Software Detailed Design

$\ddot{\mathbf{U}}\mathbf{bersicht}$

Projekt: Projekt Episko Autor: Simon Blum Datum: 27.01.2025 Zuletzt geändert: von: Paul Stöckle am: 07.02.2025

Version: 5 Prüfer:

Letzte Freigabe: durch: Ben Oeckl am: 08.02.2025

Changelog

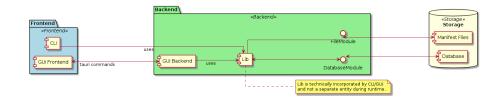
27.01.2025 Simon Blum Initiales Erstellen und Verfassen 01.02.2025 Paul Stöckle Hinzufügen von CLI Modul 03.02.2025 Simon Blum Hinzufügen von Datenbank Modul	Datum	Verfasser	Kurzbeschreibung
07.02.2025 Simon Blum Hinzurugen von Links fürs Wiki 07.02.2025 Paul Stöckle Aktualisierung der CLI Library-Abhängigkeiten	01.02.2025 03.02.2025 07.02.2025	Paul Stöckle Simon Blum Simon Blum	Hinzufügen von CLI Modul Hinzufügen von Datenbank Modul Hinzufügen von Links fürs Wiki Aktualisierung der CLI

Distribution List

- Simon Blum simon21.blum@gmail.com
- Ben Oeckl ben@oeckl.com
- $\bullet \ \ Maximilian \ Rodler \ maximilian reinerrodler@gmail.com$
- Paul Stöckle paul.stoeckle@t-online.de

Übersicht Systemarchitektur

- Aufgeteilt in Lib, Cli, Gui
- Klassendiagramme/Sequenzdiagramme in Grobdesign
- Responsibility, wer macht was, was ist wo?
- Noch ein kleines Diagramm zu Komponentenübersicht?



Module

Funktionen - Schnittstellen - Datenmodelle

Backend - Datenbank

• Mehr ...

Backend - Lib

• Mehr ...

Backend - Derive Macro

• Mehr ...

Backend - Gui

• Mehr ...

Frontend- Gui

• Mehr ...

Frontend - Cli

• Mehr ...

Technische Spezifikationen

Sprachen/Technologien

- Rust
- Sqlite
- toml
- \bullet TypeScript
- \bullet Html
- (CSS)

Frameworks

- Tauri
- SvelteKit

Libraries

Verwendet Libraries und ihre Versionen können in den einzelnen Modulen gefunden werden.

Algorithmen

• Sha256 Hashing (verwendet, implementierung durch lib)

Qualitäts- und Sicherheitsaspekte

Qualität

- Tests in Front- und Backend
- Ci/Cd
 - Automatisches Testen
 - Prüfen, dass gebaut werden kann
- Release Steps
 - feat Branch während Inkrement
 - alpha Branch während nächstem Inkrement
 - beta/next bis nächster Release

Performance

Performance wird in erster Stelle durch die Verwendung von Rust und performanten Frameworks gesichert

Sicherheit

Für die Anwendung wurden die Manifest-Dateien als primäre mögliche Angriffsstelle identifiziert, da diese in öffentlichen Repositories liegen können und direkt von dem Program verarbeitet werden. Vor allem wäre hier in der Theorie eine Sql Injektion durch bösartig gesetzte Schlüssel denkbar. Um dies zu verhindern wird die Library sqlx verwendet.

- Memory Safety und so durch Rust...
- Sonst keine Netzanbindung