Санкт-Петербургский Национальный
Исследовательский Университет Информационных
технологий, механики и оптики

Лабораторная работа 1

Реализация программной модели инфокоммуникационной системы

Выполнил: Шишминцев
Дмитрий Владимирович
Группа № К3121
Проверила: Казанова
Полина Петровна

Санкт-Петербург
2022

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ

В данной лабораторной работе была реализована система обработки данных «Программа для контроля собственных денежных средств», а так же проанализирована предметная область и требования к работе.

1 Анализ

1.1 Анализ предметной области

На рынке представлено множество программного обеспечения со схожим функционалом. Их можно разделить на несколько типов: мобильные приложения банков, приложения для ручного ввода трат, приложения которые синхронизируются с банковскими приложениями. Все представленное на рынке программное обеспечение имеет следующий общий функционал: ручное/автоматическое добавление транзакций, удаление, отображение транзакций по дате, отображение транзакций по категориям, сортировка по другим параметрам. Учитывая рост спроса на потребительские товары и услуги по всему миру, область перспективной, а разработку данной программной модели - актуальной.

1.2 Анализ требований

Проанализировав предметную область, а так же задание полученное на практике по дисциплине «Программирование», были выдвинуты следующие технические требования к программной модели:

1. Разработать алгоритм работы системы
2. Спроектировать базу данных SQLite3
3. Разработать веб-сервер для статических файлов с помощью языка программирования Python и библиотеки Flask
4. Разработать веб-сервер для обработки POST и GET запросов.
5. Разработать веб-интерфейс для программной модели.

2 Ход работы

2.1 Разработка алгоритма работы системы

Для разрабатываемой модели программной системы был разработан и представлен в виде блок-схемы следующий алгоритм. (Рисунок 1).

