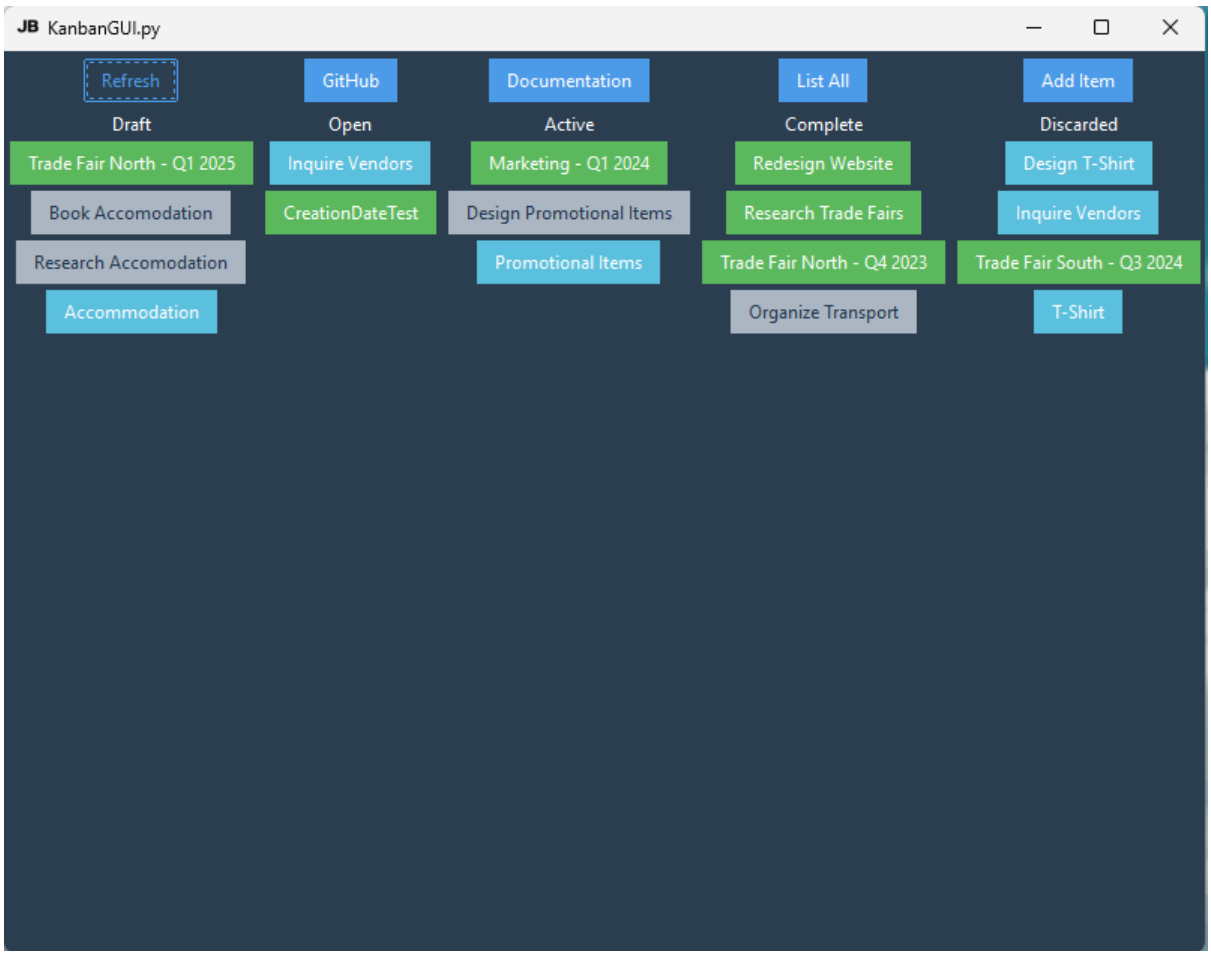
KanbanGUI.py

Kanban-Board in Python anhand BSON-Dokumenten



Jonas Brilz

Fachinformatiker für Anwendungsentwicklung

Klasse E3FS3/2024

Inhaltsverzeichnis

1. Projektziel

Eine Python-Applikation schreiben/aufbauen, mit welcher Objekte der Projektmanagement-Methode Kanban grundlegend verwaltet werden können. Orientierung an Atlassian Jira. Dazu werden MongoDB und Python Tkinter verwendet. Das Projekt dient ebenfalls als Wissensaneignung, da ich bisher keine Erfahrung mit diesen gemacht habe.

2. Pflichtenheft / Anforderungsanalyse

Im Vorfeld wurde folgende Anforderungsanalyse wurde für den schulischen Projektantrag angefertigt:

Projekt “Kanban - Projektmanagementtool”

2.1. Zielsetzung

Projekt unter Verwendung von MongoDB und Python TKinter, da ich bisher keine Erfahrung mit diesen als Komponenten habe. Das Projekt dient somit dem Wissensaufbau.

2.2. Abstrakt

2.2.1. Projektthema

In diesem Projekt werde ich eine Python-applikation schreiben/aufbauen, mit welcher die Projektmanagement Methode Kanban grundlegend verwaltet werden kann.
Ich orientiere mich dabei an Atlassian Jira.

2.2.2. Kanban

Kanban ist eine agile Projektmanagement-Methode, die auf einfachen zuständen von Teilaufgaben von Projekten besteht.

