Codemodell

Datenobjekte

Die identifizierten Datenobjekte sind:

- Webscraping
- Datenextraktion
- Datenbankverwaltung
- Excell Tabelle
- E-Mail
- Rechnung Weitergeleitet Status

Sequenzen & Programmlogik

Main

Der Einstiegspunkt in das Programm. Von hier aus werden die anderen Workflows aufgerufen, bis am Ende die E-Mail versendet wird.
Der Reihe nach:

1. Mit 01_get_pdf werden alle PDF's heruntergeladen
2. Für jedes heruntergeladene PDF werden folgende Schritte durchgeführt
    a. Mit 02_read_pdf werden alle von der Rechnung relevanten Informationen extrahiert und in einem JSON abgelegt.
    b. Dieses JSON wird dann benutzt, um mit 03_write_db die Daten auf der MongoDB abzulegend.
    c. Verschiebe das bearbeitete PDF Dokument in einen anderen Lokalen Ordner
3. Mit 04_read_db_to_excel werden von der Datenbank alle Rechnungen heruntergeladen, die noch nicht mit einer E-Mail versendet wurden. Diese Informationen werden dann in einer lokalen Excel Datei abgelegt.
4. Mit 05_email wird die soeben erstellte E-Mail versendet

01_get_pdf

Extrahieren der PDFs von der Website

1. Überprüfen, ob die nötigen Ordner zum ablegen der PDFs existieren, falls nicht erstellen.
2. Browser öffnen
3. Auf die Website navigieren
4. Übersetzungs Dialog schliessen, falls er auftaucht
5. Auf den "Rechnungen" Tab wechseln
6. Die Links für die PDFs aus der Liste extrahiere
7. Für jeden Link
    a. Überprüfen, ob es eine .pdf Datei ist

    b. Überprüfen, ob der Dateiname gültig ist. Darf nicht “Wrong” enthalten
    c. Datei in den Lokalen Ordner herunterladen
8. Auf Weitere Rechnungen klicken
9. Auf der zweiten Seite Schritte 6,7 wiederholen
10. Die Liste aller Pfade der heruntergeladenen Rechnungen zurückgeben

02_read_pdf

Test aus dem PDF auslesen und in eine JSON Struktur bringen

1. Überprüfen, ob Datei existiert
2. Den Text aus dem PDF extrahieren
3. Mittels einem LLM (zb. ChatGPT/ OLLAMA) die Informationen aus dem Text extrahieren, die für uns relevant sind und diese in einem JSON anordnen
4. Das generierte JSON zurückgeben

03_write_db

JSON auf der Datenbank ablegen.

1. Verbindung zur MongoDB herstellen
2. Überprüfen, ob die Datenbank und Collection existieren
3. Anhand einer Kombination von invoiceNr, Company, amount der Rechnung überprüfen, ob die Rechnung schon auf der Datenbank ist. Wenn es eine Rechnung mit der Gleichen Kombination dieser drei Features gibt, wird angenommen, das es die gleiche Rechnung ist, und die Rechnung wird nicht noch einmal hochgeladen.
4. Falls sie noch nicht auf der Datenbank ist, das JSON in die MongoDB hochladen.

04_read_db_to_excel

Nach nicht versendete Rechnungen herunterladen und in ein Excel Dokument einfügen.

1. Überprüfen, ob die benötigten Ordner vorhanden sind, wenn nicht erstellen.
2. Verbindung mit MongoDB aufbauen
3. Überprüfen, ob die Datenbank und die Collection existieren
4. Alle Rechnungen herunterladen, welche noch nicht versendet worden sind. Wo der invoiceForwarded Status noch auf false ist.
5. Aus den erhaltenen Rechnungen eine DateTable bauen
6. Die DataTable in eine Excel Datei schreiben
7. Bei den Rechnungen, die in das Excel geschrieben wurden, den invoiceForwarded Status auf True setzen
8. Den Pfad für den erstellten Report zurückgeben

05_email

Das Excel Dokument versenden.

1. Überprüfen, ob ein Report vorhanden ist
2. Eine E-Mail erstellen
3. Eine Kurze Nachricht schreiben
4. Den Report der E-Mail anfügen, falls vorhanden
5. Den Report in einen anderen Lokalen Ordner verschieben.

Fehlerbehandlung

Um die Main Sequenz wird ein Try-Catch Block eingebaut, um das Errorhandling dort zu verbessern. Ansonsten soll ausgiebig mit Try-Catch Blöcken im Code gearbeitete werden, um die Fehler besser behandeln zu können oder zumindest eine bessere Fehlermeldung zu werfen.
Zudem werden wichtige Informationen über Fortschritt, Fehler etc. geloggt.

