



Abschlussprüfung Sommer 2025

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

## **Anwendung zur Erfassung standardisierter Daten aus Ordnerstrukturen in einer Excel-Datei**

Abgabedatum: Kassel, den 30.04.2025

**Prüfungsbewerber:**

Domenic Jäger  
Theodor-Heuss-Allee 7  
34225 Baunatal

**Ausbildungsbetrieb:**

cdemy GmbH  
Richard-Roosen-Straße 9  
34123 Kassel



## Inhalt

Abbildungsverzeichnis.....	4
Tabellenverzeichnis.....	4
Abkürzungsverzeichnis.....	5
1 Einleitung .....	6
1.1    Projektumfeld .....	6
1.2    Projektbeschreibung.....	6
1.3    Projektziel.....	6
1.4    Projektschnittstellen .....	7
1.5    Projektabgrenzung .....	7
2 Projektplanung .....	8
2.1    Projektphasen .....	8
2.2    Entwicklungsprozess.....	8
2.3    Ressourcenplanung .....	9
3 Analyse .....	10
3.1    Ist-Analyse .....	10
3.2    Wirtschaftlichkeitsanalyse .....	10
3.3    Projektkosten .....	11
3.4    Amortisierung .....	11
3.5    Anwendungsfälle.....	12
3.6    Anforderungen an die Anwendung .....	12
4 Entwurf.....	13
4.1    Zielplattform .....	13
4.2    Architekturdesign.....	13
4.3    Entwurf der Benutzeroberfläche .....	13
4.4    Entwurf der Datenmodelle .....	14
4.5    Qualitätssicherung.....	15
5 Implementierung.....	16
5.1    Struktur der Benutzeroberfläche.....	16
5.2    State-Management und Navigation .....	16
5.3    Layout und Benutzerführung .....	16
5.4    Entwicklung der Logik .....	16
5.5    Pathfinder.....	17
5.6    Regel-Logik.....	17
5.7    Excel-Generierung .....	18

5.8 Tests .....	18
6 Abnahme .....	19
7 Fazit .....	19
7.1 Soll-Ist-Vergleich .....	19
7.2 Lessons Learned.....	19
7.3 Ausblick.....	20
Eidesstattliche Erklärung .....	21
Anhang.....	22
A.1 Quellenverzeichnis .....	22
A.2 Pflichtenheft.....	23
A.3 Use Case-Diagramm .....	24
A.4 Mockup der drei Hauptscreens .....	25
A.5 Klassenmodel Regel.....	26
A.6 Kanban-Board .....	27
A.7 Darstellung Main Screen .....	28
A.8 Darstellung Preview Screen.....	29
A.9 Darstellung Rule Editor Screen.....	30
A.10 Codebeispiel GUI 1.....	31
A. 11 Codebeispiel GUI 2.....	32
A. 12 Codebeispiel Pathfinder.....	33
A. 13 Codebeispiel Rule Class.....	34
A.14 Codebeispiel SimpleRegexRule.....	35
A. 15 Codebeispiel Rule Type Enum .....	36
A. 16 Codebeispiel Excel Exporter .....	37
A.17a Beispiel Unit Test SimpleRegExRule 1 .....	38
A.17b Beispiel Unit Test SimpleRegexRule Fortsetzung .....	39

## Abbildungsverzeichnis

Abbildung 1: Use Case-Diagramm .....	24
Abbildung 2: Mockup der drei Hauptscreens .....	25
Abbildung 3: Klassenmodel Regel .....	26
Abbildung 4: Kanban-Board.....	27
Abbildung 5: Darstellung Main Screen.....	28
Abbildung 6: Darstellung Preview Screen.....	29
Abbildung 7: Darstellung Rule Editor Screen .....	30
Abbildung 8: Codebeispiel Main Screen .....	31
Abbildung 9: Codebeispiel Directories List.....	32
Abbildung 10: Pathfinder Klasse.....	33
Abbildung 11: Codebeispiel Abstrakte Rule Klasse .....	34
Abbildung 12: Codebeispiel SimpleRegexRule .....	35
Abbildung 13: Codebeispiel Rule Type Enum.....	36
Abbildung 14: Codebeispiel Excel Exporter .....	37
Abbildung 15: Codebeispiel Unit Test.....	38
Abbildung 16: Codebeispiel Unit Test Fortsetzung .....	39

## Tabellenverzeichnis

Tabelle 1: Projektphasen .....	8
Tabelle 2: Projektkosten .....	11
Tabelle 3: Ersparnis pro Kunde .....	11
Tabelle 4: Pflichtenheft .....	23

## Abkürzungsverzeichnis

EXPATH – Name der in dieser Projektdokumentation beschriebenen Software  
FIAE – Fachinformatiker für Anwendungsentwicklung  
Git – Versionsverwaltungssystem  
IDE – Integrated Development Environment (in diesem Fall Visual Studio Code)  
IHK – Industrie- und Handelskammer  
IP – Intellectual Property (Geistiges Eigentum)  
OS – Operating System  
PC – Personal Computer  
RegEx – Regular Expression (Regulärer Ausdruck)  
SaaS – Software as a Service  
UI – User Interface (Benutzeroberfläche)  
xlsx – Excel-Dateiformat  
FIAE – Fachinformatiker

## 1 Einleitung

In dieser Projektdokumentation beschreibe ich den Ablauf meines Abschlussprojekts für die IHK Kassel/Marburg, welches ich im Rahmen meiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt habe.

### 1.1 Projektumfeld

Die Arbeit an diesem Projekt erfolgte bei der cdemy GmbH mit Hauptsitz in Kassel. Auftraggeber des Projekts ist die GSI Office Management GmbH mit Hauptsitz in München.

Die cdemy GmbH fungiert sowohl als Dienstleister, der Software im Auftrag Dritter und für den eigenen Gebrauch entwickelt, als auch als Bildungsträger für angehende Fachinformatiker:innen.

Die GSI Office Management GmbH ist ein IT-Unternehmen, das sich auf die Entwicklung und den Support von Softwarelösungen zur Verwaltung gewerblicher Schutzrechte (Designs, Marken, Patente etc.) spezialisiert hat.

Die GSI betreibt unter anderem eine cloudbasierte Software-as-a-Service-Lösung namens „IP-Now“, welche ihren Kunden ein umfassendes Tool zum IP Management bietet.

Sie ermöglicht die effektive Verwaltung von Akten, Dokumenten, Fristen und Formbriefen und verfügt über zusätzliche Funktionen wie Aufgabenmanagement, Benachrichtigungen oder Chats.

Durch ihre Skalierbarkeit ist „IP Now“ sowohl für kleinere als auch größere Unternehmen interessant, die ihre Schutzrechte verwalten müssen.

### 1.2 Projektbeschreibung

Bei der Aufnahme von Neukunden, die IP Now nutzen möchten, müssen wichtige Dokumente wie beispielsweise Fallakten, welche sich auf den lokalen Systemen der Kunden befinden, erfasst werden. Dazu benötigt IP Now einige wichtige Standardinformationen zu diesen Dokumenten, welche jedoch von Kunde zu Kunde unterschiedlich abgelegt und benannt sein können.

Außendienstmitarbeiter: innen erfassen diese Daten derzeit in einem aufwendigen Prozess vor Ort beim Kunden.

Die von mir im Rahmen dieser Abschlussprojekts entwickelte Anwendung EXPATH soll Außendienstmitarbeiter: innen der GSI dabei unterstützen, diesen Prozess erheblich zu beschleunigen.

### 1.3 Projektziel

EXPATH soll ermöglichen, die Dokumente in der Ordnerstruktur des Kunden zu erfassen und aus den Pfadbezeichnungen die gewünschten Informationen zu extrahieren.

Anhand von anpassbaren Regeln sollen bestimmte Textmuster innerhalb der Dokumentenpfade auffindbar sein, um beispielsweise die Bezeichnung oder das Erstellungsdatum der Fallakte zu erfassen.

Da IP NOW die Daten aus einer Excel-Datei heraus weiterverarbeiten kann, besteht ein weiteres Ziel der Anwendung darin, am Ende des Prozesses eine Excel-Datei zu erstellen, welche die extrahierten Daten bereitstellt.

#### 1.4 Projektschnittstellen

Als Versionsverwaltung habe ich Git eingesetzt.

Zur erweiterten Verwaltung und Planung nutze ich den Onlinedienst GitHub, unter anderem durch die Einrichtung eines Kanban-Boards zur Aufgabenorganisation.

Die Auswahl des Wurzelverzeichnis sowie die Erstellung der Excel-Datei erfolgen über externe Dart-Packages.

Die Benutzer:innen der Anwendung sind ausschließlich Außendienstmitarbeiter:innen der GSI.

Die Kontrolle der Codequalität sowie die Kommunikation mit der GSI erfolgten über meinen Ausbilder bei der cdemy GmbH.

#### 1.5 Projektabgrenzung

Meine Aufgabe im Rahmen dieses Projekts umfasst die vollständige Planung und Umsetzung einer Windows-Anwendung.

Dies beinhaltet die Konzeption und Entwicklung sowohl der grafischen Benutzeroberfläche als auch der Geschäftslogik und der Datenstruktur.

Ebenso gehört das Testen der Anwendung zu meinem Aufgabenbereich.

## 2 Projektplanung

In dieser Projektdokumentation beschreibe ich den Ablauf meines Abschlussprojekts für die IHK Kassel/Marburg, welches ich im Rahmen meiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt habe.

### 2.1 Projektphasen

Die Zeitplanung sah einen Umfang von insgesamt 80 Stunden vor.

Die Durchführung des Projekts begann am 31.03.2025 und endete am 11.04.2025. Die Arbeiten fanden während der regulären Arbeitszeit bei der cdemy GmbH statt.

