Техническое задание на систему генерации PDF и обработки данных

1. Общие сведения

1.1 Наименование системы

Автоматизированная система генерации PDF-документов и обработки данных "РепортЛаб"

1.2 Основание для разработки

* Приказ №123 от 15.05.2025;
* ГОСТ 34.602-89 "Техническое задание на создание автоматизированной системы";
* ГОСТ 19.201-78 "Техническое задание. Требования к содержанию и оформлению".

1.3 Назначение разработки

Создание программного модуля для формирования PDF-документов в заданном формате с использованием библиотеки ReportLab.

Обучение работе с библиотекой ReportLab.

Обработка динамических данных с использованием базы данных PostgreSQL, установленной на удаленном Linux-сервере.

2. Назначение и цели создания системы

2.1 Назначение

Автоматизация создания PDF-документов с предопределенной структурой, таблицами и текстами;

Обработка данных из удаленной базы данных PostgreSQL;

Генерирование Excel – файлов;

2.2 Цели

* Автоматизация процесса создания PDF-документов;
* Минимизация ошибок при оформлении;
* Обеспечение соответствия документов установленным стандартам;
* Получение опыта работы с динамическими данными и удаленными базами данных.

3. Требования к программному обеспечению

3.1 Входные данные

* Текстовые данные для заполнения полей документа (заголовки, основной текст, таблицы);
* Настройки оформления (шрифты, размеры, отступы);
* Динамические данные из базы данных PostgreSQL.

Данные из модели CharactersModels на Django, содержащие следующие поля:

Id - Уникальный внутренний идентификатор записи. Целочисленный тип данных, ограничение по символам 10.

id\_voc\_mi - Идентификатор связки РУ-Модель. Целочисленный тип данных, ограничение по символам 10.

id\_model - Идентификатор модели. Целочисленный тип данных, ограничение по символам 10.

okpd2\_code - Код ОКПД2.Целочисленный тип данных, ограничение по символам 50.

okpd2\_label - Наименование ОКПД2.Строковый тип данных, ограничение по символам 150.

id\_voc\_mi\_characteristic - ID характеристики. Целочисленный тип данных, ограничение по символам 10.

quality\_characteristic\_value - Значение качественной характеристики. Строковый тип данных, ограничение по символам 150.

quantitative\_characteristic\_value - Значение количественной характеристики. Строковый тип данных, ограничение по символам 150.

unit - Единица измерения. Строковый тип данных, ограничение по символам 25.

ktru\_code - Код КТРУ.Целочисленный тип данных, ограничение по символам 25.

3.2 Выходные данные

Сформированный PDF-документ, содержащий:

* Таблицу с текстовыми данными в верхней части страницы;
* Заголовок документа, расположенный по центру;
* Основной текст, выровненный по ширине страницы.

Сформированный список словарей с данными из базы данных.

Формат представления данных.  
Каждый словарь будет содержать ключи:

Пример выходных данных:

[

{'id': '36511', ' id\_voc\_mi ': '483869', ' id\_model: '11126', ' okpd2\_code ': '32.50.50190', ' okpd2\_label ': 'Изделия медицинские, в том числе хирургические, прочие, не включенные в другие группировки', ' id\_voc\_mi\_characteristic ': '1108', ' quality\_characteristic\_value ': 'EDTA', ' quantitative\_characteristic\_value ': '', ' unit ': '', ' ktru\_code ': '32.50.50.000-00000576'},

]

3.3 Пример кода:

from django.core.management.models import BaseCommand

from ...models import CharactersModels

class Command(BaseCommand):

help = 'Вывод всех записей из базы данных в виде списка словарей'

def handle(self, \*args, \*\*kwargs):

# Извлекаем все записи из модели CharactersModels

characters\_models = CharactersModels.objects.all()

# Генератором списков создаём список словарей

data = [

{

' id ': characters\_model. id,

' id\_voc\_mi ': characters\_model. id\_voc\_mi,

' id\_model ': characters\_model. id\_model,

' okpd2\_code ': characters\_model. okpd2\_code,

' okpd2\_label ': characters\_model. okpd2\_label,

' id\_voc\_mi\_characteristic ': characters\_model. id\_voc\_mi\_characteristic,

' quality\_characteristic\_value ': characters\_model. quality\_characteristic\_value,

' quantitative\_characteristic\_value ': characters\_model. quantitative\_characteristic\_value,

' unit ': characters\_model. Unit,

' ktru\_code ': characters\_model. ktru\_code

}

for characters\_model in characters\_models

]

# Выводим список словарей в консоль

for record in data:

print(record)

3.4 Требования к оформлению PDF-документа

* Таблица с видимыми границами ячеек и настраиваемым выравниванием текста;
* Использование шрифтов PT Astra Serif Bold и PT Astra Serif Regular;
* Размер шрифта: 14 пунктов для заголовков, 12 пунктов для основного текста;
* Межстрочный интервал: 1,5.