Vorteile:

- Transparenz: Kanban ermöglicht eine transparente Darstellung des Arbeitsflusses. So kann auch ohne Technische Kenntnisse oder tieferes untersuchen die Übersicht zu Aufwandsaufkommen bewahrt werden und so beispielsweise Engpässe abgewendet werden können.
- Flexibilität: Durch die Kapselung einzelner Aufgaben können aufwände aufgeteilt und unabhängig voneinander bearbeitet werden
- Effizienz: Durch die vereinfachte Einsehbarkeit von zeitlichen Aufwänden und Aufgabenstellungen an sich ist eine bessere Planung möglich. Vorgänge lassen sich so einfacher Parallel vollziehen.
- Nachvollziehbarkeit: Durch umfassende Rahmeninformationen und die zeitliche Protokollierung von Tätigkeiten sind detaillierte Informationen auch nach Bearbeitung des Vorgangs einsehbar.

2.2.3. Kanban-Objekttypen:

- Allgemein:
  Durch die unterschiedlichen Objekttypen wird Kanban mehr Struktur verliehen.
  Alle Objekttypen tragen folgende Informationen:
  - Objekttyp: Unterscheidung, ob es sich um Epic. Task oder Subtask handelt.
  - Erfassungszeitpunkt: Zeitstempel, wann dieser Vorgang im System erfasst wurde.
  - Ursprüngliche Aufwandsschätzung: Bevor ein Task bearbeitet werden kann, muss eine Aufwandseinschätzung zur Bearbeitungszeit hinterlegt werden.
  - Aufgewendete Zeit: Tatsächlich aufgewendete Zeit, da die Aufwandsschätzung nur als Schätzung fungiert.
  - Status: Aktueller Status des Task. Information, ob der Vorgang der Vorgang in Planung, bearbeitbar, in Bearbeitung, Abgeschlossen oder sogar verworfen etc. wurde.
  - Vorgangsbeschreibung: genauere Informationen zu Task, dessen Bearbeitungskriterien, Akzeptanzkriterien, Verweise und Aufgabenbeschreibung.
  - Vorgangshistorie: Als Kommentare können Entwicklungen bzw. Aktuelle Tätigkeiten mit einem Zeitstempel zum Task hinterlegt werden. Durch diese Transparenz kann zu jedem Zeitpunkt nachvollzogen werden, wo der Task aktuell steht.

- Epics:
  Ein ‘Epic’ in Kanban kann mehrere Vorgänge beinhalten. Epics dienen zur Sammlung von Vorgängen. In Kanban können mehrere Projekte gemanagt werden. Beispielsweise kann pro Kunde ein Epic bestehen, um Vorgänge so nach deren Themengebiet, Kritikalität oder Aufwand zu ordnen.

- Tasks:
  Ein Task stellt einen Arbeitsvorgang al seine Aufgabe dar. Beispielsweise kann eine Datenmigration oder das Verfassen eines Berichtshefts als Task geführt werden.

- Subtasks:
  Subtasks stellen untergeordnete, atomare Vorgänge eines Vorgangs dar. Subtasks müssen immer einem Task zugeordnet werden. Am Beispiel der Datenbankmigration kann dieses in mindestens 3 Subtasks aufgeteilt werden:
  1. Datenexport alter Datenbank in ein Flatfile
  2. Datenimport in neue Datenbank
  3. Archivierung alter Datenbank

2.3. Abgrenzungen

- Innerhalb dieses Projekts wird einzig die Projektmanagement-Methode Kanban umgesetzt. Nicht wie Atlassian Jira Scrum oder Scrumban.
- Daten werden einzig auf einem noch festzulegenden Hosting-Dienst Online gespeichert. Durch Applikationen findet keine lokale Datenspeicherung statt.

2.4. Verpflichtende Akzeptanzkriterien

- Datenspeicherung mit MongoDB
- Speicherung von Kanban-Objekttypen
- GUI zur Erstellung, lesen und Veränderung von Kanban-Objekttypen

2.5. Unverbindliche Akzeptanzkriterien

- Visualisierung mit einem Kanban-Board
- Berichterstattung: Graphische Darstellung der prozentualen Komplettierung eines Epics
- Lizenz hinzufügen: Das Projekt mit einer Lizenz versehen, beispielsweise
    - MIT-Lizenz oder der
    - GNU General Public License Version 2.0
- Verbindung zu MongoDB per Zertifikat anstatt Zugangsdaten verwalten
- Grafischer Dunkelmodus

