

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

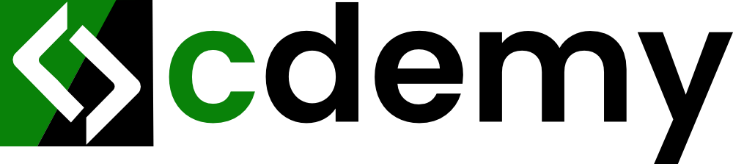
34225 Baunatal

**Ausbildungsbetrieb:**

​​cdemy GmbH​

Richard-Roosen-Straße 9

34123 Kassel



Inhalt

[Abbildungsverzeichnis 4](#_Toc196911878)

[Tabellenverzeichnis 4](#_Toc196911879)

[Abkürzungsverzeichnis 5](#_Toc196911880)

[1 Einleitung 6](#_Toc196911881)

[1.1 Projektumfeld 6](#_Toc196911882)

[1.2 Projektbeschreibung 6](#_Toc196911883)

[1.3 Projektziel 6](#_Toc196911884)

[1.4 Projektschnittstellen 7](#_Toc196911885)

[1.5 Projektabgrenzung 7](#_Toc196911886)

[2 Projektplanung 8](#_Toc196911887)

[2.1 Projektphasen 8](#_Toc196911888)

[2.2 Entwicklungsprozess 8](#_Toc196911889)

[2.3 Ressourcenplanung 9](#_Toc196911890)

[3 Analyse 10](#_Toc196911891)

[3.1 Ist-Analyse 10](#_Toc196911892)

[3.2 Wirtschaftlichkeitsanalyse 10](#_Toc196911893)

[3.3 Projektkosten 11](#_Toc196911894)

[3.4 Amortisierung 11](#_Toc196911895)

[3.5 Anwendungsfälle 12](#_Toc196911896)

[3.6 Anforderungen an die Anwendung 12](#_Toc196911897)

[4 Entwurf 13](#_Toc196911898)

[4.1 Zielplattform 13](#_Toc196911899)

[4.2 Architekturdesign 13](#_Toc196911900)

[4.3 Entwurf der Benutzeroberfläche 13](#_Toc196911901)

[4.4 Entwurf der Datenmodelle 14](#_Toc196911902)

[4.5 Qualitätssicherung 15](#_Toc196911903)

[5 Implementierung 16](#_Toc196911904)

[5.1 Struktur der Benutzeroberfläche 16](#_Toc196911905)

[5.2 State-Management und Navigation 16](#_Toc196911906)

[5.3 Layout und Benutzerführung 16](#_Toc196911907)

[5.4 Entwicklung der Logik 16](#_Toc196911908)

[5.5 Pathfinder 17](#_Toc196911909)

[5.6 Regel-Logik 17](#_Toc196911910)

[5.7 Excel-Generierung 18](#_Toc196911911)

[5.8 Tests 18](#_Toc196911912)

[6 Abnahme 19](#_Toc196911913)

[7 Fazit 19](#_Toc196911914)

[7.1 Soll-Ist-Vergleich 19](#_Toc196911915)

[7.2 Lessons Learned 19](#_Toc196911916)

[7.3 Ausblick 20](#_Toc196911917)

[Eidesstattliche Erklärung 21](#_Toc196911918)

[Anhang 22](#_Toc196911919)

[A.1 Quellenverzeichnis 22](#_Toc196911920)

[A.2 Pflichtenheft 23](#_Toc196911921)

[A.3 Use Case-Diagramm 24](#_Toc196911922)

[A.4 Mockup der drei Hauptscreens 25](#_Toc196911923)

[A.5 Klassenmodel Regel 26](#_Toc196911924)

[A.6 Kanban-Board 27](#_Toc196911925)

[A.7 Darstellung Main Screen 28](#_Toc196911926)

[A.8 Darstellung Preview Screen 29](#_Toc196911927)

[A.9 Darstellung Rule Editor Screen 30](#_Toc196911928)

[A.10 Codebeispiel GUI 1 31](#_Toc196911929)

[A. 11 Codebeispiel GUI 2 32](#_Toc196911930)

[A. 12 Codebeispiel Pathfinder 33](#_Toc196911931)

[A. 13 Codebeispiel Rule Class 34](#_Toc196911932)

[A.14 Codebeispiel SimpleRegexRule 35](#_Toc196911933)

[A. 15 Codebeispiel Rule Type Enum 36](#_Toc196911934)

[A. 16 Codebeispiel Excel Exporter 37](#_Toc196911935)

[A.17a Beispiel Unit Test SimpleRegExRule 1 38](#_Toc196911936)

[A.17b Beispiel Unit Test SimpleRegexRule Fortsetzung 39](#_Toc196911937)

Abbildungsverzeichnis

[Abbildung 1: Use Case-Diagramm 24](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911086)

[Abbildung 2: Mockup der drei Hauptscreens 25](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911087)

