InfluenzaConnect: Eine React-basierte Webanwendung für Influencer-Marketing

Technical Reports: CL-2024-42, Mai 2024

Sebastian Weidner, Jonas Hermann, Nils Bayerl, Dominik Schwagerl,
Timon Spichtinger, Christoph P. Neumann
CyberLytics-Lab at the Department of Electrical Engineering, Media, and Computer Science
Ostbayerische Technische Hochschule Amberg-Weiden
Amberg, Germany

Zusammenfassung*—*InfluenzaConnect ist eine Webandwendung, die das Ziel der strategische Vernetzung von Unternehmen und Influencern verfolgt. Dabei soll das Influencer Marketing von Unternehmen, als auch die selbstständige Vermarktung als Influencer erleichtert werden. Die Besonderheit von InfluenzaConnect ist u.a. die automatisierte Analyse des Instagram-Profils des Influencers bei der Registrierung.
Die Architektur basiert im Frontend auf dem React JavaScript-Framework [1], im Backend auf dem Python-Framework Flask [2] und für die Persistenz wird die NoSQL Datenbank MongoDB [3] verwendet. Frontend, Backend und die MongoDB wurden mittels Docker [4] containerisiert, um eine hochskalierung der Webanwendung zu ermöglichen. Als CSS Framework wurde Tailwind CSS [5] verwendet und für die Datenaquise auf die Instagram-API [6] zurückgegriffen.

***Index Terms*—Influencer-Marketing; WebApp; Business; SocialMedia.**

I. Überblick

A. Motivation

Unternehmen wollen möglichst viele ihrer Produkte verkaufen. Dazu braucht es eine gute Qualität der Produkte, preiswerte Verkaufspreise und Markenbekanntheit. Letzteres erfordert ein ausgereiftes Marketing-Konzept. Ein Marketing-Konzept ist dabei das Influencer-Marketing. Beim Influencer-Marketing lassen Unternehmen ihre Produkte von Influencern bewerben, um den Bekanntheitsgrad der Firma, sowie den Bekanntheitsgrad und die Umsatzzahlen ihrer Produkte weiter zu erhöhen.

Was ist das Problem?
Unternehmen wollen authentische, sowie glaubwürdige Stimmen von Menschen finden, die ihr Produkt präsentieren und bewerben können. Die Lösung auf dieses Problem ist das Influencer-Marketing. Dieses ist aber noch nicht so weit verbreitet und Unternehmen, die sich für dieses Marketing-Konzept entschieden haben, stehen wieder vor neuen Problemen.

1) Wie finde ich den passenden Influencer für meine Produkte?
2) Es ist aufwending mit mehreren Influencern zu verhandeln –> Wie kann ich mit mehreren Influencern effizient verhandeln?
3) Welche Social-Media-Plattformen kommen für das Bewerben meiner Produktes infrage, bzw. wie finde ich die richtige Plattform für meine Produkte? Soll es Instagram, Facebook, TikTok, YouTube, Pinterest, X, LinkedIn oder doch ein privater Blog sein, um nur einen Einblick über die Vielfalt der Plattformen zu bieten.

Dazu kommen generelle Probleme wie:

4) Es gibt noch nicht viele Influencer die im B2B-Space tätig sind.
5) Es gibt viele unbekannte Influencer.
6) Influencer mit hohen Reichweiten sehen sich selbst gar nicht als Influencer an.
7) Influencer mit kleineren Reichweiten sind schwer auffindbar und haben es schwerer, Aufträge von Unternehmen zu bekommen.

Durch unsere Webbasierte Influencer-Marketing-Plattform wollen wir es schaffen, diese Probleme zu beheben und das Influencer-Marketing zu vereinfachen.

