

# **Projekt zur Vorbereitung des automatischen Auslesens von Rechnungen**

von Sabrina Müller

Projektarbeit im Rahmen der Prüfung  
zur Applikationsentwicklerin  
Datum: 14.02.2023

# **Inhaltsverzeichnis**

Zielsetzung.....	3
Arbeitsablauf.....	4
Proof of Concept.....	4
Erstellen des Filehandlers.....	5
Erstellen der Website.....	6
Erstellen der Datenbank.....	7
Qualitätsmanagment.....	8
Verwendete Technologien.....	8
Zusammenfassung.....	9

# Zielsetzung

Die Firma Eitea hat eine Software entwickelt, mit welcher andere Firmen ihre täglichen Verwaltungstätigkeiten mittels IT effizienter gestalten. Die Software unterstützt hier von der Buchhaltung bis hin zur Mitarbeiterverwaltung.

Ich habe im Rahmen meiner Ausbildung bei der Firma Eitea mehrere Praktika absolviert und aus der Praktikumstätigkeit heraus ist das präsentierte Projekt entstanden.

Das Projekt soll die Basis für eine Softwareerweiterung darstellen, welche es dem Nutzer ermöglicht, Rechnungen aller Art hochzuladen und diese automatisch auszulesen.

Ziel des Projektes ist hierbei ein Tool zu entwickeln, welches die grundlegende Verarbeitung und Speicherung der Rechnungen darstellen und durchführen soll. Dieser Prozess ist zwischen "Scannen" und automatischer Auslesung der Rechnungsinhalte erforderlich.

Nachfolgend sind die Arbeitsschritte sowie -inhalte dargestellt, welche durch das Tool abgedeckt werden sollen.

Als Input für das Tool dienen Bilddateien von Rechnungen, welche als Datei in einen Ordner hochgeladen werden.

Im nächsten Schritt soll das Tool die Rechnungsdateien automatisch aufgreifen und mit den Dateien die folgenden Arbeitsschritte durchführen:

1. Die Rechnungsdateien werden in einen weiteren Ordner für weitere Bearbeitung verschoben.
2. Metadaten (Größe, Datum des Erstellens, Dateiformat) zu der Rechnung werden ausgelesen
3. Für die Rechnung wird eine Textdatei mit dem Dateinamen der jeweiligen Rechnung und einer Unique uuid4 ID erstellt.
4. In der Textdatei werden der Dateiname aus Schritt 3 sowie die in Schritt 2 ausgelesenen Metadaten abgespeichert.
5. Nach der Speicherung der Informationen in Schritt 4 wird für jede erfasste Rechnung ein Unterordner mit dem Dateinamen der aktuell erfassten Rechnung und der Unique ID angelegt
6. Im letzten Schritt werden die Fotodateien der jeweiligen Rechnung mit der zugehörigen Textdatei in den neu erstellten Unterordner verschoben.

Bei der Umsetzung wurden folgende weitere Inhalte berücksichtigt:

1. Für die Schritte des Verschiebens und des Auslesens werden für jede Rechnung Jobs in einer Queue angelegt
2. Ein Job kann nur ausgeführt werden, wenn der vorhergegangene Job erfolgreich ausgeführt worden ist
3. Auf einer Webseite wird der Status zu allen laufenden, wartenden und fertig ausgeführten Jobs dargestellt. Dies dient auch zur Überprüfung von fehlgeschlagenen Jobs
4. Ein Button auf der Webseite ermöglicht das Starten und Stoppen der Queue
5. Alle erfolgreich abgeschlossenen Jobs werden final in einer Datenbank gespeichert

# Arbeitsablauf

Bei der *Projektmethode* habe ich mich für *inkrementelles Arbeiten* entschieden. Ich habe Schritt für Schritt mein Projekt um die benötigten Funktionen erweitert.

## **Proof of concept (POC) für die Queue:**

Am Beginn des Projekts musste ich Technologien finden, mit welchen sich Queues in der vorgegebenen Programmiersprache Python realisieren lassen.

Nach meiner Recherche im Internet bin ich auf die Libraries *Celery* und *RQ* gestoßen, die sich für mein Vorhaben eignen könnten.

Beide Technologien verwenden als Hilfstool einen Redis Server.

In kleinen Testprogrammen habe ich die Funktionen von *Celery* und *RQ* ausprobiert.

Dabei habe ich herausgefunden, dass *Celery* sehr umfangreich, aber dadurch auch komplizierter ist als *RQ*.

*RQ* beschränkt sich nur darauf die Queue zu steuern und zu verwalten.

Da eine einfache Verwaltung der Queue für das Tool vollkommen ausreichend ist, hab ich mich dafür entschieden die Library *RQ* in meinem Projekt zu verwenden.

