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 FachinformatikerInnen. 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 Cloud-basierte SaaS 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 etwa 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 Projekbeschreibung

Bei der Aufnahme von Neukunden, die IP Now nutzen möchten, müssen wichtige Dokumente wie z.B. Fallakten, welche sich auf den lokalen Systemen des Kunden befinden erfasst werden. Dazu benötigt IP Now einige wichtige Standartinformationen zu diesen Dokumenten, welche jedoch von Kunde zu Kunde unterschiedlich abgelegt und benannt sein können. Ein Außendienstmitarbeiter erfasst diese Daten in einem aufwendigen Prozess beim Kunden direkt. Die von mir im Rahmen dieses Abschlussprojekts erstellte Anwendung soll Außendienstmitbarbeiter der GSI dabei untestützen, diesen Prozess zu beschleunigen.

1.3 Projektziel  
  
Die von mir erstellte Anwendung soll es 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 so beispielsweise die Bezeichnung oder das Erstellungsdatum einer Fallakte zu erfassen. Die Daten können von IP Now aus einer Excel-Datei heraus verarbeitet werden, daher ist es die Aufgabe dieser Anwendung, am Ende des Prozesses eine Excel-Datei zu erstellen, welche die extrahierten Daten bereitstellt.  
  
1.4 Projektschnittstellen  
  
Als Versionsverwaltung habe ich mich für das Versionsverwaltungstool Git entschieden. Als erweiterte Schnittstelle nutze ich den Onlinedienst GitHub. Die Auswahl des Wurzelverzeichnis, sowie die Erstellung einer Excel-Datei erfolgen über bereits vorhandene Schnittstellen, die über externe Dart-Packages bereitgestellt werden. Die Benutzer der Anwendung sind ausschließlich AussendienstmitarbeiterInnen der GSI. Der Ausbilder bei cdemy kontrolliert die Code Qualität und dient als Ansprechpartner und Vermittler zum Kunden.

1.5 Projektabgrenzung  
Meine Aufgabe im Rahmen dieses Projekts umfasst die Planung und Umsetzung einer Windows-Anwendung. Dies beinhaltet die Konzeption, und Umsetzung von sowohl der GUI, als auch der Logik und Datenstruktur der Anwendung. Ebenso gehört das Testen der Anwendung zu meinen Aufgaben.  
  
2. Projektplanung

2.1 Projektphasen

Die Zeitplanung sieht einen Umfang von 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 statt.

Die Einteilung der Projektphasen mit der geplanten Dauer setzten sich wie folgt zusammen:

|  |  |  |
| --- | --- | --- |
| Projektphase | Stunden pro Phase | Stunden pro Unterpunkt |
| Analyse | 8 | - |
| - Erstellen eines Pflichtenhefts | - | 5 |
| - Wirtschaftlichkeitsanalyse | - | 3 |
| Entwurf | 12 | - |
| - Planung der GUI | - | 5 |
| - Planung der Datenmodelle | - | 7 |
| Entwicklung | 40 | - |
| - Entwicklung der GUI | - | 18 |
| - Implementierung der Funktionalität | - | 22 |
| Tests und Abnahme | 4 | - |
| Dokumentation | 16 | - |
| GESAMT | 80 |  |

2.2 Enwicklungsprozess  
  
Um die Aufgabenverwaltung einfacher zu gestalten und mich an den Prinzipien des agilen Softwareentwicklungsprozesses zu orientieren habe ich ein Kanban-Board auf Github genutzt. Die Aufteilung in viele kleinere Aufgaben, zusammen mit der Einteilung in die Gruppen „To-Do“, „In-Progress“ und „Done“ waren ein guter Weg um das Projekt übersichtlich zu halten und auf eventuelle Änderungen reagieren zu können.

2.3 Ressourcenplanung

Die Entwicklung der Anwendung fand in den Räumlichkeiten der cdemy GmbH statt. Gearbeitet wurde an einem PC mit Windows 11 Betriebssystem. Zum Schreiben des Programmcodes, zum Debugging und Compilen wurde die IDE Visual-Studio-Code verwendet. Das Projekt wird auf Github gehostet und mit Git lokal verwaltet.

2.3.1 Programmiersprachen

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 objektorientierte Syntax mit Features wie Null-Sicherheit, asynchroner Programmierung, Extension-Methods und Mixins, die eine strukturierte und wartbare Codebasis unterstützen. Flutter, das Framework für die Benutzeroberfläche der Anwendung, basiert vollständig auf Dart. Dadurch wird eine gute Verzahnung zwischen Anwendungslogik und UI ermöglicht.  
  
Die Wahl von Dart für dieses Projekt wurde aus mehreren Gründen getroffen:  
- Die enge Verzahnung mit Flutter erlaubt die plattformübergreifende Entwicklung effizienter Desktop-Anwendungen, was eine spätere Portierung der Anwendung auf andere System wie macOS grundsätzlich ermöglicht.

