

Использование Flask и Pandas для автоматической генерации писем

Эрдман Александр Алексеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данном исследовании приводится реализация программы для автоматической генерации писем. Программа разработана с использованием языка программирования Python, фреймворка Flask и модуля Pandas. Результатом исследования является описание программы и её практическое применение.

Ключевые слова: python, flask, web

Using Flask and Pandas to generate emails automatically

Erdman Alexander Alekseevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This study provides an implementation of a program for automatic letter generation. The program is developed using the Python programming language, the Flask framework and the Pandas module. The result of the research is a description of the program and its practical application.

Keywords: python, flask, web

1 Введение

1.2 Актуальность

Актуальность исследования заключается в необходимости автоматизации процесса генерации и отправки писем, что особенно важно в условиях современного бизнеса и административной деятельности. Ручная обработка больших объемов данных и создание персонализированных писем требует значительных временных и трудовых затрат. Внедрение программного обеспечения, способного автоматизировать эти процессы в различных организациях, позволяет повысить эффективность работы, снизить вероятность ошибок и улучшить качество предоставляемых услуг. Кроме того, использование веб-интерфейса для взаимодействия с программой делает ее доступной для широкого круга пользователей, не требуя установки специализированного программного обеспечения на локальные компьютеры, что приводит к уменьшению общей нагрузки на локальные машины.

1.2 Обзор исследований

Д.И. Валиуллина в своей статье привела описание библиотеки написанной на языке программирования python, которая позволяет проводить различные обработки над DataFrame, содержащими структурированную информацию [1]. Н.В. Григорьева рассмотрела алгоритм по первоначальной обработке данных, включающий обработку пропусков и дубликатов, изменение типов данных [2]. А.В. Неустроев в своём труде описал работу фреймворка Flask [3]. А.Д. Гончаренко и М.И. Давидовская в совместном исследовании привели реализацию новостного сайта с использованием фреймворка Flask и Python [4]. Н.А. Севостьянов и С.А. Любич провели анализ возможностей фреймворка Flask по обеспечению информационной безопасности веб-ресурсов [5].

1.3 Цель исследования

Целью исследования является реализация программы для автоматической генерации писем с использованием модулей Pandas и Flask.

2 Материалы и методы

Для решения поставленной задачи используется язык программирования Python с модулями Flask и Pandas. В качестве среды разработки выступает PyCharm.

3 Результаты и обсуждения

Разработка программы начинается с файла “app.py”, который является основным компонентом веб-приложения. Этот файл отвечает за обработку запросов, генерацию писем и создание отчетов. Программа использует фреймворк Flask для создания веб-интерфейса, что позволяет пользователям взаимодействовать с системой через браузер.

Вначале происходит импорт необходимых библиотек, включая Flask для создания веб-приложения, Pandas для работы с данными Excel, ReportLab для генерации PDF-документов, а также библиотеки для работы с файлами и сессиями. Далее инициализируется приложение Flask и устанавливается секретный ключ для использования сессий (рис. 1). Создается директория для загрузки файлов.

```
from flask import Flask, request, jsonify, send_file, render_template, session, send_from_directory
import pandas as pd
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.platypus import Paragraph
import io
import zipfile
import os

app = Flask(__name__)
app.secret_key = 'supersecretkey'

UPLOAD_FOLDER = 'uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)
```

Рисунок 1. Импорт модулей и инициализация приложения Flask

Далее регистрируется шрифт, поддерживающий Unicode, что важно для корректного отображения текста на русском языке в PDF-документах. Если данный шрифт не будет установлен, то текст на кириллице будет отображаться неверно в конечных PDF документах.

После прописываются два маршрута: отображения главной страницы и загрузки Excel файлов (рис. 2).

