Patienten- und Terminverwaltung

Projektbericht

Projekt Agiles Software Engineering

26.09.2025

Anna Kickartz

Matrikelnr.: UPS10747892

Dr. Sheikh Radiah Rahim Rivu

Inhaltsverzeichnis

1. Einleitung

Der vorliegende Bericht dokumentiert die Entwicklung einer Softwarelösung zur effizienten Verwaltung von Patienten- und Termindaten in einer Praxisumgebung.

1.1 Problemstellung

Die Projektidee entstand aus der Erkenntnis, dass die Organisation und Pflege von Patientendaten in vielen medizinischen Einrichtungen nach wie vor überwiegend manuell oder mit unstrukturierten Hilfsmitteln erfolgt. Dies führt häufig zu Inkonsistenzen, erhöhtem Verwaltungsaufwand und einem erhöhten Risiko für Fehler bei der Terminplanung und Datenpflege. Zudem ist die Einarbeitung von neuem Personal unter diesen Voraussetzungen nicht unbedingt nur auf den fachlichen Aspekt gerichtet, sondern zum Teil auch auf die Handhabung der technischen Aspekte von Termin- und Patientenverwaltung. Vor diesem Hintergrund bestand die zentrale Problemstellung darin, eine benutzerfreundliche, funktionale und zugleich flexible Software zu entwickeln, die sowohl die Datenhaltung als auch die Terminorganisation in einem System zusammenführt.

1.2 Projektidee und -ziel

Die Projektidee zielte darauf ab, ein integriertes Anwendungssystem zu schaffen, das alle wesentlichen Funktionen einer Praxisverwaltung abbildet: die Erfassung neuer Patienten, die Bearbeitung bestehender Patientendaten, die Anlage, Verwaltung und Löschung von Terminen sowie die Möglichkeit zur gezielten Suche und Übersicht über Patienten und Termine. Das alles sollte im Idealfall von mehreren Geräten abrufbar sein.
Die Zielsetzung des Projekts war sowohl technischer als auch organisatorischer Natur. Technisch sollte eine robuste Softwarearchitektur realisiert werden, die den Zugriff auf Patientendaten und Termine effizient ermöglicht und gleichzeitig eine intuitive Benutzeroberfläche bereitstellt. Organisatorisch stand die agile Vorgehensweise im Vordergrund, um den Entwicklungsprozess kontinuierlich überprüfen, anpassen und optimieren zu können. Durch die Umsetzung in mehreren aufeinanderfolgenden Sprints sollte sichergestellt werden, dass bereits frühzeitig ein lauffähiges Produkt vorliegt, das sukzessive um zusätzliche Funktionalitäten erweitert werden kann.

Die Methodik des Projekts orientierte sich an Konzepten des agilen Projektmanagements, insbesondere an Scrum, angepasst für die Arbeit im Einzelprojekt (Miniscrum). Dies erlaubte eine strukturierte Planung, eine kontinuierliche Evaluation und die systematische Priorisierung der Aufgaben. Der Projektbericht gibt in diesem Zusammenhang nicht nur den Entwicklungsprozess und die technischen Entscheidungen wieder, sondern reflektiert auch die angewandten Methoden, Herausforderungen und Lösungsansätze sowie die daraus gewonnenen Erfahrungen.

In der vorliegenden Arbeit werden zunächst die angewandten Methoden und die Planung des Projekts dargestellt, anschließend die Durchführung und die Ergebnisse der einzelnen Sprints erläutert. Abschließend erfolgt eine kritische Reflexion des Projekts, der methodischen Vorgehensweise sowie der erzielten Ergebnisse. Ziel ist es, die Entwicklung nachvollziehbar zu dokumentieren und zugleich Erkenntnisse für die zukünftige Umsetzung ähnlicher Softwareprojekte zu gewinnen.

