IU Internationale Hochschule

Studiengang: Informatik M.Sc.

Phase 1 und Phase 2: Projektdokumentation, Anforderungsdokument, Spezifikationsdokument, Architekturdokumen

Portfolio zur Prüfung im Kurs Projekt: Software Engineering (DLMCSPSE01_D)

eingereicht von: Kevin Walter

Matrikel-Nr.: 92212082

Tutor: Prof. Dr.-Ing. David Kuhlen

Datum: 03.07.2025

Inhaltsverzeichnis

Phase 1: Konzeptionsphase – Projektdokument

Link zum GitHub Repository: https://github.com/KTWIU/SEProject

1. Projektidee und angestrebtes Ergebnis

Bevor der Zweck dieser Anwendung genannt wird, möchte ich ein Problem aus meinem Alltag nennen. Ich plane meinen Tag bisher noch auf einem klassischen Notizblock, was auf lange Sicht nicht nur unpraktisch, sondern auch ressourcenverschwendend ist. Zwar gibt es die „Erinnerungen"-App für das iPhone, jedoch fehlt mir eine einfache und schlanke Anwendung für den PC, die mich in meinem Alltag unterstützt. Die gängigen digitalen Tools sind häufig entweder zu kompliziert oder mit unnötigen Zusatzfunktionen überladen.

Die Zielsetzung dieses Portfolios ist es daher, eine schlanke Desktop-Anwendung für Windows-PCs zu entwickeln, mit der man seine Aufgaben für den nächsten Tag (oder auch die gesamte Woche) möglichst effizient und übersichtlich aufschreiben kann. Die Bedienung soll dabei so einfach wie möglich gehalten werden, also ohne unnötig viele Menüs oder Reiter, sondern mit einem klaren Fokus auf tägliche Aufgabenplanung, Deadlines und einer Kalenderansicht.

2. Potenzielle Risiken und Gegenmaßnahmen

2.1 Skala:

Wahrscheinlichkeit	niedrig	mittel	hoch
Schadensausmaß	niedrig	mittel	hoch

2.2 Tabelle potenzielle Risiken und Gegenmaßnahmen

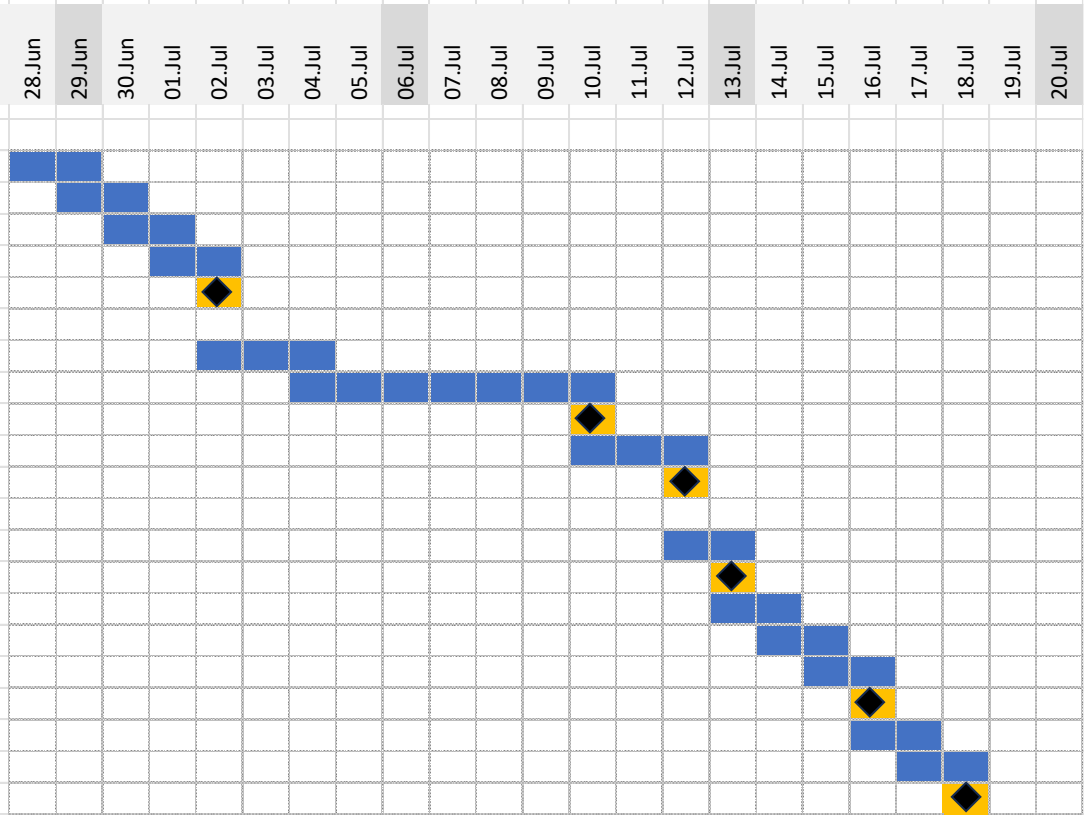
Risiko	Eintrittswahr-scheinlichkeit	Schadens-ausmaß	Gegenmaßnahme
Einarbeitung in Qt und CMake dauert zu lange	mittel	mittel	Früh mit Einarbeitung beginnen, Beispiele als Orientierung verwenden
Release-Package funktioniert nicht auf anderen Rechnern	hoch	hoch	windeployqt.exe verwenden, damit alle DLLs und Dateien vorhanden sind
Projektumfang zu groß / Zeitmangel	gering	mittel	Realistischer Zeitplan, Fokus auf Kernfunktionen (sh. Abschnitt 1)
Fehler durch Pointer / Referenzen	mittel	hoch	Einfacher Code, Debugging-Tools und Testing
Codeverlust / verschiedene Versionen	gering	hoch	Regelmäßige Pushs zu GitHub

