5. Implementierung  
  
5.1 Implementierung des Frontend  
  
5.1.1 Struktur der Benutzeroberfläche

Das Frontend der Anwendung wurde vollständig mit dem Framework Flutter umgesetzt. Im Widget-Prinzip von Flutter ist jedes Element als ein Widget definiert, vom kleinsten Textfeld bis zum kompletten Bildschirm. Ein Widget ist eine Baueinheit, welche die Darstellung, oder das Verhalten beschreiben kann.  
In Flutter wird die Benutzeroberfläche durch das Verschachteln und Kombinieren von Widgets aufgebaut. Dynamische Widgets reagieren auf Benutzerinteraktion oder Zustandsänderungen, während Statische Widgets sichtbare Elemente wie z.B. Textfelder oder Buttons darstellen.  
Anhand dieser Technik können auch komplexe Oberflächen leicht gewartet und erweitert werden.  
In dieser Anwendung wurden eigene Widgets entwickelt, um bestimmte Funktionalitäten zu kapseln.  
  
Die Struktur des Frontends gliedert sich wie folgt:  
  
Main Screen:  
Die zentrale Startseite der Anwendung. Hier werden die ausgewählten Verzeichnisse sowie die definierten Regeln verwaltet.  
  
Rule Editor Screen:  
Ermöglicht das Erstellen und Anpassen individueller Reglen, die auf die Ordnerstrukturen angewendet werden.  
  
Preview Screen  
Stellt eine Vorschau der durch die Regeln gefundenen Daten bereit. Aus dieser Ansicht heraus kann der Excel-Export angestoßen werden.  
  
Jeder dieser Screens ist in weitere Widgets untergliedert, die jeweils eine klar definierte Aufgabe übernehmen, um Wiederverwendbarkeit und Lesbarkeit des Codes zu fördern.  
  
5.1.2 Aufbau und Funktion der Screens  
  
Der Main Screen ist in zwei Hauptbereiche aufgeteilt, einem oberen Bereich zur Verzeichnisverwaltung und einem unteren Bereich zur Regelverwaltung.  
  
Die Verzeichnisverwaltung zeigt eine Liste der aktuell ausgewählten Verzeichnisse an. Sie ermöglicht das Hinzufügen neuer Wurzelverzeichnisse über eine Dateiauswahl. Die darunterliegende Regelverwaltung zeigt eine Liste aller definierten Regeln und erlaubt es neue Regeln hinzuzufügen, oder zu bearbeiten, was in beiden Fällen eine Navigation zum Rule Editor Screen zur Folge hat. Darüber hinaus können hier Sammlungen von Regeln auf dem System in einer JSON-Datei gespeichert oder geladen werden.  
  
Im Rule Editor Screen kann über ein Dropdown-Menü Regeln aus einer Auswahl von vorgebauten Regeln, oder eine benutzerdefinierte Regel ausgewählt werden.  
Je nach gewähltem Regeltyp passen sich die folgenden Widgets und Formularfelder automatisch an. In Formularfeldern können weitere Regelparameter angepasst werden. Sollte man über den Editier-Button einer Regel auf diesen Bildschirm navigiert sein, ist bereits eine Regel gewählt und kann angepasst werden. Regeln werden über einen Button gespeichert und an den Main Screen übergeben.  
  
Der Preview Screen stellt die aus der Ordnerstuktur extrahierten Informationen tabellarisch dar. Hier kann der Benutzer prüfen, ob die gesetzten Regeln korrekt angewendet wurden. Jeder Datensatz kann über eine Checkbox an- oder abgewählt (standardmäßig angewählt) um so einzelne Datensätze vom Excel-Export auszuschließen.  
Über einen Button wird dieser Excel-Export dann ausgelöst.  
  
5.1.3 State-Management und Navigation  
  
Zur Zustandsverwaltung innerhalb der App wurde das setState-Prinzip von Flutter verwendet. Für die Navigation zwischen den Screens kommt das Klassische Navigator-Konzept von Flutter zum Einsatz, bei dem über Navigator.push und Navigator.pop zwischen einzelnen Seiten gewechselt wird. Daten wie aktuelle Verzeichnislisten und Regeln werden innerhalb des App-Zustands verwaltet und beim Screen-Wechsel entsprechend übergeben oder neu geladen.  
  
5.1.4 Layout und Benutzerführung  
  
Die grafische Benutzeroberfläche ist schlicht und funktional gehalten. Listenansichten sorgen für Übersichtlichkeit und Standard-Buttons gewährleisten eine intuitive Bedienung. Dialogfenster und Pop-ups werden eingesetzt, um Eingaben zu vereinfachen, und Aktionen klar abzugrenzen. Darüber hinaus nutze ich das SnackBar-Widget um temporär Informationen über den Erfolg eines Vorgangs anzuzeigen, etwa über eine kurze Einblendung, wenn ein wichtiges Textfeld nicht ausgefüllt wurde, oder eine Excel-Generierung erfolgreich war.  
  
5.2 Entwicklung der Logik

Im Zentrum der Anwendung steht die Implementierung der Logik, die das Auswählen von Verzeichnissen, die dynamische Anwendung von Regeln auf Ordnerstrukturen sowie den Export der gewonnenen Informationen ermöglicht. Dabei wurde Wert auf Modularität, Wiederverwendbarkeit und Erweiterbarkeit gelegt.

