Projekt Raspberry Pi: Study Companion

Computer Science (cds-205) HS25

2025-12-15

Baltermia Clopath
FHGR
AISE25

Inhaltsverzeichnis

1. Abstract

Das Projekt beschreibt die Entwicklung eines Telegram-Bots, der Studierenden beim Organisieren ihres Studienalltags unterstützt. Ziel ist es, mit automatischen Erinnerungen, Tagesübersichten und KI-generierten Zusammenfassungen eine einfache, digitale Lernhilfe zu schaffen. Der Bot liest den Stundenplan über einen iCal-Kalender der FHGR ein und fasst die Lektionen zusammen. Technisch basiert der Code vom Bot auf dem .NET-Framework und wird in einem Docker-Container-Image geliefert, um eine einfache Installation auf einem Raspberry Pi zu ermöglichen. Ergänzend wurden Redis und PostgreSQL für die persistente Datenspeicherung integriert. Das Docker-Image wird automatisch über GitHub Actions bei jedem Update neu gebaut und veröffentlicht.

2. Einführung

2.1. Themenwahl / Relevanz

Während einer Präsentation eines alten Studenten vom Modul in unserer Blockwoche kam mir die Idee, etwas zu machen, was mir das ganze Studium über helfen könnte. Letztendlich habe ich mich dann für einen Telegram Bot entschieden. Einerseits, weil ich auf dieser Platform viel Zeit verbringe, anderseits, weil ich damit auch schon einige Erfahrung habe.

Der Bot - den ich *Study Companion* nenne - soll Studierenden helfen, den überblick über Lektionen, Prüfungen, Hausaufgaben und allgemeine Notizen zu behalten. Das macht er, indem er einerseits tägliche Erinnerungen schickt, aber auch auf Anfrage jegliche Informationen bereitstellt. Das wichtigste Feature wird das Einlesen und Verarbeiten des von der FHGR bereitgestellten iCal Stundenplans sein, womit der Bot automatisch weiss, wann welche Lektion stattfindet.

Ich finde das Thema relevant, einerseits, weil ich es andern Studierenden so einfach wie möglichen machen möchte, den Bot selbst auf einem Raspberry laufen zu lassen. Anderseits werde ich künstliche Intelligenz verwenden, um schlanke Zusammenfassungen zu generieren.

2.2. Eingrenzung des Themas

Der Bot soll im moment wirklich nur das nötigste können. Das Ziel ist es, auch in späteren Modulen - wenn sich die Möglichkeit ergibt - weitere Features hinzuzufügen. Das wichtigste:

- Einlesen von iCal
- Tägliche Zusammenfassung generiert mit AI
- Einfache Installation auf einem Raspberry Pi

2.3. Fragestellung

Die zentrale Fragestellung des Projekts lautet:
Wie kann man einen Telegram-Bot erstellen, welcher Studierenden mit automatischen Erinnerungen und Zusammenfassungen, durch Bereitstellung eines iCal-Kalenders und manueller Eingabe von Hausaufgaben den Alltag erleichtert und zudem leichte installation erlaubt?

Dazu sind noch konkretere Unterfragen formuliert:

1. Welche Frameworks und Bibliotheken eignen sich für die Entwicklung des Bots?
2. Wie kann der Bot den iCal-Kalender effizient einlesen und verarbeiten?
3. Wie kann die Installation auf einem Raspberry Pi möglichst einfach gestaltet werden?

2.4. Verwendung von Typst

Ich möchte noch ganz kurz über die Verwendung von Typst in meinem Projekt schreiben, da es meiner Meinung nach auch direkt mit der Projektarbeit zu tun hat. Typst ist eine Sprache welche ähnliche Ziele verfolgt wie LaTeX, aber meiner persönlichen Erfahrung nach viel einfacher zu verwenden ist. Da dies eine der ersten Projektarbeiten in unserem Studium ist, und auch LaTex empfohlen wurde, wollte ich dies hier kurz erwähnen. Ich empfehle, den Quellcode dieses Dokumentes anzuschauen, um einen Eindruck von der Sprache zu bekommen. (Verfügbar auf Github unter baltermia/study-companion/doku.typ).

