Projektantrag Programmierprojekt

Organizr-API: Entwicklung einer zentralen API für das Management persönlicher Notizen, Tasks und Kalendereinträgen

Gruppenmitglieder

- Julian Beyer, 26b (Einzelarbeit)

Erweiterte Fachspezifikation

Das Ziel des Programmierprojektes «Organizr-API» ist es, eine zentrale API zum Verwalten persönlicher Organisationsdaten. Die API soll ein zentraler Hub für Kalendereinträge, Tasks und Notizen sein. Zudem soll zu Demonstrationszwecken ein Client für die API in Form eines AI-Chatbots mit Function-Calling entwickelt werden.

- Benutzerverwaltung
    - Administrator mit allen Rechten (CRUD bei allen Usern, neue User erstellen)
    - Support mehrerer Enduser mit jeweils einmaliger User-ID
    - Authentifizierung per User-ID und API-Key
    - User hat Rechte über seine eigenen Daten, nicht auf andere
- Kalenderverwaltung
    - Create: Erstellen von Kalendereinträgen mit Titel, optional Beschreibung, Startdatum, Enddatum, optional Wiederholungsregeln nach RRULE Format von iCalendar Standard, optional Tags
    - Read: Abrufen und durchsuchen von Einträgen, Suche nach Text, Zeitspanne und/oder Tags
    - Update: Aktualisieren einzelner Felder eines Eintrags
    - Delete: Löschen von Einträgen
- Taskverwaltung
    - Create: Erstellen von Tasks mit Titel, optional Beschreibung, Status, optional Fälligkeitsdatum, optional RRULE, optional Tags
    - Read: Abrufen und durchsuchen von Tasks, Suche nach Text, Status, Fälligkeitsdatum und/oder Tags
    - Update: Aktualisieren einzelner Felder einer Task
    - Delete: Löschen von Tasks
- Notizverwaltung
    - Create: Erstellen textbasierter Notizen mit Titel, Inhalt, optional Tags
    - Read: Abrufen und Durchsuchen von Notizen, Suche nach Text und/oder Tags
    - Update: Aktualisieren einzelner Felder einer Notiz
    - Delete: Löschen von Notizen
- Technisches der API
    - Backend: Python mit FastAPI-Framework
    - Datenbank: MySQL
    - Deployment: Containerisierung mit Docker, Deployment über Docker-Compose
- Chatbot
    - Nutzung der freien Telegram API in Python

- Der Endnutzer soll die API indirekt per natürlicher Sprache bedienen können; ein LLM kommuniziert mit dem Nutzer und nutzt die API im Hintergrund per Function-Calling
- Begrenzt viel Zeit, nur Demonstrationszwecke, Umfang der unterstützten API-Funktionen voraussichtlich auf Notizen beschränkt
- Nutzung freier LLM-API von OpenAI, Mistral oder Alibaba Cloud (Qwen)

Zeitplan

Phase	Teilaufgaben	Zeit	Zwischenziel	Tests
Basis-Setup, User-API	Einrichtung GitHub-Repo, Projektstruktur, DB-Schema, User API Endpunkte	1 DL	Repository und Projektstruktur, Admin User kann erstellt werden, User können sich authentifizieren	Manuell (Insomnia API Client, Swagger UI)
Kalender API (Kern)	CRUD-Endpunkte (ohne RRULE), simple Querying-Implementation	1 DL	Einfache CRUD-Operationen funktionieren	Manuell (Insomnia API Client, Swagger UI)
Kalender API (Erweitert)	Implementierung RRULE-Logik, erweitertes Querying	1 DL	Wiederkehrende Termine funktionieren, Querying unterstützt auch komplexe Abfragen	Manuell (Insomnia API Client, Swagger UI)
Task und Notizen API	Vollständige CRUD-Endpunkte für Tasks und Notizen ähnlich vorherigem Code	1 DL	Task- und Notizenverwaltung funktionsfähig	Manuell (Insomnia API Client, Swagger UI)
Telegram Bot (Kern)	Telegram-Anbindung vorbereiten, Anbindung LLM an einen Teil der API (Functions für LLM schreiben)	1 DL	Natürliche Kommunikation gelingt einigermassen, Bot kann ungefähr mit API interagieren	Manuell (VSCode), natürliche Sprache (Telegram-Chat)
Abschluss	Verfeinern der Bot-Parameter, letztes reinigen und debuggen des Codes, Präsentation vorbereiten	1 DL	Stabiles funktionsfähiges Produkt	Manuell (VSCode, Insomnia API Client, Swagger UI)

Teststragegie

Testing mithilfe von Unit-Tests ist schwer, da der Python-Code einen MySQL-Server benötigt und primär per API bedienbar ist. Somit werde ich während der Entwicklung jeweils eine lokale Testumgebung haben, die ich mithilfe der App «Insomnia» und der in FastAPI integrierten Swagger UI testen kann. Zudem werde ich auf einem Linux-Server eine vollständige Umgebung des Projektes laufen lassen (Docker-Compose mit API und MySQL, Reverse Proxy mit TLS).

Vorwissen

Der Umfang des Projektes ist gross, jedoch habe ich in allen relevanten Bereichen schon eigene Erfahrungen gesammelt und ähnliche Projekte umgesetzt, so dass ich auf Erfahrung und evtl. ähnlichen Code zurückgreifen kann, was die Entwicklung beschleunigt. Diese relevanten Bereiche beinhalten: FastAPI, Testing mit Insomnia und Swagger UI, MySQL, Docker und Docker-Compose, Telegram-Chatbot mit Function-Calling in Python.