Hochschule Bremerhaven

Fachbereich II
Management und Informationssysteme
(Wirtschafts-)Informatik B.Sc.

Modul
Software Engineering I

PflegeDASHBOARD

Dashboard-Webanwendung für einen echten Kunden

Vorgelegt von:	Marvin Eckhoff	MatNr. 42038
	Luca Focken	MatNr. 42418
	Borai Manuth	MatNr. 42296
	Leon Stüve	MatNr. 41996
	Lukas Weber	MatNr. 41894
Vorgelegt am:	13. Februar 2026	
Dozent:in:	Prof. Dr. Bieker-Walz	

Inhaltsverzeichnis

1 Einleitung

Im Modul „Software Engineering I“ haben wir die Semesteraufgabe erhalten, ein Kundenprojekt in Kollaboration mit einer Firma durchzuführen. Die Basis der Entwicklung der geforderten Webanwendung mit Datenbankanbindung bildeten dabei die in den Veranstaltungen erlernten Methodiken und Techniken. Wir sollten dabei in selbstgewählten Teams von je drei bis fünf Personen arbeiten – unser Team, Teamname 0x173, besteht aus Marvin Eckhoff (42038), Luca Focken (42418), Borai Manuth (42296), Leon Stüve (41996) und Lukas Weber (41894).

Durch den persönlichen Kontakt eines Teammitglieds haben wir einen Auftrag von dem Klinikum Bremerhaven Reinkenheide gGmbH erhalten. Gefordert war die Konvertierung der in Microsoft-Excel dargestellten Sturzdaten in ein anschauliches Dashboard mit Datenbankanbindung.

Der von uns ausgearbeitete Prototyp ist verfügbar und über das Internet erreichbar unter unserem Team-Docker. Die Zugangsdaten für das Einfügen von Daten als autorisierter Nutzer lauten „admin“ mit dem Passwort „12345“.

Ursprünglich sind wir aufgrund der Vertraulichkeit der Daten davon ausgegangen, dass wir den Prototypen in seiner aktuellen Form nicht frei über das Internet zugänglich machen sollten. Nach weiterer Kommunikation mit dem Kunden hat sich herausgestellt, dass dies nach dem Anonymisieren des Datensatzes kein Problem darstellt.

2 Planung und Organisation

Unsere Planung haben wir an den Vorschlägen der Lehrkraft orientiert. Dementsprechend haben wir unsere Bearbeitung des Projekts an den wöchentlichen Übungsterminen angepasst, wobei wir gegen Ende in eine wesentlich selbstständigere Arbeitsphase übergegangen sind. Die von uns im Zuge der Planung und Orientierung erstellten Diagramme entsprechen den vereinbarten Unified Modeling Language (UML)-Konventionen und stellen die Basis für unser kollektives Verständnis der Anwendung dar.

2.1 Gantt-Diagramm

Im Zuge dieser Übungen haben wir zusätzlich ein grundlegendes Gantt-Diagramm angefertigt, welches für uns eine grobe Orientierung darstellen sollte. In dem Diagramm haben wir die essentiellen Meilensteine mitsamt zugehörigen Zeitpunkten eingetragen.

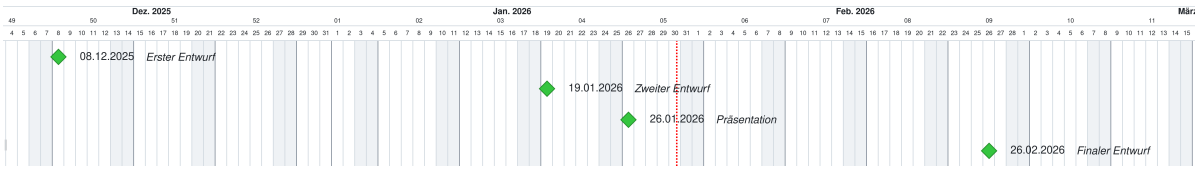

Abbildung 2.1: Gantt-Diagramm

Rückblickend haben wir das Gantt-Diagramm außerhalb der dafür vorgesehenen Übungstermine kaum verwendet. Wir können uns vorstellen, dass ein solches Diagramm gerade für große und dementsprechend weniger übersichtliche Projekte hilfreich sein kann, haben für unsere überschaubare Teamarbeit jedoch weniger Verwendungszwecke finden können.

2.2 Risikobewertung

Ebenfalls haben wir im Teamverband eine Risikobewertung durchgeführt. Wir haben verschiedene Risiken, welche wir als realistisch eingestuft haben, identifiziert und mögliche Vorkehrungen getroffen. Beispielsweise haben wir Puffer eingeplant, um die negativen Auswirkungen von Ausfällen oder Ungereimtheiten im Team zu mitigieren.

Unter den von uns identifizierten Risiken befinden sich beispielsweise Krankheitsfälle, zwischenmenschliche Konflikte, ungeahnte Wissenslücken oder auch mögliche Aussteiger aus dem Team.

Wir sind der Meinung, dass unser Zeitmanagement unseren Anforderungen in vollem Umfang entsprochen hat und dass wir weiterhin aktuell in unserem Zeitplan arbeiten können, ohne in Stress zu verfallen.

2.3 Commits und Repo-Struktur

Bevor wir mit der Umsetzung des Prototypen angefangen haben, haben wir uns im Team auf eine Commit-Struktur geeinigt, welche uns das Traversieren des Git-Logs vereinfachen sollte.

Wir haben uns dabei dazu entschieden, dem Standard der Conventional Commits [1] zu folgen. Wir haben dabei zwei Scopes verwendet: „front“ für das Frontend und „back“ für das Backend.

Dazu haben wir uns überlegt, wie das Repository strukturiert sein sollte. Wir haben dieselbe Teilung in Front- und Backend auch in dem Repository vorgenommen. Die Projektwurzel teilt sich ein in das Deploy-Skript, das Frontend, das Backend und unsere Testing-Suite.

3 Anforderungsanalyse

Wir haben uns darum für die Webanwendung entschieden, da sie unserem bisherigen Studiumsverlauf in ihrem Umfang besonders nahe kommt und die Wünsche des Kunden erfüllt. Die Umsetzung der Anwendung erfordert keine weitreichenderen Kompetenzen über sicheres Sessionhandling und die Datenbankanbindung hinaus – gleichermaßen handelt es sich um eine Anwendung, welche tatsächlich von dem Unternehmen verwendet werden möchte, was für uns eine extrinsische Motivation darstellt.

3.1 Stakeholder

Unser Team konnte drei verschiedene Stakeholder festmachen.

Die primären Nutzenden der Anwendung sind die Mitarbeitenden der Organisationseinheiten und Fachbereiche selbst. Sie sind die Personen, welche die Anwendung verwenden möchten, um die bestehenden Probleme ihres Bereichs zu erkennen. Darum haben sie das Interesse, dass die Anwendung nutzerfreundlich und besonders die Darstellung der Daten übersichtlich ist.

Die sekundären Nutzenden der Anwendung sind die „Administratoren“ – dies sind speziell solche Nutzenden, welche Daten einfügen. Zusätzlich zu den Anforderungen, welche auch die ordinären Mitarbeitenden an die Anwendung stellen, interessieren sie sich für dem Einfügeprozess neuer Daten. Für sie ist am wichtigsten, dass die Daten zuverlässig und mit Feedback an sie verarbeitet werden.

Letztlich stellen wir als Entwickelnde einen Stakeholder dar. Wir haben dabei zwei Interessen: Das Bestehen des Moduls sowie das Bereitstellen einer zufriedenstellenden Anwendung für den Kunden.

3.2 Analyse des Nutzungskontexts

Wir haben uns für das Leitfadeninterview entschieden. Die Beweggründe für diese Entscheidung waren der bereits bestehende, nahe Kontakt zum Kunden, welcher es uns einfach gemacht hat, konkrete Fragen zu stellen, um die Anforderungen erkennbar zu machen sowie der Fakt, dass die beiden anderen Interviewmethoden unseres Empfindens nach erfordert hätten, dass die Nutzenden bereits mit einem vergleichbaren System arbeiteten, was zum Zeitpunkt der Datenerhebung nicht der Fall war.

Als Vorbereitung auf das Interview haben wir einen in verschiedene Kategorien eingeteilten Fragenkatalog vorbereitet, welchen wir im Interview sequentiell abarbeiten wollten. Folgende Themen sollten im Interview thematisiert werden: allgemeine Fragen, das Design, die Performance sowie die Datenbank.

Das Interview haben wir durchgeführt, indem wir einen Termin über Jitsi mit dem Kunden vereinbart haben. Je ein Teammitglied hat die Rolle des Protokollanten sowie des Interviewers eingenommen. Bereits vorbereitete Fragen wurden planmäßig abgearbeitet und spontane oder aus Nachfragen entstandene Fragen wurden in das Protokoll aufgenommen.

Durch das Interview haben wir unser bis zu diesem Zeitpunkt grobes Verständnis der Anwendung in Bezug auf die Anforderungen erweitern können. Es haben sich weitere funktionale sowie nicht-funktionale Anforderungen herauskristallisiert und Unklarheiten konnten mit dem Kunden geklärt werden.

Beispielsweise war uns vor dem Interview noch nicht klar, ob für das Hinzufügen der Daten eine Authentifizierung notwendig sein sollte sowie in welchem Umfang wir uns um die damit verbundenen Authentifizierungsdaten kümmern sollten. Durch das Interview haben wir dann erfahren, dass ebendiese Anforderung in vollem Umfang an uns gestellt war.

Detailliertere Informationen zu den Anforderungen behandeln wir in dem dafür vorgesehenen Abschnitt.

3.3 Personas

Um das Verwenden der Anwendung realistisch zu modellieren, haben wir einige Personas angefertigt.

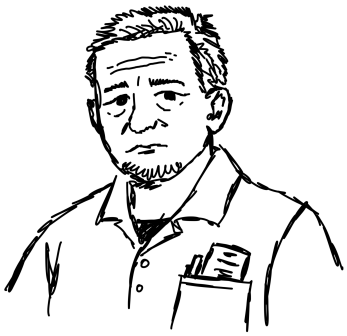

Abbildung 3.1: Beispielbild für Harald

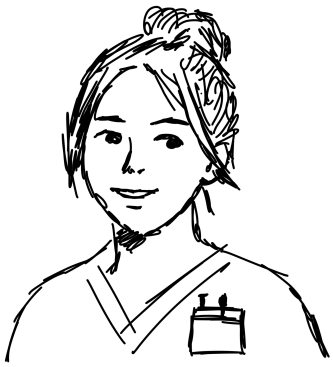

Abbildung 3.2: Beispielbild für Lena

Abbildung 3.3: Beispielbild für Hans

Die erste Persona stellt Harald Horst dar. Er ist 62 Jahre alt, männlich, kommt aus Stade, ist verheiratet und hat zwei Kinder. Er begann seine Karriere als Mechaniker, entschied sich im weiteren Lebensverlauf jedoch dazu, Stationsleiter einer Abteilung in der Klinik zu werden. Durch sein fortgeschrittenes Alter präferiert er alte Technologie und besitzt gute zwischenmenschliche Kompetenzen. Er befindet sich kurz vor der Rente und steht Änderungen in seinem

Arbeitsfluss und neuer Technologie kritisch gegenüber – so erwartet er eine persönliche, schnelle Einarbeitung in das System.

Die nächste Persona ist Lena Schmidth. Sie ist 19 Jahre alt, weiblich, kommt aus Hannover, ist ledig und hat keine Kinder. Sie macht eine Ausbildung zur Krankenpflegerin und ist im dritten Lehrjahr. Sie besitzt etwas fortgeschritteneres Fachwissen im Umgang mit Technologie, möchte Menschen helfen und engagiert sich sozial. Lena erwartet, dass die Software stetig funktioniert und sich reibungslos in den Arbeitsprozess einbinden lässt. Sie ist etwas vorsichtiger und dementsprechend langsamer im Umgang mit Software.

Unsere letzte Persona ist Hans Jürgen. Er ist 32 Jahre alt, männlich, wohnt in Bremerhaven, ist ledig, aber hat fünf Kinder. Er hat eine Ausbildung zum Krankenpfleger abgeschlossen und ist nun berufstätig als Abteilungsleiter der Organisationseinheit 11C. Er besitzt Fachwissen im medizinischen Umfeld, weist Erfahrung im Umgang mit Webbrowsern auf und hat Interesse an der Medizin. Hans möchte Geld verdienen, Menschen helfen und seine Abteilung optimieren. Er erwartet, dass das Einarbeiten in die Anwendung unter kurzem Zeitaufwand gelingt. Technisch relevante Verhaltensweisen von ihm sind, dass er seinen Computer selten neustartet und oft ungeduldig ist.

Die Personas haben uns weitere Blickwinkel auf das Projekt ermöglicht und besonders das umgesetzte Design der Weboberfläche beeinflusst – so muss diese auch von weniger technisch raffinierten Personen wie Harald navigierbar sein.

3.4 Anforderungen

Wir haben die Anforderungen gemäß der in den Veranstaltungen erlernten Prozessen in verschiedene Kategorien eingeteilt, die durch das Akronym Functionality, Usability, Reliability, Performance, Supportability (FURPS+) beschrieben werden.

Die erste Kategorie sind die funktionalen Anforderungen. Diese haben wir in einem Use-Case-Diagramm dargestellt.

