4. Entwurf

4.1 Zielplattform

Da der Anwendungsfall vorsieht, dass der Mitarbeiter an einem Desktop-System arbeitet, fiel die Wahl der Plattform schnell auf Windows. Mit der Wahl von Flutter und Dart als technologischer Basis, wurde jedoch darauf geachtet, die Anwendung künftig mit relativ geringem Aufwand auch für MacOs oder Linux umzusetzbar zu machen. Eine mobile Version ist nicht angedacht, da die Anforderungen des Auftraggebers keine mobile Datenerfassung vorsehen.

4.2 Architekturdesign

Die Anwendung baut auf die Zwei-Schichten-Architektur auf, bei der die Präsentationsschicht und die Geschäftslogik klar voneinander getrennt sind.  
  
Die Präsentationsschicht gibt dem Nutzer die Möglichkeit mit dem System zu interagieren. Hier werden Verzeichnisse gewählt, Regeln verwaltet, eine Vorschau-Anzeige erstellt, sowie der Auftrag zum Export der Daten ausgelöst. Im Projekt ist sie im Verzeichnis lib/gui organisiert und umfasst den Hauptbildschirm und die beiden Nebenbildschirme.

Die Schicht mit der Geschäftslogik stellt die eigentliche Logik der Anwendung dar. Hier werden Ordner durchlaufen, Regeln definiert und mit ihnen gearbeitet, Konfigurationen von Regeln gespeichert und geladen und der Export der Daten in Excel-Dateien ausgeführt. Sie im Verzeichnis lib/logic untergebracht und ist in diverse Module wie ‚filesystem‘, ‚rules‘ und ‚excel‘ unterteilt.  
  
Eine dritte Datenhaltungsschicht wird nicht benötigt, da die Anwendung auf lokale Daten zugreift und diese in Excel-Dateien exportiert und sie nicht im System speichert.

4.3 Entwurf der Benutzeroberfläche

Ziel des Entwurfs der Benutzeroberfläche war es, eine übersichtliche und funktionale Benutzeroberfläche zu gestalten, die den Außendienstmitarbeiter:innen eine einfache Bedienung ermöglicht. Dabei wurde Wert auf die Trennung der Hauptfunktionen gelegt: Verzeichniswahl, Regelverwaltung, Vorschau und Export.  
  
Die grafische Oberfläche wurde in drei Hauptbereiche eingeteilt:  
  
Der Haupbildschirm (Main Scree) stellt die zentrale Arbeitsfläche der Anwendung dar und gliedert sich in zwei vertikale Bereiche. Zum einen die Verzeichnisverwaltung im oberen Bereich mit den Möglichkeiten sich aktuell ausgewählte Wurzelverzeichnisse anzeigen zu lassen, neue über eine Dateiauswahl hinzuzufügen, sowie bestehende Einträge zu löschen.  
Ebenfalls im oberen Bereich findet sich ein Button zum Vorschau-Screen.  
Im unteren Bereich befindet sich die Regelverwaltung. Hier werden alle aktuell definierten Regeln aufgelistet. Es gibt die Möglichkeit neue Regeln anzulegen, was eine Navigation zum Rule Editor Screen zur Folge hat. Darüber hinaus kann man Regeln bearbeiten oder löschen.  
Außerdem besteht die Möglichkeit über einen Button den aktuellen Satz an Regeln zu speichern, oder einen gespeicherten Regelsatz zu laden.  
  
Der Rule Editor Screen dient dem Zweck individuell Regeln zu erstellen oder zu bearbeiten. Der Aufbau gliedert sich in ein Dropdown-Menü zur Auswahl des Regeltyps, sich daraus dynamisch ergebende Eingabefelder, etwa um ein konkretes Regex-Muster einzugeben, oder eine dazugehörige Excel-Spalte zu bennen, sowie Buttons zum Speichern und Abbrechen. Optional sind der Regeltyp bereits gewählt, sowie die Eingabefelder befüllt, solte eine bestehende Regel editiert werden.

Der Preview Screen dient der Überprüfung der gefundenen Ergebnisse, bevor diese exportiert werden. Eine tabellarische Vorschau zeigt die extrahierten Informationen aus den Dokumentenpfaden mit der Spaltenbenennung die in der Excel-Datei zur Anwendung kommen würde. Jede Zelle der Tabelle ist mit einer Checkbox versehen um einzelne Datensätze noch vom Export auschließen zu können, die nicht benötigt werden. Standardmäßig ist jeder Datensatz ausgewählt. Zuletzt noch ein Export-Button welcher den Exportprozess in eine Excel-Datei startet.  
  
