

Todo App

Entwicklerdokumentation



17. März 2024

Berufscschule

Moltkestraße 23

# Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc161579961)

[Projektzweck 2](#_Toc161579962)

[Entwurfsentscheidung 2](#_Toc161579963)

[Architekturübersicht 2](#_Toc161579964)

[Benutzeroberfläche 3](#_Toc161579965)

[Datenverwaltung 3](#_Toc161579966)

[Netzwerkkommunikation 3](#_Toc161579967)

[Abhängigkeiten 3](#_Toc161579968)

[Schlüsselkomponenten 4](#_Toc161579969)

[ToDoListItem 4](#_Toc161579970)

[MainScreen 4](#_Toc161579971)

[SettingScreen 4](#_Toc161579972)

[TaskStorageHandler 4](#_Toc161579973)

[MqttHandler 4](#_Toc161579974)

[MqttConfig 4](#_Toc161579975)

[Arbeitsaufteilung 5](#_Toc161579976)

[UML Diagramm 5](#_Toc161579977)

[Analyse 6](#_Toc161579978)

[Soll-Ist-Vergleich 6](#_Toc161579979)

[Grundfunktionalitäten: 6](#_Toc161579980)

[UI 7](#_Toc161579981)

[Quellen 8](#_Toc161579982)

# Projektzweck

Erschaffen einer Todo-App, die von mehreren Nutzern zur selben Zeit verwendet werden kann. Diese App kann von Familien für Ihren Einkauf genutzt werden oder für Aufgaben, die erledigt werden müssen. Damit kann jeder, der mit der gleichen Schnittstelle verbunden ist, die gemeinsame Liste bearbeiten.

# Entwurfsentscheidung

* Verwendung von KivyMD und MQTT:
  + Diese beide Technologien wurden vorgegeben
* Anwendung von OOP für die Code-Struktur:
  + Erlaubt eine klare Organisation und erleichtert die Erweiterbarkeit des Codes durch logische Strukturierung in Klassen.
* Nutzung von BoxLayout für die Scrollbarkeit:
  + Bietet eine benutzerfreundliche und flexible Anordnung der ToDo-Einträge in einem scrollbaren Feld zur vereinfachten Navigation.
* Einführung von Buttons mit Icons:
  + Verbessert die Benutzerinteraktion durch intuitive Bedienelemente, die spezifische Aktionen klar kennzeichnen.
* Implementierung von Responsive Design:
  + Stellt sicher, dass die App auf einer Vielzahl von Geräten optimal dargestellt wird und benutzerfreundlich bleibt.
* Fokus auf Einfachheit und Klarheit:
  + Konzentration auf wesentliche Funktionen und minimalistisches Design für maximale Usability.
* Integration von direkten Feedback-Mechanismen:
  + Fördert die Interaktivität, indem Nutzern sofortiges Feedback auf ihre Aktionen gegeben wird.

# Architekturübersicht

Die Anwendung ist mit der Kivy-Bibliothek für die grafische Benutzeroberfläche und der paho-mqtt-Bibliothek für die Netzwerkkommunikation entwickelt worden. Die Architektur des Systems kann in drei Hauptkomponenten unterteilt werden: die Benutzeroberfläche, die Datenverwaltung und die Netzwerkkommunikation.

## Benutzeroberfläche

Die Benutzeroberfläche ist mit Kivy und KivyMD erstellt.Sie besteht aus mehreren Bildschirmen und Widgets, die eine einfache Navigation und Interaktion ermöglichen. Dazu gehören die “MainToDoList” und “LoginScreen” Bildschirme, welche die Hauptinteraktionspunkte für die Benutzer darstellen. Die Oberfläche ermöglicht es Benutzern, Aufgaben hinzuzufügen, zu bearbeiten, zu löschen und deren Status zu aktualisieren.

