

PROJEKTDOKUMENTATION

Karmaknit System Release Editor

Webbasierter Editor für Systemfreigaben

Projekt	Webbasierter Editor für Systemfreigaben
Projektleiter	Max Schmitt
Unternehmen	Karmaknit GmbH, Schloßstrasse 15, 60499 Frankfurt am Main

Stand: 21. Oktober 2026

Inhaltsverzeichnis

1 Beschreibung der Ausgangssituation.....	2
1.1 Thema der Projektarbeit.....	2
1.2 Projektziele und wichtige Teilaufgaben.....	2
1.3 Kundenwünsche.....	2
1.4 Abweichungen vom Projektantrag	3
1.5 Qualitative und quantitative Beschreibung des Projektumfelds.....	3
1.6 Prozessschnittstellen und Ansprechpartner	3
1.7 Abgrenzung des Projekts, Ein- und Ausstiegspunkt.....	3
2 Ressourcen- und Ablaufplanung	4
2.1 Ablaufplanung	4
3 Durchführung und Auftragsbearbeitung	5
3.1 Architektur	5
3.2 Datenmodell.....	5
3.3 Persistenz	6
3.4 ViewModels	6
3.5 Views	7
3.6 Qualitätssicherung	9
3.7 Entscheidungen.....	9
4 Projektergebnisse	11
4.1 Soll-Ist-Vergleich	11
4.2 Abweichungen und Anpassungen.....	11
4.3 Qualitätskontrolle	11
4.4 Fazit	11
5 Verzeichnisse und Anhang	12
5.1 Start der Anwendung.....	12
5.2 Abkürzungsverzeichnis.....	12
5.3 Literaturverzeichnis	12

1 Beschreibung der Ausgangssituation

Die Karmaknit GmbH entwickelt Softwarelösungen für Kunden. Ein Gesamtsystem besteht dabei aus mehreren einzelnen Softwarekomponenten, die jeweils separat freigegeben werden. Im Rahmen einer Systemfreigabe werden diese Komponenten zu einem Gesamtsystem integriert und mit einer Freigabedokumentation versehen. Dieses Projekt ersetzt die bisher manuell gepflegten Freigabedokumente durch eine Webanwendung, in der die Freigabedaten strukturiert erfasst, gepflegt und angezeigt werden können.

1.1 Thema der Projektarbeit

Gegenstand der Projektarbeit ist die Konzeption und Entwicklung des „Karmaknit System Release Editor“, eines webbasierten Editors für Systemfreigaben. Die Anwendung verwaltet zwei Arten von Freigaben: die Freigabe eines Gesamtsystems und die Freigabe einer einzelnen Softwarekomponente. Beide enthalten den Namen, die Version, das Freigabedatum und die verantwortliche Person. Eine Systemfreigabe führt zusätzlich die Komponenten auf, aus denen das System besteht.

1.2 Projektziele und wichtige Teilaufgaben

Bisher wurden Freigabedokumente mit Textverarbeitungsprogrammen erstellt. Die enthaltenen Informationen lassen sich in dieser Form nur schwer automatisiert auswerten. Hauptziel ist daher die Digitalisierung der Daten, sodass sie elektronisch weiterverarbeitet werden können. Ein typischer Anwendungsfall ist die Sicherheitsüberprüfung der enthaltenen Komponenten, wie sie beispielsweise der EU Cyber Resilience Act vorsieht. Dafür muss jederzeit nachvollziehbar sein, welche Komponente in welcher Version in welchem System enthalten ist.