B. MVP

Das MVP umfasst folgende Anforderungen:

1) *Registrierungs- und Anmeldungsseite für Influencer*
2) *Automatische Analyse des Instagram-Profils des Influencers*
3) *Tabellarische Übersicht aller registrierten Influencer, inkl. Sortierung, Suche und Filterungsfunktion*
4) *Detailübersicht zum Influencer mit allen relevanten Informationen*

C. Implementierungsstand

Vom MVP wurden folgende Punkte umgesetzt:

1) *Registrierungs- und Anmeldungsseite für Influencer*
2) *Automatische Analyse des Instagram-Profils des Influencers während der Registrierung*
3) *Übersicht über alle registrierten Influencer, inkl. Suche und Spaltenfilter*

Die *Detailübersicht zum Influencer* wurde nicht implementiert. Stattdessen wurde für den *angemeldeten Influencer eine Profilseite implementiert, in der er seine aktuellen Profildaten einsehen und ändern kann*. Um alle Daten zum Influencer trotz der fehlenden Detailansicht für die Besucher der Website bereitzustellen, wurden diese vorerst in die Tabelle in der Übersichtsseite zu den Influencern integriert.

II. Architektur

A. Containerisierung

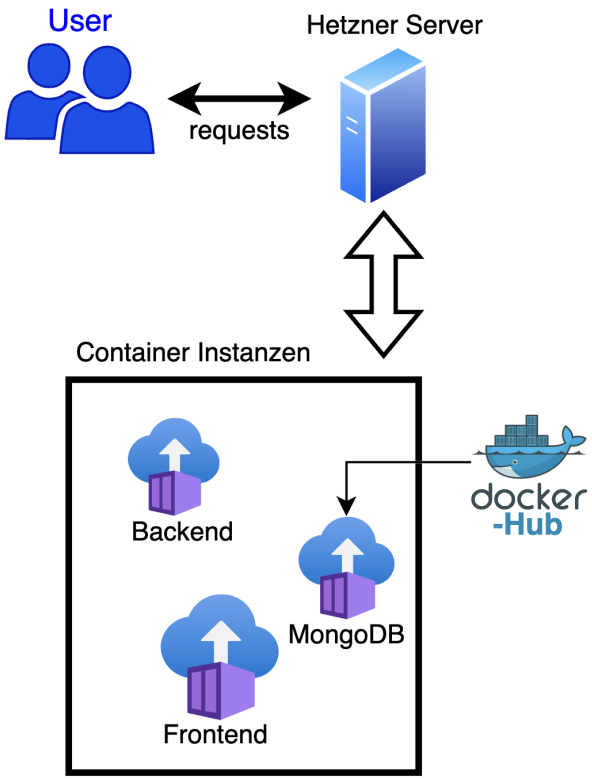

Abbildung 1. Containerisierung und Deployment

Das Projekt wurde mittels Docker [4] containerisiert, um eine flexible und skalierbare Umgebung für die verschiedenen Komponenten der Anwendung zu schaffen. Dabei wurden das Frontend, Backend sowie die MongoDB [3] in separate Docker-Container aufgeteilt, was eine einfache Verwaltung und Skalierung der einzelnen Dienste ermöglicht. Dockerfiles legen dabei fest, welche Software und Abhängigkeiten in den virtuellen Containern installiert werden. Für das Backend wird ein Python 3.10 [7] Slim Image verwendet. In dieses werden beim Starten des Containers die notwendigen Python-Pakete automatisiert installiert und der Anwendungscode kopiert. Das Frontend basiert auf einem Node.js [8] 16 Image, das ebenfalls automatisch die benötigten Node-Module installiert und den Build-Prozess des Frontends startet (Kein multi stage).

B. Deployment

InfluenzaConnect verwendet GitHub [9] als Git-Respository. Das Deployment in unsere Hetzner-Cloud [10] erfolgt demnach mittels GitHub Actions [11]. Der CD-Workflow befindet sich wird dabei ausgelöst, sobald Änderungen auf den `prod` Branch gepusht werden. Der anschließende Deployment-Job läuft auf einer Ubuntu-Maschine in der Hetzner-Cloud und umfasst folgende Schritte:

1) **Checkout Code** - Der Quellcode wird aus dem Repository ausgecheckt.
2) **Setup SSH** - SSH wird eingerichtet, um eine sichere Verbindung zum Hetzner-Server herzustellen.
3) **Deploy to Hetzner** - Der Code wird auf den Hetzner-Server deployt.

