Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСИС»

Институт компьютерных наук (ИКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №6
по дисциплине «Разработка сетевых приложений
на языке программирования Python»
на тему «Введение в фреймворк Flask»

Выполнил:
студент группы БИСТ-22-1

Медведев Д. Р.

Проверил:
доц. каф. ИКТ

Стучилин В.В.

Москва, 2025

Цель работы: приобретения навыков в работе с фреймворком Flask.

Задание: разработать клиент-серверное приложение по требованиям, приведенным ниже.

Клиентское ПО:

1. Обмен информацией с сервером должен осуществляться при помощи метода GET протокола http.
2. Клиентское ПО должно давать возможность отправлять на сервер тип функции и коэффициенты для генерации данных: y=a*sin(bx)+c, y=a*x^2+b*x+c, y=(a/(bx)) + c.
3. Клиентское ПО должно давать возможность считывать сгенерированные данные из серверной БД и выводить их на экран в текстовом виде, и в виде графика (библиотека matplotlib).

Серверное ПО:

1. Серверное ПО должно быть написано на с использованием фреймворка Flask (Python).
2. Серверное ПО должно получать данные от клиента и осуществлять генерацию данных на основе полученного типа функции и коэффициентов.
3. Датасет должен состоять не менее чем из 1000 значений X и Y.
4. Датасет должен записываться в БД (номер записи, X, Y). Рекомендуемая СУБД – MySQL.
5. Серверное ПО должно обеспечивать клиентскому ПО возможность получения датасета.

Решение

Серверное ПО:

App.py

```python
from flask import Flask, request, jsonify
import mysql.connector
import math
import random

app = Flask(__name__)

db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'function_data'
}

def generate_data(func_type, a, b, c, num_points=1000):
    data = []
    for i in range(num_points):
        x = random.uniform(-10, 10)
        if func_type == 'sin':
            y = a * math.sin(b * x) + c
        elif func_type == 'quadratic':
            y = a * x**2 + b * x + c
        elif func_type == 'rational':
            if b * x == 0:
                y = float('inf')
            else:
                y = (a / (b * x)) + c
        else:
            raise ValueError("Неизвестный тип функции")
        data.append((x, y))
    return data

def save_to_db(func_type, a, b, c, data):
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()

        cursor.execute("""
        CREATE TABLE IF NOT EXISTS function_results (
            id INT AUTO_INCREMENT PRIMARY KEY,
            func_type VARCHAR(20),
            a FLOAT,
            b FLOAT,
            c FLOAT,
            x FLOAT,
            y FLOAT
        )
        """)

        for x, y in data:
            cursor.execute("""
            INSERT INTO function_results (func_type, a, b, c, x, y)
            VALUES (%s, %s, %s, %s, %s, %s)
            """, (func_type, a, b, c, x, y))

        conn.commit()
        return True
    except Exception as e:
        print(f"Ошибка при сохранении в БД: {e}")
        return False
    finally:
        if conn.is_connected():
            cursor.close()
            conn.close()

@app.route('/generate', methods=['GET'])
def generate():
    func_type = request.args.get('func_type')
    a = float(request.args.get('a'))
```

```
    b = float(request.args.get('b'))
    c = float(request.args.get('c'))

    data = generate_data(func_type, a, b, c)
    success = save_to_db(func_type, a, b, c, data)

    return jsonify({'success': success, 'count': len(data)})

@app.route('/get_data', methods=['GET'])
def get_data():
    try:
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor(dictionary=True)

        cursor.execute("SELECT x, y FROM function_results ORDER BY id DESC LIMIT 1000")
        data = cursor.fetchall()

        return jsonify(data)
    except Exception as e:
        return jsonify({'error': str(e)})
    finally:
        if conn.is_connected():
            cursor.close()
            conn.close()

if __name__ == '__main__':
    app.run(debug=True)
```

Клиентское ПО:

Client.py

```python
import tkinter as tk
from tkinter import ttk, messagebox, font
import requests
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

BASE_URL = "http://localhost:5000"

class FunctionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Генератор функций")
        self.root.geometry("1000x800")

        self.big_font = font.nametofont("TkDefaultFont").copy()
        self.big_font.configure(size=12)

        self.bold_font = font.nametofont("TkDefaultFont").copy()
        self.bold_font.configure(size=12, weight="bold")

        self.button_font = font.nametofont("TkDefaultFont").copy()
        self.button_font.configure(size=14)

        self.create_widgets()

    def create_widgets(self):
        param_frame = ttk.LabelFrame(self.root, text="Параметры функции", padding=10)
        param_frame.pack(pady=15, padx=15, fill="x")

        ttk.Label(param_frame, text="Тип функции:", font=self.bold_font).grid(row=0, column=0, padx=10, pady=10, sticky="w")
        self.func_type = tk.StringVar()
        func_combobox = ttk.Combobox(
            param_frame,
            textvariable=self.func_type,
            values=["sin", "quadratic", "rational"],
            font=self.big_font,
            height=25
        )
        func_combobox.grid(row=0, column=1, padx=10, pady=10, sticky="ew")
        func_combobox.current(0)

        ttk.Label(param_frame, text="a:", font=self.bold_font).grid(row=1, column=0, padx=10, pady=10, sticky="w")
        self.a_var = tk.DoubleVar(value=1.0)
        ttk.Entry(param_frame, textvariable=self.a_var, font=self.big_font).grid(row=1, column=1, padx=10, pady=10, sticky="ew")

        ttk.Label(param_frame, text="b:", font=self.bold_font).grid(row=2, column=0, padx=10, pady=10, sticky="w")
        self.b_var = tk.DoubleVar(value=1.0)
        ttk.Entry(param_frame, textvariable=self.b_var, font=self.big_font).grid(row=2, column=1, padx=10, pady=10, sticky="ew")

        ttk.Label(param_frame, text="c:", font=self.bold_font).grid(row=3, column=0, padx=10, pady=10, sticky="w")
        self.c_var = tk.DoubleVar(value=0.0)
        ttk.Entry(param_frame, textvariable=self.c_var, font=self.big_font).grid(row=3, column=1, padx=10, pady=10, sticky="ew")

        button_frame = ttk.Frame(self.root)
        button_frame.pack(pady=20)

        style = ttk.Style()
        style.configure("Big.TButton", font=self.button_font, padding=10)

        ttk.Button(
            button_frame,
            text="Сгенерировать данные",
            command=self.generate_data,
            style="Big.TButton"
        ).pack(side="left", padx=15)

        ttk.Button(
```

```
            button_frame,
            text="Получить данные",
            command=self.fetch_data,
            style="Big.TButton"
        ).pack(side="left", padx=15)

        self.text_output = tk.Text(
            self.root,
            height=12,
            font=self.big_font,
            wrap=tk.WORD
        )
        self.text_output.pack(pady=15, padx=15, fill="both", expand=True)

        scrollbar = ttk.Scrollbar(self.text_output)
        scrollbar.pack(side="right", fill="y")
        self.text_output.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.text_output.yview)

        self.figure, self.ax = plt.subplots(figsize=(8, 5))
        self.canvas = FigureCanvasTkAgg(self.figure, master=self.root)
        self.canvas.get_tk_widget().pack(pady=15, padx=15, fill="both", expand=True)

    def generate_data(self):
        try:
            params = {
                'func_type': self.func_type.get(),
                'a': self.a_var.get(),
                'b': self.b_var.get(),
                'c': self.c_var.get()
            }
```

```
            response = requests.get(f"{BASE_URL}/generate", params=params)
            result = response.json()

            if result.get('success'):
                messagebox.showinfo("Успех", f"Сгенерировано {result['count']} точек данных")
            else:
                messagebox.showerror("Ошибка", "Не удалось сгенерировать данные")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка: {str(e)}")

    def fetch_data(self):
        try:
            response = requests.get(f"{BASE_URL}/get_data")
            data = response.json()

            if isinstance(data, list):
                self.text_output.delete(1.0, tk.END)
                for point in data[:50]:
                    self.text_output.insert(tk.END, f"x: {point['x']:.2f}, y: {point['y']:.2f}\n")

                self.ax.clear()
                x_values = [point['x'] for point in data]
                y_values = [point['y'] for point in data]
                self.ax.scatter(x_values, y_values, s=5)
                self.ax.set_xlabel('X', fontsize=12)
                self.ax.set_ylabel('Y', fontsize=12)
                self.ax.set_title(f"График функции {self.func_type.get()}", fontsize=14)
                self.canvas.draw()
            else:
                messagebox.showerror("Ошибка", "Неверный формат данных")
        except Exception as e:
```

