Webanwendung zur Veröffentlichung von Instagram-Beiträgen

vorgelegt am 18. Januar 2025

Fakultät Wirtschaft und Gesundheit

Studiengang Wirtschaftsinformatik

Kurs WWI2022F

von

Leonard Eckert, David Kreismann, Tobias Schnarr und Nico Wagner

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

UML Unified Modeling Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Zielsetzung Das Ziel dieses Projekts ist die Entwicklung einer Webanwendung, die das tägliche Posten von Beiträgen auf Instagram automatisiert. Dabei sollen drei verschiedene Beitragstypen unterstützt werden:

- Einzelbild-Posts
- Video-Posts
- Textbild-Posts

Jeder Beitrag soll mit passenden Hashtags versehen werden, um die Sichtbarkeit in sozialen Netzwerken zu erhöhen. Die Automatisierung soll über eine Weboberfläche gesteuert werden, sodass Nutzer die Beiträge zentral verwalten und planen können.

Aufbau der Arbeit Die Arbeit gliedert sich in die folgenden Teile. Kapitel 1 beschreibt die Zielsetzung des Projekts und legt den Grundstein für diese Arbeit. In Kapitel 2 wird die Konzeption erläutert, die das Fundament für die Entwicklung der Webanwendung bildet. Kapitel 3 zeigt die Arbeitspakete auf und gibt eine Übersicht über die Beiträge der einzelnen Gruppenmitglieder. In Kapitel 4 wird die Umsetzung detailliert beschrieben und eine Anleitung zum Starten der Anwendung gegeben.

2 Konzeption

Um die Zielsetzung zu erreichen, sind verschiedene Funktionalitäten erforderlich. Diese werden als Anforderungen definiert und in drei Kategorien unterteilt: **MUSS**, **SOLL** und **KANN**. Dadurch wird eine klare Priorisierung der Anforderungen ermöglicht. Zur Planung kommen Jira und verschiedene Diagrammtypen der Unified Modeling Language (UML) zum Einsatz. Die Anforderungen werden in Jira als Stories erfasst und einer Kategorie zugeordnet. Im Laufe der Entwicklung werden die Stories den jeweiligen Sprints zugewiesen. Diese Zuweisung erfolgt im wöchentlichen Rhythmus, um flexibel auf unerwartete Herausforderungen oder Hindernisse reagieren zu können.

Anforderung		
MUSS	**SOLL**	**KANN**
Bild/Video hochladen	Instagram-Login	Berechtigungsverwaltung
Hinzufügen von Hashtags	Planung der Veröffentlichung	Automatische Anpassung von Medien
Hinzufügen von Text	Responsive Design	Erinnerung an ausstehende Beiträge

Tab. 1: Kategoriesierung der Anforderungen

Tabelle 1 zeigt wie die Anforderungen in Kategorien unterteilt werden. Es handelt sich hierbei um besonders relevante Anforderungen aus allen drei Kategorien. Eine vollständige Liste der Anforderung kann in Jira eingesehen werden.[1]

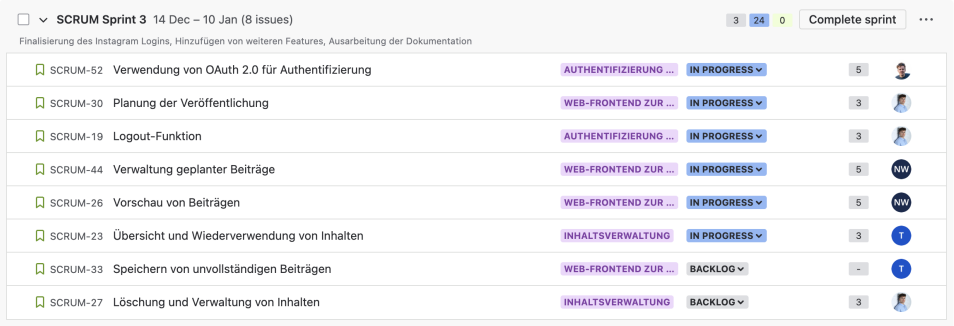

Abb. 1: Sprint 3

In Abbildung 1 ist eine Übersicht von Sprint 3 zu sehen. Die Anzahl der Stories pro Sprint variiert je nach Komplexität und Umfang. Die Entscheidung wie viele Stories in einen Sprint aufgenommen werden, wird im Team getroffen. Dabei wird darauf geachtet, dass die Stories in einem Sprint realistisch umsetzbar und die Storypoints von Sprint zu Sprint annähernd gleich sind. Wird eine Storie in einem Sprint nicht fertiggestellt, wird sie in den nächsten Sprint übernommen.