[Abbildung 3: Klassenmodel Regel 26](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911088)

[Abbildung 4: Kanban-Board 27](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911089)

[Abbildung 5: Darstellung Main Screen 28](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911090)

[Abbildung 6: Darstellung Preview Screen 29](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911091)

[Abbildung 7: Darstellung Rule Editor Screen 30](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911092)

[Abbildung 8: Codebeispiel Main Screen 31](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911093)

[Abbildung 9: Codebeispiel Directories List 32](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911094)

[Abbildung 10: Pathfinder Klasse 33](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911095)

[Abbildung 11: Codebeispiel Abstrakte Rule Klasse 34](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911096)

[Abbildung 12: Codebeispiel SimpleRegexRule 35](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911097)

[Abbildung 13: Codebeispiel Rule Type Enum 36](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911098)

[Abbildung 14: Codebeispiel Excel Exporter 37](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911099)

[Abbildung 15: Codebeispiel Unit Test 38](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911100)

[Abbildung 16: Codebeispiel Unit Test Fortsetzung 39](file:///C:\Users\DomenicJaeger\Documents\projects\dj_projektarbeit\docs\jaeger_domenic_abschlussprojekt_dokumentation.docx#_Toc196911101)

Tabellenverzeichnis

[Tabelle 1: Projektphasen 8](#_Toc196911106)

[Tabelle 2: Projektkosten 11](#_Toc196911107)

[Tabelle 3: Ersparnis pro Kunde 11](#_Toc196911108)

[Tabelle 4: Pflichtenheft 23](#_Toc196911109)

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. 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.
  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 sin 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.
  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.
  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 Wurzelverzeichnisses 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.
  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.

* 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 Zeitansätzen 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 |
| **GESAMT** | **80 Stunden** |

* 1. 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. 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

* 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 Wurzelverzeichnissen sollen anschließend alle relevanten Informationen automatisiert extrahiert werden.
  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 Außendiensts und reduziert indirekte Kosten z.B. durch entfallende Excel-Schulungen.  
     Die Investition amortisiert sich über Einsparungen bei Bearbeitungszeit und Korrekturaufwänden.
  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€ |
| **GESAMT** |  | | | **2.240,00€** |

Berechnungsgrundlagen:  
 - Durchschnittsgehalt Auszubildender: [1.255,00€/Monat](#Quelle1)  
 - Durchschnittsgehalt FIAE (Ausbilder) [5.667,00€/Monat](#Quelle2)  
 - Sozialversicherungsanteil des Arbeitgebers (19,7%)  
 - Gemeinkosten (100%)  
 - Werte wurden zur Übersicht gerundet (Auszubildender: 20€/h, Ausbilder: 80€/h).

* 1. 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](#Quelle3) €/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€ |
| **Ersparnis pro Kunde** |  |  | **96,99€** |

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.

* 1. 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](#_A.3_Use_Case-Diagramm).
  2. 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](#_A.2_Pflichtenheft).

4 Entwurf

* 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.
  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.
  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-Menu 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](#_A.4_Mockup_der).
  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 Ordnernamen. 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](#_A.5_Klassenmodel_Regel).
  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](#_A.6_Kanban-Board).

5 Implementierung

* 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 Widget 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.7_Darstellung_Main), [A.8](#_A.8_Darstellung_Preview) und [A.9](#_A.9_Darstellung_Rule).  
     Ausgewählte Codebeispiele befinden sich in den Anhängen [A.10](#_A.10_Codebeispiel_GUI) und [A.11](#_A._11_Codebeispiel)
  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 Verzeichnislisten und Regeldefinitionen werden innerhalb des App-Zustands verwaltet und bei Screen-Wechseln übergeben.
  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
  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. 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](#_A._12_Codebeispiel)
  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 übe 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 das Die Rule Klasse, die SimpleRegexRule sowie das Enum Rule Type befindet sich in den Anhängen [A.13](#_A._13_Codebeispiel), [A14](#_A.14_Codebeispiel_SimpleRegexRule) und [A.15](#_A._15_Codebeispiel).
  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](#_A._16_Codebeispiel)
  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](#_A.17a_Beispiel_Unit) und [A.17b](#_A.17b_Beispiel_Unit).