2.6. Technologien

2.6.1. Persistence/Datenspeicherung:

MongoDB

2.6.2. GUI-Desktopapplikation

Python-Tkinter

2.7. Zeitplanung

2.7.1. KW03

Analyse und Projektplanung

2.7.2. KW04

Datenmodellierung Kanban-Objekte

2.7.3. KW05-KW07

Implementierung und Testing

2.7.4. KW07-10.03.2024

Dokumentation des Projekts und Präsentationsaufbereitung

2.7.5. 10.03.2024
Abgabe

3. Komponenten

3.1. Python Version 3.12.0

3.2. Entwickungsumgebung JetBrains PyCharm Community Edition 2024.3.4

- Download
  https://www.jetbrains.com/pycharm/download/other.html

3.3. Python Bibliotheken

- Tkinter
  https://docs.python.org/3/library/tkinter.html
  Bibliothek zur graphischen Anzeige. Ermöglicht die Erstellung von grafischen Elementen mit Python.
- TTkbootstrap
  https://ttkbootstrap.readthedocs.io/en/latest/
  https://github.com/israel-dryer/ttkbootstrap/
  Erbt von Tkinter. Lässt grafische Elemente sauberer und Farblich abgestimmt aussehen. Angelehnt an JavaScript-Bibliothek Bootstrap.
- Pymongo
  https://pymongo.readthedocs.io/en/stable/
  https://www.mongodb.com/docs/drivers/pymongo/
  https://www.w3schools.com/python/python_mongodb_getstarted.asp
  Bibliothek zur Verbindung und Arbeiten mit CRUD-Funktionalitäten für MongoDB-BSON Datenbanken.
- enum
  https://docs.python.org/3/howto/enum.html
  Bibliothek zu Enumerationen in Python.
- OS
  https://docs.python.org/3/library/os.html
  Bibliothek für leichtere Interaktionen mit dem Betriebssystem der Applikation.
  Wird zur Ermittlung des Pfades der aktuellen Python-Datei verwendet.
  https://stackoverflow.com/questions/3430372/how-do-i-get-the-full-path-of-the-current-files-directory
- webbrowser
  https://docs.python.org/3/library/webbrowser.html
  Bibliothek für leichtere Interaktionen mit dem Standard-Webbrowser des Betriebssystems der Applikation.
  Wird zum Öffnen von PDF-Dokumenten und URLs verwendet.
  https://stackoverflow.com/questions/4216985/call-to-operating-system-to-open-url

- Datetime
  https://docs.python.org/3/library/datetime.html
  Bibliothek für leichtere Interaktionen mit Zeitstempeln.
  Wird zur Ermittlung des aktuellen Zeitpunkts verwendet.
  https://stackoverflow.com/questions/32490629/getting-todays-date-in-yyyy-mm-dd-in-python

3.4. MongoDB

- MongoDB Atlas
  https://www.mongodb.com/atlas/database
  Speicherung auf einem MongoDB-Atlas Server.
  Der „Basic (Free)“-Plan wurde verwendet.
- MongoDB Community Server
  https://www.mongodb.com/try/download/community
  Selbsthosting eines MongoDB-Servers.
- MongoDB Compass
  https://www.mongodb.com/products/tools/compass
  Client zur einfacheren Interaktion mit MongoDB-Datenbanken.

4. Projektschritte

4.1. Analyse und Projektplanung

- Pflichtenheft wurde am 22.01.2024 abgegeben.

4.2. Datenmodellierung Kanban-Objekte

- JSON/BSON Schema wurde am 21.01.2024 erstmals verwendet:

```
1   _id: ObjectId('65e70fb155a82f1fec2dfc5b')                    ObjectId
2   key : "Design T-Shirt"                                       String
3   type : "Task"                                                String
4   creation : "2024-03-04 00:16:39"                             String
5   estimate : "4h "                                             String
6   time_spent : "0h "                                           String
7   status : "Discarded"                                         String
8   description : "Design T-Shirt for 2024 Q1 Marketing."        String
9   parent : "T-Shirt"                                           String
10  ▸ history : Array (empty)                                    Array
```

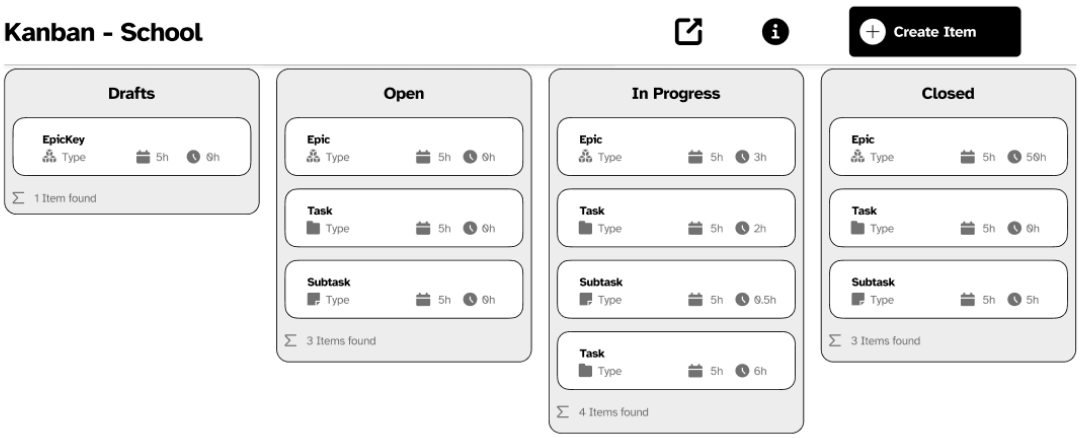
Abbildung 1: BSON-Dokument

4.3. Implementierung und Testing

Implementationen und Applikationstest haben im Verlauf der Kalenderwochen 05-07 stattgefunden.

