

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ПРОГРАММНОЕ ПОДКЛЮЧЕНИЕ К СЕРВЕРУ POSTGRESQL

Отчёт о лабораторной работе № 4
по дисциплине «Базы данных»

Выполнил: студентка гр. 430-2

_____ Лузинсан А.А.

«___»_____ 2022 г.

Проверил: ассистент каф. АСУ

_____ Яблонский Я. В.

«___»_____ 2022 г.

Томск 2022

Оглавление

| | |
|--------------------------------------|----|
| 1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ..... | 2 |
| 2 ОПИСАНИЕ ТРЕБОВАНИЙ К ОТЧЁТАМ..... | 3 |
| 3 ЛИСТИНГ..... | 4 |
| 3.1 Листинг файла settings.py..... | 4 |
| 3.2 Листинг файла main.py..... | 6 |
| 4 ОПИСАНИЕ ПРОЦЕССА РЕАЛИЗАЦИИ..... | 17 |
| 5 ВЫВОДЫ..... | 22 |

1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Цель работы: научиться создавать программы, взаимодействующие с сервером СУБД.

2 ОПИСАНИЕ ТРЕБОВАНИЙ К ОТЧЁТАМ

В данной лабораторной работе необходимо разработать программу, которая бы генерировала отчёт в формате, представленном на рисунке 2.1, где видно, что для отображения списка подписчиков необходимо выбрать столбцы «Улица», «Дом», «Квартира/Офис» и «ФИО». Отчёт формируется соединением таблиц `t_subs`, `t_pod` и `t_address`.

| Список подписчиков | | | |
|--------------------|-----|---------------|----------------------|
| Улица | Дом | Квартира/Офис | ФИО |
| Комсомольская | 133 | 34 | Рогов Олег |
| | | 45 | Дубов Василий |
| Ленина | 40 | | Иванов Иван Иванович |

Рисунок 2.1 — Примерный вид отчёта

3 ЛИСТИНГ

3.1 Листинг файла settings.py

```
import psycopg2

from psycopg2 import Error, sql

from contextlib import closing

from psycopg2.extras import DictCursor

import sys

from psycopg2 import OperationalError, errorcodes, errors, connect

import pandas as pd

from tabulate import tabulate

import dearpygui.dearpygui as dpg

DEFAULT_USER = "postgres"

DEFAULT_PASSWORD = "52981073"

DEFAULT_HOST = "localhost"

DEFAULT_PORT = "5432"

DEFAULT_DATABASE = "subscription"

dpg.create_context()
```

```
with dpg.theme() as global_theme:

    with dpg.theme_component(dpg.mvAll):

        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (77, 7, 143), category=dpg.mvThemeCat_Core)

        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_Core)

    with dpg.theme_component(dpg.mvInputInt):

        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (30, 77, 70), category=dpg.mvThemeCat_Core)

        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_Core)

    with dpg.theme_component(dpg.mvText):

        dpg.add_theme_color(dpg.mvThemeCol_FrameBg, (15, 61, 131), category=dpg.mvThemeCat_Core)

        dpg.add_theme_style(dpg.mvStyleVar_FrameRounding, 5, category=dpg.mvThemeCat_Core)

dpg.bind_theme(global_theme)

with dpg.font_registry():

    with dpg.font(f'/usr/share/fonts/truetype/freefont/FreeSerifBold.ttf', 20, default_font=True, id="Default font"):

        dpg.add_font_range_hint(dpg.mvFontRangeHint_Cyrillic)

dpg.bind_font("Default font")
```

```

def on_exit():

    connection, cursor = dpq.get_item_user_data('auth')

    if connection:

        cursor.close()

        connection.close()

        print("Соединение с PostgreSQL закрыто")

dpq.set_global_font_scale(1.25)

dpq.set_exit_callback(on_exit)

```

3.2 Листинг файла main.py

```

from settings import *

def add_task_table(columns, answer):

    dpq.delete_item('task_table', children_only=True)

    for column in columns:

        dpq.add_table_column(label=column, parent='task_table')

    for row in answer:

        with dpq.table_row(parent='task_table'):

            for field in row:

                dpq.add_text(default_value=field, color=(150, 30, 200))

def task_request():

    connection, cursor = dpq.get_item_user_data('auth')

    dpq.delete_item('task_error')

    try:

```

```

request = f"SELECT street, house, apartment, fio " \

        f"FROM t_subs " \

        f"JOIN t_pod on t_pod.id=t_subs.idpod " \

        f"JOIN t_address on t_address.id=t_pod.idaddr " \

        f"GROUP BY street, house, apartment, fio;"

cursor.execute(request)

answer = cursor.fetchall()

columns = ['Улица', 'Дом', 'Квартира\\Офис', 'ФИО']

add_task_table(columns, answer)

df = pd.DataFrame(answer, columns=columns)

array = df[columns[:-1]]

list_index = [list(df[column]) for column in array]

index = pd.MultiIndex.from_arrays(list_index, names=columns[:-1])

s = pd.Series(list(df['ФИО']), index=index, name='Список
подписчиков')

table = tabulate(df, headers=columns, tablefmt='psql')

print(f"Таблица по заданию:\n{s}")

with open('task.txt', 'w') as file:

    file.write(f"\t\t{s.name}\n")

    file.write(table)

s.to_csv('task.csv')

except Exception as err:

    # print_psycopg2_exception(err, 'task_table')

```



```
dpg.add_text(tag='task_error', default_value=Error, color=(255, 0, 0),
before='task_table')
```

```
def send_request():

    connection, cursor = dpg.get_item_user_data('auth')

    print(connection, cursor)

    list_columns = dpg.get_item_user_data('list_columns')

    values = []

    dpg.delete_item('success_insert')

    dpg.delete_item('insert_error')

    try:

        for field in list_columns:

            values.append(dpg.get_value(f"{field}"))

        values = tuple(values)

        print(values)

        table_name = dpg.get_item_user_data('list_tables')

        column_names = ', '.join(list_columns)

        insert = f"INSERT INTO {table_name}({column_names}) VALUES
{values};"

        print(insert)

        cursor.execute(insert)

        connection.commit()

        dpg.add_text(tag='success_insert', default_value=f"Строка {values}
успешно вставлена",
```

```

        color=(0, 255, 0), before='table_records')

    with dpg.table_row(parent='table_records'):

        for value in values:

            dpg.add_text(default_value=value)

except Exception as err:

    print("Error")

    dpg.add_text(tag='insert_error', default_value=Error, color=(255, 0, 0),
before='table_records')

    connect_database('table_records')


def output_records(table_name, list_columns):

    connection, cursor = dpg.get_item_user_data('auth')

    dpg.delete_item('output_records_error')

    dpg.delete_item('table_records', children_only=True)

    try:

        request = f"SELECT * FROM {table_name};"

        cursor.execute(request)

        list_info_records = cursor.fetchall()

        print(f"Записи таблицы: \n{list_info_records}")

        for column in list_columns:

            dpg.add_table_column(label=column, parent='table_records')

        for row in list_info_records:

            with dpg.table_row(parent='table_records'):

```

```

        for field in row:

            dpg.add_text(default_value=field)

    except (Exception, Error) as error:

        dpg.add_text(tag='output_records_error', default_value=Error,
color=(255, 0, 0), before='list_records')

def add_fields(list_columns, list_info_columns):

    dpg.delete_item('box_input_fields', children_only=True)

    with dpg.group(parent='box_input_fields'):

        for number, field in enumerate(list_columns):

            if list_info_columns[number][1] == 'integer':

                dpg.add_input_int(tag=field)

            else:

                dpg.add_input_text(tag=field, hint=field)

def output_columns(sender, table_name):

    dpg.configure_item('list_tables', user_data=table_name)

    connection, cursor = dpg.get_item_user_data('auth')

    dpg.delete_item('output_columns_error')

    try:

        request = f"SELECT column_name, data_type, column_default,
is_nullable, character_maximum_length FROM information_schema.columns
WHERE table_name='{table_name}';"

        cursor.execute(request)

```

```

list_info_columns = cursor.fetchall()

print(list_info_columns)

list_columns = [column[0] for column in list_info_columns]

dpg.configure_item('list_columns', items=list_columns, show=True,
                    num_items=len(list_columns), user_data=list_columns)

dpg.configure_item('insert_button', show=True)

print(f"Поля таблицы: {list_columns}")

add_fields(list_columns, list_info_columns)

output_records(table_name, list_columns)

except (Exception, Error) as error:

    dpg.add_text(tag='output_columns_error', default_value=Error,
color=(255, 0, 0), before='list_columns')


def output_tables(cursor):

    dpg.delete_item('output_list_error')

    try:

        request = "SELECT table_name FROM information_schema.tables
WHERE table_schema='public'"

        cursor.execute(request)

        tables = [table[0] for table in cursor.fetchall()]

        dpg.configure_item('list_tables', items=tables, show=True,
num_items=len(tables))

        print(f"Таблицы в базе данных: {tables}")

    except (Exception, Error) as error:

```

```

        dpg.add_text(tag='output_list_error', default_value=Error, color=(255,
0, 0), before='list_tables')

def connect_database(wrap: str):

    dpg.delete_item('connect_error')

    auth_data = {'user': dpg.get_value('user'),

                  'password': dpg.get_value('password'),

                  'host': dpg.get_value('host'),

                  'port': dpg.get_value('port'),

                  'database': dpg.get_value('database')}

    try:

        connection = psycopg2.connect(user=auth_data['user'],

                                       password=auth_data['password'],

                                       host=auth_data['host'],

                                       port=auth_data['port'],

                                       database=auth_data['database'])

        cursor = connection.cursor(cursor_factory=DictCursor)

        dpg.set_item_user_data('auth', [connection, cursor])

        return connection, cursor, auth_data['database']

    except (Exception, Error) as error:

        dpg.add_text(tag='connect_error', before=wrap, color=(255, 0, 0),

                    default_value=f"Ошибка при работе с PostgreSQL: {error}")

def open_database():

```

```
dpg.delete_item('connect_success')

connection, cursor, name_database = connect_database('send_auth')

# Распечатать сведения о PostgreSQL

print(connection.get_dsn_parameters())

dpg.add_text(tag='connect_success', before='send_auth', color=(0, 255, 0),
             default_value=f"База данных успешно подключена.")

dpg.configure_item('label_database', default_value=f"Список таблиц ба-
зы данных: {name_database}", show=True)

dpg.configure_item('task_button', show=True)

output_tables(cursor)
```

```
with dpg.window(label="AUTHORIZATION", modal=True, show=False, tag="auth", no_title_bar=True, autosize=True):  
    dpg.add_input_text(label=":USER", tag='user', default_value=DEFAULT_USER)  
    dpg.add_input_text(label=":PASSWORD", tag='password', default_value=DEFAULT_PASSWORD, password=True)  
    dpg.add_separator(tag='sep')  
    dpg.add_input_text(label=":HOST", tag='host', default_value=DEFAULT_HOST)  
    dpg.add_input_text(label=":PORT", tag='port', default_value=DEFAULT_PORT)  
    dpg.add_input_text(label=":DATABASE", tag='database', default_value=DEFAULT_DATABASE)  
    with dpg.group(horizontal=True, tag='send_auth'):  
        dpg.add_button(label="Connect", width=75, callback=open_database)  
        dpg.add_button(label="Cancel", width=75, callback=lambda: dpg.configure_item("auth", show=False))  
with dpg.window(label="Main", tag="Main"):  
    with dpg.menu_bar():  
        dpg.add_menu_item(label="Log in", callback=lambda: dpg.configure_item("auth", show=True))  
    dpg.add_text(tag='label_database', default_value='Список таблиц базы данных: ', show=False, color=(170, 4, 170))  
    dpg.add_listbox(tag='list_tables', callback=output_columns, show=False)
```

```
dpg.add_separator()
```

```
with dpg.group(horizontal=True):
```

```
    dpg.add_listbox(tag='list_columns', show=False)
```

```
    dpg.add_group(tag='box_input_fields')
```

```
dpg.add_button(tag='insert_button', label='Добавить запись', callback=send_request, show=False)
```

```
dpg.add_separator()
```

```
dpg.add_table(tag='table_records', row_background=True,
```

```
    resizable=True, policy=dpg.mvTable_SizingStretchProp,
```

```
    borders_innerH=True, borders_outerH=True, borders_innerV=True,
```

```
    borders_outerV=True)
```

```
dpg.add_separator()
```

```
dpg.add_button(tag='task_button', label="Вывод таблицы по заданию", callback=task_request, show=False)
```

```
dpg.add_table(tag='task_table', row_background=True,
```

```
    resizable=True, policy=dpg.mvTable_SizingStretchProp,
```

```
    borders_innerH=True, borders_outerH=True, borders_innerV=True,
```



```
borders_outerV=True)
```

```
dpg.set_primary_window("Main", True)
```

```
dpg.create_viewport(title='SQL CLIENT')
```

```
dpg.setup_dearpygui()
```

```
dpg.show_viewport()
```