Zur Erreichung dieses Ziels wurden folgende Teilaufgaben definiert:

- Entwurf eines Datenmodells für System- und Komponentenfreigaben
- Speicherung der Daten als JSON-Dateien im Dateisystem
- Bereitstellung einer Oberfläche zum Anlegen, Anzeigen, Ändern und Löschen beider Freigabearten (CRUD)
- Dokumentartige Ansicht einer Systemfreigabe, in der die Komponenten als Links auf die jeweilige Komponentenfreigabe dargestellt werden

1.3 Kundenwünsche

Der Kunde hat die technischen Rahmenbedingungen klar vorgegeben. Die Anwendung wird mit Blazor Server 8 und C# umgesetzt. Für die Oberfläche kommt Bootstrap zum Einsatz, für die Persistenz die Bibliothek JSON.NET. Zusätzlich sind die firmeninternen Codierichtlinien einzuhalten. Ein Benutzermanagement oder ein Rechtesystem ist ausdrücklich nicht gewünscht. Auf Unit-Tests wird verzichtet; stattdessen soll die Anwendung anhand von Beispieldaten getestet werden.

1.4 Abweichungen vom Projektantrag

Es gab keine wesentlichen Abweichungen vom Projektantrag. Lediglich die Wahl der konkreten Dateinamen und des Speicherordners war im Antrag offengelassen und wurde im Projekt festgelegt (siehe Kapitel 3.3).

1.5 Qualitative und quantitative Beschreibung des Projektumfelds

Die Anwendung kommt in der Abteilung für Systemfreigaben zum Einsatz. Die Datenmengen sind überschaubar: Es entsteht pro Freigabeversion eine kleine JSON-Datei, sodass auch bei vielen Systemen und Komponenten keine besonderen Anforderungen an Performance oder Speicherplatz entstehen.

Neben den technischen Themen sind auch nicht technische Aspekte zu beachten:

- **Datenschutz:** Gespeichert werden ausschließlich Freigabeinformationen sowie die Namen der verantwortlichen Personen. Weitere personenbezogene Daten werden nicht erfasst.
- **IT-Sicherheit:** Da es kein Rechtesystem gibt, kann jeder, der die Anwendung erreicht, Daten ändern. Der Zugriff auf die Anwendung und den Datenordner muss daher organisatorisch abgesichert werden, etwa durch Netzwerkzugriff und Dateirechte. Dies sollte bei einem produktiven Einsatz berücksichtigt werden.
- **Lizenzen:** Es werden ausschließlich quelloffene Bibliotheken unter der MIT-Lizenz verwendet (Newtonsoft.Json, CommunityToolkit.Mvvm, Bootstrap). Lizenzkosten entstehen nicht.

1.6 Prozessschnittstellen und Ansprechpartner

Projektleiter und Ansprechpartner ist Max Schmitt. Prozessschnittstelle ist die Abteilung für Systemfreigaben, die die Anwendung später nutzt. Die Eingabe der Daten erfolgt derzeit manuell auf Basis der bestehenden Freigabedokumente. Eine Anbindung an andere Systeme besteht noch nicht; die JSON-Dateien bilden jedoch eine einfache Schnittstelle für spätere Auswertungen.

1.7 Abgrenzung des Projekts, Ein- und Ausstiegspunkt

Nicht Teil des Projekts sind eine Benutzer- und Rechteverwaltung, eine Datenbankanbindung, automatisierte Tests sowie eine automatische Sicherheitsprüfung der Komponenten. Der Einstiegspunkt des Projekts sind die bestehenden, in Textverarbeitungsprogrammen erstellten Freigabedokumente, deren Inhalte künftig im Editor erfasst werden. Der Ausstiegspunkt sind die erzeugten JSON-Dateien, die von weiteren Werkzeugen gelesen und ausgewertet werden können.

2 Ressourcen- und Ablaufplanung

Das Projekt wird von Max Schmitt als Projektleiter durchgeführt. Der Aufwand wurde mit 80 Arbeitsstunden veranschlagt. An Sachmitteln werden ein Entwicklungsrechner und das .NET SDK benötigt; da ausschließlich quelloffene Bibliotheken verwendet werden, fallen keine Lizenzkosten an. Die Kosten beschränken sich damit auf die Personalkosten des Projektleiters. Die Planung ist bewusst einfach gehalten, da die Anforderungen klar umrissen und der Umfang überschaubar sind.