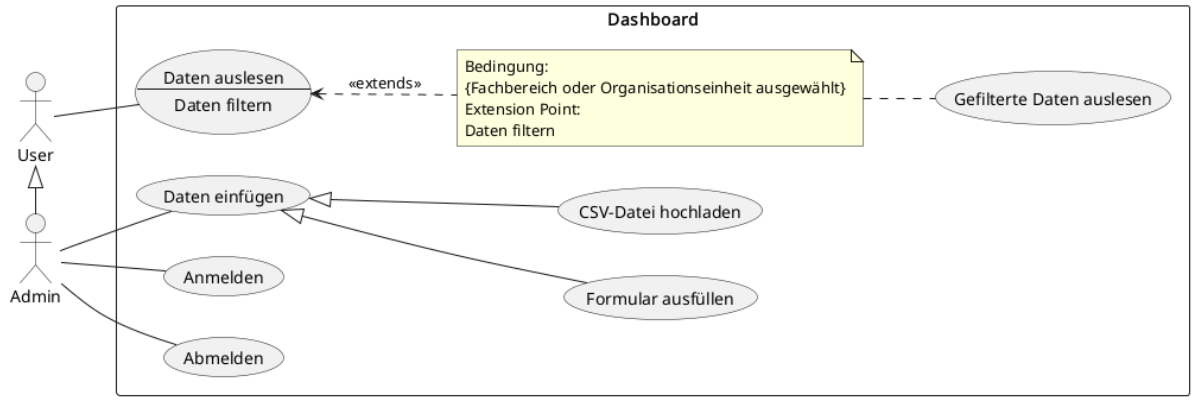

Abbildung 3.4: Use-Case-Diagramm

Wir sehen in dem Diagramm, dass es zwei verschiedene Userarten gibt: Ordinäre Nutzende und Administratoren, welche darüber hinaus Rechte besitzen. Der Administrator-Akteur ist, wie durch den Pfeil im Diagramm dargestellt, von dem ordinären Benutzenden abgeleitet. Zu ihnen beiden gehören Use Cases, welche Teil des Dashboard-Systems sind.

Ordinäre Benutzende möchten Daten auslesen und gegebenenfalls Daten filtern, was das Auswählen eines Fachbereichs oder einer Organisationseinheit erfordert. Dies ist im Diagramm dargestellt durch einen Extend-Pfeil. Administratoren haben dazu die Möglichkeit, sich an- und abzumelden sowie Daten einzufügen. Das Einfügen der Daten geschieht dabei entweder in Form eines Hypertext Markup Language (HTML)-Formulars oder als Hochladen einer Comma-Separated Values (CSV)-Datei. Dies ist im Diagramm ebenfalls durch Ableitungen dargestellt.

Die darauffolgende Kategorie ist die Benutzerfreundlichkeit. Zu dieser Kategorie gehören drei unserer Anforderungen. Einerseits ist gefordert, dass das Design der Weboberfläche, mit welcher die Benutzenden aller Arten interagieren, übersichtlich ist. Dieser Aspekt geht damit auf die bereits unter dem Abschnitt der Personas behandelte Reibungslosigkeit ein.

Die zweite Anforderung ist, dass das Design visuell ansprechend sein soll. Dies unterscheidet sich zu der ersten Anforderung darin, dass hierbei lediglich eine kosmetische Verbesserung angestrebt wird, die beispielsweise das Zurechtfinden auf der Seite nicht weiter verbessert. Der Kunde hat konkretisiert, dass wir uns an den bereits bestehenden Designelementen der Webseite des Klinikums orientieren sollen.

Während die ersten beiden Anforderungen direkt aus dem Gespräch mit dem Kunden stammen, hat sich die letzte Anforderung dieser Kategorie, dass Rückmeldungen an die Nutzenden über das User Interface (UI) ausgegeben werden, aus Arbeitsprozessen unseres Teams ergeben. Dies erachten wir als wichtig, da ohne Rückmeldungen uneinsichtig ist, ob und wieso Aktionen fehlschlagen – also ob es auf der System- oder Nutzendenseite zu Fehlverhalten kommt.

Die nächste Kategorie ist die Zuverlässigkeit. Hierbei existiert der Anspruch auf wohldefiniertes Verhalten seitens der Anwendung. Konkret heißt das, dass die Anwendung Fehlerfälle abfängt und verhindert, dass fehlerhafte Daten eingefügt werden. Dazu sollte die Anwendung soweit möglich jederzeit erreichbar sein – es sollte verhindert werden, dass Komponenten aufhören können, zu funktionieren.

Leistung ist die nächste Kategorie. In diesem Bereich sind keine strengen Anforderungen aus dem Kontakt mit dem Kunden hervorgegangen. Geringe Laufzeiten und Latenzen sind erwünscht, jedoch kein Kernelement der Anwendung.

Die vorletzte Kategorie stellt die Servicefreundlichkeit dar. Unsere Anwendung sollte wartbar sein, dies wird jedoch von der Information Technology (IT)-Abteilung des Klinikums übernommen werden. Für uns kommen damit keine weiteren Aufwände zustande, trotzdem haben wir natürlich versucht, den Code wartbar zu gestalten.

Zuletzt geht es uns um die technischen und rechtlichen Anforderungen. Es ist im Bezug auf die technische Hälfte dieser Kategorie darauf zu verweisen, dass die Webseite lediglich über das Intranet erreichbar sein muss. Rechtlich ist zu beachten, dass die Daten, welche wir im Zuge der Anwendung handhaben, von sensibler Natur sind, sodass wir den Umgang mit ihnen bewusst gestalten müssen und dass Lizenzen von Technologien, die wir verwenden, wie Schriftarten und Bibliotheken, geachtet werden.

Neben den Anforderungen an das System haben wir zusätzlich mögliche Misuse-Cases ermittelt und versucht, sie zu mitigieren.

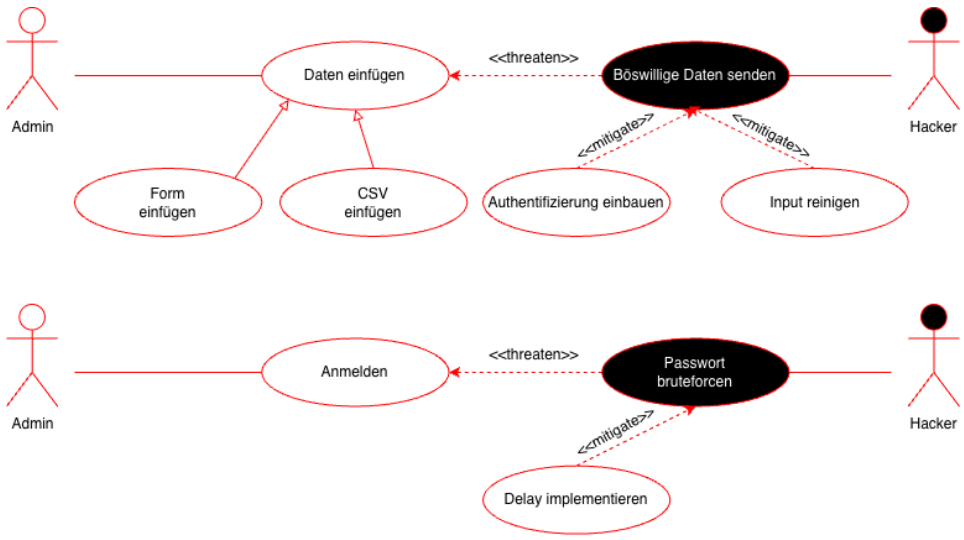

Abbildung 3.5: Misuse-Case-Diagramm

Wir haben zwei verschiedene Misuse-Cases erarbeitet. Dargestellt werden diese im Diagramm durch den Akteur Admin und Misakteur Hacker, wobei die Kanten rot eingefärbt sind, was der Notationsfolie aus den Veranstaltungen entspricht [2, Seite 10].

Der erste Misuse-Case stellt das Einsenden böswilliger Daten dar. So kann es beispielsweise vorkommen, dass nicht lediglich inkorrekt formatierte Daten übersendet werden, sondern darüber hinaus versucht wird, arbiträren Code auszuführen. Um dem entgegenzuwirken, lässt sich eine Authentifizierung einzubauen sowie den Input unter ebendiesem Gesichtspunkt böswilliger Daten zu bereinigen.

Der zweite Misuse-Case ist das Bruteforcen der Logindaten, was den ungewollten Zugriff auf das Einfügen von Daten überhaupt ermöglicht. Hier haben wir uns an Systemen wie Linux orientiert, welche einen Delay einbauen, welcher die möglichen Einlogversuche pro Sekunde drastisch reduziert – dies stellt ebenfalls unsere Mitigation dar.

4 Modellierung der Webanwendung

4.1 Storyboard

Um die Nutzungsweisen der Anwendung durch die Nutzenden für uns greifbarer zu machen, haben wir ein Storyboard erstellt, welches eine denkbare Interaktion mit der Webseite behandelt.

Die Handlung haben wir in vier Schritte unterteilt und mit passenden Darstellungen versehen.

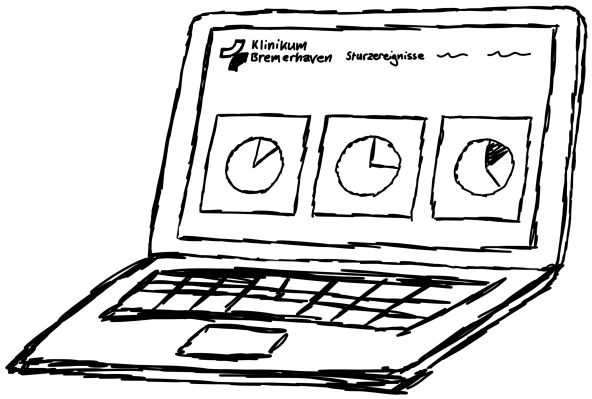

Abbildung 4.1: Storyboard-Panel 1

Im ersten Schritt öffnet Harald, eine unserer etablierten Personas, das Dashboard und landet direkt auf der Startseite. Um mögliche Probleme im Klinikum zu identifizieren und zu beheben, möchte er Sturzdaten ausgewählter Organisationseinheiten einsehen und miteinander vergleichen.

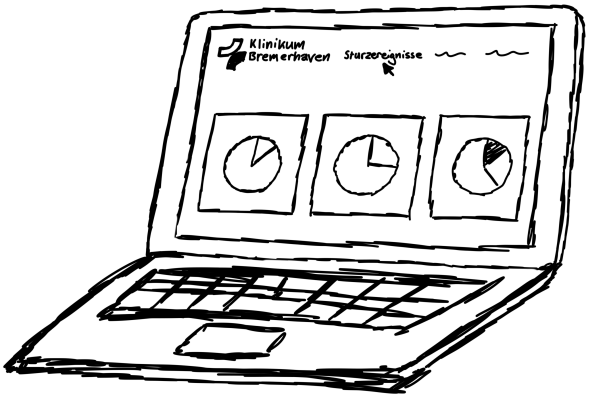

Abbildung 4.2: Storyboard-Panel 2

Er klickt folgend auf den Reiter „Sturzereignisse“.

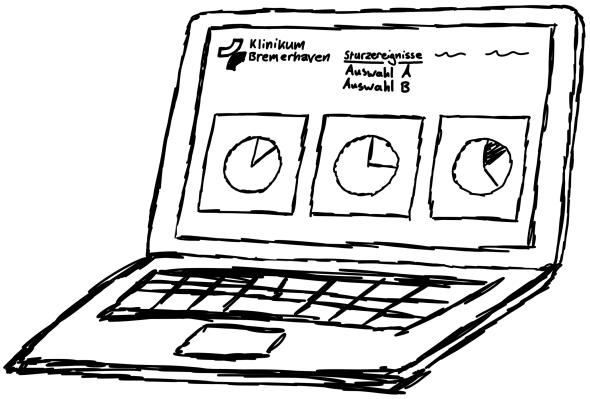

Abbildung 4.3: Storyboard-Panel 3

Es öffnet sich ein Dropdown-Menü, welches die möglichen Organisationseinheiten auswählbar macht. Harald klickt auf seine gewünschte Einheit.

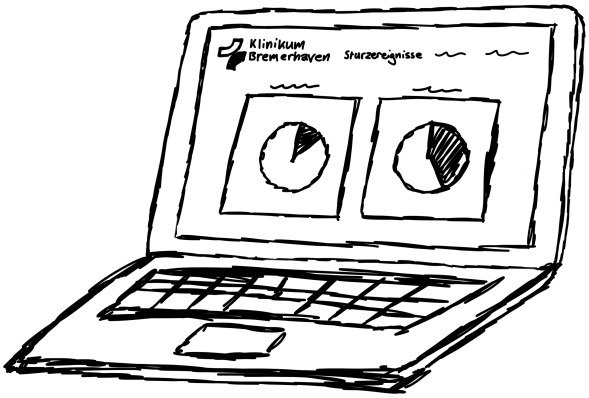

Abbildung 4.4: Storyboard-Panel 4

Somit landet er auf der Sturzereignisseite, welche die gefilterten Daten visualisiert. Nun kann er die Werte nach Belieben vergleichen und auswerten.

4.2 Wireframe

Um das Layout der Webseite konkret auszuarbeiten, haben wir Wireframes für die Startseite und die dedizierte Sturzereignisseite mit dem Kunden in stetigem Dialog entwickelt.