2. Methodik

2.1 Methodiken der agilen Entwicklung

Agile Entwicklungsmethoden haben gemein, dass sie eine iterative Arbeitsweise unterstützen, in der Aufgaben und Vorgehen kontinuierlich angepasst und verbessert werden. Besonders bekannte Vertreter dieser Vorgehensweise sind Scrum, Kanban und Scrumban, die daher zu Beginn des Projekts näher beleuchtet und hinsichtlich ihrer Eignung verglichen wurden. Hierbei zeichnet sich Kanban vor allem durch die visuelle Darstellung von Prozessen und Zielen und etwas mehr Flexibilität in den zeitlichen Abläufen aus; es eignet sich gut für die dynamischere und schnelle Arbeit in einem Team, in dem ein gewisser Fluss aufrechterhalten werden soll und Aufgaben unterschiedlich lange dauern können. Dagegen legt die Scrum-Methode höheren Wert auf klar abgesteckte Ziele, feste zeitliche Vorgaben, geplante Sprints sowie regelmäßig stattfindende Stand-Ups und Reviews. Scrumban stellt eine Kombination aus beiden Methodiken dar.
Da das Projekt allein umgesetzt wurde, war keine kontinuierliche Teamkoordination erforderlich. Stattdessen lag der Fokus auf der klaren Nachvollziehbarkeit bereits erreichter Ziele und der Übersichtlichkeit der noch anstehenden Aufgaben, wofür Scrum besser geeignet ist. Auch wegen der noch fehlenden praktischen Erfahrung im agilen Projektmanagement und in der Softwareentwicklung wurde schließlich die Entscheidung getroffen, die Grundlagen von Scrum noch einmal auf die Arbeit an einem Einzelprojekt mit „Mini-Scrum“ anzuwenden.

2.2 Rollen

Bei einem klassischen Scrum-Ansatz erfolgt die Verteilung der Aufgaben anhand folgender Rollen:
- Der Product Owner ist verantwortlich für den Product Backlog bzw. das Festlegen und die Priorisierung der dort festgelegten Aufgaben.
- Der Scrum Master fungiert als Dirigent, er organisiert und unterstützt das Team, organisiert und führt die regelmäßigen Meetings und kümmert sich um die Einhaltung der Scrum-Regeln und etwaige Hindernisse.
- Das Entwicklungsteam verantwortet die Umsetzung und bearbeitet eigenständig die

übertragenen Aufgaben.

Im Projekt wurde die Anwendung auf die Umsetzung als Einzelprojekt angepasst und alle Rollen in einer Person verkörpert. Diese Anpassung erforderte eine strukturierte Selbstorganisation und wiederum konstante Reflexion, vermittelte aber gleichzeitig auch ein tieferes Verständnis der einzelnen Rollen. Unter anderem wurde hierzu regelmäßig schriftlich die jeweils eingenommene Perspektive festgehalten und anhand räumlicher und kurzer zeitlicher Distanzierung für eine klare Trennung der Positionen gesorgt.

2.3 Vorgehensweise anhand Scrum

In einem klassischen Team, das mit Scrum arbeitet, werden Sprints gemeinsam geplant, tägliche kurze Stand-Ups zur Kommunikation des aktuellen Stands gehalten und anhand von Reviews und Retrospektive der Fortschritt und mögliche Verbesserungen besprochen.
Diese Grundsätze wurden für dieses Projekt wie folgt angepasst:
- In der Sprintplanung wurden – für jeden Sprint einzeln – die Einträge im Product Backlog nach ihrer Priorität geordnet und anhand dessen die jeweiligen Ziele definiert.
- Die täglichen Stand-Ups erfolgten zur Vermeidung von Abschweifungen schriftlich anhand klar definierter Fragestellungen, die innerhalb von etwa 5-10 Minuten knapp und klar beantwortet wurden.
- Sprint Review und Retrospektive: Am Ende eines Arbeitstags sowie am Ende jedes Sprints wurde eine schriftliche Bestandsaufnahme verfasst, in der außerdem bereits ein erstes Arbeitsziel für den nächsten Tag/den nächsten Sprint formuliert und reflektiert wurde, welche Herausforderungen zu bewältigen waren und was gut/weniger gut funktioniert hat.

Dieses Vorgehen ermöglichte eine strukturierte und reflektierte Arbeitsweise, in der Fortschritte und noch ausstehende Aufgaben jederzeit nachvollziehbar erschienen und das Projekt kontinuierlich verbessert werden konnte.