## Datenverwaltung

Die Verwaltung der To-Do-Listen erfolgt durch die “TaskStorageHandler-Klasse”, die für das Lesen und Schreiben von Aufgaben in einer lokalen JSON-Datei verantwortlich ist. Diese Komponente sorgt für die Persistenz der Benutzerdaten und ermöglicht eine einfache Synchronisation mit dem MQTT-Server. Die Datenstruktur umfasst die UUID der Aufgabe, den Zustand und die Beschreibung.

## Netzwerkkommunikation

Die Netzwerkkommunikation wird durch die “MqttHandler-Klasse” realisiert, die eine Verbindung zu einem MQTT-Broker herstellt. Diese Komponente ist verantwortlich für das Publizieren von Änderungen an den To-Do-Listen und das Abonnieren von Updates, wodurch eine Echtzeit-Synchronisation der Daten zwischen verschiedenen Instanzen der Anwendung ermöglicht wird. Die MQTT-Konfiguration, einschließlich Broker-Adresse und Zugangsdaten, wird über die “MqttConfig-Klasse” verwaltet.

## Abhängigkeiten

* kivy: Framework
* kivymd: Erweiterte UI Möglichkeiten
* re: Modul für reguläre Ausdrücke
* sys: Modul für System-spezifische Funktionalitäten
* time: Zur Erzeugung von Zeitverzögerungen
* json: Zur Verarbeitung von JSON-Daten
* uuid: Zur Generierung eindeutiger Identifikatoren
* paho-mqtt: Bibliothek zur Implementierung von MQTT-Kommunikation

# Schlüsselkomponenten

## ToDoListItem

Ein benutzerdefiniertes Widget, das eine einzelne Aufgabe in der To-Do-Liste darstellt. Es enthält eine Checkbox, um den Status der Aufgabe (erledigt/nicht erledigt) zu markieren, sowie einen Textbereich für die Aufgabenbeschreibung.

## MainScreen

Die `MainScreen`-Klasse ist eine Bildschirmklasse, die die Hauptoberfläche der Anwendung darstellt. Sie ermöglicht das Laden, Hinzufügen, Synchronisieren, Bearbeiten und Löschen von Aufgaben.

## SettingScreen

Ein Bildschirm zur Eingabe der Konfigurationsdaten für die MQTT-Verbindung. Benutzer können hier Server-Details und Authentifizierungsdaten eingeben, um die Synchronisation über MQTT zu ermöglichen.

## TaskStorageHandler

Die `TaskStorageHandler`-Klasse ist verantwortlich für das Hinzufügen, Lesen, Aktualisieren und Löschen von Aufgaben in einer persistenten Datenspeicherung sowie für die Veröffentlichung von Änderungen über MQTT.

## MqttHandler

Kümmert sich um die Kommunikation mit dem MQTT-Broker. Diese Klasse ermöglicht das Publizieren von Änderungen an der To-Do-Liste und das Abonnieren von Updates, um eine Synchronisation zwischen verschiedenen Instanzen der App zu ermöglichen.