Die Einteilung der Projektphasen mit den geplanten Zeiteinsparungen stellte sich wie folgt dar:

*Tabelle 1: Projektphasen*

Analyse	
Erstellen eines Pflichtenhefts	5 Stunden
Wirtschaftlichkeitsanalyse	3 Stunden
Entwurf	
Planung der Benutzeroberfläche	5 Stunden
Planung der Datenmodelle	7 Stunden
Entwicklung	
Entwicklung der Benutzeroberfläche	18 Stunden
Entwicklung der Funktionalität	22 Stunden
Test und Abnahme	
Tests	3 Stunden
Abnahme	1 Stunde
Dokumentation	
Dokumentation schreiben	16 Stunden
<b>GESAMT</b>	<b>80 Stunden</b>

### 2.2 Entwicklungsprozess

Um die Aufgabenverwaltung effizient zu gestalten und mich an den Prinzipien des agilen Softwareentwicklungsprozesses zu orientieren, kam ein Kanban-Board auf GitHub zum Einsatz. Die Aufteilung in viele kleinere Aufgaben, zusammen mit den Bearbeitungsständen „Backlog“, „Ready“, „In progress“, „In review“ und „Done“, erwiesen sich als effektiver Ansatz, um das Projekt übersichtlich zu halten und flexibel auf



eventuelle Änderungen reagieren zu können.

## 2.3 Ressourcenplanung

Die Entwicklung von EXPATH erfolgte in den Räumlichkeiten der cdemy GmbH. Gearbeitet wurde an einem PC mit Windows 11 als Betriebssystem. Zum Schreiben des Programmcodes, sowie zum Debugging und Kompilieren kam die Entwicklungsumgebung Visual Studio Code zum Einsatz. EXPATH wird auf GitHub gehostet und lokal mit Git verwaltet.

Zur Umsetzung des Projekts wurde die Programmiersprache Dart gewählt, in Kombination mit dem plattformübergreifenden UI-Framework Flutter.

Dart wurde ursprünglich von Google entwickelt und eignet sich besonders für die Erstellung moderner Anwendungen sowohl im Web- als auch im Desktop- und Mobile-Umfeld.

Dart ist plattformunabhängig und kann auf Windows, MacOS, Linux, Webbrowsern und mobilen Endgeräten ausgeführt werden, was langfristige Flexibilität ermöglicht. Dank Hot Reload können Änderungen am Code sofort sichtbar gemacht werden, ohne dass die Anwendung neu gestartet werden muss. Dart bietet eine klare, objektorientierte Syntax mit Features wie Null-Sicherheit, asynchroner Programmierung, Extension Methods und Mixins, die eine strukturierte und wartbare Codebasis fördern.

Flutter, das Framework für die Benutzeroberfläche von EXPATH, basiert vollständig auf Dart. Dadurch wird eine besonders enge Verzahnung zwischen Anwendungslogik und Benutzeroberfläche ermöglicht.

Die Wahl von Dart und Flutter für dieses Projekt wurde aus mehreren Gründen getroffen:

- Die enge Verzahnung mit Flutter erlaubt die plattformübergreifende Entwicklung effizienter Desktop-Anwendungen, wodurch eine spätere Portierung auf andere Systeme wie MacOS grundsätzlich möglich ist.
- Die moderne Sprachstruktur von Dart unterstützt eine saubere, wartbare und erweiterbare Projektarchitektur.
- Zudem konnte ich auf bereits vorhandene Erfahrungen im Umgang mit Dart aufbauen, sodass keine zusätzliche Einarbeitungszeit in die Sprache notwendig war.

## 3 Analyse

### 3.1 Ist-Analyse

Derzeit müssen die Außendienstmitarbeiter: innen der GSI in einem aufwendigen Prozess die Daten bei neuen Kunden manuell erfassen. Dafür müssen die jeweiligen Benennungskonventionen der Kunden bekannt sein, und Ordnerstrukturen müssen häufig händisch nachvollzogen werden. Zudem ist ein gewisses Maß an Kenntnissen im Umgang mit Excel erforderlich, um beispielsweise bestimmte Teilstrings aus Dateipfaden zu extrahieren.

Hier setzt EXPATH an:

Einmal mit den Kunden abgestimmte Benennungskonventionen können in einem Regelsatz gespeichert werden.

Ausgehend von einem oder mehreren Wurzelverzeichnis sollen anschließend alle relevanten Informationen automatisiert extrahiert werden.

### 3.2 Wirtschaftlichkeitsanalyse

Der bisherige Ablauf zur Datenerfassung bei neuen Kunden ist zeitaufwendig und fehleranfällig.

Außendienstmitarbeiter:innen müssen für jedes Projekt erneut die Ablagestrukturen und Konventionen nachvollziehen – ein Vorgang, der zudem manuell durchgeführt wird.

Nur Personen mit entsprechender Excel-Erfahrung könne aktuell zur Datenerfassung eingesetzt werden.

Die Anwendung soll:

- Zeit sparen
- Fehlerquellen reduzieren
- Folgeaufwände minimieren

Zusätzlich ermöglicht sie eine verbesserte Auslastung des Außendienstes und reduziert indirekte Kosten z.B. durch entfallende Excel-Schulungen.

Die Investition amortisiert sich über Einsparungen bei Bearbeitungszeit und Korrekturaufwänden.

### 3.3 Projektkosten

Die geschätzten Projektkosten gliedern sich wie folgt:

*Tabelle 2: Projektkosten*

Vorgang	Mitarbeiter	Zeit (h)	Stundensatz	Kosten
Planung und Entwicklung	Auszubildender	80	20,00€	1.600,00€
Fachgespräch mit Kunden	Ausbilder	3	80,00€	240,00€
Hilfestellung	Ausbilder	4	80,00€	320,00€
Abnahmetest	Ausbilder	1	80,00€	80,00€
<b>GESAMT</b>				<b>2.240,00€</b>

Berechnungsgrundlagen:

- Durchschnittsgehalt Auszubildender: [1.255,00€/Monat](#)
- Durchschnittsgehalt FIAE (Ausbilder) [5.667,00€/Monat](#)
- Sozialversicherungsanteil des Arbeitgebers (19,7%)
- Gemeinkosten (100%)
- Werte wurden zur Übersicht gerundet (Auszubildender: 20€/h, Ausbilder: 80€/h).

### 3.4 Amortisierung

Es wird angenommen, dass die Anwendung den Zeitaufwand bei der Datenerfassung um ca. 20% reduziert. Außendienstmitarbeiter: innen (FIAE) mit einem durchschnittlichen Bruttogehalt von [4.353,00 €/Monat](#) (Quelle siehe Anhang) verursachen inklusive Arbeitgeberanteil und Gemeinkosten, einen Stundensatz von rund 60,00€.

*Tabelle 3: Ersparnis pro Kunde*

Szenario	Zeit(h)	Stundensatz	Kosten
Vor Einführung von EXPATH	8,0	60,00€	480,00€
Mit Einsatz von EXPATH	6,4	60,00€	384,00€
<b>Ersparnis pro Kunde</b>			<b>96,99€</b>

Die Amortisierungsrechnung sieht dann wie folgt aus:

Kosten / Ersparnis pro Fall = Einsätze bis zur Amortisierung

2.240,00€ / 96,00€ = 23,33

Fazit: Die Anwendung amortisiert sich nach 24 Einsätzen.

### 3.5 Anwendungsfälle

Die folgenden Kernprozesse beschreiben die zentralen Anwendungsfälle:

- 1) Benutzer: in wählt ein oder mehrere Wurzelverzeichnisse für die Analyse aus.
- 2) Benutzer: in definiert Regeln zu Mustererkennung; Regeln können gespeichert, geladen, bearbeitet und gelöscht werden.
- 3) EXPATH durchsucht die ausgewählten Verzeichnisse anhand der Regeln und extrahiert relevante Informationen
- 4) Benutzer: in prüft die extrahierten Daten in einer tabellarischen Vorschau.
- 5) Ergebnisse werden als Excel-Datei exportiert und gespeichert.

Ein Use-Case-Diagramm, das diese Abläufe visualisiert, befindet sich im [Anhang A.3](#).

### 3.6 Anforderungen an die Anwendung

Für EXPATH wurden folgende funktionale und nicht-funktionale Anforderungen definiert. Sie ergeben sich aus den Zielvorgaben des Auftraggebers sowie den typischen Anforderungen des Arbeitsalltags bei der Erfassung von Aktenstrukturen durch Außendienstmitarbeitende.

#### Funktionale Anforderungen

- Auswahl eines oder mehrerer Verzeichnisse als Ausgangspunkt für die Analyse
- Definition anpassbarer Regeln zur Extraktion relevanter Informationen aus Dateipfaden
- Erstellung, Bearbeitung und Wiederherstellung von Regelsätzen
- Vorschauansicht der extrahierten Daten in Tabellenform
- Export der Daten in eine Excel-Datei im .xlsx-Format
- Möglichkeit zur An- und Abwahl einzelner Datensätze in der Vorschau

#### Nicht-funktionale Anforderungen

- Intuitive und benutzerfreundliche Bedienoberfläche
- Stabile Performance auch umfangreichen Verzeichnisstrukturen
- Flexible Regeldefinition zur Abbildung unterschiedlichster Kundenverzeichnisse
- Erweiterbare Architektur (z.B. neue Regeltypen)
- Primäre Zielplattform ist Windows
- Technische Basis soll künftige Portierung auf MacOS und Linux ermöglichen

Ein Pflichtenheft befindet sich im Anhang [A.2](#).

## 4 Entwurf

### 4.1 Zielplattform

Da der Anwendungsfall vorsieht, dass die EXPATH von Außendienstmitarbeiter:innen an einem Desktop-System genutzt wird, fiel die Wahl der Zielpattform auf Windows.

Durch die Auswahl von Flutter und Dart als technologischer Basis wurde darauf geachtet, EXPATH künftig mit relativ geringem Aufwand auch für MacOS oder Linux portieren zu können.

Eine mobile Version ist nicht vorgesehen, da der Auftraggeber keine mobile Datenerfassung wünscht.