3. Planung

3.1 Anforderungsanalyse

In der Anforderungsanalyse wurden die wichtigsten Funktionen und Rahmenbedingungen des Projekts identifiziert. Zunächst wurde definiert, welche Anforderungen die Anwendung mindestens erfüllen sollte; sowohl für die Patienten- als auch für die Terminverwaltung wurden die CRUD-Funktionen festgelegt, zudem wurden die nicht-funktionalen Anforderungen ermittelt und in die

Analyse eingebaut. So konnten die wichtigsten Anforderungen priorisiert und eine stabile Grundlage für das weitere Vorgehen gebildet werden. Daraus ergab sich die folgende:

Anforderungsanalyse für die Patienten- und Terminverwaltung

1. Zielgruppe

- Arzt und medizinisches Personal (als Nutzer)
- Kein Endbenutzer-Interface für externe Nutzer notwendig

2. Funktionale Anforderungen

- Patientenverwaltung

- **Patienten anlegen:** Name, Geburtsdatum, Telefonnummer, Adresse, Krankenkasse
- **Patienten anzeigen:** Liste aller Patienten
- **Patienten suchen:** Suche nach ID
- **Patienten bearbeiten:** Änderungen an bestehenden Patientendaten
- **Patienten löschen:** Entfernen eines Patienten aus der Datenbank

- Terminverwaltung

- **Termin anlegen:** Datum/Uhrzeit, PatientID (aus bestehender Patientenauswahl), Beschreibung
- **Termine anzeigen:** Liste aller Termine
- **Termin suchen:** Suche nach Datum/Zeit
- **Termin bearbeiten:** Änderungen an bestehendem Termin
- **Termin löschen:** Entfernen eines Termins

4. Nicht-funktionale Anforderungen

- **Benutzerfreundliche GUI:** Einfaches Fenster mit Buttons und Listen
- **Datenhaltung:** Speicherung der Daten im Arbeitsspeicher (keine Persistenz notwendig, um Komplexität zu reduzieren), ggf später in Datenbank
- **Fehlerbehandlung:** Eingabefehler abfangen (z.B. leere Felder, ungültige Daten)
- **Einfachheit:** Fokus auf grundlegende Funktionen, keine komplexen Features

5. Technische Rahmenbedingungen

- Programmiersprache: Java
- GUI: Swing (grundlegend vorhanden)
- Nutzung von Arrays, Kontrollstrukturen, Exception Handling, Interfaces
- Datenbank: SQLite

3.2 Erstellen des Product Backlogs

Anhand der Anforderungsanalyse wurde ein Product Backlog erstellt, um alle erforderlichen Anforderungen und Aufgaben zusammenzutragen. Hierbei wurde zunächst anhand eines Brainstormings jede Möglichkeit notiert, um dann in einem nächsten Schritt Doppeltes oder

Überflüssiges zu streichen und zusammenzustellen, bis eine übersichtliche und vollständige Zusammenstellung der umzusetzenden Aufgaben entstanden war. Hierzu gehörte unter anderem das Entwerfen einer sauberen Software-Architektur, das erste Recherchieren und Festlegen der genutzten Speicherung von Einträgen und das Implementieren der Modellklassen „Patient“ und „Termin“. Die Einträge wurden anhand des ersten Ziels priorisiert, zunächst eine funktionsfähige Grundlagenanwendung zu erstellen.

3.3 Sprintplanung und Zieldefinition

Auf Grundlage des Product Backlogs wurden insgesamt vier jeweils einwöchige Sprints geplant und festgelegt, welcher Teil der Aufgaben bis zu welchem Zeitpunkt spätestens umgesetzt sein sollte. Dazu wurde für jeden Sprint ein klares Ziel definiert, dessen Einhaltung überprüft werden konnte – in etwa: „Am Ende des ersten Sprints sollen die Modellklassen und die CRUD-Funktionen implementiert sein und ohne Fehler laufen“. Die Aufgaben aus dem Backlog wurden anhand ihrer Eignung für dieses Ziel und anhand ihres erwarteten zeitlichen Umfangs ausgewählt und nach dieser Auswahl noch einmal überprüft, ob Ziel, Aufgaben und zeitlicher Anspruch zusammenpassen.
Insbesondere wurde darauf geachtet, dass die Anwendung Stück für Stück erweiterbar blieb und Stabilität vor der Umsetzung möglicher zusätzlicher Features priorisiert wurde. Das Feedback aus den Selbsttests sollte berücksichtigt und eine kontinuierliche Verbesserung sichergestellt werden. Diese Vorgehensweise ermöglichte die strukturierte Umsetzung des Projekts und die systematische Dokumentation des Fortschritts.