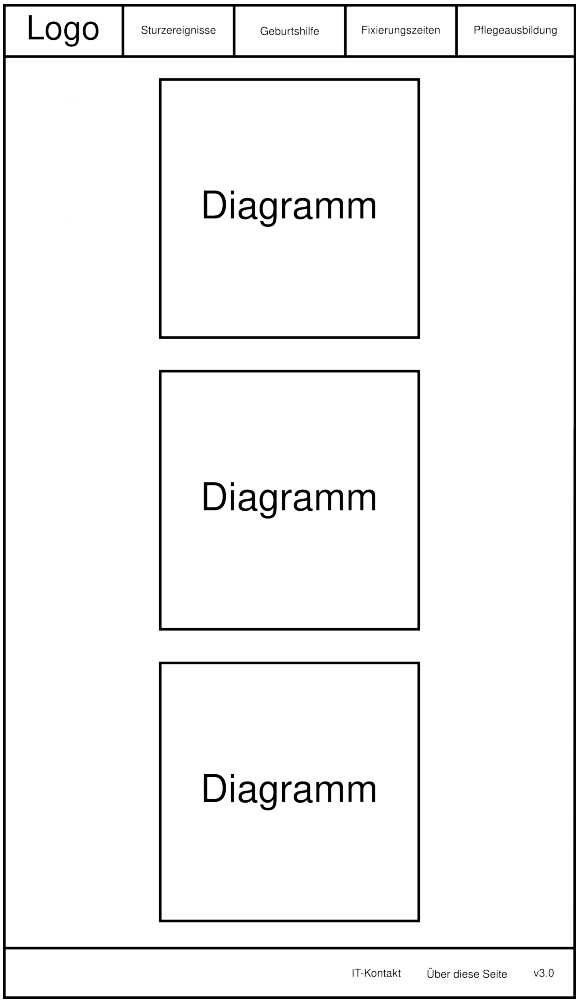

Abbildung 4.5: Wireframe der Startseite

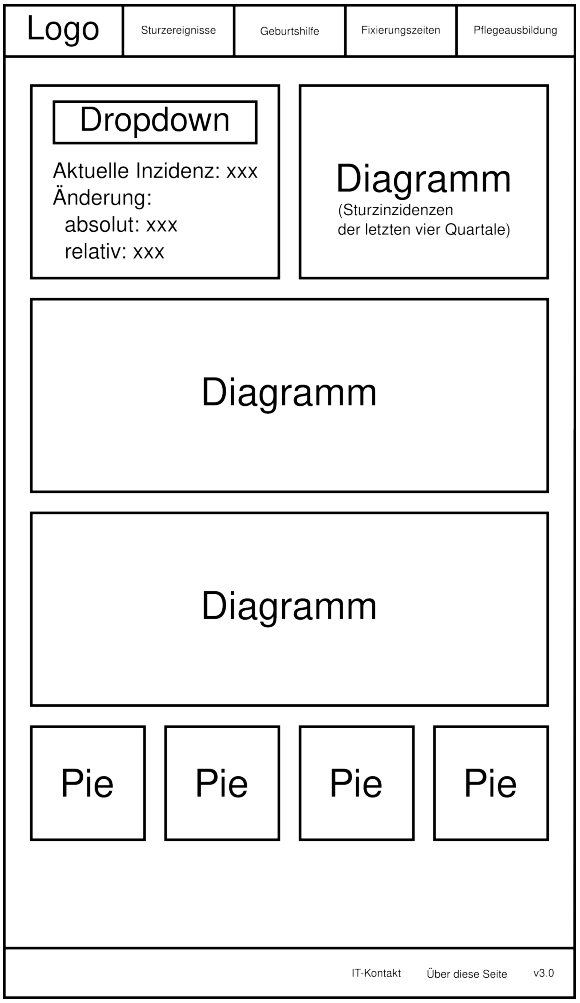

Abbildung 4.6: Wireframe der Sturzereignisseite

Für die Navigation zwischen den einzelnen Seiten ist der Header zuständig. Er ist über die Seiten hinweg identisch. Die Seiten abseits der Start- und Sturzdatenseite werden in Zukunft und unabhängig von unserem Team und unserer Arbeit entwickelt werden.

Ebenso gleicht der Footer der Startseite dem Footer jeder anderen Seite. Er enthält einen Link auf die Seite des IT-Supports für Fragen, einen Link auf die About-Seite der Anwendung und auf Kundenwunsch die aktuelle Version der Software.

Die Startseite ist so ausgelegt, dass einige der wichtigsten Diagramme untereinander angezeigt werden. So können diese sofort eingesehen werden.

Die Sturzereignisseite enthält eine Infobox mit dem Dropdown-Menü für die Auswahl der Organisationseinheit. Zusätzlich dazu enthält sie die aktuelle Sturzinzidenz mitsamt der absoluten

und relativen Änderung. Die Sturzinzidenz beschreibt die Anzahl der Stürze pro 1000 Belegungstage, wobei ein Belegungstag wiederum einem Tag entspricht, an dem ein*e Patient*in ein Bett belegt.

Die restlichen Hauptelemente der Seite sind die darzustellenden Diagramme.

4.3 Aktivitätsdiagramme

Die genauen Abläufe der Anwendung stellen wir mithilfe von Aktivitätsdiagrammen dar. Dies hilft uns, Programmiersprachen-agnostisch nachzuvollziehen, wie genau die einzelnen Funktionen zu den zugehörigen Use Cases implementiert werden sollten.

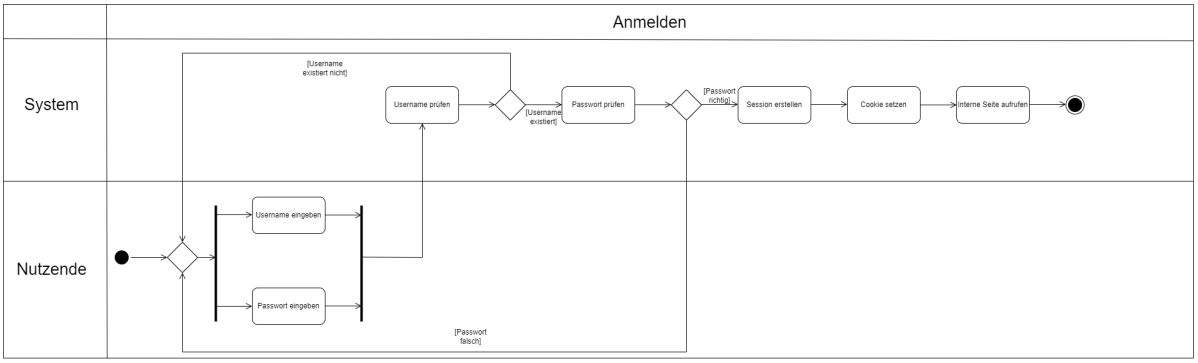

Abbildung 4.7: Anmelden

Das erste Aktivitätsdiagramm stellt das Anmelden dar.

Wir beginnen damit, dass der Username und das Passwort eingegeben werden. Da hier keine Reihenfolge notwendig ist, kennzeichnen wir die Parallelität durch die vertikalen Balken. Nach der erneuten Zusammenführung und nachfolgenden Sequentialität prüft das System den Usernamen. Existiert dieser nicht in der Datenbank, so wird zu der erneuten Eingabe der Daten aufgerufen. Existiert dieser in der Datenbank, wird das Passwort geprüft. Auch hier wird bei einer Unstimmigkeit zu dem erneuten Eingeben der Daten aufgefordert. Passt das Passwort dahingegen zum Usernamen, wird die Session erstellt, der Cookie gesetzt und die interne Seite aufgerufen.

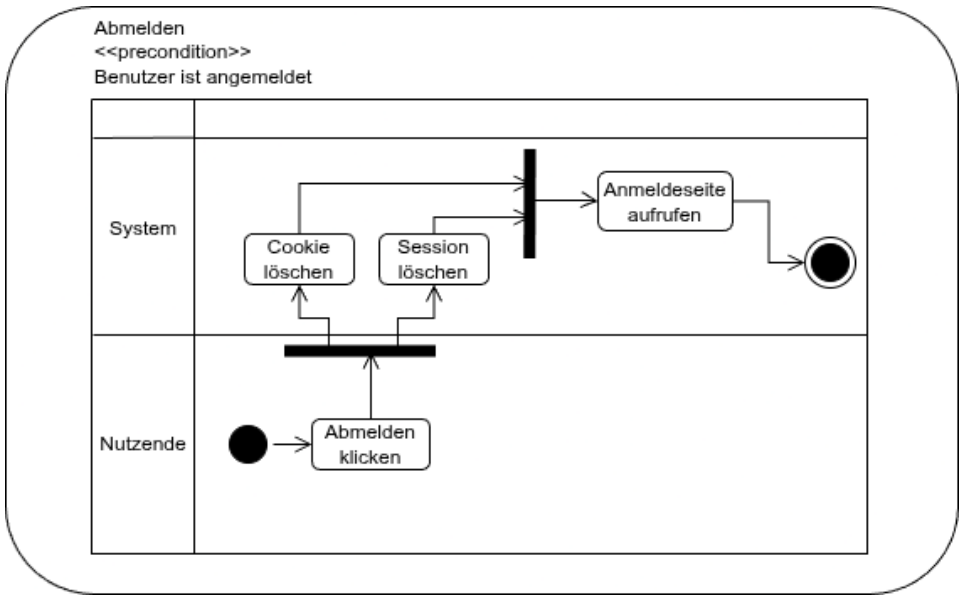

Abbildung 4.8: Abmelden

Als nächstes betrachten wir das Aktivitätsdiagramm, welches das Abmelden behandelt. Die Precondition ist, dass der User angemeldet ist.

Die Person klickt auf den Abmelde-Button. Daraufhin werden zwei Dinge von dem System getan: Der Cookie wird gelöscht und die Session wird gelöscht. Da die Reihenfolge hierbei nicht relevant ist, geschieht dies parallel. Daraufhin wird die Anmeldeseite aufgerufen.

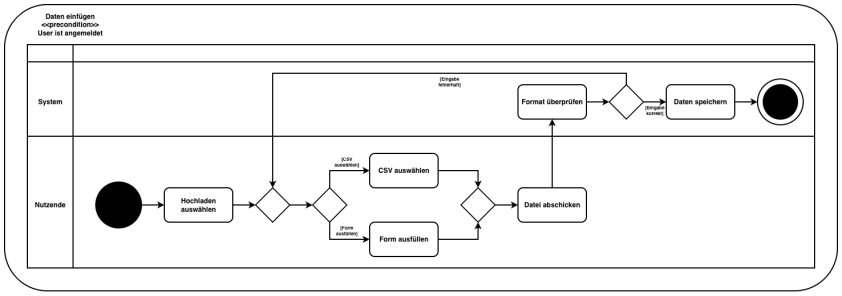

Abbildung 4.9: Daten einfügen

Das dritte Aktivitätsdiagramm behandelt das Einfügen von Daten. Die Precondition ist, dass der User angemeldet ist.

Die Person öffnet den Reiter und wählt danach eine Einfügemethode aus. Wenn das Formular ausgewählt wurde, füllt die Person dieses aus. Wurde stattdessen die CSV-Datei ausgewählt, so lädt die Person eine Datei hoch. An diesem Punkt werden die divergenten Handlungsstränge

wieder vereinigt. Der User lädt die Daten hoch. Entsprechen die Daten dem Format, so endet die Aktivität. Entsprechen die Daten nicht dem Format, so wird wieder zur Eingabe aufgefordert.

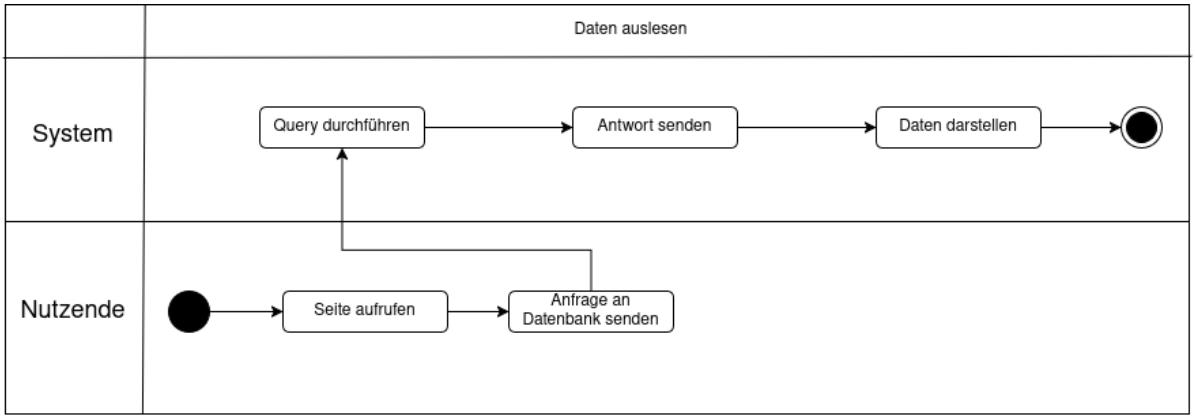

Abbildung 4.10: Daten auslesen

Das letzte Aktivitätsdiagramm behandelt das Auslesen der Daten. Dafür ruft der User die Seite auf. Dann sendet er eine Anfrage an die Datenbank. Die Query wird von dem System durchgeführt und die Antwort wird gesendet. Dann werden die Daten dargestellt.