3. Zeitplanung

Projektplan

	Projektname	Projekt SE
	Projektleiter	Kevin Walter
	Projektstart	28.06.2025

AP-Nr. & Meilenstein	Bezeichnung	Start	Tage	Ende
	Phase 1: Konzeptionsphase			
1,1	Projektdokumentation erstellen	28.06.2025	1	29.06.2025
1,2	Anforderungsdokumente erstellen	29.06.2025	1	30.06.2025
1,3	Spezifikationsdokumente erstellen	30.06.2025	1	01.07.2025
1,4	Kurzzusammenfassung für Pebble	01.07.2025	1	02.07.2025
M1	Konzeptionsphase abgeschlossen	02.07.2025		02.07.2025
	Phase 2: Erarbeitungsphase/Reflexionsphase			
2,1	Architekturdokument	02.07.2025	2	04.07.2025
2,2	Implementierung	04.07.2025	6	10.07.2025
M2	Implementierung abgeschlossen	10.07.2025		10.07.2025
2,3	GitHub-Integration	10.07.2025	2	12.07.2025
M3	Erarbeitungsphase abgeschlossen	12.07.2025		12.07.2025
	Phase 3: Finalisierungsphase			
3,1	Unit-Tests	12.07.2025	2	13.07.2025
M4	Alle Tests erfolgreich abgeschlossen	13.07.2025		13.07.2025
3,2	Testdokument	13.07.2025	1	14.07.2025
3,3	Benutzeranleitung	14.07.2025	1	15.07.2025
3,4	Abstract	15.07.2025	1	16.07.2025
M5	Dokumentation abgeschlossen	16.07.2025		16.07.2025
3,5	ZIP-Export	16.07.2025	1	17.07.2025
3,6	Finale Abgabe / alles noch einmal prüfen	17.07.2025	1	18.07.2025
M6	Projekt abgeschlossen / Finale abgabe	18.07.2025		18.07.2025



Phase 1: Konzeptionsphase - Anforderungsdokument

4. Stakeholder (Ziel- und Benutzergruppe)

Die Anwendung richtet sich an alle Nutzer, die ihren Tag gerne strukturiert planen möchten, unabhängig davon ob sie Studenten, Berufstätige oder Privatpersonen sind. Das Programm bietet einen schnelle und unkomplizierten digitalen Ersatz für einen analogen Notizblock und unterstützt dabei, einen Überblick über anstehende Aufgaben und Deadlines zu behalten. Das Ziel ist es, die Vorteile moderner Aufgabenverwaltung am PC mit der Einfachheit eines Papier-Notizblocks zu verbinden, um papierlos und effizienter planen zu können.

5. Funktionale Anforderungen

5.1 Funktionsliste

- Aufgaben anlegen, bearbeiten, löschen
- Deadlines setzen
- Aufgaben als erledigt markieren
- Kalender-/Tagesansicht
- Daten speichern/laden

5.2 User Stories

Als Benutzer möchte ich einfach per Texteingabe mit der Tastatur Aufgaben anlegen, bearbeiten und löschen können.

