|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра компьютерных технологий и программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| Ст. преп. |  |  |  | М.Д. Поляк |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |  |
| --- | --- | --- | --- |
| вид практики | производственная | |  |
| тип практики | По получению профессиональных умений и опыта профессиональной деятельности | |  |
| на тему индивидуального задания | | «Бэкенд. Парсинг отчетов студентов и их автоматическая проверка. Форма odt.» |  |
|  | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Столяров Никита Сергеевич |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.04 |  | Программная инженерия |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 02 |  | Проектирование программных систем |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4134к |  |  |  | Столяров Н.С. |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2024

**Задание  
Создать новую ветку в репозитории и подготовить пул-реквест, реализующий описанный ниже функционал.**На Python + FastAPI реализовать REST эндпоинт для проверки отчёта на корректность

* POST /report/odt - получение программы конференции
* В качестве конфигурационного файла используются yml файлы. Подробнее см. [#7](https://github.com/markpolyak/lab-grader-new/issues/7).

**POST /report/odt**

**Входные данные**course\_id — название конфигурационного файла

lab\_id - Идентификатор лабораторной работы, связанной с курсом.  
student\_fio -ФИО студента, для которого проверяется отчет.  
group\_number -Номер группы студента.

File - Загружаемый файл в формате ODT, который подлежит проверке.

**Логика работы**

1. Проверка типа файла: При получении запроса осуществляется проверка, что загружаемый файл имеет расширение .odt. Если файл не соответствует этому формату, возвращается ошибка 400 с соответствующим сообщением.
2. Чтение конфигурации лабораторной работы: Бэкенд пытается загрузить конфигурацию лабораторной работы из YAML файла, используя course\_id и lab\_id. Если файл не найден, возвращается ошибка 400.
3. Сохранение загруженного файла: Загруженный ODT файл сохраняется на сервере для дальнейшей обработки.
4. Извлечение текста из ODT файла: С помощью библиотеки odf.opendocument происходит загрузка и парсинг ODT файла, извлечение текстовых элементов и их анализ.
5. Проверка титульного листа: С помощью регулярного выражения осуществляется поиск титульного листа, который должен содержать информацию о конференции, включая название и год. Если титульный лист не найден, возвращается ошибка 400.
6. Валидация данных: Извлеченные данные из титульного листа проверяются на соответствие с переданными параметрами:

* Сравнение ФИО студента.
* Проверка номера группы.
* Проверка наличия имени и должности преподавателя.

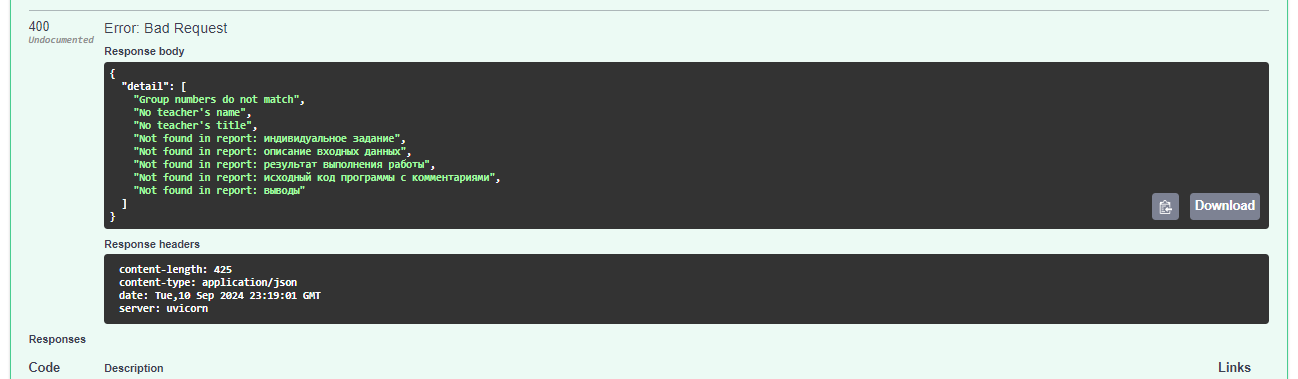
1. Проверка содержания отчета: Проверяется наличие всех необходимых заголовков в основном тексте отчета. Если какие-либо заголовки отсутствуют, формируется список ошибок.

