

Eigenarbeit OOP



Abbildung 1: Titelbild

Name der Schule:	TEKO Schweizerische Fachschule Bern
Verfasser:	- Truttmann Simon; Zollikofen; Informatik (Systemtechnik) 2022 - Gerber Livio; Bümpliz Süd; Informatik (Systemtechnik) 2022
«Eingereicht bei:»	Christian Herren (Dozent an der Teko)
Abgabedatum:	11.03.2024

1 Inhaltsverzeichnis

2	Dokumentation	3
2.1	Analyse nach rotem Faden	4
2.2	Anforderungen	8
2.3	Implementation	8
2.4	Testkonzept	9
3	Referenzen	13
3.1	Abbildungsverzeichnis	13
3.2	Tabellenverzeichnis	13
3.3	Literaturverzeichnis	13

2 Dokumentation

In dieser Dokumentation wird die Programmaufgabe im Fach [Schulfach] behandelt: **die Entwicklung eines elektronischen Tagebuchs**. Wir werden den gesamten Entwicklungsprozess durchgehen und dabei einen **roten Faden** verwenden, um eine klare Struktur zu gewährleisten und den Fortschritt zu verfolgen. Die genaue Umschreibung der Aufgabenstellung finden sie hier:

Elektronisches Tagebuch

Viele kennen es nur noch vom Hörensagen, das Tagebuch. Aber teilweise auch noch heute führen immer noch viele eine Erinnerungshilfe, in welcher sie das Erlebte Tag für Tag ablegen und es so bewahren.

Unsere Idee ist es nun eine aktuellere Version eines solchen Tagebuches zu erschaffen.

Bez. des Erfassens von Einträgen soll möglich sein:

- *Erstellen von Einträgen pro gewähltes Datum*
- *Erfassen von Freitext inkl. Sonderzeichen bis zu einer maximalen Länge von 1000 Zeichen*
- *Erfassen von 3 frei oder aus einer Domäne gewählten Identifikatoren (wie z.B. Ferien, Geburtstag, Familienfest...) für den Eintrag à «Tags»*
- *Optional: Erfassen eines Bildes pro Eintrag (als sep. Punkt der Oberfläche)*

Wir wollen bezüglich der Nutzung/Auswertung folgende Funktionen anbieten:

- *Sicherung des Tagebuches mit einem Login (mittels Username und Passwort)*
- *Ausgeben eines Eintrages via gewählttem Datum*
- *Ausgeben aller Beiträge zu einem bestimmten Identifikator (z.B. alle Einträge, welche zu Geburtstagen zugeordnet sind)*
- *Anzeigen zu welchen Tagen keine Einträge vorhanden sind*

Dazu wurden folgende Rahmenbedingungen platziert:

Die Gestaltung der Oberfläche ist neben der Funktionalität ein wichtiger Aspekt.

Es wird ein vollständiger Nachweis der Funktionalität in Form von geeigneten Tests erwartet. (Planung und Durchführung) --> Beweise mit Hilfe von Screenshots

Es müssen maximal 100 Einträge verwaltet werden können.

2.1 Analyse nach rotem Faden

Bei der Analyse geht es darum das geplante Programm: **Elektronisches Tagebuch** mit Hilfe von Diagrammen: Sequenzdiagramm, Kommunikationsdiagramm bis und mit Klassendiagramm aufzuzeigen, um am Schluss nur noch anhand diesen Diagrammen den Programm-Code zu erstellen.

