Abschlussprüfung Winter 2021

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Planung und Erstellung einer automatisierten und erweiterbaren Webanwendung zur Verwaltung von Prüfungsdokumenten mit integrierter Suchfunktion

Abgabedatum: Berlin, den 09.12.2021

Prüfungsbewerber:

Kai Richter

Münsterlandstraße 38

10317 Berlin

Ausbildungsbetrieb:

Liebenow IT

Lynarstraße 15

13585 Berlin

IHK Berlin

FACHINFORMATIKER/IN FÜR ANWENDUNGSENTWICKLUNG

Protokoll zur betrieblichen Projektarbeit

Diese Erklärung ist der Dokumentation als erste Seite vor dem Deckblatt beizufügen.

Hiermit versichere ich,

Kai Richter

Vor- und Zuname des Auszubildenden

die betriebliche Projektarbeit

Planung und Erstellung einer automatisierten und erweiterbaren Webanwendung zur Verwaltung von Prüfungsdokumenten

Genaue Bezeichnung der Projektarbeit

sowie die eingereichte Dokumentation unter Betreuung von

Markus Liebenow

Vor- und Zuname des Ausbilders/ Name des Ausbildungsbetriebes

selbstständig und **ohne fremde Hilfe** konzipiert, verfasst und durchgeführt zu haben. Teile der Dokumentation, die ich **nicht selbstständig** erstellt habe, sind von mir entsprechend **gekennzeichnet** worden. Die von der Verordnung vorgesehene Richtzeit wurde eingehalten. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Ich bin darüber aufgeklärt worden, dass meine betriebliche Projektarbeit bei **Täuschungshandlungen, bzw. Ordnungsverstößen mit „Null" Punkten bewertet** wird und als nicht bestanden gilt.

Ich bin weiter darüber aufgeklärt worden, dass dies auch dann gilt, wenn festgestellt wird, dass meine Projektdokumentation im Ganzen oder zu Teilen mit der eines anderen Prüfungsteilnehmers übereinstimmt. Ich nehme zur Kenntnis, dass ggf. stichprobenartige Kontrollen durchgeführt werden können.

Berlin 08.12.2021

Ort und Datum

Unterschrift des Prüfungsteilnehmers

Unterschrift Projektverantwortliche/-r des Auftraggebers

Industrie- und Handelskammer zu Berlin
Livia Kalwa | E-Mail: livia.kalwa@berlin.ihk.de
Fasanenstraße 85 | 10623 Berlin | Tel.: +49 30 31510-380 | Fax: +49 30 31510-179

Inhaltsverzeichnis

Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API *Application Programming Interface*
CSS *Cascading Style Sheets*
ERM *Entity-Relationship-Modell*
HTML *Hypertext Markup Language*
HTTP *Hypertext Transfer Protocol*
ID *Identifikator*
IDE *integrated development environment (integrierte Entwicklungsumgebung)*
IHK *Industrie- und Handelskammer*
OCR *Optical Character Recognition (Texterkennung)*
ORM *object-relational mapper*
PC *Personal Computer*
PDF *Portable Document Format*
PHP *PHP: Hypertext Preprocessor*
SCM *Software-Configuration-Management*
URL *Uniform Resource Locator*

Literaturverzeichnis

Laravel Dokumentation: https://laravel.com/docs/8.x

Wikipedia diverse Seiten: https://de.wikipedia.org/wiki/Wikipedia:Hauptseite

PHP-Dokumentation: https://www.php.net/docs.php

Stackoverflow: https://stackoverflow.com

Google: https://www.google.com

TailwindCSS Dokumentation: https://tailwindcss.com/docs

Alpine.js Dokumentation: https://alpinejs.dev/start-here

Laracast: https://laracasts.com

Git: https://git-scm.com

1. Einleitung

1.1. Projektumfeld

In der folgenden Projektdokumentation wird der Ablauf des Abschlussprojektes, das durch den Autor im Rahmen seiner Abschlussprüfung zum Fachinformatiker mit der Fachrichtung Anwendungsentwicklung durchgeführt wurde, erläutert. Das Projekt wird in der Firma Liebenow IT durchgeführt, welche der Praktikumsbetrieb des Autors ist. Die Firma befindet sich aktuell in der Lynarstraße in Spandau in einem kleinen Büroobjekt. Dort befinden sich sechs eingerichtete Büroarbeitsplätze und ein kleines Testlabor. Der Auftraggeber, Herr Liebenow, beschäftigt derzeit insgesamt sechs Mitarbeiter: innen.

1.2. Projektbeschreibung

Es handelte sich um einen Auftrag, der in zwei Teilprojekte unterteilt wurde. Das eine Teilprojekt befasste sich mit der Planung und Konfiguration der gesamten Serverstruktur. Dieses Teilprojekt übernahm Herr Weiland.

Der Autor und Antragsteller übernahm das softwareseitige Teilprojekt, in dem eine Webanwendung erstellt werden sollte, die automatisiert Prüfungsdokumente aus dem Portable Document Format (PDF) mit Hilfe von Optical Character Recognition (OCR) in Abschnitte aufteilt und in eine Datenbank einspeist, damit eine Volltextsuche ermöglicht werden kann.

1.3. Projektziel

Es sollte eine Möglichkeit geschaffen werden den Prüfungserstellungsprozess und die Unterrichtsvorbereitung des Auftraggebers effektiver, zeitarmer und effizienter zu gestalten. Durch einfache Bedienung über Intuitive Führung und ansprechende Auflistung der Daten sollte eine angenehme Arbeit mit der Weboberfläche gewährleistet werden. Schnelles und effektives finden von Themen sollte über eine Volltextsuche innerhalb der gespeicherten Dokumente erfolgen. Die Dokumente sollten über eine Weboberfläche ansprechend dargestellt werden und sortierbar sein.

Das Erstellen neuer Dokumente mit Upload der PDF- Dateien und anschließendem automatischen Parsen und zerteilen in Abschnitte sollte eine zentrale Funktionalität haben.

Des Weiteren musste die Möglichkeit zur Bearbeitung und Löschung der Daten erstellt werden. Um den Verlust von Daten zu verhindern, sollten diese in einer gesicherten Datenbank gespeichert werden. Die Webanwendung sollte durch ein Benutzerlogin geschützt werden, um Zugriff Dritter zu verhindern.

1.4. Projektbegründung

Herr Liebenow ist Dozent und IT-Berater und musste vorher seine Unterlagen von Hand nach Themen durchsuchen, um seinen Unterricht und seine Kurse vorzubereiten. Des Weiteren erstellte er für Übungszwecke und Lernkontrollen und Prüfungen vorher manuell. Das nahm sehr viel Zeit in Anspruch. Die zu entwickelnde Anwendung sollte dem Auftraggeber durch automatische Verfahren einen Großteil der Arbeit abnehmen bzw. Zeit einsparen. Durch die Volltextsuche könnten so schnell entsprechende Themen aus den Dokumenten gesucht und anschließend geordnet dargestellt werden.

1.5. Projektschnittstellen

1.5.1. Technische Schnittstellen

Der Auftraggeber kann die Webapplikation in jedem aktuellen Browser und auf einer Vielzahl von Geräten wie PC's und Notebooks aufrufen. Die Webapplikation läuft auf einem nginx-Webserver[1] mit MySQL[2] und PHP[3]. Zusätzlich wird eine externe Lösung zum Parsen benötigt da die Entwicklung eines eigenen PDF- Parsers die Zeitvorgabe der IHK überschreiten würde.

1.5.2. Personalschnittstellen und Projektabgrenzung

Da es sich um ein Teilprojekt handelte, und die Erstellung und Konfiguration der Serverstruktur in den Aufgabenbereich von Herrn Weiland fiel, grenzte ich diese Tätigkeiten aus dem Projekt aus. Außerdem wurde das Layout und Design von Frau Hybsier geplant und erstellt.

2. Projektplanung

Zur Ermittlung der Gesamtdauer des ganzen Projektes wurde ein Netzplan erstellt. Dieser stellt die zeitliche und logische Abfolge des Projektes dar und kennzeichnet den Kritischen Pfad. Der Netzplan und die dazugehörige Vorgangsliste befindet sich im Anhang A.1: Netzplan und Vorgangsliste.

2.1. Projektphasen

Es standen, wie es die IHK Berlin vorschreibt, 70 Stunden für die Umsetzung des Projektes zur Verfügung. Um den kompletten Prozess der Softwareentwicklung abzudecken, wurde die Aufteilung in verschiedene Phasen vor dem Projektbeginn vorgenommen. Die grobe Zeitplanung mit den Hauptphasen lässt sich aus Tabelle 1: Grobe Zeitplanung entnehmen. Die detaillierte Zeitplanung mit den einzelnen Schritten der unterschiedlichen Phasen findet sich im Anhang A.2: Detaillierte Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	8 h
Entwurfsphase	13 h
Implementierungsphase	32 h
Abnahmephase	3 h
Dokumentation	14 h
Gesamt	**70 h**

Tabelle 1: Grobe Zeitplanung

[1] Nginx ist eine Webserver-Software (siehe https://de.wikipedia.org/wiki/Nginx)
[2] MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme. (siehe https://de.wikipedia.org/wiki/MySQL)
[3] PHP ist eine weitverbreitete Open Source Skriptsprache speziell für Webentwicklungen. (siehe https://www.php.net/manual/de/faq.general.php)

2.2. Ressourcenplanung

Im Anhang A.3: Verwendete Ressourcen wurden alle verwendeten Ressourcen aufgelistet, die im Rahmen des Projektes verwendet wurden. Neben Hard- und Softwareressourcen, wurde auch das Personal in die Planung mit einbezogen.