**Выходные данные**

При успешном выполнении проверки, если все данные корректны, возвращается сообщение о том, что отчет правильный. В случае обнаружения ошибок, возвращается ошибка 400 с описанием всех найденных проблем.

Таким образом, данный эндпоинт обеспечивает функциональность проверки отчетов в формате ODT, гарантируя высокую степень точности и надежности в обработке данных, что является важным аспектом в образовательной среде.

**Тестовые данные**



Пример с использованием абсолютно не корректного отчёта

Пример с использованием корректного отчёта

**Обработка ошибок**

Если **spreadsheet\_id** у конференции пустой:  
code: 404  
**"detail":** **"spreadsheet\_id is empty"**

Если id конференции не существует:  
code: 404  
**"detail": "Conference not found"**

Если токен не указан:  
code: 403  
**"detail": "Not authenticated"**

Если токен указан неправильный:

code: 401  
**"detail": "Invalid token"**

**Код**

main.py

|  |
| --- |
| from fastapi import FastAPI  from fastapi import Request  app = FastAPI(  title='Grader-New'  )  from routers import report\_router  ################################################################################  app.include\_router(report\_router)  ################################################################################ |

odt.py

|  |
| --- |
| from fastapi import UploadFile, File, Form, HTTPException  from odf.opendocument import load  from odf.text import P  from odf import text, draw  from xml.etree import ElementTree as ET  from typing import Annotated  import os  import re  from globals import config  from globals import read\_config  from globals import parse\_student\_info  from globals import fix\_full\_name  from globals import transform\_name\_format  from globals import extract\_digits  from globals import fix\_full\_name  from globals import all\_names\_varians  def read\_odt(file\_path):  document = OpenDocumentText(file\_path)  paragraphs = []  for element in document.getElementsByType(P):  paragraphs.append(element.firstChild.data)  return paragraphs  async def odt(  course\_id: Annotated[str, Form()],  lab\_id: Annotated[str, Form()],  student\_fio: Annotated[str, Form()],  group\_number: Annotated[str, Form()],  file: UploadFile = File(...)  ):  """  Проверка odt файла на корректность  """  if not file.filename.endswith('.odt'):  raise HTTPException(status\_code=400, detail="File type not supported. Please upload an ODT file")  ############################################################################  try:  lab\_config = read\_config(f"courses/{course\_id}.yaml", lab\_id)  except FileNotFoundError:  raise HTTPException(status\_code=400, detail="Error reading course file: File not found")  print(lab\_config)  reports = [s.lower() for s in lab\_config["report"]]  print(reports)  teacher\_name = lab\_config['staff'][1]['name']  teacher\_name = fix\_full\_name(teacher\_name)  teacher\_name\_with\_space, teacher\_name\_without\_space = transform\_name\_format(teacher\_name)  teacher\_title = lab\_config['staff'][0]['title']  teacher\_title\_no\_spaces = teacher\_title.replace(" ", "")  ############################################################################  file\_path = os.path.join(config.get("reports")["path"], file.filename)  with open(file\_path, 'wb') as f:  content = await file.read()  f.write(content)  doc = load(file\_path)  xml\_content = doc.xml()  # Парсим XML и извлекаем текст  root = ET.fromstring(xml\_content)  text\_elements = root.findall('.