Beim deployen wird dabei unser Projektverzeichnis aktualisiert, Docker Compose [12] verwendet, um die bestehenden Container zu stoppen, neu zu bauen und schließlich die Anwendung in die Produktionsumgebung hochzufahren. Durch diese Pipeline wird sichergestellt, dass jede Änderung im `prod` Branch automatisch auf dem Produktionsserver deployed wird. Dadurch wird eine kontinuierliche Integration und Bereitstellung neuer Features und Bugfixes ermöglicht, Entwicklungszyklen beschleunigt und die Qualität der Anwendung verbessert.

III. Techstack

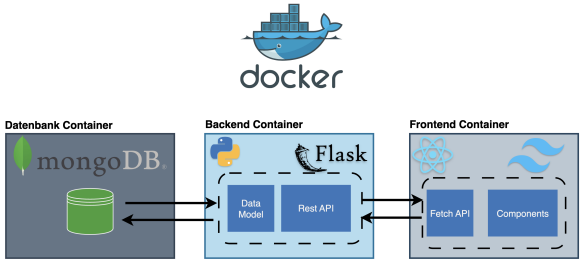

Abbildung 2. Architektur von InfluenzaConnect

A. Frontend

Für die Realisierung des Frontends wird das JavaScript-Framework React [1] verwendet. Es erleichtert die Entwicklungsarbeit im Frontend, indem es dabei hilft, wiederverwendbaren Code zu schreiben, sowie graphische Inhalte dynamisch aufzubauen und zu befüllen. HTML, CSS und JavaScript-Code wird dabei in sogenannten Komponenten gekapselt. In React gibt es sog. Hooks, die dazu benutzt werden, bestimmte Funktionalitäten in den Komponenten zu erreichen. Erwähnenswert ist hier der Hook `useState()`, der bestimmte Zustände zwschenspeichern kann, sodass diese beim erneuten Rendern der Komponente nicht verloren gehen. Des Weiteren haben wir uns für die Gestaltung der Webpages für Tailwind CSS [5] entschieden. Tailwind CSS wird oft mit React [1] verwendet und ist zur Zeit sehr im Trend, weil es leicht zu erlernen ist, gutes vordefiniertes Design und trotzdem große Gestaltungsfreiräume bietet, da man notfalls interne CSS-Werte speziell für sein Projekt überschreiben kann. Außerdem haben wir entschlossen, anstatt JavaScript [13] auf TypeScript [14] zu setzen, da es komplett kompatibel mit JavaScript ist, eine Typisierung zulässt und somit logische Typfehler beseitigt. Für die Kommunikation zwischen Frontend und Backend wird Axios [15], ein Promise-basierter HTTP-Client sowie die Fetch API [16] verwendet.

B. Backend

Im Backend haben wir uns für das Python-Webframework Flask [2] entschieden. Flask ermöglicht die Erstellung von Webanwendungen in Python durch eine minimalistiche und flexible Struktur, sodass es sich gut für kleine bis mittelgroße Projekte eignet. Durch das Definieren von Routen, basierend auf den Anforderungen der Clients werden HTTP-Anfragen vom Frontend verarbeitet und entsprechende Antworten im JSON-Format [17] zurückgeben.

C. Persistenz

Zum persistieren der Daten setzen wir auf eine MongoDB [3] als Datenbanksystem. Die `pymongo`-Bibliothek [18] von Python fungiert dabei als offizieller Treiber für die MongoDB, die Funktionen für die Interaktion der Datenbank bereitstellt, um Daten zu speichern, abzurufen, aktualisieren und zu löschen.

IV. Implementierung

A. Frontend

1) *Landing Page*
Die Landing Page wurde einladend gestaltet und enthält ansprechende Beschreibungen zu den Hauptfunktionen von InfluenzaConnect. Der User kann zwischen "Spectator" und "Influencer" auswählen, um die passende Seite zu besuchen. Außerdem kann er über die Navigationsleiste zur gewünschten Seite navigieren.

