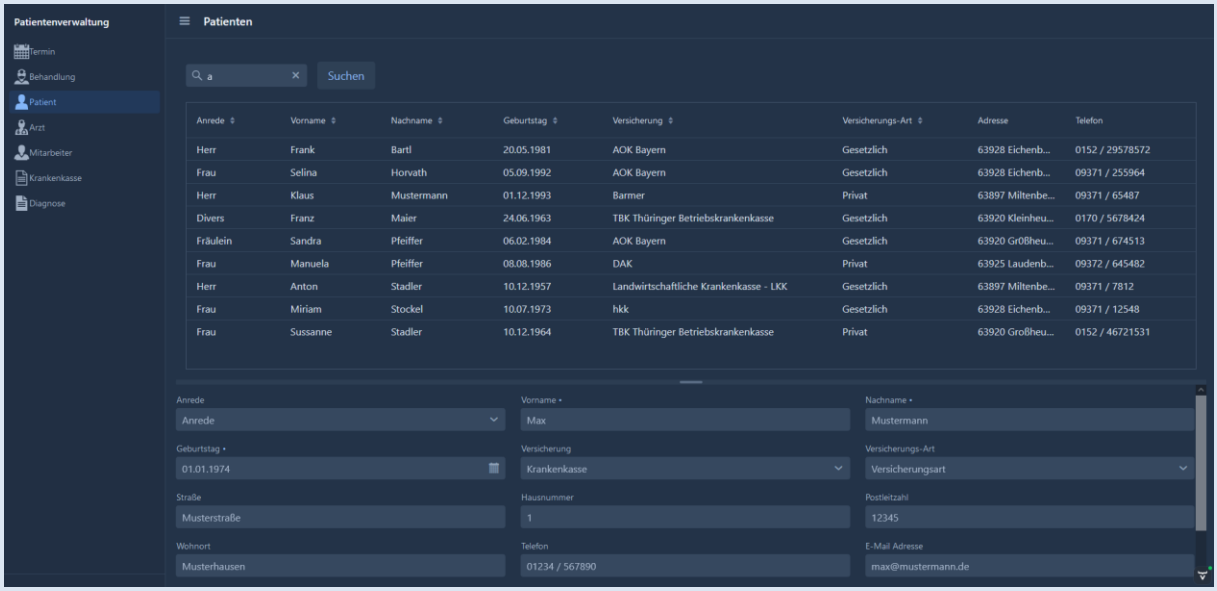


Abbildung 7 Suchergebnis mit Einzel Buchstaben

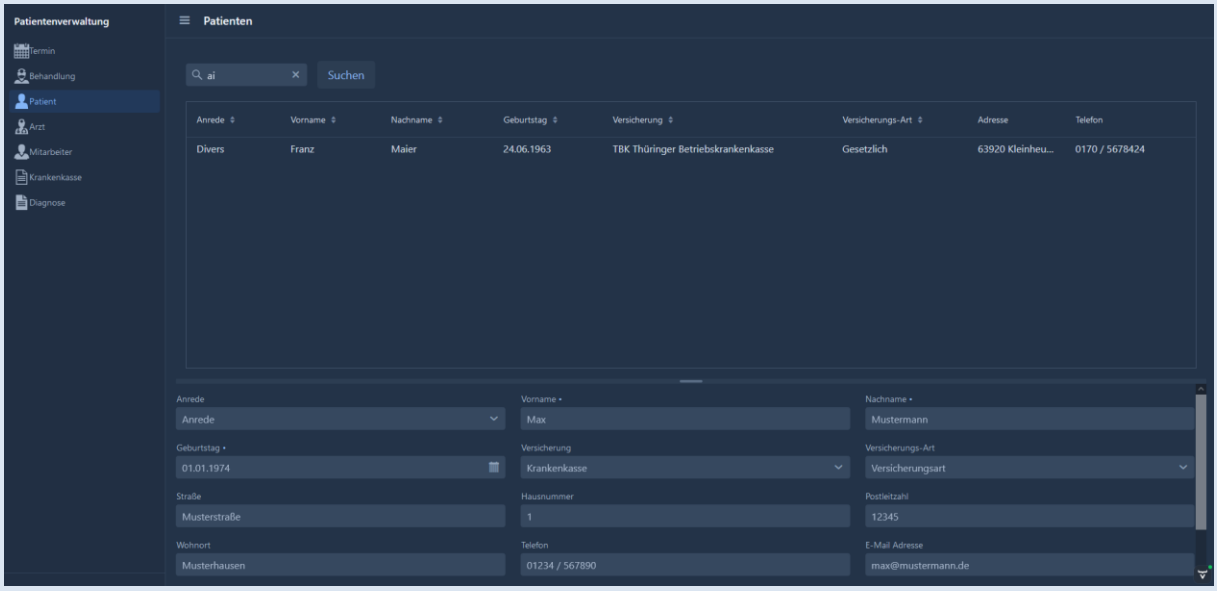


Abbildung 8 Suchergebnis mit Buschstaben Kette

4.5 Probleme bei der Implementierung

Während der Implementierungsphase kam es zu Probleme, die es zu lösen galt.

Probleme die aufgetreten sind:

- Exception bei Datenbank Anbindung.
- Elemente aus der Grid werden nicht in den Textfeldern dargestellt.
- Absicherung für die doppelte Terminvergabe funktioniert nicht.
- Suchfunktion ohne Funktion.

4.6 Lösungen der entstandenen Probleme

Die Exeption die bei der Anbindung der Datenbank aufgerufen wurde, war leicht zu beheben. In der Datenbank war bereits eine Bespieldatenbank hinterlegt, was zu einem Konflikt geführt hat. Die Beispieldatenbank wurde gelöscht und der Fehler somit behoben.

Bei der Problematik das Elemente der Grid nicht in den Textfeldern dargestellt wurden, hat es sich um simple Rechtschreibfehler bei der Benennung gehandelt. Was aber ein größerer Zeitaufwand war diese zu finden.

Die Probleme der Absicherung bei doppelter Terminvergabe und Suchfunktion, lag in der Schreibweise des SQL-Statement. Bei bisherigen Projekten konnte die Statements in der erlernten SQL-Syntax erfolgen. Hibernate benötigt aber für die korrekte Ausführung eine andere Syntax, diese muss den JPQL entsprechen.

5 Qualitätsmanagement

Im Kapitel Qualitätsmanagement werden die im Zuge der Implementierung durchgeführten Tests aufgelistet.

5.1. Systemtest

Für den Systemtest wurde ein Black-Box Test gewählt. Dieser beschränkt sich auf die Funktionalität und dass Verhalten der getesteten Anwendung, ohne zu wissen, wie sie intern funktioniert. Es sollen keine Fehler gefunden werden, sondern verstanden werden, wie die Anwendung funktioniert und welche Möglichkeiten sie bietet. Das Protokoll der durchgeführten Testest ist dem Anhang zu entnehmen.