[1] Vgl. `https://engineeringproject.atlassian.net/jira/software/projects/SCRUM/boards/1`

Die Definition der Anforderungen und die Planung der Sprints in Jira sind wesentliche Bestandteile des Projekts. Sie gewährleisten einen klaren Überblick über die Anforderungen und den Fortschritt während des gesamten Entwicklungsprozesses. Im Folgenden wird die technische Konzeption dargelegt.

2.1 Technische Konzeption

Im Rahmen der Entwicklung der Webanwendung wurden verschiedene Frameworks und Tools eingesetzt, um eine effiziente, flexible und skalierbare Lösung zu gewährleisten. Diese werden nachfolgend erläutert.

Next.js[2] wurde als zentrales Framework für die Entwicklung der Webanwendung verwendet. Es bietet leistungsstarke Funktionen wie Server-Side Rendering (SSR) und eine hohe Performance, wodurch die Ladezeiten optimiert werden. Zudem ermöglicht die Flexibilität von Next.js eine reibungslose Integration weiterer Technologien.

Für die Automatisierung der Instagram-Beiträge wird die offizielle Application Programming Interface (API) von **Meta**[3] genutzt. Diese erlaubt den Zugriff auf Beitragsdaten sowie die Planung von Posts, was die zentrale Funktion der Anwendung unterstützt.

Zur Implementierung einer modernen und modularen Benutzeroberfläche wurde **shadcn/ui**[4] eingesetzt. Dieses UI-Framework zeichnet sich durch modulare Komponenten aus, die einfach erweiterbar sind. Zudem profitiert das Projekt von einer aktiven Community-Driven-Entwicklung, die regelmäßige Updates und Verbesserungen ermöglicht.

Für das Styling der Anwendung wurde **Tailwind CSS**[5] verwendet. Der Utility-First-Ansatz erlaubt eine schnelle und konsistente Gestaltung der Benutzeroberfläche. Dank Responsive Design passt sich die Anwendung an verschiedene Bildschirmgrößen an. Zudem steigert Tailwind die Produktivität, da wiederverwendbare Klassen eine effiziente Entwicklung ermöglichen.

Zur sicheren und skalierbaren Verwaltung von Medieninhalten wurde **uploadthing** integriert. Dieses Tool bietet eine hohe Sicherheit für den Datei-Upload, eine nahtlose Next.js-Integration und eine gute Skalierbarkeit, um den wachsenden Anforderungen der Anwendung gerecht zu werden.

Diese beschriebenen Technologien bilden das Fundament der Webanwendung. Im nächsten Abschnitt wird die verwendete Architektur näher betrachtet.

[2] Vgl. Vercel 2025
[3] Vgl. Meta Platforms, Inc. 2025
[4] Vgl. shadcn 2025
[5] Vgl. Tailwind Labs 2025

2.2 Architektur

Um eine durchdachte und strukturierte Architektur zu gewährleisten, wurden zunächst verschiedene UML-Diagramme erstellt. Diese dienen als Grundlage für die Implementierung und ermöglichen eine klare Strukturierung, aus der die einzelnen Komponenten der Anwendung abgeleitet werden können. Durch die Zusammenführung dieser Komponenten entsteht die Gesamtarchitektur der Anwendung.