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

* 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.
  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.
  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

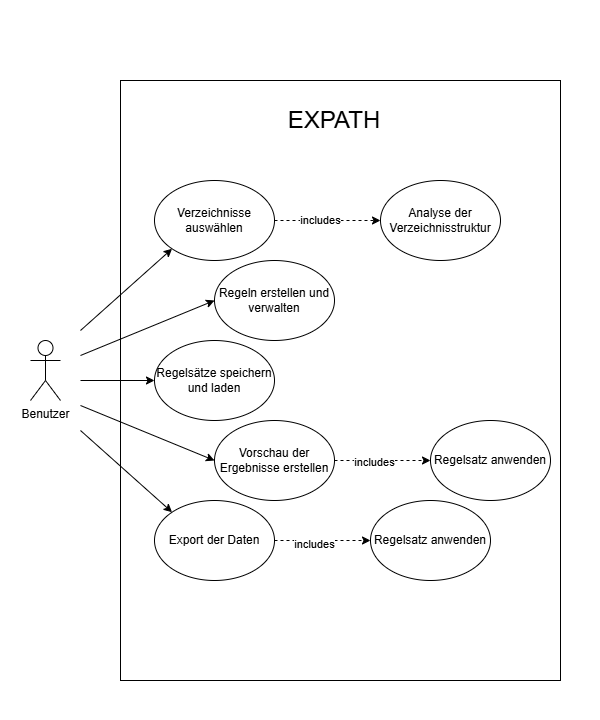


Abbildung : Use Case-Diagramm