5.2.1 Pathfinder  
  
Die Auswahl und Verarbeitung der Verzeichnisse erfolgt über die Klasse Pathfinder. Diese kapselt den rekursiven Zugriff auf das lokale Dateisystem und bereitet die Pfade zur späteren Anwendung auf. Die Klasse nutzt Funktionen des Dart-Standardpakets dart:io, um plattformunabhängig auf Dateien und Verzeichnisse zugreifen zu können.  
  
Pathfinder enthält ein Attribut rootDirectoryPath, welches den Startpunkt unserer Suche darstellt und über die Benutzeroberfläche ausgewählt wird. Die Methode getAllFilePaths() gibt eine Liste aller Dateipfade innerhalb des Wurzelverzeichnisses zurück. Das erreicht sie durch den rekursiven Aufruf der Methode scanDirectory() welche jeden Dateipfad findet und der Liste aller Dateipfade hinzufügt.

5.2.2 Regel Editor

Die zentrale Funktionalität der Anwendung besteht in der Definition und Anwendung von Regeln, um relevante Informationen aus den eingelesenen Dateipfadne zu extrahieren. Für die Umsetzung wurde ein eigenes Regelsystem entworfen, das nach dem Strategy Pattern aufgebaut ist.   
Dieses ermöglicht es, verschiedene Regeltypen zur Laufzeit auszuwählen und anzuwenden, ohne dass der aufrufende Code angepasst werden muss. In der App definiert die abstrakte Klasse Rule die gemeinsame Schnittstelle für alle Regeltypen, während konkrete Klassen (SimpleRegexRule, PathSegmentRule, ReversePathSegmentRule) jeweils unterschiedliche Strategien zur Datenextraktion implementieren.  
  
Die Basisklasse Rule definiert ein allgemeines Interface für alle Regeltypen. Wichtige Methoden sind:  
- apply(String input), welche die Regel auf einen Pfad anwendet und ein Ergebnis zurückgibt  
- toJson() und fromJson() welche zum Speichern und Laden von Regeln genutzt werden.

Konkrete Implementierungen von Rule sind:  
- SimpleRegexRule: Durchsucht Pfadnamen mithilfe regulärer Ausdrücke (RegEx). Diese Klasse kann am freiesten verwendet werden, da hier jeder denkbare RegEx eingegeben werden kann. Dies erfodert natürlich Kenntnisse im Umgang mit RegEx.  
- PathSegmentRule: Extrahiert bestimmte Ordnersegmente besierend aud der Position im Pfad, beispielsweise den dritten Ordner im Pfad.  
- ReversePathSegmentRule: Das Gegenstück zur vorherigen Klasse um zum Beispiel den vorletzten Ordner im Pfad zu finden.  
  
Die Auswahl der Regeltypen erfolgt über ein eigens definiertes Enum RuleType. Durch die Nutzung dieses Enums wird die Eweiterbarkeit der Anwendung sichergestellt: Neue Regeltypen können hinzugefügt werden, indem lediglich das Enum ergänzt und eine neue Strategieklasse implementiert wird, ohne bestehende Logik zu verändern.  
  
Benutzer können Regeln dynamisch erstellen, anpassen und speichern. Jede Regel wird zur Laufzeit einer Sammlung hinzugefügt und kann individuell auf die eingelesenen Pfade angewandt werden. Durch die klare Trennung zwischen Regeldefinition und Anwendung bleibt die Codebasis übersichtlich und wartungsfreundlich.

5.2.3 Excel Generierung

Das wesentliche Ziel der Anwendung, die gewonnenen Informationen in einem standardisierten Format zu exportieren wird durch die Excel-Exportfunktion entwickelt.  
Diese ist im Modul excel\_export gekapselt.  
  
Zur Erstellung und Bearbeitung von Excel-Dateien wird die Dart-Bibliothek excel verwendet. Diese ermöglicht es, xlsx-Dateien programmatisch zu erzeugen, Zellen zu befüllen und Formatierungen vorzunehmen, ohne auf externe Software oder Office-Installationen angewiesen zu sein.  
  
Der Ablauf gestaltet sich so: Die gewonnenen Informationen werden in einer strukturierten Liste bereitgestellt. Jeder Datensatz enthält dabei die extrahierten Werte, sortiert nach den definierten Regelspalten. Ein neues Excel-Dokument wird instanziiert und für jede Regel eine Spalte erstellt die anhand des vom Nutzer definierten Feldnamens benannt wird, beispielsweise „Aktennummer“, oder „Erfinder“. Danach werden alle extrahierten Werte zeilenweise in das Excel-Dokument eingetragen. Zuletzt wird die fertige Datei lokal auf dem Computer gespeichert. Über die Methode saveFile() des file\_picker-Packets wird dem Benutzer ein Dateidialog angeboten, um Speicherort und Dateinamen festzulegen. Die Excel-Datei wird anschließend an dem angegebenen Ort im .xlsx-Format gespeichert.

5.3 Tests  
  
Zur Qualitätssicherung wurden begleitend Unit-Tests erstellt. Diese Tests dienen dazu, die Korrektheit einzelner Programmbestandteile unabhängig vom Gesamtsystem zu prüfen. Die Unit-Tests wurden mit dem Test-Framework flutter\_test umgesetzt, das fester Bestandteil der Flutter-Entwicklungsumgebung ist.  
  
Es wurden insbesondere die Modellklassen aus dem Bereich der Regelverarbeitung getestet, da diese den Kern der fachlichen Logik der Anwendung darstellen.  
  
Ein Beispiel für einen Unit Test findet sich im Anhang.