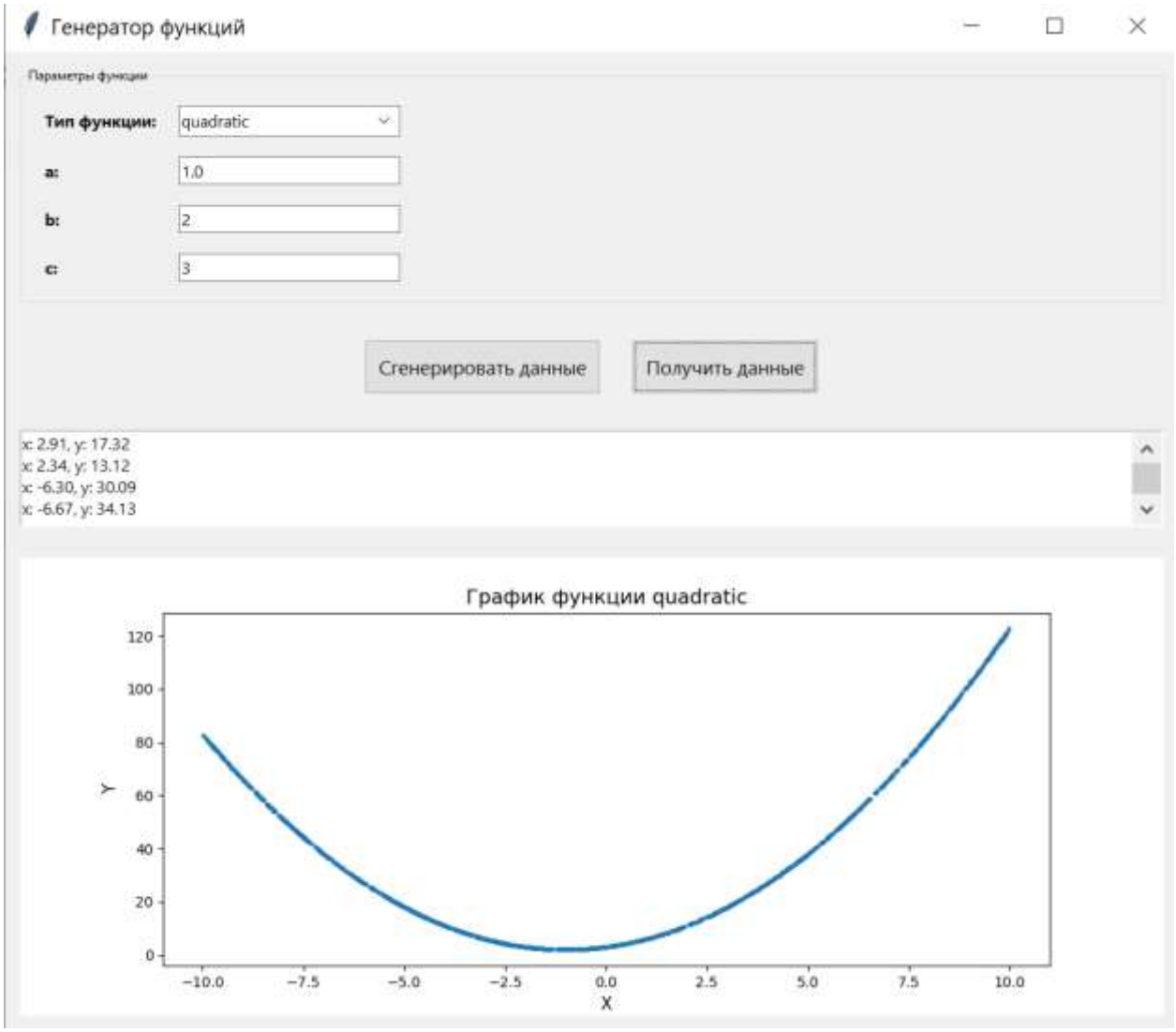
```
            messagebox.showerror("Ошибка", f"Произошла ошибка: {str(e)}")

if __name__ == "__main__":
    root = tk.Tk()
    app = FunctionApp(root)
    root.mainloop()
```