## A.4 Mockup der drei Hauptscreens

Abbildung : Mockup der drei Hauptscreens

## A.5 Klassenmodel Regel

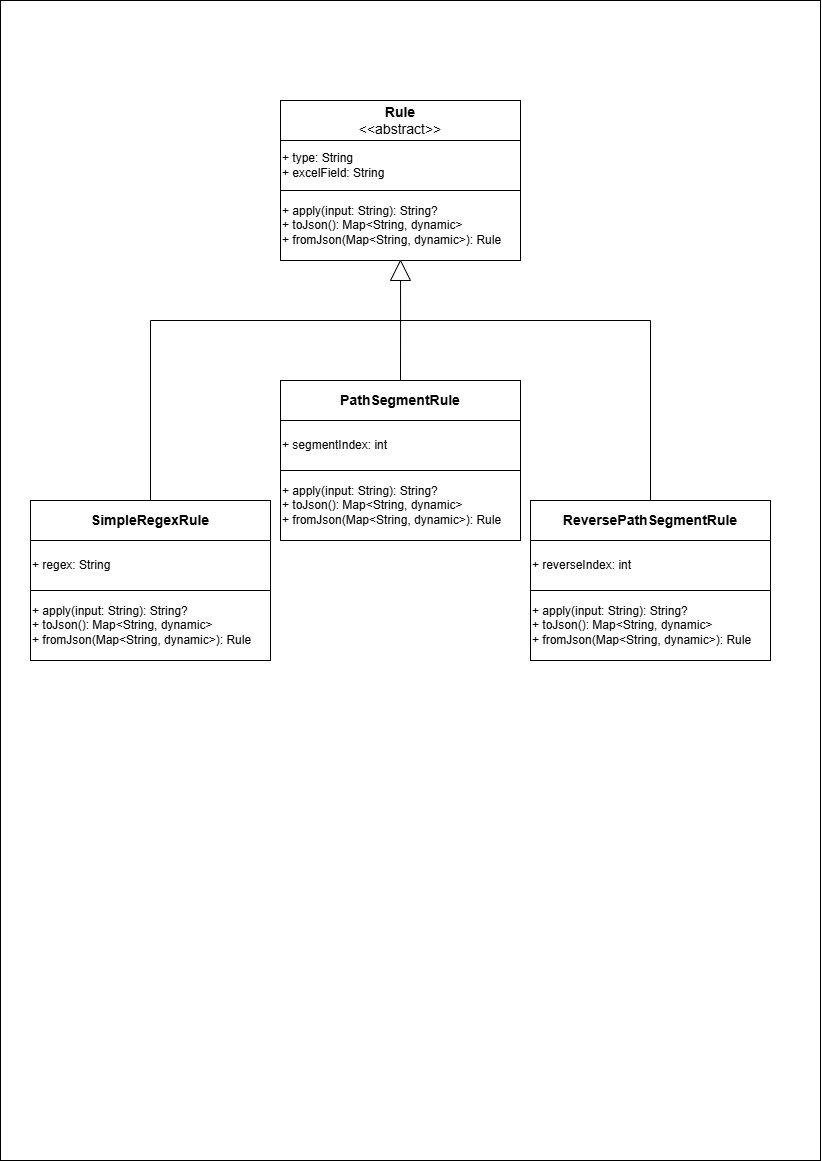


Abbildung : Klassenmodel Regel

## A.6 Kanban-Board

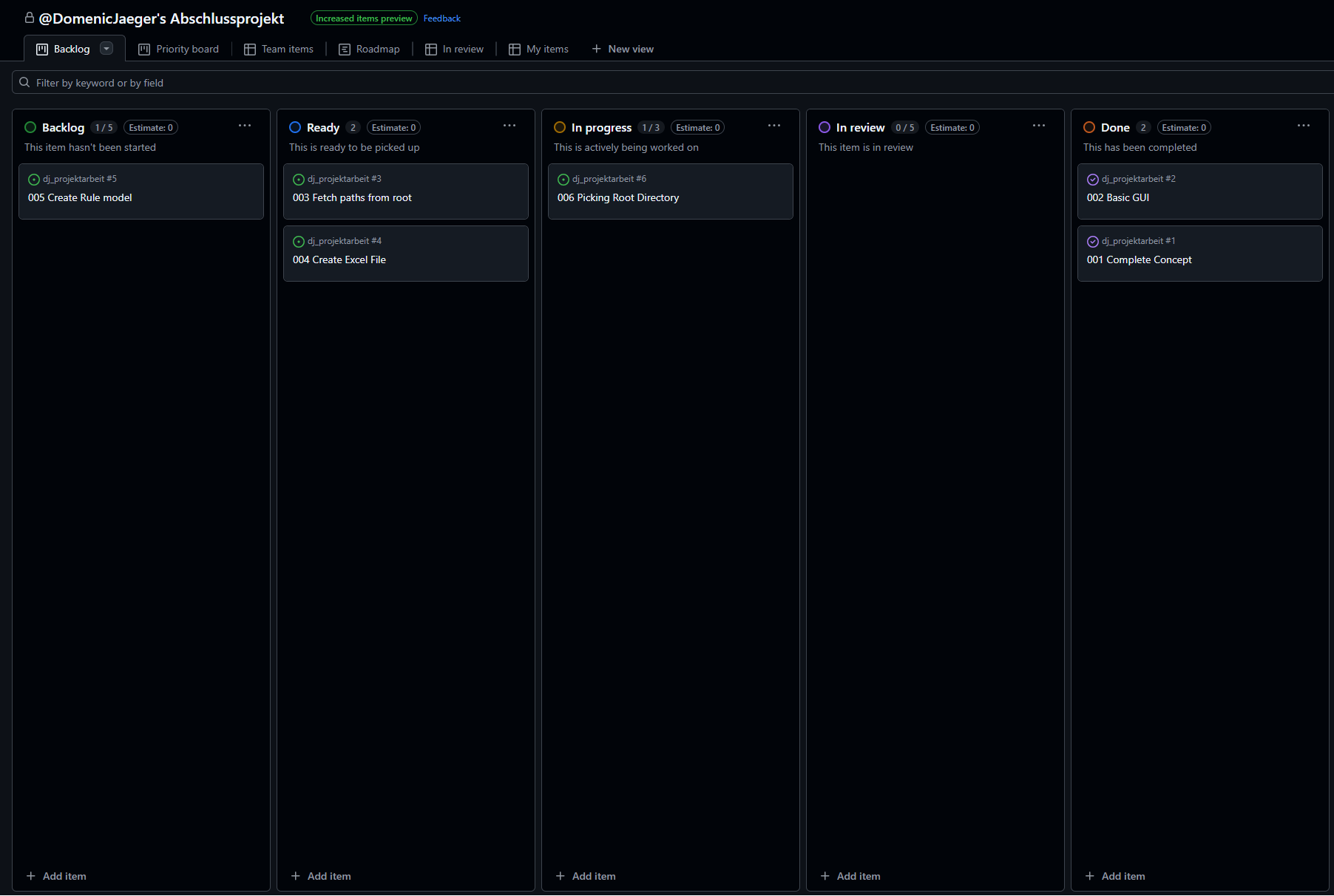


Abbildung : Kanban-Board

## A.7 Darstellung Main Screen

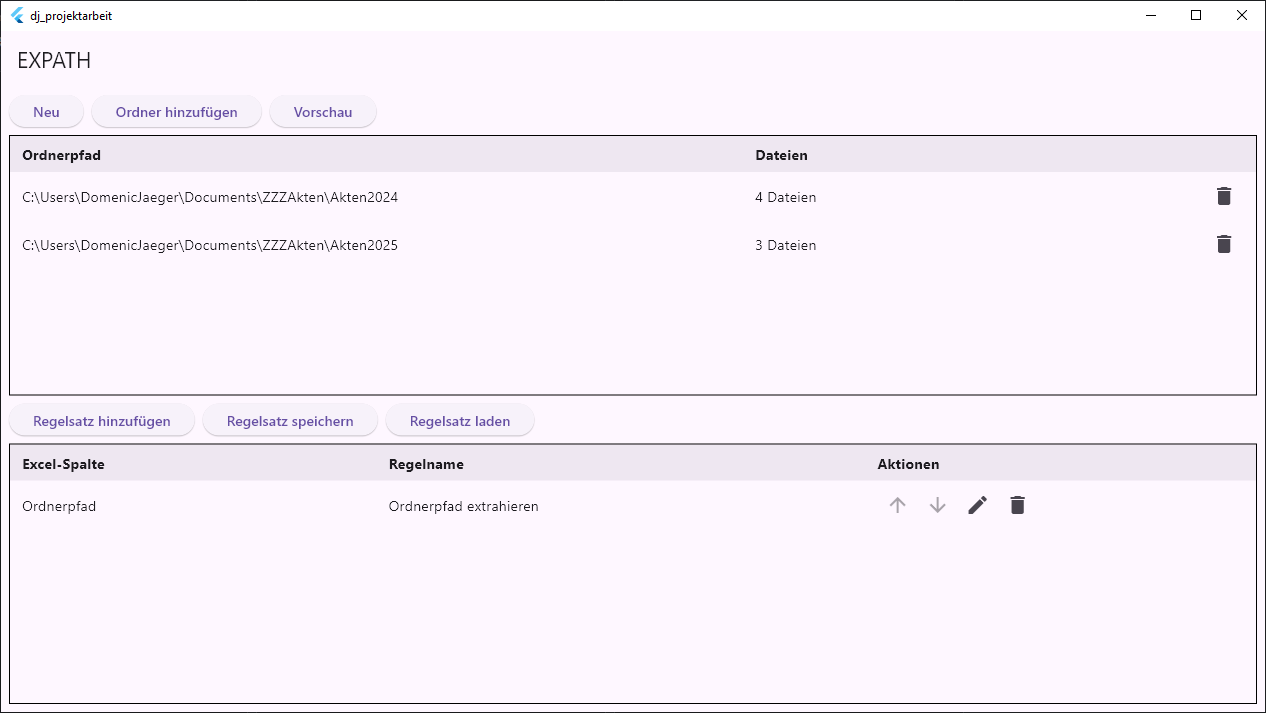