- Die morderne Sprachstruktur von Dart unterstützt eine saubere, wartbare Projektarchitektur.  
- Ich selbst habe bereits Erfahrungen im Umgang mit Dart sammeln können und musste mich nicht in die Möglichkeiten und Eigenheiten der Sprache einarbeiten.  
  
3. Analyse

3.1 Ist-Analyse

Ein Außendienstmitarbeiter der GSI muss in einem aufwendigen Prozess die Daten beim Kunden erfassen. Dafür müssen die Benennungskonventionen des Kunden bekannt sein, und Ordnerstrukturen häufig manuell nachvollzogen werden. Außerdem bedarf es eines gewissen Grundwissens im Umgang mit Excel-Dateien, um etwa Substrings aus Dateipfaden zu extrahieren. Hier soll die Anwendung ansetzen, indem die einmal mit dem Kunden abgeklärten Bennenungskonventionen in einem Regelsatz festgehalten werden und von einem oder mehreren Wurzelverzeichnissen heraus alle relevanten Informationen automatisiert extrahiert werden können.

3.2 Wirtschaftlichkeitsanalyse

Der bisherige Ablauf ist zur Datenerfassung bei Neukunden ist zeitaufwendig und fehleranfällig. Außendienstmitarbeiter müssen für jeden Kunden erneut die Benennungskonventionen und die Ordnerstrukturen nachvollziehen und diese in einem manuellen Vorgang abarbeiten. Zusätzlich können nur Mitarbeiter eingesetzt werden, die im Umgang mit Excel geschult wurden. Die Anwendung soll daher Zeit sparen und Fehler reduzieren, die weitere Nacharbeiten zur Folge hätten.  
Darüber hinaus ermöglicht die Anwendung eine bessere Zeitauslastung der Außendienstmitarbeiter und senkt die indirekten Kosten durch Excel-Schulungen. Durch Einsparungen bei der Bearbeitungszeit sowie bei der Behebung von Fehlern wird der Entwicklungsaufwand amortisiert.

3.2.1 Projektkosten

Die Kosten des Projekts schlüsseln sich wie folgt auf:  
Mitarbeiter:  
1 Auszubildender FIAE im dritten Lehrjahr  
 - durchschnittliches monatliches Gehalt laut Bundesinstitut für Berufsbildung: 1255,00€

- bei durchschnittlich 21 Monatsarbeitstagen und fünf Wochenarbeitstagen: 7,47€ pro Stunde  
  
1 Ausbilder FIAE

- durchschnittliches monatliches Gehalt in Hessen laut Bundesagentur für Arbeit: 5667,00€  
 - bei durchschnittlich 21 Monatsarbeitstagen und fünf Wochenarbeitstagen: 33,73 pro Stunde  
  
Auf das Gehalt wird der Arbeitgeberanteil zur Sozialversicherung von insgesamt 19,7% aufgeschlagen. Weitere Kosten etwa durch Arbeitsmaterial, Lizenzgebühren, Miete, Strom, Verwaltungskosten, Sonderzahlungen oder Abschreibungen werden gesammelt mit einem weiteren Aufschlag von 100% behandelt. Aus Gründen der Übersichtlichkeit werden diese Beträge auf 20€ pro Stunde für den Auszubildenden und 80€ pro Stunde für den Ausbilder gerundet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorgang | Mitarbeiter | Zeit | Stundensatz | Kosten |
| Planung und Entwicklung | Auszubildender | 80 | 20,00€ | 1600,00€ |
| Fachgespräch mit Kunden | Ausbilder | 3 | 80,00€ | 240,00€ |
| Hilfestellungen im Projektverlauf | Ausbilder | 4 | 80,00€ | 320,00€ |
| Abnahmetest | Ausbilder | 1 | 80,00€ | 80,00€ |
| GESAMT |  |  |  | 2240,00€ |

3.2.2 Amortisierung  
  
Als Basis zur Berechnung der Amortisierung wird angenommen, dass sich durch den Einsatz der Anwendung der zeitliche Aufwand beim Kunden für den Außendienstmitarbeiter um ca. 20% verringert. Der Außendienstmitarbeiter selbst ist Fachinformatiker für Anwendungsentwicklung und das durchschnittliche Gehalt für FIAE in Bayern, dem Sitz von GSI, gemäß Statistik der Agentur für Arbeit, liegt bei 4353€. Bei 21 Monatsarbeitstagen und fünf Wochenarbeitstagen ergibt sich mit dem Arbeitgeberanteil an der Sozialversicherung ein Stundensatz von 30,83€. Dazu ein Aufschlag von 100% für weitere Kosten ergibt gerundete Kosten von 60€ pro Stunde.  
  
Es wird davon ausgegangen, dass der Mitarbeiter bisher 8 Stunden pro Kunde vor Ort verbracht hat.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stundensatz | Zeit | Kosten |
| Vor Einsatz der Anwendung | 60,00€ | 8 | 480€ |
| Mit Einsatz der Anwendung | 60,00e | 6,4 | 384,00€ |
| Ersparnis pro Kunde |  |  | 96,00€ |

Berechnung der Amortisierung:  
2240,00 Projektkosten / 96,00€ Ersparnis pro Kunde = 23,33  
  
Der Schätzung nach wäre das Projekt also nach 24 Einsätzen amortisiert.