Schnittstellen

Wir benötigen eine Schnittstelle zur MongoDB, und eine, um eine Excel Datei zu erstellen.
Der Datenbankzugriff wird über ein UIPath Plugin gemacht.
Die Excel Anbindung besteht bereits durch eine Standard UIPath Activity.

Sprach- und Codierungsstandards

Dateipfade werden Betriebssystem unabhängig definiert, so dass sie auf jedem System gültig sind. Dazu Funktionen wie relative Pfade, und Utility Funktionen wie zb. Path.Combine() verwenden, die sicherstellen, dass die Ordner Separierungszeichen des Betriebssystem verwendet werden.
Der Code wird Modular aufgebaut, wo das nicht mit einem erheblichen Mehraufwand verbunden ist.
Wenn zusätzlich zu UIPath noch weiterer Code benötigt wird, soll Python gemäß PEP8 verwendet werden

Dokumentation

Die Dokumentation des Programmes erfolgt durch die Dokumente Code-, Daten-, Architekturmodell. Des Weiteren wird der Code so geschrieben, dass er “self-documenting” ist. Gut lesbar und verständlich. Es wird, wo nötig, mit Kommentare ergänzt, die das Verständniss und die Lesbarkeit des Codes fördern.

Namenskonventionen

Für die Benennung von Variablen usw. soll snake_case verwendet werden. Diese sollen aussagekräftige Namen erhalten, um die Lesbarkeit und das Verständnis zu erhöhen.

Struktur und Architektur

Das Projekt wird wie oben beschrieben in einen Main Workflow und mehrere Sub-Workflows aufgeteilt, welche vom Main aus aufgerufen werden.
Das ganze Programm kann vom Orchestrator aus gestartet werden.

Sicherheitsaspekte

Um die Sicherheit zu erhöhen, sollen UIPath und notwendige Abhängigkeiten aktualisiert werden, wenn bekannt wird, dass eine Schwachstelle vorhanden ist.
Auf die Sicherheit wird aber nicht höchste Priorität gesetzt. Da wir nicht mit Kundendaten arbeiten müssen wir nicht ganz so viele sicherheitsvorkehrungen treffen, da wir in kauf nehmen können, dass zb. Die Rechnungen unverschlüsselt übertragen werden.

Leistungsoptimierung

Die Leistungsoptimierung ist für diese Aufgabenstellung sekundär und wird nicht speziell behandelt. Externe Bibliotheken werden verwendet, um den Code so effizient und sicher wie möglich auszuführen. Dies gilt für die Datenbank und das Extrahieren der wichtigen Informationen aus den PDFs.

Fehlerbehandlung und Ausnahmen:

Standards für die Behandlung von Fehlern, das Logging von Ausnahmen und den Umgang mit unerwarteten Situationen.

Testbarkeit

Mögliche Fehler können mit einem Try-Catch-Block abgefangen werden. Ziel ist es, den menschlichen Operator auf den Fehler aufmerksam zu machen, damit dieser eingreifen und den Fehler beheben kann. Fehlermeldungen sollen geloggt oder ausgegeben werden, damit der Operator schnell handeln kann.

Abhängigkeiten und Bibliotheken

- UIPath Community Version 2023.4.4
- MongoDB Compass 1.43.1
- InternalLabs.MongoDB.Activities für den Datenbankzugriff
- OpenAI / OLLAMA für das Extrahieren der Informationen aus den Texten
- UIPath.WebAPI.Activities für das Arbeiten mit JSON
- UIPath.Excel.Activities
- UIPath.Mail.Activities
- UIPath.OpenAI.IntegrationService.Activities
- UIPath.System.Activities
- UIPath.UIAutomation.Activities
- UIPath.DocumentUnderstanding.Activities

Das Programm wurde in einer Mac Umgebung entwickelt, sollte aber auch auf einem anderen Betriebssystem funktionieren.

Versionskontrolle

Für die Versionskontrolle wurde Github verwendet. Commits werden gemacht, sobald eine funktionsfähige Erweiterung des Codes vorgenommen wurde. Branching wird nicht benötigt. Commits sollen aussagekräftig kommentiert werden, damit die Teammitglieder auf dem Laufenden sind. Code-Reviews werden in regelmäßigen Abständen durchgeführt.

Internationalisierung und Lokalisierung

Es wird keine Mehrsprachigkeit umgesetzt. Alle Benutzerinteraktionen finden auf Englisch statt.