**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра CАПР**

отчет

**по лабораторной работе №4**

**по дисциплине «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ KOTLIN»**

Тема: **Подключение базы данных к Telegram-боту.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0302 |  | Кузнецов К.Е. |
| Студент гр. 0302 |  | Головатюк К.А. |
| Преподаватель |  | Кулагин М.В. |

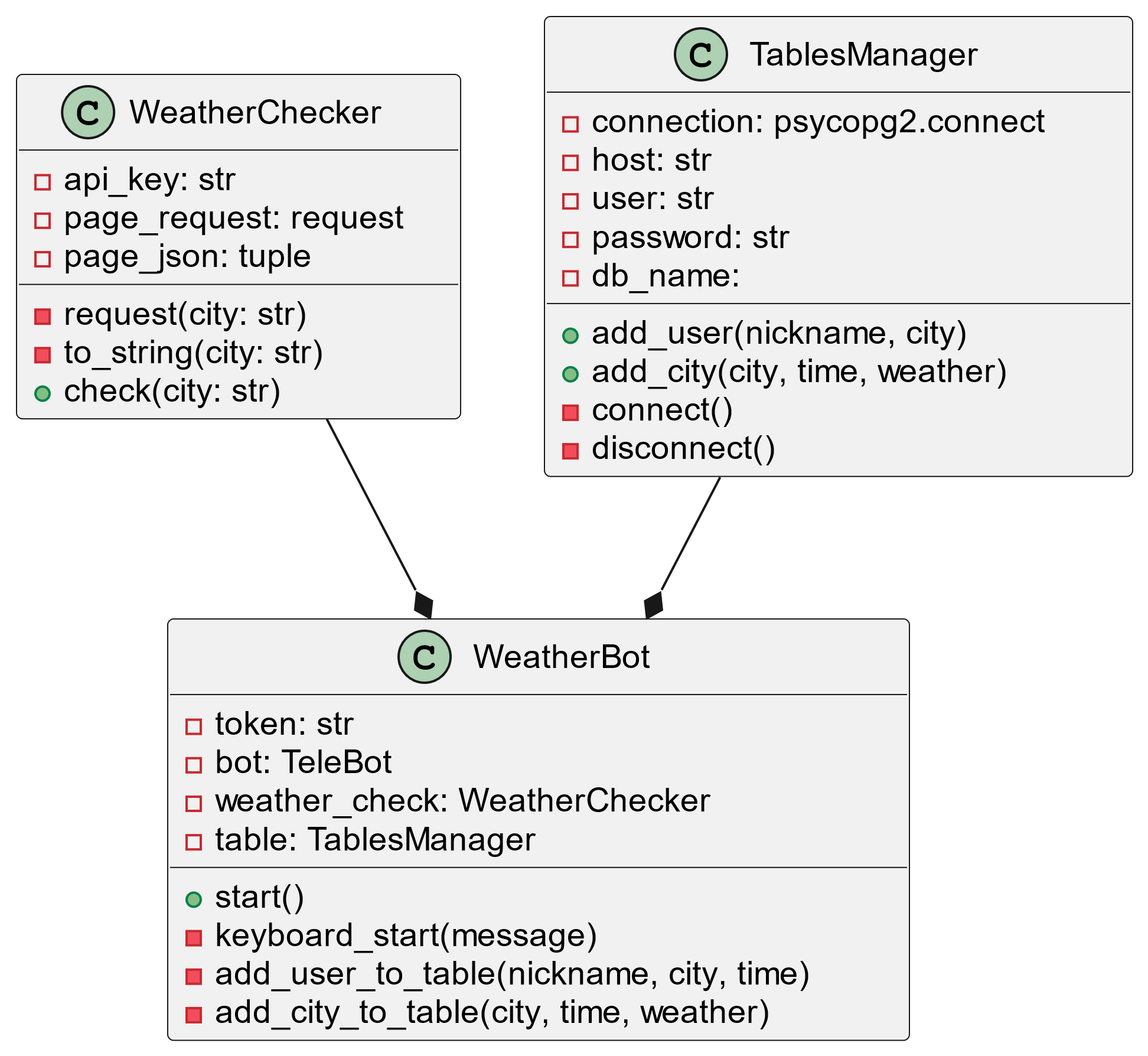
Санкт-Петербург

2022

# Цель работы

В рамках практической работы требуется реализовать взаимодействие разработанного бота с таблицами в базе данных.