### 4.2 Architekturdesign

EXPATH basiert auf einer Zwei-Schichten-Architektur, bei der die Präsentationsschicht und die Geschäftslogik klar voneinander getrennt sind:

#### 1) Präsentationsschicht

Diese Schicht ermöglicht die Interaktion der Benutzer:innen mit dem System.

Hier werden Verzeichnisse ausgewählt, Regeln verwaltet, eine Vorschau der Ergebnisse angezeigt und der Export der Daten angestoßen.

Im Projekt ist die Präsentationsschicht im Verzeichnis *lib/gui* organisiert und umfasst den Hauptbildschirm sowie zwei Nebenscreens (Regel-Editor und Vorschau).

#### 2) Geschäftslogik

Diese Schicht stellt die eigentliche Programmlogik dar.

Hier werden Verzeichnisse eingelesen, Regeln verarbeitet, Konfigurationen gespeichert und geladen sowie der Export von Daten in Excel-Dateien durchgeführt.

Die Logikschicht ist im Verzeichnis *lib/logic* untergebracht und in Module wie *filesystem*, *rules* und *excel* unterteilt.

Eine zusätzliche Datenschicht ist nicht erforderlich, da alle Daten nur temporär verarbeitet und anschließend exportiert werden.

### 4.3 Entwurf der Benutzeroberfläche

Ziel des Benutzeroberflächen-Entwurfs war es, eine übersichtliche, intuitive und funktionale Benutzeroberfläche zu gestalten, die den Außendienstmitarbeiter:innen eine einfache Bedienung ermöglicht.

Die Benutzeroberfläche ist in drei Hauptbereiche unterteilt:

#### 1. Hauptbildschirm

Er stellt die zentrale Arbeitsfläche dar und gliedert sich in zwei vertikale Bereiche:

##### a) Verzeichniswahl (oberer Bereich): Anzeige der aktuell ausgewählten

Wurzelverzeichnisse, Hinzufügen neuer Verzeichnisse über eine Dateiauswahl sowie Löschen bestehender Einträge. Außerdem findet sich hier der Button für die Navigation zum Vorschau-Screen.

##### b) Regelverwaltung (unterer Bereich): Auflistung der definierten Regeln für die aktuellen



Wurzelverzeichnisse. Möglichkeit neue Regeln anzulegen (führt zum Rule-Editor-Screen), bestehende Regeln zu bearbeiten oder zu löschen sowie Regelsätze zu speichern und zu laden.

## 2. Rule-Editor-Screen

Ermöglicht die Erstellung oder Bearbeitung individueller Regeln.  
Der Aufbau gliedert sich in:

- Dropdown-Menü zur Auswahl des Regeltyps
- Dynamisch angepasste Eingabefelder (z.B. für ein RegEx-Muster)
- Buttons zum Speichern oder Abbrechen.

Beim Bearbeiten bestehender Regeln sind Felder bereits vorausgefüllt.

## 3. Preview-Screen

Darstellung der extrahierten Informationen in einer tabellarischen Vorschau. Jede Zeile enthält eine Checkbox, um einzelne Datensätze vom Export auszuschließen.  
Standardmäßig sind alle Datensätze angewählt.

Ein Export-Button startet den Export der Daten in eine Excel-Datei.

Die Benutzeroberfläche folgt einem klaren und einfachen Bedienkonzept:

- Jede Funktion ist über Buttons oder Listen erreichbar.
- Aktionen werden durch Snackbar-Nachrichten rückgemeldet
- Fehlerhafte Eingaben oder Warnungen werden benutzerfreundlich angezeigt.

Ein Mockup der Benutzeroberfläche befindet sich in [Anhang A.4](#).

## 4.4 Entwurf der Datenmodelle

Ziel des Datenmodellentwurfs war es, eine modulare und erweiterbare Struktur zu schaffen.

Die wichtigsten Datenmodelle sind:

### 1. Rule (abstrakt)

Definiert das Interface für alle Regeltypen. Legt zentrale Methoden wie *apply()*, *toJson()* und *fromJson()* sowie gemeinsame Attribute fest.

### 2. SimpleRegexRule

Spezialisiert auf die Anwendung regulärer Ausdrücke (RegEx) auf Dateipfade.

### 3. PathSegmentRule

Extrahiert ein bestimmtes Segment eines Dateipfades basierend auf seiner Position.

4. ReversePathSegmentRule Extrahiert ein Pfadsegment rückwärts vom Ende des Pfades aus betrachtet.

Die Regelmodellierung basiert auf dem Strategy Pattern, wodurch Regeltypen flexibel

austauschbar sind. Die abstrakte Klasse *Rule* enthält gemeinsame Schnittstellenmethoden sowie Attribute wie *type* (Regeltyp) und *excelField* (Spaltenname). Die spezialisierten Regelklassen *PathSegmentRule* und *ReversePathSegmentRule* extrahieren Pfadsegmente basierend auf Indexwerten, suchen also letztlich einen Ordernamen. Hierbei sucht ersteres von vorne (den ersten, zweiten, dritten Ordner usw.), letzterer von hinten (den letzten, vorletzten, drittletzten Ordner usw.). *SimpleRegexRule* wendet allgemein einen regulären Ausdruck als Regel und bietet dem Nutzer dadurch viele Freiheiten in der Mustersuche.

Darüber hinaus verfügen Regeln über die *toJson()* Methode über die Option, in ein speicherbares Map-Objekt umgewandelt zu werden. Die *fromJson()* Methode wiederum erzeugt ein Regelobjekt aus einer gespeicherten Map.

Durch diese Mechanismen können Konfigurationen gespeichert und zu einem späteren Zeitpunkt wiederhergestellt werden.

Die Architektur ist bewusst so gestaltet, dass neue Regeltypen ohne tiefgreifende Änderungen an bestehenden Klassen integriert werden können (Open-Closed-Prinzip).

Ein Klassendiagramm zur Regelmodellierung befindet sich in [Anhang A.5](#).

#### 4.5 Qualitätssicherung

Für jede Aufgabe, die im Kanban-Board auf Github eingetragen wurde, ist ein separater Branch erstellt worden. Änderungen wurden nur dann in den main-Branch überführt, wenn sie mit dem Ausbilder besprochen und überprüft worden sind.

Auffälligkeiten und Fehler wurden gemeinsam mit dem Ausbilder identifiziert und behoben.

Ein Screenshot des Kanban-Boards befindet sich in [Anhang A.6](#).

## 5 Implementierung

### 5.1 Struktur der Benutzeroberfläche

Das Frontend von EXPATH wurde vollständig mit dem Framework Flutter umgesetzt. Im Widget-Prinzip von Flutter ist jedes Element als Widget definiert – vom kleinsten Textfeld bis zum kompletten Bildschirm.

Ein Widget ist eine Baueinheit, welche die Darstellung oder das Verhalten eines Elements beschreibt. In Flutter wird die Benutzeroberfläche durch das Verschachteln und Kombinieren von Widgets aufgebaut.

- Dynamische Widgets reagieren auf Benutzerinteraktionen oder Zustandsveränderungen
- Statische Widgets stellen sichtbare Elemente wie Textfelder oder Buttons dar.

Durch dieses Prinzip können auch komplexe Oberflächen leicht gewartet und erweitert werden.

In dieser Anwendung wurden eigene Widgets entwickelt, um bestimmte Funktionalitäten zu kapseln und die Wiederverwendbarkeit zu fördern.

Aufbau des Frontend:

Main Screen	Zentrale Startseite der Anwendung: Verwaltet Verzeichnisse und Regeln
Rule Editor Screen	Erstellung und Anpassung individueller Regeln
Preview Screen	Vorschau der extrahierten Daten, Start des Excel-Exports.

Jeder dieser Screens ist in weitere Widgets untergliedert, die jeweils eine klar definierte Aufgabe übernehmen.

Abbildungen der Hauptscreens befinden sich in den Anhängen [A.7](#), [A.8](#) und [A.9](#).  
Ausgewählte Codebeispiele befinden sich in den Anhängen [A.10](#) und [A.11](#)

### 5.2 State-Management und Navigation

Für die Zustandsverwaltung wurde das setState-Prinzip von Flutter verwendet. Für die Navigation zwischen den Screens kommt das klassische Navigator-Konzept (Navigator.push, Navigator.pop) zum Einsatz.

Daten wie aktuelle Verzeichnisse und Regeldefinitionen werden innerhalb des App-Zustands verwaltet und bei Screen-Wechseln übergeben.

### 5.3 Layout und Benutzerführung

Die grafische Benutzeroberfläche ist schlicht und funktional gehalten.

- Listenansichten sorgen für Übersichtlichkeit
- Standard-Buttons gewährleisten eine intuitive Bedienung
- Dialogfenster und Pop-ups unterstützen Eingaben und grenzen Aktionen klar ab.
- Erfolgreiche oder fehlerhafte Aktionen werden über SnackBar kommuniziert

### 5.4 Entwicklung der Logik

Im Zentrum der Anwendung steht die Implementierung der Logik, die das Auswählen von



Verzeichnissen, die Anwendungen von Regeln sowie den Export der extrahierten Daten ermöglicht. Wert wurde auf Modularität, Wiederverwendbarkeit und Erweiterbarkeit gelegt.

### 5.5 Pathfinder

Die Auswahl und Verarbeitung der Verzeichnisse erfolgt über die Klasse Pathfinder. Diese kapselt den rekursiven Zugriff auf das lokale Dateisystem und nutzt Funktionen aus dem Dart-Standardpaket *dart:io*.

Das Attribut `rootDirectoryPath` dient als Startpunkt der Suche und die Methode `getAllFilePaths()` gibt eine Liste aller Dateipfade innerhalb des Wurzelverzeichnisses zurück. Intern ruft `getAllFilePaths()` rekursiv die Methode `scanDirectory()` auf, um alle Dateien zu finden zur Liste hinzuzufügen.

Codebeispiel für die Klasse Pathfinder befindet sich in [Anhang A.12](#)

### 5.6 Regel-Logik

Die zentrale Funktionalität von EXPATH ist die Definition und Anwendung von Regeln, um relevante Informationen aus Dateipfaden zu extrahieren.