2) *Registrierung*
Die Registrierung dient dazu, sich als Influencer registrieren und vermarkten zu können. Die Registrierung ist in drei Schritten aufgebaut: Zuerst muss er seine Anmeldeinformationen - eingeben. Im Zweiten Schritt werden seine Persönliche Daten abgefragt, die später teilweise für die Unternehmen veröffentlicht werden und im Dritten Schritt muss er seinen Instagram-Account hinterlegen, der anschließend im Backend analysiert wird. Nachfolgend finden Sie eine Detailierte Auflistung der Eingaben:
   - Anmeldeinformationen - Email, Passwort
   - Persönliche Daten - Anrede, Vorname, Nachname, Nationalität, Telefonnr., Gesprochene Sprachen, Beschreibung über sich selbst
   - Verknüpfung Socail-Media Accounts - Instagram Username

   Jeder der drei Schritte wurde in einem Eigenständigen Formular realisiert, zwischen denen durch einen Weiter- und Zurück-Button hin- und hergewechselt werden kann. Für die Eingabefelder wurden von Grund auf eigene Komponenten erstellt. Ein Eingabefeld besitzt immer ein Label und ein Ausgabefeld für Fehlermeldungen. Dabei taucht unter jedem Eingabefeld eine Fehlermeldung auf, falls die eingegebenen Daten nicht den Richtlinien entsprechen. Um das Datenhandling mit den Input-Felder zu vereinfachen und um Codezeilen zu sparren, wurde die Bibliothek `React Hook Form` [19] im Frontend eingebunden. Durch diese kann man vereinfacht die Logik hinter den Formularen erstellen. Formulareingaben können so z.B. ganz einfach zwischenspeichert werden, um diese z.B. beim zurückgehen nicht zu verlieren. Auch das einbinden einer Datenvalidierung wird unterstützt. So werden die Eingaben mit der Bibliothek `Yup` [20] validiert, die gleichzeitig das Setzten eigener Fehlermeldungen, nach jedem Validierungsschritt erlaubt. Die Fehlermeldungen werden anschließend in ein internes Objekt, dass `React Hook Form` zu jedem registrierten Input-Element erstellt, geschrieben und durch unsere Komponente automatisch unter dem Input-Feldern angezeigt.
   Wurden alle Eingaben getätigt werden diese gesammelt an das Backend gesendet und dort nochmals überprüft. Neben einem einfachen Eingabefeld, wurden Komponenten für ein Select-, ein Multiselectdropdown und für zwei Buttons passend zur Registrierung erstellt. Das Multiselectdropdown wurde dabei als interaktive HTML-Liste in React realisiert, dass ausgewählte Einträge im Input-Feld der Komponente anzeigt. Durch ein kleines rotes Kreuz, kann man diese direkt ohne das Dropdown erneut zu öffnen wieder deselektieren.

3) *An- und Abmeldung*
Die Anmeldung erfolgt durch eine eine Popup-Dialog-Komponente, die unsere wiederverwendbaren Input-Komponenten enthält. Nach erfolgreicher Anmeldung wird zum Benutzer eine Session gesetzt und das Profil des Nutzers in der Navigationsbar anzeigt. Der Logout-Button löscht die Session und stellt das ursprüngliche Erscheinungsbild wieder her.

4) *Profildaten Bearbeiten*
Die Profildaten können über eine Eingabemaske nachträglich verändert werden. Die Daten werden anhand der in der Session gespeicherten E-Mail abgerufen. Standardmäßig ist die Eingabemaske nicht editierbar. Sie wird es erst, wenn der Edit-Button betätigt wird. Nachdem die Daten eingegeben wurden, können diese Bestätigt werden. Nach einer Validierung der Daten im Frontend werden diese anschließend ins Backend gesendet, dort nochmals validiert und schlussendlich persistiert, indem die alten Daten überschrieben werden.

