XVIII CONINC
CONGRESSO DE INICIAÇÃO CIENTÍFICA EDUVALE
ISSN 26755734
20 a 24 de outubro de 2025 – Avaré/SP

APLICAÇÃO WEB PARA EXTRAÇÃO E MANIPULAÇÃO DE EXTRATOS EM TXT PARA IMPORTAÇÃO NA DOMÍNIO SISTEMAS

Bruno David Martins – UNEDUVALE – dbrunobruno369@gmail.com

Bruno Aoki Tenorio– UNEDUVALE – brunotenorio15@hotmail.com

Yuri Nascimento Lopes – UNEDUVALE – yrnl700@gmail.com

Alexandre José Rufca Rossini – UNEDUVALE – alexandrejrossini@gmail.com

Marcos Alfredo Mendes do Rego – UNEDUVALE – marcos.rego@ead.eduvaleavare.com.br

ÁREA: Exatas e Tecnologia

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação web voltada à automação da extração e padronização de dados de extratos bancários em PDF, com finalidade de otimizar processos contábeis e permitir integração ao sistema Domínio Sistemas. A solução foi desenvolvida em Python 3.12 utilizando o *framework* Flask, com suporte do SQLAlchemy e banco MySQL para persistência de dados, além de implantação em infraestrutura baseada em Gunicorn e Nginx. O fluxo inicia com o *upload* do documento em ambiente seguro, seguido por pré-processamento de imagens com OpenCV, extração de texto em arquivos digitais por PyMuPDF e pdfplumber, ou reconhecimento por OCR com pytesseract. Os dados extraídos passam por normalização, com datas convertidas ao padrão ISO 8601 e valores para ponto flutuante, sendo organizados em arquivos TXT, sem armazenamento sensível no servidor, garantindo segurança e conformidade. Os resultados indicaram ganhos significativos de eficiência, reduzindo o tempo médio de lançamento manual de aproximadamente uma a duas horas para 10 a 20 minutos, alcançando acurácia acima de 98% em arquivos digitais. Apesar dos avanços, observaram-se limitações relacionadas ao alto custo computacional do OCR em documentos extensos ou de baixa qualidade e à dependência de expressões regulares, sujeitas a mudanças nos layouts bancários. Conclui-se que a aplicação atendeu aos objetivos propostos, proporcionando eficiência, padronização e segurança, ao mesmo tempo em que aponta para desafios futuros, como a otimização do OCR, uso de serviços mais avançados de reconhecimento textual e aplicação de técnicas de machine learning para maior resiliência diante da diversidade de extratos bancários.

PALAVRAS-CHAVE: Automação; Extratos Bancários; Flask; OCR; Domínio Sistemas.

INTRODUÇÃO

A automação de processos empresariais por meio de sistemas de informação

ISSN 26755734

baseados em tecnologia *web* tem se consolidado como estratégia fundamental para otimização operacional e redução de custos em organizações contemporâneas. No contexto específico da gestão financeira, o processamento manual de extratos bancários em formato Portable Document Format (PDF) representa um gargalo operacional significativo, demandando tempo considerável e apresentando suscetibilidade a erros humanos.

Os Sistemas de Informação baseados na tecnologia web proporcionam características distintivas que "permitem supor que o seu desenvolvimento apresenta diferenças em relação ao de sistemas não-web" (ANTONIO; 2005.), oferecendo maior flexibilidade, escalabilidade e facilidade de manutenção comparativamente às soluções desktop tradicionais. Nesse contexto, frameworks web como Flask destacam-se pela simplicidade arquitetural e modularidade, permitindo o desenvolvimento de aplicações robustas com menor complexidade implementacional.

A extração e padronização automatizada de dados bancários assume relevância estratégica quando consideramos que "a automação de rotinas manuais é fundamental para remover ou diminuir processos burocráticos, e obter foco somente em análises complexas, como predições de dados" (VENTURA, 2019.). A implementação de processos automáticos de Extração, Transformação e Carga (ETL) em ambientes Python propicia transformação eficiente de dados não estruturados em informações organizadas e utilizáveis.

Esta pesquisa apresenta o desenvolvimento e implementação de um software para extração de dados bancários, uma aplicação web construída com Flask que automatiza o processamento de extratos de múltiplas instituições brasileiras, convertendo documentos PDF em arquivos de texto (TXT) padronizados. O sistema integra tecnologias de reconhecimento óptico de caracteres (OCR), processamento de linguagem natural e gerenciamento de banco de dados relacional, constituindo uma solução completa para escritórios contábeis e departamentos financeiros que necessitam processar grandes volumes de documentos bancários com precisão e eficiência operacional.