Als Benutzer möchte für meine Aufgabe auch ein Fälligkeitsdatum setzen können, damit ich mir nicht selbst notieren muss, bis wann ich die Aufgabe erledigen will. Es ist vorerst nicht vorgesehen, eine Benachrichtigung zu versenden, dass die Aufgabe überfällig ist. Über das heutige Datum und dem notierten Fälligkeitsdatum kann ich mir das selbst herleiten.

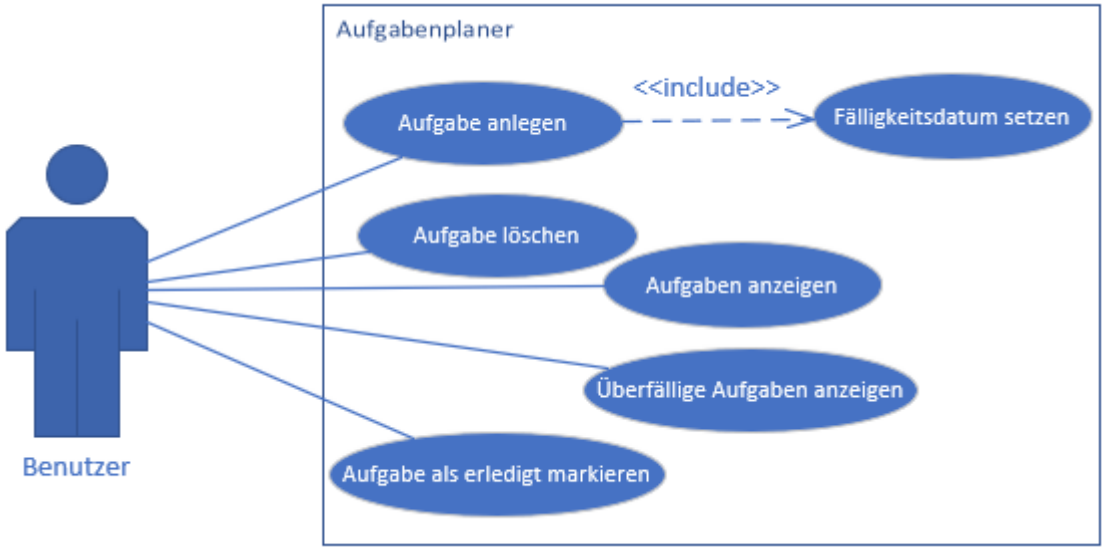
Als Benutzer möchte ich die Aufgabe als erledigt markieren können, damit ich direkt sehen kann, welche Aufgaben ich heute noch vor habe.

Als Benutzer möchte ich eine Kalender- und Tagesansicht sehen können, damit ich so beispielsweise auch eine ganze Woche planen kann.

Als Benutzer möchte ich die Daten speichern können, das heißt entweder per „save“ Button, oder mit automatischer Speicherung.

Als Benutzer muss ich mich nicht anmelden oder registrieren. Die Anwendung funktioniert wie ein klassisches Notizbuch: Ich kann direkt starten, ohne Account oder Internetverbindung. Die Aufgaben werden lokal auf meinem Windows-PC gespeichert. Eine Synchronisierung mit einer Cloud ist nicht vorgesehen.

Abbildung 1: Use-Case-Diagramm



Quelle: Eigene Darstellung mit Hilfe von Visio

6. Nicht-funktionale Anforderungen

- **Benutzerfreundlichkeit:** Die Anwendung soll eine einfache Bedienung und eine übersichtliche Oberfläche besitzen
- **Performance:** Die Anwendung ist schlank und überzeugt durch schnelle Ladezeiten, z. B. keine langen Ladezeiten für Reiter oder Seiten
- **Datensicherheit:** Alle Daten werden lokal auf dem PC gespeichert, es besteht keine Internetverbindung und keine Cloud-Synchronisierung. Dadurch sind die Daten bestmöglich vor externen Angriffen geschützt
- **Verschlüsselung:** Eine Verschlüsselung der Daten ist vorerst nicht vorgesehen
- **Plattform:** Die Anwendung wird als Windows Desktop-Anwendung bereitgestellt und nicht als Web-App
- **Eingabevalidierung:** Es wird eine Eingabevalidierung für Pflichtfelder und sinnvolle Werte umgesetzt