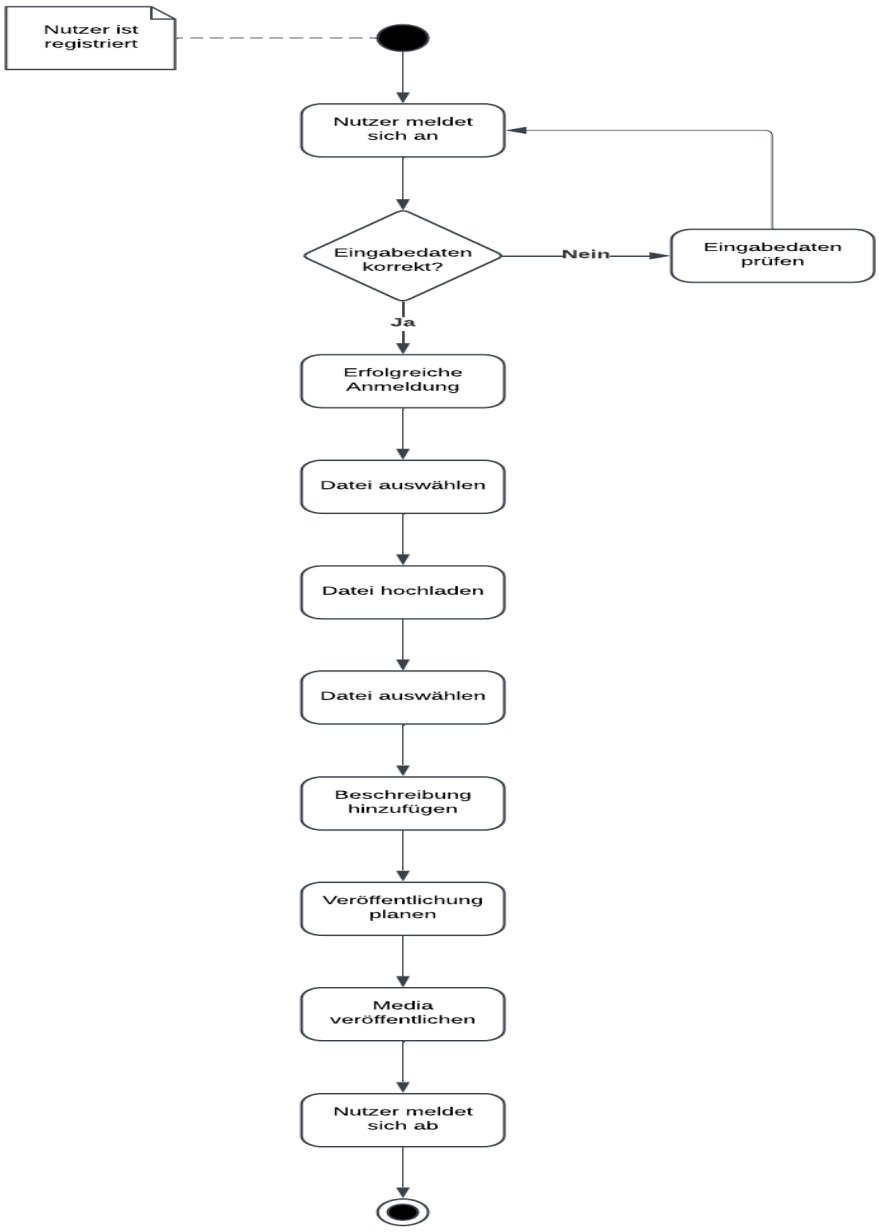

Abb. 2: Aktivitätsdiagramm[6]

[6]Notation nach: `https://www.lucidchart.com/pages/uml-activity-diagram`

In Abbildung 2 ist das **Aktivitätsdiagramm** dargestellt, das den typischen Ablauf der Benutzerinteraktion mit der Webanwendung zur automatisierten Veröffentlichung von Instagram-Beiträgen veranschaulicht. Es zeigt die einzelnen Schritte von der Anmeldung bis zur Veröffentlichung eines Beitrags und der anschließenden Abmeldung des Nutzers. Der Prozess beginnt mit einem bereits registrierten Nutzer, der sich in die Anwendung einloggt. Anschließend erfolgt eine Überprüfung der Eingabedaten. Falls diese nicht korrekt sind, wird der Nutzer zur erneuten Eingabe aufgefordert. Andernfalls erfolgt die erfolgreiche Anmeldung, und der Nutzer gelangt zur Hauptfunktionalität der Anwendung. Nach der Anmeldung kann der Nutzer eine Datei auswählen und diese anschließend hochladen. Dieser Schritt kann sich wiederholen, falls der Nutzer mehrere Dateien hochladen möchte. Anschließend wird eine Beschreibung hinzugefügt, um den Beitrag zu vervollständigen. Danach erfolgt die Planung der Veröffentlichung, bei der der Nutzer den gewünschten Veröffentlichungszeitpunkt festlegt. Sobald die Planung abgeschlossen ist, wird die Medienveröffentlichung durchgeführt. Abschließend meldet sich der Nutzer von der Anwendung ab, womit der Ablauf endet. Das Diagramm bietet eine strukturierte Darstellung der Benutzerinteraktion und stellt alle relevanten Schritte für eine erfolgreiche Beitragserstellung dar.