Abbildung : Darstellung Main Screen

## A.8 Darstellung Preview Screen

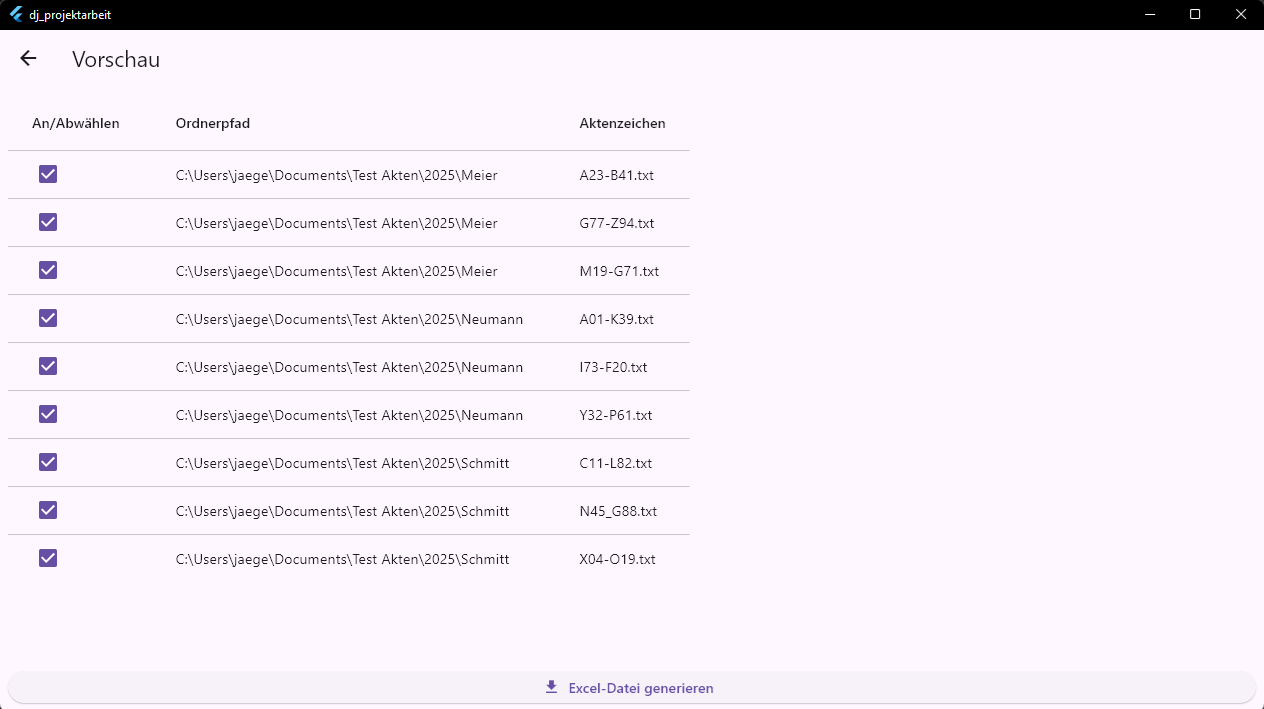


Abbildung : Darstellung Preview Screen

## A.9 Darstellung Rule Editor Screen

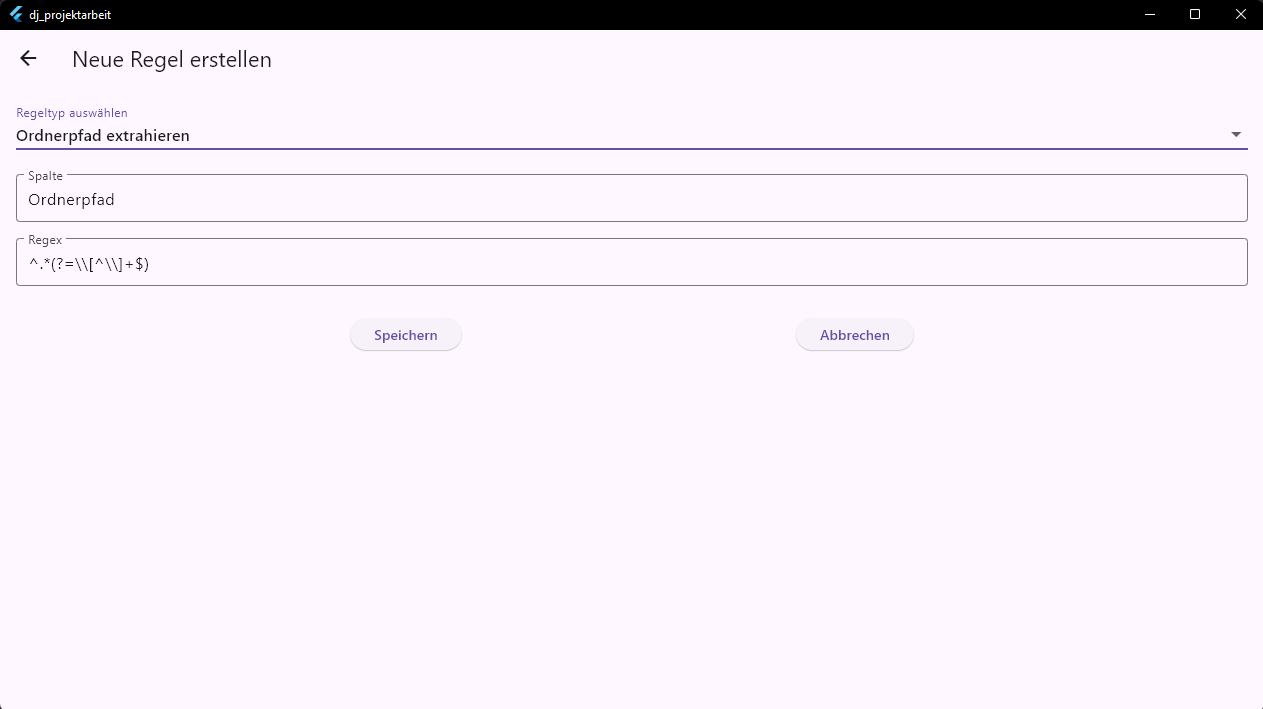


Abbildung : Darstellung Rule Editor Screen

## A.10 Codebeispiel GUI 1

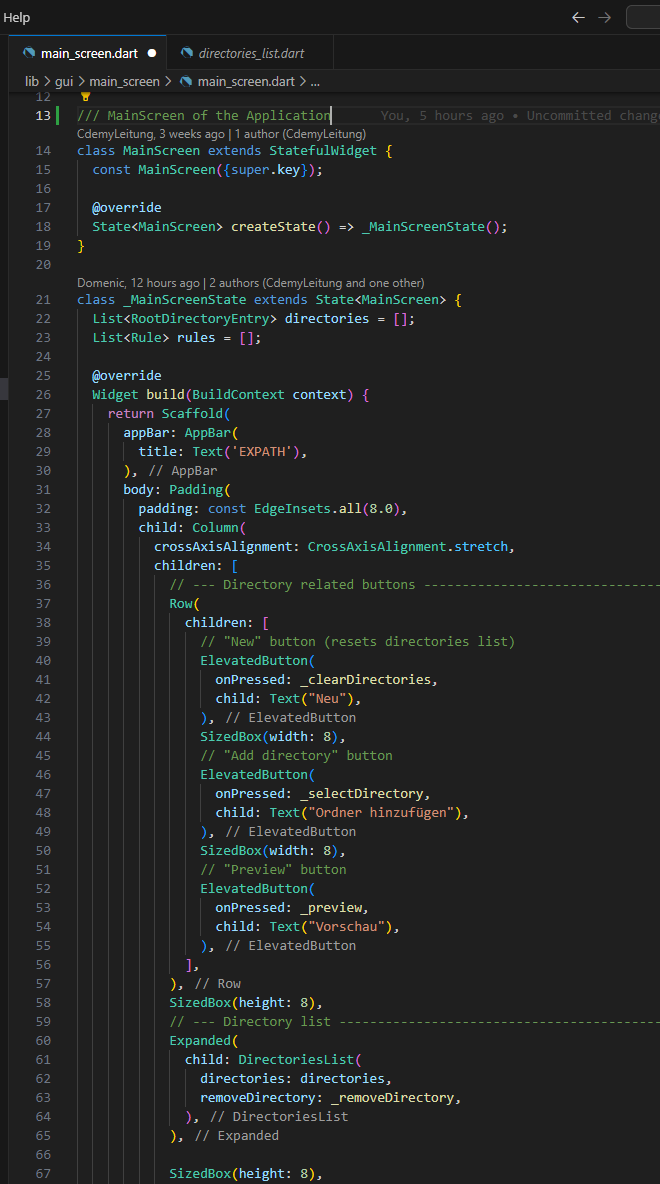


