Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Рязанский государственный радиотехнический
университет имени В.Ф. Уткина»
Кафедра ВПМ

Отчёт
о лабораторной работе №3
по дисциплине
«Распределённые системы обработки информации»
Тема:
«Вычисления на основе взаимодействия сервисов»

Выполнила
студентка гр. 343М
Торжкова А.О.
Проверил
доц. Князьков П.А.

Рязань 2023

Цель работы

Получение практических навыков реализация взаимодействия сервисов.

Выполнение

Задание 1. Создать публичный удаленный репозиторий на GitHub. Загрузить в репозиторий полученные ранее файлы тетрадей «xmlrpc_client.ipynb», «xmlrpc_server.ipynb», «xmlrpc_stats_server.ipynb» (возможно сначала только зафиксировать версии файлов в локальном).

Удалённый репозиторий на GitHub (рис. 1):

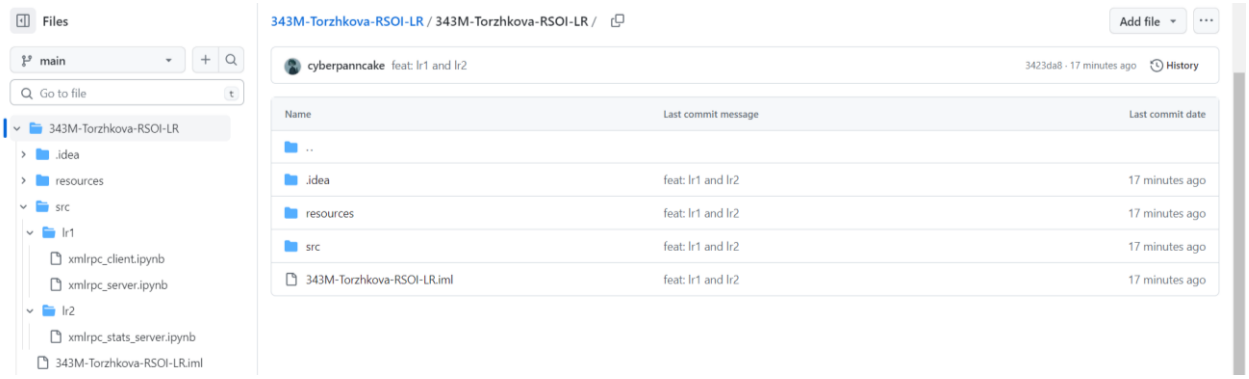

Рисунок 1 – Удалённый репозиторий на GitHub

Задание 2. Создать тетрадь «xmlrpc_proxy_server.ipynb» для прокси сервиса. В этот сервис должны приходить запросы от клиента «xmlrpc_client.ipynb» и передаваться на сервер «xmlrpc_server.ipynb».

Код тетради xmlrpc_proxy_server.ipynb (рис. 2):

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler
import xmlrpc.client as client


class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)
proxy_server = SimpleXMLRPCServer(("localhost", 8010), requestHandler=RequestHandler)
server = client.ServerProxy("http://localhost:8008")


# Приём команд от клиента и передача их на сервер, а затем возврат результата
def execute_function(func_name, *params):
    try:
        if len(params) == 0:
            result = eval('server.' + func_name)()
        elif len(params) == 1:
            result = eval('server.' + func_name)(params[0])
        elif len(params) == 2:
            result = eval('server.' + func_name)(params[0], params[1])
        elif len(params) == 3:
            result = eval('server.' + func_name)(params[0], params[1], params[2])
        return result
    except:
        return 'Server is not available'
proxy_server.register_function(execute_function, 'exec')

print ("Listening on port 8010...")
proxy_server.serve_forever()
```

Рисунок 2 – Прокси-сервер

Задание 3. В прокси сервисе должны регистрироваться время события, тип события сервера (по типу выполняемой операции), время выполнения задания на сервере «xmlrpc_server.ipynb».

Код тетради xmlrpc_proxy_server.ipynb (рис. 3):

```
stats_server = client.ServerProxy("http://localhost:8018")


# Приём команд от клиента и передача их на сервер, а затем возврат результата
def execute_function(func_name, *params):
    try:
        work_time = datetime.datetime.now()
        if len(params) == 0:
            result = eval('server.' + func_name)()
        elif len(params) == 1:
            result = eval('server.' + func_name)(params[0])
        elif len(params) == 2:
            result = eval('server.' + func_name)(params[0], params[1])
        elif len(params) == 3:
            result = eval('server.' + func_name)(params[0], params[1], params[2])
        work_time = datetime.datetime.now() - work_time
        stats_server.add_log(func_name, work_time.total_seconds())
        return result
    except:
        return 'Server is not available'
proxy_server.register_function(execute_function, 'exec')