Die Analyse wurde anhand folgender Übersicht gemacht:

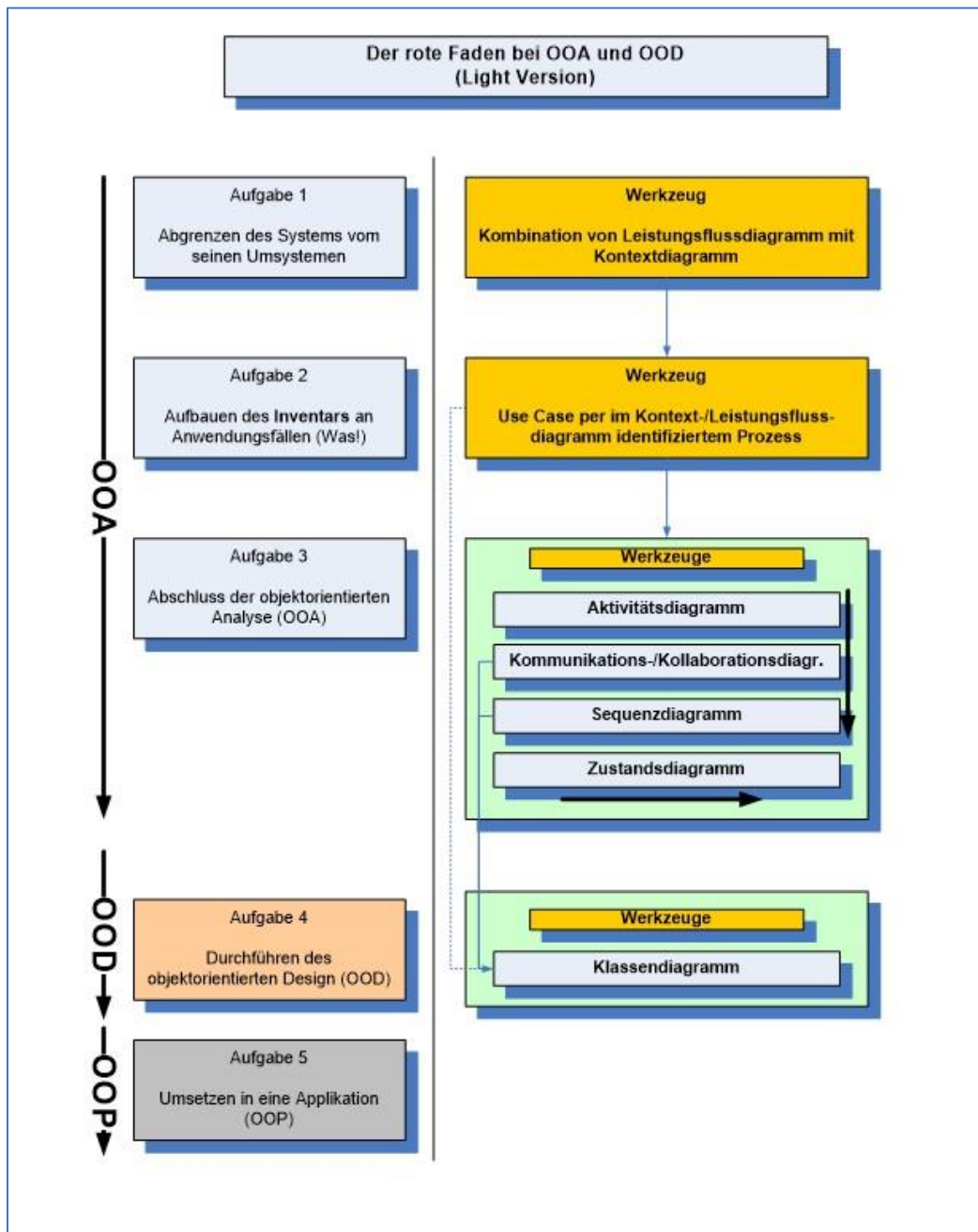


Abbildung 2: Ablauf-Diagramm

Wir verwenden also für unsere Analyse folgende Diagramme:

Anbei einen kurzen Satz zur Erklärung was das genannte Diagramm beinhaltet.

- **Kontext Diagramm:**

Ein Kontextdiagramm in einer Programmierplanung ist eine visuelle Darstellung der Interaktionen zwischen einem System und seinen externen Entitäten.