Abbildung : Codebeispiel Main Screen

## A. 11 Codebeispiel GUI 2



Abbildung : Codebeispiel Directories List

## A. 12 Codebeispiel Pathfinder

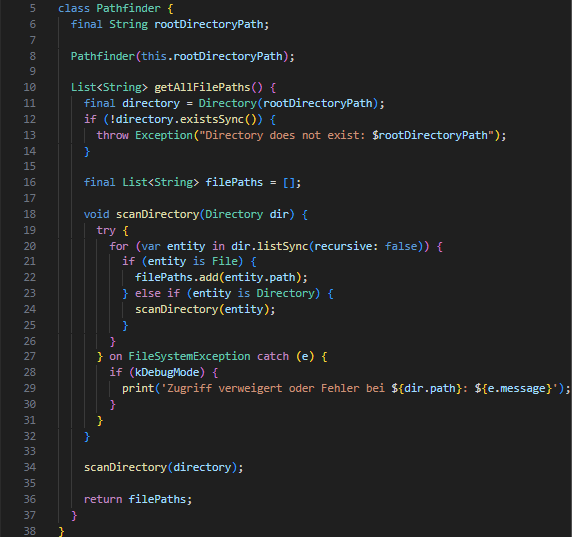


Abbildung : Pathfinder Klasse

## A. 13 Codebeispiel Rule Class

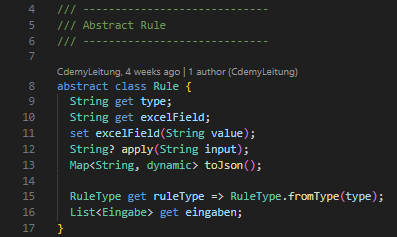