# Добавление в лог через сервер
def add_log(func_name, work_time):
    try:
        stats_server.add_log(func_name, work_time)
        return True
    finally:
        return False
```

Рисунок 3 – Регистрация события в прокси-сервере

Время события регистрируется в серверестатистики.

Задание 4. Данные зарегистрированных событий из прокси сервиса должны сохраняться через доработанный сервер статистики «xmlrpc_stats2_server.ipynb» в БД SQLite (log.db). При отсутствии работы (выключенном) сервере «xmlrpc_stats2_server.ipynb» работа функций сервера «xmlrpc_server.ipynb» должна выполняться.

Код тетради xmlrpc_stats2_server.ipynb (рис. 4):

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler
import datetime
import os
import sqlite3


class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)
server = SimpleXMLRPCServer(("localhost", 8019),
                            requestHandler=RequestHandler)

# Путь к файлу БД
log_db_path = '../../resources/lr3/log.db'


# Добавление строки в лог
def add_log(sname, work_time):
    conn = sqlite3.connect(log_db_path)
    conn.cursor().execute('INSERT INTO log VALUES (?, ?, ?)', (sname, datetime.datetime.now().strftime('%Y%m%dT%H:%M:%S'), work_time))
    conn.commit()
    conn.close()
    return True
server.register_function(add_log, 'add_log')


def create_db_if_not_exist():
    conn = sqlite3.connect(log_db_path)
    conn.cursor().execute('''CREATE TABLE IF NOT EXISTS log (type TEXT, ev_time TEXT, work_time REAL)''')
    conn.commit()
    conn.close()
```

Рисунок 4 – Логирование в БД на новом сервере статистики

Задание 5. В рамках клиента «xmlrpc_client.ipynb» реализовать получение содержимого журнала событий сервера статистики с возможностью получения среза по типу выполняемой функции, времени вызова и ее длительности.

Код тетради xmlrpc_stats2_server.ipynb (рис. 5):

```
def get_logs_slice(p_type, s_time, e_time, max_work_time):
    conn = sqlite3.connect(log_db_path)
    query, params = get_query(p_type, s_time, e_time, max_work_time)
    cursor = conn.execute(query, params)
    logs = []
    for row in cursor:
        logs.append(row)
    conn.close()
    return logs
server.register_function(get_logs_slice, 'get_logs_slice')


def get_query(p_type, s_time, e_time, max_work_time):
    wheres = []
    params = {}
    if p_type != 'all':
        wheres.append('log.type = :type')
        params['type'] = p_type
    if s_time != '':
        wheres.append('log.ev_time >= :s_time')
        params['s_time'] = s_time
    if e_time != '':
        wheres.append('log.ev_time <= :e_time')
        params['s_time'] = s_time
    if max_work_time != '':
        wheres.append('log.work_time <= :max_work_time')
        params['max_work_time'] = max_work_time
    query = 'SELECT * FROM log' + (' WHERE ' + ' AND '.join(wheres) if len(wheres) > 0 else '')
    return query, params
```

Рисунок 5 – Получение среза лога на новом сервере статистики

Код тетради и результаты выполнения (рис. 6-7):

```
# Лабораторная 3. Задание 5. Получение содержимого журнала событий сервера статистики с возможностью получения среза
# по типу выполняемой функции, времени вызова и ее длительности

import datetime

all_logs = server_proxy.get_logs_slice('all', '', '', '', 2)
sliced_logs = server_proxy.get_logs_slice('type', '2023-09-26 19:18:33', '', 10, 2)
print('All logs:\n' + (all_logs if isinstance(all_logs, str) else '\n'.join(map(str, all_logs))))
print('\nSliced logs:\n' + (sliced_logs if isinstance(sliced_logs, str) else '\n'.join(map(str, sliced_logs))))
```

```
All logs:
['ping', '20231010T19:18:58', 2.045328]
['ping', '20231010T19:21:51', 2.03219]
['type', '20231010T19:27:45', 2.032124]
['type', '20231010T19:27:54', 2.042791]
['ping', '20231010T19:28:02', 2.023743]
['now', '20231010T19:28:10', 2.014645]
['type', '20231010T19:28:18', 2.040612]
['type', '20231010T19:28:27', 2.063181]
['type', '20231010T19:28:35', 2.04586]
['type', '20231010T19:28:43', 2.03642]
['type', '20231010T19:28:51', 2.046238]
['type', '20231010T19:29:00', 2.040126]
['type', '20231010T19:29:08', 2.043436]
['sum', '20231010T19:29:16', 2.030601]
['pow', '20231010T19:29:25', 2.040622]
['send_back_binary', '20231010T19:29:34', 2.264879]
['color_inversion', '20231010T19:29:49', 7.8239]
['black_list_check', '20231010T19:29:58', 2.033844]
['black_list_check', '20231010T19:30:06', 2.018803]
['black_list_check_by_name_and_birthday', '20231010T19:30:14', 2.031115]
['black_list_check_by_name_and_birthday', '20231010T19:30:23', 2.037029]
['color_inversion', '20231010T19:30:37', 7.929933]
['color_inversion', '20231010T19:30:47', 3.41989]
['binarize_by_threshold', '20231010T19:31:01', 7.292312]
['binarize_by_threshold', '20231010T19:31:15', 7.298012]
['binarize_by_threshold', '20231010T19:31:25', 3.298242]
['mirror_vertically', '20231010T19:31:35', 3.748187]
['mirror_vertically', '20231010T19:31:44', 2.416654]
```