Die Benutzeroberfläche folgt einem einfachen Bedienkonzept. Jede Funktion ist direkt über Buttons oder übersichtliche Listen erreichbar. Benutzerinteraktionen wie das löschen oder Hinzufügen von Verzeichnissen erfolgen unmittelbar und intuitiv. Wichtige Aktionen werden über SnackBar-Nachrichten rückgemeldet (z.B. „Regelsatz erfolgreich geladen“). Genauso werden auch Fehlermeldungen oder Warnungen zu Fehleingaben behandelt.  
  
Bedienelemente sind beschriftet, und jedes Eingabefeld ist mit einem Label versehen, das Hinweise auf die korrekte Eingabe gibt.

4.4 Datenmodelle

Ziel des Datenmodell-Entwurfs war es, eine modulare und erweiterbare Struktur zu schaffen.  
  
Überblick über die wichtigsten Datenmodelle

|  |  |
| --- | --- |
| Modellklasse | Beschreibung |
| RootDirectoryEntry | Repräsentiert ein Wurzelverzeichnis, das vom Benutzer ausgewählt wurde. Dient als Ausgangspunkt für die Analyse der enthaltenen Dateien |
| Rule (abstrakt) | Definiert das Interface für alle Regeltypen. Legt zentrale Methoden wie apply(), toJson() und fromJson() und die Struktur der Regel fest. |
| SimpleRegexRule | Spezialisiert sich auf die Anwendung eines regulären Ausdrucks auf Dateipfade, um gezielt Informationen zu extrahieren. Sehr flexible Implementierung der Rule Klasse. |
| PathSegmentRule | Implementierung der Rule Klasse, die ein bestimmtes Segment eines Dateipfads basierend auf seiner Position extrahiert. |
| ReversePathSegmentRule | Implementierung der Rule Klasse, die ein Segment eines Pfades von hinten betrachtet (z.B. zweiter Ordner von hinten). |

RootDirectory Entry  
Diese Klasse stellt ein Verzeichnis dar, das als Ausgangspunkt für die Dateianalyse dient.  
Ein wichtiges Attribut ist String path – der vollständige Pfad zum Wurzelverzeichnis. Diese Klasse kapselt einfache Pfadinformationen und erleichtert die Verwaltung der gewählten Verzeichnisse innerhalb der Anwendung.  
  
Regelmodellierung  
Das Regelmodell basiert auf dem Strategy Pattern, das eine flexible Austauschbarkeit verschiedener Regeltypen ermöglicht.

Abstrakte Klasse Rule:  
- Definiert die gemeinsame Schnittstelle(apply(), toJson(), fromJson() für alle Regeltypen.

Legt gemeinsame Attribute wie type (Typ der Regel) und excelField (Name der Spalte im Excel-Export) fest.  
  
Spezialisierte Regelklassen:  
- SimpleRegexRule: nimmt einen regulären Ausruck entgegen und wendet ihn auf den vollständigen Dateipfad an.

- PathSegmentRule: Extrahiert ein bestimmtes Pfadsegment basierend auf einem Positionsindex.

- ReversePathSegment: Extrahiert ein Pfadsegment rückwärts vom Ende des Pfads.  
  
Alle Regelklassen implementieren Methoden zur Serialisierung und Deserialisierung:  
- toJson() konvertiert ein Regelobjekt in ein Map<String, dynamic>, das gespeichert werden kann zur Wiederverwendung von Regelsätzen.

- fromJson() Erzeugt ein Regelobjekt basierend auf gespeicherten JSON-Daten, was die Ladefunktionalität der Anwendung ermöglicht.  
  
Durch diese Mechanismen kann der aktuelle Arbeitsstand der Benutzer:innen problemlos gespeichert und zu einem späteren Zeitpunkt wiederhegestellt werden.  
  
Die Modellarchitektut ist bewusst so gestaltet, dass sie einfach um neue Regeltypen ergänzt werden kann. Neue Regeltypen müssen lediglich eine neue Klasse auf Basis von Rule implementieren. Eine Ergänzung des RuleType-Enums ermöglicht die Integration neuer Regeln in die Benutzeroberfläch.  
Dadurch bleibt die Anwendung offen für zukünftige Erweiterungen ohne tiefgreifende Änderungen an bestheend Codebestandteilen (Open-Closed-Prinzip).

4.5 Qualitätssicherung  
  
Jede Aufgabe, die auf dem Kanban-Board auf Github festgehalten wurde, wurde in einem eigenen Branch bearbeitet und erst mit dem main-Branch zusammengeführt, wenn die Änderungen mit dem Ausbilder besprochen und dieser mit dem Ergebnis zufrieden war.  
Bei Auffälligkeiten oder Fehlern wurden Änderungen mit dem Ausbilder diskutiert.