Somit habe ich zuerst damit begonnen einen Redis Server aufzusetzen und den Code für das Handling der Queues geschrieben.

Im Code zur Handling der Queues hab ich die folgenden Funktionen definiert:

- Erstellen von Jobs
- Auslesen der aktiven Jobs vom Redis Server
- Speichern aller fertigen Jobs in der Datenbank
- Pausieren der Queue
- Starten der Queue
- Auslesens der fertigen Jobs aus der Datenbank

## Erstellen des File-Handlers:

Für das Handling der Files, die in dem Ordner der neuen Rechnungen landen, habe ich eine Python Datei *"File-Handler"* erstellt.

Der *"File-Handler"* enthält die folgenden Funktionen:

- Überprüfen der Ordner (check-first):  
Überprüft ob sich eine neue Datei im Ordner *"new\_bills"* befindet und ob der Ordner *"edit\_bills"* leer ist.
- Verschieben der Datei vom Ordner *"new\_bills"* zum Ordner *"edit\_bills"*:  
Dabei wird die erste Datei, die sich im Ordner *"new\_bills"* befindet in den Ordner *"edit\_bills"* verschoben.  
Es wird auch eine Textdatei erstellt, die den Namen der Rechnung trägt und eine Unique uuid.
- Auslesen der Metadaten der Datei:  
Die Metadaten der Rechnung im Ordner *"edit\_bills"* werden ausgelesen (Filename, Pfad, Größe, Erstelldatum, Typ der File).  
Diese werden in die zugehörige Textdatei geschrieben.
- Verschieben der Dateien in den Ordner *"done\_bills"*:  
Die Rechnung und die Textdatei werden in einen neu erstellten Unterordner innerhalb des Ordners *"done\_bills"* verschoben.  
Dieser trägt den Namen der Rechnung und einer erstellten Unique ID.



Abbildung 1: Abbildung der Textdatei zu einer erfassten Rechnung (Beispiel)

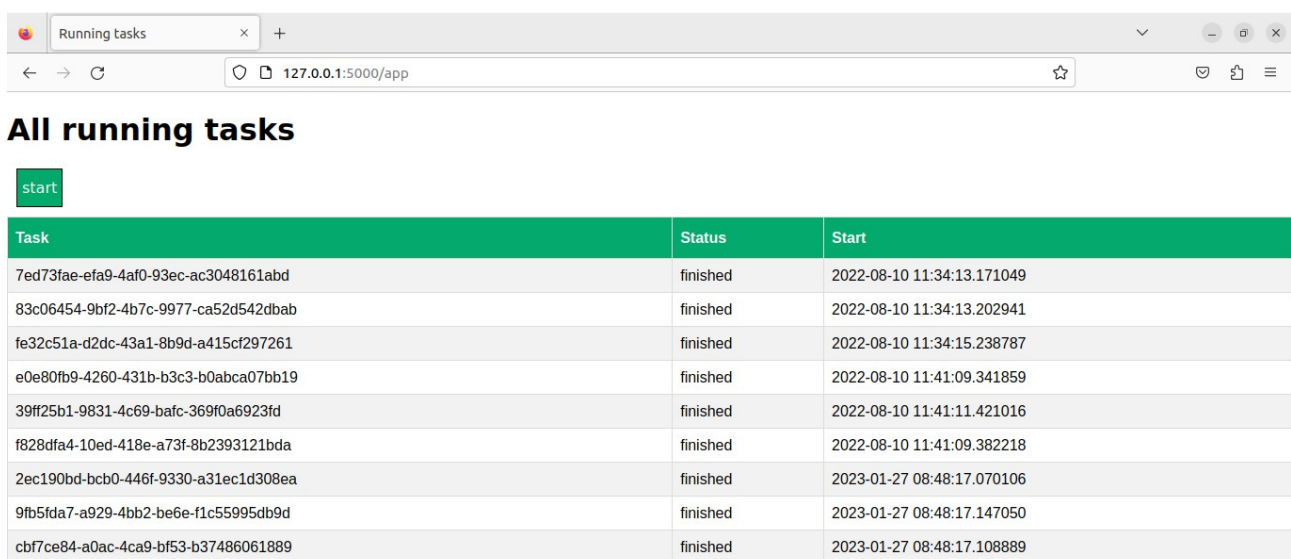
## Erstellen der Webseite:

Im nächsten Schritt habe ich mit Python ein File erstellt, in der ich mittels der Library *Flask* die Routen der Webseite definieren konnte. Die Routen verweisen auf die zu erstellende *HTML-Datei*, wobei dabei auch die Daten der Jobs übergeben werden.

Dann habe ich mich der *HTML-Datei* gewidmet. Ein dynamischer Button am Beginn der Seite ermöglicht die Kontrolle über das Starten und Stoppen der Queue.