//text:p', namespaces={'text': 'urn:oasis:names:tc:opendocument:xmlns:text:1.0'})  extracted\_text = []  for elem in text\_elements:  text = ''.join(elem.itertext())  if len(text) > 0:  extracted\_text.append(text)  title\_page = []  # Регулярное выражение для поиска "Санкт-Петербург" и года  target\_pattern = re.compile(r"Санкт-Петербург\s+\d{4}", re.IGNORECASE)  # Ищем титульный лист  for i, text in enumerate(extracted\_text):  if target\_pattern.search(text):  title\_page = extracted\_text[:i]  title\_lower\_page = [s.lower() for s in extracted\_text[:i]]  extracted\_lower\_text = [s.lower() for s in extracted\_text[:i]]  extracted\_lower\_text = [s.lower() for s in extracted\_text[i + 1:]]  break  else:  raise HTTPException(status\_code=400, detail="Failed to detect cover page")  error\_list = []  ############################################################################  # проверяем титульник  student\_info = parse\_student\_info(" ".join(title\_page))  print(student\_info)  full\_extracted\_name = all\_names\_varians(student\_info["student\_name"])  print(full\_extracted\_name)  full\_student\_name = fix\_full\_name(student\_fio)  print(full\_student\_name)  if (extract\_digits(student\_info["group\_number"]) != extract\_digits(group\_number)):  error\_list.append("Group numbers do not match")  if (full\_extracted\_name != full\_student\_name):  error\_list.append("Student name do not match")  for teacher in [teacher\_name, teacher\_name\_with\_space, teacher\_name\_without\_space]:  find\_flag = False  for line in title\_lower\_page:  if line.find(teacher.lower()) != -1:  find\_flag = True  break  if not find\_flag:  error\_list.append("No teacher's name")  break  for title in [teacher\_title, teacher\_title\_no\_spaces]:  find\_flag = False  for line in title\_lower\_page:  if line.find(title.lower()) != -1:  find\_flag = True  break  if not find\_flag:  error\_list.append("No teacher's title")  break  # for teacher in [teacher\_name, teacher\_name\_with\_space, teacher\_name\_without\_space]:  # find\_flag = False  # for line in title\_lower\_page:  # if line.find(teacher.lower()) != -1:  # find\_flag = True  # break  # if not find\_flag:  # error\_list.append("No student name")  ############################################################################  # проверяем основной отчёт (заголовки)  for report in reports:  find\_flag = False  for line in extracted\_lower\_text:  if line.find(report) != -1:  find\_flag = True  break  if not find\_flag:  error\_list.append("Not found in report: " + report)  # os.remove(file.filename)  # return {"checklist": reports, "errors": error\_list, "title": title\_page, "body": extracted\_text}  if len(error\_list) > 0:  raise HTTPException(status\_code=400, detail=error\_list)  return {"result": "The report is correct"} |