Das Regelsystem basiert auf dem Strategy Pattern.

Basisklasse:

- Rule definiert gemeinsame Schnittstelle für Regeltypen (*apply()*, *toJson()*, *fromJson()*).

Konkrete Regeltypen

- SimpleRegexRule: Durchsucht Pfade mithilfe regulärer Ausdrücke.
- PathSegmentRule: Extrahiert Ordnersegmente basierend auf ihrer Position im Pfad.
- ReversePathSegmentRule: Extrahiert Ordnersegmente vom Ende des Pfades.

Die Auswahl des Regeltyps erfolgt über ein eigenes Enum `RuleType`.

Dadurch ist EXPATH flexibel und neue Regeltypen können einfach ergänzt werden.

Benutzer können Regeln dynamisch erstellen und auf Pfade anwenden.

Ein Codebeispiel für die Rule Klasse, die SimpleRegexRule sowie das Enum Rule Type befindet sich in den Anhängen [A.13](#), [A.14](#) und [A.15](#).



## 5.7 Excel-Generierung

Das Ziel der Anwendung ist die Erstellung einer Datei im xlsx-Format, die die extrahierten Informationen enthält.

Diese Funktion ist im excel-export-Modul implementiert. Dabei wird auf das Dart-Paket excel zurückgegriffen.

Der Ablauf:

1. Erstellen eines neuen Excel-Dokuments.
2. Anlegen von Spalten anhand der Regeldefinition (z.B. Aktennummer, Erfinder, usw.).
3. Zeilenweises Befüllen der Tabelle mit den extrahierten Werten.
4. Speichern der Datei über die saveFile()-Methode des file-picker-Pakets, mit Auswahl des Speicherorts und Dateinamen.

Ein Codebeispiel für den Excel Exporter findet sich in [Anhang A.16](#)

## 5.8 Tests

Zur Qualitätssicherung wurden begleitend Unit-Tests erstellt. Die Tests prüfen die Korrektheit einzelner Programmbestandteile unabhängig vom Gesamtsystem. Besonders getestet wurden die Modellklassen im Bereich der Regelverarbeitung, da diese den fachlichen Kern der Anwendung bilden.

Die Tests wurden mithilfe des Flutter-Test-Frameworks *flutter-test* implementiert.

Ein Beispiel für einen Unit-Test befindet sich in den Anhängen [A.17a](#) und [A.17b](#).

## 6 Abnahme

Eine gesonderte Abnahme durch den Auftraggeber konnte während der Projektlaufzeit noch nicht erfolgen, da die Anwendung zum Zeitpunkt der Dokumentation noch nicht offiziell vorgestellt wurde. Der Fortschritt des Projekts wurde jedoch regelmäßig mit meinem Ausbilder besprochen, der in engem Kontakt mit dem Auftraggeber steht.

Durch diese enge Abstimmung konnte sichergestellt werden, dass die Anforderungen des Auftraggebers korrekt umgesetzt wurden und eventuelle Änderungswünsche frühzeitig einfließen konnten.

## 7 Fazit

### 7.1 Soll-Ist-Vergleich

Ziel des Projekt war es eine Anwendung zu entwickeln, die Außendienstmitarbeiter:innen der GSI bei der Erfassung von Daten aus lokalen Ordnerstrukturen unterstützt und diese Daten in ein einheitliches Format für die Weiterverarbeitung in IP Now überführt.

Im Soll-Zustand waren folgende Hauptanforderungen definiert:

- Auswahl von Verzeichnissen
- Definition von anpassbaren Regeln zur Informationsgewinnung
- Vorschau der extrahierten Daten
- Export der Daten in eine Excel-Datei (xlsx-Format)
- Benutzerfreundliche Bedienung unter Windows

Im Ist-Zustand wurden alle gesetzten Anforderungen vollständig umgesetzt:

- Verzeichnisse können ausgewählt und rekursiv eingelesen werden.
- Regeln lassen sich flexibel erstellen, speichern und laden
- Die Anwendung der Regeln erfolgt zuverlässig und zeigt eine Vorschau der Ergebnisse
- Der Export erfolgt im kompatiblen Excel-Format
- Die Benutzeroberfläche wurde funktional und übersichtlich gestaltet

Insgesamt entspricht das Projektergebnis vollständig den definierten Anforderungen.

### 7.2 Lessons Learned

Im Verlauf der Projektarbeit konnten zahlreiche praktische Erfahrungen gesammelt werden.

Die Nutzung von GitHub als Versionsverwaltung und Aufgabenmanagement-Tool hat sich als sehr hilfreich erwiesen, um auch in Einzelarbeit den Überblick zu bewahren. Der Einsatz von Flutter für eine Desktop-Anwendung unter Windows brachte spezifische Herausforderungen mit sich, beispielsweise beim Dateizugriff auf Ordner mit eingeschränkten Zugriffsrechten. Diese wurden jedoch erfolgreich gelöst. Die bewusste Entscheidung für eine modulare Architektur (z.B. Zwei-Schichten-Architektur, Strategy-Pattern) erleichterte die Implementierung und wirkte sich positiv auf die Wartbarkeit und Erweiterbarkeit der Anwendung aus. Die Implementierung von Unit-Tests verdeutlichte die Bedeutung von frühzeitiger Fehlererkennung und Testabdeckung.



### 7.3 Ausblick

EXPATH erfüllt zwar alle definierten Projektziele, bietet jedoch Potenzial für künftige Erweiterungen. Denkbar wäre hier die Implementierung weiterer vorgebauter Regeltypen, die Erweiterung der Exportformate beispielsweise in CSV-Dateien, eine Portierung auf Linux oder MacOS, oder eine Verbesserung der Benutzeroberfläche, etwa durch Filter-Funktionen.

Diese Optionen zeigen, dass EXPATH eine solide Basis darstellt, auf der zukünftige Erweiterungen und Anpassungen problemlos aufbauen können.

## Eidesstattliche Erklärung

Ich, Domenic Jäger, versichere hiermit, dass ich meine Dokumentation zur betrieblichen Projektarbeit mit dem Thema

„Anwendung zur Erfassung standardisierter Daten aus Ordnerstrukturen in einer Excel-Datei“

selbstständig verfasst und keine anderen als angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Kassel, den 29.04.2025

## Anhang

### A.1 Quellenverzeichnis

Durchschnittliche Ausbildungsvergütung für Auszubildende Fachinformatiker in Westdeutschland 2024 laut Bundesinstitut für Berufsbildung:

<https://www.bibb.de/de/12209.php>

Entgelt für den Beruf Fachinformatiker für Anwendungsentwicklung (Region Hessen):

<https://web.arbeitsagentur.de/entgeltatlas/beruf/13659?region=9>

Entgelt für den Beruf Fachinformatiker für Anwendungsentwicklung (Region Bayern):

<https://web.arbeitsagentur.de/entgeltatlas/beruf/13659?region=12>

## A.2 Pflichtenheft

Allgemeine Informationen	
Projektname	EXPATH
Ziel	Entwicklung einer Windows-Anwendung zur Analyse von lokalen Ordnerstrukturen und zum Export extrahierter Informationen
Auftraggeber	GSI Office Management GmbH
Ausführender	Domenic Jäger (cdemy GmbH)
Zielnutzer: innen	Außendienstmitarbeitende der GSI
Plattform	Windows 10/11
Funktionale Anforderungen	
F1	Verzeichnisse könne ausgewählt und eingelesen werden
F2	Benutzer: innen können Regeln definieren, bearbeiten, speichern und laden
F3	Regeln durchsuchen Pfade nach konfigurierbaren Mustern (z.B. RegEx)
F4	Die Anwendung zeigt eine Vorschau der extrahierten Informationen
F5	Der Export der Daten erfolgt in eine .xlsx-Datei (Excel-kompatibel)
F6	Einzelne Datensätze können vom Export ausgeschlossen werden
Nicht funktionale Anforderungen	
N1	Die Benutzeroberfläche ist intuitiv bedienbar und erfordert keine Einarbeitung
N2	Die Anwendung arbeitet stabil auch mit umfangreichen Ordnerstrukturen
N3	Die Regeldefinition ist flexibel, um unterschiedliche Kundenstrukturen abzubilden
N4	Die Softwarearchitektur erlaubt Erweiterungen (z.B. neue Regeltypen)
N5	Die Anwendung ist für Windows ausgelegt, weitere Plattformen sollen unterstützbar sein.
Abgrenzung	
Keine Benutzerverwaltung	
Kein Zugriff auf Netzwerkfreigaben oder Cloud-Systeme	
Keine automatisierte Integration in IP Now – nur Export als Excel-Datei	