3. Durchführung

3.1. Raspberry Pi Installation

Die Installation des Raspberry Pi's wird kurz gehalten. Da der Bot den Raspberry Pi lediglich als Host verwendet, ist die Installation eines Betriebssystem mit einer Grafischen Benutzeroberfläche nicht nötig. Daher habe ich mich entschieden, Debian zu installieren. Debian stellt eigene Images für den Raspberry Pi zur Verfügung und auch direkt ein Online-Tutorial, welches zeigt, wie man das Betriebssystem auf eine SD-Karte schreibt. [1] Da Docker verwendet wird, um den Bot laufen zu lassen, wird dies auch benötigt. Dazu gibt es auch eine offizielle Anleitung für Debian. [2]

3.2. Gewähltes Framework für den Telegram Bot

Durch meine mehrjährige Erfahrung mit dem .NET Framework habe ich mich auch für dieses für den Bot entschieden. Telegram stellt bereits eine eigene Bibliothek zur Verfügung. [3] Ich habe mich aber für eine erweiterte Bibliothek - *Minimal Telegram Bot* - entschieden, da diese viel Boilerplate Code abnimmt und, laut den Entwicklern selbst, „die Entwicklung an die Workflows von .NET anpasst". [4] Im hintergrund verwendet diese Bibliothek aber immer noch die offizielle Telegram Bibliothek.

Die verwendete Telegram-Bibliothek bietet eine `StateMachine` zur Verwaltung von Konversationen und Zuständen. Dadurch kann der Bot parallel mit mehreren Nutzern ohne Konflikte interagieren. Zustände werden standardmässig im Arbeitsspeicher gehalten, können aber auch persistent (z. B. in einer Datenbank) gespeichert werden. Für kleine Anwendungen ist das meist übertrieben, aber beim Testen ist Persistenz nützlich, da im-Memory-Zustände bei jedem Neustart verloren gehen. Deshalb dafür wird Redis verwendet. [5]

Beim erstellen von Hausaufgaben wird ein Datum benötigt. Da es für Nutzer etwas aufwändig ist, immer das volle Datum anzugeben, wird die .NET *Recognizers Text* Bibliothek genutzt, welche es ermöglicht, natürliche Sprache in strukturierte Daten umzuwandeln. Das macht sie, indem sie ein Machine Learning Modell verwendet, welches auf verschiedene Sprachen trainiert wurde und komplett lokal läuft. [6]

3.3. iCal Verarbeitung

Auch für die iCal Verarbeitung gibt es eine geeignete .NET Bibliotheke, nämlich *iCal.NET*. [7]

Erst muss der Kalender heruntergeladen werden, dazu kann direkt die `.ics` -Datei über den .NET-integrierten `HttpClient` heruntergeladen werden. Danach kann die Datei mit der `Calendar.Load` Methode der iCal.NET Bibliothek eingelesen werden.

Da der User den Kalender jederzeit anschauen kann, macht es keinen Sinn, diesen bei jeder Anfrage neu herunterzuladen. Daher wird diese Datei lokal in einer Datenbank gespeichert. Ich habe mich für eine Postgres Datenbank entschieden, es kann aber dank der Nutzung des .NET Entity Frameworks auch einfach auf eine andere Datenbank gewechselt werden. Das Entity Framework ist eine Art Repository-Pattern, welches die Datenbankzugriffe abstrahiert und es so ermöglicht, mit verschiedenen Datenbanken zu arbeiten, ohne den Code anpassen zu müssen. [8]

3.4. AI Integration

Der Bot muss keine komplexen Probleme mit AI Lösen, sondern lediglich Zusammenfassungen generieren. Dazu gibt es ja heutzutage schon eine Vielzahl an vortrainierten Modelle zur Verfügung, welche dies ermöglichen. [9] Ich habe mich für die Verwendung der OpenAI API entschieden, da diese aus Erfahrung einfach zu verwenden ist und gute Resultate liefert.

3.5. Dockerization

Um den Bot möglichst einfach auf einem Raspberry Pi installieren zu können, habe ich mich entschieden, den Bot in einem Docker Container laufen zu lassen. Docker ermöglicht es, Anwendungen in Containern zu verpacken, welche alle Abhängigkeiten enthalten und somit auf jedem System mit Docker-Unterstützung laufen können. [10]

Docker hat ein Erweitertes Tool namens Docker Compose, welches es ermöglicht, mehrere Container zu orchestrieren. Einerseits ist es nützlich, da der Bot mehrere Abhängigkeiten hat (Postgres, Redis), anderseits ermöglicht es auch eine einfache Konfiguration der Container (welche auch gespeichert werden kann). [11]

Bereits für Testing habe ich ein Docker-Compose erstellt, das Postgres und Redis enthält. Für die Produktion gibt es ein weiteres Docker-Compose, das zusätzlich den Bot-Container enthält.