2.1 Ablaufplanung

Das Projekt gliedert sich in fünf aufeinander aufbauende Phasen. Jede Phase endet mit einem klar benennbaren Ergebnis, das als Meilenstein dient. Da die Phasen inhaltlich voneinander abhängen, werden sie nacheinander durchgeführt.

Phase	Ergebnis / Meilenstein
Konzeption der Architektur	Entscheidung für MVVM und die Aufteilung in drei Projekte
Erstellung des Datenmodells	Datenklassen und Aufbau der JSON-Dateien
Implementierung	ViewModels und Views, CRUD-Funktionen lauffähig
Test	Manueller Test mit Beispieldaten
Dokumentation	Vorliegende Projektdokumentation

3 Durchführung und Auftragsbearbeitung

3.1 Architektur

Als Architekturmuster wurde im Projekt MVVM gewählt (siehe Kapitel 3.7). Die Anwendung ist in drei Projekte aufgeteilt, die in der Lösung `src/Karmaknit.SystemReleaseEditor.sln` zusammengefasst sind. Die Trennung sorgt dafür, dass die Oberfläche keine Kenntnis über die Details der Datenspeicherung benötigt und die Darstellungslogik unabhängig von den Razor-Seiten bleibt.

Projekt	Namespace	Inhalt
DataModels	Karmaknit.SystemReleaseEditor.DataModels	Datenklassen, JSON-Persistenz (ReleaseRepository), Beispieldaten
ViewModels	Karmaknit.SystemReleaseEditor.ViewModels	ViewModels auf Basis von CommunityToolkit.Mvvm
UI	Karmaknit.SystemReleaseEditor.UI	Blazor-Server-Anwendung mit den Views, Bootstrap

Die Abhängigkeiten verlaufen in eine Richtung: Die UI kennt die ViewModels, die ViewModels kennen die DataModels. Der Browser kommuniziert bei Blazor Server über eine SignalR-Verbindung mit dem Server, auf dem die gesamte Anwendungslogik läuft.

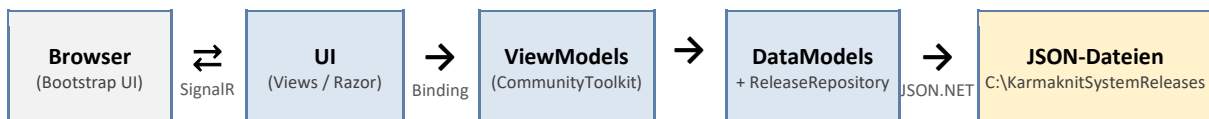


Abbildung: Architektur auf hoher Ebene (Blazor-Server-Anwendung: UI, ViewModels, DataModels)

Das ReleaseRepository wird beim Start einmalig erzeugt und als Singleton bereitgestellt, da es den Zugriff auf den gemeinsamen Datenordner kapselt. Die ViewModels werden pro Seitenaufruf neu erzeugt (transient) und per Dependency Injection mit dem Repository versorgt.

3.2 Datenmodell

Die gemeinsamen Eigenschaften beider Freigabearten sind in der abstrakten Basisklasse ReleaseBase zusammengefasst. Davon abgeleitet sind SoftwareComponentRelease für Komponentenfreigaben und SystemRelease für Systemfreigaben. Letztere ergänzt eine Liste von Komponentenverweisen.

```
ReleaseBase (Name, Version, ReleaseDate, ResponsiblePerson, Description)
├── SoftwareComponentRelease
└── SystemRelease — Components: List<ComponentReference(Name, Version)>
```

Eine Systemfreigabe enthält keine Kopien der Komponentenfreigaben, sondern nur Verweise (ComponentReference) aus Name und Version. Damit kann dieselbe Komponentenfreigabe in beliebig vielen Systemfreigaben verwendet werden, ohne dass Daten doppelt gepflegt werden müssen. Wird eine Komponentenfreigabe korrigiert, ist die Änderung automatisch in allen Systemfreigaben sichtbar, die darauf verweisen.