5.2 Code-Review

Um die Qualität des geschrienen Codes zu gewährleisten, wurde in regelmäßigen Abständen ein Code-Review durch einen erfahrenen Entwickler durchgeführt.

5.3 Abnahme

Im Zuge der Abnahme wurde die Patienten- und Terminverwaltung mehrfach getestet und keine Fehler festgestellt. Die Anwendung wurde durch Herrn Dr. Klein abgenommen. Bevor es zur Abnahme gekommen ist, wurde bereits ein Black-Box-Test und Code Review durchgeführt.
Die im Black-Box-Test aufgetretene Auffälligkeiten wurden mit dem Kunden besprochen und Lösungsansätze präsentiert.

6 Fazit

Zu Beginn des Projektes stand der Autor vor der großen Herausforderung, wie er das ganze umsetzten soll. Es waren sehr viel Informationen zu bearbeiten, zu sortieren und zu verstehen. Wie können die Wünsche des Kunden umsetzten werden? Wie geht man am besten vor? Welche Features sind für eine reibungslose Funktion wichtig? Welche Features können später noch hinzugefügt werden.

Dank der guten Vorbereitung ist es dem Autor gelungen ein funktionierendes Programm zu entwickeln und umzusetzen. Die einzelnen Phasen des Projekts waren allesamt sehr interessant und haben das Wissen in jedem Bereich erweitert. Auch mit wenig Java Kenntnissen, ist es mir dank einer strukturierten Arbeitsweise, gelungen mein Projekt erfolgreich abzuschließen. Natürlich ist nicht immer alles ohne Probleme gelaufen. Wie im Kapitel *4.5 Probleme bei der Implementierung* zu sehen ist.
Bei dem Black-Box-Test sind noch ein paar gravierende Probleme zu Tage getreten, unter anderem, welche einen Autorisierten Zugang zur App und der Oberfläche Behandlung erlaubt. Diese wurden mit dem Kunden besprochen und zugesichert mit den denn nächsten Updates bis spätestens Frühjahr 2023 behoben zu sein.