7. Glossar

- **Aufgabe:** Ein Eintrag mit Titel, Beschreibung und Fälligkeitsdatum
- **Tagesansicht:** Auflistung aller an diesem Tag zu erledigenden/eingetragenen Aufgaben
- **Deadline:** Datum, bis zu dem die Aufgabe abgeschlossen sein soll
- **GUI:** Grafische Benutzeroberfläche
- **User Story:** Beschreibung aus Anwendersicht
- **UML:** Unified Modeling Language

8. Datenmodell

8.1 Aufgabe

- Es ist das wichtigste Objekt
- Attribute einer Aufgabe sind: Titel, Beschreibung, Fälligkeitsdatum, Status (offen/erledigt) mit Anzeigen in grün/rot
- Benötigt mindestens einen Titel (Pflichtfeld), ein optionales Fälligkeitsdatum und einen Status (offen/erledigt), kann zusätzliche Beschreibung haben

8.2 UML-Klassendiagramm

Abbildung 2: UML-Klassendiagramm

Quelle: Eigene Darstellung mit Hilfe von Visio

9. Geschäftsprozesse

Geschäftsprozess 1: Aufgabe anlegen

1. Der Benutzer startet die Anwendung
2. Im Hauptfenster klickt er auf „Aufgabe hinzufügen“
3. Es öffnet sich ein Modal
    - Der Benutzer gibt Titel, Beschreibung, Fälligkeitsdatum ein
    - Mit „Speichern“ wird die Aufgabe übernommen
4. Die neue Aufgabe erschein in der Aufgabenliste

Geschäftsprozess 2: Aufgabe als erledigt markieren

1. Benutzer sieht die Aufgabenliste
2. Bei einer Aufgabe klickt er auf „Aufgabe als erledigt markieren“
3. Die Aufgabe wird als erledigt markiert und ggf. anders dargestellt (je nach Qt Funktionen)

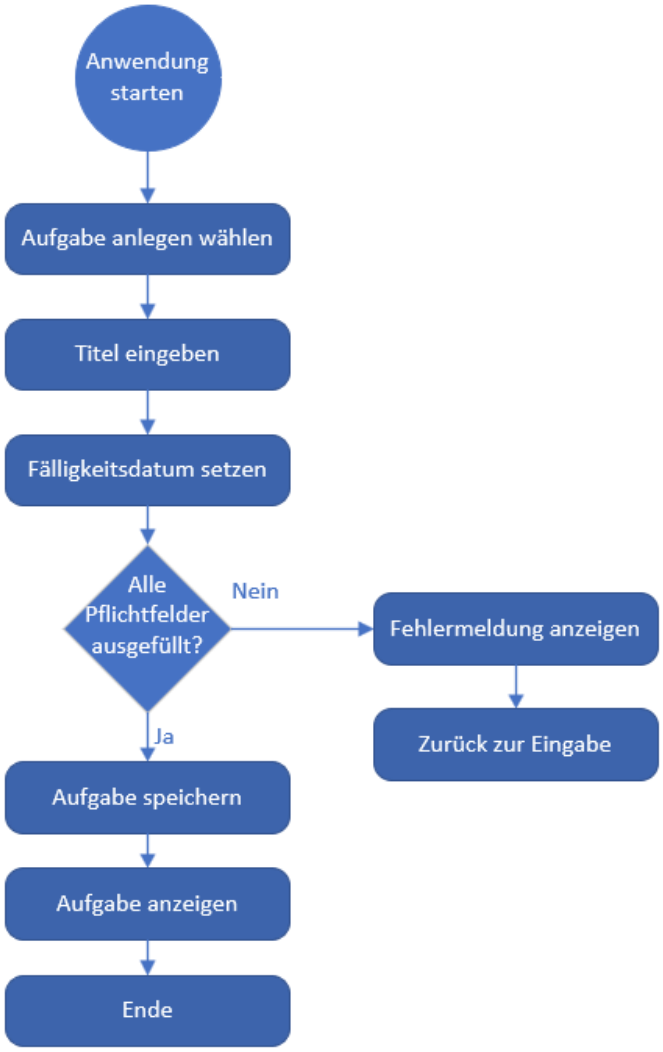
Geschäftsprozess 3: Überfällige Aufgaben anzeigen

1. Beim Start prüft das Programm alle Aufgaben
2. Aufgaben mit Fälligkeitsdatum < wie aktuelles Datum werden als „überfällig“ markiert/angezeigt

Geschäftsprozess 4: Aufgabe löschen

1. In der Aufgabenliste wählt der Nutzer eine Aufgabe aus
2. Mit Klick auf „Aufgabe löschen“ wird die Aufgabe entfernt (komplett gelöscht)