4.4. Implementierung des GUI

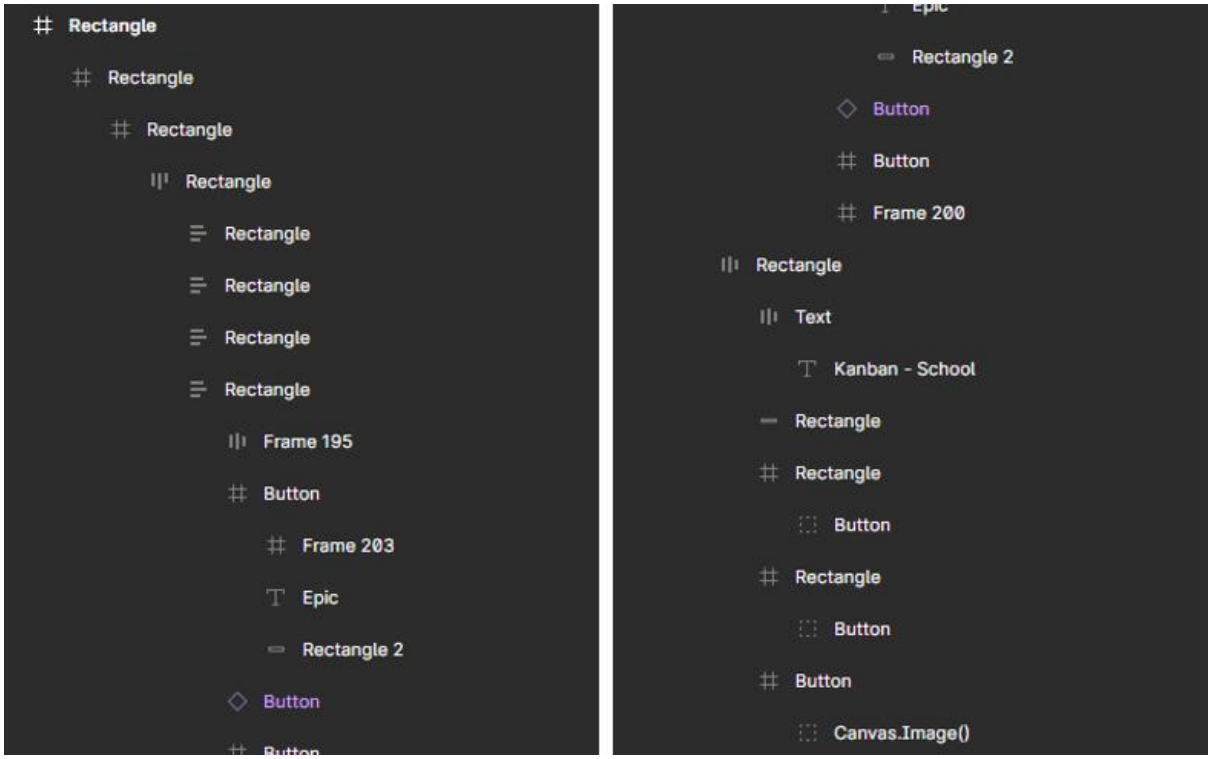
Folgender Entwurf wurde mit Figma per Webanwendung erstellt:



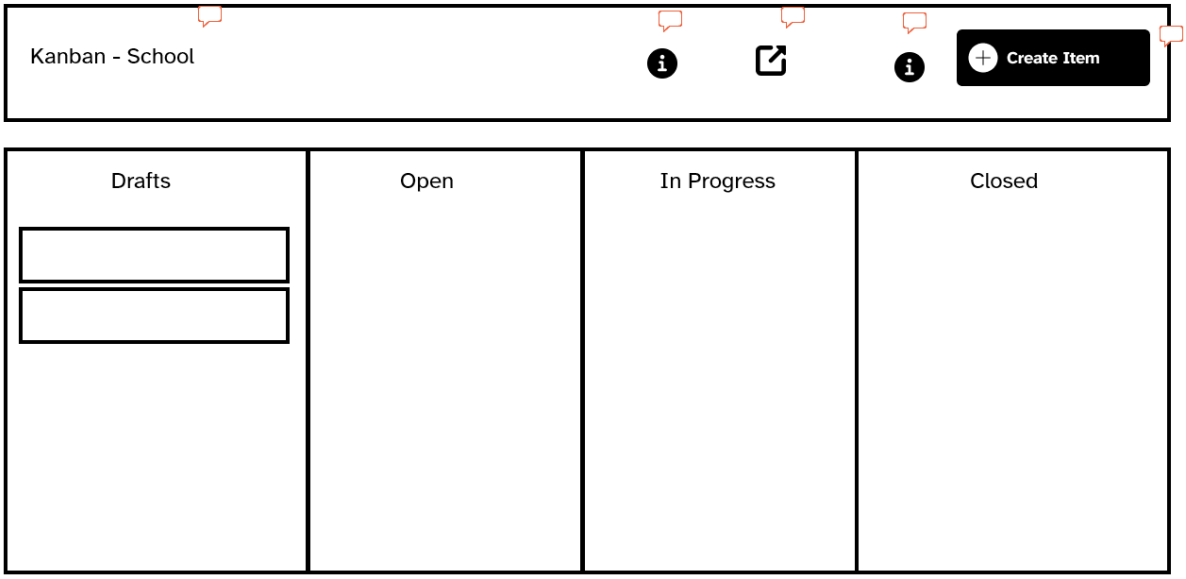
Ursprünglich sollte dieser mit einer Importsoftware direkt aus Figma zu Python importiert werden.

Dieses Importieren war hiermit geplant: https://github.com/ParthJadhav/Tkinter-Designer. Mithilfe eines accountgebundenen API-Tokens kann die Implementation auf Figma-Projekte des verknüpften Accounts zugreifen. Sie kann allerdings keine Objekttypen unterscheiden, weshalb diese Information in den entsprechenden Ebenennamen manuell hinterlegt werden musste.