4.4 Klassendiagramm

Um die für die Anwendung relevanten Objekte übersichtlich zu gestalten, haben wir sie mitsamt ihren Relationen untereinander in einem Klassendiagramm visualisiert. Hier lassen sich unter anderem Baumuster auffinden, welche wir in den Veranstaltungen kennengelernt haben [3, S. 24f.].

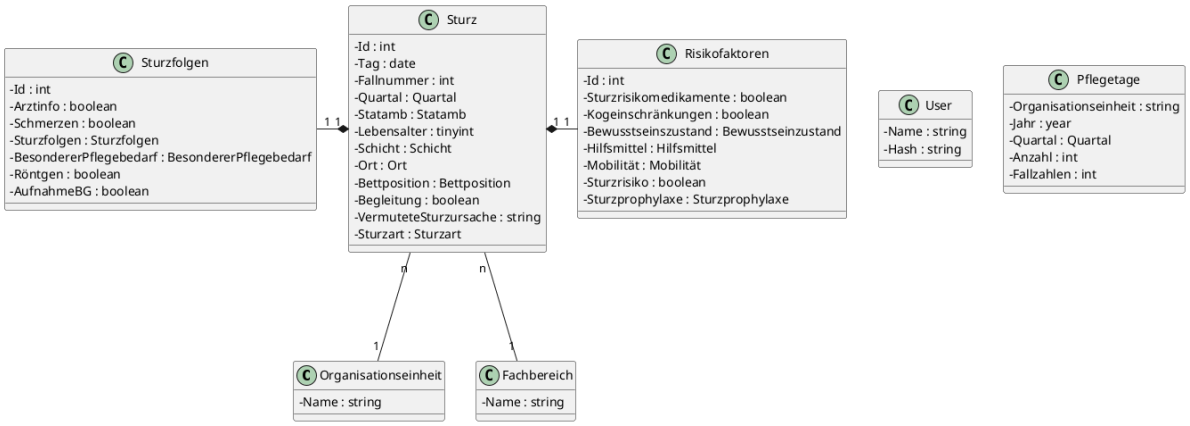

Abbildung 4.11: Klassendiagramm

Die User sind alleinstehend. Sie weisen einen Namen und einen Hash auf. Beide dieser Daten werden für das Anmelden verwendet.

Die Pflegetage speichern, wie viele Belegungstage eine Organisationseinheit pro Quartal aufweist. Diese werden von uns bereits für das Berechnen der Sturzinzidenzen verwendet, sollen aber in der Zukunft, nach unserer Bearbeitung, weitere Relevanz erlangen.

In der Erarbeitung unserer Klassenstrukturen, haben wir ausgehend von einer Gottklasse gearbeitet. Auf eine Rückmeldung der Lehrkraft hin haben wir diese weiter aufgeteilt und dabei Baumuster verwendet.

Es liegt nun ein Koordinator vor – die Klasse „Sturz“ stellt den Koordinator dar, wobei die Organisationseinheit und der Fachbereich Teilnehmer sind. Bei den Teilnehmern handelt es sich um dedizierte Klassen statt einfach Enums, da diese in Zukunft erweitert werden sollen.

Darüber hinaus ist der Sturz das Ganze einer Baugruppe, dessen Teile die Sturzfolgen und die Risikofaktoren sind. Hier ist ersichtlich, dass die Aufteilung aus Übersichtlichkeitsgründen geschehen ist.

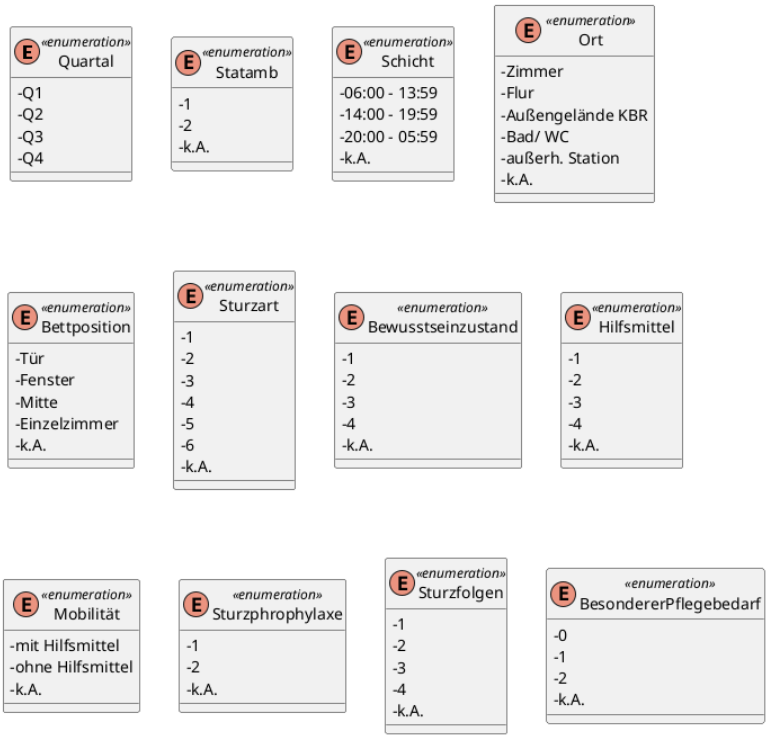

Abbildung 4.12: Klassendiagramm der Enums

Wo wir es als sinnvoll erachtet haben, haben wir Enums definiert und verwendet.

4.5 Sequenzdiagramm

Um die Interaktionen der einzelnen Komponenten des Gesamtsystems zu visualisieren, haben wir ein Sequenzdiagramm entwickelt. Dieses bildet ab, wie es aussehen kann, wenn ein Administrator versucht, Daten über die Weboberfläche in die Datenbank einzufügen.

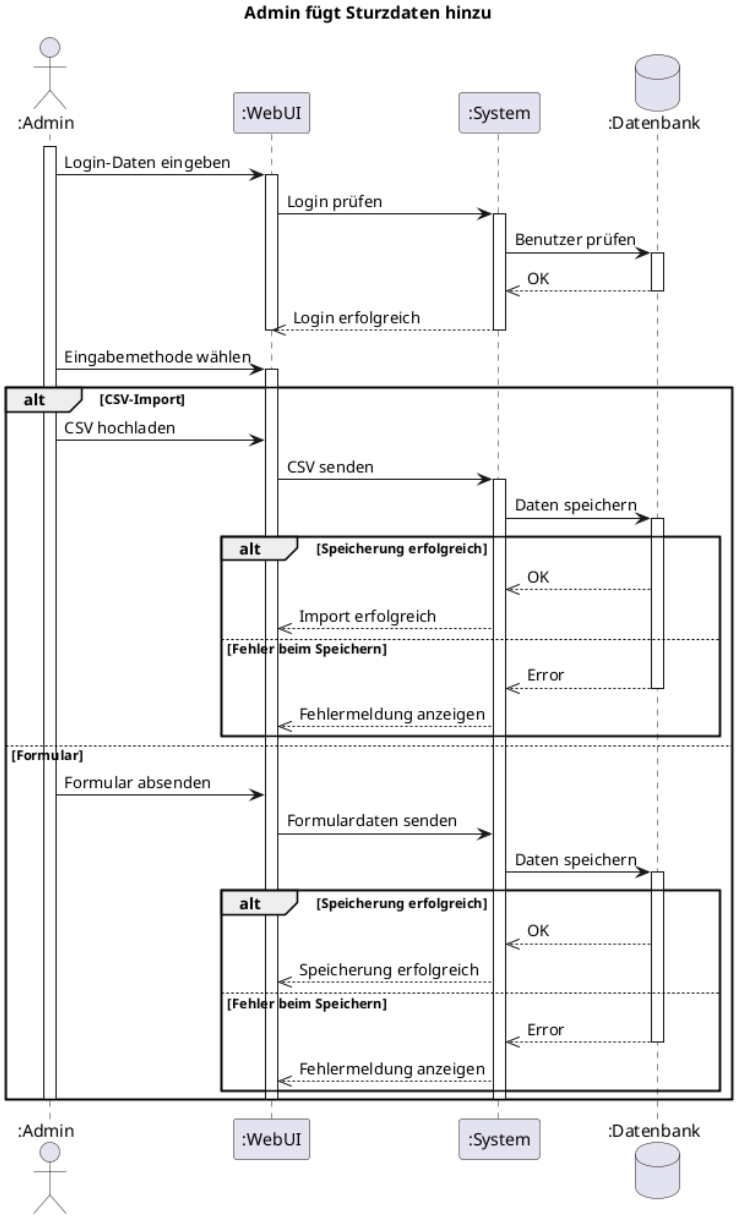

Abbildung 4.13: Sequenzdiagramm

Es beginnt damit, dass der Admin seine Anmeldedaten eingibt. Die Weboberfläche fragt das

System an, ob das Einloggen erfolgreich ist. Um dies zu überprüfen, sendet das System eine Anfrage an die Datenbank. Diese antwortet, wie es durch einen gestrichelten Pfeil mit offenem, nicht ausgefülltem Pfeilkopf gemäß der Folie [4, Seite 41] dargestellt ist, mit einem OK. Daraufhin antwortet das System an die Weboberfläche, dass das Anmelden funktioniert hat.

Bei diesem Vorgang kann es dazu kommen, dass die Eingabe falsch ist und somit das Einloggen fehlschlägt. Dies stellt zwar einen alternativen Ablauf dar, aber da es uns eigentlich um das Einfügen der Daten geht, welches gemäß unserer Aktivitätsdiagramme erfordert, dass der User angemeldet ist, gehen wir davon aus, dass das Einloggen erfolgreich geschieht. Die Notation des alternativen Ablaufs soll hier nicht im Vordergrund stehen – dies hat eine verbesserte Lesbarkeit des Diagramms zufolge. Wir orientieren uns an diesem Punkt an der Folie [4, Seite 37].

Nun wählt der Admin eine Eingabemethode aus. Hier kann es zu alternativen Abläufen kommen: Entweder der CSV-Import oder das Formular wird ausgewählt.

Wurde die erste Option gewählt, so wird die CSV von dem Admin an die Weboberfläche hochgeladen. Die Weboberfläche sendet die CSV-Datei dann an das System. Dieses speichert die Daten in der Datenbank. Hier kann es erneut zu alternativen Abläufen kommen, denn das Einfügen kann aufgrund fehlgebildeter Daten fehlschlagen.

Ist die Speicherung erfolgreich, so kommt dem System eine OK-Antwort der Datenbank zu. Daraufhin antwortet das System an die Weboberfläche, dass der Import erfolgreich war.

Ist die Speicherung dagegen nicht erfolgreich, so sendet die Datenbank dem System einen Fehler. Dieses zeigt dann in der Weboberfläche eine passende Fehlermeldung an.

Wurde das Formular als Eingabemethode gewählt, so wird statt der CSV-Datei das Formular an die Weboberfläche und von dieser an das System gesendet und folgend von dem System verarbeitet. Die danach folgenden Abläufe gleichen dabei den bereits behandelten Abläufen.

Es sind in dem Diagramm die Lebzeiten der Objekte ablesbar. Die erste Gruppe von Lebzeiten umfasst den Anmeldeprozess. Die zweite Gruppe umfasst das Hochladen und Verarbeiten der Dateien. Aus diesem Schema fällt der Admin selbst, welcher durchgehend aktiv ist.

An dieser Stelle möchten wir darauf verweisen, dass die dargestellten Pfeiltypen den Anforderungen der Vorlesungsinhalte entsprechen und lediglich in ihrem Aussehen durch die Anzeigedarstellung von PlantUML abweichen.

5 Umsetzung und Test der Webanwendung

5.1 Beschreibung der Anwendung

Unsere Webanwendung ist ein Dashboard, welches Sturzdaten in einem wohldefinierten Format anzeigt. Das Speichern dieser Daten geschieht in einer MariaDB- oder MySQL-Datenbank.

Zusätzlich dazu gibt es eine Anmeldefunktion und die Möglichkeit für angemeldete User, Daten in die Datenbank einzufügen.

Im Frontend verwenden wir HTML, Cascading Style Sheets (CSS) und JavaScript. Dazu verwenden wir PHP: Hypertext Preprocessor (ehemals Personal Home Page) (PHP) im Backend für die Kommunikation mit der Datenbank, das Verifizieren der gehashten Passwörter im Anmeldevorgang sowie für die Authentifizierung in Form der PHP-Sessions.

Für das Anzeigen der Daten in dem Dashboard verwenden wir die Massachusetts Institute of Technology (MIT)-lizenzierte JavaScript-Bibliothek Plotly [5].

5.2 Dokumentation eines Anwendungsfalls

Der Anwendungsfall, den wir mit diesem Abschnitt behandeln möchten, lautet wie folgt: Ein Admin, unsere Persona Hans Jürgen, ruft die Seite auf, fügt aktuelle Daten des Klinikums ein und überprüft im Anschluss daran die Sturzwerte einer Organisationseinheit, 11C, in Relation zu den Werten des gesamten Klinikums. Diese Verwendung der Anwendung umfasst damit konkret alle unsere Use Cases: Von dem Einsehen und Filtern der Daten bis zu dem Einfügen, was das An- und konsekutive Abmelden erfordert. Im Folgenden arbeiten wir diesen schrittweise durch.

Als Erstes ruft der Admin die Seite auf und klickt auf den Reiter „Sturzereignisse“.