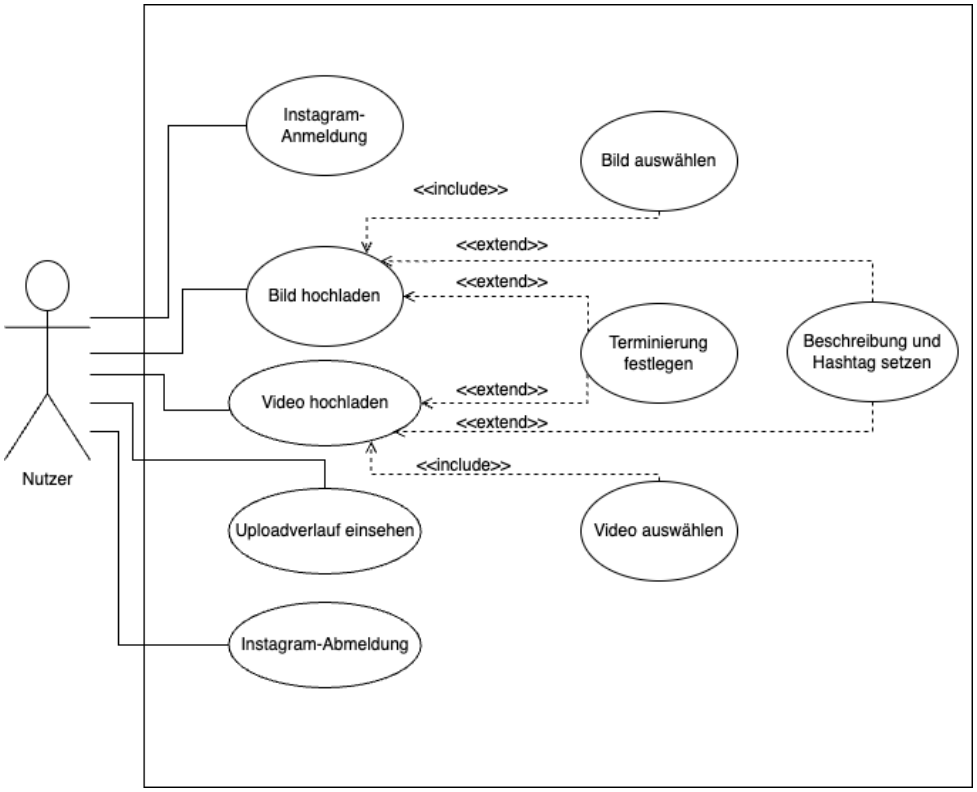

Abb. 3: Use Case Diagramm[7]

Abbildung 3 stellt das **Use Case Diagramm** dar, das die wesentlichen Use Cases der Webanwendung beschreibt. Es zeigt die Interaktionen des Nutzers mit dem System und verdeutlicht die Beziehungen zwischen den einzelnen Anwendungsfällen. Der Nutzer kann sich zunächst über die Instagram-Anmeldung in die Anwendung einloggen.

[7]Notation nach: `https://www.lucidchart.com/pages/uml-use-case-diagram`

Nach erfolgreicher Anmeldung stehen ihm mehrere Hauptfunktionen zur Verfügung:

- Bild hochladen
- Video hochladen
- Uploadverlauf einsehen
- Instagram-Abmeldung

Die beiden Anwendungsfälle Bild hochladen und Video hochladen beinhalten jeweils optionale Erweiterungen (*«extend»*):

- Terminierung festlegen: Der Nutzer kann den Zeitpunkt der Veröffentlichung bestimmen.
- Beschreibung und Hashtags setzen: Zusätzlich kann er eine Bildbeschreibung und relevante Hashtags hinzufügen.

Zudem gibt es *«include»*-Beziehungen, die darauf hinweisen, dass bestimmte Aktionen zwingend erforderlich sind:

- Bild hochladen und Video hochladen beinhalten jeweils das Auswählen einer Datei (Bild oder Video), bevor der Upload erfolgen kann.
- Uploadverlauf einsehen ist ein separater Use Case, der dem Nutzer ermöglicht, eine Übersicht über bereits hochgeladene Inhalte zu erhalten.

Das Diagramm visualisiert die Modularität und Erweiterbarkeit der Anwendung, indem es optionale (*«extend»*) und zwingend notwendige (*«include»*) Abhängigkeiten zwischen den Use Cases darstellt. Dadurch wird eine strukturierte und skalierbare Architektur unterstützt.