Tabelle 4: Pflichtenheft

### A.3 Use Case-Diagramm

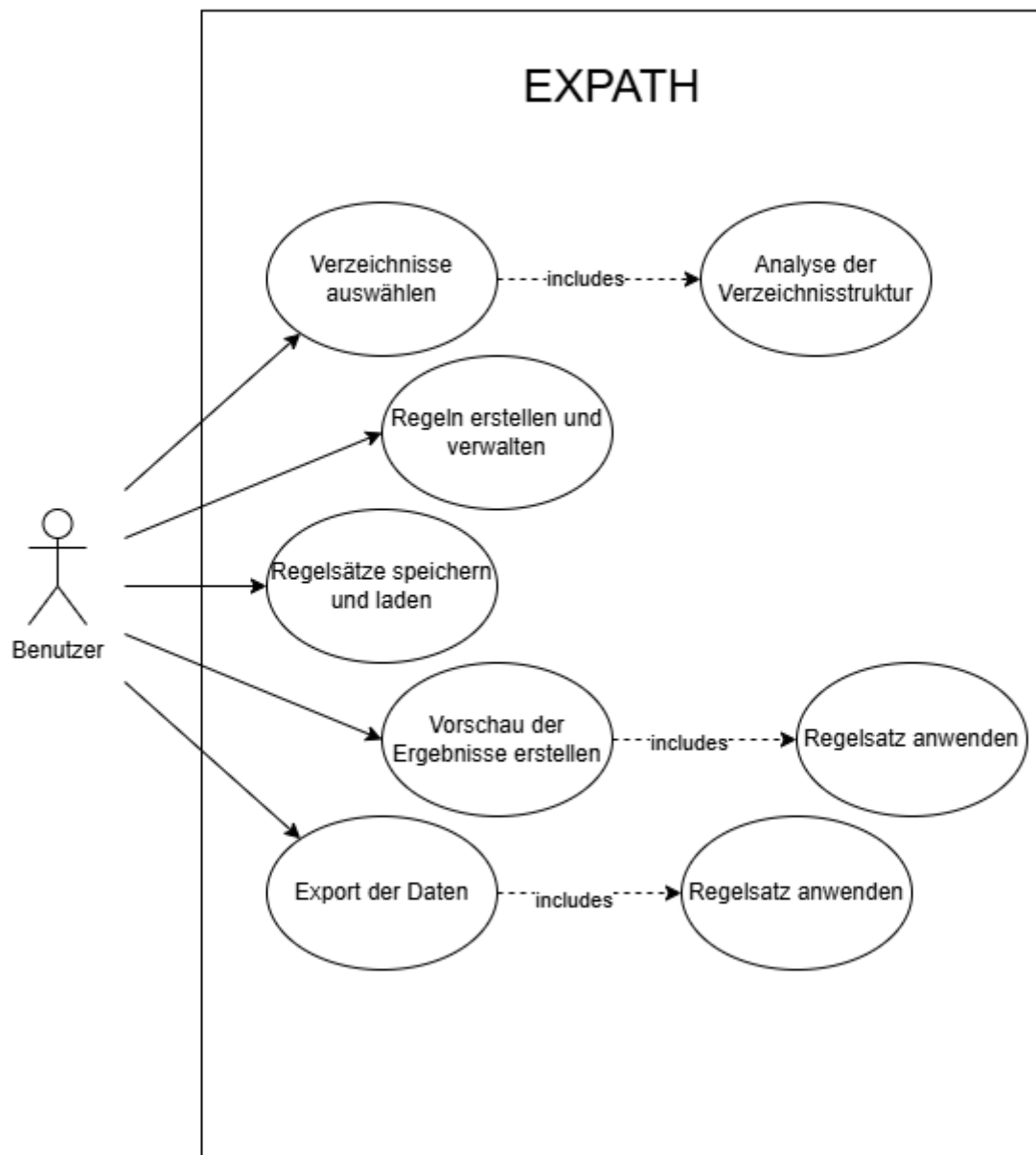


Abbildung 1: Use Case-Diagramm



A.4 Mockup der drei Hauptscreens

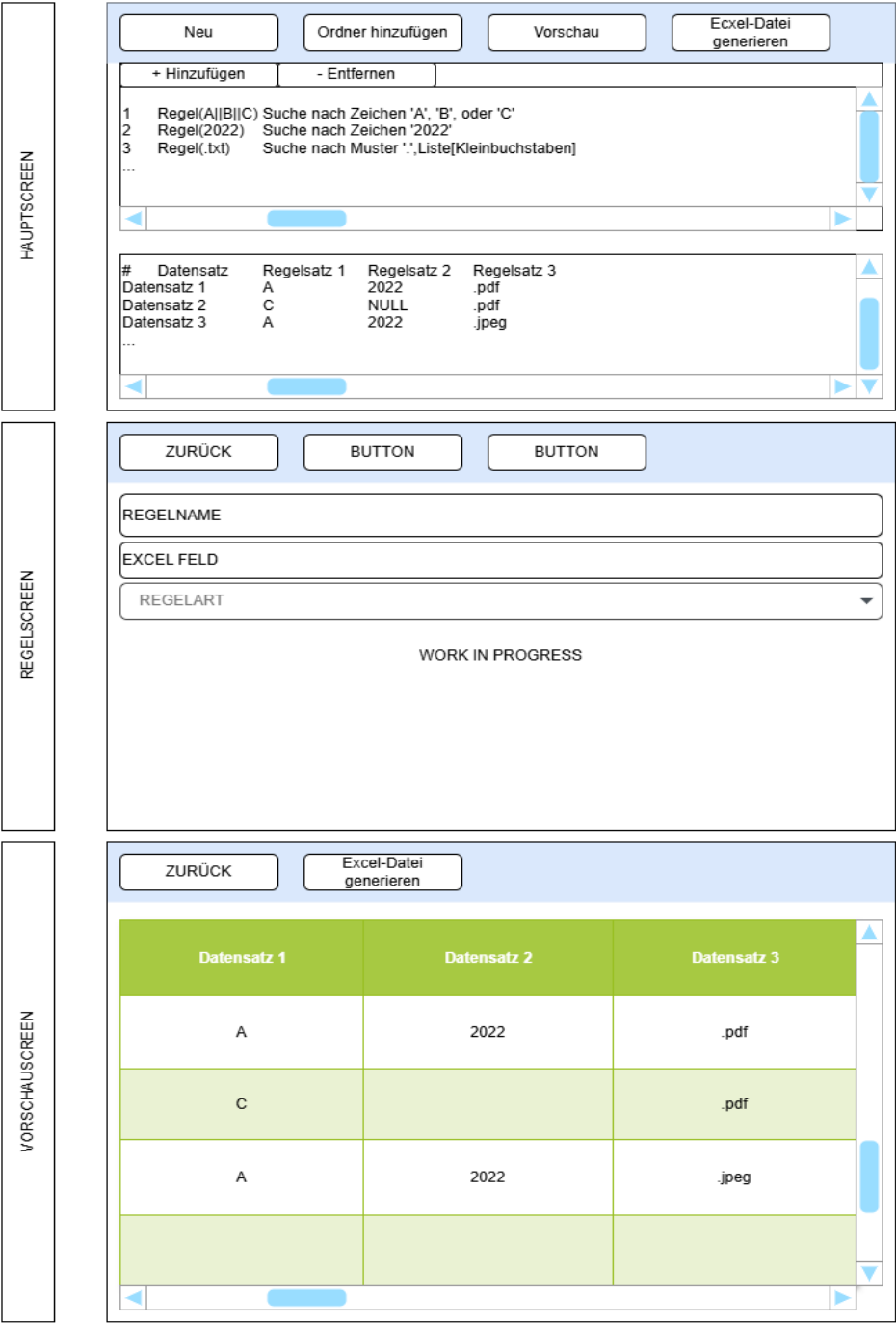


Abbildung 2: Mockup der drei Hauptscreens

### A.5 Klassenmodel Regel

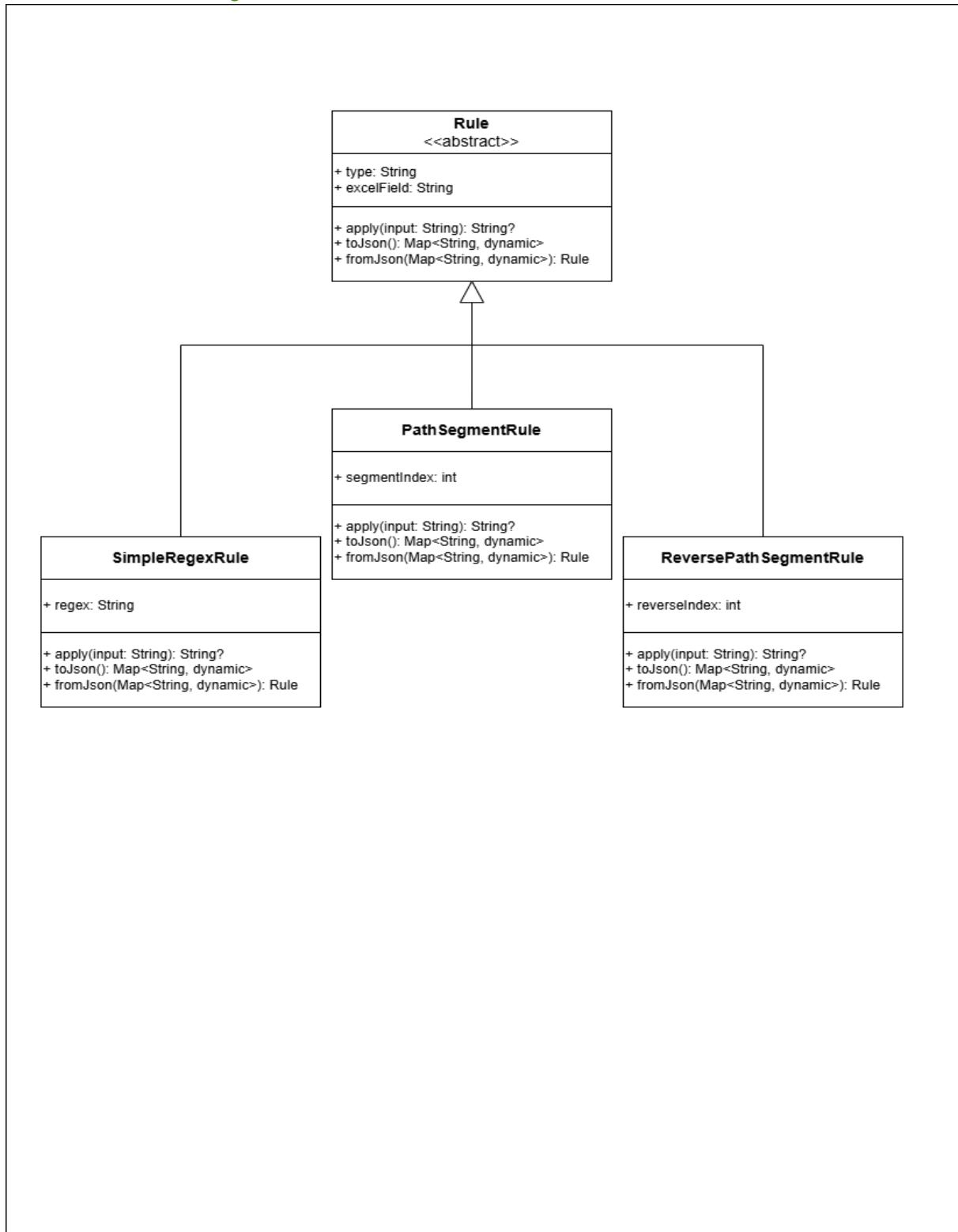