Abbildung 3: UML-Aktivitätsdiagramm für Kernprozess "Aufgabe anlegen"



Quelle: Eigene Darstellung mit Hilfe von Visio

10. Geschäftsregeln

- Eine Aufgabe muss immer mindestens einen Titel besitzen
- Das Fälligkeitsdatum einer Aufgabe darf nicht in der Vergangenheit liegen
- Eine Aufgabe kann erst gespeichert werden, wenn alle Pflichtfelder (Titel) ausgefüllt sind
- Jede Aufgabe ist eindeutig identifizierbar (z.B. durch eine ID)
- Bereits erledigte Aufgaben können nicht bearbeitet werden (nur noch löschen möglich)
- Das Fälligkeitsdatum von bereits erledigten Aufgaben führt nicht mehr zu einem Zustandswechsel (bspw. Verändern der Schrift von grün auf rot o.Ä.)
- Beim Löschen einer Aufgabe öffnet sich ein Modal, um unbeabsichtigtes löschen zu verhindern
- Alle angegebenen Daten werden beim Schließen der Anwendung gespeichert

- Das Fälligkeitsdatum ist optional, aber wenn es gesetzt wird, muss es nach dem Erstellungsdatum liegen

11. Systemschnittstellen

In diesem Projekt sind keine externen technischen Schnittstellen (z.B. HTTP, FTP) vorgesehen. Die Anwendung läuft komplett lokal auf dem Rechner des Benutzers. Der Datenformat der Aufgaben wird zum Beispiel TXT, CSV oder JSON sein.

12. Benutzerschnittstellen

Die Anwendung bietet eine schlanke grafische Benutzeroberfläche (GUI), die für einfache Bedienung und Übersichtlichkeit ausgelegt ist, ohne viele Reiter / Dialoge.

12.1 Struktur der Oberfläche

- Hauptfenster: Zeigt alle Aufgaben übersichtlich als Liste
- Neue Aufgabe anlegen: Button „+ Aufgabe anlegen“ öffnet Dialog
- Aufgabendetails: Im Dialog kann der Nutzer die Attribute eingeben
- Löschen/Erledigt: Neben jeder Aufgabe befinden sich Buttons für „Löschen“ oder „als erledigt markieren“
- Kalender: Der Kalender soll im Hauptfenster angezeigt werden

12.2 Wichtigste Dialoge & Abläufe

Neue Aufgabe anlegen

- Klick auf „+ Aufgabe hinzufügen“
- Eingabefeld für Titel (Pflichtfeld)
- Eingabefeld für Fälligkeitsdatum (optional)
- Eingabefeld für Beschreibung (optional)
- „Speichern“ Button, der nur aktiv ist, wenn das Pflichtfeld ausgefüllt ist
- Fehlermeldung, falls Pflichtfeld leer bleibt oder Datum ungültig ist

Aufgabe löschen

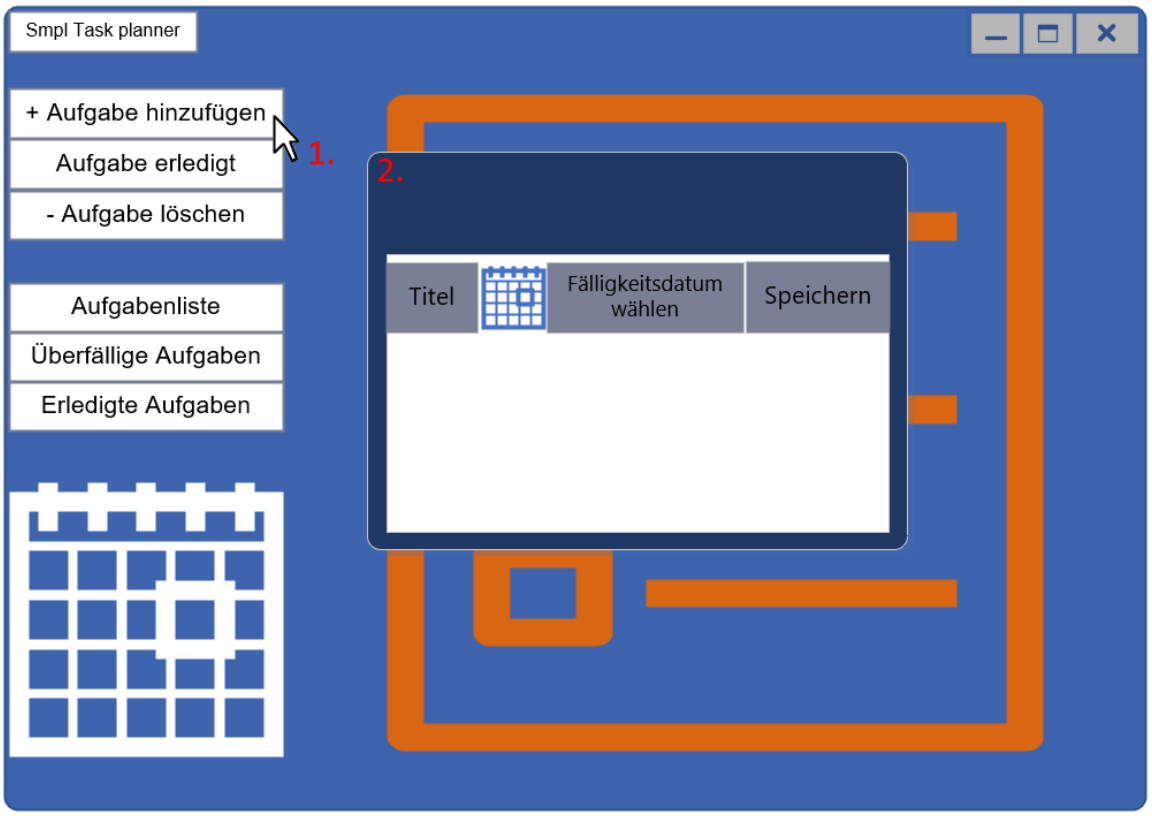
- Klick auf „Löschen“-Button öffnet ein Modal/Bestätigungsdialog und erst danach wir die Aufgabe gelöscht