6.1 Soll-/Ist-Vergleich

Die vom Kunden gewünschte Anforderungen sind bis auf den Autorisierten Zugang für die App selbst und die Oberfläche Behandlung allesamt erfüllt. Die in den Funktionstests aufgefallenen Mängel, wurden bis auf die oben genannten Mängel beseitigt. Die Mitarbeiter des Kunden wurden geschult und das Programm wurde erfolgreich eingeführt. Der Kunde ist mit der Leistungserbringung vollends zufrieden und erwartet freudig neue Updates, die bereits besprochen und in Auftrag gegeben wurden. In den folgenden *Tabelle 8 Gesamtkosten Soll-/Ist-Vergleich* und *Tabelle 9 Soll-/Ist-Vergleich Zeit*, werden die geplanten Kosten und die geplante Zeit, mit den tatsächlichen Daten verglichen und ausgewertet.

Gesamtkosten	**Geplant**	**Tatsächlich**
Stundenlohn	95,00 €	95,00 €
Arbeitszeit	79 h	80 h
Kosten o MwSt.	7.505,00 €	7.600,00 €
MwSt. 19%	1.425,95 €	1.444,00 €
Kosten inkl. MwSt.	**8.930,95 €**	**9.044,00 €**

Tabelle 8 Gesamtkosten Soll-/Ist-Vergleich

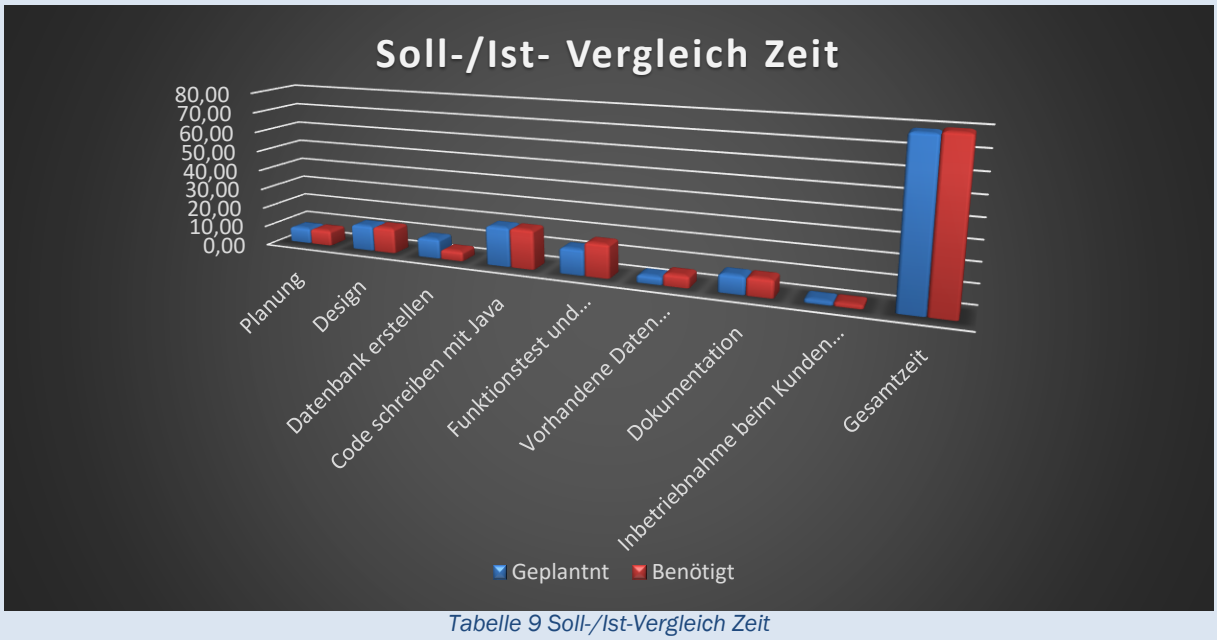


Tabelle 9 Soll-/Ist-Vergleich Zeit

Wie in den Tabellen zu erkennen ist, hat sich das Projekt in der Gesamtzeit um eine Stunde verzögert. In der Projektphase Datenbank erstellen, wurde im Vergleich zur Planung nur die Hälfte der Zeit benötigt, da durch das Framework Hibernate die Datenbank automatisch anhand der Klassen erstellt wurde. Dafür wurde im Bereich Funktionstest und Fehlerbehebung die Zeit um 4 h und bei Vorhandene Daten einpflegen um 2 h überschritten.