4. Durchführung

Für jeden Sprint wurden Ziele festgelegt und Aufgaben aus dem Product Backlog übernommen, deren Erreichen am Ende jeweils reflektiert wurde.

4.1 – Sprint 1: Planung, Grundstruktur und erste Funktionen

Im ersten Sprint lag der Fokus zunächst auf der Projektplanung, Recherche und dem Erstellen der Grundstruktur der Anwendung. Dazu wurden die Anforderungen systematisch analysiert und daraus das Product Backlog erstellt. Die darin festgehaltenen Kernfunktionalitäten (Erstellen, Anzeigen, Bearbeiten und Löschen von Patienten- und Termineinträgen) wurden priorisiert. Zudem wurde festgelegt, welche nicht-funktionalen Anforderungen erforderlich waren: Eine einfache Benutzeroberfläche, die Fehlerbehandlung und die Form der Datenspeicherung. Hierbei wurde der Fokus zunächst auf eine einfache, lokale Speicherung gelegt, die eventuelle spätere Umstellung

auf Datenbanken wurde offen gehalten und die Anwendungsarchitektur entsprechend dahingehend aufgebaut, dass sie einfach zu erweitern sein sollte. So sollte es ermöglicht werden, die Kernlogik zunächst unabhängig von einer möglichen späteren Datenbankstruktur zu entwickeln und zu testen. Das Ziel war es, den ersten Entwurf einer grundlegenden, aber funktionsfähigen Anwendung zu erhalten.
Im nächsten Schritt wurde auf dieser Basis mit der Implementierung der grundlegenden Modellklassen begonnen, also „Patient“ und „Termin“. Zur Verwaltung der erforderlichen CRUD-Operationen wurden dann die Serviceklassen „PatientVerwaltung“ und „TerminVerwaltung“ erstellt. Nach der Implementierung der Methoden für das Erstellen, Anzeigen, Bearbeiten und Löschen der jeweiligen Einträge wurde ein erster manueller Testlauf durchgeführt, eventuelle Fehler identifiziert und bis zu diesem Stand behoben. Auf Basis der bis dahin fehlerfrei laufenden Anwendung wurde ein einfaches GUI implementiert, das die Auflistung der Patentien- und Termineinträge und erste einfache Interaktionen ermöglichte.
Der Sprint endete mit einem Abgleich des definierten Ziels und einer Reflektion der genutzten Arbeitsweise. Zudem wurden ein erster Arbeitsschritt und ein erstes Etappenziel für die neue Woche festgelegt, um den Einstiegsaufwand für den nächsten Sprint gering zu halten.

4.2 – Sprint 2: Vertiefung, Design, Funktionserweiterung und Verknüpfung

Im zweiten Sprint lag der Fokus und das Ziel auf der Planung und Integration einer relationalen Datenbank.
Insbesondere wurde die Speicherung über eine Datenbank wegen des größeren Praxisbezuges eingerichtet, da auch die Arbeit in einem echten medizinischen Umfeld die gleichzeitige Nutzung der Anwendung auf mehreren Geräten erfordern würde und diese nur mit lokaler Speicherung nicht möglich gewesen wäre. Anhand der Größe der Anwendung, wegen der einfachen Einrichtung und da hier keine separate Serverinstallation erforderlich ist, fiel die Entscheidung für ein möglichst geeignetes Datenbank-Management-System zugunsten von SQLite aus; in einer größeren Anwendung wären PostgreSQL oder MySQL wegen der Optimierung für parallele Nutzung vorzuziehen.
Im nächsten Schritt wurden die Datenbankverknüpfung und die beiden Repository-Klassen „PatientDB“ und „TerminDB“ eingerichtet und die Verknüpfung zu den Serviceklassen hergestellt. Die Anpassung der Serviceklassen erfolgte unter konstanter manueller Testung, um ihre grundsätzliche Funktionsfähigkeit sicherzustellen. Etwaige Fehler wurden dokumentiert und, soweit möglich, sofort behoben, ansonsten zur Behebung im ersten Schritt im dritten Sprint vorgesehen. Auch der zweite Sprint endete mit einem Review der erreichten Ziele, einer Überprüfung der bisherigen Arbeitsmethoden sowie einem kurzen Ausblick auf den dritten Sprint. Die gewählte Vorgehensweise erwies sich in diesem Abschnitt als solide Basis für die weitere Entwicklung der Anwendung, insbesondere das Sicherstellen der Erweiterbarkeit und die

konstanten Tests ermöglichten ein kontrolliertes und gleichzeitig flexibles Arbeiten und wurde daher auch für den dritten Sprint so fortgeführt.