Benutzererfassung:

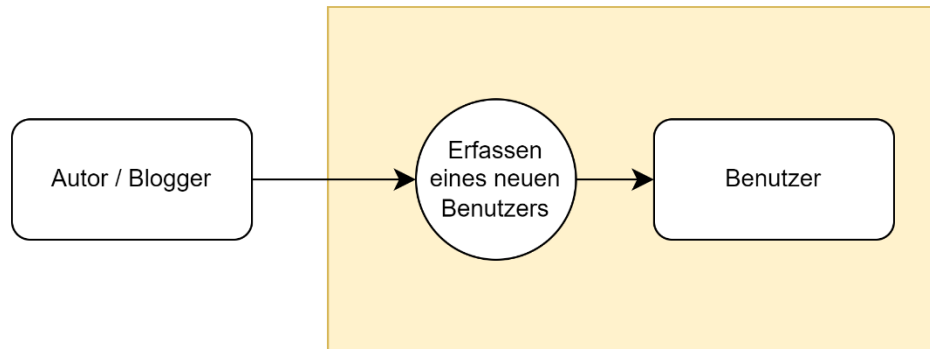


Abbildung 3: Kontextdiagramm_1

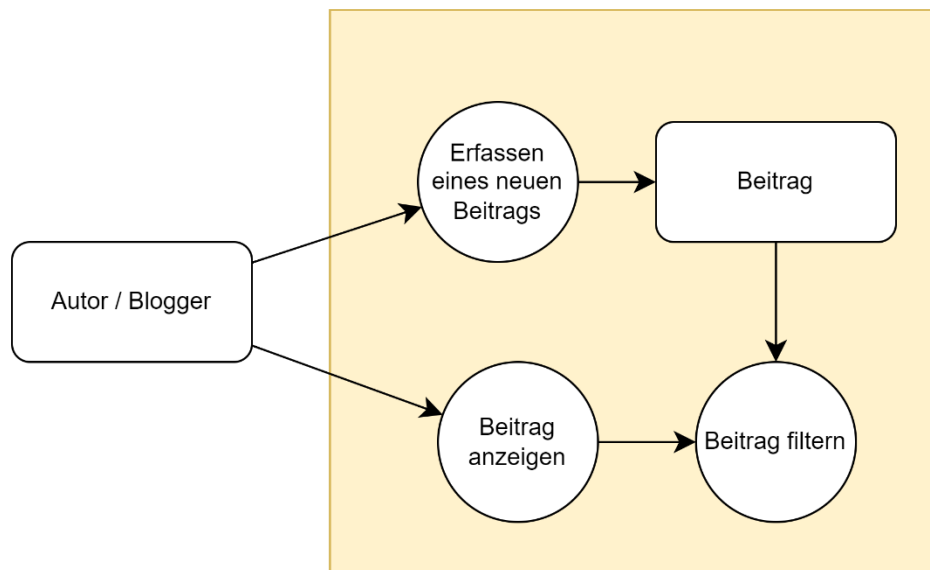


Abbildung 4: Kontextdiagramm_2

- **Use-Case Diagramm**

Ein Use Case Diagramm in einer Programmierplanung ist eine grafische Darstellung, die die Interaktionen zwischen den Akteuren und den verschiedenen Funktionen oder Anwendungsfällen eines Systems veranschaulicht.