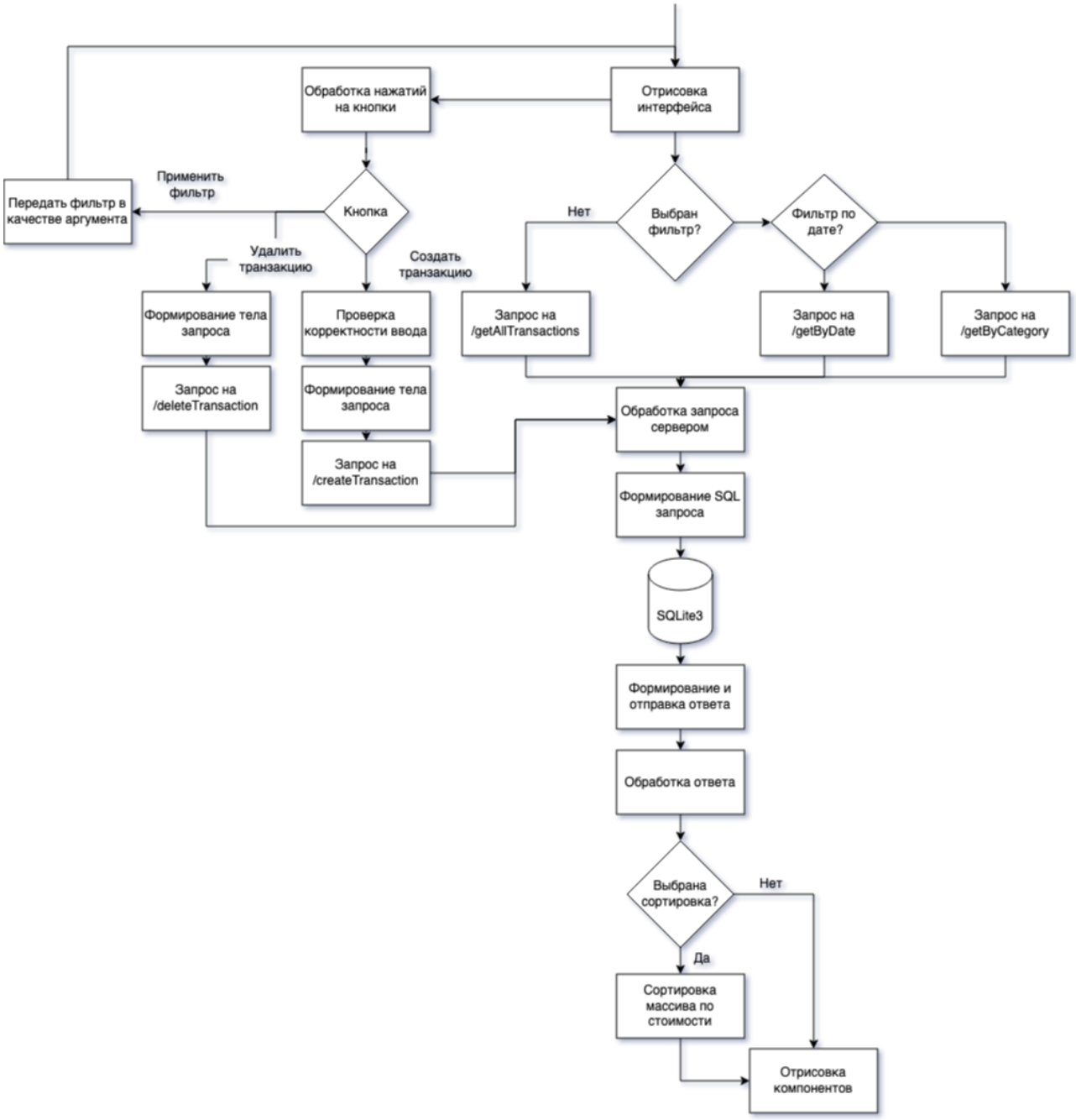


Рисунок 1 — Блок-схема алгоритма

2.2 Проектирование базы данных SQLite3

Для разработки модели программной системы было решено использовать систему управления базами данных SQLite3. Данная СУБД имеет такие преимущества, как: высокая скорость, кроссплатформенность, надежность, а так же не нуждается в конфигурации.

Для хранения транзакций была спроектирована следующая модель базы данных. Модель включает в себя одну таблицу «transactions», записи в которой имеют такие свойства, как:

1. tr-id - id транзакции, так же является primary key для таблицы (integer)
2. name - название транзакции (string)
3. cost - стоимость транзакции (integer)
4. date - дата транзакции. Для хранения дат было приято решение хранить их в формате unix-time. (integer)
5. category - категория транзакции (string)

Ниже демонстрируется схема базы данных (Рисунок 2).

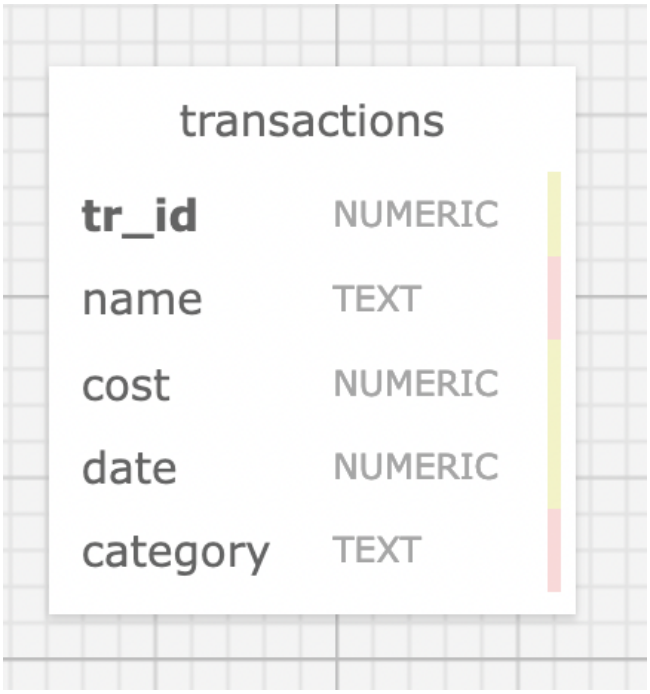


Рисунок 2 — Схема базы данных

2.3 Разраотка сервера для статических файлов

Для реализации модели программной системы используется высокоуровневый язык программирования Python. Для ускорения процесса разработки было принято использовать фреймворк для создания веб-приложений Flask.