Eingabevalidierung

- Titel muss ausgefüllt sein
- Fälligkeitsdatum darf nicht in der Vergangenheit liegen

12.3 Skizze der Anwendung

Abbildung 4: Skizze Aufgabenplaner



Quelle: Eigene Darstellung mit Hilfe von Visio

Phase 2: Erarbeitungs- und Reflexionsphase – Architekturdokument

13. Technologieübersicht

13.1 Programmiersprache

Die für dieses Projekt ausgewählte Programmiersprache ist C++, folgend werden die wichtigsten Gründe für die Auswahl erläutert.

- Motivation: Vertiefung der eigenen Kenntnisse, wichtig für Ingenieurs- und Software-Engineering Berufe
- Fördert allgemeines Verständnis für Speicherverwaltung, effiziente Algorithmen und systemnahe Entwicklung
- Große Ökosystem, viele Libraries und plattformübergreifend einsetzbar
- Auch wenn Entwicklung schwerer als bspw. mit Python ist, ist der Lerneffekt für Studium/Beruf höher
- GUI-Anbindung mit Qt

13.2 Frameworks

In diesem Projekt werden primär die Frameworks Qt (Qt6) und googletest verwendet. In diesem Abschnitt werden die Gründe dafür erklärt.

- Qt (Qt6)
    - Wird für grafische Oberfläche (GUI) verwendet
    - Ermöglicht plattformübergreifende Desktop-Entwicklung
    - Gute Unterstützung für UI-Design, Events und eigene Widgets
    - Integration in C++ Projekt und Build-System (CMake) ist Standard
- googletest
    - Framework für automatisierte Unit-Tests
    - Weit verbreitet im C++ Umfeld, auch im industriellen Einsatz
    - Moderne C++ Features für Testfälle und Test Suites
    - Bringt professionelle Entwicklungsmethodik (Test Driven Development, Absicherung von Code…)

13.3 Bibliotheken

Bisher werden im Projekt folgende Bibliotheken verwendet, diesen können sich jedoch je nach Projektablauf noch verändern.

- C++ Standardbibliothek (STL): Für Vektoren, Strings, Maps usw.
- QtCore, QtGui, QtWidgets

13.4 Entwicklungswerkzeuge

- VS Code mit C++ und Qt Extension
- Qt Creator 17.0.0
- CMake als Buildsystem
- GitHub zur Versionskontrolle
- windeployqt zum Erstellen von Release-Paketen (Deployment für Windows) → wird in Phase 3 implementiert

14. Architekturübersicht

In diesem Projekt habe ich mich auf eine Schichtenarchitektur entschieden (3-Layer). Diese ist übersichtlich, leicht erweiterbar und testbar. Für das Projekt ist diese Architektur meiner Meinung nach ausreichend.