Darunter befindet sich eine Tabelle, welche die aktuellen und fertigen Jobs der Queue darstellt.

Mit einer in der *HTML-Datei* verlinkten *CSS-Datei* wird die optische Darstellung optimiert.



Task	Status	Start
7ed73fae-efa9-4af0-93ec-ac3048161abd	finished	2022-08-10 11:34:13.171049
83c06454-9bf2-4b7c-9977-ca52d542dbab	finished	2022-08-10 11:34:13.202941
fe32c51a-d2dc-43a1-8b9d-a415cf297261	finished	2022-08-10 11:34:15.238787
e0e80fb9-4260-431b-b3c3-b0abca07bb19	finished	2022-08-10 11:41:09.341859
39ff25b1-9831-4c69-bafc-369f0a6923fd	finished	2022-08-10 11:41:11.421016
f828dfa4-10ed-418e-a73f-8b2393121bda	finished	2022-08-10 11:41:09.382218
2ec190bd-bcb0-446f-9330-a31ec1d308ea	finished	2023-01-27 08:48:17.070106
9fb5fda7-a929-4bb2-be6e-f1c55995db9d	finished	2023-01-27 08:48:17.147050
cbf7ce84-a0ac-4ca9-bf53-b37486061889	finished	2023-01-27 08:48:17.108889

Abbildung 2: Darstellung der Webseite mit “Start”-Button und Übersichtstabelle

Den Template Code in der HTML-Datei, welcher die Kommunikation mit Python ermöglicht, wurde mit Jinja2 umgesetzt.

## Erstellen der Datenbank:

Für die Datenbank habe ich mich für Sqlite3 entschieden. In dieser Art von Datei lassen sich schnell und einfach die für mich wichtigen Daten speichern.

In der Tabelle *“finishedjobs”* habe ich die Daten der fertigen Jobs gespeichert:

- ID des Jobs vom Redis Server
- Status des Jobs
- Startzeitpunkt des Jobs

Außerdem habe ich eine Tabelle *“paused”* erstellt, in der ich als Boolean gespeichert habe, ob die Queue im Moment pausiert ist oder läuft.