Для корректной работы программной модели был разработан сервер статических файлов с помощью библиотеки Flask. Сервер работает на порту 5000, рабочей директорией для него является «./www/». Сервер устанавливает MIME-типы для файлов формата html, css, js. Ниже представлен фрагмет кода на языке программирования Python. (Рисунок 3).

```
11
12  def getFile(filename):
13      try:
14          src = os.path.join('./www', filename)
15          return open(src).read()
16      except IOError as exc:
17          return str(exc)
18
19
20  @app.route("/")
21  def responseRoot():
22      return Response(getFile('index.html'))
23
24  @app.route('/<path:path>')
25  def serveStatic(path):
26      ext = path.split('.')[-1]
27      print(ext)
28      if ext == 'html':
29          type='text/html'
30      elif ext == 'css':
31          type='text/css'
32      elif ext == 'js':
33          type='text/javascript'
34      else:
35          type=''
36      content = getFile(path)
37      return Response(content, mimetype=type)
38
```

Рисунок 3 — Фрагмент кода сервера статических файлов

В качестве интерфейса для программной модели было принято решение использовать веб интерфейс. Веб интерфейс расположен в директории «./www/» и имеет главную страницу на языке разметки гипертекста, каскадную таблицу стилей, а так же логика работы описана на языке программирования высокого уровня JavaScript. В результате веб-интерфейс будет доступен по адресу http://localhost:5000.

2.4 Разработка сервера для обработки POST и GET запросов

Аналогично, для реализации веб-сервера для обработки POST и GET запросов был использован язык программирования высокого уровня Python, а так же библиотека для разработки веб-приложений Flask.

Для реализации обработки GET запросов был написан следующий код. (Рисунок 4).

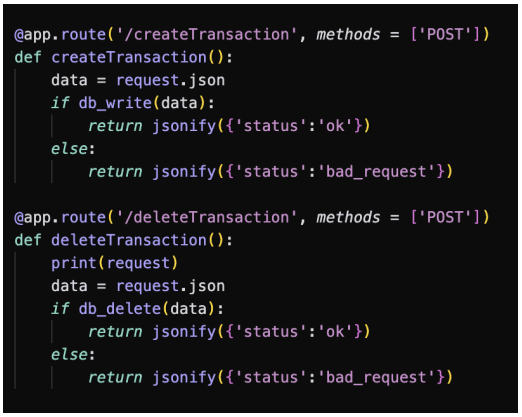
```python
@app.route('/getAllTransactions', methods = ['GET'])
def getAllTransations():
    r = db_readAll()
    # print(r)
    if r:
        return jsonify(r)
    else:
        return jsonify({'status':'server_error'})

@app.route('/getCategories', methods = ['GET'])
def getCategories():
    r = db_getCategories()
    if r:
        return jsonify(r)
    else:
        return jsonify({'status':'bad_request'})

@app.route('/getByCategory' , methods = ['GET'])
def getByCategory():
    args = request.args
    if 'category' in args.keys():
        r = db_getByCategory(args['category'])
        return jsonify(r)
    else:
        return jsonify({'status':'bad_request'})


@app.route('/getByDate' , methods = ['GET'])
def getByDate():
    args = request.args
    if 'startDate' in args.keys() and 'endDate' in args.keys():
        r = db_getByDate(args['startDate'], args['endDate'])
        print(r)
        return jsonify(r)
    else:
        return jsonify({'status':'bad_request'})
```

Рисунок 4 — Фрагмент кода для обработки GET запросов

В представленном фрагменте кода обрабатываются GET запросы по адресам «/getAllTransactions» (возвращает все транзакции), «/getCategories» (возвращает категории), «/getByCategory» (возвращает транзакции по определенной категории) и «/getByDate» (возвращает транзакции за заданный временной период). Получая запрос, осуществляется запрос к базе данных SQLite3. Возможность обращения к базе данных реализована в отдельном файле db.py. (Рисунок 6). Получая ответ от базы данных, обрабатываются

ошибки и возвращается ответ на запрос в формате JSON.