3.3 Persistenz

Die Freigaben werden als JSON-Dateien im Dateisystem abgelegt. Für jede Freigabeversion existiert eine eigene Datei, und zwar getrennt nach Systemfreigaben und Komponentenfreigaben. Der Dateiname setzt sich aus einem Präfix, dem Namen und der Version zusammen, sodass er auch für Menschen auf den Inhalt schließen lässt. Leerzeichen und unzulässige Zeichen im Namen werden dabei durch Bindestriche ersetzt. Als Speicherort dient der Ordner C:\KarmaknitSystemReleases, der über den Eintrag DataFolder in der appsettings.json angepasst werden kann.

```
System_Shop-Suite_1.1.0.json
Component_Auth-Service_1.3.0.json
```

Der Inhalt einer Systemfreigabedatei sieht beispielsweise so aus:

```
{
  "Components": [ { "Name": "Auth Service", "Version": "1.3.0" } ],
  "Name": "Shop Suite",
  "Version": "1.1.0",
  "ReleaseDate": "2025-06-20T00:00:00",
  "ResponsiblePerson": "Max Schmitt",
  "Description": "Adds reporting and updated authentication."
}
```

Name und Version bilden zusammen den Schlüssel einer Freigabe. Ändert der Benutzer einen der beiden Werte, wird die Datei unter dem neuen Namen gespeichert und die alte entfernt. Bei Komponentenfreigaben werden dabei zusätzlich die Verweise in allen Systemfreigaben angepasst, damit keine toten Referenzen entstehen. Wird eine Komponentenfreigabe gelöscht, entfernt das Repository die Verweise darauf aus den Systemfreigaben; die Oberfläche weist vor dem Löschen darauf hin. Schreib- und Lesezugriffe sind synchronisiert, weil Blazor Server mehrere Sitzungen parallel bedient. Ist der Datenordner beim ersten Start leer, legt der SampleDataSeeder Beispieldaten an, mit denen die Anwendung direkt ausprobiert werden kann.

3.4 ViewModels

Die ViewModels kapseln die Logik der Oberfläche und stellen den Views Daten und Operationen zur Verfügung. Sie leiten von `ObservableObject` des `CommunityToolkit.Mvvm` ab. Die Properties werden dabei über das Attribut `[ObservableProperty]` vom Source Generator erzeugt, sodass kein Boilerplate-Code für Änderungsbenachrichtigungen geschrieben werden muss.

ViewModel	Aufgabe
SystemReleaseListViewModel, ComponentReleaseListViewModel	Laden der Freigabelisten (Read) und Löschen (Delete)
SystemReleaseEditViewModel	Anlegen und Ändern einer Systemfreigabe (Create/Update) inklusive Auswahl der Komponenten
ComponentReleaseEditViewModel	Anlegen und Ändern einer Komponentenfreigabe; liefert auch die Systeme, die sie verwenden

ViewModel	Aufgabe
SystemReleaseDocumentViewModel	Schreibgeschützte Dokumentansicht, in der die Komponentenverweise zu den eigentlichen Freigaben aufgelöst sind

Die Edit-ViewModels übernehmen auch die Validierung: Name und Version sind Pflichtfelder, und die Kombination aus beiden muss eindeutig sein. Bei einem Fehler setzen sie eine Meldung, die die View anzeigt. Ein Ausschnitt:

```
public partial class ComponentReleaseEditViewModel : ObservableObject
{
    [ObservableProperty]
    private string _name = string.Empty;
    ...
    public bool Save() // validates (required fields, unique key) and stores
```

3.5 Views

Die Views sind Razor-Komponenten im Projekt UI und verwenden Bootstrap für Layout, Tabellen, Formulare und Schaltflächen. Sie binden direkt an die Properties der ViewModels. Die Navigation erfolgt über eine Menüleiste mit den Bereichen „System releases“ und „Component releases“. Die Schlüssel einer Freigabe (Name und Version) werden als Query-Parameter an die Seiten übergeben.