3.5.1. Image

Für das laufen lassen von Containern wird entweder ein `Dockerfile` mit dem Source-Code benötigt, oder ein bereits gebautes `Image` . Um den Bot so einfach wie möglich installieren zu können, macht es Sinn, ein Image des Bots zu erstellen und dieses auf einem Container Repository wie Docker Hub zu speichern. [12]

Ich wollte den ganzen Prozess aber automatisieren, daher habe ich mich entschieden, GitHub Actions zu verwenden, um bei jedem Push auf den `main` Branch ein neues Image zu bauen und dieses auf das GitHub Container Repository zu pushen. GitHub stellt dafür eigene CI/CD Workflows sowie auch ein eigenes Container Registry zur Verfügung. [13], [14]

Die erstellen Images können im Repository abgerufen werden.

4. Resultate

4.1. Bot Installation

Das Installieren des Bots auf einen Raspberry Pi ist so einfach wie geplant. Es gibt nur 3 Vorgaben:

1. Docker und Docker Compose müssen installiert sein.
2. Ein Telegram Bot Token wird benötigt (kann über den BotFather auf Telegram erstellt werden [15]).
3. Ein OpenAI API Key wird benötigt [16]

Danach muss nur das Docker-Compose File angepasst werden, um die beiden Keys einzutragen. Danach kann der Bot mit den folgenden Befehlen gestartet werden:

```shell
docker compose pull # zieht das neuste Image vom Repository
docker compose up -d # startet den Bot im Hintergrund
```

Die gleichen Kommandos können für eine Update des Bots verwendet werden. Denn das `pull` Kommando zieht immer das neuste Image vom Repository.

Die genauen Installationsschritte sind im Repository-Readme nochmals besser und detaillierter beschrieben.

4.2. Bot Funktionalitäten

Detaillierte Bilder der Funktionalitäten vom Bot sind im Repository README dokumentiert.

4.2.1. Start

Der Bot bietet eine einfache Navigation über Buttons die jederzeit angezeigt werden:

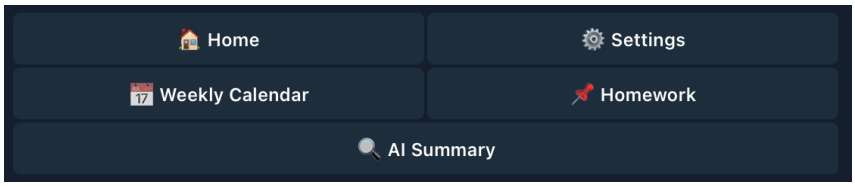

Abbildung 1 — Telegram Bot Navigation durch Buttons

Ganz am Anfang fragt der Bot nach dem iCal Link, damit der Stundenplan eingelesen werden kann. Danach wird täglich um (standardmässig) 8 Uhr morgens eine Zusammenfassung der anstehenden Lektionen und Hausaufgaben geschickt.

4.2.2. Einstellungen

In den Einstellungen kann der User die Sprache zwischen Deutsch und Englisch wechseln, sowie die Zeitzone anpassen.

4.2.3. Hausaufgaben

Hausaufgaben können einfach hinzugefügt, aufgelistet und gelöscht werden. Zudem wird der Nutzer am vorherigen Tag um 12 Uhr an die anstehenden Hausaufgaben erinnert. Erledigte Hausaufgaben werden automatisch aus der Liste gelöscht und bis zum Tagesende noch als erledigt angezeigt.

4.2.4. Kalender

Der Kalender, welcher über den iCal Link eingelesen wird, kann jederzeit abgerufen werden. Der Bot zeigt die nächsten 7 Tage an und listet alle Lektionen auf, inklusive Raum und Dozent. Über Buttons kann man im Wochentakt vor- und zurückspringen. Der Bot erinnert an Lektionen (standardmässig) 1 Stunde vor beginn.

4.2.5. AI Zusammenfassungen

Neben der automatischen täglichen Zusammenfassung kann der User auch jederzeit eine Zusammenfassung der nächsten Tage anfordern.

5. Beantwortung der Fragestellung

1. Welche Frameworks und Bibliotheken eignen sich für die Entwicklung des Bots?
Es eignen sich verschiedene .NET Bibliotheken, wie z.B. *Minimal Telegram Bot* für die Bot-Entwicklung, *iCal.NET* für die iCal Verarbeitung, *Recognizers Text* für die Datumserkennung und das *Entity Framework* für die Datenbankzugriffe.