Nach einem Import in das Python-Projekt konnte also keine Referenz hergestellt werden. Außerdem wurden Ebenen des Figma-Projekts nicht berücksichtigt, weshalb ein Import durch Tkinter-Designer mehrere zusätzliche Aufwände mit sich gebracht hätte.
Die Ebenen des Figma-Projekts trugen nach Anpassung auf die entsprechende Namenskonvention (https://github.com/ParthJadhav/Tkinter-Designer/blob/master/docs/instructions.md#1-reference) folgende Bezeichnungen:

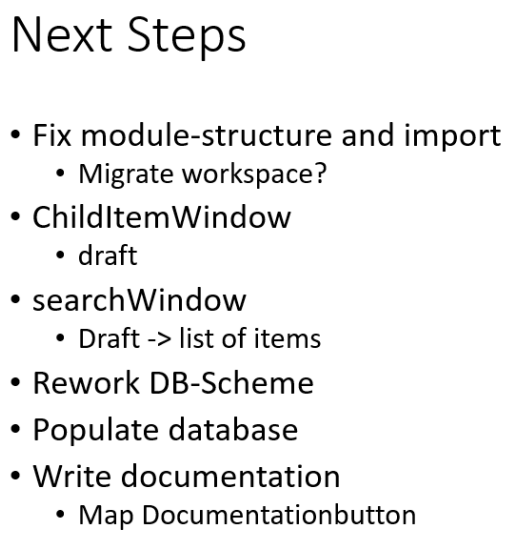


Somit entschied ich mich, das GUI ohne weitere Hilfsmittel zu implementieren.
Folgende Skizze wurde am 27.02.2024 in Microsoft PowerPoint erstellt:

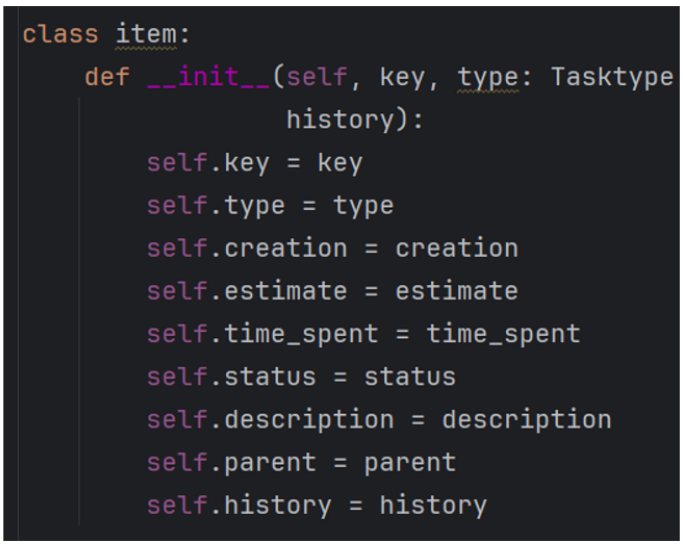


4.5. Implementation der Datenbankoperationen

Nach der initialen Implementierung dieses Entwurfs wurden die nächsten Schritte aufgelistet:



Während das Kanban-Board bereits implementiert wurde und BSON-Dokumente, welche über MongoDB Compass eingefügt wurden, bereits durch die Applikation abgefragt werden, musste das Erstellen, Lesen und Verändern der Kanban-Aufgabentypen noch implementiert werden. Dazu wurde folgende Skizze in PowerPoint erstellt:



```
class item:
    def __init__(self, key, type: Tasktype
                 history):
        self.key = key
        self.type = type
        self.creation = creation
        self.estimate = estimate
        self.time_spent = time_spent
        self.status = status
        self.description = description
        self.parent = parent
        self.history = history
```

- Key – Entry - Read
- Type – Combo - Write
- Creation – Date – Read
- Estimate – Entry - Write
- Time_spent – Entry – Write
- Status – Combo – Write
- Description – Entry –Write
- Parent – Entry - Write
- History – Entry - Write

Die Skizze resultierter in der Implementation der „itemGUI.py“ und „itemUtil.py“ Dateien.

4.6. Tests

Während der Implementation wurden Funktionalitäten getestet. Nach erfolgreichem Test einer neuen Funktion wurde dann das Zusammenspiel der Applikation mit dieser überprüft. So sind mehrere Themen aufgefallen und behoben worden.
Unittests wie beispielsweise mit Pythons Unittest-Bibliothek (https://docs.python.org/3/library/unittest.html) wurden aufgrund von Zeitmangel nicht implementiert.