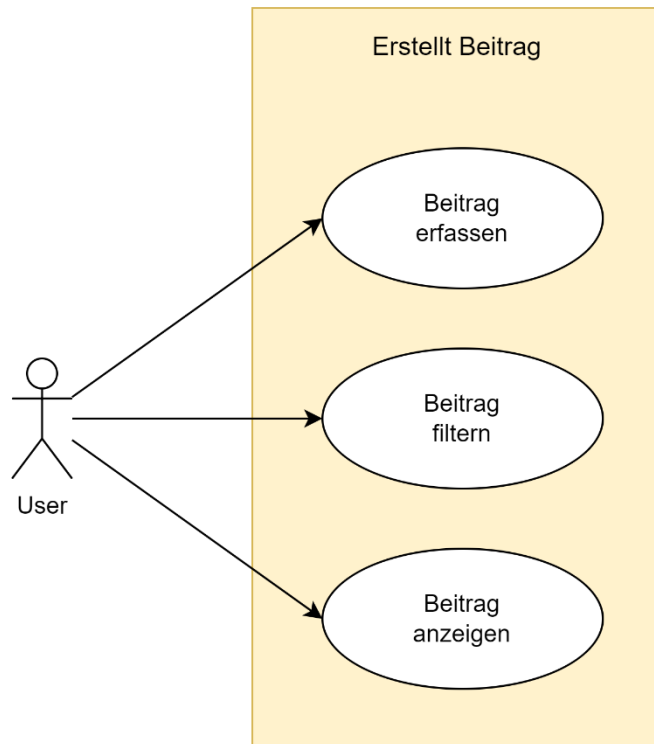


Abbildung 5: UseCase-Diagramm

- **Aktivität Diagramm**

Ein Aktivitätendiagramm in einer Programmierplanung ist eine grafische Darstellung, die den Ablauf von Aktivitäten und Prozessen innerhalb eines Systems veranschaulicht.

- **Kommunikation Diagramm**

Ein Kommunikationsdiagramm in einer Programmierplanung ist eine grafische Darstellung, die die Interaktionen und Nachrichtenaustausche zwischen verschiedenen Objekten oder Komponenten innerhalb eines Systems oder zwischen verschiedenen Systemen veranschaulicht.

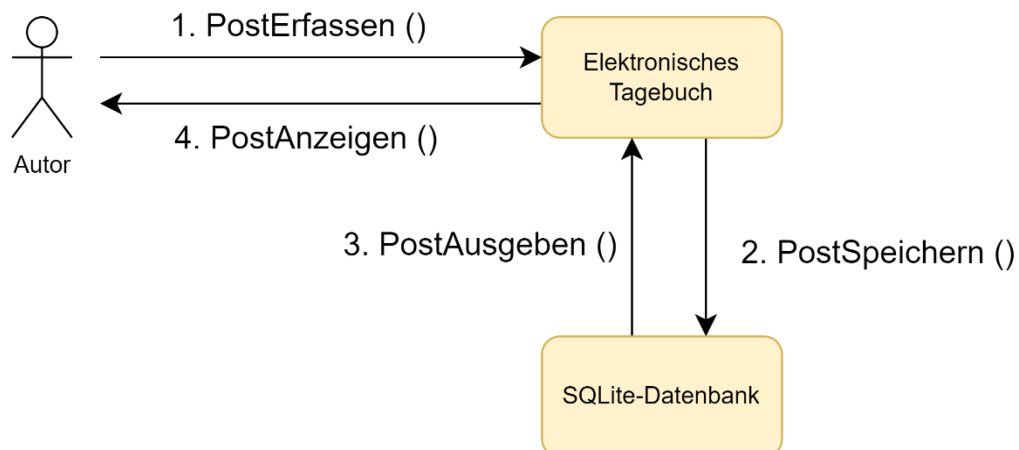


Abbildung 6: Kommunikations-Diagramm

- Sequenz Diagramm

Ein Sequenzdiagramm in einer Programmierplanung ist eine grafische Darstellung, die die zeitliche Abfolge von Interaktionen zwischen Objekten oder Komponenten in einem System veranschaulicht.

