Einführung

Im Zeitraum vom 01.03.2025 – 31.08.2025 habe ich mein Pflichtpraktikum bei der ilume Informatik AG in Mainz absolviert. Es handelt sich um ein IT-Beratungshaus, das sich auf CRM-Systeme, die Digitalisierung von Geschäftsprozessen sowie Cloud-Lösungen spezialisiert. Neben der Integration von Standardsoftware in bestehende IT-Landschaften entwickelt ilume auch individuelle Softwarelösungen für Kunden aus verschiedensten Branchen, darunter Pharma, Chemie, Bankwesen, Versicherungen und Logistik.

Das Unternehmen wurde im Jahr 2000 gegründet und beschäftigt mittlerweile rund 150 Mitarbeitende, die in drei Abteilungen organisiert sind: DEV (Custom Development), SMA (Smart Automation) und CRM (Customer Relationship Management). Während meines Praktikums war ich im Bereich DEV tätig und dort dem Team Digitalization zugeordnet, das sich vor allem mit Web- und Mobile-Entwicklung beschäftigt und sowohl externe Kunden- als auch interne Projekte umsetzt.

Vor dem Pflichtpraktikum war ich bereits als Werkstudentin bei ilume tätig, wodurch ich mit vielen Unternehmensprozessen vertraut war und mein Praktikum effizient beginnen konnte. Etwa 5–10 % meiner Aufgaben lagen im Content-Management-Bereich, in dem ich sowohl Magnolia CMS-Seiten als auch AWS-basierte Webseiten für Kunden aus der Pharma-Branche betreute. Dabei passte ich Inhalte an, implementierte Änderungen an Funktionen und übernahm die Bildbearbeitung. Ein wesentlicher Bestandteil meiner Tätigkeit war die Mitarbeit am internen KI-basierten Webprojekt „Blog Writer“. In den letzten beiden Monaten, von Juli bis August, lag mein Schwerpunkt auf der Einarbeitung in Compose Multiplatform.

1. Woche (03.03. – 07.03.2025)

In der ersten Woche meines Praktikums fand ein Meeting mit dem Management des DEV-Teams statt, bei dem das Thema für das Projekt vorgestellt wurde. Die Grundidee war, ein Tool zu entwickeln, das täglich aktuelle Nachrichtenartikel sammelt, diese anhand der individuellen Interessen eines ilume-Mitarbeiters bewertet und daraus KI-gestützte LinkedIn-Beiträge generiert. Diese Beiträge werden per E-Mail an die Mitarbeiter geschickt und können dort für LinkedIn genutzt werden.

Das Projekt wurde in einem Zweierteam umgesetzt, wobei ich die Verantwortung für die Backend-Entwicklung übernahm. Zur Abstimmung führten wir tägliche Meetings (Dailys) durch, in denen wir den Projektfortschritt sowie offene Fragen und Herausforderungen besprachen.

Als ersten Schritt erarbeitete ich einen Fragebogen für die Kollegen, der die notwendigen Profilinformationen abfragte und es der KI dadurch erleichterte, Artikel gezielt den jeweiligen Interessen zuzuordnen. Der Fragebogen umfasste Themen wie

Studium, aktuelle Position und Aufgaben, Berufserfahrung, Interessengebiete sowie den gewünschten Schreibstil, in dem ein LinkedIn-Beitrag formuliert sein sollte. Zusätzlich konnten die Mitarbeitenden optional weitere Angaben machen, um die Beiträge noch individueller und persönlicher zu gestalten.

Da mir bereits zuvor von ilume ein MacBook zur Verfügung gestellt wurde, richtete ich darauf meine Entwicklungsumgebung ein. Dazu übertrug ich das bestehende Projektgerüst aus dem Bitbucket-Repository (Atlassian) auf meinen Rechner, das zunächst nur aus einer grundlegenden Ordnerstruktur und wenigen Konfigurationsdateien bestand. In der zugehörigen pyproject.toml-Datei waren bereits einige Einstellungen und Bibliotheken, wie Python, Poetry (Paket- und Dependency-manager für Python) und LangChain (Framework zur Erstellung von Anwendungen mit KI-Modellen) vordefiniert. Die weiteren für das Projekt notwendigen Tools musste ich eigenständig recherchieren und in meiner Entwicklungsumgebung einrichten.

LangChain als Framework bildet die Grundlage für die KI-Anwendung und ermöglicht unter anderem die Verarbeitung von Textdaten und das Verknüpfen verschiedener Modelle zu Ketten (Chains). Darauf aufbauend beschäftigte ich mich mit der Erweiterung LangGraph zur Visualisierung und Strukturierung komplexer Workflows sowie mit Agenten, die es der KI erlauben, selbstständig Entscheidungen zu treffen und auf Basis von Regeln und Eingaben Aktionen auszuführen. Dazu arbeitete ich die Tutorials von DeepLearning.AI durch, um die Funktionsweise praxisnah kennenzulernen.