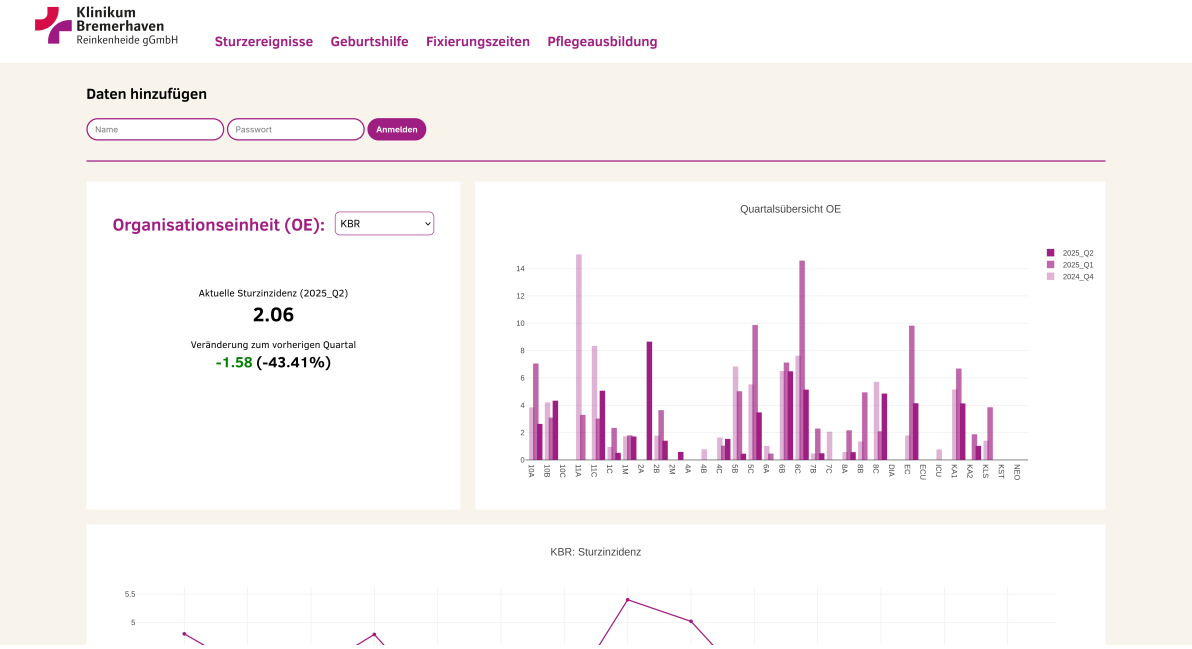

Abbildung 5.1: Screenshot der Sturzereignisseite

Nun möchte er Daten einfügen. Damit er dies tun kann, muss er angemeldet sein. Somit gibt er seine Logindaten ein.

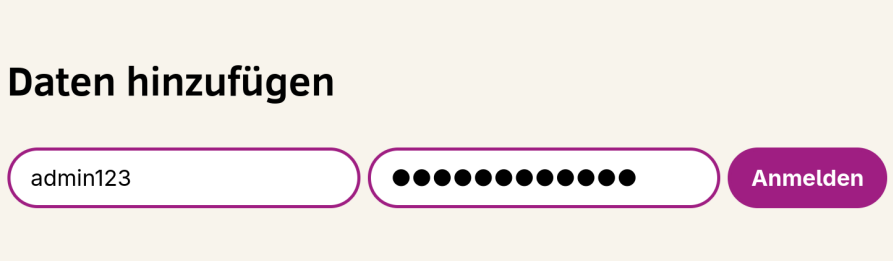

Abbildung 5.2: Screenshot der Anmeldung

Er klickt auf „Anmelden“.

In der Datenbank liegt eine Tabelle vor, welche wir bereits in dem Klassendiagramm aufgegriffen haben: User. Es wird für jeden User ein Name à 255 maximale Zeichen und ein Passwort à 60 Zeichen im Binärformat gespeichert. Diese Daten sind relevant für den Anmeldeprozess.

Einträge werden auf Kundenwunsch nicht in der Weboberfläche angelegt. Es existiert ein Skript, welches von der Kommandozeile aus aufgerufen werden soll.

```
12 $sql = 'INSERT INTO users (name, pass) VALUES (?, ?)';
13 try {
14   $conn->execute_query($sql, [$name, password_hash($pass, PASSWORD_BCRYPT)]);
15   echo("added user\n");
16 } catch (Exception $e) {
17   echo('failed to add user: ' . $conn->error . "\n");
18 }
```

Listing 5.1: Einfügen eines Users

Das Einfügen geschieht mithilfe der `password_hash`-Funktion. Als Hashing-Algorithmus verwenden wir den etablierten Bcrypt-Algorithmus, dessen Ausgaben passgenau in das 60 Bytes große Feld in der Datenbank passen. Es wird bereits durch das Verwenden der Funktion ein Salt eingefügt, sodass wir uns darum nicht weiter kümmern müssen.

```
23 $userquery = $conn->execute_query('select pass from users where name=?',
   ↪  [$creds['user']]);
24 if($row = $userquery->fetch_assoc()) {
25   if (!$row['pass']) panic();
26   $valid_creds = password_verify($creds['pass'], $row['pass']);
27 } else {
```

```
28    $valid_creds = false;
29 }
```

Listing 5.2: Überprüfen der Anmeldung

Bei der Überprüfung verwenden wir darum die `password_verify`-Funktion. Da der für das Einspeichern verwendete Algorithmus in dem Hash enkodiert ist, müssen wir hier keine Angabe diesbezüglich treffen.

Nach erfolgreicher Authentisierung wird er auf die interne Seite weitergeleitet.

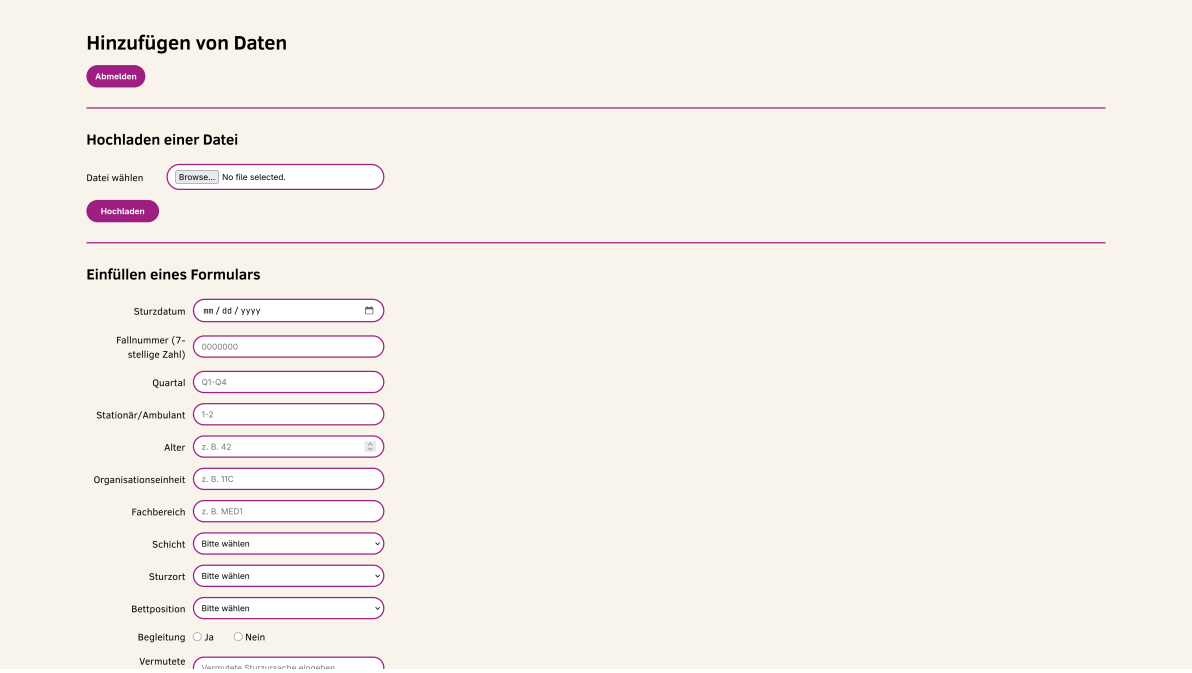

Abbildung 5.3: Screenshot der internen Seite

Er wählt seine CSV-Datei aus und lädt sie hoch.

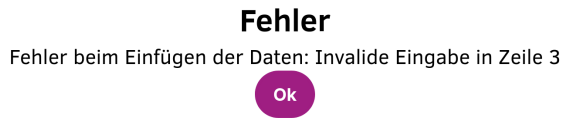

Abbildung 5.4: Screenshot bei Abgabe einer fehlerhaften Datei

Über die Weboberfläche bekommt er mitgeteilt, dass seine Datei einen Formatfehler in der dritten Zeile enthält.

```php
if (!is_null($_FILES) && !is_null($_FILES["csv"])) {
  $file = $_FILES["csv"]["tmp_name"];
  $fp = fopen($file, "r");
  if ($fp === false)
    error("Fehler beim Öffnen der übersandten Datei");
  handle_file($fp, $conn);
  fclose($fp);
} else if (!is_null($_POST)) {
  foreach ($_POST as $key => $value) {
    $line_fields[] = $value;
  }
  if (!handle_row($line_fields, $conn))
    error("Invalide Eingabe im Formular");
} else {
  error("Eingabedaten fehlen");
}
```

Listing 5.3: Einfügen von Daten

Im Backend bearbeiten wir die einzufügenden Daten in Form der CSV-Datei oder des ausgefüllten HTML-Formulars in einem Skript. Wir unterscheiden einfach an einer Stelle, in welcher der beiden Formen die Daten bei uns ankommen.

Für das Handhaben der übersandten Datei verwenden wir die Filesystem-Funktionen von PHP. Wir erlauben das Arbeiten mit der Datei durch `fopen` und schließen sie danach wieder mit `fclose`. Den Filepointer verwenden wir in der von uns definierten Funktion `handle_file` für das sequentielle Abarbeiten der einzelnen Zeilen.

```php
function handle_file($fp, $conn) {
  // Handle file's rows sequentially
  $row = 1;
  while (($line_fields = fgetcsv($fp)) != false) {
    if (!handle_row($line_fields, $conn)) error("Invalide Eingabe in Zeile $row");
    ++$row;
  }
}
```

Listing 5.4: Die Funktion handle_file

Das Aufteilen der einzelnen Zeilen der CSV-Datei delegieren wir an die Funktion `fgetcsv`. Für jede Zeile der CSV-Datei verwenden wir unsere Funktion `handle_row`. Ebendiese Funktion wird von uns auch für das Bearbeiten des Formulars verwendet.

`handle_row` tut nichts Weiteres, als wo nötig Typkonvertierungen durchzuführen, dann die Daten einzufügen und im Falle von Misserfolgen sofort den booleschen Wert false zu returnen. Kommt kein Fehler auf, wird der Wert true returned.

Wir fangen den Rückgabewert von `handle_row` bei jedem Aufruf ab. Kommt es einmal dazu, dass er false ist, so brechen wir den Einfügevorgang ab und zeigen eine passende Fehlermeldung an – wie die, die Hans erhalten hat.

Er berichtigt den Fehler in Zeile 3 und versucht es erneut.

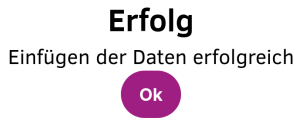

Abbildung 5.5: Screenshot bei erfolgreicher Abgabe einer Datei

Er hat die Daten erfolgreich hochladen können. Er klickt auf den „Abmelden“-Button und wird damit wieder auf die Sturzereignisseite geleitet. Nun kann er die Daten auslesen.

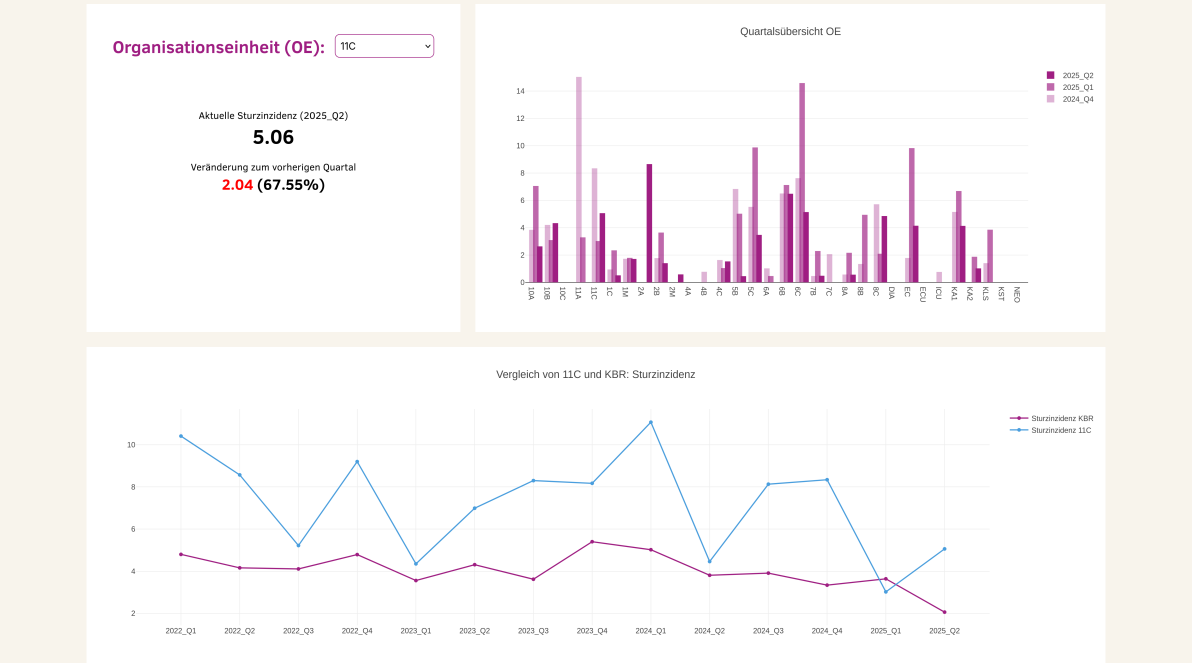

Abbildung 5.6: Screenshot der oberen Seitenhälfte

Er wählt die Organisationseinheit 11C in dem Dropdown-Menü aus. In der Infobox sieht er die aktuelle Sturzinzidenz der ausgewählten Organisationseinheit sowie die absolute und relative Änderung dieser zum vorherigen Quartal.