2.3. Entwicklungsprozess

Es musste von dem Autor ein Vorgehensmodell gewählt werden, um dem Projektablauf einen organisatorischen Rahmen zu geben. Durch diesen wurde das Projekt strukturiert durchgeführt.

Da die Anforderungen durch den Auftraggeber überwiegend eindeutig definiert wurden, konnte das Projekt sich an den Phasen des erweiterten Wasserfallmodells orientieren. Durch das erweiterte Wasserfallmodell hatte der Autor die Möglichkeit, sofern in der aktuellen Phase Probleme auftreten sollten, diese auf der aktuellen oder vorherigen Phase zu beheben.

3. Analysephase

3.1. Ist- Analyse

Wie bereits unter 1.3 (Projektbegründung) beschrieben muss der Auftraggeber seine Dokumente von Hand nach Themen durchsuchen, um seinen Unterricht und seine Kurse vorzubereiten. Die Menge an Dokumenten wird sich in Zukunft immer weiter steigern und somit den zeitlichen Aufwand erhöhen, um diese zu durchsuchen. Da diese Methode nicht mehr zeitgemäß ist sollte dieses Projekt umgesetzt werden.

3.2. Soll- Konzept

Um die Probleme, die in der Ist- Analyse angesprochen wurden zu lösen, wurde ein automatisiertes System konzipiert.

Die PDF- Dateien sollen über die Webanwendung auf den Server hochgeladen und gespeichert werden. Anschließend wird die Datei an den Parsingprozess weitergeleitet, durch Optical Character Recognition (OCR) liefert dieser Prozess als Resultat die Textinformationen der PDF- Datei zurück. Diese Textinformationen werden dann in Abschnitte aufgeteilt und zusammen mit den Dokumentdaten in der Datenbank gespeichert.

Nachfolgend wird der Datenbankeintrag für das entsprechende Dokument mit der auf dem Server gespeicherten Datei verknüpft. Durch die geparsten Textinformationen kann die Volltextsuche realisiert werden. Zusätzlich soll der Auftraggeber die Dokumentdaten ändern und löschen können. Der Auftraggeber möchte das die Daten optisch ansprechend und geordnet aufgelistet werden. Zudem soll es eine Sortierfunktion geben, um die Dokumente z.B. nach Jahr, Beruf, Fachrichtung und/ oder Art zu sortieren. Da es sich um private vertrauliche Daten handelt, soll ein Benutzerlogin den unbefugten Zugriff Dritter verhindern.

Im Anhang A.4: Auszug: Lastenheft ein Auszug aus dem Lastenheft zu finden.

3.3. Wirtschaftlichkeitsanalyse

Aufgrund der Probleme der vorherigen Unterrichts- und Kursvorbereitung, die in den Abschnitten 1.3 (Projektbegründung) und 3.1 (Ist-Analyse) geschildert und erläutert wurden, war die Umsetzung des Projektes unbedingt erforderlich. Ob die Realisierung aber auch aus wirtschaftlichen Gesichtspunkten gerechtfertigt war, soll in den folgenden Abschnitten geklärt werden.

3.3.1. Make or Buy- Entscheidung

Da auf dem Markt keine Lösung für die speziellen Anforderungen des Auftraggebers gefunden werden konnte, wurde dieses Projekt in Eigenentwicklung durchgeführt.

3.3.2. Projektkosten

Im Folgenden wurden die Projektkosten, die während der Entwicklung des gesamten Projektes anfielen, kalkuliert. Für die Mitarbeiter wurde mit einem Stundensatz von 12€/h kalkuliert. Für den Geschäftsführer Herr Liebenow wurde ein Stundensatz von 100€/h festgelegt. Die aufgeführten Stundensätze setzten sich aus dem Bruttostundenlohn und den Sozialaufwendungen des Arbeitgebers zusammen.
Neben den Personalkosten mussten auch noch die Aufwendungen für die Ressourcen die im Anhang A.3: Verwendete Ressourcen aufgelistet wurden, berücksichtigt werden. Des Weiteren mussten die Gemeinkosten (Miete, Strom, Gas, Internet usw.) berücksichtigt werden. Für diese Kosten wird ein Stundensatz von 10€/h veranschlagt.

Die Kosten, die für die einzelnen Vorgänge des Projektes anfielen, sowie die gesamten Projektkosten lassen sich der Tabelle 2: Kostenaufstellung entnehmen.

Es ist anzumerken, dass die Zeit, die für das Schreiben dieser IHK- Dokumentation im Projektantrag eingeplant wurde, nicht in die Projektkostenrechnung einfließt, da sie vom Auftraggeber nicht angefragt wurde und so nicht zu den eigentlichen Projektkosten gehört.

Die Personalkosten berechnen sich aus Zeit * Stundenkostensatz.

Die Ressourcenkosten berechnen sich aus Zeit * Stundenkostensatz.

Vorgang	Mitarbeiter	Zeit	Personal	Ressourcen	Gesamt
Entwicklungskosten	Herr Richter	60 h	720 €	600 €	1320 €
Serverentwicklung	Herr Weiland	28 h	336 €	280 €	616 €
Abnahme	Herr Liebenow	4 h	400 €	40 €	440 €
	Herr Weiland		48 €	40 €	88 €
	Herr Richter		48 €	40 €	88 €
Layout/Design	Frau Hybsier	6 h	72 €	60 €	132 €
				Projektkosten gesamt:	2.552 €

Tabelle 2: Kostenaufstellung

3.3.3. Amortisationsdauer

Durch die Webanwendung reduzierte sich die Summe der benötigen Zeit zur Vorbereitung der Kurse und des Unterrichts pro Tag um ca. 15 Minuten, das entspricht 5 Stunden im Monat bei 5 Arbeitstagen in der Woche. (15 min * 5 Tage * 4 Wochen = 300 min ≈ 5h) Daraus ergibt sich folgende Amortisationsgleichung

$$Amortisationszeit = \frac{2.552\ €}{5\ \frac{h}{Monat} * 100\ \frac{€}{h}} = 5{,}104 \approx 5\ \text{Monate}$$

Nach ungefähr 5 Monaten sind die Kosten für die Entwicklung der Anwendung, von den durch sie entstehenden Einsparungen, gedeckt. Ein entsprechendes Diagramm liegt im Anhang A5 Amortisationsdiagramm.

3.4. Anwendungsfälle

Im Laufe der Analysephase wurde ein Use-Case-Diagramm erstellt, um eine grobe Übersicht über die Anwendungsfälle zu erhalten. Dieses Diagramm befindet sich im Anhang A.6: Use-Case-Diagramm und bildet alle Funktionen ab, die aus Endanwendersicht benötigt werden.

4. Entwurfsphase

4.1. Zielplattform

Eine Webanwendung hätte sich mit vielen verschiedenen Programmiersprachen und/oder Frameworks realisieren lassen. Da der Autor mit der Programmiersprache PHP am meisten Erfahrung hatte und die Zeitvorgabe der IHK keinen Lernprozess für eine komplett neue unbekannte Programmiersprache zulässt, wurde sich für PHP entschieden. Um den Entwicklungsprozess zu erleichtern und Zeit zu sparen wurde ein PHP-Framework verwendet.

Ein Framework ist ein Programmiergerüst für Entwickler, es kann als Vorprogrammierung verstanden werden. Verschiedene Funktionen und Elemente sind bereits enthalten und müssen vom Entwickler nicht jedes Mal neu programmiert werden. Die zwei bekanntesten PHP- Frameworks sind Laravel[4] und Symfony[5]. Die Entscheidung, welches dieser beiden Frameworks verwendet wurde, wurde mit Hilfe einer Nutzwertanalyse entschieden, diese ist im Anhang A:7 Nutzwertanalyse zu finden.

An diesem Punkt wurde mit Herr Weiland Rücksprache gehalten, weil sich sein Teilprojekt mit der Serverstruktur und Entwicklungsumgebung befasste und er die nötigen Installationen und Konfigurationen durchführen musste.

[4] Laravel ist ein PHP- Framework (siehe https://laravel.com)

[5] Symfony ist ein PHP- Framework (siehe https://symfony.com)

4.2. Architekturdesign

Die Umsetzung des Projektes erfolgte auf Basis des *MVC*[6] Konzeptes, weil Laravel diesem Konzept folgt. *MVC*[7] ist ein Muster zur Unterteilung einer Software in die drei Komponenten model, view, controller. Jede Komponente hat einen speziellen Aufgabenbereich, der eine spätere Erweiterung oder Änderung erleichtert und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglicht.

Das model, auch Datenmodell genannt, enthält die Daten. Diese kommen in der Regel aus einer Datenbank. Die view, auch Präsentation genannt, ist für die Darstellung der Daten des Datenmodells verantwortlich. Zusätzlich werden über die Präsentation auch die Benutzerinteraktionen realisiert. Die Steuerung des Datenmodells und der Präsentation wird vom controller, auch Steuerung genannt, verwaltet. Dieser erhält von der Präsentation Informationen über die Benutzerinteraktionen, wertet diese aus und nimmt Anpassungen an der Präsentation sowie Änderungen an den Daten im Modell vor.

