1. ОБЩИЕ СВЕДЕНИЯ

1.1. Наименование системы

Программный модуль для автоматизированной генерации PDF-документов.

1.2. Основание для разработки

Разработка осуществляется на основании задания пользователя и в соответствии с нормативными документами:

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 на основе данных, полученных из базы данных PostgreSQL, с заданными параметрами форматирования.

1.4. Требования к системе

Поддержка библиотеки ReportLab для формирования PDF.

Настройка стилей текста и таблиц (шрифты, отступы, размеры).

Формирование структуры документа в соответствии с представленным шаблоном.

Подключение и работа с базой данных PostgreSQL.

Регистрация и использование шрифтов PT Astra Serif (Bold, Regular).

2. НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

2.1. Назначение

Модуль предназначен для автоматического формирования PDF-документов, содержащих текстовую информацию и таблицы, с учетом заданного форматирования.

2.2. Цели

Автоматизация процесса генерации PDF-документов.

Сокращение временных затрат на подготовку документов.

Соответствие документации утвержденным стандартам (ГОСТ, корпоративные требования).

3. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

3.1. Входные данные

Данные из базы данных PostgreSQL, содержащие:

ID

Заказчика

Исполнителя

Настройки форматирования документа (шрифты, отступы, размеры таблицы).

3.2. Выходные данные

Сформированный PDF-документ с данными в заданном формате.

3.3. Требования к оформлению PDF-документа

Структура документа:

Таблица заголовка (в верхней части страницы)

Левая ячейка: выравнивание текста по левому краю.

Правая ячейка: выравнивание текста по левому краю с отступом 100 мм.

Объединение ячеек для заказчиков с несколькими исполнителями.

Заголовок:

Шрифт PT Astra Serif Bold, размер 12pt.

Выравнивание по центру.

Основной текст:

Шрифт PT Astra Serif Regular, размер 10pt.

Выравнивание по ширине страницы.

3.4. Функциональные требования

Настройка шрифтов (регистрация TTF-шрифтов в ReportLab).

Генерация таблицы с поддержкой объединения ячеек и границ.

Выравнивание текста и задание отступов в правой ячейке таблицы.

Обработка данных из PostgreSQL перед выводом в PDF.

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ РЕАЛИЗАЦИИ

4.1. Используемая библиотека

ReportLab – библиотека для генерации PDF.

psycopg2 – библиотека для подключения к PostgreSQL.

4.2. Поддерживаемые форматы шрифтов

TrueType Fonts (TTF).

4.3. Среда выполнения

Python 3.7+.

4.4. Структура кода

Регистрация шрифтов (PT Astra Serif Bold, Regular) в ReportLab.

Подключение к БД и извлечение данных.

Формирование заголовочной таблицы с учетом отступов.

Создание основной таблицы с данными.

Генерация PDF-файла.

4.5. Производительность

Формирование PDF-документа не должно превышать 5 секунд при обработке до 100 записей.

5. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Проектирование

Анализ требований.

Разработка структуры документа.

Разработка

Написание кода генерации PDF.

Настройка стилей и таблиц.

Тестирование

Проверка соответствия макету.

Оптимизация производительности.

6. ТРЕБОВАНИЯ К ДОКУМЕНТАЦИИ

Комментарирование кода с описанием функций и параметров.

Руководство пользователя с примерами использования модуля.

7. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ

Формирование тестовых PDF-файлов для проверки соответствия макету.

Валидация отображения текста и таблиц в PDF.

Сравнение с эталонным образцом для удостоверения в корректности вывода данных.

8. ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ

8.1. 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. Пример запроса

9. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

9.1 Входные данные

Данные из базы данных PostgreSQL, содержащие следующие поля:

TF\_OKATO — код субъекта Российской Федерации.

IDUMP — код условий оказания медицинской помощи.

BZTSZ — значение базовой ставки.

DATEBEG — дата начала действия записи.

DATEEND — дата окончания действия записи.

9.2 Выходные данные

Сформированный список словарей с данными из базы данных.

Формат представления данных

Каждый словарь будет содержать ключи:

TF\_OKATO — строковое значение.

IDUMP — строковое значение.

BZTSZ — строковое значение.

DATEBEG — дата в формате YYYY-MM-DD.

DATEEND — дата в формате YYYY-MM-DD.

9.3 Пример выходных данных:

[

{'TF\_OKATO': '12345', 'IDUMP': 'abc', 'BZTSZ': '1000', 'DATEBEG': '2023-01-01', 'DATEEND': '2023-12-31'},

{'TF\_OKATO': '67890', 'IDUMP': 'xyz', 'BZTSZ': '2000', 'DATEBEG': '2023-02-01', 'DATEEND': '2023-12-31'}

]

9.4 Функциональные требования

Извлечение данных: Использование Django ORM для извлечения всех записей из таблицы модели BaseRate.

Генерация списка словарей: Сформировать список словарей с помощью генераторов списков, где каждый словарь содержит данные одной строки таблицы.

Вывод данных в консоль: Печать полученного списка словарей в консоль с помощью функции print().

10. ТРЕБОВАНИЯ К ПРОГРАММНОЙ РЕАЛИЗАЦИИ

10.1 Используемые библиотеки

Django ORM: для взаимодействия с базой данных PostgreSQL.

Python: для реализации логики генерации списка словарей.

10.2 Среда выполнения

Python 3.7+.

Django (версия соответствующая проекту).

Структура кода

Извлечение данных из базы данных: Использование Django ORM для получения всех объектов модели BaseRate из базы данных.

Генерация списка словарей с помощью генераторов списков: Сформировать список словарей, используя генераторы списков.

Вывод данных: Список словарей выводится в консоль с помощью функции print().

10.3 Пример кода:

from django.core.management.base import BaseCommand

from ...models import BaseRate

class Command(BaseCommand):

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

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

# Извлекаем все записи из модели BaseRate

base\_rates = BaseRate.objects.all()

# Генератором списков создаём список словарей

data = [

{

'TF\_OKATO': base\_rate.TF\_OKATO,

'IDUMP': base\_rate.IDUMP,

'BZTSZ': base\_rate.BZTSZ,

'DATEBEG': base\_rate.DATEBEG,

'DATEEND': base\_rate.DATEEND

}

for base\_rate in base\_rates

]

# Выводим список словарей в консоль

for record in data:

print(record)

11. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

11.1 Проектирование

Анализ требований.

Разработка структуры модели и команды.

Разработка

Написание кода для извлечения данных из базы данных.

Реализация генератора списков для создания списка словарей.

Вывод данных в консоль.

11.2 Тестирование

Проверка правильности извлечения данных из базы данных.

Тестирование корректности формирования списка словарей.

Проверка вывода данных в консоль.

12. ТРЕБОВАНИЯ К ДОКУМЕНТАЦИИ

Комментарирование кода: Комментарии, объясняющие функции и параметры, используемые в коде.

Руководство пользователя: Руководство, описывающее, как использовать команду для извлечения данных и вывода их в консоль.

13. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ

Формирование тестовых данных в базе и проверка вывода в консоль.

Валидация корректности данных в выводе: проверка соответствия формата списка словарей с требованиями.

Сравнение с эталонным выводом в консоль.

14. ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ

Оптимизация: Учитывая, что данные могут быть объемными, важно, чтобы извлечение и обработка данных происходили быстро и эффективно. Использование генераторов списков помогает оптимизировать работу с памятью.

Поддержка изменений в базе данных: Модуль должен быть гибким и позволять легко адаптироваться к изменениям в структуре базы данных или добавлению новых полей в модель BaseRate.