Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о лабораторной работе №2

«Вычисления на основе взаимодействия сервисов»

по дисциплине

«Распределенные системы обработки информации»

Выполнил:

Студент группы 4413М

Анохин В.А.

Проверил:

доц. Князьков П.А.

**Цель работы**: получение практических навыков реализация взаимодействия сервисов.

**Условия выполнения**

Файл тетради клиента «xmlrpc\_client.ipynb» и файл тетради сервера «xmlrpc\_server.ipynb», полученные по результатам выполнения ЛР 1. Сервис-ориентированная архитектура вычислений на основе протокола XML-RPC.

**Ход работы**

В ходе работы были выполнены следующие действия.

1. Создать тетрадь «xmlrpc\_stats\_server.ipynb» с отдельный сервер сервиса статистики. В этот сервис должны приходить события работы функций сервера «xmlrpc\_server.ipynb».
2. Регистрироваться должны время события, тип события сервера (по типу выполняемой операции). Сохранять журнал событий в файл формата CSV. При отсутствии работы (выключенном) сервере «xmlrpc\_stats\_server.ipynb» работа функций сервера «xmlrpc\_server.ipynb» должна выполняться.
3. Сервис статистики должен иметь настройку ограничения на максимальный размер журнала событий в количестве записей. При превышении размера сохранять существующий журнал событий под новым именем, содержащим дату сохранения в формате YYYMMDD\_hhmmss. После этого создавать новый файл журнала событий и вести запись в него.

**Код сервера статистики**

import os  
from xmlrpc.server import SimpleXMLRPCServer  
from xmlrpc.server import SimpleXMLRPCRequestHandler  
  
import datetime  
  
  
class RequestHandler(SimpleXMLRPCRequestHandler):  
 rpc\_paths = ('/RPC2',)  
  
server = SimpleXMLRPCServer(("localhost", 8018),   
 requestHandler=RequestHandler)  
  
with open("stats\_server\_setting.txt", "r") as setting\_file:  
 MAX\_LINES = int(setting\_file.readline())  
  
def get\_file\_len(file):  
 with open(file, "r") as f:  
 return sum(1 for line in f)  
  
def create\_log\_file(file\_name):  
 f = open(file\_name, "w")  
 f.write("type,datetime\n")  
 return f  
  
  
*# Добавление строки в лог*def add\_log(sname):  
 current\_time = datetime.datetime.now()  
 log\_format = current\_time.strftime("%Y-%m-%d %H:%M:%S")  
 file\_format = current\_time.strftime("%Y%m%d\_%H%M%S")  
 files = os.listdir("log")  
 if files:  
 *# Получаем список файлов* files = [os.path.join("log", f) for f in files]  
 *# Выбираем последний файл* last\_file = max(files)  
 *# Определяем количество строк в файле* lines = get\_file\_len(last\_file)  
 if lines > MAX\_LINES:   
 f = create\_log\_file("log/"+file\_format+".csv")  
 else: f = open(last\_file, "a+")  
 else: f = create\_log\_file("log/"+file\_format+".csv")  
 f.write(str(sname)+','+ log\_format +'\n')  
 f.close()  
 return True  
  
server.register\_function(add\_log, 'add\_log')  
  
  
print ("Listening on port 8018...")  
server.serve\_forever()

**Код сервера**

*# Добавление в лог через сервер*def add\_log(log\_line):  
 try: stats\_server.add\_log(log\_line)  
 except Exception as e:   
 print("Не удалось сохранить лог: "+log\_line+"\n"+str(e))  
*# Тест*def ping():  
 add\_log("ping")  
 return True

*# Для остальных методов аналогичный код*

1. В рамках клиента «xmlrpc\_client.ipynb» реализовать получение содержимого журнала событий сервера статистики с возможностью получения среза по типу выполняемой операции (события) и времени.