Abbildung 3: Klassenmodel Regel

## A.6 Kanban-Board

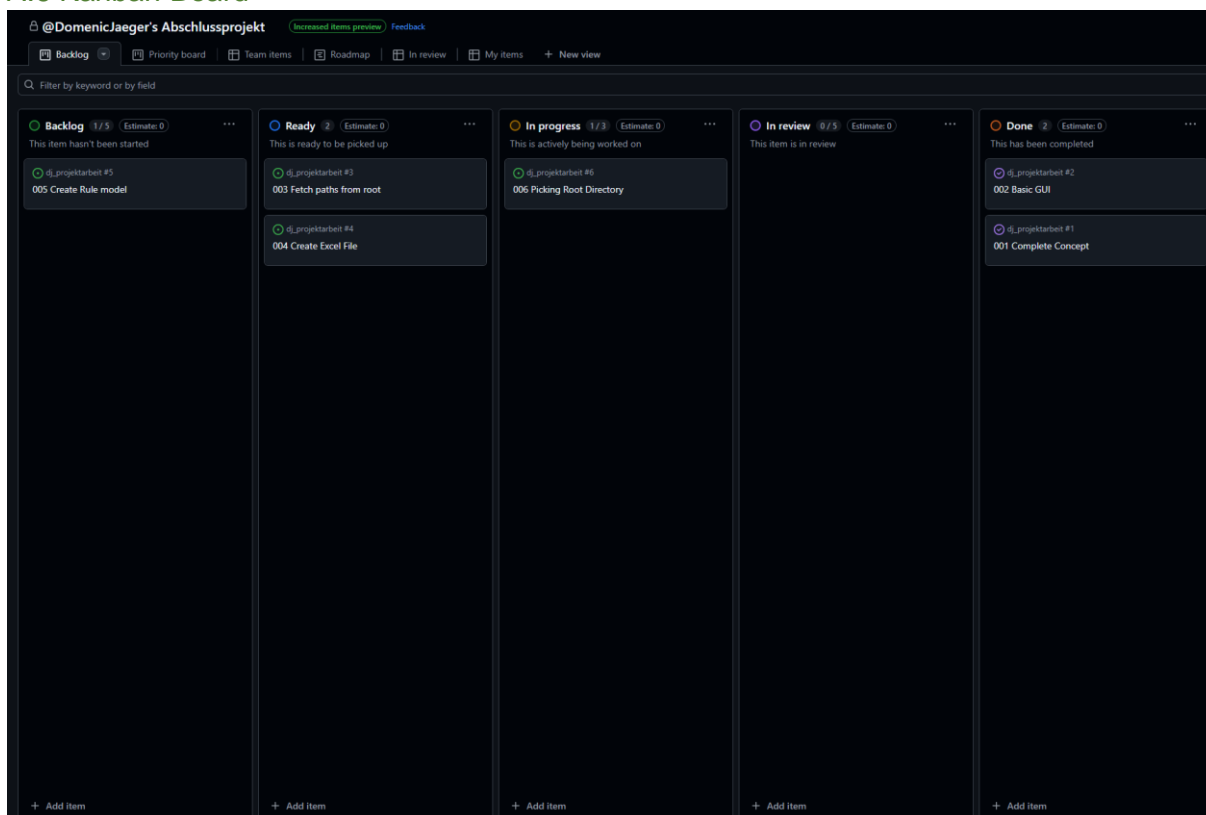


Abbildung 4: Kanban-Board



A.7 Darstellung Main Screen

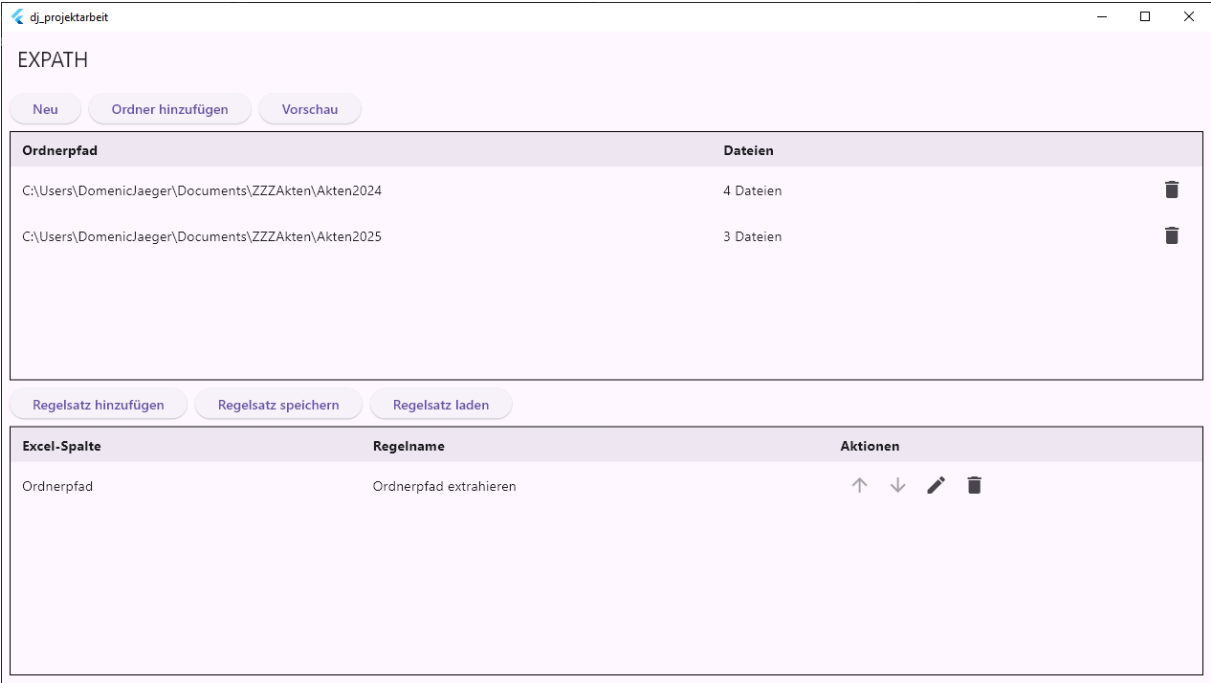


Abbildung 5: Darstellung Main Screen



A.8 Darstellung Preview Screen

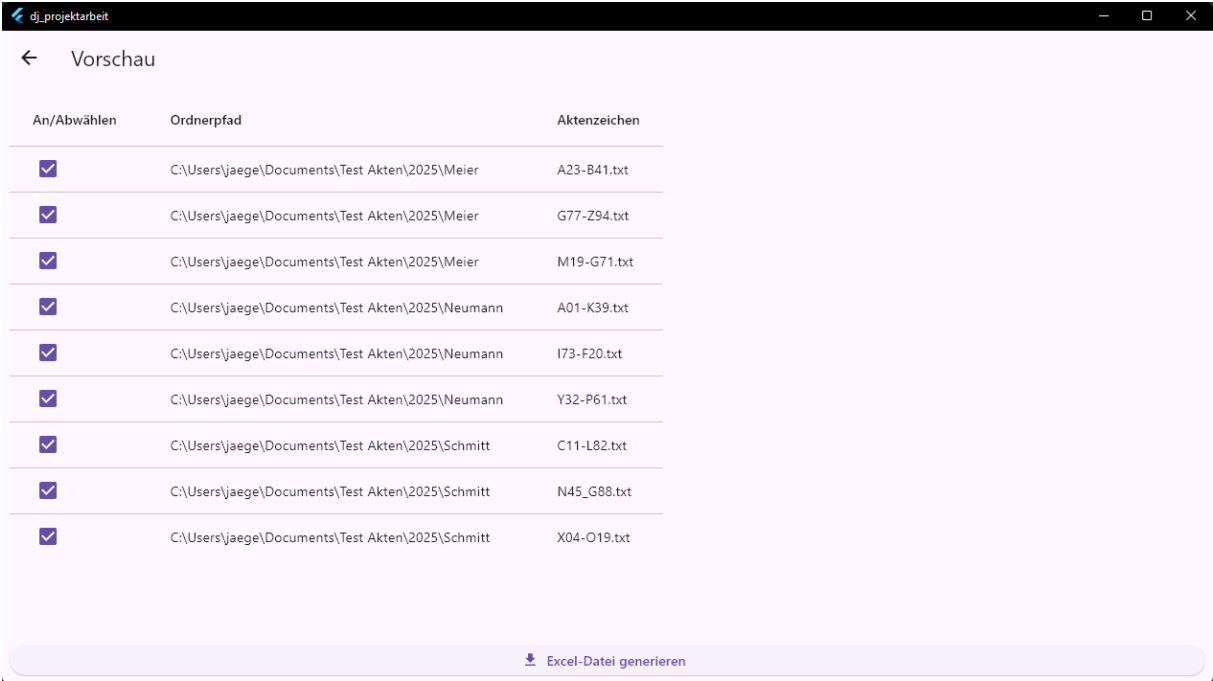
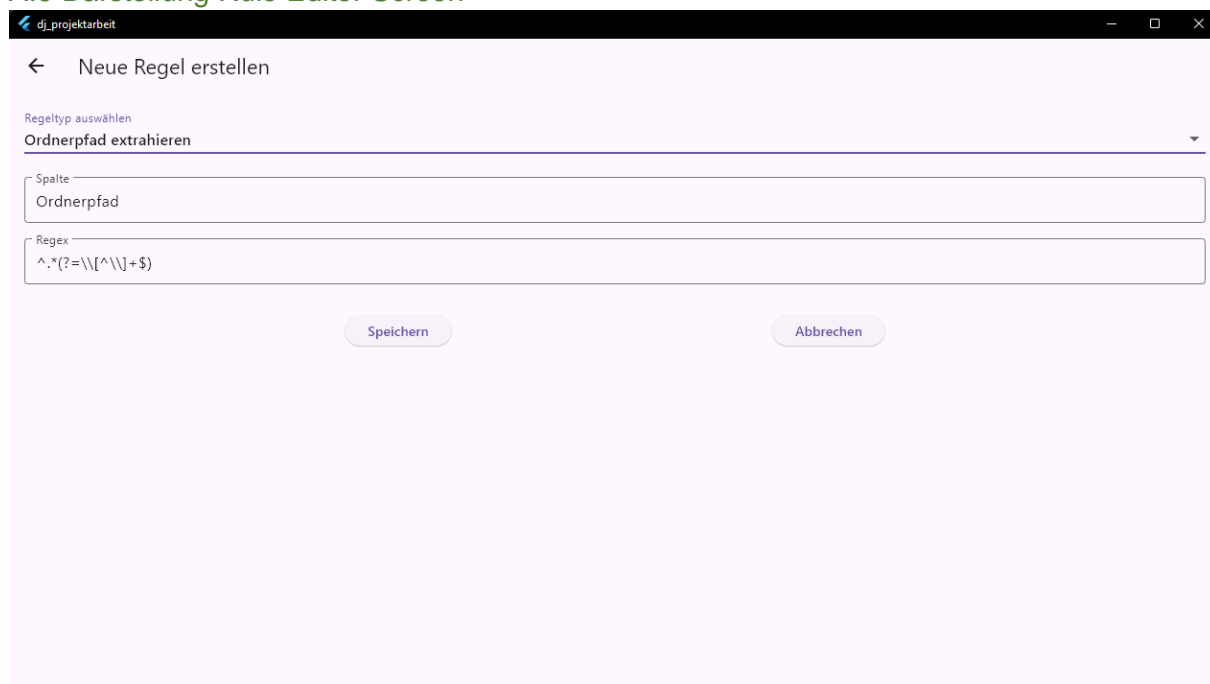