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route(rule: '/upload', methods=['POST'])
def upload_file():
    file = request.files['excelFile']
    df = pd.read_excel(file)
    people = df.to_dict(orient='records')
    session['people'] = people
    return jsonify({'people': people})
```

Рисунок 2. Маршруты отображения главной страницы и загрузки Excel файлов

Основной маршрут “/” отображает главную страницу приложения, используя шаблон “index.html”. Маршрут “/upload” обрабатывает загрузку Excel-файла. Пользователь загружает файл через форму на веб-странице, данные из файла читаются с помощью Pandas и сохраняются в сессии для последующего использования.

Для генерации писем на основе загруженных данных и введенных пользователем параметров, таких как тема письма, содержание, информация об отправителе и контакты отправителя, отвечает маршрут “/generate” (рис. 3).

Данные о людях извлекаются из сессии, и для каждого человека генерируется PDF-документ с письмом. Все сгенерированные PDF-документы упаковываются в ZIP-архив, а также создается отчет о проделанной работе.

Сразу после этого реализуется программный код, отвечающий за возможность пользователям скачивать сгенерированные ZIP-архив с письмами и отчет соответственно.

Маршрут “/create_template” позволяет пользователям создавать шаблонный Excel-файл для заполнения. Этот файл содержит необходимые колонки для ввода данных о людях.

```

@app.route('/generate', methods=['POST'])
def generate_letters():
    data = request.json
    subject = data['subject']
    content = data['content']
    sender_info = data['senderInfo']
    sender_contacts = data['senderContacts']

    people = session.get('people', [])

    pdf_files = []
    for person in people:
        pdf_buffer = io.BytesIO()
        c = canvas.Canvas(pdf_buffer, pagesize=A4)
        c.setFont(pdfFontname, 'DejaVuSans', size=12)

        c.drawString(450, 750, text=f'{person["адрес проживания"]}')

        c.drawCentredString(A4[0] / 2, 650, subject)

        styles = getSampleStyleSheet()
        style = ParagraphStyle(
            name='Normal',
            fontName='DejaVuSans',
            fontSize=12,
            leading=14,
            alignment=1
        )
        content_text = f'Ваажнаый, {person["ФИО"]}. {content.replace("{ФИО}", person["ФИО"])}'
        paragraph = Paragraph(content_text, style)
        paragraph.wrapOn(c, A4[0] - 100, A4[1] - 100)
        paragraph.drawOn(c, 50, 550)

        c.drawString(450, 50, sender_info)

        c.drawString(100, 50, text=f'Информация об отправителе: {sender_info}')
        c.drawString(100, 50, text=f'Контакты отправителя: {sender_contacts}')

    c.save()
    pdf_files.append(pdf_buffer)

    zip_filename = os.path.join(UPLOAD_FOLDER, 'letters.zip')
    with zipfile.ZipFile(zip_filename, mode='w') as zipf:
        for i, pdf_buffer in enumerate(pdf_files):
            zipf.writestr(f'letter-{i}.pdf', pdf_buffer.getvalue())

    report_filename = os.path.join(UPLOAD_FOLDER, 'report.pdf')
    report_c = canvas.Canvas(report_filename, pagesize=A4)
    report_c.setFont(pdfFontname, 'DejaVuSans', size=12)
    report_c.drawString(100, 750, text='Отчёт о проделанной работе')
    report_c.drawString(100, 750, text=f'Количество сгенерированных писем: {len(pdf_files)}')
    report_c.save()

    return jsonify({'zipUrl': f'/download/zip/{os.path.basename(zip_filename)}', 'reportUrl': f'/download/report/{os.path.basename(report_filename)}'})

@app.route('/download/zip/<filename>', methods=['GET'])
def download_zip(filename):
    return send_from_directory(UPLOAD_FOLDER, filename, as_attachment=True)

@app.route('/download/report/<filename>', methods=['GET'])
def download_report(filename):
    return send_from_directory(UPLOAD_FOLDER, filename, as_attachment=True)

@app.route('/create_template', methods=['POST'])
def create_template():
    columns = ['ФИО', 'регион проживания', 'адрес проживания', 'дата рождения', 'род деятельности']
    df = pd.DataFrame(columns=columns)
    output = io.BytesIO()
    df.to_excel(output, index=False)
    output.seek(0)
    return send_file(output, mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')

if __name__ == '__main__':
    app.run(debug=True)

```

Рисунок 3. Реализация генерации писем, формирования шаблона Excel файла, архива с готовыми письмами и отчётом

Следующим шагом является реализация “script.js”, который содержит JavaScript-код для взаимодействия с веб-приложением (рис. 4). Этот скрипт обрабатывает загрузку файлов, отображение списка людей, генерацию писем и создание шаблона Excel.