- **UI-Schicht:** Besteht aus den Klassen MainWindow und den Dialogklassen
  - Zuständig für grafische Anzeige, Benutzerinteraktion und Aufruf der Logik
- **Logikschicht:** Hauptkomponente ist Klasse TaskManager
  - Aufgaben hinzufügen, löschen, bearbeiten, speichern…
  - Schnittstellt zwischen UI und Datenhaltung
- **Datenhaltung:** durch Klasse Task und Datei-Verwaltung innerhalb TaskManager realisiert
  - Aufgaben werden lokal als .csv gespeichert

15. Struktur

15.1 Hauptkomponenten

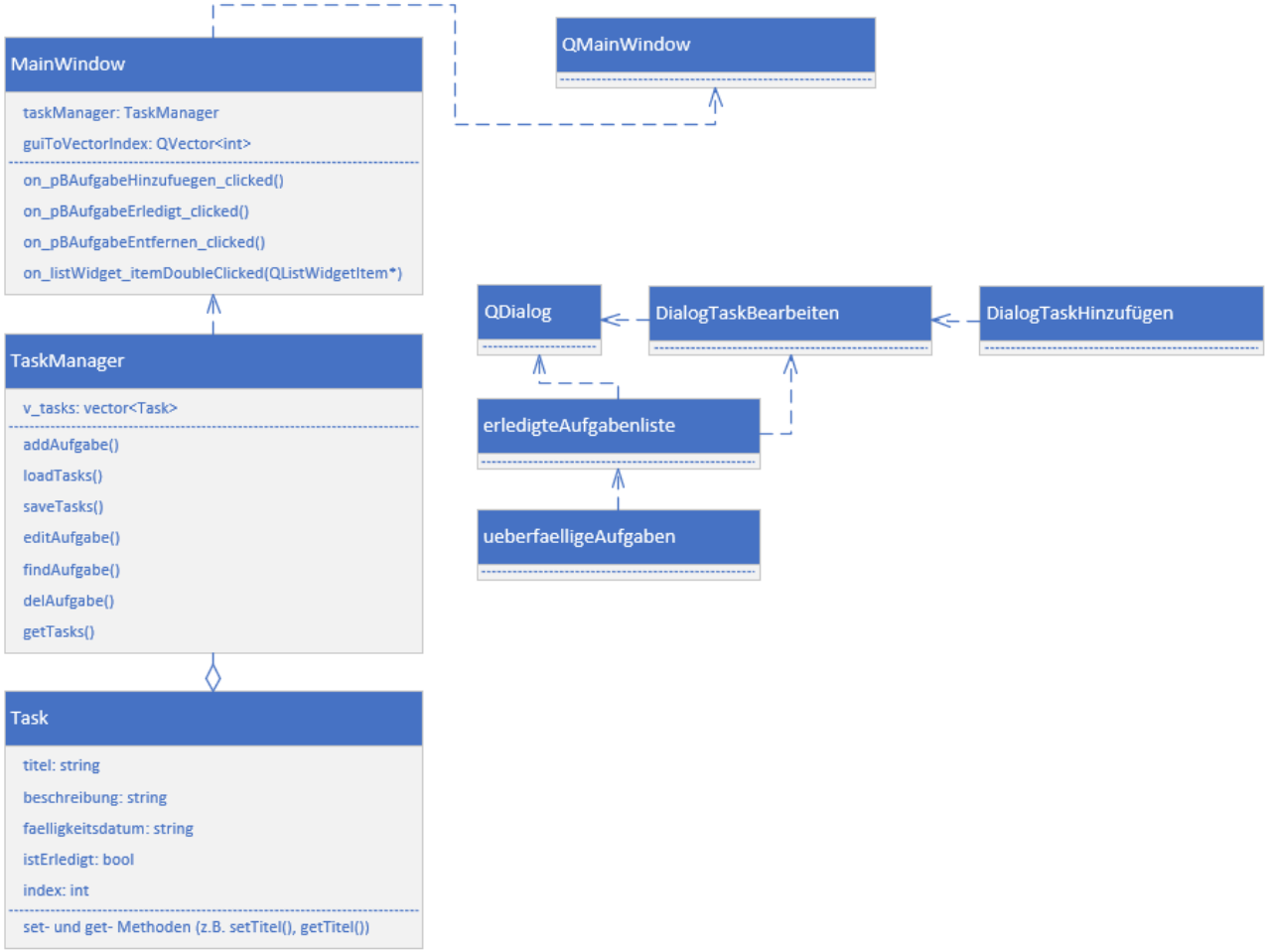
- MainWindow
  - Verantwortlich für das Hauptfenster und die Steuerung der Oberfläche
  - Zeigt die Liste der offenen Aufgaben und bietet Buttons für Aktionen
  - Öffnet bei Bedarf die verschiedenen Dialogfenster
- DialogTaskHinzufuegen / DialogTaskBearbeiten
  - Separate Dialogfenster für das Hinzufügen bzw. Bearbeiten einer Aufgabe
  - Erfassen und Bearbeiten der Aufgabendaten durch den Benutzer
- erledigteAufgabenliste / ueberfaelligeAufgaben
  - Dialogfenster zur Anzeige aller erledigten bzw. überfälligen Aufgaben
- TaskManager
  - Verwalten der Sammlung aller Aufgaben (Hinzufügen, Löschen, Suchen, Bearbeiten)
  - Schnittstelle zwischen UI und Dateispeicherung
  - Verantwortlich für das Speichern und Laden der Aufgaben
- Task