globals.py

|  |
| --- |
| import os  from config import Config  config = Config("config.ini")  config.print()  ################################################################################  # import gspread  # from oauth2client.service\_account import ServiceAccountCredentials  # scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]  # credentials = ServiceAccountCredentials.from\_json\_keyfile\_name('gconfig.json', scope)  # gc = gspread.authorize(credentials)  ################################################################################  import re  #  def all\_names\_varians(name):  # Удаляем все пробелы  name = re.sub(r'\s+', '', name)  # Проверка форматов имен  patterns = [  r'([А-ЯЁA-Z][а-яёa-z]+)([А-ЯЁA-Z])\.([А-ЯЁA-Z])\.', # "ФамилияИ.О."  r'([А-ЯЁA-Z][а-яёa-z]+) ([А-ЯЁA-Z])\.([А-ЯЁA-Z])\.', # "Фамилия И.О."  r'([А-ЯЁA-Z])\.([А-ЯЁA-Z])\.([А-ЯЁA-Z][а-яёa-z]+)', # "И.О.Фамилия"  r'([А-ЯЁA-Z])\.([А-ЯЁA-Z])\. ([А-ЯЁA-Z][а-яёa-z]+)' # "И.О. Фамилия"  ]  for pattern in patterns:  match = re.match(pattern, name)  if match:  if len(match.groups()) == 3:  surname = match.group(1)  initial1 = match.group(2)  initial2 = match.group(3)  return f"{surname} {initial1}.{initial2}."  # Если ни один формат не подошел, возвращаем исходное имя  return name  def fix\_full\_name(full\_name):  name\_parts = re.sub(r'\s+', ' ', full\_name).strip().split()  if len(name\_parts) == 3:  return f"{name\_parts[0]} {name\_parts[1][0]}.{name\_parts[2][0]}."  return full\_name  def parse\_student\_info(title\_text):  # Паттерн для поиска номера группы  group\_pattern = r'((?:\d{4}К)|(?:М\d{3})|(?:\d{4}K)|(?:M\d{3})|(?:\d+))'  # Паттерн для поиска ФИО  name\_pattern = r'([А-ЯЁA-Z]\.[А-ЯЁA-Z]\.\s\*[А-ЯЁA-Z][а-яёa-z]\*|[А-ЯЁA-Z][а-яёa-z]+ [А-ЯЁA-Z]\.[А-ЯЁA-Z]\.|[А-ЯЁA-Z]\.[А-ЯЁA-Z]\.[А-ЯЁA-Z][а-яёa-z]+)'  # Полный паттерн для поиска группы и ФИО с промежуточными данными  full\_pattern = rf'{group\_pattern}\s\*(?:\S\*\s\*)?{name\_pattern}'  match = re.search(full\_pattern, title\_text, re.UNICODE)  group\_number, student\_name = None, None  if match:  group\_number = match.group(1)  student\_name = match.group(2)  return {  "group\_number": group\_number,  "student\_name": student\_name  }  def transform\_name\_format(name):  parts = name.split()  if len(parts) == 2:  return f"{parts[1]} {parts[0]}", f"{parts[1]}{parts[0]}"  return name, name  ################################################################################  def extract\_digits(input\_string):  return ''.join(filter(str.isdigit, input\_string))  ################################################################################  def verify\_student\_registration(spreadsheet\_id, student\_name, group\_name):  try:  # Открытие листа группы  sheet = gc.open\_by\_key(spreadsheet\_id).worksheet(group\_name)  # Проверка наличия имени студента в списке  return student\_name in sheet.col\_values(2)  except gspread.exceptions.WorksheetNotFound:  return False  ################################################################################  import yaml  def read\_config(file, lab\_id):  lab\_key = str(lab\_id)  with open(file, 'r', encoding='utf-8') as file:  config = yaml.safe\_load(file)  course = config.get('course', None)  return {  "report": course.get('labs', None).get(lab\_key, None)["report"],  "staff": course.get('staff', None)  } |

**Вывод**

В результате выполнения задания по созданию REST API на базе Python и FastAPI для проверки отчетов в формате ODT, можно сделать следующие выводы:

Проект успешно реализовал функционал, позволяющий проверять корректность отчетов, загружаемых в формате ODT, с использованием современных технологий и подходов, включая:

* Обработку загружаемых файлов: Реализована проверка типа загружаемого файла, что предотвращает загрузку неподдерживаемых форматов и обеспечивает корректность данных.
* Чтение и анализ ODT файлов: Использование библиотеки для работы с ODT файлами позволяет извлекать текстовые данные и проводить их анализ, что является ключевым для проверки содержания отчетов.
* Валидация данных: Проект включает в себя механизмы для проверки соответствия данных, таких как имя студента, номер группы и информация о преподавателе, что гарантирует высокую степень точности и надежности проверок.
* Обработка ошибок: Реализована система обработки ошибок, которая информирует пользователя о проблемах с загружаемыми отчетами, что улучшает взаимодействие с API и повышает его удобство.
* Гибкость и расширяемость: Архитектура проекта позволяет легко добавлять новые функции и улучшения, что делает его подходящим для дальнейшего развития и адаптации под изменяющиеся требования.

Этот проект не только демонстрирует навыки разработки веб-приложений с использованием FastAPI и Python, но и подчеркивает важность точности и надежности в обработке и проверке данных, что особенно актуально в образовательной среде.

**Источники**

* https://fastapi.tiangolo.com/
* https://pypi.org/project/odfpy/