Dem Diagramm zur Rechten der Infobox entnimmt er, dass 11C eher dem Mittelfeld in puncto Sturzinzidenz angehört. Ebenso erkennt er, dass eine wesentliche Verbesserung zu dem Wert von vor zwei Quartalen vorliegt.

Das Fetchen der Daten wird per Asynchronous JavaScript and XML (AJAX) unternommen. Das Skript, welches das Ziel dieser Anfrage darstellt, baut dabei eine Verbindung zu der Datenbank auf, bereitet dann die Strukturen vor, welche Plotly für das Rendern erwartet, führt eine relativ komplexe Query aus und appendiert die daraus resultierenden Daten zeilenweise an das dafür vorgesehene Feld der vorbereiteten Struktur.

```
69 $traces = array_fill(0, 3, [
70   'x' => [],
71   'y' => [],
72   'type' => 'bar',
73   'marker' => [ 'color' => '#9f1e82' ],
74   'width' => 0.35,
75 ]);
76 $layout = [ 'title' => [ 'text' => 'Quartalsübersicht OE' ], 'barmode' => 'group' ];
```

Listing 5.5: Die von Plotly erwartete Struktur in PHP

Das Filtern nach den Daten spezifisch zu der Organisationseinheit geschieht durch Prepared Structured Query Language (SQL)-Statements in PHP.

```
363 $station = $_GET['station'];
364 if ($station == 'KBR') $station = null; // This makes filtering easier
365 $should_filter = !is_null($station);
```

Listing 5.6: Zuweisen der Station zu einer Variable

Wir weisen zu der Variable „station“ einen String hinzu, welcher dem Namen der Einheit gleicht, dessen Daten abgefragt werden sollen. Sind stattdessen die Daten des gesamten Klinikums gefragt, so weisen wir den Wert null zu.

Die Idee lautet wie folgt: In der SQL-Abfrage haben wir eine WHERE-Klausel. Es wird dort abgefragt, ob „station“ null ist und wenn nicht, ob die Organisationseinheit der Abfrage mit dem Wert von „station“ übereinstimmt.

```
304   SELECT
305       COALESCE(ort, 'k.A.') AS sturzort,
306       COUNT(*) AS anzahl_stuerze
307   FROM sturz
308   WHERE ? IS NULL OR organisationseinheit = ?
```

```
309 GROUP BY sturzort
310 ORDER BY
311     CASE sturzort
312         WHEN 'Zimmer' THEN 1
313         WHEN 'Flur' THEN 2
314         WHEN 'Bad/ WC' THEN 3
315         WHEN 'Außengelände KBR' THEN 4
316         WHEN 'außerh. Station' THEN 5
317         WHEN 'k.A.' THEN 6
318     END;
```

Listing 5.7: Filtern in dem SQL-Statement

Damit fragen wir alle Reihen ab, wenn wir „station“ auf null gesetzt haben.

Dem unteren Diagramm entnimmt Hans den direkten Vergleich der Werte des gesamten Klinikums und der Einheit 11C. Das Diagramm zeigt die Entwicklung der Sturzinzidenz über alle Quartale seit Messdatenaufzeichnungsbeginn. Die Organisationseinheit weist eine in der Regel überdurchschnittliche Sturzinzidenz auf. Lediglich in dem ersten Quartal von 2025 war die Inzidenz unterdurchschnittlich.

Der Admin kann aus diesen Daten schließen, dass besonders auf der zu der Organisationseinheit 11C gehörenden Station ein Aufwand zur Sturzreduktion betrieben werden muss. Dazu kann er die Trends der Station gesondert betrachten – ein leicht abfallender Trend ist erkennbar, also scheinen Verbesserungen aufzutreten.

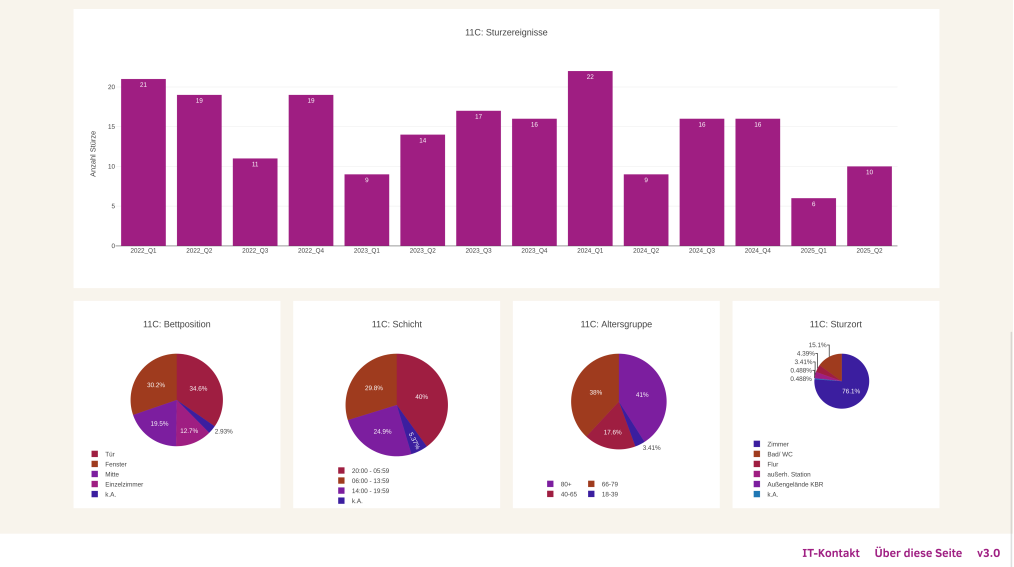

Abbildung 5.7: Screenshot der unteren Seitenhälfte

Er scrollt herunter und macht damit die zweite Hälfte der Diagrammansichten anschaubar.

Das obere Diagramm lässt die absoluten Anzahlen an Stürzen einsehen. So lassen sich besonders die aktuellen Sturzzahlen betrachten und miteinander vergleichen.

Darunter lassen sich die Pie-Charts einsehen. Hier lassen die Verteilungen von Stürzen in den Kategorien Bettposition, Schicht, Altersgruppe und Sturzort vergleichen.

Damit hat der Admin erfolgreich Daten eingefügt und ausgewertet.

5.3 Beschreibung unserer Testabläufe

Für das Testen der Anwendung haben wir uns dazu entschieden, Integrationstests zu entwickeln. Die funktionalen Anforderungen unserer Anwendung sind überschaubar. Wir hätten Unit Tests erstellen können, doch dies hätte erfordert, dass wir entweder eine weitere Dependency auf uns nehmen, um dies zu ermöglichen, oder aber wir hätten unseren Quellcode so umstrukturieren müssen, dass die Funktionen in Dateien zugreifbar sind, um sie zu testen. Dies hätte erfordert, dass die Struktur unseres Quellcodes grundlegend geändert wird, was wir als zu eingreifend für ein solch kleines Projekt eingeordnet haben.

Die Integrationstests geben eine klare Struktur an, welche durchgetestet wird. Durch die konkrete Umsetzung dieser, für welche wir uns entschieden haben, wird genau ersichtlich, bei welchem Schritt der Tests der Fehler aufgetreten ist. Damit wird auch durch diese klar, wo das Problem liegen könnte.

Wir sind so vorgegangen, dass wir eine Struktur ausgeplant haben, die uns erlaubt, einzelne Tests zu entwickeln, welche zwar auch alleinstehend funktionieren und aufrufbar sind, die sich jedoch auch von einer alles umfassenden Struktur aufrufen lassen, sodass wir einen primären Test haben, der alle anderen aufruft.

Die Tests weisen dabei gemeinsame Strukturen auf, welche wir in eine Bibliothek ausgelagert haben. Diese Bibliothek binden wir dann einfach in jedem Test ein – so haben wir es auch in dem Modul Infrastruktur gelernt. Somit können wir Prinzipien wie „Don't repeat yourself" beachten.

5.3.1 Bibliothek

```
1 #!/usr/bin/env bash
2 export BASE="http://localhost:8080/$USER-web/swe-pflegedashboard/"
3 [ -z "$JAR" ] && echo 'Define $JAR before sourcing common.sh' && exit 1
4
5 # Try to login using name and password
```

```
6  # args: <name> <pass>
7  function common_login {
8    echo "Credentials: ($1|$2)"
9    curl \
10     -c "$JAR" -b "$JAR"\
11     -X POST\
12     -d "{\"user\":\"$1\",\"pass\":\"$2\"}"\
13     "$BASE/assets/php/login.php" &>/dev/null
14 }
15
16 # Logout (delete session and associated data)
17 function common_logout {
18   curl -c "$JAR" -b "$JAR" "$BASE/assets/php/logout.php"
19 }
```

Listing 5.8: Test-Bibliothek

In unserer Bibliothek lassen sich drei zentrale Aspekte wiederfinden. Einmal gebrauchen wir an verschiedenen Stellen in jeglichen Tests die Fähigkeit, Anfragen über die Uniform Resource Locator (URL) unserer Anwendung zu stellen. Somit haben wir eine Variable namens BASE definiert.

Für das Handhaben von Sessions werden bei uns Cookies verwendet, welche wir während der Tests speichern müssen. Dafür verwenden wir, was wir ebenso in dem Modul Infrastruktur erlernt haben, curl mit den Schaltern b und c. Die Dateien, welche dabei von den Tests verwendet werden, sollten nicht identisch heißen, um zu verhindern, dass es zu Konflikten zwischen Testaufrufen kommt, welche das Ergebnis verfälschen würden.

Da diese Dateien aber in der Bibliothek gebraucht werden, stellen wir die Anforderung, dass eine Variable JAR bereits bei dem Einbinden der Bibliothek definiert ist und den Namen enthält.

Dann haben wir darüber hinaus zwei Funktionen, welche zentral für die Tests sind: Das Anmelden und das Abmelden. Für sie gebrauchen wir ebenfalls curl mit den eben genannten Schaltern und Mechanismen.

5.3.2 Integrationstests

Der erste Integrationstest, den wir angelegt haben, behandelt unser Session-Handling. Hier wird getestet, dass das Anmelden mit korrekten Daten funktioniert, das Abmelden für angemeldete Nutzende funktioniert sowie dass das versuchte Anmelden mit inkorrekten Daten eine etwa dreisekündige Wartezeit verursacht. Die Wartezeit wird nicht von dem Test automatisiert abgedeckt, aber ist manuell zu verzeichnen.

Um zu überprüfen, ob wir angemeldet sind, existiert ein PHP-Skript, welches für genau diesen Zweck sowie Debugging angelegt wurde. Seine einzige Funktion ist, zu antworten, ob man aktuell angemeldet ist.

Unser zweiter Integrationstest behandelt das Einfügen von Daten in Form einer CSV-Datei und des HTML-Formulars.

Dieser zweite Test ist wesentlich ausgiebiger als der erste Integrationstest. Wir müssen an vielen Stellen überprüfen, ob sich die aktuellen Daten, die aus der Datenbank gefetched werden, seit des letzten Fetchens geändert haben. Für diesen Zweck haben wir eine Funktion definiert.

```
28 # Updates $latest_data.
29 # Helper function
30 function compare_data {
31   old_latest_data="$latest_data"
32   latest_data="$(get_plot_data)"
33   diff <(echo "$old_latest_data") <(echo "$latest_data") &>/dev/null
34 }
```

Listing 5.9: Funktion compare_data

Die Idee ist wie folgt: In der Variable latest_data wird zu jedem Zeitpunkt der Zustand nach dem letzten Fetchen gespeichert. Bei dem Aufruf dieser Funktion aktualisieren wir diesen Wert und vergleichen den neuen gleichzeitig mit dem alten Wert.

Für das Vergleichen der Daten verwenden wir das Kommando diff. Dies erwartet zwei Dateien, wobei wir zwei Variablen miteinander vergleichen möchten. Um dies zu tun, verwenden wir Prozesssubstitution – diese erlaubt es uns, zwei Subshells zu starten und als Dateien für das Kommando zu verwenden. Alles, was wir dann tun müssen, ist es, die Werte der beiden Variablen in den Subshells auszugeben.

Da das diff-Kommando der letzte Aufruf in unserer Funktion ist, ist auch der Return-Wert des Kommandos identisch zu dem Return-Wert unserer Funktion. Dies nutzen wir dann in ausgeprägteren Strukturen für den Vergleich und damit die Überprüfung, ob zwischen Funktionsaufrufen Änderungen an den Daten in der Datenbank vorgenommen wurden.