4.7. Dokumentation des Projekts und Präsentationsaufbereitung

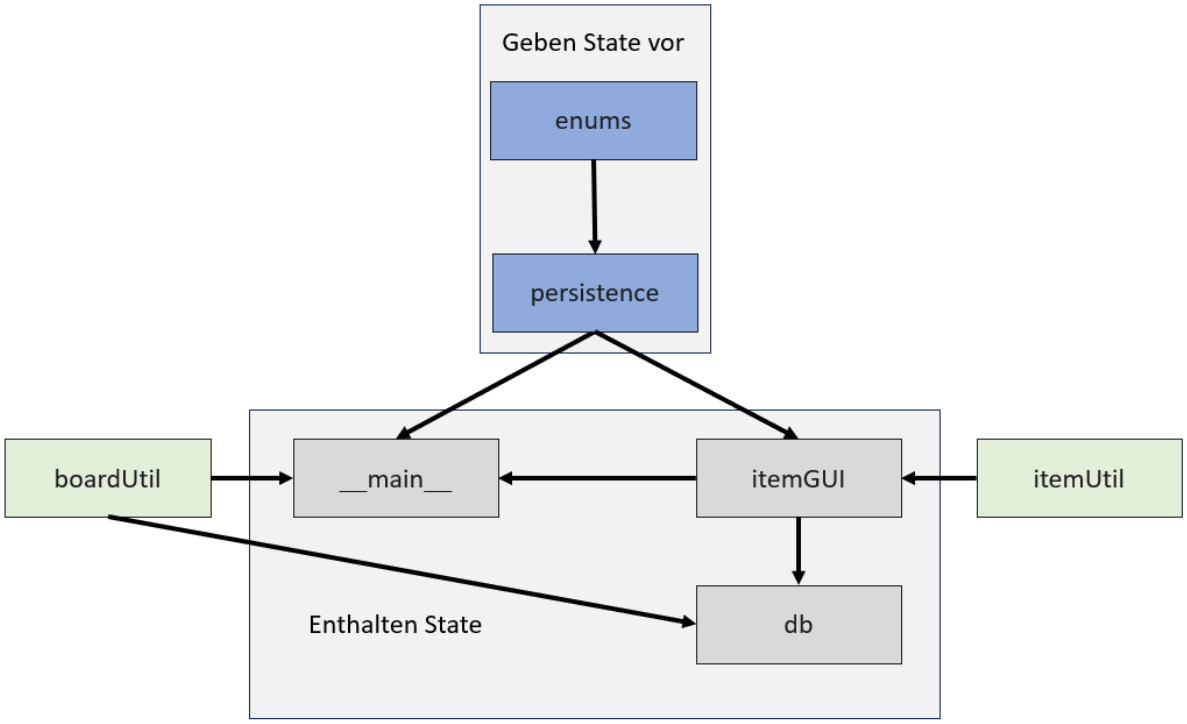
Projektdokumentation wird am 10.03.2024 abgegeben.
Präsentation wird am 19.03.2024 gehalten.

4.8. Abgabe

Projekt wurde am 10.03.2024 abgegeben.

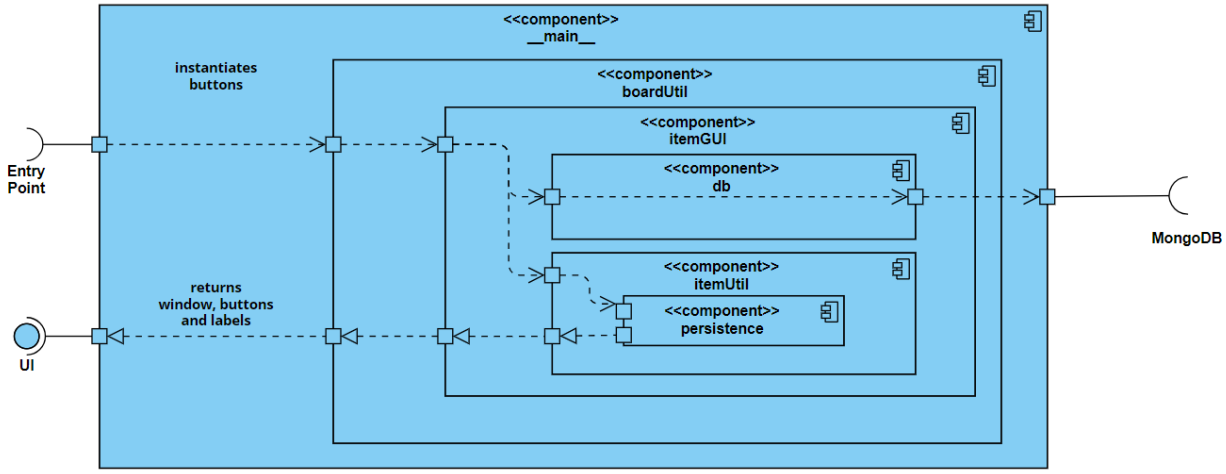
5. Softwareaufbau

5.1. Aufbauskizze

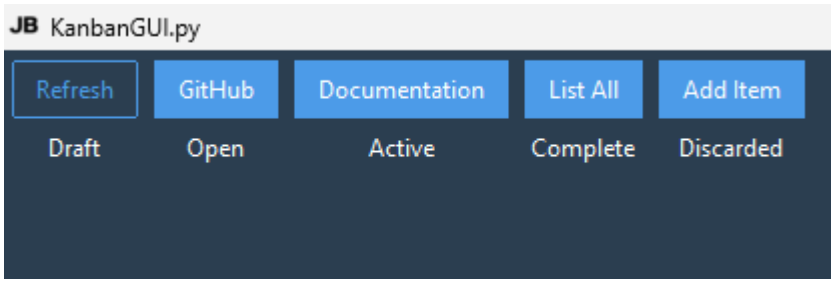


Durch die Enumerationen „Taskstatus“ und „Tasktype“ in der Datei enums gibt die Persistence-Datei mit der Klasse „Item“ den Datentyp des Kanban-Boards vor.