4.3. Entwurf der Benutzeroberfläche

Um die neue Webanwendung möglichst benutzerfreundlich bedienen zu können, sollte eine klar strukturierte, einfache Benutzeroberfläche entwickelt werden. Mit Hilfe von Mockups, die von Frau Hybsier erstellt wurden, wurde die Oberfläche mit Platzhaltern angefertigt. Auf der linken Seite sollten fixe Bedienelemente (Logout, Einstellungen, Volltextsuche) dargestellt werden. Auf der rechten Seite sollten die Sortierfunktion und darunter eine Tabellarische Darstellung der Dokumente erstellt werden. Dabei sollte die Tabellarische Darstellung durch Pagination (Seitennummerierung) begrenzt werden, damit nicht gescrollt werden muss.

Die Entwürfe befinden sich im Anhang A8: Mockups.

4.4. Datenmodell

Da keine Datenbank mit den Dokumentendaten vorhanden war, musste sich der Autor Gedanken machen, wie sich diese realisieren lässt. Zuerst musste der Autor überlegen welche Daten relevant sind und in welcher Beziehung diese Entitäten zueinanderstehen.

Der Autor entschied sich die Dokumente als **pruefung** zu deklarieren. Diese pruefungsentität steht in Beziehung zu folgenden Entitäten: **fachrichtung**, **beruf**, **art** und **abschnitt**. Jede Entität (Entity) enthält wichtige Eigenschaften die später in einer relationalen Datenbank (MySQL) in Form einer Tabelle abgebildet werden.

In Anhang A9: Entity-Relationship-Modell wird ein ERM dargestellt, welches lediglich Entitäten, Eigenschaften, Relationen und die dazugehörigen Kardinalitäten enthält.

4.5. Externer Parser

Wie bereits in 1.5.1. Technische Schnittstellen angedeutet, wäre die Erstellung eines eigenen PDF- Parsers zu Zeitaufwendig. Aus diesem Grund musste von dem Autor eine passende externe Lösung gefunden werden. Während der Recherche im Internet nach einem geeigneten Produkt, fand der Autor einige vielversprechende Lösungen. Er entschied sich für

[6] MVC Model View Controller (siehe https://de.wikipedia.org/wiki/Model_View_Controller)

[7] MVC Model View Controller (siehe https://de.wikipedia.org/wiki/Model_View_Controller)

eine Open Source Lösung, um Kosten zu einzusparen. Zusätzlich bot diese Lösung eine bereits optimierte Variante für deutschsprachige Texte, indem Umlaute erkannt werden können. Dieser externe Parser bietet eine Docker Installation (docker-compose konfiguration), welche für Herrn Weiland eine optimale Möglichkeit zum Einbinden in seine Serverstruktur bietet. Dieser Docker beinhaltet eine API welche benötigt wird, um ihm die PDF- Dateien zu senden und die geparsten Daten von diesem zu erhalten.

4.6. Geschäftslogik

Die Geschäftslogik setzt sich auch mehreren Controllern zusammen. Diese steuern die Views und Models und verarbeiten die relevanten Daten. Im *SessionsController* wird die Logik für das Benutzerlogin erstellt. Darunter fallen bspw. die Validierung der Benutzereingaben, Authentifizierung und Passwortwiederherstellungsfunktion.

Der *AppController* wird die Steuerung der Webapplikation im Allgemeinen übernehmen. Dieses beinhaltet unteranderem die Steuerung der View, in der die Dokumentdaten tabellarisch dargestellt werden. Des Weiteren wird er den komplexen Parsingprozess beinhalten, der die Dokumente zum Parser sendet und dessen Antwort verarbeitet.

Der *PruefungsController* und der *AbschnittsController* werden die Logik zur Verwaltung der beiden dazugehörigen Models (*Pruefung*, *Abschnitt*) und deren Views enthalten.

4.7. Planung Qualitätssichernder Maßnahmen

Zur Qualitätssicherung wird ein automatischer Test erstellt, um die Funktionalitäten des Benutzerlogins zu testen. Durch diesen wird sichergestellt das diese sicherheitskritische Funktion einwandfrei funktioniert und somit Dritte keinen unberechtigten Zugang erhalten können.

Der Parsingprozess wird durch einen White-box-test vom Autor getestet. Da dieser eine wichtige Anforderung an die Applikation ist und demnach sehr ausführlich getestet werden muss.

5. Implementierungsphase

Während der gesamten Implementierungsphase wurde mit Eloquent gearbeitet. Eloquent ist ein object-relational mapper (ORM), dass für jede Datenbanktabelle ein dazugehöriges Model erstellt, welches mit dieser Tabelle interagiert. Neben dem Abrufen von Datensätzen aus der Datenbanktabelle ermöglichen die Eloquent-Modelle auch das Einfügen, Aktualisieren und Löschen von Datensätzen.

Alle Eloquent Methoden liefern eine Collection als Rückgabewert. Collections sind ein weiteres Laravel Feature, mit ihnen arbeitet es sich wie mit einem PHP- Array. Allerdings sind sie praktischer, denn sie erlauben die Verkettung von Methoden, welche das Arbeiten mit ihnen erleichtert.

5.1. Implementierung der Datenstrukturen

Die Datenbankverbindung wird von Laravel bereits als Feature angeboten, weshalb keine Programmierung der Verbindung nötig war. Es musste lediglich die */config/database.php* geändert werden. Diese Vorbereitung gehörte zum Teilprojekt von Herrn Weiland, weshalb er diese Konfiguration vornahm.

Ein weiteres Laravel Feature ist Artisan, eine Befehlszeilenschnittstelle, die dem Autor mit einer Reihe von hilfreichen Befehlen die Erstellung der Anwendung erleichtert. Während der Implementierungsphase wurde Artisan immer wieder verwendet, um z.B. Views oder Klassen zu erstellen. Laravel legt bei der Initialisierung eines neuen Projektes eine bestimmte Ordnerstruktur an. In dieser Struktur haben alle Elemente einen bestimmten Platz. Das bedeutet das z.B. die Views in */resources/views* angelegt werden sollten, damit später die Pfade automatisch generiert werden können. Das spart nicht nur Zeit, sondern sorgt auch für eine Struktur, die bei allen mit Laravel erstellten Programmen ähnlich ist und somit in eine kürzere Einarbeitungszeit neuer Mitarbeiter resultiert.

Nach dem Umwandeln des ERM in eine Tabelleform konnte mit der Erstellung der Datenbanktabellen begonnen werden. Dies geschah über die Kommandozeile mit Hilfe des Artisanbefehls *make:migration ‚tabellenname'*. In der von Artisan erstellten Datei mussten dann die Eigenschaften der Entität inklusive Datentyp und Bezeichnung hinzugefügt oder geändert werden. Zusätzlich zur Migration wurden mit dem Artisanbefehl *make:model ‚modelname'* die Klassen für die Models erstellt. In dieser musste dann die Beziehung zu der entsprechenden anderen Klasse geschrieben werden.

Zu jeder Modelklasse wurden über den Artisanbefehl *make:factory ‚factoryname'* Factorys angelegt, um später über den Artisanbefehl *db:seed* die Datenbank mit „Dummy Daten" zufüllen. Das hatte den Vorteil das später Datensätze vorhanden waren und die Erstellung der Benutzeroberfläche damit vereinfacht wurde.

Da diese Befehle ohne Fehlermeldung durchgeführt werden konnten, musste die Datenbankanbindung nicht extra getestet werden. Da dies zeigte das die Anbindung und deren Konfiguration einwandfrei funktionierte.

Beispiele dieser Dateien befinden sich im Anhang B1: Migrations, B2: Models und B3: Factorys.

5.2. Implementierung der Benutzeroberfläche

Die von Frau Hybsier erstellten Mockups dienten als Vorlage für die Benutzeroberfläche, diese befinden sich im Anhang A7: Mockups auf Seite E.

Da es sich um eine Webanwendung handelt, wurde die Benutzeroberfläche mit HTML erstellt. Für die Gestaltung wurde Tailwind CSS verwendet. Tailwind CSS ist ein CSS- Framework mit einem fest definierten Befehlssatz, der nach eigenen Ansprüchen angepasst werden kann. Dadurch war eine schnellere Entwicklung möglich, weil kein separates CSS geschrieben werden musste sondern inline mit den Befehlssätzen von Tailwind gearbeitet werden konnte.

Für die JavaScript Funktionalitäten, wie z.B. Dropdown Menüs, wurde AlpineJS verwendet. Dieses minimalistische Framework erlaubte es in kürzester Zeit einfache JavaScript Funktionalitäten zu implementieren ohne lange Zeilen JavaScript Code schreiben zu müssen.

Zusätzlich kam ein weiteres Laravel Feature zum Einsatz, die Blade Templates. Die Verwendung dieser Template Engine ermöglichte unteranderem das Erstellen von Komponenten, die in der Webanwendung eine Wiederverwendung von Code ermöglichten.

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A10: Screenshots der Anwendung.

5.3. Implementierung der Geschäftslogik

Während der Entwicklung der Benutzeroberfläche wurden schon Teile der Geschäftslogik implementiert. Unter anderem mussten die Routen erstellt werden, diese entscheiden was für

eine Aktion bei welcher URL ausgeführt wird. Diese Aktionen wurden den entsprechenden Controllern zugeordnet. Meistens waren dies einfache Aufrufe einer View ohne größere Logik, um die Oberfläche korrekt erstellen zu können. Später wurden diesen Basisfunktionen komplexere Funktionalitäten hinzugefügt.