Seite	Route	Funktion
Startseite	/	Einstieg und Navigation
Systemfreigaben	/systems	Liste mit Ansehen, Bearbeiten und Löschen
Systemfreigabe bearbeiten	/systems/edit	Anlegen und Ändern, Auswahl der Komponenten
Dokumentansicht	/systems/document	Dokumentartige Darstellung einer Systemfreigabe
Komponentenfreigaben	/components	Liste mit Ansehen, Bearbeiten und Löschen
Komponente bearbeiten / Detail	/components/edit, /components/detail	Anlegen, Ändern und Anzeigen

Die Dokumentansicht stellt eine Systemfreigabe ähnlich einem klassischen Freigabedokument dar, mit Kopfdaten, Beschreibung und einer nummerierten Liste der Komponenten. Jede Komponente ist ein Link auf ihre Detailseite, von der aus wiederum die Systeme erreichbar sind, die sie verwenden. Fehlt die Datei einer referenzierten Komponente, wird dies in der Liste rot markiert. Zwei Ausschnitte zeigen das Zusammenspiel von View und ViewModel:

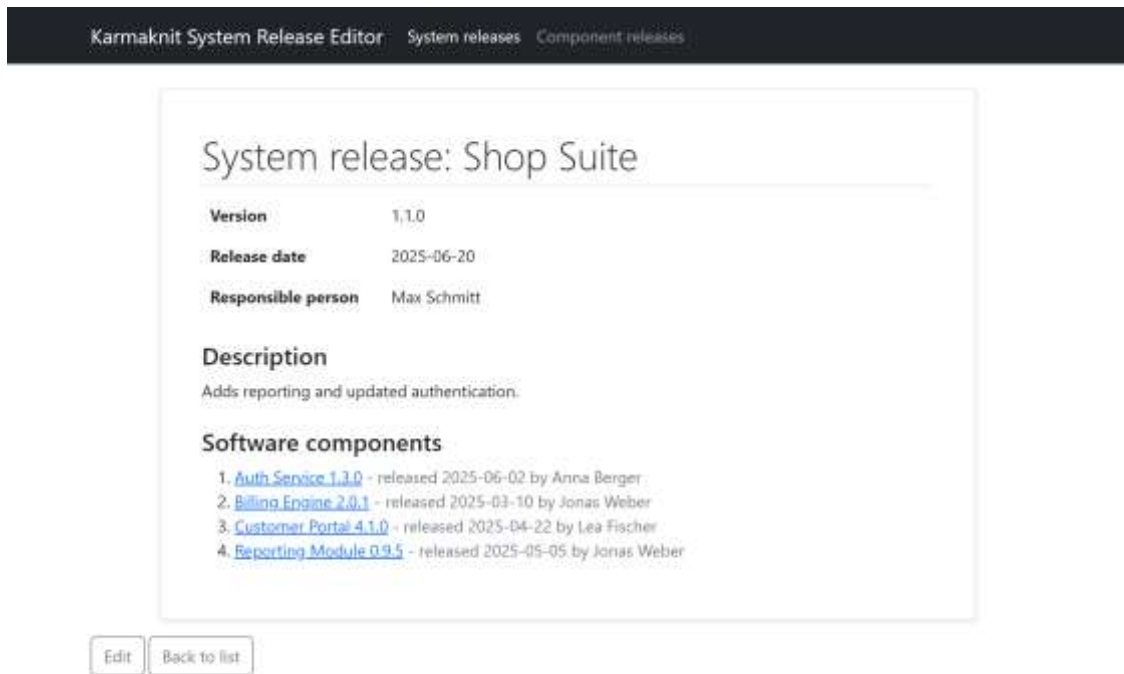
```
<input class="form-control" @bind="ViewModel.Name" />
...
<a href="@Links.ComponentDetail(currentEntry.Reference.Name, currentEntry.Reference.Version)">
    @currentEntry.Reference.Name @currentEntry.Reference.Version</a>
```