Рисунок 6 – Результат получения среза (начало)

```
Sliced logs:
['type', '20231010T19:27:45', 2.032124]
['type', '20231010T19:27:54', 2.042791]
['type', '20231010T19:28:18', 2.040612]
['type', '20231010T19:28:27', 2.063181]
['type', '20231010T19:28:35', 2.04586]
['type', '20231010T19:28:43', 2.03642]
['type', '20231010T19:28:51', 2.046238]
['type', '20231010T19:29:00', 2.040126]
['type', '20231010T19:29:08', 2.043436]
```

Рисунок 7 – Результат получения среза (окончание)

Задание 6. Обновить содержимое удаленного репозитория GitHub по результатам выполнения пунктов 2-5.

Обновлённый репозиторий GitHub (рис. 8):

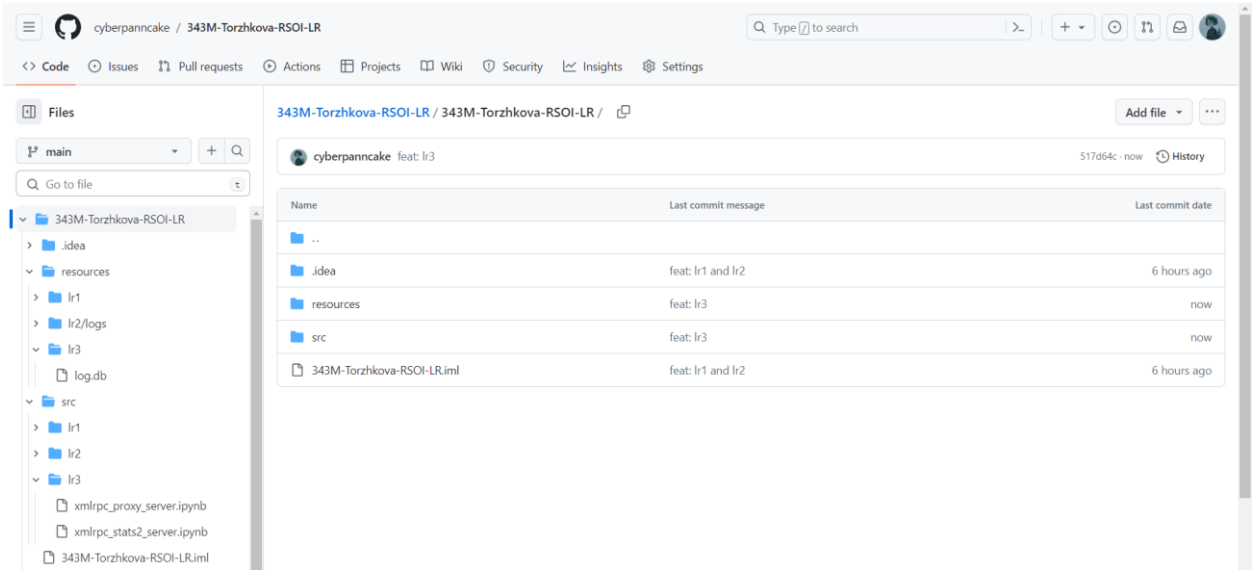

Рисунок 8 – Обновлённый репозиторий GitHub

Задание 7. Оформить отчет по результатам выполнения лабораторной работы. В отчете дополнительно должна быть ссылка на репозиторий в формате https://github.com/%ИМЯ_АККАУНТА%/%ИМЯ_ПРОЕКТА%. В репозитории должно быть минимум две фиксации файлов (пункты 1, 6).

Ссылка на репозиторий GitHub:
https://github.com/cyberpanncake/343M-Torzhkova-RSOI-LR/

Описание полученных навыков

В ходе лабораторной работы были получены практические навыки создания и настройки взаимодействия двух сервисов.

Вывод

Взаимодействие сервисов позволяет разделить различные по типу задачи между разными сервисами таким образом, чтобы при сбое работы одного сервиса другом мог корректно продолжать свою работу