```

async function uploadFile() {
    const formData = new FormData(document.getElementById('uploadForm'));
    const response = await fetch('/upload', {
        method: 'POST',
        body: formData
    });
    const data = await response.json();
    displayPeopleList(data.people);
    document.getElementById('letterForm').style.display = 'block';
}

function formatDate(dateString) {
    const date = new Date(dateString);
    if (isNaN(date)) {
        return dateString;
    }
    const day = String(date.getDate()).padStart(2, '0');
    const month = String(date.getMonth() + 1).padStart(2, '0');
    const year = date.getFullYear();
    return `${day}.${month}.${year}`;
}

function displayPeopleList(people) {
    const peopleList = document.getElementById('peopleList');
    peopleList.innerHTML = '<h2>Список людей:</h2>';
    people.forEach(person => {
        const formattedDate = formatDate(person['дата рождения']);
        peopleList.innerHTML += `<p>${person['ФИО']}, ${person['регион проживания']}, ${person['адрес проживания']}, ${formattedDate}</p>`;
    });
}

async function generateLetters() {
    const subject = document.getElementById('subject').value;
    const content = document.getElementById('content').value;
    const senderInfo = document.getElementById('senderInfo').value;
    const senderContacts = document.getElementById('senderContacts').value;

    const response = await fetch('/generate', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ subject, content, senderInfo, senderContacts })
    });
    const data = await response.json();
    document.getElementById('downloadZip').href = data.zipUrl;
    document.getElementById('downloadReport').href = data.reportUrl;
    document.getElementById('downloadLinks').style.display = 'block';
}

async function createTemplate() {
    const response = await fetch('/create_template', {
        method: 'POST'
    });
    const blob = await response.blob();
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.style.display = 'none';
    a.href = url;
    a.download = 'template.xlsx';
    document.body.appendChild(a);
    a.click();
    window.URL.revokeObjectURL(url);
}

```

Рисунок 4. Скрипт JavaScript

Последним программируемым файлом является “index.html”, представляющий из себя HTML-шаблон, который отображается пользователю. Он содержит формы для загрузки файлов, ввода данных для генерации писем и кнопки для выполнения различных действий (рис. 5).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Генерация писем</title>
</head>
<body>
  <h1>Генерация писем</h1>
  <form id="uploadForm" enctype="multipart/form-data">
    <input type="file" id="excelFile" name="excelFile" accept=".xlsx, .xls"><br><br>
    <button type="button" onclick="uploadFile()">Загрузить файл</button>
  </form>
  <button type="button" onclick="createTemplate()">Создать шаблон Excel</button>
  <div id="peopleList"></div>
  <form id="letterForm" style="display:none;">
    <input type="text" id="subject" placeholder="Тема письма"><br><br>
    <textarea id="content" placeholder="Содержание письма"></textarea><br><br>
    <input type="text" id="senderInfo" placeholder="Информация об отправителе"><br><br>
    <input type="text" id="senderContacts" placeholder="Контакты отправителя"><br><br>
    <button type="button" onclick="generateLetters()">Сформировать письма для отправки</button>
  </form>
  <div id="downloadLinks" style="display:none;">
    <a id="downloadZip" href="#">Скачать архив с письмами</a><br>
    <a id="downloadReport" href="#">Скачать отчёт</a>
  </div>
  <script src="{ url_for('static', filename='script.js') }"></script>
</body>
</html>
```

Рисунок 5. HTML разметка страницы

Функционал программы реализован. Для его работоспособности происходит тестирование (рис 6, 7, 8).