Zuerst musste sich der Autor um die Funktionalität des Benutzerlogins kümmern. Da dieses bereits vorkonfiguriert von Laravel erstellt wird, musste der Autor nur einige Stellen am Code anpassen. Weitere mit dem Login verbundenen Funktionen wie bspw. Logout konnten durch Laravels integrierte Helferfunktionen schnell umgesetzt werden.

Ein Ausschnitt des SessionControllers ist im Anhang B4: SessionController abgebildet.

Da der Parsingprozess und die Weiterverarbeitung der Daten nach dem Upload einer Datei erfolgt und der Benutzer währenddessen weiter mit der Anwendung arbeiten können soll, muss dieser Prozess im Hintergrund ausgeführt werden. Für diese Prozesse bietet Laravel ein weiteres Feature, die sogenannten Queues (Jobs). Queues lassen sich in der */config/queue.php* konfigurieren. Sie werden mit dem Artisanbefehl *make:job ‚jobname'* erstellt. Zum Abarbeiten der Queues wird ein Worker benötigt. Da der Worker ein Systemprozess ist und das Erstellen und Verwalten von Systemprozessen in den Bereich von Herrn Weiland fiel, musste an dieser Stelle mit ihm kommuniziert werden.

In dem erstellten Job musste dann die Logik implementiert werden. Zuerst musste dem Parser per HTTP- request (Anfrage) die PDF- Datei gesendet werden, dieser lieferte dann als response (Antwort) eine ID. Mit dieser ID wurde die API des Parsers abgefragt und man erhält die geparsten Daten des Dokuments. Anschließend werden diese Daten in Abschnitte eingeteilt, dem korrekten Dokument zugeordnet und in der Datenbank gespeichert. Zudem wurden unnötige Sonderzeichen entfernt, um Speicherplatz zu sparen.

Im Anhang B5: ParsingJob befindet sich ein Auszug des oben beschriebenen Jobs.

Im Anschluss wurde vom Autor die Such- und Sortierfunktion implementiert, indem er in der Modelklasse *Pruefung* einen Scope Filter definierte. Je nach Querystring in der URL wird so eine andere Collection der View übergeben und damit ändern sich die in der Tabelle dargestellten Daten für den Benutzer. Dafür wurden Links in die Blade- Component- Seiten eingefügt, in der die Daten dargestellt werden. Je nach Link ändert bzw. ergänzt sich der Querystring in der URL und somit auch das dargestellte Ergebnis.

Der Quellcode des Scope Filters befindet sich im Anhang B6: Scope Filter.

5.4. Versionierung

Während des Projektes wurde in regelmäßigen Abständen z.B. nach Abschluss einer Funktion, über die IDE Visual Studio Code, der aktuelle Stand festgehalten. Hierfür wurde das bereits integrierte SCM, welche Git[8] integriert, konfiguriert und benutzt.

5.5. Abschließende Testphase und Fehlerbehandlung

Die unter 4.3 Maßnahmen zur Qualitätssicherung aufgeführten Tests wurden von dem Autor zur Qualitätssicherung erstellt und durchgeführt.

[8] Git is a free and open source distributed version control system… (siehe https://git-scm.com)

5.5.1. Automatische Tests

Zur Erstellung der Logintests wurde der Artisanbefehl: *make:test ‚testname'* verwendet und anschließend wurden in der automatisch generierten Datei die Testfälle implementiert. Im Anhang B7: Login Tests befindet sich ein Ausschnitt dieser Datei. Anschließend konnte, über die Konsole mit Hilfe des Artisanbefehl: *test*, der Test ausgeführt werden. Ein Screenshot der Konsolenausgabe nach diesem Test befindet sich im Anhang A11: Login Test Konsolenausgabe.

5.5.2. Manuelle Tests

Zum Testen des Parsingprozesses führte der Autor einen White-Box-Test durch. Dazu hat der Autor neue Prüfungen hinzugefügt und die dazugehörigen PDF- Dateien hochgeladen. Anschließend konnte er mit Hilfe von TablePlus[9] auf die Datenbank zugreifen und die erwarteten Ergebnisse mit den tatsächlichen Ergebnissen vergleichen. Bei einigen dieser Testversuchen traten Unterschiede in den Ergebnissen auf. Daraufhin musste der Autor herausfinden warum diese Unterschiede auftraten. Bei der Auswertung der PDF- Dateien fiel auf das einige PDF- Dateien horizontal gedreht waren, diese konnte der Parser nicht korrekt interpretieren, weshalb er falsche Textdaten zurücksendete. Da der Parser eine nicht vom Autor erstellte Anwendung ist, konnte er dieses Problem nicht beheben. Deshalb entschied er sich auf der Seite mit dem Upload einen Hinweis hinzuzufügen, der diese PDF- Anforderungen beschreibt.

In diesem Zusammenhang wurden auch die anderen Funktionalitäten wie bspw. die Sortier- und Suchfunktion, das Löschen und das Ändern der Dokumente, getestet. Dabei wurde festgestellt, wenn mit bestimmten Sonderzeichen gesucht wird (? oder &) oder z.B. in einer Berufs- oder Fachrichtungsbezeichnung ein &- Zeichen vorkommt, dass dann nicht die erwarteten Suchergebnisse angezeigt wurden. Das lag daran das diese Sonderzeichen eine spezielle Bedeutung innerhalb eines Querystrings haben und somit falsch interpretiert wurden. Dieses Problem wurde dann mit einer Umwandlung dieser Sonderzeichen gelöst.

6. Deployment und Abnahme

Nach erneutem Testen mit korrekten PDF- Dateien wurden keine weiteren Probleme gefunden und so konnte die Anwendung an Herrn Weiland übergeben werden, damit dieser das Deployment durchführen konnte.

Im Anschluss wurden die vom Auftraggeber bereitgestellten PDF- Dateien über die erstellte Webanwendung in die Datenbank eingepflegt. Zu diesem Zeitpunkt waren alle relevanten Dokumente des Auftraggebers in der Anwendung verfügbar und sie konnte an den Auftraggeber übergeben werden.

Zu diesem Zweck wurde ein Meeting mit Herr Weiland, dem Autor und dem Auftraggeber abgehalten, in dem das Projekt dem Auftraggeber präsentiert wurde. Der Auftraggeber hatte die Gelegenheit die Anwendung zu testen und ggf. Fragen zu stellen und/oder Erweiterungswünsche, die in einem späteren Projekt realisiert werden könnten, zu äußern.

[9] TablePlus ist eine GUI für relationale Datenbanken (siehe https://tableplus.com)

7. Dokumentation

Da während der Planung bereits viele der verwendeten Anlagen und Inhalte erstellt werden mussten, wurden diese Teile der Dokumentation bereits während der Durchführung des Projektes erstellt. Die restlichen Teile der Dokumentation erstellte der Autor nach Durchführung des Projektes.

Die Erstellung der Entwicklerdokumentation musste auf Grund von zeitlichen Problemen weggelassen werden, da sonst die Zeitvorgabe der IHK nicht eingehalten werden konnte. Eine genauere Erläuterung der Umstände befindet sich im nachfolgenden Abschnitt.

8. Fazit

8.1. Soll-/Ist- Vergleich

Der im Abschnitt 2.1 Projektphasen erstellte Projektplan konnte im gesamten zwar eingehalten werden allerdings musste dafür die Entwicklerdokumentation entfallen. Der Autor hat sich selbst ein wenig überschätzt und die Implementierungsphase dauerte länger als geplant, da er ein ihm unbekanntes Framework mit sehr vielen Features und Konventionen während der Projektdurchführung lernen musste. In der Tabelle 3: Soll-/Ist- Vergleich wird die Zeit, die tatsächlich für die einzelnen Phasen benötigt wurde, der zuvor eingeplanten Zeit gegenübergestellt.

Projektphase	Soll- Zeit in Stunden	Ist- Zeit in Stunden	Differenz in Stunden
Analyse	8	8	0
Entwurf	13	13	0
Implementierung	32	36	+ 4
Abnahme	3	3	0
Dokumentation	14	10	- 4
gesamt	70	70	0

Tabelle 3: Soll-/Ist- Vergleich

Dennoch konnte das Projekt zur vollen Zufriedenheit des Auftraggebers abgeschlossen werden und da eine Erweiterung bereits in Planung ist, besteht in dem nächsten Projekt die Möglichkeit die Entwicklerdokumentation nachzuholen.

8.2. Gewonnene Erkenntnisse

Das IHK- Abschlussprojekt war für den Autor eine große Bereicherung. Er konnte sich selbst sehr viel neues Wissen aneignen, sowohl über Laravel und dessen Features als auch über die Wichtigkeit von z.B. UML, Dokumentation und Diagrammen. Des Weiteren konnte er bereits vorhandenes Wissen festigen, indem er selbstständig diese Dokumentation schreiben musste. In diesem Zusammenhang kann das Projekt nach der Meinung des Autors und dessen Auftraggebers als großer Erfolg bezeichnet werden, welcher sowohl einen didaktischen als auch einen praktischen Mehrwert lieferte.

8.3. Ausblick

Wie bereits im Abschnitt 8.1 Soll-/Ist- Vergleich erwähnt befindet sich eine Erweiterung der Webanwendung bereits in Planung. Während des Projektes hatte das Team bereits viele Ideen zur Verbesserung bzw. Erweiterung. Da das Projekt ein großer Erfolg war, hat der Auftraggeber sich entschieden weitere finanzielle Mittel und personelle Ressourcen zur Verfügung zu stellen, um die Erweiterungen und Verbesserungen zu realisieren.