5.3.3 Haupttest

```
1 #!/usr/bin/env bash
2
3 # Run a single test
4 function run {
5   unit="$1"
```

```
6   echo "-----   RUNNING TEST: $unit -----"
7   start_measure="${EPOCHREALTIME/[,.]}"
8   "./$unit"
9   end_measure="${EPOCHREALTIME/[,.]}"
10  echo "-----   TEST SUCCESS: $unit -----"
11  echo "Time taken: $(( (end_measure - start_measure)/1000 ))ms"
12  echo
13  echo
14 }
15
16 # Trap gets reset after all tests were successfully run
17 function fail {
18   echo "-----   TEST FAILED: $unit -----"
19 }
20 trap fail EXIT
21
22 set -e
23 run 'session-handling.sh'
24 run 'data-submission.sh'
25 trap - EXIT
```

Listing 5.10: Haupttest

Unser Haupttest ist so konzipiert, dass wir eine run-Funktion definiert haben. Diese Funktion nimmt als erstes Argument den Namen des (Integrations-)Tests an, welcher ausgeführt werden soll.

Um übersichtlicher zu gestalten, welcher Test erfolgreich durchlaufen wurde plus seine Laufzeit anzuzeigen, gibt die run-Funktion Zeilen vor und nach dem Test aus. Es wird eine visuelle Struktur geschaffen, welche das Navigieren des Outputs aller Tests vereinfachen soll.

Um diese Trennzeilen hervorzuheben, verwenden wir Emojis als farblich hervorgehobene Piktogramme, welche aufgrund technischer Limitationen von minted leider nicht im Codeblock angezeigt werden.

Die zweite Funktion, die wir für den Haupttest definiert haben, ist die fail-Funktion. Der Test bricht frühzeitig ab, wenn bei dem Durchlaufen eines Tests ein Fehler auftritt. In diesem Fall greift die Trap, welche die fail-Funktion aufruft. Es wird damit eine deutliche Fehlermeldung angezeigt, die ebenso den Namen des fehlschlagenden Tests ausgibt.

Letztlich rufen wir die Tests sequentiell auf. Durch Ändern der Laufzeitparameter der Bash wird erreicht, dass das Skript bei einem Fehler beendet. Damit sparen wir uns das Verunden aller Tests.

Laufen alle Tests erfolgreich durch, so entfernen wir die fail-Funktion von der EXIT-Trap und das Skript beendet ordinär.

5.3.4 Gefundene Fehler

Für das Einfügen von Daten verwenden wir PHP mit Prepared Statements von MySQLi. Das Einsetzen der Werte geschieht dabei also durch Variablen, welche wir dem Funktionsaufruf als Parameter mitgeben. Hier haben wir eine Schwierigkeit festgestellt, denn die Datentypen der Variablen haben einen Einfluss darauf, ob das Einfügen gelingt.

Zuerst haben wir die Funktionalität umgesetzt und manuell getestet. Als dies funktioniert hat, haben wir den Integrationstest geschrieben, welcher bestätigt hat, dass unser Vorgehen gelingt.

Wir haben danach eine Anpassung an dem Einfügen vornehmen müssen und den Test erneut ausgeführt. An dieser Stelle hat der zweite Integrationstest einen Fehler erkannt. Dadurch konnten wir den Fehler orten, beheben und dementsprechend sicherstellen, dass sich das System so verhält, wie es von uns eingeplant war.

Abseits davon haben die Tests zwar keine Fehler identifiziert, aber aufgezeigt, dass die von uns implementierten Funktionalitäten in einem von uns geplanten Umfang funktionieren – sowohl individuell als auch ineinander verzahnt.

Dies mag sich besonders für zukünftige Entwicklungen als wertvoll herausstellen, sollte es noch einmal dazu kommen, dass diese Codeabschnitte refactored werden.

5.3.5 Reflexion

Auch, wenn wir der Meinung sind, dass unsere Integrationstests dem Umfang der Anwendung gut abdecken, so gilt es letztendlich doch zu erwähnen, dass keine Unit Tests existieren. Wollte man vollständig sicherstellen, dass sich die Anwendung in vollem Umfang verhält, wie sie sollte, so kommt man schlecht darum, jede Funktion tiefgründig und individuell zu behandeln.

Darüber hinaus behandeln wir zwar Fehler- und Erfolgsfälle, jedoch lassen sich an verschiedenen Stellen noch wesentlich mehr Grenzfälle in die Tests einbauen. Diese haben wir aufgrund ihrer Menge außen vor lassen müssen. Wir meinen, trotzdem sicherheitsrelevante Probleme ausschließen zu können.

Schließlich müssen wir betonen, dass wir aktuell durch keine Tests überprüfen können, ob die Anzeigedaten korrekt sind. Wir beachten zwar die Zustände, die aus dem Backend ersichtlich werden, doch überprüfen dabei nicht die konkreten Antworten, die von dem Backend an das Frontend gesendet werden sowie das Sichtbarmachen dieser Daten im Frontend.

6 Fazit

Zusammenfassend sind wir der Meinung, dass der Umfang des Projektes sehr passend dafür war, in das Thema der Projektarbeit mit einem echten Unternehmen im Umfang von Software Engineering 1 einzusteigen.

Es wurden bei unserer Bearbeitung genau die Werkzeuge gefordert, welche uns bereits aus anderen Modulen bekannt waren. Die Modellierung aus Software Engineering 1 hat dann die Basis unserer Planung dargestellt. Trotzdem haben wir neue Dinge gelernt und anwenden müssen: Darunter sind beispielsweise der richtige Umgang mit PHP, Plotly oder das sichere Speichern von Kennwörtern.

Trotzdem wäre es für den Umfang der Ausarbeitung vorteilhaft gewesen, noch ein paar funktionale Anforderungen mehr verzeichnen zu können.

6.1 Erkenntnisse während der Teamarbeit

Bei der Zusammenarbeit mit einem Kunden kann es dazu kommen, dass die Ansprüche an das Projekt und an die Zusammenarbeit nicht statisch sind. So ändern sich Wünsche zwischen Gesprächen, was den Arbeitsaufwand deutlich erhöht. Wir denken, dass ein Lastenheft damit auch abseits rechtlicher Absicherungen von Vorteil ist und für zukünftige Zusammenarbeit in Erwägung gezogen werden sollte.

Als nächstes ist uns bei der Teamarbeit aufgefallen, dass es von Vorteil wäre, wenn Commits, bevor sie vollends in den main-Branch aufgenommen werden können, einer kleinen Überprüfung durch mindestens zwei andere Teammitglieder unterzogen werden würden. So kann vermieden werden, dass sich beispielsweise Tippfehler oder stilistische Unreinheiten in das Repository einschleichen. Es ist damit erwägenswert, das Vier-Augen-Prinzip in einer Form einzubeziehen.

In der Teamarbeit haben wir weiterhin erkannt, dass Aufgabenteilung den zeitlichen Aufwand verringern kann. Wir haben dadurch eine Art Vorausblick erhalten, wie „echte“ Aufgabenteilung bei einem Projekt verlaufen kann. Wir haben trotzdem großen Wert darauf gelegt, die Aufgaben gemeinsam zu planen, zu verstehen und nach der Implementation zu besprechen, um dem Lernanspruch, wie er beispielsweise in Infrastruktur vermittelt wurde, gerecht zu werden. Wir haben hauptsächlich Mob-Programming betrieben, also zusammen und in einem gemeinsamen digitalen Raum mit einer Bildschirmübertragung an der Anwendung programmiert.

Zuletzt haben wir erkannt, dass unser ausgewähltes Commit-Nachrichten-Format, basierend auf den Conventional Commits [1], viel Potenzial aufweist und dass wir es in ähnlichen Formen in zukünftigen Projekten verwenden möchten. Unser Hauptkritikpunkt findet seinen Fuß darin, dass wir das Scope zu breit definiert hatten. Unser Aufteilung geschah lediglich in Front-

und Backend. Wir denken, dass es hier von Vorteil ist, eine feinere Definition in einzelne Komponenten vorzunehmen. Dies werden wir dann in zukünftigen Bearbeitungen tun.

Besonders über diesen Aspekt unserer Planung haben wir uns weitgehender ausgetauscht und miteinander diskutiert.

6.2 Lessons-Learned

Generell lässt sich formulieren, dass jegliche Erkenntnisse, welche wir in der Teamarbeit erlangt haben, auch Lessons-Learned darstellen können. Wir möchten dabei besonderen Fokus auf das Anlegen eines Lastenhefts legen, welches verhindert, dass bereits verrichtete Arbeit durch neue Anforderungen überschrieben wird.

Dazu möchten wir versuchen, in zukünftiger Arbeit weniger Dateien in das Repository aufzunehmen, die sich mit Daten des Repositorys generieren lassen, insofern dies nicht notwendig ist. Auch sollten wir versuchen, Duplikate in dem Repository zu vermeiden, da dies unnötig Festplattenspeicher verwendet und dafür sorgen kann, dass die vermeintlich identischen Dateien Versionsunterschiede aufweisen.

6.3 Was lief gut? Was ginge besser?

Gut lief bei uns die Zeitplanung. Wir konnten die wöchentlichen Übungstermine gut einhalten, was uns diesem Arbeitsaufwand gegen Ende der Abgabe gespart hat.

Dazu haben wir das Gefühl, dass uns die Risikoanalyse geholfen hat, indem sie Unsicherheiten aus der Welt geräumt hat. Wir konnten besser vorplanen, wie wir vorzugehen haben, in dem Fall, dass die besprochenen Risiken eintreten.

Auch, wenn wir der Meinung sind, dass unsere Organisation gut funktioniert hat und wir dementsprechend ziemlich zufrieden damit sind, hätte man besser dokumentieren können, welche aktuellen Ziele bestehen und welche zeitnah abgearbeitet wurden. An dieser Stelle lässt sich auch das Gantt-Diagramm aufgreifen, welches einem ähnlichen Zweck dient, jedoch zielen wir in diesem Punkt auf eine weniger vorgeplante Struktur ab, sondern eine solche, welche dynamisch und im Moment geschrieben und verarbeitet wird.

Unsere relevanten UML-Diagramme entsprechen zwar dem Standard, doch haben wir das Gefühl, dass es hilfreich gewesen wäre, vor dem Erstellen im Team genau festzulegen, welches Werkzeug mit welchen Einstellungen für die Erstellung verwendet werden sollte. Aktuell weichen unsere Diagramme stilistisch ein wenig voneinander ab.

Wir sind der Meinung, dass unsere UML-Diagramme eine gute Übersicht dafür dargestellt haben, wie die Anwendung und ihre Komponenten auszusehen haben, jedoch haben wir auch das Gefühl, dass die Feinheiten in den Implementationen, welche unter Umständen auch durch besondere

Eigenschaften der verwendeten Programmiersprachen und Technologien erst in der Bearbeitung relevant geworden sind, in den Diagrammen schlecht umsetzbar sind. Die Darstellung in den UML-Diagrammen ist im Wesentlichen unabhängig von der gewählten Programmiersprache, sodass wir manchmal das Gefühl hatten, dass ebendiese Besonderheiten in der Implementierung ein weiteres Mal besprochen werden mussten. Durch weiteres Erstellen von UML-Diagrammen und Arbeiten mit diesen sollten wir sicherer im Umgang mit ihnen werden und genau solche Probleme vermeiden können.

7 Diskussion

Das Thema, welches wir uns ausgesucht haben, ist „Die Veränderung von Software Engineering“. Wir haben uns für dieses Thema entschieden, da wir es als relativ breitgefächert einschätzen. Somit konnten wir uns verschiedene Subthemen aussuchen, welche sich zu einem gefügigen Gesamtbild zu diesem Thema ergänzen.

So umfasst auch die Formulierung „Die Veränderung“ nicht lediglich die Veränderung, wie sie bereits in der Vergangenheit aufgetreten ist, sondern auch die solche, welche von heute an auftreten könnte oder sollte.

Zwei unserer ausgesuchten und von uns zusammengefassten wissenschaftlichen Artikel befassen sich mit bereits aufgetretener Veränderung, wobei die anderen drei Artikel sich eher mit aktuell bestehenden Veränderungen befassen. Einer dieser drei Artikel beschäftigt sich dabei mit sozial relevanten Themen.

7.1 Die Artikel

Der erste sich mit der Vergangenheit auseinandersetzende Artikel, welchen wir aufgreifen möchten, wertet eine Vielzahl an Daten aus, mit dem Ziel, zu untersuchen, welche Themen zu welchen Zeitpunkten in der Geschichte des Software Engineering relevant waren [6]. Der Artikel ist deswegen für unser Thema der Veränderung relevant, da er sich mit der Veränderung der Interessen im Software Engineering auseinandersetzt.

Der zweite Artikel, welcher sich ebenfalls mit der Vergangenheit des Software Engineering befasst, betrachtet fünf Iterationen der Entwicklung empirischer Methoden von den 1960ern bis einschließlich den 2010ern [7]. Dieser Artikel ist relevant für das Thema, da er die Veränderung eines gesonderten Themas im Software Engineering betrachtet.

Als nächstes wollen wir die Artikel betrachten, welche aktuelle und zukünftige Veränderungen thematisieren.