5) *Influncerübersicht*
Die Influencerübersicht dient dazu, dass Unternehmen einen passenden Influencer für sich finden können. Demnach sind in einer Tabelle alle Registrierten Influencer mit allen relevanten Daten, inkl. Informationen, die durch die Analyse des Instagram-Accounts gewonnen wurden zu sehen. Für die Datendarstellung in der Tabelle wurde in TypeScript eine Datenstruktur erstellt, auf die die Daten vom Backend gamappt werden.
   Ein **Spaltenfilter** ermöglicht es, Spalten ein- und wieder auszublenden. Die Komponente ist dabei so ähnlich wie das Multiselect-Dropdown in der Registrierung aufgebaut,

nur das die Werte mittels einer Checkbox ausgewählt werden können und ausgewählte Einträge nicht außerhalb des Dropdowns sichtbar werden. Die Komponente wurde ohne `React Hook Form` realisiert. Stattdessen wird ihr eine `onChange()` Event-Handler übergeben, der die ausgewählten Spalten in einer Zustands-Variable vom Typ `useState()` speichert. Vor dem Rendering der Spalten muss somit nur noch überprüft werden, ob die angegebene Spalte in dieser Variable enthalten ist.

Durch **Suchfeld** kann der User die Tabelle direkt nach dem Namen eines bestimmten Influencer durchsuchen. Dazu wird ebenfalls der Wert des Suchfeldes in eine Variable vom Typ `useState()` gespeichert. Anschließend wird eine Filterfunktion mithilfe des Inhalts dieser Variable auf allen Daten angewendet und die Tabelle aktualisiert.

B. Backend

Die Hauptaufgaben des Backends umfassen die Aufbereitung und Bereitstellung von Daten für das Frontend sowie die Verarbeitung von Benutzeranfragen. Durch das Definieren von Routen in Flask, kann das Frontend mit dem Backend kommunizieren und entsprechende Antworten auf die Anfragen im JSON-Format zurückgeben. In unserem Fall haben wir folgende Routen definiert:

Route	Zweck der Route
`/signup`	Registrierung: Die eingegebenen Daten werden validiert und verschlüsselt in der MongoDB gespeichert.
`/login`	Anmeldung: Die Eingaben werden überprüft, und bei erfolgreicher Authentifizierung eine Session gestartet.
`/profile`	Profildaten ändern: Gibt die Profilinformationen des eingeloggten Benutzers zurück.
`/logout`	Abmelden: Beendet die aktuelle Sitzung und meldet den Benutzer ab.
`/collect`	Influencer Übersichtstabelle: Gibt die Daten aller registrierten Influencer gesammelt zurück, um diese im Frontend anzuzeigen
`/<session>`	Routen für Session-Handling

Tabelle I
DEFINIERTE API-ROUTEN

C. Datenvalidierung und -speicherung

Die Validierungsfunktionen prüfen die Benutzereingaben mithilfe von regulären Ausdrücken und einfachen Bedingungen, um sicherzustellen, dass sie den erwarteten Standards entsprechen. Dabei wird z.B. überprüft, ob die E-Mail im richtigen Format vorliegt, das Passwort ausreichend lang ist oder obligatorische Informationen wie Vor- und Nachname korrekt angegeben wurden. Spezielle Validierungen wie die Überprüfung des Instagram-Benutzernamens und optionaler Informationen wie der Telefonnummern werden ebenfalls durchgeführt. Darüber hinaus nutzen diese Funktionen externe Ressourcen, beispielsweise eine Webanfragen für die Überprüfung von Instagram-Benutzernamen auf ihre Existenz und ob es sich um ein privates oder öffentliches Profil handelt.

1) Session: In unserer Flask-Anwendung nutzen wir die mittgelieferte Session-Verwaltung der Flask-Bibliothek. Wenn sich ein Benutzer Anmeldet oder Registriert, wird seine E-Mail-Adresse in einer Session gespeichert, um den Benutzer während seines Besuchs auf der Website identifizieren und personalisierte Funktionen bereitstellen zu können. Die Bibliothek bietet uns ebenfalls die Möglichkeit, Session-Daten sicher zu verwalten. Zum Schutz sensibler Informationen, werden die Daten verschlüsselt übertragen und nach 30 Minuten der Inaktivität läuft die Session automatisch ab, sodass die Daten vor unautorisiertem Zugriff bewahrt werden.