Die folgenden Screenshots zeigen die Liste der Systemfreigaben, die Dokumentansicht und die Bearbeitungsmaske mit der Komponentenauswahl:



Karmaknit System Release Editor System releases Component releases					
System releases					
New system release					
Name	Version	Release date	Responsible	Components	
Shop Suite	1.0.0	2025-04-30	Max Schmitt	3	View Edit Delete
Shop Suite	1.1.0	2025-06-20	Max Schmitt	4	View Edit Delete

Abbildung: Liste der Systemfreigaben



Karmaknit System Release Editor System releases Component releases

System release: Shop Suite

Version 1.1.0

Release date 2025-06-20

Responsible person Max Schmitt

Description
Adds reporting and updated authentication.

Software components

1. [Auth Service 1.3.0](#) - released 2025-06-02 by Anna Berger
2. [Billing Engine 2.0.1](#) - released 2025-03-10 by Jonas Weber
3. [Customer Portal 4.1.0](#) - released 2025-04-22 by Lea Fischer
4. [Reporting Module 0.9.5](#) - released 2025-05-05 by Jonas Weber

[Edit](#) [Back to list](#)

Abbildung: Dokumentansicht einer Systemfreigabe

Edit system release

Name	Shop Suite	Version	1.1.0
Release date	20.06.2025	Responsible person	Max Schmitt
Description Adds reporting and updated authentication.			
Software components			
☐ Auth Service 1.2.0 ☒ Auth Service 1.3.0 ☒ Billing Engine 2.0.1 ☒ Customer Portal 4.1.0 ☒ Reporting Module 0.9.5			
Save		Cancel	

Abbildung: Bearbeiten einer Systemfreigabe mit Komponentenauswahl

3.6 Qualitätssicherung

Gemäß Vorgabe wurden keine Unit-Tests erstellt. Die Anwendung wurde stattdessen manuell im Browser mit den Beispieldaten getestet. Geprüft wurden die Listenansichten, die Dokumentansicht einschließlich der Komponentenlinks, das Bearbeiten einer Systemfreigabe mit der Komponentenauswahl sowie das Anlegen einer neuen Komponentenfreigabe, bei dem die erwartete JSON-Datei mit lesbarem Namen im Datenordner entstand. Das Löschen sowie das Umbenennen mit Anpassung der Referenzen wurden nicht separat über die Oberfläche getestet; hier ist ein weiterer Test empfehlenswert. Die Einhaltung der Codierrichtlinien (expliziter Typ statt var, ein einzelnes `return result`; am Methodenende, Block-Namespaces, Unterstrich bei privaten Feldern, englische Kommentare) wurde bei der Implementierung beachtet.

3.7 Entscheidungen

Im Projekt wurden folgende wesentliche Entscheidungen getroffen. Sie betreffen die Architektur und Umsetzung, soweit sie nicht vom Kunden vorgegeben waren:

- **MVVM als Architekturmuster:** Die Anwendung folgt dem Model-View-ViewModel-Muster. Diese Entscheidung wurde im Projekt getroffen und ist kein Kundenwunsch. Sie trennt Darstellung (Razor-Views), Darstellungslogik (ViewModels) und Daten (DataModels), hält die Views schlank und erleichtert Wartung und spätere Erweiterungen.
- **CommunityToolkit.Mvvm für die ViewModels:** Auch der Einsatz dieser Bibliothek ist eine Designentscheidung des Projekts. Ihr Source Generator ([`ObservableProperty`]) erzeugt die Properties samt Änderungsbenachrichtigung und reduziert so den Schreibaufwand und die Fehleranfälligkeit gegenüber einer manuellen Implementierung von

INotifyPropertyChanged. Die Bibliothek ist quelloffen (MIT-Lizenz) und wird von Microsoft gepflegt.