Abbildung : Codebeispiel Abstrakte Rule Klasse

## A.14 Codebeispiel SimpleRegexRule

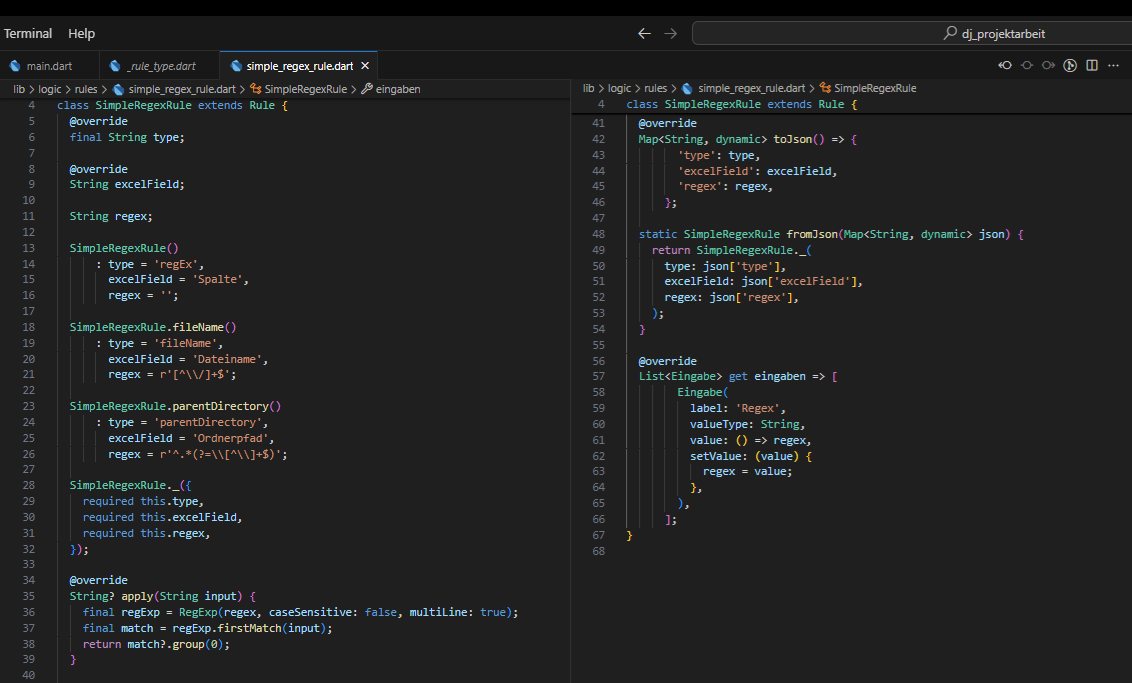


Abbildung : Codebeispiel SimpleRegexRule

## A. 15 Codebeispiel Rule Type Enum



Abbildung : Codebeispiel Rule Type Enum

## A. 16 Codebeispiel Excel Exporter

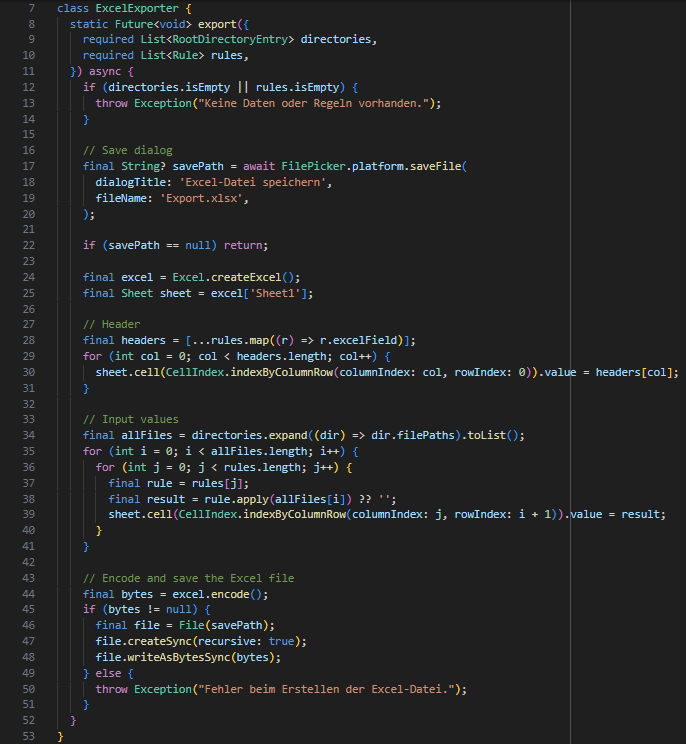