Der erste dieser Artikel befasst sich mit Unternehmen, die im Rahmen der kontinuierlichen Softwareentwicklung Nutzerfeedback systematisch erfassen und nutzen [8]. Dies passt zu unserem Thema, da der Artikel aus 2025 stammt, damit also vergleichsweise neu ist, und Prinzipien von der agilen und „lean“ Softwareentwicklung thematisiert. Dass diese die Weiterentwicklungen von statischeren Modellen wie dem Wasserfallmodell sind, haben wir bereits in Arbeitstechniken erfahren.

Der nächste Artikel beschäftigt sich mit der Verwendung von künstlicher Intelligenz in der Softwareentwicklung. Zuerst wird die aktuelle Verwendung von künstlicher Intelligenz thematisiert – aktuell sei künstliche Intelligenz nicht in der Lage, eigenständig ganze Projekte zu erstellen und biete somit nur eine Hilfestellung für die Entwickelnden [9]. Das Thema ist für uns relevant, da die Verwendung moderner Technologie wie künstliche Intelligenz behandelt wird sowie die Veränderungen, welche noch auftreten müssen, um diese als Werkzeug zu verbessern.

Der letzte Artikel behandelt einen Software-Engineering-Workshop für Studierende, welcher analyisiert wird, um die Rolle von Geschlechtern in Teamdynamiken in der agilen Softwareentwicklung aufzuzeigen [10]. Dieses Thema weicht insofern von dem übergestellten Thema ab, dass es keine Veränderungen per se beschreibt, sondern impliziert, welche Veränderungen auftreten müssen.

7.2 Die Gegenüberstellung

Aus dem ersten behandelten Artikel wird deutlich, dass sich das Gebiet des Software Engineering in vergleichbarer Geschwindigkeit zu dem Rest des Feldes der Informatik entwickele [6, S. 21]. Es seien ein paar ausgewählte Themen, unter ihnen neue Prozeduren und Techniken sowie die Entwicklung neuer Werkzeuge, welche immerwährend relevant seien [6, S. 20]. Trotzdem weise das Gebiet des Software Engineering eine hohe Vielfalt an Interessen auf, welche behandelt werden würden [6, S. 18].

Während sich der erste Artikel mit einem Gesamtüberblick über das Thema der behandelten Themen auseinandersetzt, befasst sich der zweite Artikel mit einem bestimmten Thema: Empirische Methoden. Der Artikel zeigt insgesamt auf, dass sich empirische Methoden von einer selten genutzten Ergänzung [7, S. 8] zu einer häufig angewandten Methodik [7, S. 12] entwickelt haben.

Hier lässt sich eine Verbindung zwischen den beiden Artikeln ziehen. In dem ersten Artikel wird die Kategorie „empirical software engineering“ behandelt. Es ist dabei aus beispielsweise einem Diagramm ersichtlich, dass das Thema vor dem Jahr 2000 wesentlich weniger relevant war als nach 2010 [6, S. 19]. Empirische Studien seien bereits 1988 zwar präsent jedoch weniger relevant gewesen, seien dann jedoch zu einem weitverbreiteten Thema geworden [6, S. 20]. Dies stimmt überein mit der Feststellung in dem zweiten Artikel, dass das Thema der empirischen Methoden an Relevanz gewonnen habe [7, S. 12].

Ergebnis des dritten Artikels ist, dass Unternehmen gewarnt werden, dass beispielsweise unzureichend zugeteilte Ressourcen das Auswerten der Feedbackdaten zunichtemachen könne [8, S. 45f.]. Passend dazu werden Telemetrie-Dashboards empfohlen [8, S. 43], um die Sichtbarkeit der Nutzerinformationen zu erhöhen. Insgesamt verdeutlicht der Artikel, dass es durch wachsende Mengen an Feedback und steigenden Druck durch Deadlines dazu kommen kann, dass das Analysieren des Feedbacks misslingt [8, S. 45].

Aus dem vierten Artikel geht hervor, dass Programmieren aktuell eine manuelle Tätigkeit bleibe [9, S. 128]. Künstliche Intelligenz habe aktuell Schwierigkeiten bei der „kontextbezogene[n] Interpretation“ [9, S. 123], woraus wir schlussfolgern, dass künstliche Intelligenz Schwierigkeiten bei komplexen Projektanforderungen haben sollte. Auch das Testen von Software durch künstliche Intelligenz sei noch nicht ausgereift, da die künstliche Intelligenz aktuell nicht in der Lage sei, sich an an Änderungen an der Anwendung, welche das Objekt der Tests darstellt, anzupassen [9, S. 125] [11, S. 2]. Es ist an dieser Stelle wichtig zu erwähnen, dass sich der Autor an diesem Punkt auf eine Quelle bezieht. Um den Anforderungen an unsere Diskussion, das Zitieren aus den Artikeln, gerecht zu werden, zitieren wir an dieser Stelle sowohl den Autor, welcher sich auf die Quelle bezieht, als auch die Quelle selbst.

Trotzdem weise künstliche Intelligenz gewisse Vorteile auf. Im vorherigen Artikel wurde darauf eingegangen, dass es bei der Analyse der anfallenden Nutzerdaten zu wachsenden Mengen an Daten komme [8, S. 45]. Dies sei einer der Aufgabenbereiche, für die eine künstliche Intelligenz gut geeignet sei, da sie in der Lage sei, schnell große Datenmengen zu analysieren [9, S. 121].

Der letzte Artikel beschreibt große Barrieren, die weibliche Personen davon abhalte, über technische Themen zu reden [10, S. 155f.]. Der Artikel lässt sich insofern im Bezug auf die anderen Artikel einordnen, dass in ihnen viel über die Entwicklung vom Methoden, Techniken und Werkzeugen in Software Engineering diskutiert wird [6, S. 20], aber es auch wichtig ist, den Wandel der Geschlechterrolle zu berücksichtigen. Der Artikel konkretisiert, dass weibliche Personen unterstützt und dass sich Stereotypen widersetzt werden solle [10, S. 156]. Damit behandelt dieser Artikel einen Aspekt im Software Engineering, welchen die anderen Artikel nicht behandeln.

Zusammenfassend lässt sich formulieren, dass unsere Artikel, welche verschiedene Aspekte des Software Engineering betrachten, ein stimmiges Gesamtbild ergeben. Wir konnten zwischen den verschiedenen Informationen keine Widersprüchlichkeiten finden, jedoch viele Gemeinsamkeiten untereinander verzahnen.

Literaturverzeichnis

[1] C. Commits. „Conventional Commits.“ en. Adresse: https://www.conventionalcommits.org/en/v1.0.0/.

[2] P. D. L. Bieker-Walz, *SWE 1 UML-Aktivitätsdiagramme(V06)*, Accessed: 2026-02-01, Nov. 2025.

[3] P. D. L. Bieker-Walz, *SWE 1 Sequenzdiagramm(V10)*, Accessed: 2026-02-13, Jan. 2026.

[4] P. D. L. Bieker-Walz, *SWE 1 Sequenzdiagramm(V11)*, Accessed: 2026-02-08, Jan. 2026.

[5] P. T. Inc. „Collaborative data science.“ en. Adresse: https://plot.ly.

[6] B. L. Sousa, M. M. Ferreira, K. A. M. Ferreira und M. A. S. Bigonha, „Software Engineering Evolution: The History Told by ICSE,“ in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, Ser. SBES '19, Salvador, Brazil: Association for Computing Machinery, 2019, S. 17–21, ISBN: 9781450376518. DOI: 10.1145/3350768.3350794. Adresse: https://doi.org/10.1145/3350768.3350794.

[7] M. Felderer und G. H. Travassos, „The Evolution of Empirical Methods in Software Engineering,“ in *Contemporary Empirical Methods in Software Engineering*, M. Felderer und G. H. Travassos, Hrsg. Cham: Springer International Publishing, 2020, S. 1–24, ISBN: 978-3-030-32489-6. DOI: 10.1007/978-3-030-32489-6_1. Adresse: https://doi.org/10.1007/978-3-030-32489-6_1.

[8] A. Tkalich, E. Klotins, T. Sporsem, V. Stray, N. B. Moe und A. Barbala, „User feedback in continuous software engineering: revealing the state-of-practice,“ *Empirical Software Engineering*, Jg. 30, Nr. 79, 2025. DOI: 10.1007/s10664-024-10557-2. Adresse: https://link.springer.com/article/10.1007/s10664-024-10557-2.

[9] M. Barenkamp, „Künstliche Intelligenz in der Softwareentwicklung,“ *Wirtschaftsinformatik & Management*, Jg. 12, S. 120–129, 2020. DOI: https://doi.org/10.1365/s35764-020-00235-5. Adresse: https://link.springer.com/article/10.1365/s35764-020-00235-5.

[10] G. E. Sæter, C. K. Lund und V. Stray, „Agile Software Engineering Capstone Courses: Exploring the Impact of Gender,“ in *Agile Processes in Software Engineering and Extreme Programming – Workshops*, L. Marchesi u. a., Hrsg., Cham: Springer Nature Switzerland, 2025, S. 150–158, ISBN: 978-3-031-72781-8.

[11] D. Santiago, T. King und P. Clarke, „AI-Driven Test Generation: Machines Learning from Human Testers,“ in *2018 Pacific NW Software Quality Conference*, Accessed: 2026-02-13, Pacific NW Software Quality Conference, Sep. 2018. Adresse: https://pnsqc.org/archives/wp-content/uploads/2018/09/38-Santiago-AI-Driven-Test-Generation.pdf.

Abbildungsverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML	26
CSV	Comma-Separated Values	9
CSS	Cascading Style Sheets	21
FURPS+	Functionality, Usability, Reliability, Performance, Supportability	8
HTML	Hypertext Markup Language	9
IT	Information Technology	10
MIT	Massachusetts Institute of Technology	21
PHP	PHP: Hypertext Preprocessor (ehemals Personal Home Page)	21
SQL	Structured Query Language	26
UI	User Interface	9
UML	Unified Modeling Language	4
URL	Uniform Resource Locator	29

Anhang

I Testeinfügedaten

Für das Testen der Einfügefunktionalität haben wir zwei Testdateien vorbereitet. Eine korrekt formatierte Datei, die fünf neue Datensätze einfügt, und eine inkorrekt formatierte Datei, welche einen Fehler in der dritten Zeile aufweist. In dieser Zeile befindet sich ein Komma zu viel.

Wichtig ist uns an dieser Stelle zu erwähnen, dass dieses Einfügen nicht dem echten und etwas aufwändigerem Einfügeprozess des Klinikums gleicht, sodass es durch diese etwas inkorrekte Nutzung dazu kommt, dass die Sturzinzidenzen inkorrekt angezeigt werden. Dies liegt daran, dass wir mit den Dateien Datensätze für Quartale einfügen, für welche keine Pflegetagsdaten existieren.

```
1 2025-1-1,4444444,Q2,1,87,UEB,ENTL,20:00 - 05:59,Bad/ WC,Fenster,0,NULL,1,0,0,1,4,mit Hilfsmittel,0,2,1,1,3,NULL,1,NULL
2 2025-1-2,4444444,Q3,2,87,10A,ENTL,20:00 - 05:59,Bad/ WC,Mitte,0,NULL,1,1,1,2,3,ohne Hilfsmittel,1,2,0,1,3,NULL,0,NULL
3 2025-1-3,4444444,Q1,1,87,11C,ENTL,20:00 - 05:59,außerh. Station,Tür,0,NULL,2,0,1,2,2,mit Hilfsmittel,1,1,0,1,3,NULL,1,NULL
4 2025-1-4,4444444,Q4,2,87,UEB,ENTL,14:00 - 19:59,Flur,Fenster,1,NULL,2,0,1,2,4,mit Hilfsmittel,0,2,0,1,3,NULL,1,NULL
5 2025-1-5,4444444,Q2,1,87,UEB,ENTL,06:00 - 13:59,Zimmer,Fenster,NULL,NULL,1,1,1,2,4,ohne Hilfsmittel,0,2,NULL,1,3,NULL,1,NULL
```

Listing I.1: Korrekt formatiert

```
1 2025-1-1,9999999,Q2,1,87,UEB,ENTL,20:00 - 05:59,Bad/ WC,Fenster,0,NULL,1,0,0,1,4,mit Hilfsmittel,0,2,1,1,3,NULL,1,NULL
2 2025-1-2,9999999,Q3,2,87,10A,ENTL,20:00 - 05:59,Bad/ WC,Mitte,0,NULL,1,1,1,2,3,ohne Hilfsmittel,1,2,0,1,3,NULL,0,NULL
3 2025-1-3,9999999,,Q1,1,87,11C,ENTL,20:00 - 05:59,außerh. Station,Tür,0,NULL,2,0,1,2,2,mit Hilfsmittel,1,1,0,1,3,NULL,1,NULL
4 2025-1-4,9999999,Q4,2,87,UEB,ENTL,14:00 - 19:59,Flur,Fenster,1,NULL,2,0,1,2,4,mit Hilfsmittel,0,2,0,1,3,NULL,1,NULL
5 2025-1-5,9999999,Q2,1,87,UEB,ENTL,06:00 - 13:59,Zimmer,Fenster,NULL,NULL,1,1,1,2,4,ohne Hilfsmittel,0,2,NULL,1,3,NULL,1,NULL
```

Listing I.2: Inkorrekt formatiert (Zeile 3)

Selbstständigkeitserklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit habe ich mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegt.

Bremerhaven, den 13. Februar 2026

Unterschrift:

Marvin Eckhoff

Luca Focken

Borai Manuth

Leon Stüve

Lukas Weber