In Abbildung 4 ist das Sequenzdiagramm dargestellt, das den Ablauf für das Hochladen und Veröffentlichen eines Bildes in der Webanwendung beschreibt. Es zeigt die Interaktionen zwischen dem User, dem Webfrontend, dem Webbackend, dem Uploadthing-Service und der Instagram-API. Im Folgeden wird der Ablauf der Interaktion beschrieben:

1. **Nutzerinteraktion im Frontend**
    - Der Nutzer lädt ein Bild hoch, indem er eine Vorschau anfordert.
    - Das Webfrontend zeigt dem Nutzer eine Bildvorschau an, aus der er ein Bild auswählen und markieren kann.
    - Nach Auswahl klickt der Nutzer auf den Hochladen-Button, um den Upload-Vorgang zu starten.

[8]Notation nach: `https://www.lucidchart.com/pages/uml-sequence-diagram`

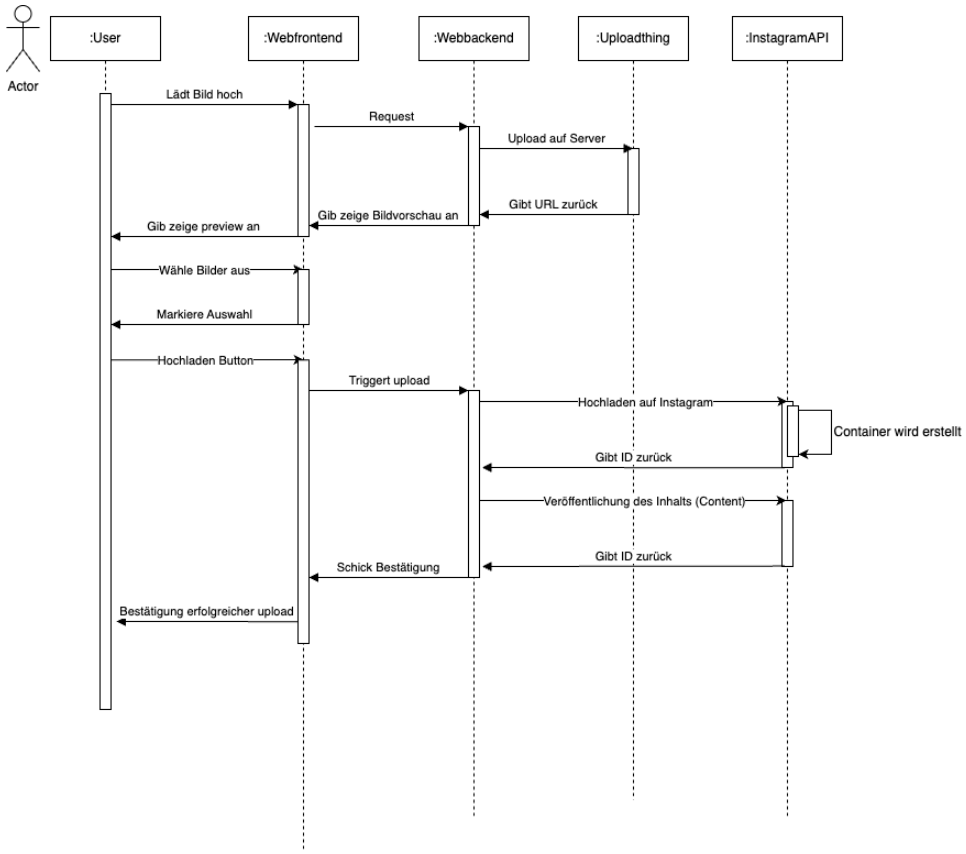

Abb. 4: Sequenzdiagramm[8]

2. **Kommunikation zwischen Frontend und Backend**

   - Das Webfrontend sendet eine Request-Nachricht an das Webbackend, um den Upload-Vorgang auszulösen.

3. **Upload auf Uploadthing**

   - Das Webbackend kommuniziert mit dem Uploadthing-Service, um die Datei auf den Server hochzuladen.
   - Nach erfolgreichem Upload gibt Uploadthing eine URL zurück, die die hochgeladene Datei referenziert.

4. **Verarbeitung durch die Instagram-API**

   - Das Webbackend verwendet die Instagram-API, um das Bild zu veröffentlichen:
     - Zuerst wird ein Container erstellt, um die Metadaten des Inhalts zu speichern.
     - Anschließend wird das Bild hochgeladen, und die ID des Containers wird zurückgegeben.
     - Die Veröffentlichung des Inhalts erfolgt, und die Content-ID wird zurückgegeben.