- Datenklasse für einzelne Aufgaben (Attribute: Titel, Beschreibung, Fälligkeitsdatum, Status, Index)
- Methoden zum Ändern und Auslesen der Aufgabendaten

15.2 Abhängigkeiten

- MainWindow nutzt TaskManager für Logik und Aufgabeliste
- TaskManager verwaltet eine Sammlung von Task-Objekten
- MainWindow öffnet die Dialogfenster zur Interaktion mit dem Benutzer
- Dialogfenster übergeben ihre Eingaben an MainWindow/TaskManager

Abbildung 5: UML-Klassendiagramm

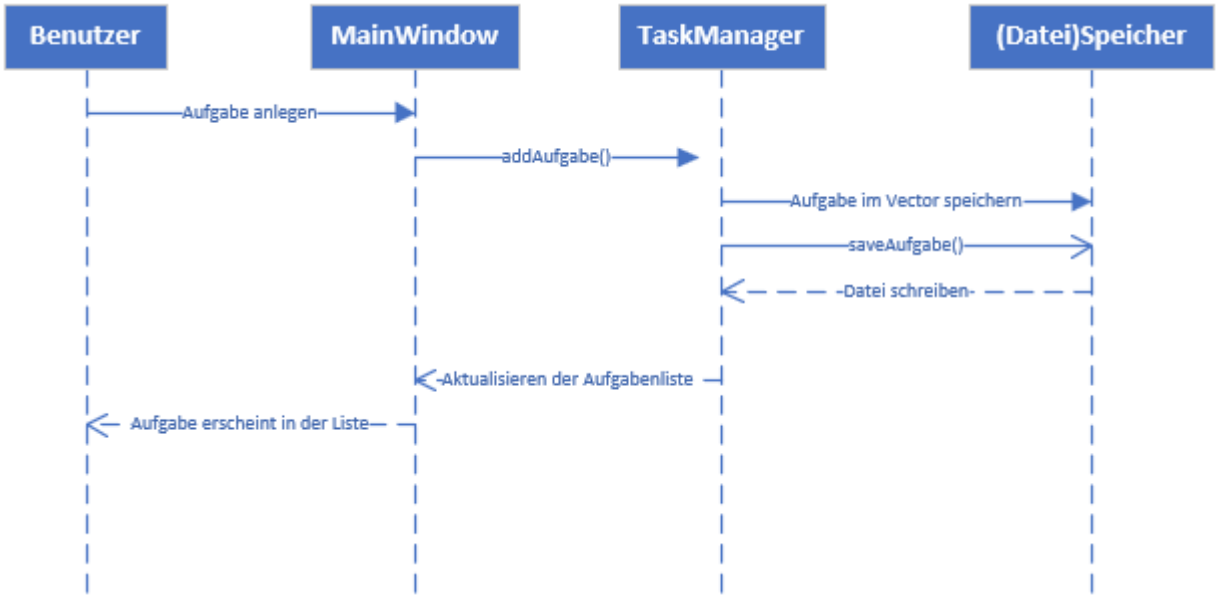


Quelle: Eigene Darstellung mit Hilfe von Visio

16. Verhalten

1. Der Benutzer gibt die Aufgabendaten im Dialog ein
2. MainWindow ruft addAufgabe im TaskManager auf
3. TaskManager legt ein neues Task-Objekt an und speichert alle Aufgaben in der Datei
4. MainWindow aktualisiert die Aufgabenliste in der Benutzeroberfläche

Abbildung 6: UML-Sequenzdiagramm



Quelle: Eigene Darstellung