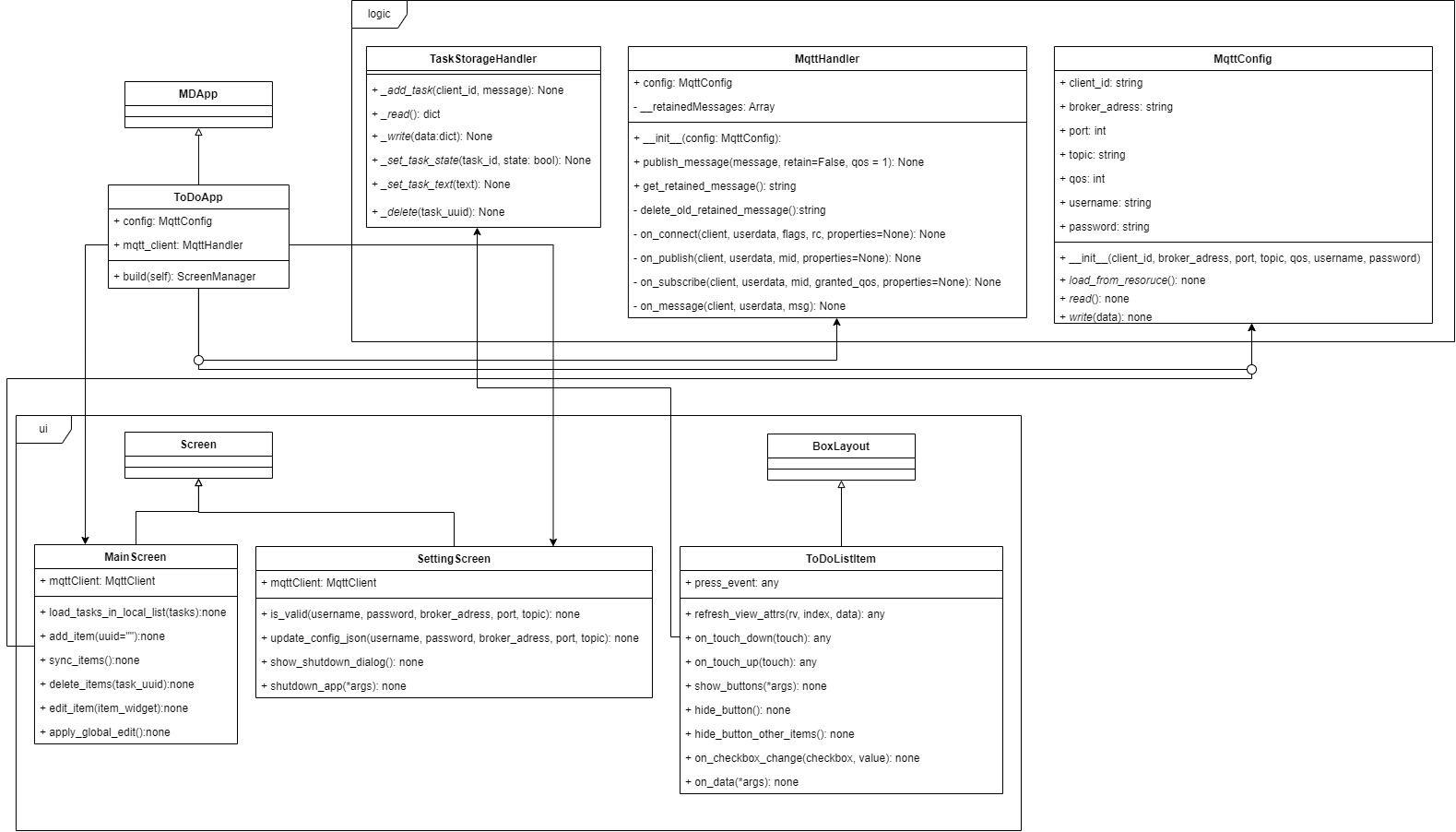
## MqttConfig

Verwaltet die Konfigurationsdaten für die MQTT-Verbindung. Diese Klasse lädt die Konfiguration aus einer JSON-Datei und stellt diese Informationen anderen Teilen der Anwendung zur Verfügung.

# Arbeitsaufteilung



# UML Diagramm



# Analyse

Um sicherzustellen, dass alle Funktionen ordnungsgemäß implementiert wurden, wird nun eine Abschlussanalyse durchgeführt. Diese umfasst die Anwendung der Soll-Ist-Vergleichsmethode.