Abbildung 6: Darstellung Preview Screen

### A.9 Darstellung Rule Editor Screen



← Neue Regel erstellen

Regeltyp auswählen  
Ordnerpfad extrahieren


Spalte  
Ordnerpfad

Regex  
^.\*(?=\\\[^\]]+\$)

Speichern Abbrechen

Abbildung 7: Darstellung Rule Editor Screen

## A.10 Codebeispiel GUI 1



```
lib > gui > main_screen > main_screen.dart > ...
12
13 | /// MainScreen of the Application You, 5 hours ago • Uncommitted change
14 | CdemyLeitung, 3 weeks ago | 1 author (CdemyLeitung)
15 | class MainScreen extends StatefulWidget {
16 |   const MainScreen({super.key});
17 |
18 |   @override
19 |   State<MainScreen> createState() => _MainScreenState();
20 | }
21 |
22 | Domenic, 12 hours ago | 2 authors (CdemyLeitung and one other)
23 | class _MainScreenState extends State<MainScreen> {
24 |   List<RootDirectoryEntry> directories = [];
25 |   List<Rule> rules = [];
26 |
27 |   @override
28 |   Widget build(BuildContext context) {
29 |     return Scaffold(
30 |       appBar: AppBar(
31 |         title: Text('EXPATH'),
32 |       ), // AppBar
33 |       body: Padding(
34 |         padding: const EdgeInsets.all(8.0),
35 |         child: Column(
36 |           crossAxisAlignment: CrossAxisAlignment.stretch,
37 |           children: [
38 |             // --- Directory related buttons -----
39 |             Row(
40 |               children: [
41 |                 // "New" button (resets directories list)
42 |                 ElevatedButton(
43 |                   onPressed: _clearDirectories,
44 |                   child: Text("Neu"),
45 |                 ), // ElevatedButton
46 |                 SizedBox(width: 8),
47 |                 // "Add directory" button
48 |                 ElevatedButton(
49 |                   onPressed: _selectDirectory,
50 |                   child: Text("Ordner hinzufügen"),
51 |                 ), // ElevatedButton
52 |                 SizedBox(width: 8),
53 |                 // "Preview" button
54 |                 ElevatedButton(
55 |                   onPressed: _preview,
56 |                   child: Text("Vorschau"),
57 |                 ), // ElevatedButton
58 |               ],
59 |             ), // Row
60 |             SizedBox(height: 8),
61 |             // --- Directory list -----
62 |             Expanded(
63 |               child: DirectoriesList(
64 |                 directories: directories,
65 |                 removeDirectory: _removeDirectory,
66 |               ), // DirectoriesList
67 |             ), // Expanded
68 |             SizedBox(height: 8),
```

Abbildung 8: Codebeispiel Main Screen

## A. 11 Codebeispiel GUI 2

```

Help
main_screen.dart • directories_list.dart M X
lib > gui > main_screen > _widgets > directories_list.dart > ...
4 class DirectoriesList extends StatelessWidget {
5   final List<RootDirectoryEntry> directories;
6   final Function(RootDirectoryEntry) removeDirectory;
7   const DirectoriesList({
8     required this.directories,
9     required this.removeDirectory,
10    super.key,
11  });
12
13  @override
14  Widget build(BuildContext context) {
15    return Container(
16      decoration: BoxDecoration(
17        border: Border.all(),
18      ), // BoxDecoration
19      child: directories.isEmpty
20        ? Center(child: Text('Noch keine Verzeichnisse hinzugefügt.'))
21        : Column(
22          crossAxisAlignment: CrossAxisAlignment.stretch,
23          children: [
24            Container(
25              padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 8),
26              // ignore: deprecated_member_use
27              color: Theme.of(context).colorScheme.secondary.withOpacity(0.1),
28              child: Row(
29                children: [
30                  Expanded(flex: 6, child: Text('Ordnerpfad', style: TextStyle(fontWeight: FontWeight.bold))),
31                  Expanded(flex: 2, child: Text('Dateien', style: TextStyle(fontWeight: FontWeight.bold))),
32                  Expanded(flex: 2, child: SizedBox()),
33                ],
34              ), // Row
35            ), // Container
36            Expanded(
37              child: ListView.builder(
38                itemCount: directories.length,
39                itemBuilder: (context, index) {
40                  final dir = directories[index];
41                  return Padding(
42                    padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 4),
43                    child: Row(
44                      children: [
45                        Expanded(flex: 6, child: Text(dir.path)),
46                        Expanded(flex: 2, child: Text('${dir.fileCount} Dateien')),
47                        Expanded(
48                          flex: 2,
49                          child: Row(
50                            mainAxisAlignment: MainAxisAlignment.end,
51                            children: [
52                              IconButton(
53                                icon: Icon(Icons.delete),
54                                onPressed: () => removeDirectory(dir),
55                              ), // IconButton
56                            ],
57                          ), // Row
58                        ), // Expanded
59                      ],
60                    ), // Row
61                  ); // Padding
62                },

```

Abbildung 9: Codebeispiel Directories List



#### A. 12 Codebeispiel Pathfinder

```
5  class Pathfinder {
6      final String rootDirectoryPath;
7
8      Pathfinder(this.rootDirectoryPath);
9
10     List<String> getAllFilePaths() {
11         final directory = Directory(rootDirectoryPath);
12         if (!directory.existsSync()) {
13             throw Exception("Directory does not exist: $rootDirectoryPath");
14         }
15
16         final List<String> filePaths = [];
17
18         void scanDirectory(Directory dir) {
19             try {
20                 for (var entity in dir.listSync(recursive: false)) {
21                     if (entity is File) {
22                         filePaths.add(entity.path);
23                     } else if (entity is Directory) {
24                         scanDirectory(entity);
25                     }
26                 }
27             } on FileSystemException catch (e) {
28                 if (kDebugMode) {
29                     print('Zugriff verweigert oder Fehler bei ${dir.path}: ${e.message}');
30                 }
31             }
32         }
33
34         scanDirectory(directory);
35
36         return filePaths;
37     }
38 }
```