Abbildung : Codebeispiel Excel Exporter

## A.17a Beispiel Unit Test SimpleRegExRule 1

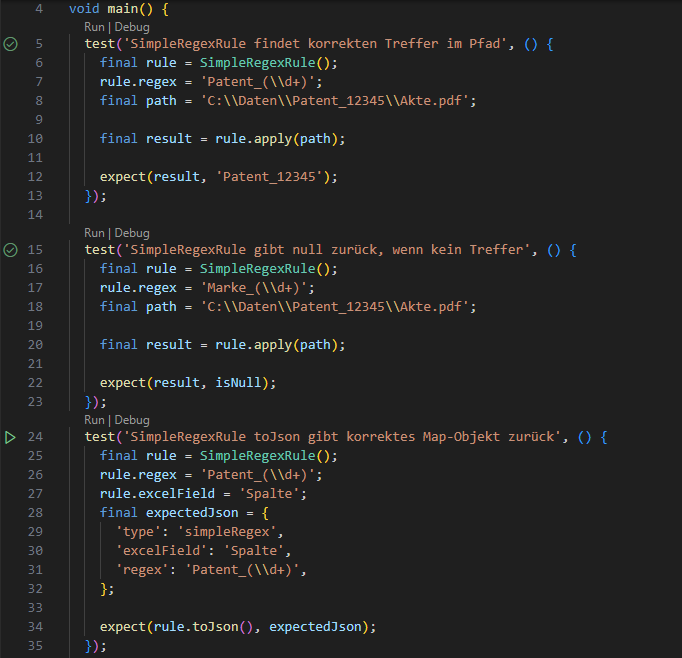


Abbildung : Codebeispiel Unit Test

## A.17b Beispiel Unit Test SimpleRegexRule Fortsetzung

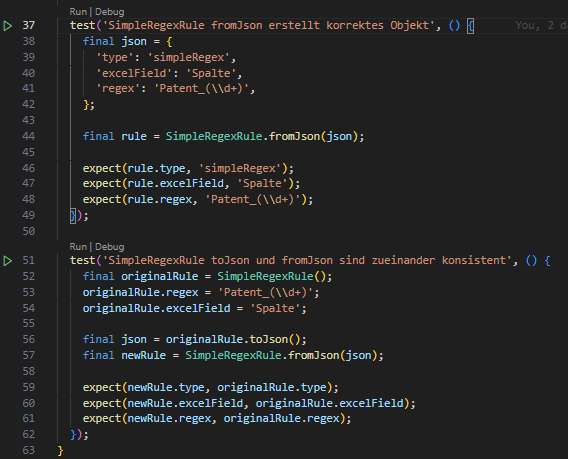


Abbildung : Codebeispiel Unit Test Fortsetzung