3.3 Anwendungsfälle  
  
Die wichtigsten Anwendungsfälle der Software lassen sich in mehrere Kernprozesse unterteilen:

|  |  |
| --- | --- |
| Anwendungsfall | Beschreibung |
| Verzeichnis auswählen | Der Benutzer wählt ein oder mehrere Wurzelverzeichnisse für die Analyse aus. |
| Regeln erstellen und verwalten | Der Benutzer definiert Regeln zur Mustererkennung und kann diese speichern, laden, bearbeiten oder löschen. |
| Analyse der Verzeichnisstruktur | Die Anwendung durchsucht die ausgewählten Verzeichnisse anhand der definierten Regeln und extrahiert relevante Informationen. |
| Vorschau der Ergebnisse | Der Benutzer kann eine tabellarische Vorschau der extrahierten Daten einsehen und prüfen. |
| Export der Daten | Die validierten Ergebnisse können in eine Excel-Datei exportiert und gespeichert werden. |

Ein Use-Case-Diagramm, das die wichtigsten Benutzerinteraktionen visualisiert, wird im Anhang dieser Dokumentation bereitgestellt.

3.5 Qualitätsanforderungen  
Folgende Qualitätsanforderungen wurden definiert:  
- Die Anwendung muss einfach und intuitiv bedienbar sein, ohne lange Einarbeitung.

- Die Anwendung muss stabil arbeiten und auch bei größeren Datenmengen zuverlässig Ergebnisse liefern.  
- Unterschiedliche Kundenstrukturen müssen durch anpassbare Regeln abgedeckt werden können.

- Die Architektur muss eine spätere Erweiterung um neue Regeln ermöglichen.  
  
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.

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.  
  
6. Abnahme  
Eine gesonderte Abnahme durch den Auftraggeber konnte in meiner Projekt-Durchführungszeit noch nicht durchgeführt werden, da sie zum jetzigen Zeitpunkt dem Auftraggeber noch nicht vorgeführt wurde.  
  
Der Fortschritt des Projekts wurde jedoch ausführlich mit meinem Ausbilder besprochen, der im engen Kontakt mit dem Auftraggeber steht.  
  
7. Fazit

7.1 Soll-Ist-Vergleich

Ziel es Projekts war es, eine Anwendung zu entwickeln, die Außendienstmitarbeiter:innen der GSI bei der Erfassung von Daten aus lokalen Ordnerstrukturen unterstützt un ddiese Datne in ein einheitliches Format für die Weiterverarbeitung in „IP Now“ überführt.  
  
Im Soll-Zustand waren folgende Hauptanforderungen definiert:  
  
- Auswahl von Verzeichnissen zur Analyse  
- Definition anpassbarer Regeln zur Informationsgewinnung  
- Vorschau der extrahierten Daten  
- Export der Daten in eine Excel-Datei im .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:  
  
- Projektorganisation: Die Nutzung von GitHub als Versionsverwaltung und Aufgabenmanagement-Tool (Kanban-Board) hat sich als sehr hilfreich erwiesen, um auch in Einzelarbeit den Überblick zu bewahren.  
  
- Flutter und Dart: Der Einsatz von Flutter für eine Desktop-Anwendung unter Windows brachte spezifische Herausforderungen, beispielsweise der Dateisystemzugriff und der Umstand dass nicht für alle Ordner zwingend Zugriffsrechte bestehen. Diese wurden jedoch erfolgreich gelöst.  
  
- Designprinzipien: Die bewusste Entscheidung für eine modulare Architektur (z.B. Zwei-Schichten-Architektur, Strategy Pattern) erleichterte die Implementierung und wirkte sich positiv auf die Wartung und Erweiterbarkeit der Anwendung aus.  
  
- Fehlertoleranz und Testing: Die Implementierung von Unit-Test hat die Bedeutung von frühzeitiger Fehlererkennung und Testabdeckung verdeutlicht.

7.3 Ausblick  
  
Die Anwendung erfüllt zwar alle definierten Projektziele, bietet jedoch viel Raum für künftige Erweiterungen.  
  
So ist es denkbar dass mehr vorgebaute Regeltypen implementiert werden könnten, was durch die Struktur mit geringem Aufwand durchführbar wäre.  
Es wäre auch denkbar die Exportformate zu erweitern, etwa als CSV-Datei oder direkt in eine Datenbank.  
Die Benutzeroberfläche bietet Raum für weitere Komfortmöglichkeiten, wie etwa Filtermöglichkeiten in der Vorschau.  
Außerdem wäre eine Erweiterung der Plattformunterstützung denkbar, etwa für den Einsatz auf Linux- oder MacOS-Systemen.  
  
Diese Optionen zeigen, dass die entwickelte Anwendung eine solide Basis darstellt, auf der zukünftige Erweiterungen und Anpassungen problemlos aufbauen können.