D. Instagram-Account Analyse

Durch die Kombination der `Instaloader`-Labrary [21] zur Extraktion der Profil- und Postdaten sowie GPT-3.5-turbo [22], können umfangreiche Auswertungen erstellen erstellt werden. Die Library extrahiert sämtliche Daten des Influencers, mitsamt der Daten der 100 zuletzt geposteten Beiträge, inkl. Bild, Bildunterschriften, Kommentare und vieles mehr. Nachfolgend sind die durchgeführten Analysen aufgelistet:

- **Zuordnung Produkt-Werbesparte:** Das Sprachmodell führt anhand der Hashtags unter den Beiträgen eine semantischen Analyse durch, indem es den Influencer eine primäre und sekundäre Produkt-Werbesparte zuordnet. Diese soll dann die Interessensgebiete des Influencers widerspiegeln.
- **Erfassung der Kerndaten des Profils:** Desweiteren werden anhand der letzen 100 Posts durchschnittliche Werte wie Anzahl der Kommentare und Likes des Influencers berechnet. Auch wird der Zeitpunkt des letzten Posts erfasst. Diese Information ist wichtig, um die Frequenz und Aktualität der Beitragsaktivität des Influencers bewerten zu können.
- **Download von Posts und Profilbild:** Das Profilbild des Accounts und die neuesten Posts werden für spätere Analysen heruntergeladen und gespeichert. Später soll eine visuelle Auswertung der Bilder weitere Informationen Unternehmen zur Verfügung stellen. Das Profilbild wird dabei als Profilbild in InfluenzaConnect übernommen.
- **Berechnung der Engagement-Rate:** Die Engagement-Rate ist ein Maß für die Interaktion der Follower mit den Beiträgen eines Influencers, die aus Likes-, Kommentar und Follower-Anzahl berechnet wird. Kommentare werden dabei stärker gewichtet, da sie mehr Interaktionsaufwand erfordern als Likes. Die Engagementrate wird mit folgender Formel berechnet:

$$\text{Engagement-Rate} = \frac{\text{Likes} \times 0.3 + \text{Kommentare} \times 0.7}{\text{Follower}}$$

E. Persistenz

In der Datenbank sind die Benutzerinformationen strukturiert in der Collection `registration` als flaches JSON-Objekt gespeichert. Die Analysedaten des Instagram-Accounts werden hingegen seperat, wieder als flaches JSON-Objekt in einer Collection `analysis` hinterlegt.

Die Hauptaufgaben der Datenbankschicht umfassen das Speichern, Aktualisieren und das Abrufen dieser Benutzerdaten. Es wird sichergestellt, dass zurückgegebene Daten immer das definierte Standardformat einhalten, um die Konsistenz weiterer Verarbeitungsschritte innerhalb der Anwendung sicherzustellen.

Fehler während dem Speichern oder dem Abrufen von Benutzerdaten werden für eine zuverlässige Fehlerbehandlung genau protokolliert.

V. Testen

A. Frontend

Im Frontend wurde das Testing-Framework Jest [23] verwendet. Es kann sowohl Komponenten einzeln auf ihre Funktionalität (Unit-Test), als auch die Interaktion mit anderen Komponenten überprüfen. Es gibt folgende Testarten:

- **Rendering-Tests** - Überprüfen, ob alle Formularelemente korrekt gerendert werden.
- **Tests der Benutzerinteraktion** - Simulieren von Benutzereingaben und Interaktionen und Sicherstellung der korrekten Erfassung von Eingaben und das anzeigen von Fehlermeldungen.
- **Zugänglichkeitstests** - Überprüfung der Zugänglichkeit von Formularelementen über ihr Text-Label, sowie der Navigierbarkeit von interaktiven Elementen der Anwendung über die Tastatur.

B. Backend

Die Verwendung von pytest [24] ermöglichte uns umfassendes, aber effizientes Testen unserer Datenvalidierungslogik. Pytest ist ein Testframework für Python, das das Schreiben, Organisieren und Ausführen von Tests vereinfacht, da diese mit einfachen Mitteln zu implementieren sind. Durch die Nutzung der Funktionen von pytest wie Fixtures, Mocks und Parametrisierung konnten wir eine robuste Validierungslogik für die eingegebenen Daten des Users sicherstellen, bzw. potenzielle Probleme beseitigen und so schlussendlich die Zuverlässigkeit der gesamten Anwendung verbessern.