# Спецификация программы



# Описание интерфейса пользователя программы

Пользователю выводится информация и кнопка “Погода”. После нажатия на кнопку, требуется ввести название города. Если название города введено верно, то выводится погода в этом городе, иначе выводится информация об ошибке. Если запрос успешен, в базу данных (PostgreSql) заносится никнейм пользователя, город, время запроса и погода. Так же была написана программа инициализации таблицы.

# Текст программы

main.py:

from controler import WeatherBot

if \_\_name\_\_ == '\_\_main\_\_':

test = WeatherBot()

test.start()

pass

controler.py:

import telebot

from weather import WeatherChecker

from telebot import types

from tablesManager import TablesManager

from datetime import datetime

class WeatherBot:

def \_\_init\_\_(self, path\_weather\_api\_key="weather\_api\_key", path="telegram\_bot\_token"):

self.\_\_token = None

try:

self.\_\_token = open(path, "r").read()

except FileNotFoundError:

self.\_\_token = None

print("Неверный путь")

return

self.\_\_bot = telebot.TeleBot(self.\_\_token)

self.\_\_weather\_check = WeatherChecker(path\_weather\_api\_key)

self.\_\_table = TablesManager()

@self.\_\_bot.message\_handler(content\_types=['text'])

def start(message):

self.\_\_keyboard\_start(message)

def weather(message):

result = self.\_\_weather\_check.check(message.text)

self.\_\_bot.send\_message(message.from\_user.id, result[1])

if result[0]:

self.\_\_add\_city\_to\_table(message.text, datetime.now(), result[1])

self.\_\_add\_user\_to\_table(message.from\_user.username, message.text, datetime.now())

self.\_\_keyboard\_start(message)

@self.\_\_bot.callback\_query\_handler(func=lambda call: True)

def callback\_worker(call):

if call.data == "weather":

self.\_\_bot.send\_message(call.message.chat.id, 'Введите название города.')

self.\_\_bot.register\_next\_step\_handler(call.message, weather)

def \_\_keyboard\_start(self, message):

keyboard = types.InlineKeyboardMarkup()

key\_weather = types.InlineKeyboardButton(text='Погода', callback\_data='weather')

keyboard.add(key\_weather)

text = 'Нажми на кнопку'

self.\_\_bot.send\_message(message.from\_user.id, text=text, reply\_markup=keyboard)

def \_\_add\_user\_to\_table(self, nickname, city, time):

self.\_\_table.add\_user(nickname, city, time)

def \_\_add\_city\_to\_table(self, city, time, weather):

self.\_\_table.add\_city(city, time, weather)

def start(self):

if self.\_\_token is None:

print("Нет токена")

self.\_\_bot.polling(none\_stop=True, interval=0)

weather.py:

import requests

import json

# icons = ["☀", "🌤", "⛅", "🌥", "☁", "🌧", "🌦", "🌩", "❄", "🌫"]

icons = {"01d": "☀", "01n": "☀",

"02d": "🌤", "02n": "🌤",

"03d": "☁", "03n": "☁",

"04d": "☁", "04n": "☁",

"09d": "🌧", "09n": "🌧",

"10d": "🌦", "10n": "🌦",

"11d": "🌩", "11n": "🌩",

"13d": "❄", "13n": "❄",

"50d": "🌫", "50n": "🌫"}

class WeatherChecker:

def \_\_init\_\_(self, path="weather\_api\_key"):

self.\_\_api\_key = None

self.\_\_page\_request = None

self.\_\_page\_json = None

try:

self.\_\_api\_key = open(path, "r").read()

except FileNotFoundError:

print("Неверный путь")

def \_\_request(self, city):

self.\_\_page\_request = requests.get(

f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={self.\_\_api\_key}")