- **Referenzen statt Kopien:** Systemfreigaben verweisen auf Komponentenfreigaben, statt sie zu duplizieren. Das vermeidet Redundanz und Inkonsistenzen und erfüllt die Anforderung, Komponenten in mehreren Systemen zu verwenden.
- **Name und Version als Schlüssel:** Beide Werte bestimmen auch den Dateinamen. Eine technische ID wäre nicht nötig und würde die Dateinamen weniger lesbar machen.
- **Repository im DataModels-Projekt:** Da nur drei Projekte vorgesehen sind, liegt die Persistenz im Datenmodell-Projekt und nicht in einem eigenen Projekt.
- **Synchronisierter Dateizugriff:** Ein lock im Repository verhindert Konflikte, wenn mehrere Sitzungen gleichzeitig schreiben.
- **Tolerantes Verhalten bei fehlenden Dateien:** Eine fehlende Komponentendatei führt in der Dokumentansicht zu einer Markierung statt zu einem Fehler.

4 Projektergebnisse

4.1 Soll-Ist-Vergleich

Alle in der Projektbeschreibung genannten Anforderungen wurden umgesetzt:

Anforderung	Status
Blazor Server 8, C#, Bootstrap	erfüllt
Drei Projekte, Namespaces, src/-Struktur mit .sln-Datei	erfüllt
JSON.NET, eine Datei je Freigabeversion, lesbare Dateinamen	erfüllt
Komponentenfreigabe in mehreren Systemfreigaben ohne Kopie	erfüllt
CRUD für System- und Komponentenfreigaben	erfüllt
Dokumentartige Ansicht, Komponenten als Links	erfüllt
Kein Benutzer- und Rechtemanagement	erfüllt (bewusst nicht umgesetzt)

4.2 Abweichungen und Anpassungen

Inhaltlich gab es keine Abweichungen. Der tatsächliche Stundenaufwand wurde nicht erfasst, ein Vergleich mit den geplanten 80 Stunden ist daher nicht möglich. Das Erscheinungsbild ist bewusst schlicht gehalten und nutzt die Standardgestaltung von Bootstrap.

4.3 Qualitätskontrolle

Die Funktionen wurden wie in Kapitel 3.6 beschrieben getestet. Eine formale Abnahme durch den Projektleiter steht noch aus.

4.4 Fazit

Die Anwendung erfüllt die gestellten Anforderungen und ermöglicht es, Systemfreigaben und Komponentenfreigaben digital zu erfassen und anzuzeigen. Die Entscheidung für Referenzen statt Kopien und für lesbare Dateinamen hat sich als einfache und nachvollziehbare Lösung erwiesen. Für künftige Versionen bieten sich folgende Erweiterungen an: automatisierte Tests, ein Export (z. B. als SBOM oder PDF), eine Benutzer- und Rechteverwaltung sowie eine Suche bei der Komponentenauswahl, falls die Zahl der Komponenten stark wächst.

5 Verzeichnisse und Anhang

5.1 Start der Anwendung

Die Anwendung wird mit folgendem Befehl im Hauptordner des Projekts gestartet und ist danach unter <http://localhost:5226> erreichbar:

```
dotnet run --project src/Karmaknit.SystemReleaseEditor.UI --launch-profile http
```

5.2 Abkürzungsverzeichnis

Abkürzung	Bedeutung
CRUD	Create, Read, Update, Delete
MVVM	Model-View-ViewModel
JSON	JavaScript Object Notation
UI	User Interface
SBOM	Software Bill of Materials

5.3 Literaturverzeichnis

- Microsoft: ASP.NET Core Blazor Dokumentation, <https://learn.microsoft.com/aspnet/core/blazor>
- CommunityToolkit.Mvvm, <https://learn.microsoft.com/dotnet/communitytoolkit/mvvm/>
- Newtonsoft.Json, <https://www.newtonsoft.com/json>
- Bootstrap, <https://getbootstrap.com>