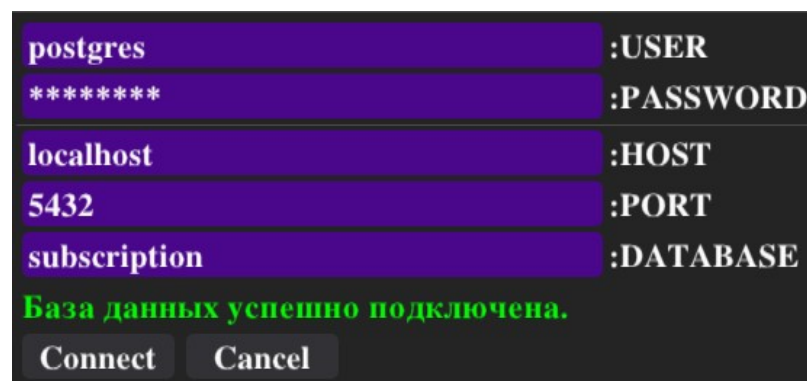
```
dpg.start_dearpygui()
```

```
dpg.destroy_context()
```

4 ОПИСАНИЕ ПРОЦЕССА РЕАЛИЗАЦИИ

В качестве библиотеки, с помощью которой осуществлялось подключение к серверу PostgreSQL был выбран модуль `psycopg2`, потому как он является одной из самых продвинутых и широко используемых систем управления реляционными базами данных. `psycopg2` чрезвычайно популярен по многим причинам, среди которых есть открытый код, его расширяемость и способность обрабатывать различные типы приложений и различные нагрузки. Помимо этого, приложение было реализовано в виде GUI. В качестве библиотеки был выбран `deargui`.

После нажатия в панели меню кнопки `Log In`, начинается процесс заполнения данных в окно, представленное на рисунке 4.1, и инициирования соединения с базой данных. За это отвечает функция `connect`, куда передаются аргументы: имя пользователя, пароль, сервер, порт и указывается база данных. Далее инициализируется курсор для последующего выполнения операций с базой данных. В случае выброса исключение, оно ловится и выводится в интерфейс пользователя.



| | |
|--------------|-----------|
| postgres | :USER |
| ***** | :PASSWORD |
| localhost | :HOST |
| 5432 | :PORT |
| subscription | :DATABASE |

База данных успешно подключена.

Connect Cancel

Рисунок 4.1 — Вход в базу данных subscription

Далее пользователь должен закрыть окно с авторизацией. Сразу после соединения с базой данных приложение запрашивает список таблиц и выводит в элемент `listbox`, как показано на рисунке 4.2. За выполнение запроса к базе данных отвечает функция `execute()`. После запроса вызывается функция

commit() для объекта connection для того чтобы сохранить запись в базе дан-
ны. Запрос таблиц базы данных выглядит следующим образом:

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema='public';
```

Вместе с выводом таблиц у пользователя появляется возможность вы-
вести таблицу, сформированную по рисунку 2.1, по нажатию кнопки «Вывод
таблицы по заданию».

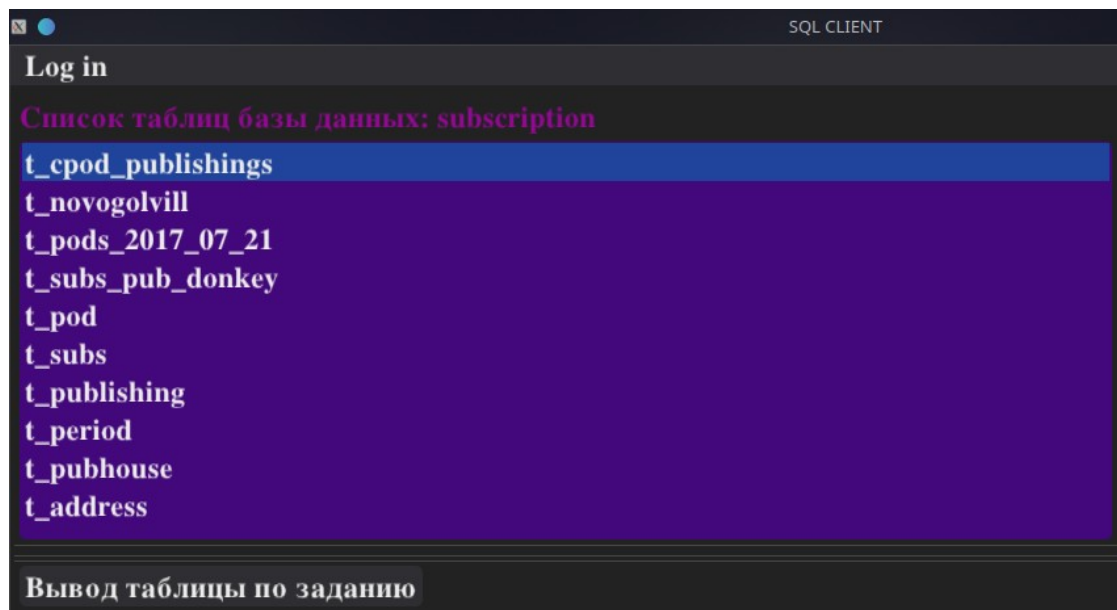


Рисунок 4.2 — Вывод таблиц базы данных subscription

Нажимая на элемент вышеприведённого списка приложение за-
прашивает список полей выбранной таблицы. Запрос выглядит следующим
образом:

```
SELECT column_name, data_type, column_default, is_nullable,
character_maximum_length
FROM information_schema.columns
WHERE table_name='{table_name}';
```

Здесь вместо table_name в форматную строку вставляется имя выбран-
ной таблицы. Помимо имён столбцов, запрашиваются также данные: тип дан-
ных поля, значение по умолчанию, NULL поле, максимальная длина, величи-

на стоки/значения. Сведения о типе данных будут использоваться для добавления ограничений при занесении новых строк в выбранную таблицу.

Таким образом, после нажатия на определённую таблицу выводится список полей для занесения новых данных, а также приложением запрашиваются записи которые эта таблица содержит и выводит в виде таблицы, показанной на рисунке 4.3.

| id | post_code | region | city | street | house | apartment |
|----|-----------|------------------|--------------|-----------------|-------|-----------|
| 1 | 847275 | Пивной край | Воблинок | Трезвых | 20 | 52 |
| 2 | 634043 | Райская долина | Мухомуромск | Нечистой силы | 74 | 438 |
| 4 | 029817 | Досадный регион | Новоголвилль | Провал Синицина | 404 | 137 |
| 5 | 105736 | Томная область | Шпзарнумск | Ленивка | 1 | 1 |
| 6 | 847277 | Пивной край | Воблинок | Трезвых | 21 | 88 |
| 7 | 634045 | Райская долина | Мухомуромск | Нечистой силы | 101 | 555 |
| 8 | 029819 | Досадный регион | Новоголвилль | Провал Синицина | 500 | 404 |
| 9 | 029818 | Досадный регион | Новоголвилль | Горшок | 77 | 47 |
| 10 | 029819 | Досадный регион | Новоголвилль | Золотова | 38 | 26 |
| 11 | 029820 | Досадный регион | Новоголвилль | Золотова | 39 | 47 |
| 3 | 572963 | Стеклянный округ | Скрежетальск | Артефактов | 336 | 77 |

Рисунок 4.3 — Вывод записей таблицы t_address

Далее протестируем ввод данных в таблицу. Для начала попробуем ввести некорректные данные: длина почтового кода в таблице t_address не равна 6. В результате получаем ошибку, как показано на рисунке 4.4. Напротив, если ввести данные, представленные на рисунке 4.5, то запрос будет осуществлён корректно и занесённые данные добавятся в таблицу ниже. Данный запрос реализуется за счёт следующей форматной строки:

```
INSERT INTO {table_name}({column_names}) VALUES {values};
```

| | |
|-----------|----------------|
| id | 74 |
| post_code | 16459 |
| region | Алтайский Край |
| city | Шорох |
| street | Связистов |
| house | 5 |
| apartment | 9 |

Добавить запись

Некорректно введенные данные. Попробуйте ещё раз.

Рисунок 4.4 — Обработка ошибки

| | |
|-----------|----------------|
| id | 74 |
| post_code | 164598 |
| region | Алтайский Край |
| city | Шорох |
| street | Связистов |
| house | 5 |
| apartment | 9 |

Добавить запись

Строка (74, '164598', 'Алтайский Край', 'Шорох', 'Связистов', 5, 9) успешно вставлена

| id | post_code | region | city | street | house | apartment |
|----|-----------|------------------|--------------|-----------------|-------|-----------|
| 1 | 847275 | Пивной край | Воблинск | Трезвых | 20 | 52 |
| 2 | 634043 | Райская долина | Мухомуромск | Нечистой силы | 74 | 438 |
| 4 | 029817 | Досадный регион | Новоголвилль | Провал Синицина | 404 | 137 |
| 5 | 105736 | Томная область | Шизариумск | Ленивка | 1 | 1 |
| 6 | 847277 | Пивной край | Воблинск | Трезвых | 21 | 88 |
| 7 | 634045 | Райская долина | Мухомуромск | Нечистой силы | 101 | 555 |
| 8 | 029819 | Досадный регион | Новоголвилль | Провал Синицина | 500 | 404 |
| 9 | 029818 | Досадный регион | Новоголвилль | Горшок | 77 | 47 |
| 10 | 029819 | Досадный регион | Новоголвилль | Золотова | 38 | 26 |
| 11 | 029820 | Досадный регион | Новоголвилль | Золотова | 39 | 47 |
| 3 | 572963 | Стекланный округ | Скрежетальск | Артефактов | 336 | 77 |
| 74 | 164598 | Алтайский Край | Шорох | Связистов | 5 | 9 |

Рисунок 4.5 — Успешная вставка строки

Последним заданием стал вывод таблицы, проиллюстрированной на рисунке 4.6, по нажатию кнопки «Вывод таблицы по заданию», реализуемый посредством следующего вызова:

```
SELECT street, house, apartment, fio
FROM t_subs
JOIN t_pod on t_pod.id=t_subs.idpod
JOIN t_address on t_address.id=t_pod.idaddr
GROUP BY street, house, apartment, fio;
```

| Вывод таблицы по заданию | | | |
|--------------------------|-----|---------------|-----------------------------------|
| Улица | Дом | Квартира\Офис | ФИО |
| Золотова | 39 | 47 | Шорохов Олег Евгеньевич |
| Горшок | 77 | 47 | Фамильяров Геннадий Иннокентьевич |
| Провал Синицина | 404 | 137 | Камрад Артём Златоустович |
| Провал Синицина | 500 | 404 | Добрыйвечер Добромир Миронов |
| Артефактов | 336 | 77 | Какаев Аркадий Акакиевич |
| Ленивка | 1 | 1 | Холявка Елена Васильевна |
| Трезвых | 20 | 52 | Пьяных Татьяна Николаевна |
| Золотова | 38 | 26 | Шорохова Евгения Олеговна |

Рисунок 4.6 — Вывод таблицы по заданию

Для формирования отчёта используются библиотеки `pandas` и `tabulate`. С помощью `pandas` формируется сгруппированный по столбцам «Улица», «Дом» и «Квартира/Офис» объект `Series`. Таким образом строки для некоторой группы не будут дублироваться. Из данного объекта `Series` составляется `csv` файл. С другой стороны, с помощью библиотеки `tabulate` формируется отчёт, который далее выводится в файл `task.txt`, представленные на рисунке 4.7.

| | | | | | |
|----------|---------------------------------|---------------------------|--|-----------------|--|
| task.txt | | | | | |
| 1 | » Список подписчиков | | | | |
| 2 | +-----+-----+-----+-----+-----+ | | | | |
| 3 | | Улица | | Дом | |
| 4 | | Квартира\Офис | | | |
| 5 | | ФИО | | | |
| 6 | | +-----+-----+-----+-----+ | | | |
| 7 | | 0 | | Золотова | |
| 8 | | 1 | | Горшок | |
| 9 | | 2 | | Провал Синицина | |
| 10 | | 3 | | Провал Синицина | |
| 11 | | 4 | | Артефактов | |
| 12 | | 5 | | Ленивка | |
| 13 | | 6 | | Трезвых | |
| 14 | | 7 | | Золотова | |
| 15 | | +-----+-----+-----+-----+ | | | |

Рисунок 4.7 — Содержимое отчёта `task.txt`

5 ВЫВОДЫ

В ходе выполнения данной лабораторной работы я научилась подключаться к серверу PostgreSQL посредством библиотеки `psycopg2` языка программирования Python, а также выводить список таблиц, список полей таблицы, список записей некоторой таблицы, добавлять новую запись, и формировать необходимый отчёт.