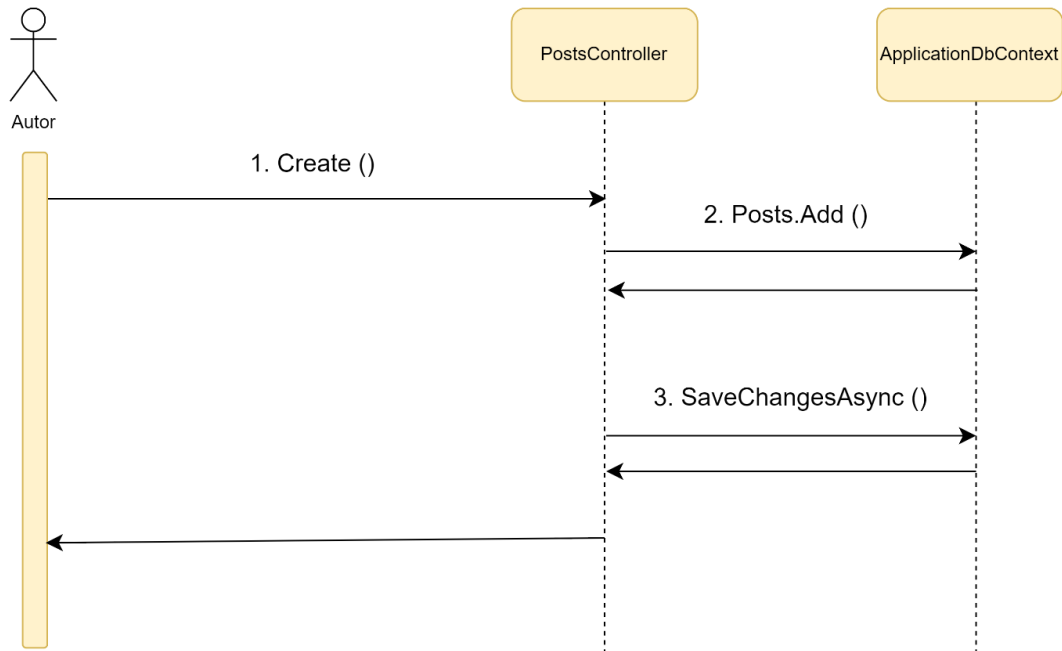


Abbildung 7: Sequenz-Diagramm

- Zustands Diagramm

Ein Zustandsdiagramm in einer Programmierplanung ist eine grafische Darstellung, die die verschiedenen Zustände eines Objekts oder Systems sowie die Übergänge zwischen diesen Zuständen veranschaulicht.

In unserem Fall wurde das Zustandsdiagramm ausgelassen, weil es bei uns nirgends einen konkreten Zustand definiert ist und wir diesen auch nicht benötigen. Eine Transaktion kann man mit einem Zustandsdiagramm zum Beispiel gut aufzeichnen, da es verschiedene Zustände gibt: «Bezahlt, Erhalten..» in unserem Fall gibt es lediglich den Zustand gespeichert. Es gibt keinen konkreten Zustand für das Erstellen von Beiträgen.

- Klassen Diagramm

Ein Klassendiagramm in einer Programmierplanung ist eine grafische Darstellung, die die Struktur eines Systems durch die Darstellung von Klassen und ihren Beziehungen zueinander beschreibt.