6.2 Lessons Learned

Im Laufe des Projekts wurde in allen Bereichen etwas dazu gelernt. Eine der wichtigsten Erkenntnisse war es, dass es beim Programmieren nicht die eine Lösung gibt. Auch der Umgang mit IntelliJ und HeidiSQL wurde gefestigt. Im Vergleich mit anderen Projekten, die in Visual Studio mit C# umgesetzt wurden, war die neue Oberfläche von IntelliJ und der Programmiersprache Java zu Beginn eine echte Herausforderung die es zu bewältigen galt. Durch eine gute Planung und setzen der Teilziele zu Beginn des Projektes, war ein strukturiertes Arbeiten dennoch möglich. Die Kommunikation mit dem Kunden war sehr hilfreich, da die eigenen Vorstellungen doch öfters andere waren als die des Kunden. Bei der Dokumentation der Zeiten ist aufgefallen, dass wenn man die Trennung der einzelnen Arbeitsschritte nicht sorgfältig trennt, kommt es bei der Zeitauswertung zu Problemen. Abschließend wird festgestellt, dass es noch viel zu lernen gibt.

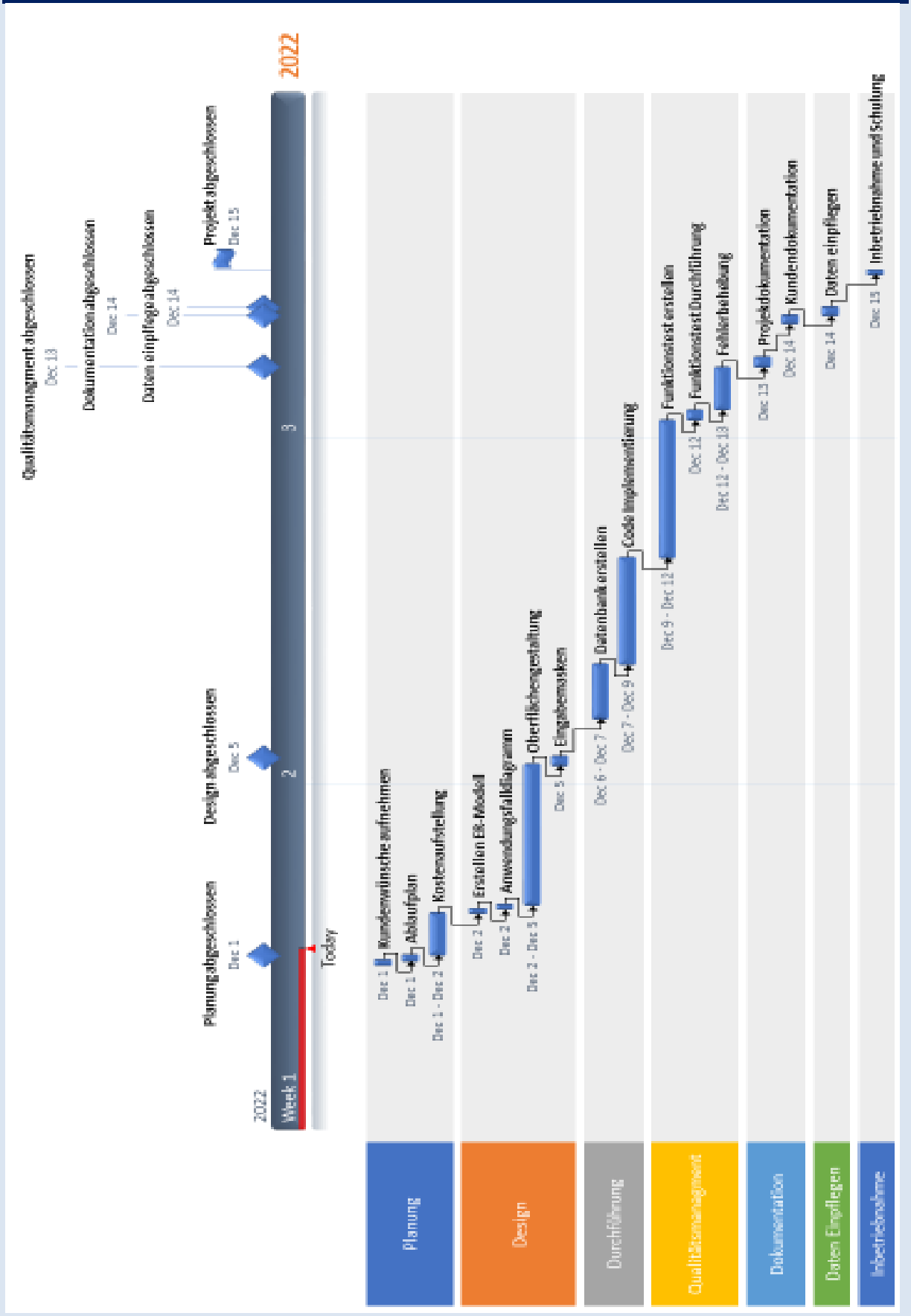
6.3 Ausblick

Obwohl schon einige Features umgesetzt worden sind, gibt es noch ein großes Entwicklungspotenzial. Eine wichtige Ergänzung, die mit dem Kunden auch bereits besprochen ist, soll es sein, eine Anmeldemaske zu implementieren, wo sich jede Arzthelferin und jeder Arzt mit seinen eigenen Daten anmelden kann. Zudem soll der Bereich

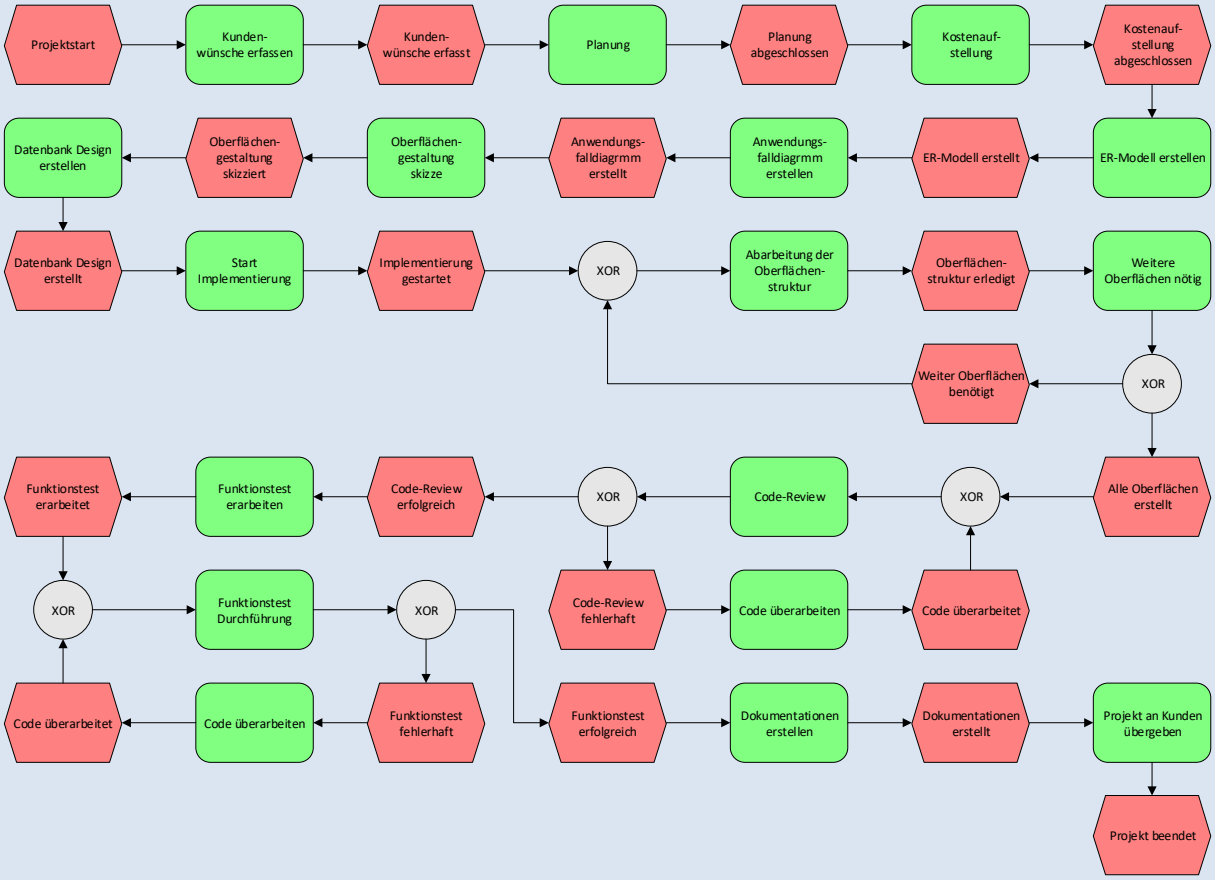
Behandlung nur für die Ärzte zugänglich sein. Weiter soll es die Möglichkeit geben, bei den Patientendaten noch Röntgenbilder und Befunde von Externen Ärzten hinzuzufügen. Diese sollen dann auch bei einem Termin für den Arzt in der Oberfläche Behandlung aufrufbar sein.