3.5 Функциональные требования

* Поддержка пользовательских TTF-шрифтов;
* Автоматическое разбиение текста на строки;
* Гибкие настройки таблиц (выравнивание, отступы, ширина столбцов);
* Генерация титульной страницы на основе шаблона;
* Подключение к удаленной базе данных PostgreSQL;
* Выполнение SQL-запросов для получения динамических данных.

3.6 Нефункциональные требования

* Производительность: время генерации документа не должно превышать 5 секунд;
* Кроссплатформенность: работа на Windows, macOS и Linux;
* Простота использования: понятная структура кода для дальнейшего сопровождения.

4. Требования к программной реализации

4.1 Используемые библиотеки

* ReportLab для создания PDF-документов;
* Драйвер базы данных PostgreSQL (psycopg2, psycopg2-binary);
* OpenPyExcel.(версии 2.5.14)

4.2 Поддерживаемые форматы шрифтов

* TrueType Fonts (TTF).

4.3 Структура программы

* Регистрация пользовательских шрифтов;
* Настройка стилей текста;
* Формирование таблиц с заданными параметрами;
* Добавление текста в документ;
* Подключение к базе данных и получение данных;
* Генерация PDF-файла.

4.4 Среда выполнения

* Python версии 3.6 и выше;
* IDE PyCharm для работы со скриптом;
* Удаленный Linux-сервер с установленным PostgreSQL;
* pgAdmin, установленный на локальном ПК;
* Django(последняя версия).

4.5 Взаимодействие с пользователем

* Входные данные предоставляются в виде структуры данных (JSON, словарь или аналогичный формат);
* Выходной файл сохраняется в указанной директории.

5. Стадии и этапы разработки

5.1 Этапы разработки

* Проектирование (2 рабочих дня);
* Реализация (5 рабочих дней);
* Тестирование (2 рабочих дня);
* Документация (1 рабочий день).

6. Требования к документации

* Инструкция по установке и настройке модуля;
* Примеры использования кода;
* Комментарии в коде для ключевых функций;
* Руководство пользователя;
* Руководство администратора;
* Программа и методика испытаний;
* Инструкция по настройке удаленного сервера PostgreSQL;
* Документация по структуре базы данных.

7. Порядок контроля и приемки

* Проверка соответствия функциональности требованиям;
* Тестирование с использованием тестовых данных;
* Проверка оформления выходного PDF-документа на соответствие шаблону;
* Согласование итогового решения с заказчиком;
* Функциональное тестирование всех модулей;
* Нагрузочное тестирование при работе с большими объемами данных.

8. Источники разработки

* ГОСТ 34.602-89 "Техническое задание на создание автоматизированной системы";
* Образец титульной страницы из интернет-источника;
* Внутренние стандарты организации по оформлению отчетов;
* Документация по PostgreSQL;
* Документация по ReportLab.

9. Технические детали

Обработка данных

Перед выводом данных в EXCEL, необходимо реализовать следующие шаги:

Генераторы списков (List Comprehensions)

Генератор списков в Python — это удобный и эффективный способ создания новых списков на основе существующих данных, с применением условий или преобразований. В нашем случае мы использовали генераторы списков для обработки данных, извлеченных из базы данных, а затем их корректного представления в Excel.

Работа с Excel с использованием библиотеки openpyxl

Для работы с Excel-файлами мы использовали библиотеку openpyxl, которая позволяет:

Создавать новые Excel-файлы.

Заполнять их данными.

Настроить таблицы, стили, шрифты и т.д.

Извлечение данных из PostgreSQL, включая заказчиков и исполнителей.

Обработка данных перед выводом в Excel

В этом коде важно, что, прежде чем записать данные в Excel, мы их предварительно обрабатываем. Обработка данных заключается в проверке каждого элемента строки на наличие цифры '6'. Если цифры '6' нет, то этот элемент меняется на 'null'. Мы используем это условие в генераторе списка.

Фильтрация данных на уровне SQL-запроса

Мы также обсуждали, как сделать фильтрацию данных на уровне SQL-запроса. Это важно для оптимизации работы программы, чтобы не загружать в память ненужные данные.

В Django создаем проект и приложение с помощью команд:

django-admin startproject[название проекта]

python manage.py starapp[название приложения]

В файл settings.py в поле INSTALLED\_APPS пишем название нашего приложения. В этом же файле в поле DATABASES задаём подключение к нашей базе данных.

В файле models.py создаём модель данных.

Далее нужно создать команду, для этого в корневой папке создаём папку management, в ней создаём папку commands, уже в этой папке создаём mycommand.py с кодом нашей команды.

Для использования нашей команды в терминале необходимо ввести:

python manage.py mycommand