Da die Anwendung bereits erweiterbar geplant wurde, wird das Implementieren weiterer Funktionen, wie z.B. mehrere Benutzer mit unterschiedlichen Rechten zu verwalten, Dokumente als Favoriten zu markieren oder Verbesserungen an der Benutzeroberfläche vorzunehmen, kein Problem darstellen.

Anhang

A1: Netzplan und Vorgangsliste

Vorgangs-nummer	Vorgang	Zeit in h	Vorgänger	Nachfolger	Bearbeiter
1	Unterstützen des Auftraggebers beim Erstellen des Lastenheftes und Durchführung einer Ist-Analyse	4 (2 + 2)	x	2	1
2	Nutzwertanalyse (Framework) und Amortisationsrechnung durchführen	4 (2+2)	1	3	1
3	Teammeeting	3 + 2	2	4,5	1, 2
4	ER-Modell der Datenbank erstellen	3	3	6	1
5	Planung der Konfigurationen für die Entwicklerumgebung (Serverarchitektur)	4	3	9	2
6	Erstellen der UML Diagramme und Planung der Softwarearchitektur	5 (3 + 2)	4	7	1
7	Planung qualitätssichernder Maßnahmen	2	6	8	1
8	Einrichten der Entwicklungsumgebung (Software)	2	7	15	1
9	Planung der Konfigurationen für die Produktionsumgebung	2	5	10	2
10	Planung der Backup-Lösung	1	9	11	2
11	Planung und Erstellung eines Kostenplans	1	10	12	2
12	Planung der qualitätssichernden Maßnahmen	1	11	13	2
13	Realisierung der geplanten Entwicklerumgebung	9	12	14, 15	2
14	Realisierung beziehungsweise ändern der Konfiguration der Container für eine Produktionsumgebung	4	13	22	2
15	Implementieren der Oberfläche	3	13	16	1

16	Implementierung der Datenbankanbindung incl. Testen der Datenbankanbindung	5 (4 + 1)	15	17	1
17	Erstellung der Login Funktion	2	16	18	1
18	OCR Funktion erstellen und testen	8	17	19	1
19	Darstellung der Daten mit Sortierfunktion	4	18	20	1
20	Implementierung der Volltextsuche incl. Testen der Volltextsuche	3 (2+ 1)	19	21	1
21	Abschließende Testphase der Anwendung incl. Fehlerbehebung	5 (3 + 2)	20	22	1
22	Sichere Anbindung ans öffentliche Netz	1	21,14	23	2
23	Durchführung der geplanten qualitätssichernden Maßnahmen	2	22	24,25	2
24	IHK- und Entwicklerdokumentation Kai Richter	14 (10 + 4)	23	26	1
25	Dokumentation Sascha Weiland	6	23	26	2
26	Abnahme durch den Auftraggeber und Projektfazit	3 + 1	24	x	1, 2

Mitarbeiter Herr Kai Richter = 1

Mitarbeiter Herr Sascha Weiland = 2

Netzplan:

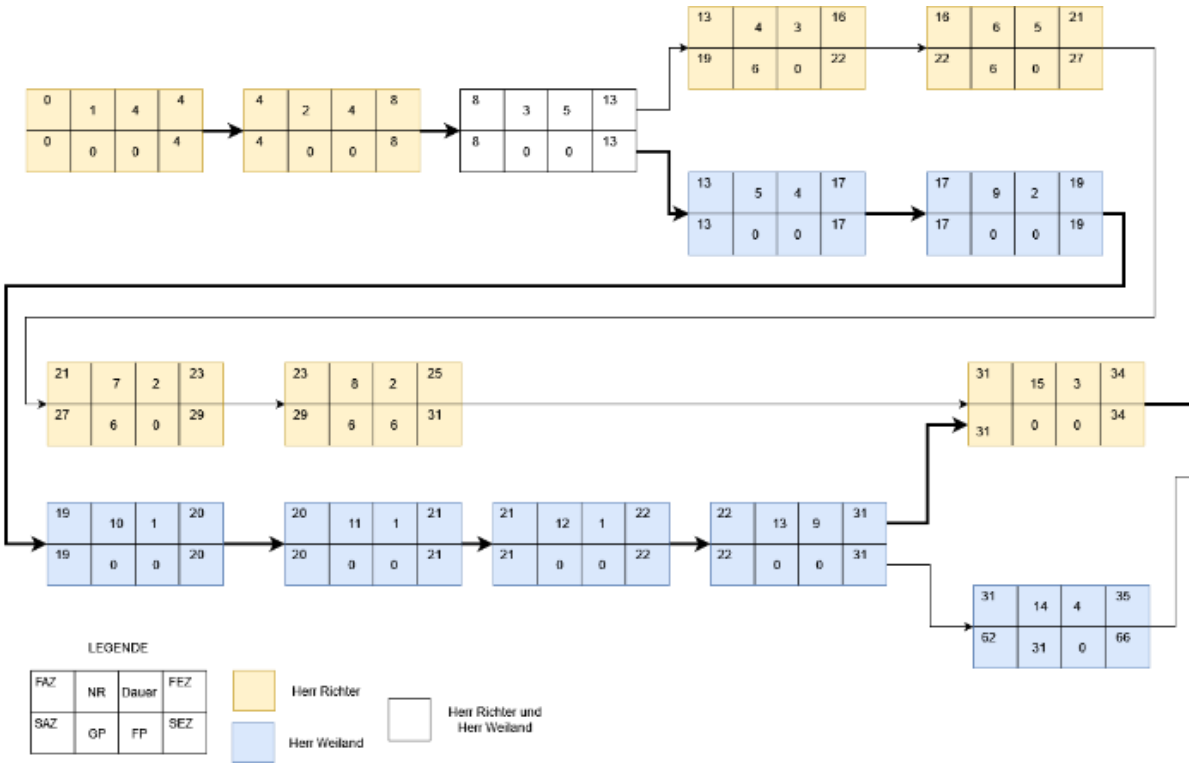

Abbildung 1: Netzplan Teil 1

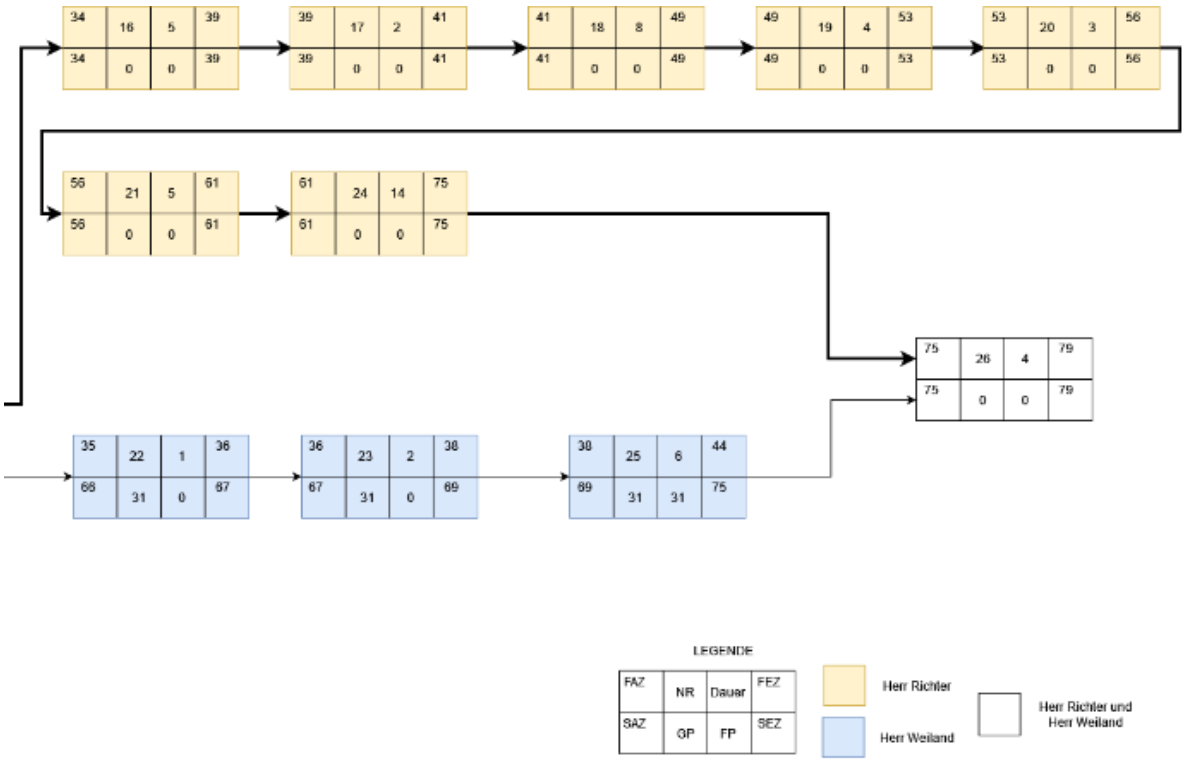

Abbildung 2: Netzplan Teil 2

A.2: Detaillierte Zeitplanung

Analysephase		**8 h**
1. Durchführung einer Ist-Analyse	2 h	
2. Durchführen der Amortisationsrechnung	2 h	
3. Nutzwertanalyse durchführen	2 h	
4. Unterstützen des Auftraggebers beim Erstellen des Lastenheftes	2 h	
Entwurfsphase		**13 h**
1. Teammeeting	3 h	
2. ER- Modell der Datenbank erstellen	3 h	
3. Erstellen der UML Diagramme	3 h	
4. Planung der Softwarearchitektur	2 h	
5. Qualitätssichernde Maßnahmen planen	2 h	
Entwicklungsphase		**32 h**
1. Einrichten der Entwicklungsumgebung inkl. Erstellen des Projektes und Installation und Konfiguration der benötigten Bibliotheken	2 h	
2. Implementieren der Oberfläche inkl. Tests	3 h	
3. Implementierung der Datenbankanbindung	4 h	
4. Testen der Datenbankanbindung	1 h	
5. Erstellung der Login Funktion	2 h	
6. OCR Funktion erstellen und testen	8 h	
7. Darstellung der Daten mit Sortierfunktion	4 h	
8. Implementierung der Volltextsuche	2 h	
9. Testen der Volltextsuche	1 h	
10. Abschließende Testphase der Anwendung	3 h	
11. Fehlerbehebung durchführen	2 h	
Abnahmephase		**3 h**
1. Abnahme durch den Auftraggeber	3 h	
Erstellen der Dokumentation		**14 h**
1. Erstellung der Projektdokumentation	10 h	
2. Erstellung der Entwicklerdokumentation	4 h	
Gesamt		**70 h**