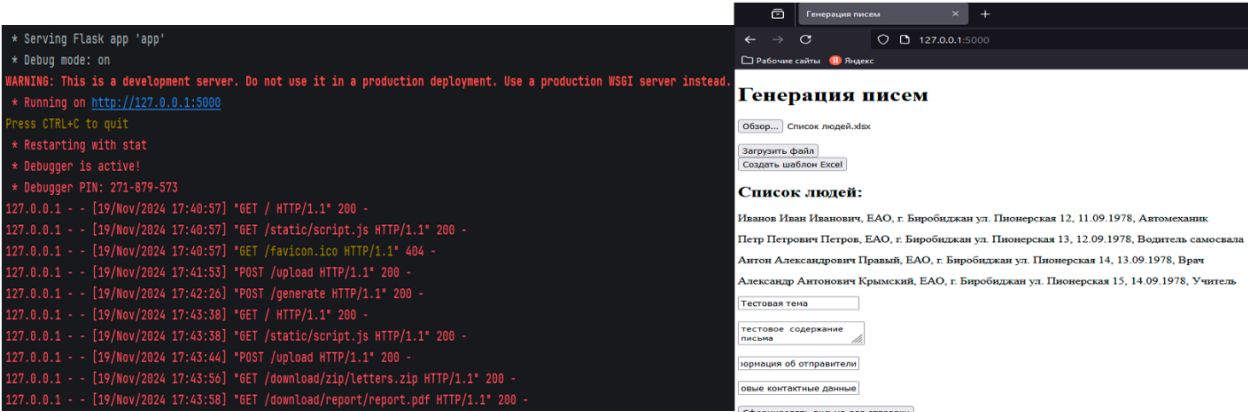


Рисунок 6. Запуск программы и внешний вид веб-интерфейса

	A	B	C	D	E	F
1	ФИО	регион проживания	адрес проживания	дата рождения	род деятельности	
2	Иванов Иван Иванович	ЕАО	г. Биробиджан ул. Пионерская 12	11.09.1978	Автомеханик	
3	Петр Петрович Петров	ЕАО	г. Биробиджан ул. Пионерская 13	12.09.1978	Водитель самосвала	
4	Антон Александрович П	ЕАО	г. Биробиджан ул. Пионерская 14	13.09.1978	Врач	
5	Александр Антонович И	ЕАО	г. Биробиджан ул. Пионерская 15	14.09.1978	Учитель	
6						
7						
8						