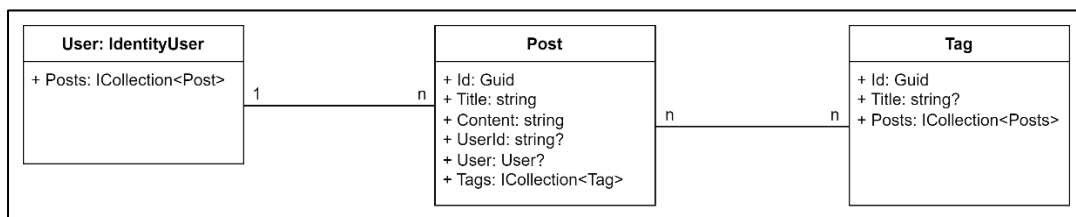


Abbildung 8: Klassendiagramm

2.2 Anforderungen

Technische-Anforderungs-ID	Anforderung	Beschreibung
TA-01	Erfassen von Einträgen	Ein Artikel soll erfasst werden können.
TA-01-2	Erfassen von Einträgen pro Datum	Ein Artikel kann bei der Erstellung mit einem Datum erfasst werden.
TA-02	Erfassen von Freitext inkl. Sonderzeichen bis zu 1000 Zeichen pro Artikel	Beim Erstellen von einem Artikel soll ein Freitext von bis zu 1000 Zeichen inklusive Sonderzeichen erfasst werden können.
TA-03	Erfassen von min. 3 Identifikatoren	Beim Erstellen von einem Artikel sollen 3 freie oder aus einer Domäne gewählte Identifikatoren «Tags» gesetzt werden.
TA-04	Sicherung	Das Tagebuch soll mittels Login (Username, Passwort) gesichert sein.
TA-05	Ausgabe per Datum	Die im Tagebuch gespeicherten Einträge können mittels Datum Filter ausgegeben werden.
TA-06	Ausgabe per Tag	Die im Tagebuch gespeicherten Einträge können mittels Tag Filter ausgegeben werden.
TA-07	Ausgabe ohne Tag	Alle Einträge, die keinen Tag besitzen können via Filter angezeigt werden.
TA-08	Benutzeroberfläche	Das Elektronische Tagebuch kann via GUI benutzt werden.

Tabelle 1: Anforderungsauflistung

2.3 Implementation

Da wir bei C# nicht nur den Projekt-Code programmieren wollten, sondern auch was Neues lernen wollten, haben wir uns für ASP.NET Core entschieden. Wir implementierten mit diesem Framework unser Projekt. Leider gab es dabei einige Tücken, mit denen wir zu Beginn nicht ganz klar kamen. Eine besonders mühsame ist die Dokumentation von ASP.NET Code selbst. Mit Hilfe von Chat GPT konnten wir dann aber auch dieses File und Erklärung Wirrwarr lösen.

Schlussendlich haben wir mit dem Generator ein neues ASP.NET Core Projekt erstellt, dass auch das Entity- und Identity Framework verwendet. Im Anschluss wurde das elektronische Tagebuch designt und aufgebaut. Um Daten zu speichern, erstellten wir mit dem Entity-Framework eine SQLite Datenbank und interagierten mit dieser dank Modellen. Es wurden jeweils ein Model für **User**, **Post** und **Tag** erstellt.

Ein weiterer Knackpunkt hatten wir beim Post Controller also quasi beim Device, dass für das CREATE, READ und UPDATE vom Tag zuständig ist. Wir verstanden zuerst nicht wie genau das Routing der http Request vom ASP.NET Core funktioniert. Nach mehrfachen Fehlversuchen, Testen und Weinen funktionierte es nun endlich. Leider gabs danach Probleme mit der GRUD Implementation des Posts. Da wir die Validation der Pages nicht verstanden haben.

Nach der Ganzen Implementation hat es leider nicht gereicht die «Filter Funktionen» zu implementieren. Wir haben alle Settings bis auf diese geplant, programmiert und getestet.

Der Code befindet sich auf GitHub in einem Public Repository. [GitHub Repository](#)

2.4 Testkonzept

Testmittel

Auflistung der verwendeten Testmittel, um die Tests durchzuführen. Die Projekttests werden mit Hilfe von **Black Box** Tests durchgeführt.

Objekt-ID	Testmittel	Beschreibung
TM-01	Notebook	System 76, PopOS (Pang 11) Linux Client (Pangolin)
TM-02	Browser	Firefox (Version 123.0)
TM-03	Terminal	Linux Standard Terminal

Tabelle 2: Testmittel

Testziele

- Nachweis des Funktion Standes (Funktioniert, Funktioniert (Mit Fehlern), Funktioniert nicht)
- Fehler aufdecken
- Noch nicht korrekt funktionierende Komponenten testen, um diese später zu überarbeiten.

Testobjekt

Nach Start des Codes, wird die Webseite getestet, die vom System geöffnet wird.