MATERIAL E MÉTODOS

O ambiente de desenvolvimento da extratora baseou-se em Python 3.12 associado ao microframework Flask 3.1.0, estabelecendo as camadas de apresentação, controle e

configuração da aplicação. A persistência dos dados foi garantida pelo uso do SQLAlchemy como ORM conectado a um servidor MySQL 8.0. Para a execução em produção, o Gunicorn atuou como servidor WSGI por meio de *socket* Unix, enquanto o Nginx desempenhou a função de servidor web e proxy reverso, assegurando eficiência e balanceamento de carga.

A organização do código seguiu a arquitetura modular recomendada para projetos Flask, a criação e configuração da instância Flask encontram-se em app.py, ao passo que models.py define as entidades e relacionamentos do banco de dados. As rotas principais e os mecanismos de autenticação foram implementados em views.py e views_usuarios.py, respectivamente, pois foi necessário a implementação de uma tela de login, enquanto funções utilitárias de pré-processamento de documentos residem em helpers.py. A extração de dados específica para cada instituição bancária (Bradesco, Caixa, C6, Sicoob, Sicredi, Banco do Brasil e PagBank) foi encapsulada no módulo banks.py, em um monólito. Arquivos estáticos como folhas de estilo e scripts JavaScript estão agrupados na pasta static, e os templates HTML em Jinja2 encontram-se na pasta templates. O ponto de entrada do servidor Gunicorn está definido em wsgi.py.

O pipeline de processamento de extratos bancários inicia-se com o upload do arquivo PDF por meio de interface web protegida por sessão autenticada. Em seguida, as páginas escaneadas passam por processamento de imagem com OpenCV para redução de ruído e ajuste de brilho e contraste. A extração de texto utiliza PyMuPDF e pdfplumber em documentos digitais, já que são bibliotecas perfeitas para a manipulação de arquivos PDF puros, já o pytesseract foi utilizado para os casos onde o reconhecimento óptico de caracteres foi necessário, ou seja, em PDF digitalizados. Os blocos de transação são identificados por meio de expressões regulares, convertendo datas para o padrão ISO 8601 e valores monetários para ponto flutuante. Por fim, os registros resultantes são consolidados em arquivo TXT com colunas separadas por “;” que são elas: data, descrição, valor, tipo e saldo, sendo enviado diretamente ao usuário sem qualquer persistência de extratos no banco de dados ou no servidor, garantindo que nenhum dado sensível seja armazenado permanentemente.

O controle de versão foi mantido em repositório Git, adotando o modelo Git Flow para ramificações de desenvolvimento. A implantação em servidor Linux Ubuntu 22.04 LTS contemplou a criação de ambiente virtual Python (venv) para isolamento de dependências, o

uso de arquivo ".env" para variáveis sensíveis em ambiente de planejamento, e a utilização de Variáveis de Ambiente na área de produção, na configuração do serviço Systemd para gerenciamento do ciclo de vida do Gunicorn. A configuração do Nginx incluiu a aplicação de certificados *Let's Encrypt* para garantir comunicação via HTTPS, ou seja, criando uma rota segura, assegurando a total proteção dos dados sensíveis.

Além disso, foram utilizadas outras formas de segurança e conformidade, sendo uma delas a autenticação de usuários realizada com Flask-Bcrypt, armazenando-se apenas *hashes* de senha, e o Nginx que aplica limitação de taxa (*rate limiting*) nas rotas de login e de processamento de arquivos, foram também implementadas permissões em diretórios críticos como os: run, logs e uploads, sendo restritas ao usuário que não contiver a permissão www-data.

RESULTADOS E DISCUSSÃO

A aplicação proporcionou ganhos operacionais e qualitativos significativos no processamento de extratos bancários. Antes da adoção da ferramenta, o lançamento manual de cada extrato demandava cerca de uma a duas horas, já com o software, esse tempo foi reduzido drasticamente para uma média entre 10 e 20 minutos por documento, incluindo a revisão e validação dos dados extraídos. A validação se dá pela própria Domínio onde ela indica os possíveis erros e impede que sejam importados, retornando ao usuário os itens que não foram importados e o motivo. Já falando sobre o tempo de extração em PDFs digitais este se situa entre 3 e 5 segundos por documento puro, garantindo agilidade nos fluxos de trabalho. Além disso, a interface simplificada tornou o processo de lançamento mais intuitivo, e a padronização dos dados, especialmente em formatos TXT uniformes, melhorando a consistência das bases de dados contábeis.

Entretanto, o uso do OCR em páginas escaneadas apresenta limitações de desempenho: o processamento de cada página por meio do pytesseract pode consumir recursos elevados, resultando em tempos de 7 a 12 segundos por página, dependendo da qualidade da imagem. Para PDFs grandes, foi necessário subdividir arquivos em partes menores, sob risco de sobrecarregar a aplicação e provocar falhas de memória. Esses pontos evidenciam a necessidade de otimizações, seja por meio de paralelização do OCR, seja pela adoção de serviços de OCR mais performáticos.