Так же, для реализации обработки POST запросов, был реализован следующий код. (Рисунок 5)

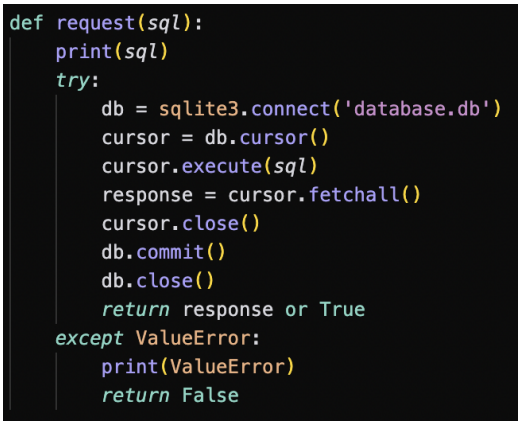


```python
@app.route('/createTransaction', methods = ['POST'])
def createTransaction():
    data = request.json
    if db_write(data):
        return jsonify({'status':'ok'})
    else:
        return jsonify({'status':'bad_request'})

@app.route('/deleteTransaction', methods = ['POST'])
def deleteTransaction():
    print(request)
    data = request.json
    if db_delete(data):
        return jsonify({'status':'ok'})
    else:
        return jsonify({'status':'bad_request'})
```

Рисунок 5 — Фрагмент кода для обработки POST запросов

В представленном фрагменте кода представлена реализация обработки POST запросов. Запросы обрабатываются по адресам «/createTransaction» и «/deleteTransaction». В теле запроса передается информация о транзакции. Корректность тела запроса проверяется, далее идет запрос к базе данных SQLite3. В результате возвращается ответ в формате JSON.

Для реалзизации возможности работы с базой данных SQLite3 была написана следующая функция для работы с базой данных (Рисунок 6). Для реализации функции была использована библиотека sqlite3.



```python
def request(sql):
    print(sql)
    try:
        db = sqlite3.connect('database.db')
        cursor = db.cursor()
        cursor.execute(sql)
        response = cursor.fetchall()
        cursor.close()
        db.commit()
        db.close()
        return response or True
    except ValueError:
        print(ValueError)
        return False
```

Рисунок 6 — Фрагмент кода для работы с базой данных

В приведенном фрагменте функция получает на вход строку - SQL запрос, и выполняет его для базы данных databse.db. В случае ошибки функция возвращает False, в случае успешного выполнения запроса функция возвращает True или результат выполнения запроса (если он присутствует)

Для обработки каждого GET и POST запроса был написан соответствующий SQL запрос, соответствующие функции реализованы в файле db.py.

2.5 Разработка веб-интерфейса для программной модели

Для реализации взаимодействия пользователя с программой был выбран веб-интерфейс. Веб-интерфейс реализован с помощью языка разметки гипертекста HTML, каскадных таблиц стилей CSS, а так же языка программирования высокого уровня JavaScript. Веб-интерфейс отображается при GET запросе к серверу статических файлов по адресу http://localhost:5000. Код на языке JavaScript осуществляет работу логики приложения и осуществляет POST и GET запросы к веб-серверу написанному с помощью языка Python.

В представленном ниже фрагменте кода (Рисунок 7) демонстрируется функция отправка HTTP запроса написанная на языке JavaScript.

```
function request(r, callback) {
    if (r.body) {
        r.body = JSON.stringify(r.body)
    }
    fetch(r.url, {
        headers: {
            'Content-Type': 'application/json'
        },
        method: r.method,
        body: r.body
    }).then(function(res){
        return res.json();
    }).then(function(data){
        callback(data);
    })
}
```

Рисунок 7 — Фрагмент кода для отправки HTTP запроса

Функция принимает в качестве аргумента объект r с параметрами, а так же функцию callback.