- Webseite: <http://localhost:5059>

Testvoraussetzungen

- Alle Testmittel sind vorhanden.
- Programm läuft und ist bereit zum Testen.

Fehlerklassen

Fehlerklassen-ID	Fehlerklasse	Beschreibung
FK-01	Fehlerfrei	Der Test wurde erfolgreich abgeschlossen
FK-02	Unwesentlicher Mangel	Es gibt sehr kleine Abweichungen zwischen dem erwarteten Ergebnis und dem Resultat des Tests. Massnahmen bedingt nötig
FK-03	Kleiner Mangel	Massnahmen sind von Bedarf
FK-04	Schwerer Mangel	Massnahmen sind nötig
FK-05	Kritischer Mangel	Die Ziele wurden gar nicht erfüllt. Massnahmen sind notwendig!

Tabelle 3: Fehlerklassen

Technische-Anforderungs-ID	Anforderung	Beschreibung
TA-01	Erfassen von Einträgen	Ein Artikel soll erfasst werden können.
TA-01-2	Erfassen von Einträgen pro Datum	Ein Artikel kann bei der Erstellung mit einem Datum erfasst werden.
TA-02	Erfassen von Freitext inkl. Sonderzeichen bis zu 1000 Zeichen pro Artikel	Beim Erstellen von einem Artikel soll ein Freitext von bis zu 1000 Zeichen inklusive Sonderzeichen erfasst werden können.
TA-03	Erfassen von min. 3 Identifikatoren	Beim Erstellen von einem Artikel sollen 3 freie oder aus einer Domäne gewählte Identifikatoren «Tags» gesetzt werden.
TA-04	Sicherung	Das Tagebuch soll mittels Login (Username, Passwort) gesichert sein.
TA-05	Ausgabe per Datum	Die im Tagebuch gespeicherten Einträge können mittels Datum Filter ausgegeben werden.
TA-06	Ausgabe per Tag	Die im Tagebuch gespeicherten Einträge können mittels Tag Filter ausgegeben werden.
TA-07	Ausgabe ohne Tag	Alle Einträge, die keinen Tag besitzen können via Filter angezeigt werden.
TA-08	Benutzeroberfläche	Das Elektronische Tagebuch kann via GUI benutzt werden.

Abbildung 9: Technische Anforderungen

Testfall 1


Test-ID	Betroffene Anforderung	Testmethode
TF-01	TA-01, TA-01-2, TA-02, TA-03	Blackbox
Beschreibung: Wird bei der Erfassung eines Artikels, Freitext (1000 Zeichen), Datum, 3 Tags angegeben und der Artikel wurde korrekt erstellt, gilt dieser Test als bestanden.		
Testschritte: <ol style="list-style-type: none"> 1. Webseit öffnen: http://localhost:5059 2. Anmelden mit Username und Password → Log In (Blauer Button) 3. Artikel erstellen → Klick auf Create New 4. Titel setzen 5. Freitext eingeben von 1000 Zeichen 6. 3 Tags mit Komma getrennt bei Tags hineinfügen: Network, Linux, Test 7. Auf Create klicken. 8. Auf Back to List klicken. 		
Erwartetes Resultat: Post wird angezeigt.		
Ergebnis: <div> Potaso  </div>		

