Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

Рязанский государственный радиотехнический университет

имени В.Ф. Уткина

Кафедра ВПМ

«Распределенные системы обработки информации»

Отчет по лабораторной работе № 3

Выполнил:

студент группы 9413

Чистяков П.А

Проверил:

проф. Овечкин Г.В.

Рязань, 2023

**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).
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».
3. В прокси сервисе должны регистрироваться время события, тип события сервера (по типу выполняемой операции), время выполнения задания на сервере «xmlrpc\_server.ipynb».
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.
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.
6. Обновить содержимое удаленного репозитория GitHub по результатам выполнения пунктов 2-5.
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%](https://github.com/%25ИМЯ_АККАУНТА%25/%25ИМЯ_ПРОЕКТА%25). В репозитории должно быть минимум две фиксации файлов (пункты 1, 6).

**Ход выполнения работы**

Был разработан прокси сервис для перехвата запросов пользователя, сбора статистики и отправки ее на сервер, перенаправление запроса пользователя на сервер в соответствии с пунктами задания. Исходный код прокси-сервера и сервера статистики представлен на листинге 1 и 2 соответсвенно.

Листинг 1 – Исходный код прокси сервиса

from xmlrpc.server import SimpleXMLRPCServer

from xmlrpc.server import SimpleXMLRPCRequestHandler

import xmlrpc.client as client

import datetime

import time

import os

import pandas as pd

server = client.ServerProxy("http://localhost:8010/RPC2")

stats = client.ServerProxy("http://localhost:8011/RPC2")

class RequestHandler(SimpleXMLRPCRequestHandler):

rpc\_paths = ('/RPC2',)

proxy = SimpleXMLRPCServer(("localhost", 8012),

requestHandler=RequestHandler)

# Добавление строки в лог

class MyDispatcher:

def \_dispatch(self, method, args):

print(f"Received XML-RPC request for method '{method}' with arguments {args}")

sm = getattr(server, method)

start\_time = time.time()

result = sm(\*args)

end\_time = time.time()

# Вычислим разницу во времени в секундах

time\_difference\_seconds = end\_time - start\_time

# Преобразуем разницу в миллисекунды

time\_difference\_milliseconds = time\_difference\_seconds \* 1000

try:

stats.add\_log(method, datetime.datetime.fromtimestamp(start\_time).strftime("%Y-%m-%d %H:%M:%S"), time\_difference\_milliseconds)

except Exception as e:

print("Stat server exception!")

print(e)

return result

proxy.register\_instance(MyDispatcher())

print ("Listening on " + str(proxy.server\_address))

try:

proxy.serve\_forever()

print('That was a nice nap.')

except KeyboardInterrupt:

proxy.server\_close()

print('What a rude awakening!')

Листинг 2 – Исходный код сервера статистики

from xmlrpc.server import SimpleXMLRPCServer

from xmlrpc.server import SimpleXMLRPCRequestHandler

import datetime

import os

import pandas as pd

import sqlite3

# Подключение к базе данных (если базы данных нет, она будет создана)

conn = sqlite3.connect('log.db')

# Создание объекта курсора для выполнения SQL-запросов

cursor = conn.cursor()

# Создание таблицы

cursor.execute('''

CREATE TABLE IF NOT EXISTS server\_log (

id INTEGER PRIMARY KEY,

action\_name TEXT NOT NULL,

date TEXT,

duration INTEGER

)

''')

class RequestHandler(SimpleXMLRPCRequestHandler):

rpc\_paths = ('/RPC2',)

stat\_server = SimpleXMLRPCServer(("localhost", 8011),

requestHandler=RequestHandler)

insert = "INSERT INTO server\_log (action\_name, date, duration) VALUES (?, ?, ?)"

select = 'SELECT \* FROM server\_log'

# Добавление строки в лог

def add\_log(action\_name, date, duration):

cursor.execute(insert, (action\_name, date, duration))

conn.commit()

return True

stat\_server.register\_function(add\_log, 'add\_log')

def get\_log\_df():

df = pd.DataFrame()

cursor.execute(select)

rows = cursor.fetchall()

columns = [desc[0] for desc in cursor.description]

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

return df.to\_json()

stat\_server.register\_function(get\_log\_df, 'get\_log\_df')

print ("Listening on " + str(stat\_server.server\_address))

try:

stat\_server.serve\_forever()

print('That was a nice nap.')

except KeyboardInterrupt:

stat\_server.server\_close()

conn.close()

print('What a rude awakening!')

Результаты работы, разработанных серверов прикладываются к данному отчету отдельно.

Проект на github: https://github.com/paavill/distributed-information-development-systems

**Вывод**

В результате выполнения лабораторной работы были получены навыки работы с xmlrpc сервером и расширением его функционала. Также изучены принципы проксирования запросов.