5. **Bestätigung des erfolgreichen Uploads**

    - Nach Abschluss aller Schritte sendet das Webbackend eine Bestätigung an das Webfrontend, welches dem Nutzer den erfolgreichen Abschluss des Uploads anzeigt.

Das Diagramm verdeutlicht den gesamten Workflow des Upload-Prozesses, einschließlich der Interaktionen zwischen verschiedenen Systemkomponenten. Es zeigt die Synchronität der Kommunikation und die Abhängigkeiten zwischen Frontend, Backend, externen Services und APIs. Dies unterstützt die Entwickler bei der Implementierung und Fehlerbehebung des Upload- und Veröffentlichungsprozesses.

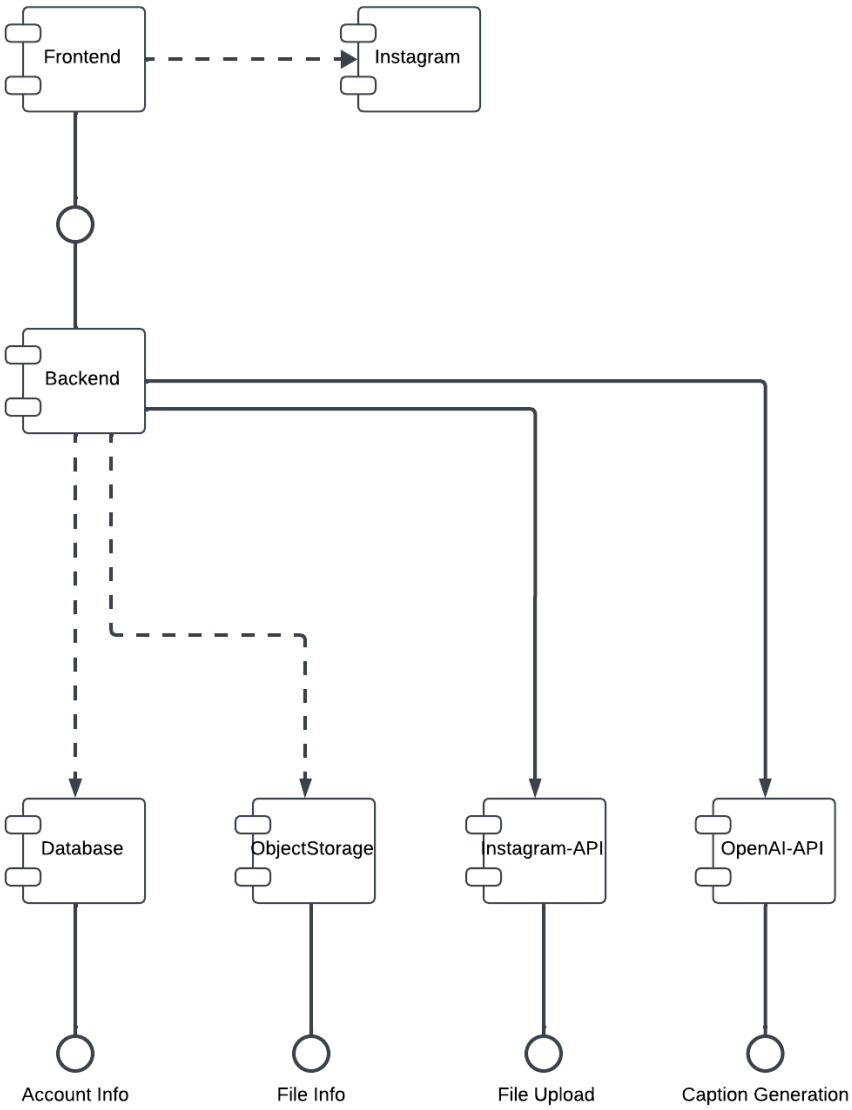

Abb. 5: Komponentendiagramm[9]

[9]Notation nach: `https://www.lucidchart.com/pages/uml-component-diagram`

Abbildung 5 zeigt das **Komponentendiagramm**, das die Architektur der Webanwendung beschreibt. Es zeigt die Hauptkomponenten sowie ihre Beziehungen und Abhängigkeiten zueinander. Nachfolgend werden die einzelnen Komponenten und ihre jeweiligen Funktionen beschrieben sowie die Beziehungen zwischen ihnen erläutert:

1. **Frontend**
    - Das Frontend stellt die Benutzeroberfläche bereit und ermöglicht die Interaktion des Nutzers mit der Anwendung.
    - Es kommuniziert direkt mit der Instagram-Plattform, um Benutzeranmeldungen oder Beitragsvorschauen zu ermöglichen.
2. **Backend**
    - Das Backend ist das zentrale Steuerungselement der Anwendung. Es verarbeitet die Daten aus dem Frontend und kommuniziert mit mehreren anderen Komponenten, darunter:
        - Datenbank: Zum Speichern von Benutzerdaten, Beitragsinformationen und Metadaten.
        - Object Storage: Für die sichere Speicherung hochgeladener Mediendateien wie Bilder oder Videos.
        - Instagram-API: Zum Veröffentlichen von Beiträgen oder Planen von Inhalten direkt auf Instagram.
        - OpenAI-API: Zur Generierung von Hashtags oder Beschreibungen basierend auf den hochgeladenen Inhalten.
3. **Datenbank**
    - Die Datenbank speichert strukturierte Daten wie Benutzerinformationen, geplante Veröffentlichungen und Protokolle der Anwendung.
4. **Object Storage**
    - Der Object Storage dient der Speicherung und Verwaltung von Mediendateien wie Bildern und Videos, die mit Beiträgen verknüpft sind.
5. **Instagram-API**
    - Die Instagram-API wird verwendet, um Inhalte auf Instagram zu veröffentlichen oder Interaktionen mit der Plattform zu ermöglichen.
6. **OpenAI-API**
    - Die OpenAI-API wird verwendet, um KI-basierte Funktionen wie die Generierung von Textinhalten oder Vorschlägen für Hashtags bereitzustellen.

Beziehungen zwischen den Komponenten

- Das Frontend kommuniziert direkt mit dem Backend, das als zentrales Gateway für alle weiteren Abhängigkeiten fungiert.
- Das Backend verbindet sich mit der Datenbank und dem Object Storage, um Inhalte zu speichern und zu verwalten.
- Die Instagram-API und die OpenAI-API erweitern die Funktionalität, indem sie externe Dienste integrieren.

Das Diagramm verdeutlicht die Modularität und Erweiterbarkeit der Anwendung und stellt sicher, dass die Komponenten klar voneinander getrennt sind, was die Wartbarkeit und Skalierbarkeit der Architektur unterstützt.

3 Arbeitspakete und Dokumentation der Beiträge

Arbeitspakete Zu Beginn des Projekts erfolgte eine allgemeine Aufteilung in zwei Teams. Team 1, bestehend aus David und Nico, war für die Entwicklung und das Aufsetzen des Frontends verantwortlich. Team 2, bestehend aus Leonard und Tobias, übernahm die Entwicklung des Backends.Die Aufteilung der Entwicklungsaufgaben in Arbeitspakete erfolgte anhand von Sprints. Im Rahmen dieses Projekts wurden insgesamt vier Sprints durchgeführt. Im Folgenden werden die wichtigsten Ziele und wesentlichen Meilensteine der einzelnen Sprints dargestellt.

1. **Sprint 1** (03.12-06.12.2024)
    - Das Ziel des ersten Sprints war die Definition der MUSS-, SOLL- und KANN-Anforderungen sowie die Entwicklung des Konzepts, das ausführlich in Kapitel 2 beschrieben wurde. Außerdem sollte ein erstes Frontend aufgesetzt und eine Interaktion mit der Instagram-API ermöglicht werden. Ein weiteres Ziel war die Durchführung eines ersten Bild-Uploads, um die Verbindung zur API zu testen. In diesem Sprint konnten folgende Erfolge erzielt werden:
        - Bild-Upload
        - Frontend mit Drag & Drop Support
        - Verbindung zur Instagram-API
2. **Sprint 2** (06.12-13.12.2024)
    - Das Ziel des zweiten Sprints war die Implementierung eines Backend-Logins, um Nutzern eine sichere Authentifizierung zu ermöglichen. Zudem sollte das Frontend um ein responsive Design erweitert werden, um die Anwendung auf unterschiedlichen Geräten optimal nutzbar zu machen. Zudem wurde die Architektur der Anwendung weiter verfeinert. Die Ergebnisse dieses Sprints umfassen:
        - Backend-Log-In
        - Responsive Design
3. **Sprint 3** (14.12-10.01.2025)
    - Der dritte Sprint hatte das Ziel, eine Datenbankanbindung herzustellen, um Benutzerdaten und Inhalte effizient zu speichern und abzurufen. In der Datenbank werden zentrale Informationen zu hochgeladenen Medien, wie z.B. Medien-URLs, Erstellungszeitpunkte und Veröffentlichungsstatus, verwaltet. Darüber hinaus sollte die Möglichkeit des Video-Uploads implementiert werden, um die Funktionalität der Anwendung zu erweitern. In diesem Sprint konnten folgende Erfolge erzielt werden:
        - Datenbankanbindung