Отчёт о проделанной работе
Количество сгенерированных писем: 4

Рисунок 7. Таблица с данными и сгенерированный отчёт работы программы

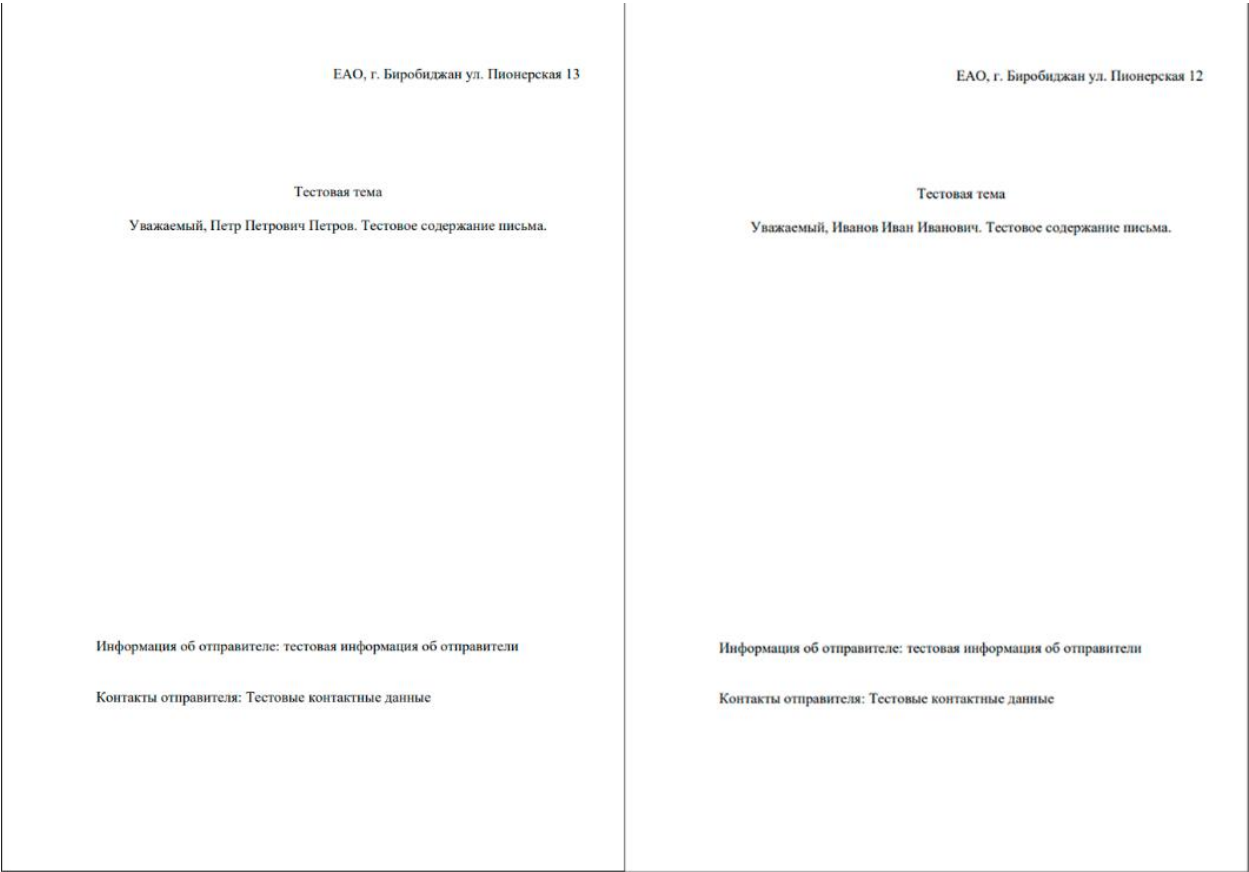


Рисунок 8. Сгенерированные письма

Вывод

В результате исследования была разработана программа на языке программирования Python с модулями Flask и Pandas для автоматической генерации писем для каждого получателя из загружаемого списка. Программа имеет веб-интерфейс, что в свою очередь предоставляет возможность пользователям работать с данной системой через сеть. В результате работы программы были получены сгенерированные письма, отчёт о работе программы и сгенерированный шаблон таблицы для заполнения данных о получателях.

Библиографический список

1. Валиуллина Д.И. Применение библиотеки Pandas для обработки информации на языке Python // В сборнике: Учёный XXI века. сборник

- статей Международного научно-исследовательского конкурса. Пенза, 2022. С. 28-31.
2. Григорьева Н.В. Алгоритм обработки данных с использованием Python Pandas // В сборнике: Информационные технологии, системный анализ и управление (ИТСАУ-2022). Сборник трудов XX Всероссийской научной конференции молодых ученых, аспирантов и студентов. Ростов-на-Дону, Таганрог, 2022. С. 226-229.
 3. Неустроев А.В. Использование в фреймворке Flask библиотеки Sqlalchemy // Наука и образование сегодня. 2016. № 10 (11). С. 21-23.
 4. Гончаренко А.Д., Давидовская М.И. Проектирование веб-приложения на основе Flask и Python // Научные исследования XXI века. 2023. № 1 (21). С. 40-44.
 5. Севостьянов Н.А., Любич С.А. Анализ возможностей фреймворка Flask по обеспечению различных аспектов защиты данных // В сборнике: Информационная безопасность: современная теория и практика. Сборник научных трудов студентов, аспирантов и преподавателей по материалам межвузовской научно-практической конференции. Ответственный редактор З.В. Семенова. 2018. С. 103-106.