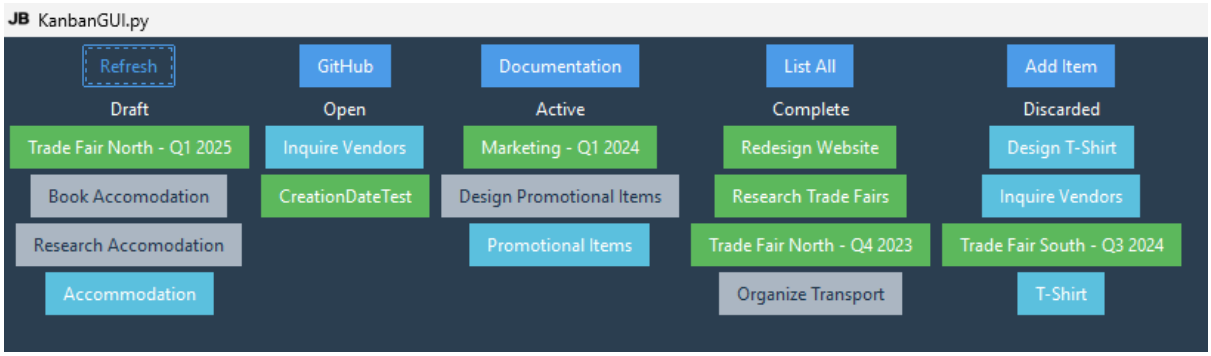
5.2. UML Component Diagram



Die Dateien boardUtil und itemUtil sind als Stateless konzipiert und bilden Funktionalitäten für __main__ (zuerst boardGUI) und itemGUI ab.
Wenn die Applikation in Form von __main__ gestartet wird, sieht das Programm folglich aus:



Alle Buttons werden durch Funktionen aus boardUtil erstellt.
Funktionen, die durch „Refresh“, „GitHub“ und „Documentation“ aufgerufen werden, liegen ebenfalls in boardUtil.
„List All“ und „Add Item“ rufen Funktionen aus itemGUI auf.
Durch Drücken des „Refresh“-Buttons wird mit der db-Datei eine Datenbankverbindung hergestellt und das Kanban-Board bzw. die Applikation mit Items gefüllt:

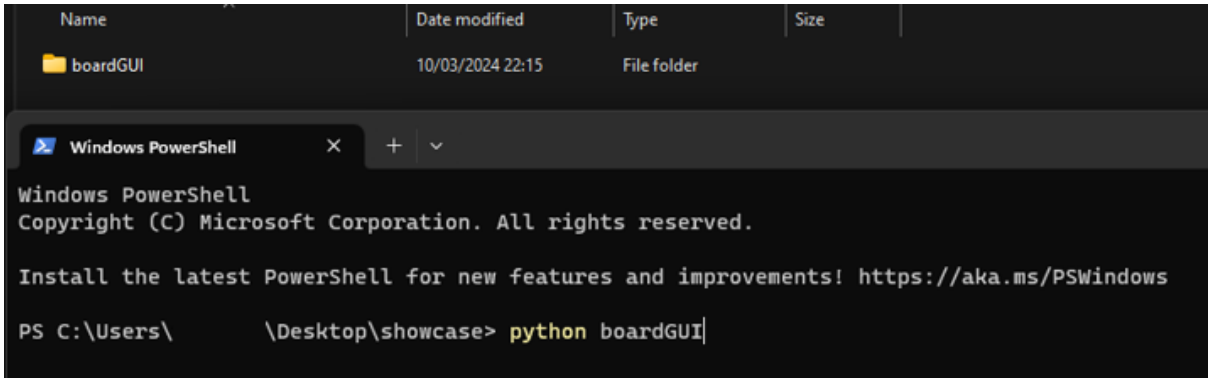


Bei Druck der Buttons werden folgende Funktionen aufgerufen:

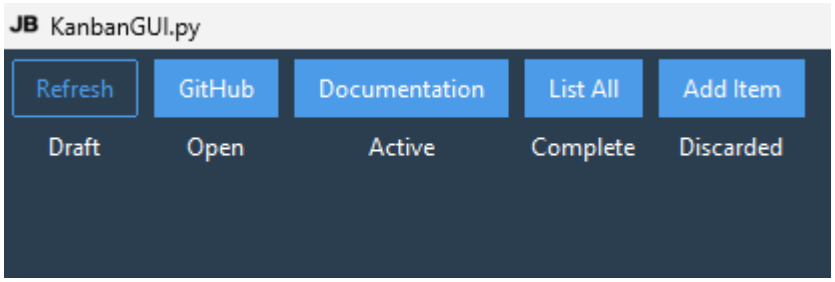
- Refresh: boardUtil.refresh
- GitHub: boardUtil.openFromProjectRoot
- Documentation: boardUtil.openURL
- List All: itemUtil.listItems
- Add Item: itemUtil.createItem
- Vorgänge allgemein: itemUtil.editItem

6. Benutzerhandbuch

Das Programm kann mit Python 3.12.0 gestartet werden:

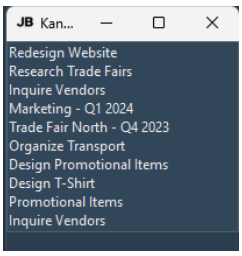


Nach dem Start erscheint folgendes Fenster:

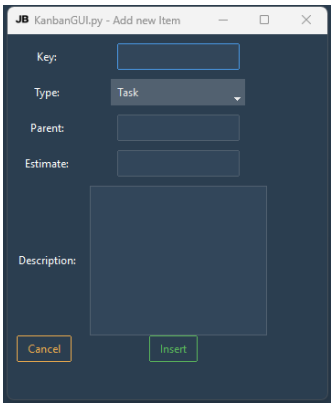


Buttons haben folgende Funktionalitäten:

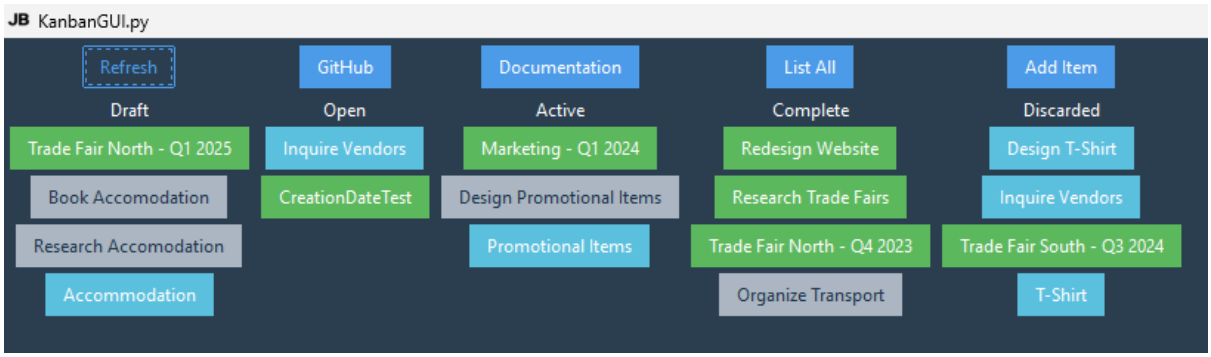
- „Refresh“
  Stellt Verbindung zur Datenbank her und füllt das Board.
- „GitHub“
  Öffnet das GitHub-Repository dieses Projekts in einem Webbrowser
- „Documentation“
  Öffnet die PDF-Dokumentation dieses Projekts in einem Webbrowser
- „List All“
  Stellt Verbindung zur Datenbank her und listet alle Vorgänge auf
  Mit Klick auf einen Vorgang kann dieser regulär verändert werden.



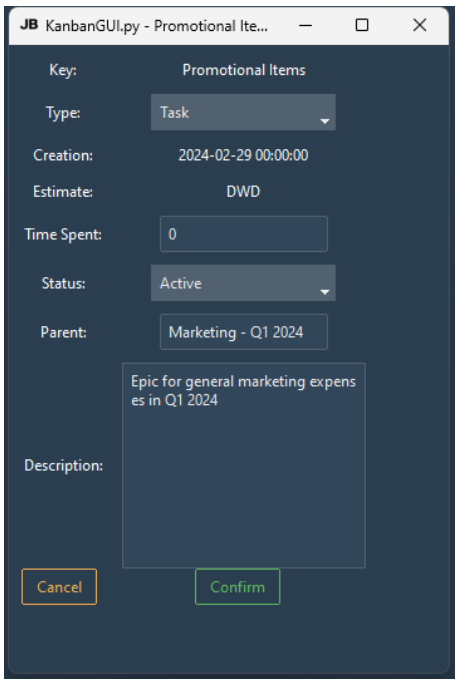
- „Add Item“
  Öffnet ein zusätzliches Fenster, mit dem ein Vorgang angelegt werden kann.



Folgendes Board wurde aktualisiert („refreshed“) und arbeitet mit einer mit Beispieldaten gefüllte Datenbank:



Mit Klick auf einzelne Vorgänge bzw. alle Knöpfe außerhalb der ersten Reihe öffnet sich der entsprechende Vorgang:



7. Veränderungen im Projektverlauf

7.1. Schema

Ursprünglich war geplant, dass Objekte in ihren Eltern-Objekten gespeichert werden. Allerdings war dies mit MongoDB Query-Syntax deutlich anspruchsvoller umzusetzen. (https://www.mongodb.com/docs/manual/tutorial/query-documents/)

7.2. Aktualisieren/Refreshen des Boards

Wenn ein Objekt mehrmals durch ein itemUtil.editItem-Fenster geöffnet wird und von einem verändert und gespeichert wird während, das andere Fenster geöffnet bleibt, wird das ursprüngliche Objekt nicht entfernt.
Dieses Problem trat erst zum Ende des Projekts auf. Bisher wurde das Problem nicht behoben, bspw. wurde ein Weg zur Limitierung der Fenster noch nicht implementiert.

8. Fazit

Folgende verpflichtende Akzeptanzkriterien wurden erfüllt:

- Datenspeicherung mit MongoDB
- Speicherung von Kanban-Objekttypen
- GUI zur Erstellung, lesen und Veränderung von Kanban-Objekttypen

Folgende Unverbindliche Akzeptanzkriterien wurden erfüllt:

- Visualisierung mit einem Kanban-Board
- Grafischer Dunkelmodus

Folgende Unverbindliche Akzeptanzkriterien wurden nicht erfüllt:

- Berichterstattung: Grafische Darstellung der prozentualen Komplettierung eines Epics
- Lizenz hinzufügen: Das Projekt mit einer Lizenz versehen, beispielsweise
    - MIT-Lizenz oder der
    - GNU General Public License Version 2.0
- Verbindung zu MongoDB per Zertifikat anstatt Zugangsdaten verwalten

9. Ausblick

Die Applikation könnte um folgende Funktionen erweitert werden:

- Auswahl einer Datenbankverbindung/Zugangsdateneingabe
- Connection Pooling: Verbessertes Pooling der Verbindungen unterschiedlicher Funktionen
- Ladeanzeige nach Refreshen
- Db.py-Datei anpassen, dass Zugangsdaten nicht als Default hinterlegt sind.