ISSN 26755734

A manutenção da acurácia também depende da estabilidade dos layouts dos extratos. Alterações na posição de campos críticos, como data ou saldo, podem inviabilizar o *parsing* baseado em expressões regulares, exigindo atualização dos padrões de captura. Durante os testes, identificou-se que mudanças simples no layout de um extrato podiam comprometer a identificação automática de transações, demandando intervenção manual na manutenção do módulo banks.py.

Em suma, os resultados confirmam que a extratora atende aos objetivos de redução de tempo e padronização de dados, mas também apontam para desafios futuros em relação ao desempenho do OCR e à robustez frente a variações de layout. Recomenda-se investigar formas de otimizar o pipeline de OCR, implementar monitoramento automático de falhas de *parsing* e explorar técnicas de machine learning para aumentar a resiliência do reconhecimento de campos em extratos com formatações diversas.

CONCLUSÃO

O desenvolvimento do software demonstrou ser uma solução eficiente e prática para a automação do processamento de extratos bancários em PDF, convertendo-os em arquivos TXT padronizados e integráveis a sistemas contábeis. A arquitetura baseada em Flask, SQLAlchemy e MySQL, aliada ao uso de Gunicorn e Nginx para a orquestração de requisições, provou-se robusta e escalável, permitindo a rápida extração de dados em poucos segundos para documentos digitais e mantendo um fluxo de trabalho intuitivo para o usuário.

As melhorias de produtividade foram expressivas: o tempo de lançamento de extratos caiu de aproximadamente uma a duas horas por documento para apenas 10 a 20 minutos, e a padronização homogênea dos campos fortaleceu a consistência dos registros contábeis. O pipeline de processamento, composto por pré-tratamento de imagens, extração de texto com PyMuPDF, pdfplumber e OCR via pytesseract, normalização de dados e geração de CSV, funcionou conforme esperado, com acurácia superior a 98% em casos onde o PDF não continha rasuras e que foram aplicadas técnicas de pré-processamento.

Contudo, a utilização de OCR em páginas escaneadas evidenciou limitações de desempenho, exigindo subdivisão de arquivos muito grandes e apontando para a necessidade de otimizações de processamento ou adoção de soluções de reconhecimento mais eficientes.

Adicionalmente, a dependência de expressões regulares para parsing de layouts variáveis mostrou-se sensível a alterações nos extratos, o que demanda manutenção contínua dos módulos específicos de cada banco.

Futuros trabalhos devem focar na melhoria do desempenho do OCR, possivelmente por paralelização ou serviços especializados, e na incorporação de técnicas de aprendizado de máquina para detecção e adaptação automática a mudanças de *layout*. A implementação de *dashboards* de monitoramento também contribuirá para a identificação precoce de falhas no *parsing*. Em síntese, o software atendeu aos objetivos iniciais de agilidade, segurança e padronização, servindo como base sólida para evoluções que consolidem sua aplicação em contextos corporativos e contábeis de maior escala.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIO ZANET JUNIOR, Luiz; GERALDO DA ROCHA VIDAL, Antonio; RA USR - REVISTA DE ADMINISTRAÇÃO. **Construção de sistemas de informação baseados na Tecnologia Web**. [s.l.: s.n.], 2005. Disponível em: <https://file.notion.so/f/f/46deb299-e17e-817d-92f9-000352c18bea/9aacf419-a50b-4482-b397-fc8e8c938c5e/zeluizv41n3a02_(1).pdf?table=block&id=263eb299-e17e-8013-bcbb-edab72b3ab39&spaceId=46deb299-e17e-817d-92f9-000352c18bea&expirationTimestamp=1757052000000&signature=7vA9FtlvxBxDkn4B2RNnrCPkTrYwmTHYjDTVSyHWeY0&downloadName=zeluiz%2C%2Bv41n3a02+%281%29.pdf>. Acesso em: 4 set. 2025.

FLASK. *Documentação oficial do Flask.* Disponível em: https://flask.palletsprojects.com/. Acesso em: 5 jun. 2025.

NOTION. *Documentação – Notion.* Disponível em: https://www.notion.com/pt/help/guides/category/documentation. Acesso em: 5 jun. 2025.

PYTHON SOFTWARE FOUNDATION. *Python Release Python 3.12.6.* Disponível em: https://www.python.org/downloads/release/python-3126/. Acesso em: 5 jun. 2025

VENTURA, Marcelo; CASTRO, Bruno; COUTINHO, João. ETL E MODELAGEM DE EXTRATOS DO BANCO DO BRASIL: SELENIUM + TIDYVERSE. [s.l.: s.n.], 2019. Disponível em: <https://scispace.com/pdf/etl-e-modelagem-de-extratos-do-banco-do-brasil-selenium-6yto8g5emq.pdf>. Acesso em: 4 set. 2025.

VISUAL STUDIO CODE. *Documentação do Visual Studio Code.* Disponível em: https://code.visualstudio.com/docs. Acesso em: 5 jun. 2025.