4.3 – Sprint 3: Überarbeitung, Nutzerführung, Systemtests & Fehlerbehandlung

Im dritten Sprint wurden plangemäß zuerst weitere manuelle Tests durchgeführt und die verbliebenen Fehler aus der Integration der Datenbank beseitigt. Nach der Durchführung weiterer manueller und auch automatisierter Tests wurden die noch ausstehenden Aufgaben ein weiteres Mal geprüft und hinsichtlich ihrer Umsetzbarkeit noch einmal priorisiert. Das GUI war vor der Umstellung der Datenspeicherung durch SQLite entfernt worden, um zunächst sicherzustellen, dass die Kernlogik auch nach dieser Umstellung funktionierte. Somit war die Anpassung und erneute Verknüpfung des GUI einer der zentralen Aufgaben für diesen Sprint, ebenso die Implementierung jeweils einer Suchfunktion in der Patienten- und in der Terminliste, die für den Anfang des vierten Sprints vorgesehen wurde. Unter diesen Aspekten wurde die angedachte Einrichtung einer Authentifizierungsfunktion, anhand derer sich die Nutzer der fiktiven Arztpraxis hätten registrieren und anmelden können, verworfen. Ausschlaggebend für diese Entscheidung war der hohe Zeitaufwand bei einer verhältnismäßig hohen Fehleranfälligkeit; Richtlinie für das Projekt war von Anfang an, dass nach jedem Sprint eine funktionsfähige Kernanwendung vorliegen sollte, bevor zusätzliche Features implementiert werden.

Nach Anpassung des GUI wurde im Rahmen mehrerer manueller Tests die Funktionalität sämtlicher Kernfunktionen sichergestellt.

Das iterative Vorgehen in diesem Sprint erlaubte es, auf unvorhergesehene Herausforderungen flexibel zu reagieren, beispielsweise Verzögerungen bei der Testdurchführung oder Änderungen in der Datenbankstruktur. Abschließend wurde das Review genutzt, um die Ergebnisse zu evaluieren und Aufgaben für den vierten Sprint zu priorisieren.

4.4 – Sprint 4: Feinschliff, Dokumentation, Abschluss

Im vierten Sprint erfolgte zunächst eine Bestandsaufnahme der erreichten Ziele sowie der noch ausstehenden Aufgaben. Priorität hatte die Einrichtung der Suchfunktion sowie das Sicherstellen der Funktionalität der gesamten Anwendung durch letzte Tests. Die iterative Vorgehensweise erlaubte es dabei, kleine Anpassungen unmittelbar in die laufende Entwicklung einzubringen. In diesem Zusammenhang wurden auch die Entscheidungen getroffen, die Suche nach Terminen zunächst nur anhand des Datums und der Uhrzeit zu ermöglichen; da die Suche zurzeit noch nur auf ein Ergebnis ausgelegt ist, ermöglicht dieses Vorgehen noch das präziseste Ergebnis.

Anhand des konstant geführten Entwicklungstagebuchs wurden in diesem letzten Sprint tägliche Abschluss-Reviews geführt. Insbesondere wurden jeweilige Tagesziele definiert, deren Einhalten das Befolgen des Zeitplans sicherstellte, sowie die Aufgabenliste abgeglichen, um an den passenden Punkten eventuelle noch offene Fragen beantworten zu können.

5. Reflexion

Im folgenden Abschnitt werden die Arbeitsweise, Herausforderungen, Lösungsansätze und Erfolge reflektiert und die eigenen Fortschritte analysiert. Daraus sollen im Sinne der agilen Vorgehensweise auch mögliche Verbesserungen für das nächste Projekt dieser Art gezogen werden können.