Anhang

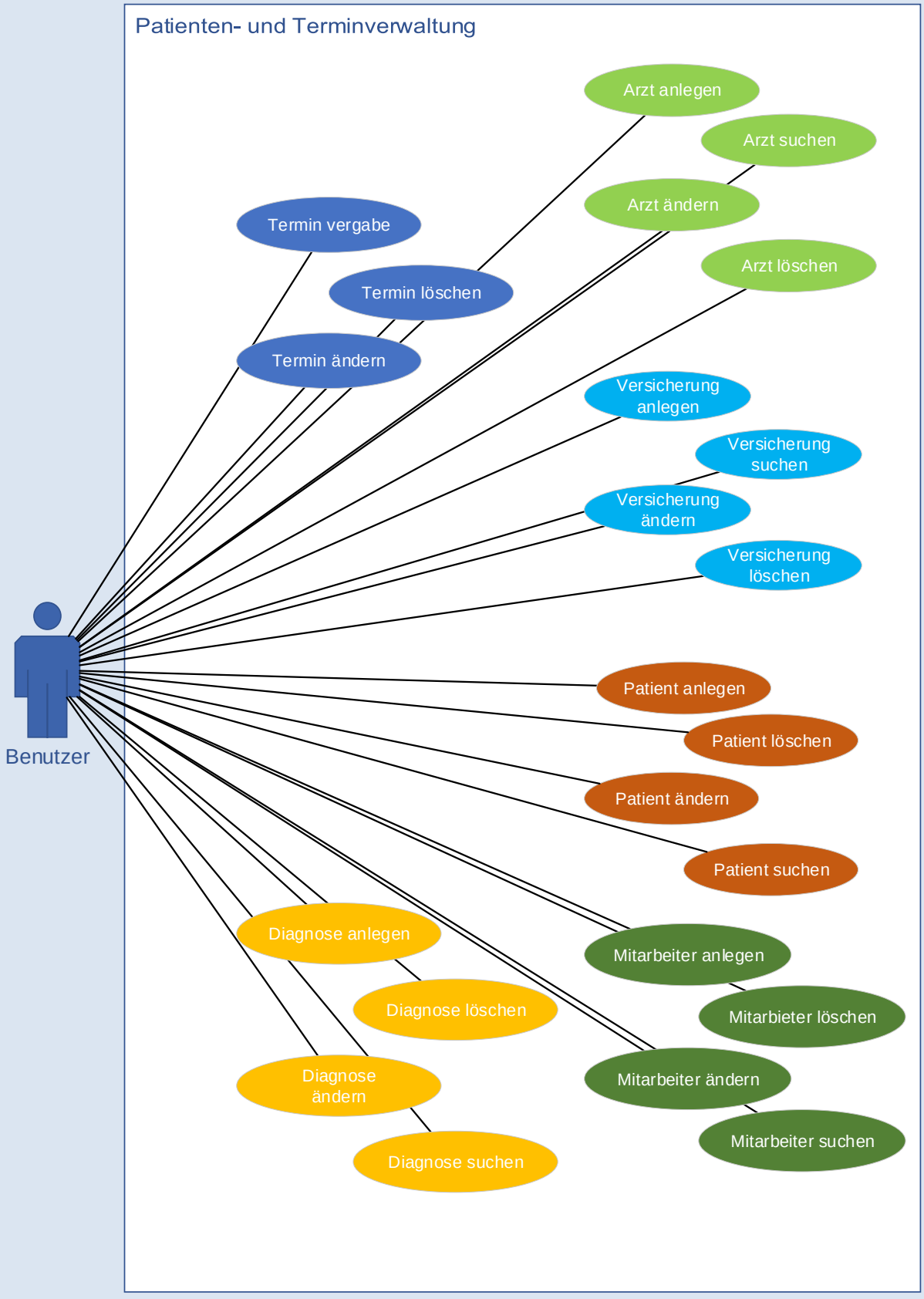
A1 Gantt-Diagramm



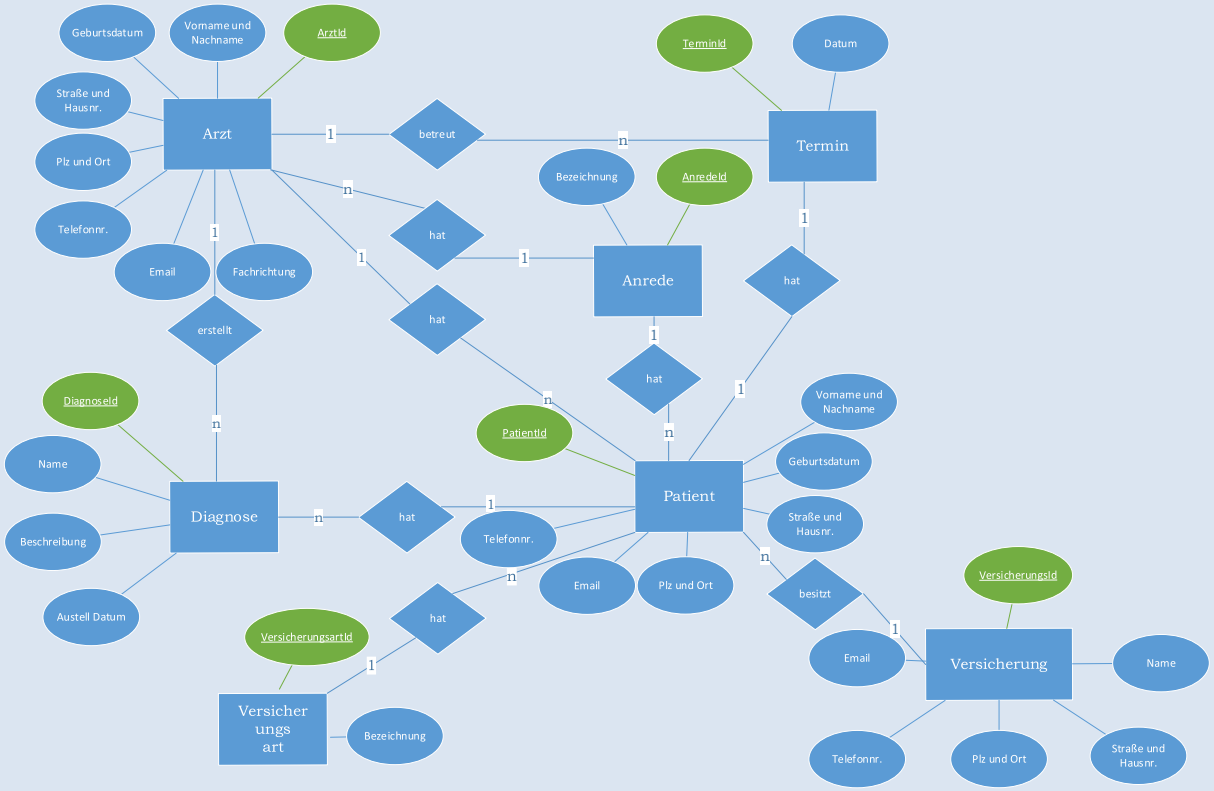
A2 EPK-Modell



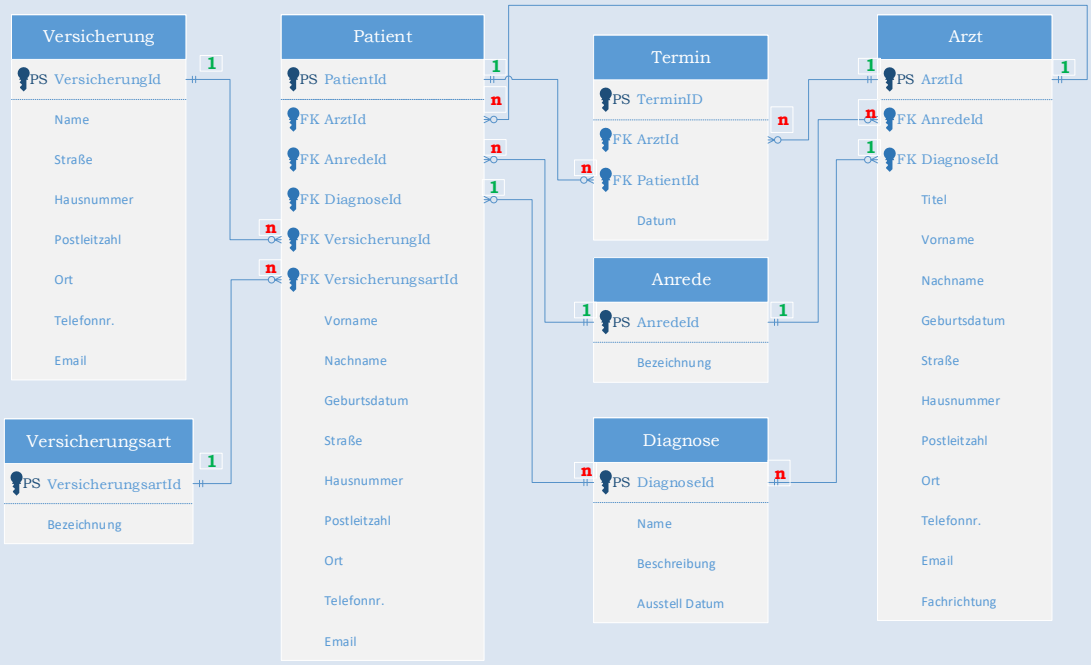
A3 Use-Case-Diagramm



A4 ER-Modell



A5 Relationales Datenmodell



A6 Klasse Behandlung

```java
@Data
@Entity
@Table(name = "`behandlung`")
public class Behandlung {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "`behandlung_id`", nullable = false)
    private int behandlungId;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "`patient_id`", nullable = true, columnDefinition = "INT")
    private Patient patient;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "`diagnose_id`", nullable = true, columnDefinition = "INT")
    private Diagnose diagnose;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "`arzt_id`", nullable = true, columnDefinition = "INT")
