

# Grobdesign

---

## Übersicht

**Projekt:** Projekt Episkos

*Iteration:* 4

*Arbeitspaket:* -

**Autor:** Simon Blum, Ben Oeckl, Paul Stöckle

**Datum:** 05.12.2024

**Zuletzt geändert:**

*von:*

*am:*

**Version:** 1

**Prüfer:** Max Rodler

**Letzte Freigabe:**

*durch:* Max Rodler

*am:* 05.12.2024

## Changelog

---

Datum	Verfasser	Kurzbeschreibung
05.12..2024	Simon Blum	Initiales Erstellen und Verfassen

---

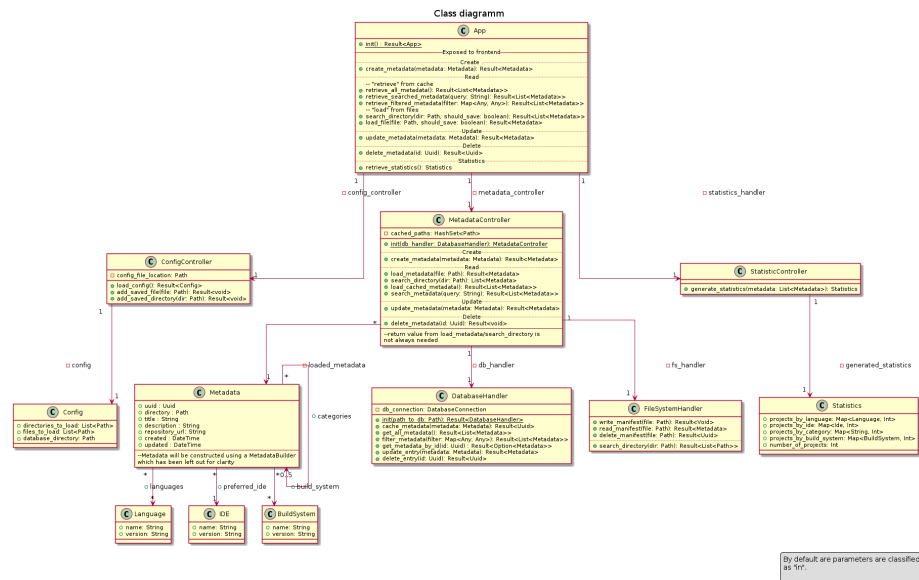
## Distribution List

- Simon Blum [simon21.blum@gmail.com](mailto:simon21.blum@gmail.com)
  - Ben Oeckl [ben@oeckl.com](mailto:ben@oeckl.com)
  - Maximilian Rodler [maximilianreinerrodler@gmail.com](mailto:maximilianreinerrodler@gmail.com)
  - Paul Stöckle [paul.stoeckle@t-online.de](mailto:paul.stoeckle@t-online.de)
- 

## Klassendiagramm

### Anmerkung:

Während das Klassendiagramm als Grundlage zur Implementierung sicherlich eine Stütze stellen kann, sollte trotzdem beachtet werden, dass Rust als Sprache sich nur begrenzt zur Objektorientierung anbietet und einiges durch leichte Veränderung idiomatischer, sauberer und effizienter implementiert werden kann und sollte. Hierfür sollen bei auftretenden Fällen, entsprechende Anmerkungen im Klassendiagramm angefügt werden.



## Sequenzdiagramme

Die Sequenzdiagramme basieren auf den Use-Cases und sind dementsprechend aufgeteilt.

U1.1

U1.2

U1.3

U1.4

U2.1

U2.2

U3.1

U3.2

U3.3

## Anforderungstracing

### Struktur

Die Anforderungsverfolgung ist aktuell nach folgender Struktur aufgebaut:

[Use Case] -> [Anforderung] -> [Klassenattribut]

In Zukunft soll diese noch in ein passendes Diagramm überführt werden.

**Verfolgung**

UC1.1 -> FA1.1.1 -> App.retrieve\_all\_metadata()  
UC1.1 -> FA1.1.2 -> FileSystemHandler.read\_manifest()  
UC1.2 -> FA1.2.1 -> App.create\_metadata(), MetadataController.create\_metadata(), DatabaseHandler.update\_entry() FileSystemHandler.write\_manifest()  
UC1.2 -> FA1.2.2 -> FileSystemHandler.write\_manifest()  
UC1.2 -> FA1.2.3 -> DatabaseHandler.cache\_metadata()  
UC1.3 -> FA1.3.1 -> App.update\_metadata()  
UC1.3 -> FA1.3.2 -> MetadataController.update\_metadata(), DatabaseHandler.update\_entry()  
UC1.3 -> FA1.3.3 -> MetadataController.load\_metadata(), MetadataController.search\_directory(), FileSystemHandler.read\_manifest()  
UC1.4 -> FA1.4.1 -> App.delete\_metadata, MetadataController.delete\_metadata, FileSystemHandler.delete\_manifest()  
UC1.4 -> FA1.4.2 -> DatabaseHandler.delete\_entry(),  
UC1.4 -> FA1.4.3 -> MetadataController.load\_metadata(), MetadataController.search\_directory(), MetadataController.delete\_metadata, DatabaseHandler.delete\_entry()  
UC2.1 -> FA2.1.1 -> App.loadFile, ConfigController.add\_saved\_file()  
UC2.1 -> FA2.1.2 -> Siehe FA1.1.2  
UC2.1 -> FA2.1.3 -> siehe FA1.2.3  
UC2.1 -> FA2.1.4 -> App.search\_directory(), ConfigController.add\_saved\_directory()  
UC2.1 -> FA2.1.5 -> MetadataController.search\_directory(), FileSystemHandler.search\_directory()  
UC2.2 -> FA2.2.1 -> Siehe FA2.1.4  
UC2.2 -> FA2.2.2 -> Siehe FA2.1.5  
UC2.2 -> FA2.2.3 -> MetadataController.load\_metadata(), MetadataController.update\_metadata(), DatabaseHandler.update\_entry()  
UC2.2 -> FA2.2.4 -> Siehe FA2.1.5  
UC2.2 -> FA2.2.5 -> Siehe FA2.1.5  
UC3.1 -> FA3.1.1 -> MetadataController.search\_metadata()  
UC3.1 -> FA3.1.2 -> App.retrieve\_searched\_metadata() - Input der Methode stellt Nutzereingabe da

UC3.1 -> FA3.1.3 -> App.retrieve\_searched\_metadata() - Output Methode wird nutzer angezeigt

UC3.2 -> FA3.2.1 -> MetadataController.filter\_metadata()

UC3.2 -> FA3.2.2 -> App.retrieve\_filtered\_metadata() - Input der Methode stellt Nutzereingabe da

UC3.2 -> FA3.2.3 -> App.retrieve\_filtered\_metadata() - Output Methode wird nutzer angezeigt

UC3.3 -> FA3.3.1 -> StatisticsController.generate\_statistics()

UC3.3 -> FA3.3.2 -> StatisticsController.retrieve\_statistics()