5.1 Herausforderungen und Lösungsansätze

Im Lauf der Arbeit an dem Projekt traten diverse Herausforderungen auf, die sowohl organisatorischer als auch technischer Natur waren. Insbesondere die Integration der und die Arbeit mit der Datenbank sowie die Implementierung des GUI stellten zentrale technische Herausforderungen dar. Mit der Einführung der relationalen Datenbank mussten Datenkonsistenz und Synchronisation zwischen GUI, Serviceklassen und Datenbank sichergestellt werden.
Um dieser Situation entgegenzuwirken, wurde konstant und sehr kleinschrittig manuell getestet. So sollte jederzeit nachvollziehbar sein, bis zu welchem Entwicklungsschritt ein Fehler nicht auftrat, um den kritischen Zeitpunkt bestmöglich eingrenzen zu können. Zudem galt die Vorgabe, dass vor der Fehlerfreiheit des bisher entwickelten Projekts keine zusätzlichen Aufgaben angegangen werden.

Eine organisatorische Herausforderung stellte die Entscheidung über zusätzliche Features dar, insbesondere die geplante Authentifizierungsfunktion. Nach Abwägung des Aufwands und der verfügbaren Zeit wurde entschieden, dieses Feature nicht umzusetzen, um die Kernfunktionalität nicht zu gefährden. Stattdessen wurde das Augenmerk auf die saubere Implementierung der Suchfunktionen für Patienten und Termine implementiert, die den praktischen Nutzen der Anwendung erhöhen und damit Vorrang hatten.
Eine weitere Herausforderung in der Organisation bestand in der Organisation an sich, also in der Planung und Strukturierung der Arbeit innerhalb eines begrenzten Zeitrahmens bei gleichzeitig hoher Komplexität der Aufgaben. Durch die Einzelarbeit entfiel die Arbeitsteilung und damit auch die gelegentlichen äußeren Impulse, wie sie in einem Team üblich sind. Daher mussten insbesondere alle Aufgaben eigenständig priorisiert und die Sprints kontinuierlich angepasst werden.
Der Lösungsansatz bestand hier darin, konsequent die Prinzipien von (Mini-)Scrum anzuwenden: Zum einen wurden tägliche Standups mit Personen von außerhalb gehalten, in denen in etwa 5 Minuten knapp der kommende Tagesplan sowie die Ziele umrissen wurden. Zum anderen wurde konstant und täglich schriftlich nachverfolgt und sowohl die Reflexionsphasen als auch das

Erreichen der jeweiligen Ziele festgehalten. Auf diese Weise wurden Abschweifungen in Thematik oder Realismus der gewünschten Ziele vermieden und das Schreiben des Entwicklungstagebuchs als roter Faden genutzt. Insbesondere stellten sich hier auch die Abschlussreviews als äußerst hilfreich heraus.

5.2 Bewertung der Methodik

Die Kombination von MIniscrum und „größeren“ Scrum-Methoden durch das wiederholte Reflektieren und Organisieren mit außenstehenden Personen erwies sich als effizient und motivierend. Insbesondere waren Fortschritte durch die konstante Formulierung von Zielen jederzeit nachvollziehbar und konnten so auch nach außen kommuniziert werden; gleichzeitig blieb durch die vollständige Planungsfreiheit jederzeit die nötige Konzentration auf das Befolgen der festgelegten Arbeitsschritte, um diese nicht aus den Augen zu verlieren. Hier erwies sich gerade auch das schriftliche Festhalten der einzelnen Aufgaben- und Tagesbereiche als hilfreich.

Aus methodischer Sicht zeigte sich, dass die Arbeit in Iterationen besonders sinnvoll war, um komplexe Anforderungen schrittweise umzusetzen. Die frühzeitige Implementierung von Kernfunktionen, gefolgt von der sukzessiven Integration von Datenbank und GUI, stellte sicher, dass jederzeit eine funktionierende Basis vorhanden war. Dies reduzierte Risiken und erhöhte die Planungssicherheit für die nachfolgenden Sprints. Aufgrund dieser Planungssicherheit und vor allem durch die Vermeidung der Sorge, schließlich alles erneut anfangen zu müssen, lag der Fokus vollständig auf der Bewältigung der nächsten Aufgabe. In diesen Bereich fällt auch die Praxis, von Anfang an eine klare Entscheidungsrichtlinie für Fragen wie die zusätzliche Implementierung weiterer Features zu formulieren und zu befolgen; hierdurch entfielen langwierige Überlegungen und Abwägungen, da für den Zweifelsfall bereits eine Vorgabe bestand, wie sie in einem größeren Team vom Product Owner gefällt worden wäre. Insgesamt bestätigte die methodische Vorgehensweise die Eignung von Miniscrum für Einzelprojekte, insbesondere wenn iterative Planung, kontinuierliche Reflektion und flexible Priorisierung zentral umgesetzt werden.