2. Woche (10.03. – 14.03.2025)

In der zweiten Woche stand die praktische Einarbeitung in die eingesetzten Methoden und Frameworks im Mittelpunkt. Mir wurde ein OpenAI-API-Key zur Verfügung gestellt, mit dem ich erste Experimente durchführen konnte. Ziel war es, die Funktionen von LangChain und seinen Erweiterungen systematisch zu verstehen und praxisnah anzuwenden.
Zunächst erarbeitete ich die Grundlagen der Textanalyse und Datenaufbereitung. Hierzu erstellte ich kleinere Python-Skripte, um zentrale Konzepte wie Clustering zu wiederholen und den Aufbau von Embeddings zu testen. Embeddings wandeln Texte in hochdimensionale Zahlenvektoren um, sodass ähnliche Inhalte in diesem Raum nahe beieinander liegen. Diese Darstellung bereitet die Daten für weitere Verarbeitungsschritte vor, insbesondere für Retrievers, die gezielt relevante Informationen aus einer Sammlung von Dokumenten abrufen können.

Auf dieser Grundlage begann ich, die LangChain-API praktisch anzuwenden. Zunächst arbeitete ich mit Chains, um Verarbeitungsschritte wie Dokumentenabruf, Textaufbereitung und Embeddings linear hintereinander zu testen. Chains sind einfach und eignen sich gut für standardisierte Workflows. Für komplexere Szenarien, in denen die KI Entscheidungen treffen, unterschiedliche Pfade wählen oder externe Tools

einbinden muss, kommen Graphs mit Agenten zum Einsatz. Die Agenten steuern den Ablauf innerhalb des Graphs, treffen selbstständig Entscheidungen und wählen z. B. aus, welche Dokumente oder Tools genutzt werden sollen. Dadurch lassen sich dynamische und interaktive Workflows flexibel abbilden.

Ein weiterer Schwerpunkt lag auf dem Prompt Engineering. Durch systematisches Üben verschiedener Eingabeformate konnte ich beobachten, wie sich die Qualität und Präzision der KI-Antworten durch klare Anweisungen, Kontextgestaltung und Beispiele beeinflussen lassen. Diese Erkenntnisse bildeten die Grundlage für spätere Aufgaben, bei denen Prompts gezielt für die Generierung von LinkedIn-Beiträgen gestaltet werden müssen.

Als nächstes beschäftigte ich mich mit der ersten Aufgabe für das „Blog Writer“-Projekt: wie die relevanten Nachrichtenartikel automatisiert gesammelt werden können. Zunächst testete ich klassische Web-Scraping-Methoden, unter anderem mit BeautifulSoup zur Extraktion von Textinhalten aus HTML-Seiten sowie Selenium, um dynamisch geladene Webseiten und Interaktionen im Browser zu simulieren. Zusätzlich prüfte ich APIs wie Tavily Search und Reddit API. API-basierte Suchvorgänge lieferten oft unvollständige oder thematisch unpräzise Ergebnisse. Selenium ermöglichte zwar die Simulation von Benutzerinteraktionen, führte jedoch zu hohem Aufwand und war nicht zuverlässig, da moderne Webseiten häufig automatisierte Browser blockieren oder zusätzliche Sicherheitsmechanismen implementieren. Auch die Kombination aus API und Web-Scraping konnte nicht alle notwendigen Inhalte stabil extrahieren.

Nach intensiver Recherche entschied ich mich schließlich für einen RSS-Feed-basierten Ansatz. Zwar müssen die Feed-Links zuvor definiert werden, dafür ermöglicht die Kombination aus der Python-Bibliothek feedparser und dem RSSFeedLoader von LangChain eine schnelle und stabile Erfassung der Artikel. feedparser übernimmt die Validierung, das Parsen und die Bereinigung der Feeds, einschließlich der Extraktion von Titel, Link, Veröffentlichungsdatum und Inhaltszusammenfassung. Der RSSFeedLoader liefert den vollständigen Artikeltext direkt als Fließtext, enthält jedoch häufig keine Datumsinformationen. Durch die Kombination beider Werkzeuge lassen sich sowohl vollständige Inhalte als auch wichtige Metadaten zuverlässig erfassen und in optimaler Qualität für die weitere Verarbeitung bereitstellen.

3. Woche (17.03. – 21.03.2025)

In der dritten Woche konzentrierte ich mich auf die Auswahl einer geeigneten Datenbank für das Projekt. Da die zu verarbeitenden Inhalte wie Artikeltexte, RSS-Feeds und Personenprofile sehr unterschiedlich aufgebaut sind und sich ihre Struktur häufig ändern kann, fiel die Entscheidung auf eine NoSQL-Datenbank.