VI. Fazit und Ausblik

InfluenzaConnect bietet zum aktuellen Zeitpunkt nur einen kleinen Ausschnitt der geplanten Funktionalität. Das MVP wurde in der zu verfügbaren Zeit nur knapp verfehlt, dafür wurde die WebApp sehr ansprechend und solide aufgebaut. Ein wichtiger Schritt wurde hinsichtlich der automatischen Analyse des Instagram-Profils des Influencers getan. Diese kann zukünftig noch erweitert, verbessert und auf weitere Social-Media-Plattformen angewendet werden.

Es gibt, wie im bereits Fachkonzept beschrieben, noch etliche vielversprechende Möglichkeiten zur Erweiterung der Anwendung, wie eine Detailansicht und eine Vergleichsfunktion zwischen Influencern, sowie eine Registrierung für Unternehmen und eine Kommunikationsfunktionalität. Darüber hinaus sollte die Sicherheit und der Datenschutz der Anwendung weiter verbessert werden, insbesondere in Bezug auf die Verarbeitung und Speicherung personenbezogener Daten. Die CD-Automatisierung mittels Docker [4] und GitHub Actions [11] bietet dafür eine robuste Basis für den Betrieb und die Weiterentwicklung von InfluenzaConnect.

Literatur

[1] Facebook. *React*. [Online]. URL: https://react.dev/.

[2] Pallets. *Flask User's Guide*. [Online]. URL: https://flask.palletsprojects.com/.

[3] Dwight Merriman, Eliot Horowitz und Kevin Ryan. *MongoDB*. [Online]. URL: https://www.mongodb.com/.

[4] Docker. *Docker: Accelerated Container Application Development*. [Online]. URL: https://www.docker.com/.

[5] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* URL: https://tailwindcss.com/ (besucht am 01. 07. 2024).

[6] Facebook. *Instagram Basic Display API*. [Online]. URL: https://developers.facebook.com/docs/instagram-basic-display-api?locale=de_DE.

[7] Guido van Rossum. *Python*. [Online]. URL: https://www.python.org.

[8] Ryan Dahl. *Node.js*. [Online]. URL: https://www.nodejs.org/.

[9] *GitHub*. URL: https://github.com/ (besucht am 29. 06. 2024).

[10] *Günstiges Cloud Hosting*. URL: https://www.hetzner.com/de/cloud/ (besucht am 01. 07. 2024).

[11] *GitHub*. URL: https://github.com/features/actions (besucht am 29. 06. 2024).

[12] Docker. *Compose: Defining and Running Multi-Container Docker Applications*. [Online]. URL: https://docs.docker.com/compose/.

[13] *ECMA-262*. Ecma International. URL: https://ecma-international.org/publications-and-standards/standards/ecma-262/ (visited on 07/01/2024).

[14] Microsoft. *TypeScript: JavaScript With Syntax For Types*. [Online]. URL: https://www.typescriptlang.org/.

[15] *Axios*. URL: https://axios-http.com/ (besucht am 01. 07. 2024).

[16] *Fetch Standard*. URL: https://fetch.spec.whatwg.org/#fetch-method (besucht am 01. 07. 2024).

[17] *JSON*. URL: https://www.json.org/json-de.html (besucht am 01. 07. 2024).

[18] MongoDB, Inc. *PyMongo*. [Online]. URL: https://pypi.org/project/pymongo/.

[19] *Home*. URL: https://www.react-hook-form.com/ (besucht am 01. 07. 2024).

[20] Jason Quense. *jquense/yup*. original-date: 2014-09-22T23:54:22Z. 1. Juli 2024. URL: https://github.com/jquense/yup (besucht am 01. 07. 2024).

[21] *Instaloader — Download Instagram Photos and Metadata*. URL: https://instaloader.github.io/ (besucht am 01. 07. 2024).

[22] *gpt-3-5-turbo*. URL: https://platform.openai.com (visited on 07/01/2024).

[23] Facebook. *Jest is a delightful JavaScript Testing Framework with a focus on simplicity*. [Online]. URL: https://jestjs.io/.

[24] *pytest: helps you write better programs - pytest documentation*. URL: https://docs.pytest.org/en/8.2.x/ (besucht am 01. 07. 2024).