5.3 Erfolge

Die konsequente Anwendung des agilen Vorgehens führte zu mehreren messbaren und nachvollziehbaren Erfolgen. Zunächst konnte eine funktionierende Softwarelösung entwickelt werden, die sämtliche Kernanforderungen erfüllt: Patienten- und Termindaten können erstellt, angezeigt, bearbeitet und gelöscht werden, und die Benutzeroberfläche stellt alle Informationen übersichtlich dar. Ein weiterer Erfolg war die Implementierung der Datenbankintegration, die eine dauerhafte und konsistente Speicherung der Daten gewährleistet. Die Suchfunktionen für Patienten und Termine erhöhen die Nutzbarkeit des Systems und ermöglichen gezieltes Arbeiten mit größeren Datenbeständen.

Schließlich konnte das Projekt durch die methodische Vorgehensweise in einem realistischen Zeitrahmen abgeschlossen werden. Alle geplanten Sprintziele wurden erreicht, und das Endprodukt entspricht den definierten Anforderungen. Die Analyse der Vorgehensweise und die Reflexion der Entscheidungen liefern zudem wertvolle Erkenntnisse für zukünftige Projekte, insbesondere im Hinblick auf Zeitmanagement, Priorisierung und iterative Softwareentwicklung.

6. Fazit & Ausblick

Der Projektbericht dokumentiert die systematische Entwicklung einer Softwarelösung zur Verwaltung von Patienten- und Termindaten in einer Praxisumgebung. Die Umsetzung erfolgte unter Anwendung agiler Methoden, konkret in Form von Miniscrum, wodurch eine strukturierte, iterative Bearbeitung der einzelnen Entwicklungsphasen ermöglicht wurde. Durch die Kombination von theoretischem Wissen und praktischer Anwendung konnte ein funktionales Endprodukt geschaffen werden, das die zentralen Anforderungen erfüllt.

Die Analyse der durchgeführten Sprints verdeutlicht, dass die iterative Vorgehensweise insbesondere bei der Integration von Datenbankanbindungen, der Anpassung des GUI und der Einbindung zusätzlicher Funktionen wie der Suchfunktion effizient war. Herausforderungen, sowohl technischer als auch organisatorischer Natur, konnten durch kontinuierliche Reflexion, Priorisierung und gezielte Lösungsansätze bewältigt werden. Die Entscheidung, auf komplexere Features wie eine Authentifizierungsfunktion zu verzichten, unterstreicht die pragmatische Orientierung auf ein stabiles, funktionsfähiges Produkt innerhalb des gegebenen Zeitrahmens.

Der Rückblick zeigt, dass der Einsatz von Miniscrum als Methode gerade auch zur Steuerung eines Einzelprojekts eine klare Struktur bietet, die sich flexibel an unvorhergesehene Probleme anpassen lässt. Insbesondere die konsequente Dokumentation, Review- und Planungsprozesse führten zu einer effizienten Nutzung der zur Verfügung stehenden Ressourcen und erlaubten eine kontinuierliche Qualitätskontrolle des Entwicklungsprozesses.

Für zukünftige Entwicklungen lassen sich mehrere Ansatzpunkte ableiten. Eine mögliche Erweiterung umfasst die Implementierung von Benutzerkonten mit Authentifizierung, um die Sicherheit und Integrität der Daten zu erhöhen. Darüber hinaus könnten zusätzliche Filter- und Suchoptionen sowie erweiterte Auswertungsfunktionen die Benutzerfreundlichkeit und den praktischen Nutzen der Software steigern.

Abschließend zeigt das Projekt, dass die methodische Kombination von theoretischem Wissen, agiler Planung und praktischer Umsetzung zu einem Ergebnis führt, das sowohl den Anforderungen des Praxisprojekts entspricht als auch wichtige Erfahrungen für die zukünftige berufliche Praxis vermittelt.

Hinweis: Der vollständige Quellcode ist über das Repository verfügbar (vgl. Anhang C).