Tabelle 4: Testfall1

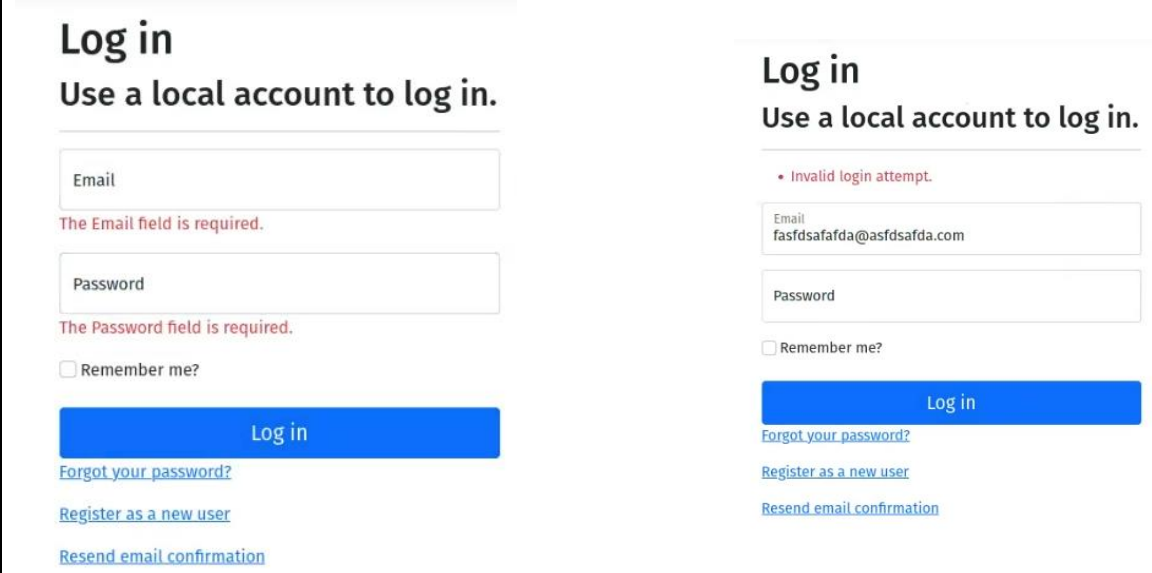
Test-ID	Betroffene Anforderung	Testmethode
TF-02	TA-04	Blackbox
Beschreibung: Das Elektronische Tagebuch ist mit einem Login geschützt, bei dem man Passwort und Benutzername eingeben muss.		
Testschritte: <ol style="list-style-type: none"> 1. Webseit öffnen: http://localhost:5059 2. Email leer lassen 3. Password leer lassen 4. Enter klicken <p>Nun testen wir noch mit der Eingabe von einem falschen Passwort:</p> <ol style="list-style-type: none"> 1. Eingabe falsches Passwort 2. Enter klicken 		
Erwartetes Resultat: <ol style="list-style-type: none"> 1. Rote Schrift erscheint: «The Email field is required» 2. Rote Schrift erscheint: «Invalid Login Attempt» 		
Ergebnis: 		

Abbildung 10: Testfall_2

Die Anforderungen **TA-05** bis **TA-08** konnten noch nicht umgesetzt werden!

Grund dafür ist, dass wir das Framework noch nicht genug gut kennen und wir nicht alles Implementieren konnten. Deshalb befinden wir uns im Zeitplan etwas zurück.

3 Referenzen

In diesem Kapitel werden alle Verzeichnisse aufgelistet.

3.1 Abbildungsverzeichnis

Abbildung 1: Titelbild	1
Abbildung 2: Ablauf-Diagramm	4
Abbildung 3: Kontextdiagramm_1	5
Abbildung 4: Kontextdiagramm_2	5
Abbildung 5: UseCase-Diagramm	6
Abbildung 6: Kommunikations-Diagramm	6
Abbildung 7: Sequenz-Diagramm	7
Abbildung 8: Klassendiagramm	7
Abbildung 9: Technische Anforderungen	10
Abbildung 10: Testfall_2	12

3.2 Tabellenverzeichnis

Tabelle 1: Anforderungsauflistung	8
Tabelle 2: Testmittel	9
Tabelle 3: Fehlerklassen	9
Tabelle 4: Testfall1	11

3.3 Literaturverzeichnis

App Diagramms. (kein Datum). Von <https://app.diagrams.net/> abgerufen

ChatGPT. (kein Datum). Von <https://chat.openai.com/> abgerufen

GitHub. (kein Datum). Von https://github.com/ohchikadoni/programming_herren.git abgerufen