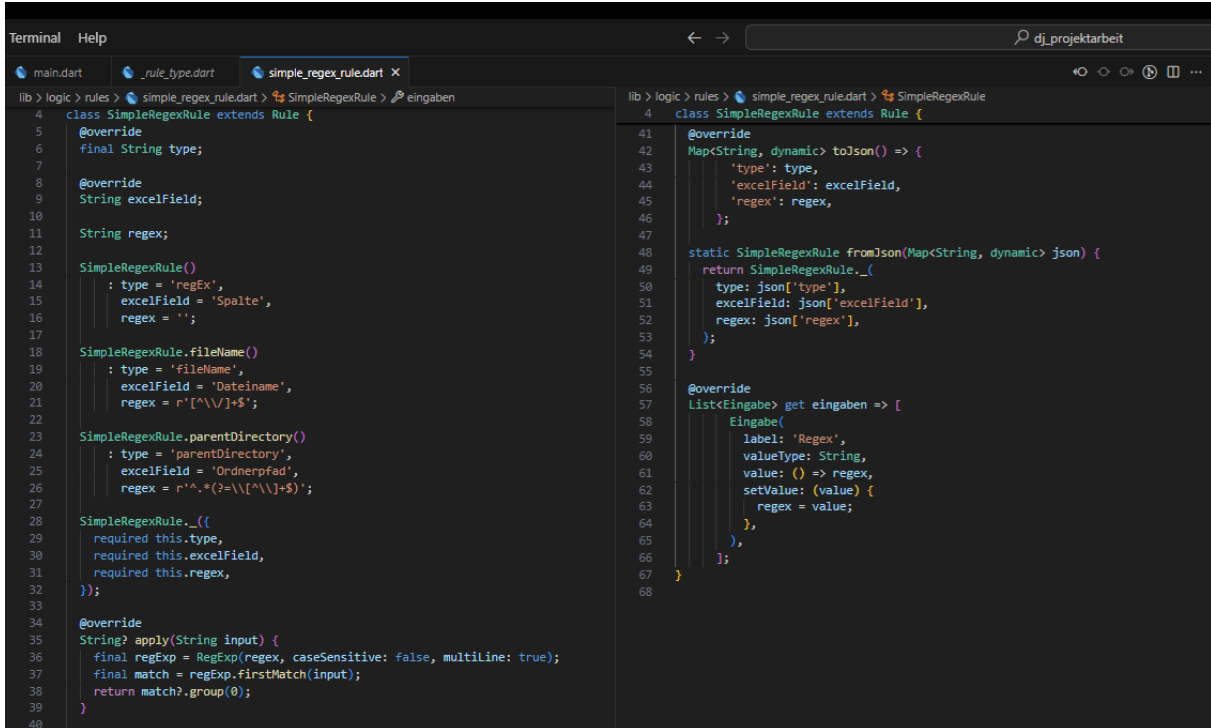
Abbildung 10: Pathfinder Klasse

### A. 13 Codebeispiel Rule Class

```
4  /// -----  
5  /// Abstract Rule  
6  /// -----  
7  
CdemyLeitung, 4 weeks ago | 1 author (CdemyLeitung)  
8  abstract class Rule {  
9      String get type;  
10     String get excelField;  
11     set excelField(String value);  
12     String? apply(String input);  
13     Map<String, dynamic> toJson();  
14  
15     RuleType get ruleType => RuleType.fromType(type);  
16     List<Eingabe> get eingaben;  
17 }
```

Abbildung 11: Codebeispiel Abstrakte Rule Klasse

## A.14 Codebeispiel SimpleRegexRule



```
lib > logic > rules > simple_regex_rule.dart > SimpleRegexRule > eingabe
4 class SimpleRegexRule extends Rule {
5   @override
6   final String type;
7
8   @override
9   String excelField;
10
11   String regex;
12
13   SimpleRegexRule()
14     : type = 'regex',
15       excelField = 'Spalte',
16       regex = '';
17
18   SimpleRegexRule.fileName()
19     : type = 'fileName',
20       excelField = 'Dateiname',
21       regex = '^[^\\V]+$';
22
23   SimpleRegexRule.parentDirectory()
24     : type = 'parentDirectory',
25       excelField = 'Ordnungspfad',
26       regex = '^[^\\V]*(?=[^\\V]+$)';
27
28   SimpleRegexRule._({
29     required this.type,
30     required this.excelField,
31     required this.regex,
32   });
33
34   @override
35   String? apply(String input) {
36     final regExp = RegExp(regex, caseSensitive: false, multiline: true);
37     final match = regExp.firstMatch(input);
38     return match?.group(0);
39   }
40
41
42 class SimpleRegexRule extends Rule {
43   @override
44   Map<String, dynamic> toJson() => {
45     'type': type,
46     'excelField': excelField,
47     'regex': regex,
48   };
49
50   static SimpleRegexRule fromJson(Map<String, dynamic> json) {
51     return SimpleRegexRule._(
52       type: json['type'],
53       excelField: json['excelField'],
54       regex: json['regex'],
55     );
56   }
57
58   @override
59   List<Eingabe> get eingabe => [
60     Eingabe(
61       label: 'Regex',
62       valueType: String,
63       value: () => regex,
64       setValue: (value) {
65         regex = value;
66       },
67     ),
68   ];
69 }
```

Abbildung 12: Codebeispiel SimpleRegexRule

## A. 15 Codebeispiel Rule Type Enum

```

10  enum RuleType {
11      fileName(
12          type: 'fileName',
13          label: 'Dateiname extrahieren',
14          constructor: SimpleRegexRule.fileName,
15          fromJson: SimpleRegexRule.fromJson,
16      ),
17      parentDirectory(
18          type: 'parentDirectory',
19          label: 'Ordnerpfad extrahieren',
20          constructor: SimpleRegexRule.parentDirectory,
21          fromJson: SimpleRegexRule.fromJson,
22      ),
23      pathSegment(
24          type: 'pathSegment',
25          label: 'Ordner an Position extrahieren',
26          hint: '0 = Partition, 1 = erster Ordner, 2 = zweiter Ordner, ...',
27          constructor: PathSegmentRule.new,
28          fromJson: PathSegmentRule.fromJson,
29      ),
30      reversePathSegment(
31          label: 'Ordner invertiert extrahieren',
32          hint: '0 = Datei, 1 = letzter Ordner, 2 = vorletzter Ordner, ...',
33          type: 'reversePathSegment',
34          constructor: ReversePathSegmentRule.new,
35          fromJson: ReversePathSegmentRule.fromJson,
36      ),
37      regex(
38          label: 'Benutzerdefinierter Regex',
39          type: 'regex',
40          constructor: SimpleRegexRule.new,
41          fromJson: SimpleRegexRule.fromJson,
42      );
43
44      const RuleType({
45          required this.type,
46          required this.label,
47          required this.constructor,
48          required this.fromJson,
49          this.hint,
50      });
51
52      final String type;
53      final String label;
54      final String? hint;
55      final Rule Function() constructor;
56      final Rule Function(Map<String, dynamic>) fromJson;
57
58      static RuleType fromType(String type) {
59          return RuleType.values.firstWhere(
60              (ruleType) => ruleType.type == type,
61              orElse: () => throw Exception('Unbekannter Regeltyp: $type'),
62          );
63      }
64  }

```

Abbildung 13: Codebeispiel Rule Type Enum

## A. 16 Codebeispiel Excel Exporter

```
7 class ExcelExporter {
8     static Future<void> export({
9         required List<RootDirectoryEntry> directories,
10        required List<Rule> rules,
11    }) async {
12        if (directories.isEmpty || rules.isEmpty) {
13            throw Exception("Keine Daten oder Regeln vorhanden.");
14        }
15
16        // Save dialog
17        final String? savePath = await FilePicker.platform.saveFile(
18            dialogTitle: 'Excel-Datei speichern',
19            fileName: 'Export.xlsx',
20        );
21
22        if (savePath == null) return;
23
24        final excel = Excel.createExcel();
25        final Sheet sheet = excel['Sheet1'];
26
27        // Header
28        final headers = [...rules.map((r) => r.excelField)];
29        for (int col = 0; col < headers.length; col++) {
30            sheet.cell(CellIndex.indexByColumnRow(columnIndex: col, rowIndex: 0)).value = headers[col];
31        }
32
33        // Input values
34        final allFiles = directories.expand((dir) => dir.filePaths).toList();
35        for (int i = 0; i < allFiles.length; i++) {
36            for (int j = 0; j < rules.length; j++) {
37                final rule = rules[j];
38                final result = rule.apply(allFiles[i]) ?? '';
39                sheet.cell(CellIndex.indexByColumnRow(columnIndex: j, rowIndex: i + 1)).value = result;
40            }
41        }
42
43        // Encode and save the Excel file
44        final bytes = excel.encode();
45        if (bytes != null) {
46            final file = File(savePath);
47            file.createSync(recursive: true);
48            file.writeAsBytesSync(bytes);
49        } else {
50            throw Exception("Fehler beim Erstellen der Excel-Datei.");
51        }
52    }
53 }
```

Abbildung 14: Codebeispiel Excel Exporter

## A.17a Beispiel Unit Test SimpleRegexRule 1

```
4 void main() {  
    Run | Debug  
    ✓ 5 test('SimpleRegexRule findet korrekten Treffer im Pfad', () {  
6         final rule = SimpleRegexRule();  
7         rule.regex = 'Patent_(\\d+)';  
8         final path = 'C:\\Daten\\Patent_12345\\Akte.pdf';  
9  
10        final result = rule.apply(path);  
11  
12        expect(result, 'Patent_12345');  
13    });  
14  
    Run | Debug  
    ✓ 15 test('SimpleRegexRule gibt null zurück, wenn kein Treffer', () {  
16        final rule = SimpleRegexRule();  
17        rule.regex = 'Marke_(\\d+)';  
18        final path = 'C:\\Daten\\Patent_12345\\Akte.pdf';  
19  
20        final result = rule.apply(path);  
21  
22        expect(result, isNull);  
23    });  
    Run | Debug  
    ▶ 24 test('SimpleRegexRule toJson gibt korrektes Map-Objekt zurück', () {  
25        final rule = SimpleRegexRule();  
26        rule.regex = 'Patent_(\\d+)';  
27        rule.excelField = 'Spalte';  
28        final expectedJson = {  
29            'type': 'simpleRegex',  
30            'excelField': 'Spalte',  
31            'regex': 'Patent_(\\d+)',  
32        };  
33  
34        expect(rule.toJson(), expectedJson);  
35    });
```

Abbildung 15: Codebeispiel Unit Test

## A.17b Beispiel Unit Test SimpleRegexRule Fortsetzung

```
Run | Debug
37 test('SimpleRegexRule fromJson erstellt korrektes Objekt', () {
38     final json = {
39         'type': 'simpleRegex',
40         'excelField': 'Spalte',
41         'regex': 'Patent_(\\d+)',
42     };
43
44     final rule = SimpleRegexRule.fromJson(json);
45
46     expect(rule.type, 'simpleRegex');
47     expect(rule.excelField, 'Spalte');
48     expect(rule.regex, 'Patent_(\\d+)');
49 });
50
Run | Debug
51 test('SimpleRegexRule toJson und fromJson sind zueinander konsistent', () {
52     final originalRule = SimpleRegexRule();
53     originalRule.regex = 'Patent_(\\d+)';
54     originalRule.excelField = 'Spalte';
55
56     final json = originalRule.toJson();
57     final newRule = SimpleRegexRule.fromJson(json);
58
59     expect(newRule.type, originalRule.type);
60     expect(newRule.excelField, originalRule.excelField);
61     expect(newRule.regex, originalRule.regex);
62 });
63 }
```

Abbildung 16: Codebeispiel Unit Test Fortsetzung