Abbildung 3: Detaillierte Zeitplanung

A.3 Verwendete Ressourcen

Hardware

- Arbeitsplatz mit Thin-Client inkl. Zwei Bildschirmen
- Maus und Tastatur

Software

- Visual Studio Code – Entwicklungsumgebung
- Browser (Mozilla- Firefox, Google- Chrome, Microsoft Edge)
- Table Plus (Datenbank Manager)
- Microsoft Word und Exel (Dokumentation)
- Draw.io
- Windows 10 Pro

Personal

- Umschüler/ Auszubildender – Umsetzung des Projektes
- Auftraggeber – Festlegung der Anforderungen, Abstimmung der Oberfläche und Abnahme des Projektes
- Mitarbeiter: innen (Frau Hybsier, Herr Weiland) – Layout und Design, Serverarchitektur

A.4 Auszug: Lastenheft

Die im folgenden Auszug aus dem Lastenheft definierten Anforderungen beziehen sich auf die im Use-Case-Diagramm dargestellten Anwendungsfälle. Dieses Use-Case-Diagramm ist im Anhang A.5 Use-Case-Diagramm auf Seite C dargestellt.

Anforderungen

Es werden folgende Anforderungen an die Anwendung gestellt:

- Alle Anwendungsfälle sind nur nach erfolgreicher Anmeldung möglich, um die privaten Dokumente vor Missbrauch von Dritten zu schützen
- Als Benutzer muss ich neue Dokumente hinzufügen können, um meine Sammlung stetig erweitern zu können
- Als Benutzer muss ich vorhandene Dokumente löschen können, da ein Dokument fehlerhaft sein kann oder es besteht die Möglichkeit das es nicht mehr benötigt wird
- Als Benutzer muss ich vorhandene Dokumente und deren Daten bearbeiten können, damit Fehler korrigiert werden können
- Als Benutzer muss ich in der Dokumentenliste ein Dokument auswählen und anschließend ansehen können, um die Dokumente aufrufen zu können
- Als Benutzer muss ich die Dokumentenliste sortieren können, um effizient die Darstellung der Dokumente in der Liste zu ändern

[…]

A.5 Amortisationsdiagramm

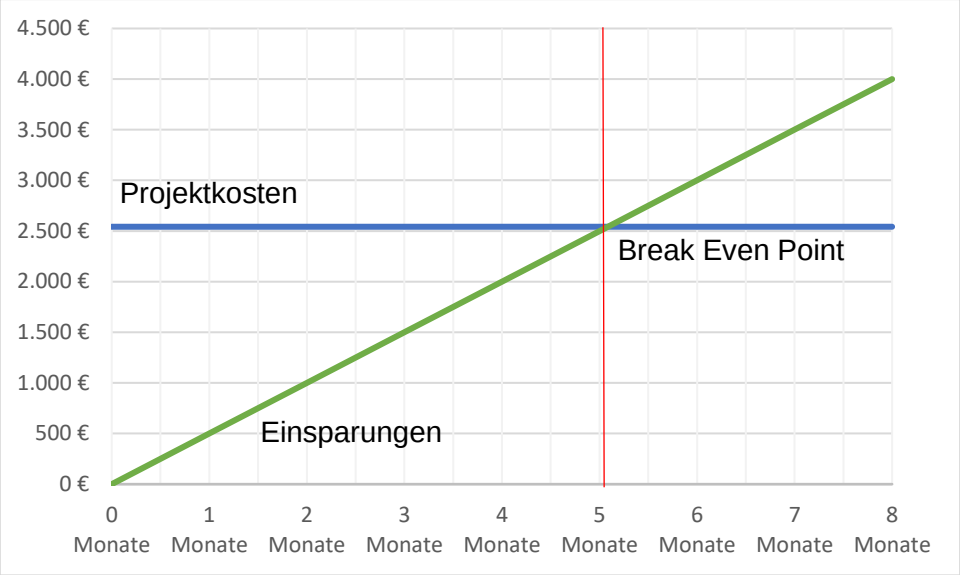

Abbildung 4: Amortisationsdiagramm

A.6 Use-Case-Diagramm

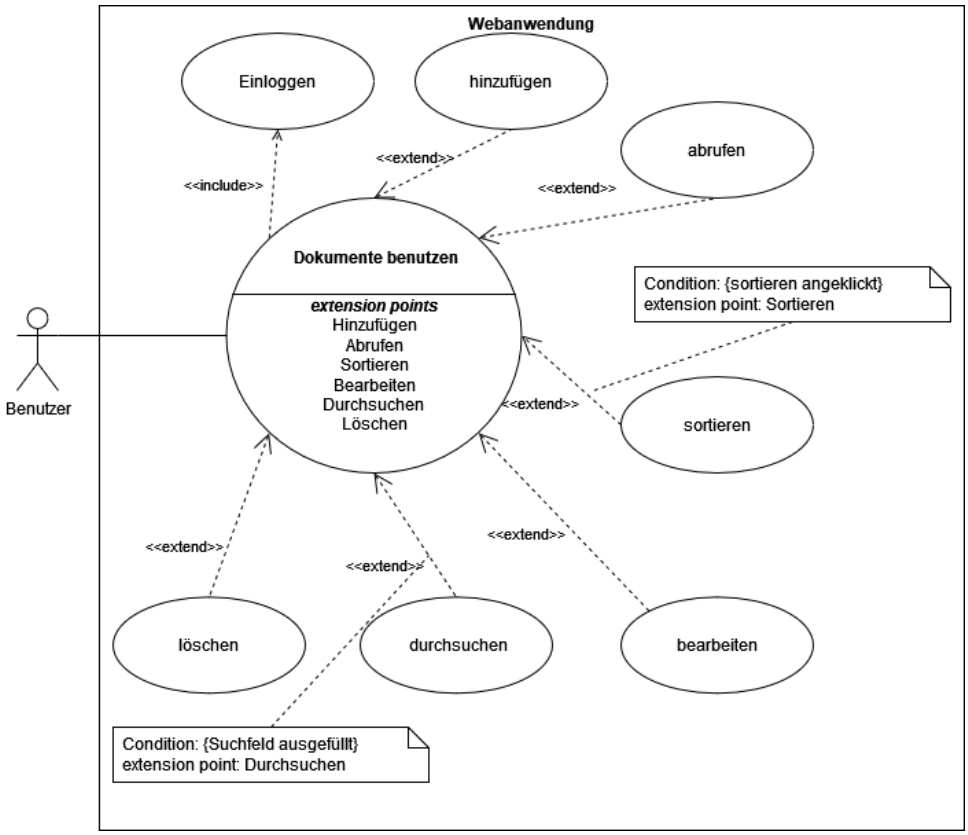

Abbildung 5: Use-Case-Diagramm

A:7 Nutzwertanalyse

Auswahlkriterien	Gewichtung	Laravel		Symfony	
		Bewertung	Teilnutzwert	Bewertung	Teilnutzwert
Angebotene Features	3	3	9	3	9
Verfügbare Lernmaterialien	4	3	12	3	9
Nötige Einarbeitungszeit	4	3	12	2	8
Dokumentation	2	2	4	3	6
Community	3	3	9	2	6

Erklärung zu den Auswahlkriterien:

1. Wie viele Features bietet das Framework und können sie im Projekt sinnvoll verwendet werden?
2. Lernmittel die zur Verfügung stehen, um sich in das Framework einzuarbeiten.
3. Wie lang dauert es sich in das Framework einzuarbeiten? Wie flach ist die Lernkurve?
4. Ist eine übersichtliche Dokumentation des Frameworks vorhanden und in welchen Sprachen ist diese vorhanden?
5. Wie bekannt ist das Framework? Wie viele benutzen es? Wie viel Hilfe kann man erwarten?