    private Arzt arzt;

    @Column(name = "`create_date`")
    private LocalDateTime createDate;

    @Column(name = "`note`")
    private String note;

    1 usage  BartIfan
    @Transient
    public String getDateAsString (){
        return getCreateDate() != null ? getCreateDate().format(DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm")): "";
    }
}
```

A7 System Test

Black-Box-Test

Projekt:

Patienten- und Terminverwaltung

Getestete Funktionen	Resultat
01 Allgemeine Übersichtlichkeit. Bedienelemente leicht zu finden und zu verstehen, was diese tun.	IO
02 Jede der einzelnen Oberflächen werden fehlerfrei dargestellt	IO
03 Termine können angelegt werden	IO
04 Termine ändern und löschen	IO
05 Warnung vor dem Löschen eines Datensatzes	NIO
06 Absicherung für versehentliches Löschen	IO
07 Splitt Layout lässt sich verschieben	IO
08 Termine werden angezeigt	IO
09 Pflichtfelder funktionieren	IO

10 Termine absichern gegen eine doppelvergabe	IO
11 Die Tests 03 -08 sind auch für die Oberflächen Behandlung, Patient, Arzt, Mitarbeiter, Krankenkasse und Diagnose durchgeführt worden.	IO
12 Suchfunktion	IO
13 Login Funktion	NIO

Fehler, Wünsche, Notizen	Arbeitsstatus
Warnung vor Löschen eines Datensatzes.	Beheben bis zu Abnahme
Login Bereich für die Anwendung an sich und speziell für die Behandlungsoberfläche	Wird mit dem nächsten Update behoben.
Infobox nach Speichern und löschen	Beheben bis zu Abnahme
Möglichkeit Befunde von Externen Ärzten hinzuzufügen	Update erfolgt im Frühjahr 2023

A8 Abnahme Protokoll