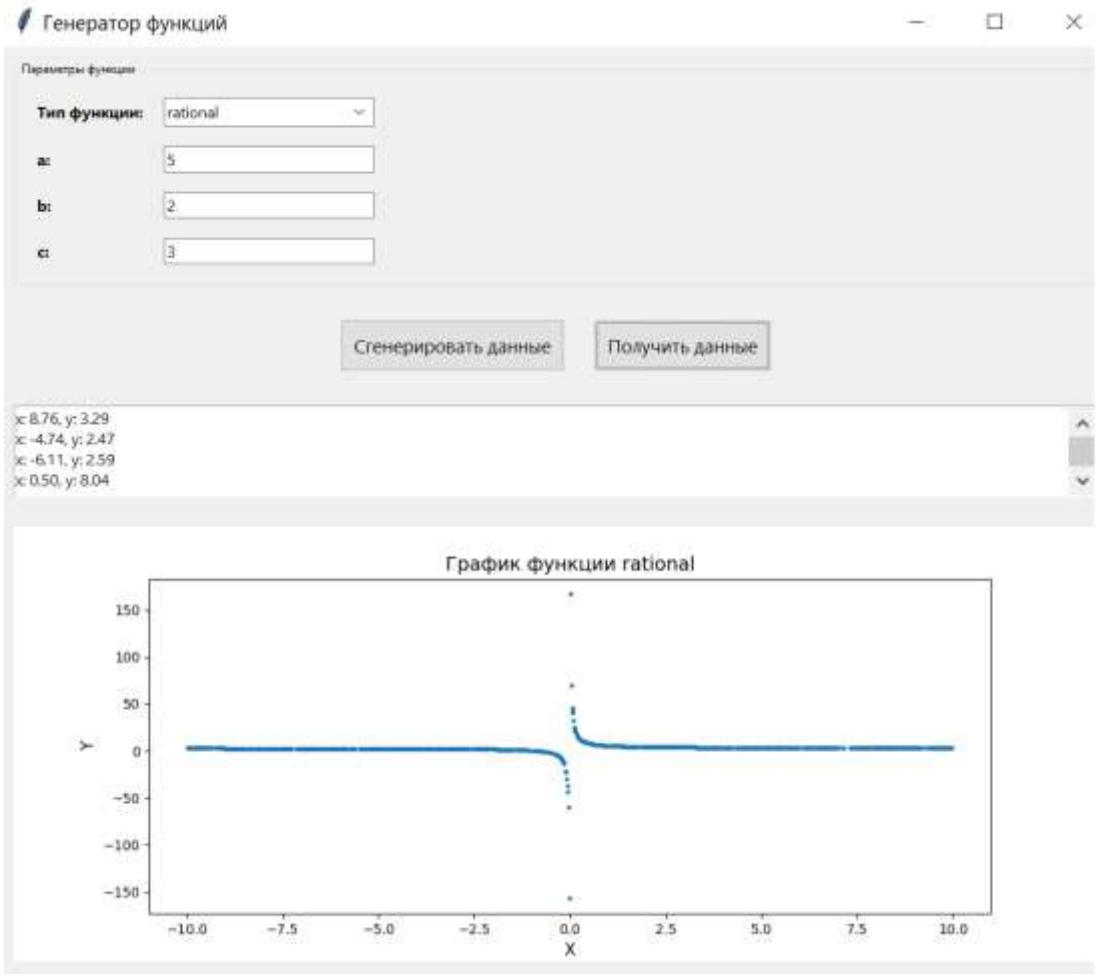
База данных

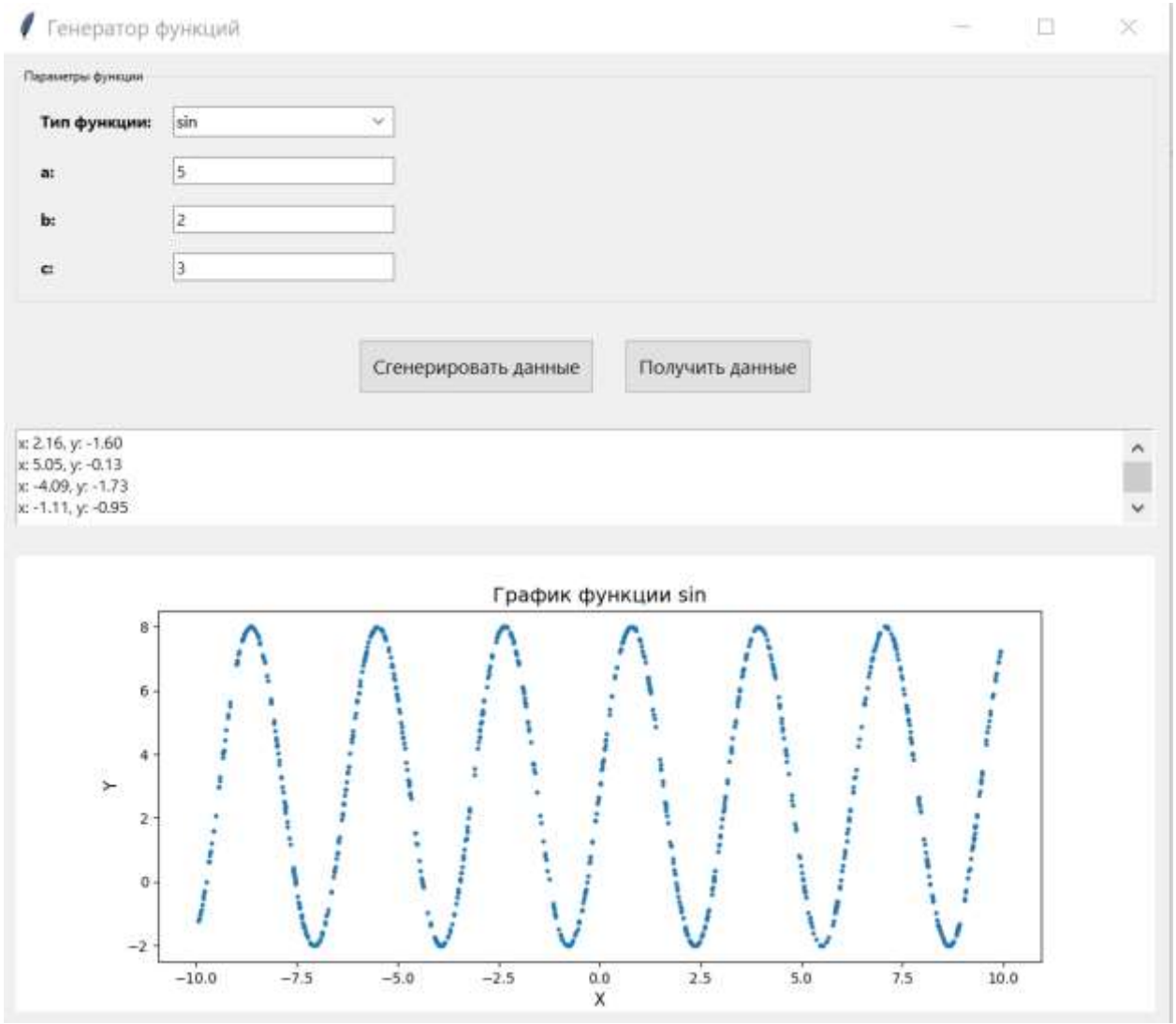
		id	func_type	a	b	c	x	y
☐	Изменить Копировать Удалить	1	sin	1	1	0	-8.75849	-0.61807
☐	Изменить Копировать Удалить	2	sin	1	1	0	5.99184	-0.287239
☐	Изменить Копировать Удалить	3	sin	1	1	0	8.23496	0.928303
☐	Изменить Копировать Удалить	4	sin	1	1	0	5.79009	-0.473356
☐	Изменить Копировать Удалить	5	sin	1	1	0	1.30454	0.964762
☐	Изменить Копировать Удалить	6	sin	1	1	0	-0.143886	-0.14339
☐	Изменить Копировать Удалить	7	sin	1	1	0	-5.25422	0.856767
☐	Изменить Копировать Удалить	8	sin	1	1	0	-3.26545	0.123537
☐	Изменить Копировать Удалить	9	sin	1	1	0	6.07051	-0.211081
☐	Изменить Копировать Удалить	10	sin	1	1	0	-0.0436369	-0.043623
☐	Изменить Копировать Удалить	11	sin	1	1	0	-8.8596	-0.535569