Bewertungsskala 1 (schlecht) bis 3 (gut):

1. Angebotene Features:

   Keine Features (1), Einige moderne Konzepte (2), Viele moderne Konzepte (3)

2. Verfügbare Lernmaterialien:

   Keine Lernmittel (1), Einige Lernmittel (2), Viele Lernmittel (3)

3. Nötige Einarbeitungszeit:

   Dauert sehr lang (1), dauert einige Zeit (2), schnelle Einarbeitungszeit (3)

4. Dokumentation:

   Kurze Doku in Englisch (1), ausführliche Doku in Englisch (2), Doku in Deutsch (3)

5. Community:

   Kleine Community (1), große Community (2), sehr große hilfsbereite Community (3)

Gewichtungsskala für die Kriterien reicht von 1 (weniger wichtig) bis 4 (sehr wichtig)

1. Wenn das Framework viele gute Features beinhaltet, dann verkürzt sich die Entwicklungszeit.
2. Das Framework sollte genügend verfügbare Lernmaterialen bereitstellen, da es sich um das Abschlussprojekt des Autors mit fester Zeitvorgame handelt.
3. Das Framework sollte eine flache Lernkurve haben, damit der Autor sich schnell in dieses Einarbeiten kann.
4. Eine gute Dokumentation erleichtert zwar das Lernen des Frameworks, aber es ist nicht zwingend notwendig.
5. eine große und gute Community ermöglicht dem Autor schnell eigene Fehler oder Probleme zu finden und zu lösen.

A8 Mockups

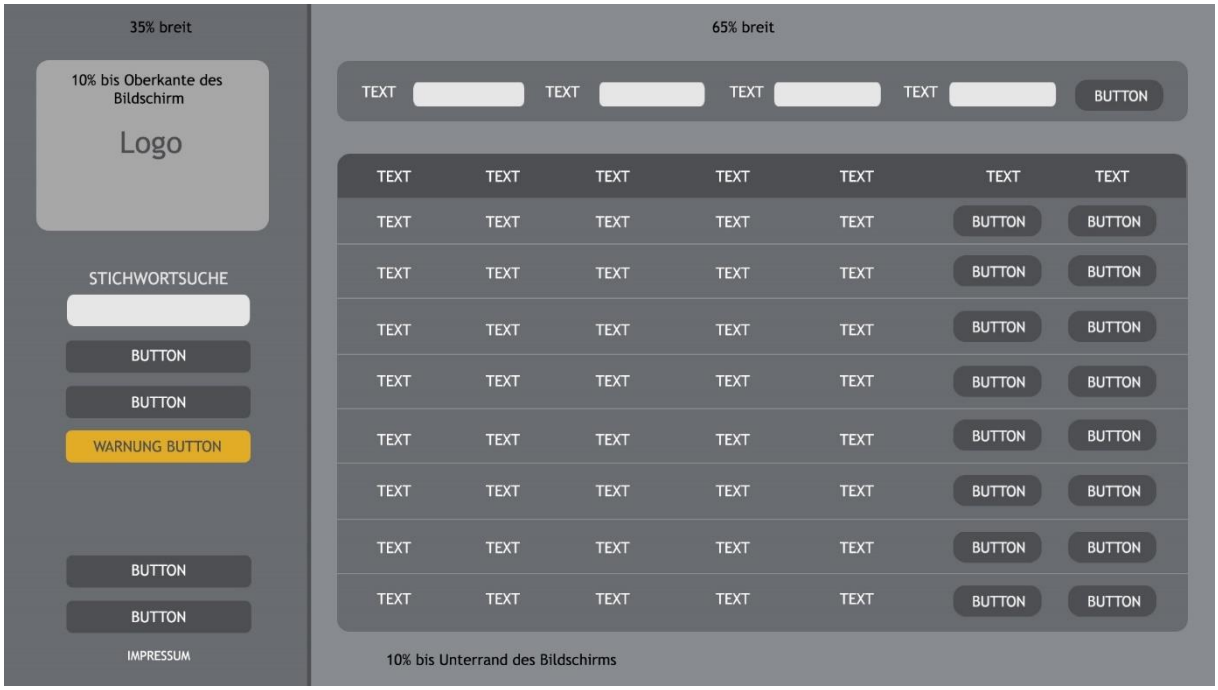

Abbildung 6: Mockup 1

Abbildung 7: Mockup 2

A9 Entity-Relationship-Modell

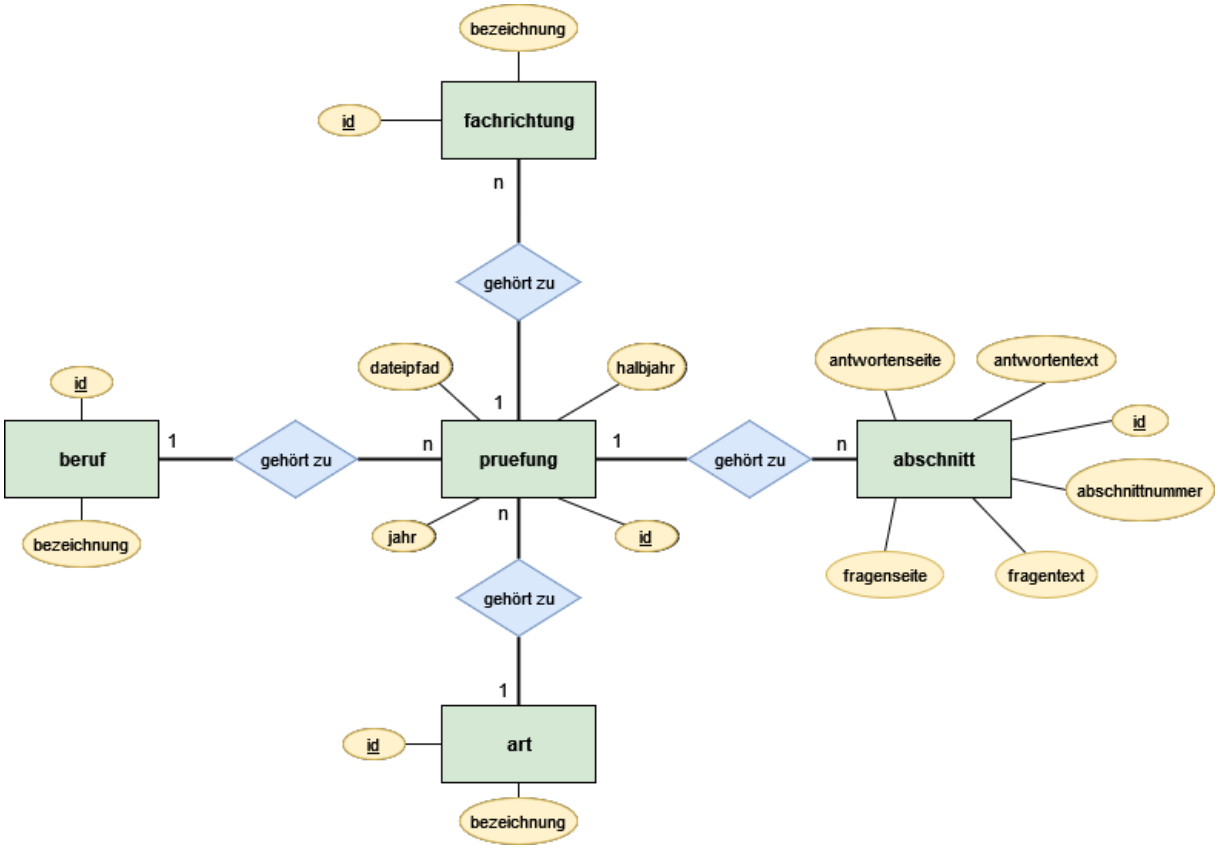

Abbildung 8: Entity-Relationship-Modell

A10 Screenshots der Anwendung

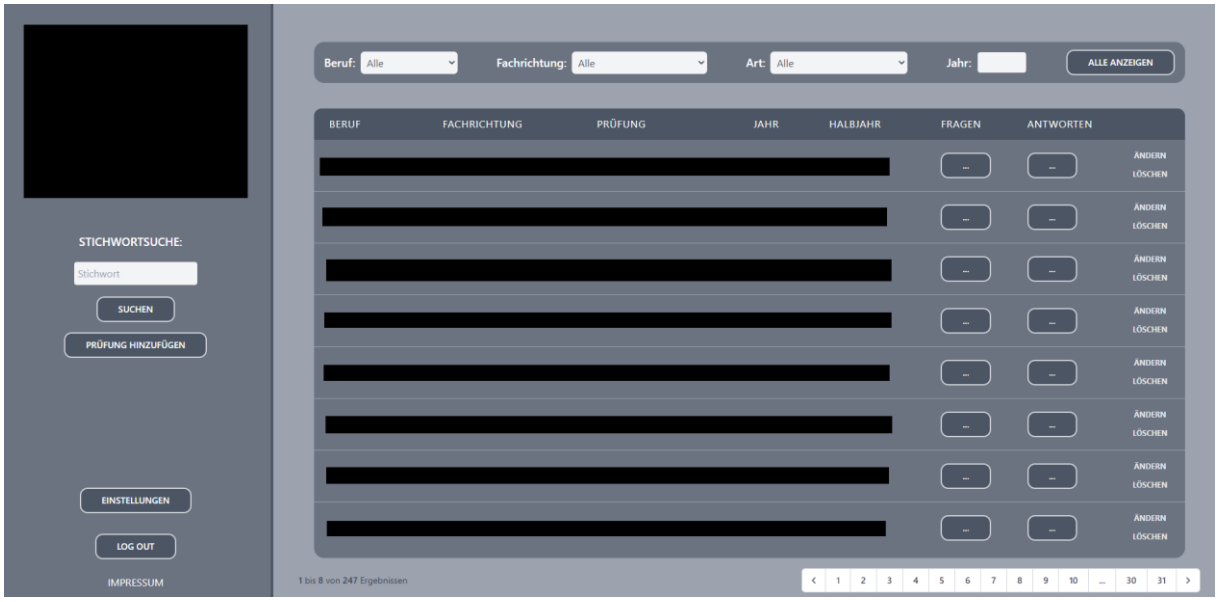

Abbildung 9: Screenshot der Anwendung

A11 Login Test Konsolenausgabe

```
[redacted]:/dockers/laravel-04-fixed$ docker-compose run artisan test
Creating laravel-04-fixed_artisan_run ... done

  PASS  Tests\Feature\Auth\LoginTest
  ✓ user can view a login form
  ✓ user cannot view a login form when authenticated
  ✓ user can login with correct credentials

  Tests:  3 passed
  Time:   0.18s
```