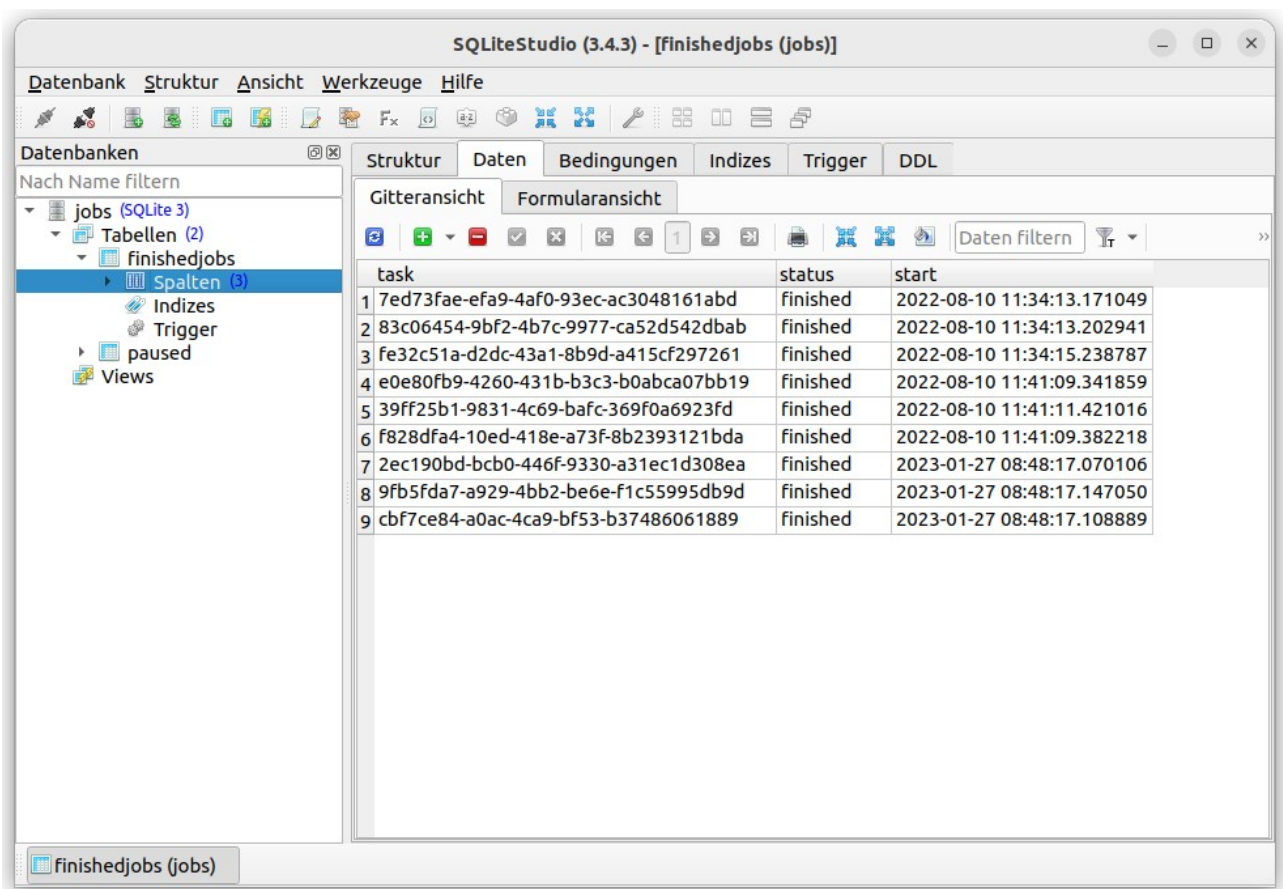


Abbildung 3: Darstellung der Datenbank

# Qualitätsmanagment

Um die Qualität und Funktion meines Projekts zu gewährleisten habe ich während meiner Arbeit Unit-Tests und Integrations-Tests angewendet.

## Unit-Test:

Nach der Erstellung einer jeden Teilkomponente habe ich diese ausführlich auf deren Funktion überprüft. Dies wurde so lange durchgeführt, bis ein reibungsloser Ablauf dargestellt werden konnte.

## Integrations-Test:

Mit jeder weiteren neu erstellten Komponente musste ich auch sicherstellen, dass die Kommunikation zwischen der neuen Komponente und dem restlichen Programm gewährleistet ist. Dabei habe ich die Funktionen der einzelnen Komponenten untereinander getestet.

Nach der Fertigstellung des Projektes wurde darüber hinaus durch meinen Vorgesetzten ein "System- und Abnahme- Test" mit dem fertigen Tool durchgeführt.

Nach der erfolgreichen Durchführung des Tests waren die Arbeiten am Projekt abgeschlossen.

# Verwendete Technologien

Als vorgegebene Programmiersprache habe **Python 3.10.6** verwendet.

Zusätzlich wurden folgende Libraries im Projekt genutzt:

- **Flask:** Zum Realisieren der nötigen Pfade um eine Verbindung zur HTML-Datei herzustellen.
- **Jinja2:** Um einen Template Code in der HTML einzufügen, mit dem die geforderten Daten dargestellt werden.
- **Magic:** Python-magic um das Format der auszulesenden Datei festzustellen.
- **RQ:** Zum Erstellen einer Queue, mit der sich auch die Jobs in der Queue anzeigen und verwalten lassen.

Außerdem habe ich noch einen **Redis-Server** verwendet, der mit seinem in-Memory Datenstruktur-Store als Datenbank für die laufenden Jobs verwendet wird. Der "*rq worker*" der RQ Library, der für die Abarbeitung der Jobs dient, steht in direkter Kommunikation mit dem Redis-Server.

Für die dauerhafte Speicherung der Jobs-Daten habe ich eine **Sqlite3** Datenbank ausgewählt.



# Zusammenfassung

Ab Start der Aufgabenstellung am 25. Juli 2022 bis zum Abschluss meines Projekts habe ich 2 Wochen benötigt.

Nach der Recherche, welche Technologien ich am besten für die Umsetzung der Teilschritte für mein Projekt verwenden kann, wurden die Teilelemente Schritt für Schritt umgesetzt und erprobt.

Zuerst habe ich gelernt, wie der *"rq worker"* in Verbindung mit der dem Redis-Server funktioniert. Nachdem dieser Ablauf funktioniert hat, habe ich mich dem Auslesen der Verzeichnisse gewidmet.

In meinem File-Handler habe ich alle notwendigen Schritte programmiert, um die Dateien auszulesen und in die richtigen Verzeichnisse zu verschieben.

Die Webseite, auf der sich die Queue optisch darstellen lässt, habe ich danach erstellt. Dazu habe ich noch die sqlite Datenbank modelliert um die Daten der fertigen Jobs dort zu speichern.

Zwischendurch habe ich immer wieder Unit- und Integrations-Tests der einzelnen Komponenten gemacht, um deren Funktion und einwandfreie Kommunikation zu gewährleisten.

Nach der Fertigstellung des Projektes wurde darüber hinaus durch meinen Vorgesetzten ein "System- und Abnahme- Test" mit dem fertigen Tool durchgeführt.

Nach der erfolgreichen Durchführung des Tests waren die Arbeiten am Projekt abgeschlossen. Das Tool erfüllte alle definierten Inhalte und wurde für die weitere Verwendung freigegeben.