

Todo App

Entwicklerdokumentation



17. März 2024

Berufscschule

Moltkestraße 24

# Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc161340048)

[Projektzweck 2](#_Toc161340049)

[Architekturübersicht 2](#_Toc161340050)

[Benutzeroberfläche 2](#_Toc161340051)

[Datenverwaltung 2](#_Toc161340052)

[Netzwerkkommunikation 3](#_Toc161340053)

[Abhängigkeiten 3](#_Toc161340054)

[Schlüsselkomponenten 4](#_Toc161340055)

[ToDoListItem 4](#_Toc161340056)

[MainScreen & SettingScreen Screens 4](#_Toc161340057)

[TaskStorageHandler 4](#_Toc161340058)

[MqttHandler 4](#_Toc161340059)

[MqttConfig 5](#_Toc161340060)

# 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.

# 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

Um das Projekt in einer Entwicklungsumgebung zum Laufen zu bringen müssen folgende Bibliotheken mit pip installiert werden:  
  
KivyMD (pip install kivymd)  
paho-mqtt 1.6.1 (pip install paho-mqtt<2.0.0)

yaml (pip install yaml)

Konfiguration

Damit zur Laufzeit eine Verbindung zum Mqtt-Broker hergestellt werden kann müssen folgende Parameter entweder im Vorraus oder innerhalb der Anwendung auf der Seite der Einstellungen eingetragen werden:  
  
Broker-Adresse: Domäne des Mqtt-Broker.

Port: entsprechender Port zur Adresse.

Username: Name des Klienten

Password: Passwort des Klienten

# Schlüsselkomponenten

## ToDoListItem

Beschreibung: 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.

Interaktion: Benutzer können den Status einer Aufgabe durch das Aktivieren der Checkbox ändern. Durch langes Drücken auf eine Aufgabe werden Bearbeiten- und Löschen-Buttons angezeigt, die weitere Interaktionen ermöglichen.

## MainScreen & SettingScreen Screens

MainScreen: Der Hauptbildschirm der App, der die To-Do-Liste anzeigt. Benutzer können hier neue Aufgaben hinzufügen, vorhandene bearbeiten oder löschen und ihre Aufgaben synchronisieren.

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

Funktionalität: Verwaltet das Speichern, Laden und Ändern von Aufgaben in einer lokalen JSON-Datei. Diese Klasse ist zentral für die Persistenz der Benutzerdaten.

Methoden: Beinhaltet Methoden zum Hinzufügen, Bearbeiten, Löschen von Aufgaben und zum Ändern des Status einer Aufgabe.

## MqttHandler

Funktionalität: 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.

Konfiguration: Die Konfigurationsdetails für die MQTT-Verbindung werden durch die “MqttConfig”-Klasse bereitgestellt und beinhalten Informationen wie Broker-Adresse, Port und Authentifizierungsdaten

## MqttConfig

Funktionalität: 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.

Anpassung: Benutzer können die Konfigurationsdatei bearbeiten, um die Verbindungsdetails für ihren spezifischen MQTT-Broker anzupassen.