if self.\_\_page\_request.status\_code != 200:

return False

self.\_\_page\_json = json.loads(self.\_\_page\_request.text)

return True

def \_\_to\_string(self, city):

return str(f"{city}: {icons[self.\_\_page\_json['weather'][0]['icon']]}\n"

f"Температура: {int(self.\_\_page\_json['main']['temp'] - 273.15)} °C,\n"

f"ощущается как: {int(self.\_\_page\_json['main']['feels\_like'] - 273.15)} °C.\n"

f"Влажность: {self.\_\_page\_json['main']['humidity']}%.\n"

f"Скорость ветра: {self.\_\_page\_json['wind']['speed']} км/ч.\n"

)

def check(self, city):

if self.\_\_api\_key is None:

return "Нет ключа api!!!"

if self.\_\_request(city) is False:

return "Ошибка запроса!!!"

return self.\_\_to\_string(city)

tablesManager.py

import psycopg2

class TablesManager:

def \_\_init\_\_(self):

self.\_\_connection = None

self.\_\_host = "127.0.0.1"

self.\_\_user = "postgres"

self.\_\_password = "123456789"

self.\_\_db\_name = "postgres"

def add\_user(self, nickname, city):

self.\_\_connect()

if not self.\_\_connection.closed:

try:

with self.\_\_connection.cursor() as cursor:

cursor.execute("""SELECT id FROM cities WHERE (city='%s')""" % (city))

id = 0

for i in cursor:

id = i[0]

cursor.execute("""INSERT INTO users (nickname, city\_id) VALUES ('%s', %d)""" %

(nickname, id))

self.\_\_connection.commit()

except Exception as ex:

print("While adding user exception: ", ex)

self.\_\_disconnect()

def add\_city(self, city, time, weather):

self.\_\_connect()

if not self.\_\_connection.closed:

try:

with self.\_\_connection.cursor() as cursor:

cursor.execute(

"""INSERT INTO cities (city, time\_searching, weather) VALUES

('%s', '%s', '%s')""" %

(city, time, weather)

)

self.\_\_connection.commit()

except Exception as ex:

print("While adding city exception: ", ex)

self.\_\_disconnect()

def \_\_connect(self):

try:

self.\_\_connection = psycopg2.connect(

host=self.\_\_host,

user=self.\_\_user,

password=self.\_\_password,

database=self.\_\_db\_name

)

except Exception as ex:

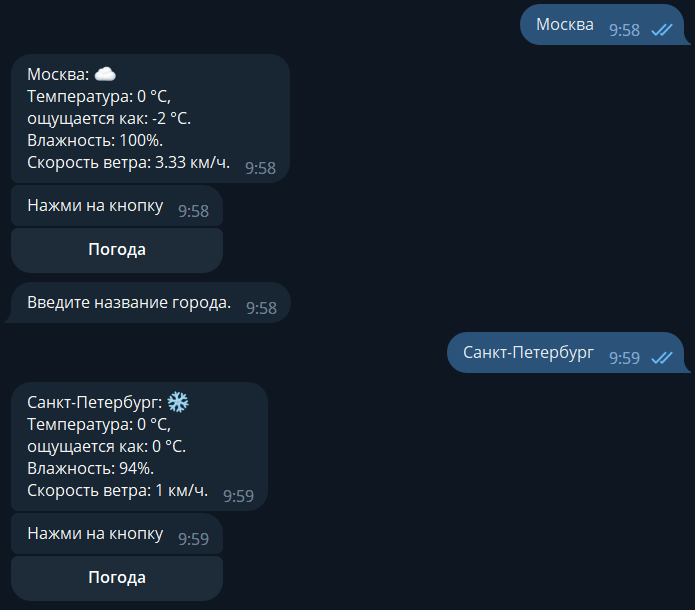
self.\_\_connection.close()

print("Table connection failed. Exception: ", ex)

def \_\_disconnect(self):

self.\_\_connection.close()

# Пример работы



# Выводы

В ходе данной работы был получен навык интеграции базы данных в программу.