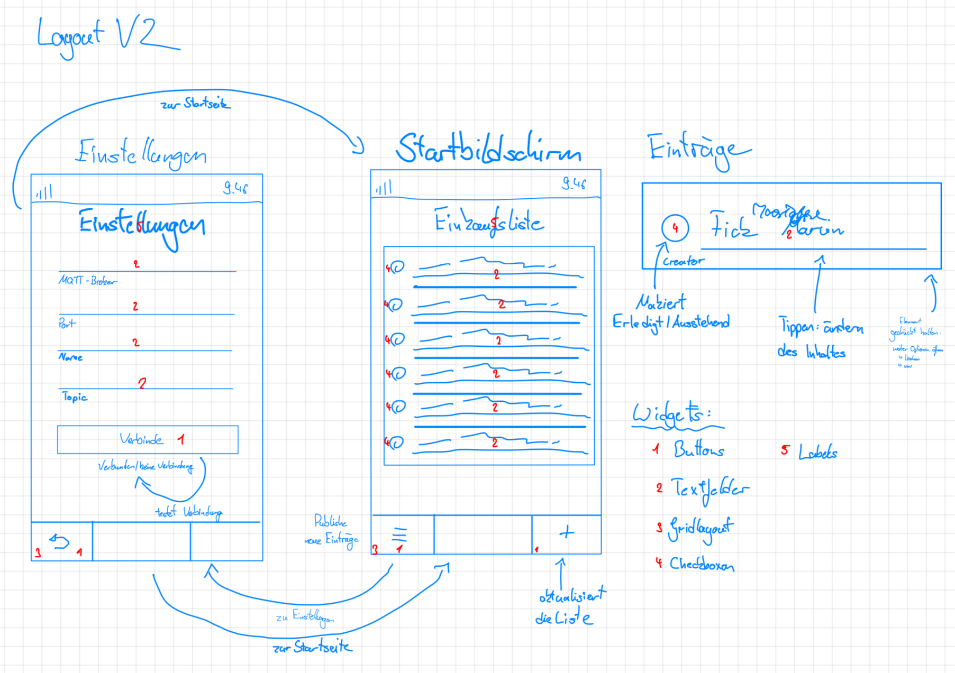
## Soll-Ist-Vergleich

### Grundfunktionalitäten:

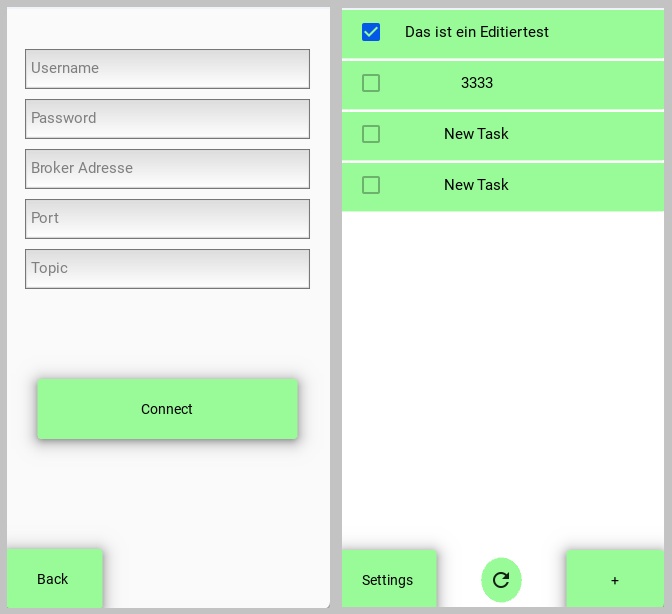
* Todo’s erstellen **Implementiert**
* Todo’s bearbeiten **Implementiert**
* Todo’s löschen **Implementiert**
* Mqtt-Broker Konfiguration anpassen **Implementiert**
* Todo’s hochladen **Implementiert**
* Todo’s runterladen **Implementiert**

### UI

#### Geplant



#### Umgesetzt



# Quellen

Hier werden Quellen aufgeführt die uns zur Vorbereitung und auch während der Durchführungsphase geholfen haben:

* <https://smarthome-blogger.de/blog/tutorial/python-mqtt-tutorial>
* <https://www.youtube.com/watch?v=6XoNpA_oYdw>
* <https://buildozer.readthedocs.io/en/latest/>
* <https://kivy.org/doc/stable/guide/basic.html>
* <https://www.geeksforgeeks.org/python-recycleview-in-kivy/>
* <https://www.geeksforgeeks.org/python-boxlayout-widget-in-kivy/>
* <https://www.geeksforgeeks.org/python-screenmanager-in-kivy-using-kv-file/>
* <https://docs.hivemq.com/hivemq-cloud/index.html>