  - Video-Upload

4. **Sprint 4** (10.01-17.01.2025)

   - Im vierten und letzten Sprint lag der Fokus auf der Implementierung eines Kalenders, der Nutzern ermöglicht, Veröffentlichungen zu planen und geplante Beiträge einzusehen. Zusätzlich wurde ein Scheduler integriert, um geplante Veröffentlichungen automatisch durchzuführen. Schließlich wurde die OpenAI-API angebunden, um KI-gestützte Funktionen wie das Generieren von Hashtags und Bildbeschreibungen bereitzustellen. Darüber hinaus wurden verschiedene Tests implementiert, um die volle Funktionalität der Anwendung sicherzustellen. Im Rahmen dieses Sprints wurden folgende Ziele erreicht:

     - Kalender
     - Scheduler
     - OpenAI-API-Anbindung
     - Tests

Es ist zu beachten, dass die hier dargestellten Punkte nicht den vollständigen Umfang der umgesetzten Features und Funktionalitäten widerspiegeln. Sie stellen lediglich einen Auszug dar, enthalten jedoch alle MUSS-Anforderungen. Eine detaillierte Beschreibung der gesamten Anwendung erfolgt im nachfolgenden Kapitel 4.

Dokumentation der Beiträge Die Teams und die Aufgabenteilung (Backend und Frontend) wurden während des gesamten Projektzeitraums beibehalten. Es ist zu beachten, dass häufig mit Pair Programming gearbeitet wurde, weshalb einzelne Personen deutlich weniger Commits in GitHub aufweisen. Die Dokumentation wurde von allen Mitgliedern zu gleichen Teilen erstellt.

4 Umsetzung

4.1 Beschreibung des Programms

4.2 Beschreibung der verwendeten Services

4.3 Beschreibung der APIs

4.4 Anleitung zum Starten der Anwendung

Literaturverzeichnis

Meta Platforms, Inc. (2025): Facebook Graph API Documentation. URL: https://developers.facebook.com/docs/graph-api/.

shadcn (2025): shadcn/ui Documentation. URL: https://ui.shadcn.com/docs.

Tailwind Labs (2025): Tailwind CSS v2 Documentation. URL: https://v2.tailwindcss.com/docs.

Vercel (2025): Next.js Documentation. URL: https://nextjs.org/docs.

Erklärung zur Verwendung generativer KI-Systeme

Bei der Erstellung der eingereichten Arbeit habe ich die nachfolgend aufgeführten auf künstlicher Intelligenz (KI) basierten Systeme benutzt:

1. ChatGPT-4o

Ich erkläre, dass ich

- mich aktiv über die Leistungsfähigkeit und Beschränkungen der oben genannten KI-Systeme informiert habe,[10]
- die aus den oben angegebenen KI-Systemen direkt oder sinngemäß übernommenen Passagen gekennzeichnet habe,
- überprüft habe, dass die mithilfe der oben genannten KI-Systeme generierten und von mir übernommenen Inhalte faktisch richtig sind,
- mir bewusst bin, dass ich als Autorin bzw. Autor dieser Arbeit die Verantwortung für die in ihr gemachten Angaben und Aussagen trage.

Die oben genannten KI-Systeme habe ich wie im Folgenden dargestellt eingesetzt:

Arbeitsschritt in der wissenschaftlichen Arbeit	**Eingesetzte(s) KI-System(e)**	**Beschreibung der Verwendungsweise**
Korrektur der Arbeit	ChatGPT-4o	Einzelne Kapitel ChatGPT zum Korrigieren gegeben. Erfolg: geringfügig, nach der Korrektur wurden noch einige Fehler gefunden.

[10] U.a. gilt es hierbei zu beachten, dass an KI weitergegebene Inhalte ggf. als Trainingsdaten genutzt und wiederverwendet werden. Dies ist insb. für betriebliche Aspekte als kritisch einzustufen.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Webanwendung zur Veröffentlichung von Instagram-Beiträgen* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

(Ort, Datum) (Unterschrift)