2. Wie kann der Bot den iCal-Kalender effizient einlesen und verarbeiten?
Der Bot kann den iCal-Kalender effizient einlesen, indem er die `.ics` -Datei herunterlädt und mit der `Calendar.Load` Methode der iCal.NET Bibliothek verarbeitet. Die Daten können dann in einer Datenbank gespeichert werden, um schnelle Zugriffe zu ermöglichen und wiederholtes Herunterladen zu vermeiden.

3. Wie kann die Installation auf einem Raspberry Pi möglichst einfach gestaltet werden?
Die Installation kann durch die Verwendung von Docker und Docker Compose vereinfacht werden, da der Bot und seine Abhängigkeiten in Containern verpackt werden können.

6. Fazit

Das Projekt hat gezeigt, wie es möglich ist, einen Telegram Bot zu erstellen, der Studierenden durch automatische Erinnerungen und Zusammenfassungen den Alltag erleichtern sollte. Die Verwendung von .NET Bibliotheken und Docker hat die Entwicklung und Installation des Bots vereinfacht.

Das erstellen des Bots ist mir leicht gefallen, da ich bereits Erfahrung mit .NET und Telegram Bots hatte. Die grösste Herausforderung war definitiv das erstellen der Docker Compose und das Einrichten der CI/CD Pipeline mit GitHub Actions für das erstellen des Images. Schlussendlich hat aber alles gut funktioniert und ich bin sehr zufrieden mit dem Resultat.

Die Dokumentation hat mir auch Schwierigkeiten bereitet, denn ich wäre gerne noch etwas detaillierter auf die einzelnen Implementationen und Resultate eingegangen. Leider sind wir aber auf eine vordefinierte Seitenanzahl limitiert, welche ich jetzt bereits etwas überschritten habe.

Der Bot liefert noch viel Potenzial für Erweiterung - z.B. sollten AI-Anbieter direkt im Compose-File konfigurierbar sein, statt aktuell nur OpenAI zu unterstützen. Ich hoffe, dass ich in zukünftigen Projekten die Möglichkeit haben werde, weitere Features hinzuzufügen.

Bibliografie

[1] Debian, „RaspberryPiImages“. [Online]. Verfügbar unter: https://wiki.debian.org/RaspberryPiImages

[2] Docker, „Install Docker Engine on Debian“. [Online]. Verfügbar unter: https://docs.docker.com/engine/install/debian/

[3] TelegramBots,.*NET Client for Telegram Bot API*. [Online]. Verfügbar unter: https://github.com/TelegramBots/Telegram.Bot

[4] Pavel Kulakov, *Minimal Telegram Bot*. [Online]. Verfügbar unter: https://github.com/k-paul-acct/minimal-telegram-bot

[5] Redis, „Introduction to Redis“. [Online]. Verfügbar unter: https://redis.io/about/

[6] Wenhao Huang and Zijia Lin and Chris McConnell and Borje F. Karlsson, *Recognizers-Text*. Zenodo. [Online]. Verfügbar unter: https://doi.org/10.5281/zenodo.6860598

[7] ical-org, *iCal.NET Repository*. [Online]. Verfügbar unter: https://github.com/ical-org/ical.net

[8] CodeMaze, „Using Multiple Databases in ASP.NET Core via Entity Framework Core“. [Online]. Verfügbar unter: https://code-maze.com/aspnetcore-multiple-databases-efcore/

[9] Xtra Computing Group, *Benchmark of Small Language Models for News Summarization*. [Online]. Verfügbar unter: https://github.com/Xtra-Computing/SLM_Summary_Benchmark

[10] Docker, „What is Docker?“. [Online]. Verfügbar unter: https://docs.docker.com/get-started/docker-overview/

[11] Docker, „Docker Compose“. [Online]. Verfügbar unter: https://docs.docker.com/compose/

[12] Geeks for Geeks, „What is Docker Hub?“. [Online]. Verfügbar unter: https://www.geeksforgeeks.org/devops/what-is-docker-hub/

[13] Github, „Creating a Docker container action“. [Online]. Verfügbar unter: https://docs.github.com/en/actions/tutorials/use-containerized-services/create-a-docker-container-action

[14] Github, „Working with the Container registry“. [Online]. Verfügbar unter: https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry

[15] Telegram, „Obtain Your Bot Token“. [Online]. Verfügbar unter: https://core.telegram.org/bots/tutorial#obtain-your-bot-token

[16] OpenAI, „ApiKeys“. [Online]. Verfügbar unter: https://platform.openai.com/api-keys