**Код клиента**

datetime\_format = "%Y-%m-%d %H:%M:%S"  
filename\_format = "log\%Y%m%d\_%H%M%S.csv"  
  
def get\_files\_by\_date(start\_date, end\_date):  
 files = os.listdir("log")  
 if files:  
 files = [os.path.join("log", f) for f in files]  
 *# Выбираем только те файлы, которые находятся в нужном временном диапазоне* filtered = [file for file in files  
 if start\_date.strftime(filename\_format) <= file <= end\_date.strftime(filename\_format)]  
 early\_files = [file for file in files if file < start\_date.strftime(filename\_format)]  
 if len(early\_files) > 0:  
 *# Добавляем последний файл перед указанной датой* last\_file = early\_files[-1]  
 filtered.insert(0, last\_file)  
 return filtered  
   
def get\_log(start\_date, end\_date):  
 files = get\_files\_by\_date(start\_date, end\_date)  
 logs = []  
 for file in files:  
 logs.append(pd.read\_csv(file, sep=',', encoding='utf8'))  
 log = pd.concat(logs, ignore\_index=True)  
 return log[(start\_date.strftime(datetime\_format) <= log['datetime']) &   
 (log['datetime'] <= end\_time.strftime(datetime\_format))]  
  
start\_time = datetime(year=2024, month=11, day=7, hour=12, minute=0, second=0)   
end\_time = datetime(year=2024, month=11, day=7, hour=12, minute=30, second=0)  
  
types = []  
*# types = ['ping', 'now']*filter\_log = get\_log(start\_time, end\_time)  
if len(types) > 0:  
 filter\_log = filter\_log[filter\_log['type'].isin(types)]  
display(filter\_log)

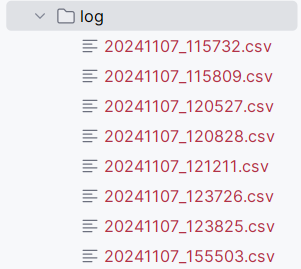


Рисунок 1 – Запись логов в каталог

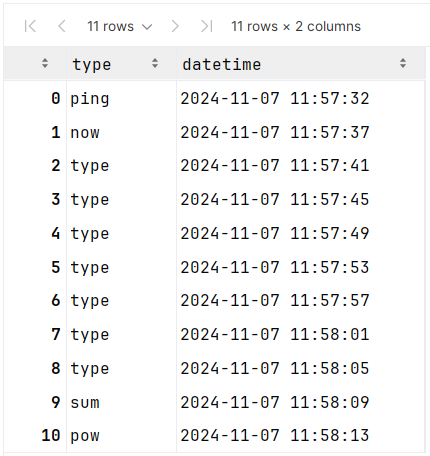


Рисунок 2 – Загрузка логов по времени

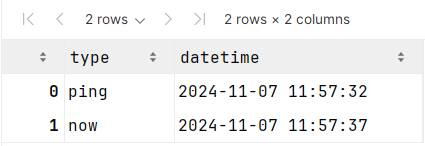


Рисунок 3 – Загрузка логов по времени с фильтрацией типов

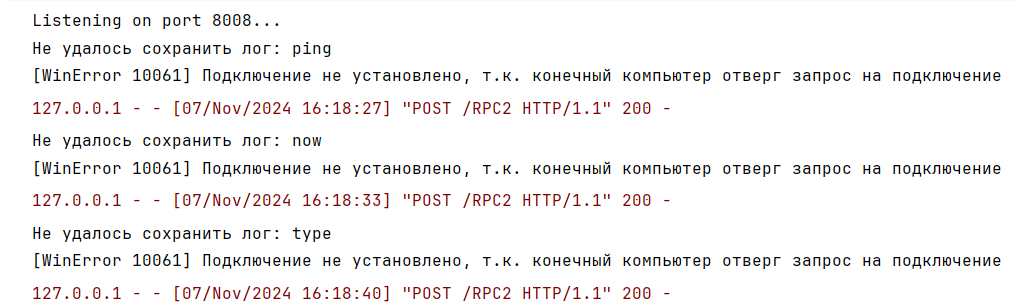


Рисунок 4 – Продолжение работы сервера при отсутствии соединения

**Вывод**: в ходе выполнения работы были получены практические навыки реализации взаимодействия сервисов.