☐	Изменить Копировать Удалить	12	sin	1	1	0	5.22981	-0.8691
☐	Изменить Копировать Удалить	13	sin	1	1	0	2.93606	0.204084
☐	Изменить Копировать Удалить	14	sin	1	1	0	-0.110429	-0.110205
☐	Изменить Копировать Удалить	15	sin	1	1	0	0.2735	0.270103
☐	Изменить Копировать Удалить	16	sin	1	1	0	-7.25307	-0.824822
☐	Изменить Копировать Удалить	17	sin	1	1	0	-0.882986	-0.772638
☐	Изменить Копировать Удалить	18	sin	1	1	0	6.3646	0.0813216
☐	Изменить Копировать Удалить	19	sin	1	1	0	5.51108	-0.697643
☐	Изменить Копировать Удалить	20	sin	1	1	0	-1.72257	-0.988505
☐	Изменить Копировать Удалить	21	sin	1	1	0	6.89558	0.574831

Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
function_results	Обзор Структура Поиск Вставить Очистить Удалить	3 000	InnoDB	utf8mb4_general_ci	192,0 Кб	-
1 таблица	Всего	3 000	InnoDB	utf8mb4_general_ci	192,0 Кб	0 Байт

Результат:


Генератор функций
Параметры функции
Тип функции:
quadratic
a:
1.0
b:
2
c:
3
Сгенерировать данные
Получить данные
x: 2.91, y: 17.32
x: 2.34, y: 13.12
x: -6.30, y: 30.09
x: -6.67, y: 34.13
График функции quadratic
120
100
80
60
40
20
0
Y
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X


Генератор функций
Параметры функции
Тип функции:
rational
a:
5
b:
2
c:
3
Сгенерировать данные
Получить данные
x: 8.76, y: 3.29
x: -4.74, y: 2.47
x: -6.11, y: 2.59
x: 0.50, y: 8.04
График функции rational
150
100
50
0
−50
−100
−150
Y
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X



Вывод:

Я приобрел навыки в работе с фреймворком Flask.