Abbildung 10: Login Test Konsolenausgabe

Projektbezogene Anlagen:

Projektauftrag:

Markus Liebenow
Lynarstraße 15, 13585 Berlin
Telefon: 030 / 984 585 33
Email: markus@liebenow.it
www.liebenow.it

Herr
Kai Richter und Sascha Weiland
im Haus

Projektauftrag: *Planung und Erstellung einer automatisierten und erweiterbaren Webanwendung zur Verwaltung von Prüfungsdokumenten mit integrierter Suchfunktion.*

Sehr geehrter Herr Weiland und Herr Richter,

wir bitten Sie für die Firma Liebenow IT eine Webanwendung zur Verwaltung von Prüfungsdokumenten (PDF) zu planen und zu erstellen. Die Anwendung soll bis zum 19.12.2021 online verfügbar sein.

Folgende Anforderungen bitten wir Sie in der Anwendung umzusetzen:

- Automatisches gliedern der PDF-Dateien (Optical Character Recognition) in Abschnitte und anschließendes speichern in einer Datenbank
- Die Möglichkeit zur manuellen Einfügung und Bearbeitung der Daten soll vorhanden sein
- Sie soll von überall erreichbar sein (Online)
- Es soll eine Volltextsuche innerhalb der Dokumente möglich sein
- Eine übersichtliche und strukturierte Sortierung der Prüfungen
- Die Darstellung soll innovativ und zeitgemäß sein
- Die zu verwendende Serverstruktur soll leichtgewichtig sein
- Der Zugriff auf die Anwendung soll über ein Benutzerlogin geschützt sein

B1 Migrations

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePruefungsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('pruefungs', function (Blueprint $table) {
            $table->id();
            $table->bigInteger('jahr');
            $table->foreignId('art_id')->constrained();
            $table->bigInteger('halbjahr');
            $table->string('dateipfad');
            $table->foreignId('beruf_id')->constrained();
            $table->foreignId('fachrichtung_id')->constrained();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('pruefungs');
    }
}
```

B2 Models

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Abschnitt extends Model
{
    use HasFactory;
    public function pruefung()
    {
        return $this->belongsTo(Pruefung::class);
    }

    protected $fillable = ['pruefung_id', 'abschnittnummer', 'fragentext',
'antwortentext', 'fragenseite', 'antwortenseite'];
}
```

B3 Factorys

```php
<?php

namespace Database\Factories;

use App\Models\Abschnitt;
use Illuminate\Database\Eloquent\Factories\Factory;

class AbschnittFactory extends Factory
{
    private static $p_id = 1;
    private static $counter = 1;
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = Abschnitt::class;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        self::$counter++;
        if (self::$counter % 6 === 0) {
            self::$p_id++;
            self::$counter = 1;
        }
        return [
            'pruefung_id' => self::$p_id,
            'abschnittnummer'=>self::$counter,
            'fragentext' => $this->faker->paragraph(1),
            'antwortentext' => $this->faker->paragraph(1),
            'fragenseite' => rand(1, 5),
            'antwortenseite' => rand(1, 5)
        ];
    }
}
```

B4 SessionController (Auszug)

```
                                        {...}
class SessionsController extends Controller
{
    public function create()
    {
        return view('auth.login');
    }

    public function authenticate(Request $request)
    {
        $attributes = request()->validate([
            'email' => 'required|email',
            'password' => [
                'required',
                'string',
                new isValidPassword(),
            ],
        ]);

        if (User::all()->isEmpty()) {
            User::create(
                ['email' => $attributes['email'],'password'=>$attributes['password']]
            );
        }

        if (Auth::attempt($attributes)) {
            session()->regenerate();

            redirect('/home')->with('status', 'Willkommen zurück!');
        }
        throw ValidationException::withMessages([
                'all' => 'Die Einlogdaten sind nicht korrekt.'
            ]);
    }
    public function destroy()
    {
        Auth::logout();

        return redirect('/login')->with('status', 'Bis bald!');
    }
                                        {...}
```

B5 ParsingJob (Auszug)

```
                                                {…}
class ParsingData implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
    const PARSERIP = "http://xxx.xxx.xxx.xxx:xxxx";
    /**
     * Create a new job instance.
     *
     * @return void
     */
    protected $pruefung;
    public function __construct(Pruefung $pruefung)
    {
        $this->pruefung = $pruefung;
    }

    /**
     * Execute the job.
     *
     * @return void
     */
    public function handle()
    {
        $fragenfile =  Storage::get($this->pruefung->dateipfad.'/fragen.pdf');
        $antwortenfile =  Storage::get($this->pruefung->dateipfad.'/antworten.pdf');

        $response = Http::attach('pdf', $fragenfile, 'fragen.pdf', ['type' =>
'application/pdf'])
            ->post(self::PARSERIP, ['lang' => 'de', 'fast' => '0']);
        $fragendocid = $response['id'];
        $response = Http::attach('pdf', $antwortenfile, 'fragen.pdf', ['type' =>
'application/pdf'])
            ->post(self::PARSERIP, ['lang' => 'de', 'fast' => '0']);
        $antwortendocid = $response['id'];

                                                {…}
```

Fortsetzung B5 ParsingJob (Auszug)

```
                                        {...}
do {
                sleep(10);
                $fragenresponse =
Http::get(self::PARSERIP.'/update/'.$fragendocid)->body();
                if (str_contains($fragenresponse, '"failed":true')) {
                    throw new Error($fragenresponse);
                }
            } while (str_contains($fragenresponse, '"position":') ||
str_contains($fragenresponse, '"running":true'));

            do {
                sleep(10);
                $antwortenresponse =
Http::get(self::PARSERIP.'/update/'.$antwortendocid)->body();
                if (str_contains($antwortenresponse, '"failed":true')) {
                    throw new Error($antwortenresponse);
                }
            } while (str_contains($antwortenresponse, '"position":') ||
str_contains($antwortenresponse, '"running":true'));

            $fragen = $this->getAbschnitte($this-
>umlautReplace($fragenresponse));
            $antworten = $this->getAbschnitte($this-
>umlautReplace($antwortenresponse));

             $startseitefragen = 2;
            $startseiteantworten = 2;

for ($i=0; $i <= sizeof($fragen) - 1 ; $i++) {

$abschnitt = Abschnitt::create([
                        'pruefung_id' => $this->pruefung->id,
                        'abschnittnummer' => $i+1,
                        'fragenseite' => $startseitefragen,
                        'antwortenseite' => $startseiteantworten,
                        'fragentext' => $fragentext,
                        'antwortentext' => $antwortentext,
                    ]);

                                        {...}
```

B6: Scope Filter

```
public function scopeFilter($query, array $filters)
{
    $query->when(
        $filters['stichwort'] ?? false,
        fn ($query, $search) =>
        $query->whereHas(
            'abschnitte',
            fn ($query) =>
            $query->where('fragentext', 'like', '%' . strtolower(preg_replace(
                array('/[^a-zA-Z0-9 -]/', '/[ -]+/', '/^-|-$/'),
                array('', '-', ''),
                $search
            )) . '%')
                ->orWhere('antwortentext', 'like', '%' . strtolower(preg_replace(
                    array('/[^a-zA-Z0-9 -]/', '/[ -]+/', '/^-|-$/'),
                    array('', '-', ''),
                    $search
                )) . '%')
        )
    );

    $query->when(
        $filters['beruf'] ?? false,
        fn ($query, $bezeichnung) =>
        $query->whereHas(
            'beruf',
            fn ($query) =>
            $query->where('bezeichnung', $bezeichnung)
        )
    );

                                                {…}
```

B7: Login Tests

```
                                    {...}
class LoginTest extends TestCase
{
    use RefreshDatabase;
    /**
     * A basic feature test example.
     *
     * @return void
     */

    public function test_user_can_view_a_login_form()
    {
        $response = $this->get('/');

        $response->assertSuccessful();
        $response->assertViewIs('auth.login');
    }
    public function test_user_cannot_view_a_login_form_when_authenticated()
    {
        $user = User::make();

        $response = $this->actingAs($user)->get('/login');

        $response->assertRedirect('/home');
    }
    public function test_user_can_login_with_correct_credentials()
    {
        $user = User::create([
            'email' => 'xxx@xxx.com',
            'password' => bcrypt($password = 'XXXXXXXXXXXX'),
        ]);
        $response = $this->post('/login', [
            'email' => $user->email,
            'password' => $password,
        ]);
        $this->actingAs($user);
        $response->assertRedirect('/');
        $this->assertAuthenticatedAs($user);
    }
}
                                    {...}
```