Итоговый внешний вид программной модели выглядит следующим образом: (Рисунок 8)

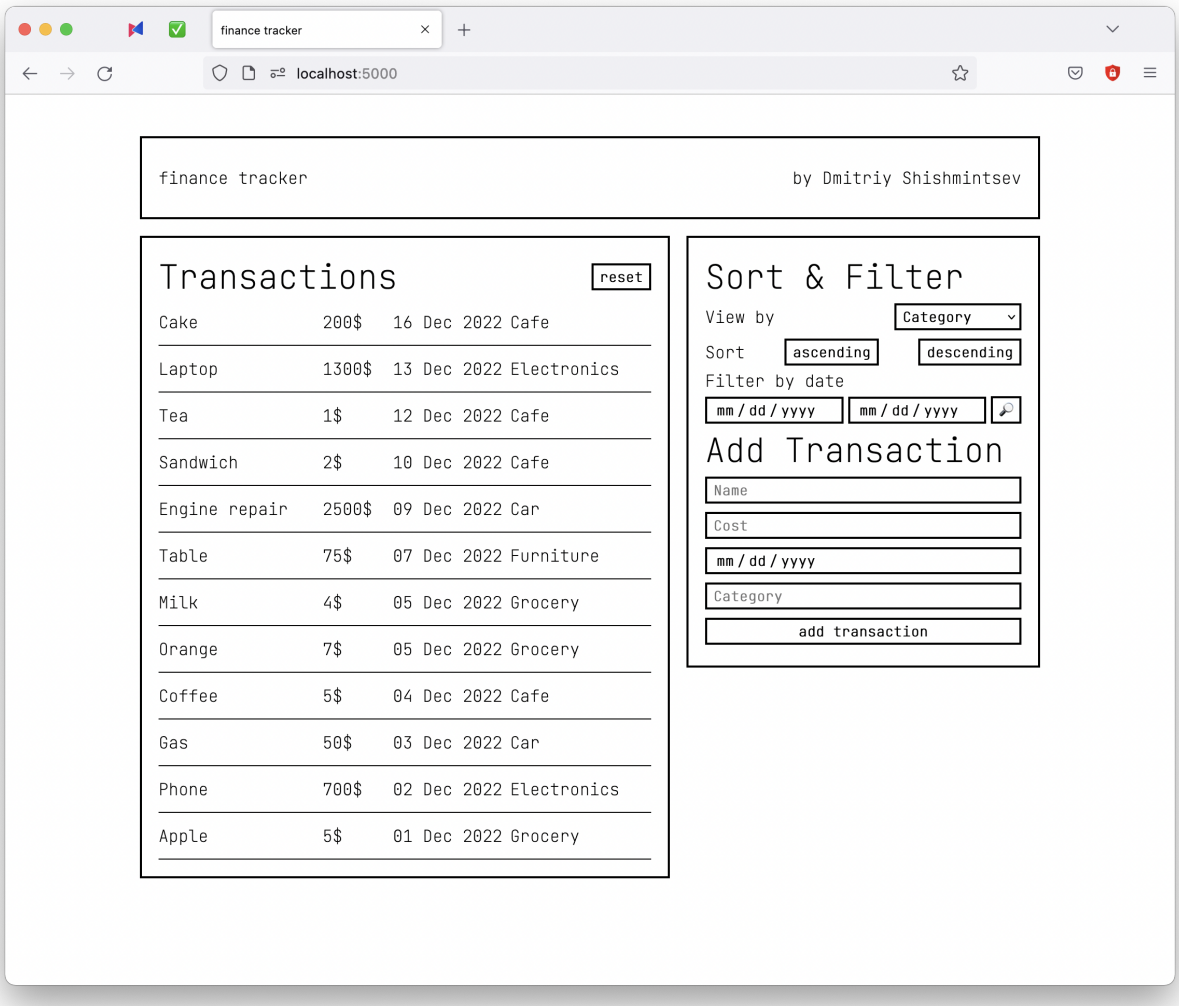


Рисунок 8 — Интерфейс программы

ЗАКЛЮЧЕНИЕ

В результате данной работы была реализована программная модель инфокоммуникационной системы для контроля личных финансов. Была проанализирована предметная область. В процессе работы были использованы современные технологии и подходы к разработке программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SQLite3 - Официальный сайт (URL:`https://www.sqlite.org/`)

2. Python Docs - Официальный сайт (URL:`https://docs.python.org/3/`)

3. Flask - Официальный сайт (URL:`https://flask.palletsprojects.com/en/2.2.x/`)