МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГAОУ ВО

«Пермский государственный национальный исследовательский университет»

**Отчёт**

по лабораторной работе №3 «Разработка распределенного приложения, использующего технологию веб-служб»

по дисциплине «Технологии разработки распределенных приложений»

Работу выполнил Проверил

студент гр. ФИТ-2 доцент кафедры

Ханов Дамир математического обеспечения

Радикович вычислительных систем,

18.03 .2021 к.ф.-м.н., доц.

Деменев Алексей

Геннадьевич

Пермь, 2021 г

**Содержание**

[**Введение** 3](#_Toc1)

[1. **Основная часть** 4](#_Toc2)

[**1.1 Постановка задачи** 4](#_Toc3)

[**1.2 Выбор предметной области и инструментов** 4](#_Toc4)

[1.3 **Описание программ сервера и клиента** 5](#_Toc5)

[**1.4 Тестирование** 6](#_Toc6)

[**Заключение** 9](#_Toc7)

[**Библиографический список** 10](#_Toc8)

# **Введение**

**Цель:** изучение возможностей технологии веб-служб для создания распределенных приложений.

**Формируемые компетенции:** способность применять на практике теоретические основы и общие принципы разработки распределенных систем; способность использовать на практике стандарты сетевого взаимодействия компонент распределенной системы.

Необходимо реализовать веб-сервис и приложение, его использующее

1. Веб-сервис и приложение, его использующее, должны быть разработаны на разных объектно-ориентированных языках программирования.
2. Веб-сервис должен принимать параметры и передавать приложению результат.
3. В качестве параметров должны передаваться объекты классов, написанных самостоятельно, т.е. не должны передаваться строки, числа или другие простейшие типы.

# **Основная часть**

## **1.1 Постановка задачи**

Спроектировать, реализовать, протестировать и оценить приложение в соответствии с таблицей №1.

***Таблица 1. Распределение баллов за выполнение работы (***Максимальное количество баллов — 6)***:***

|  |  |
| --- | --- |
| **Требование к заданию** | **Максимальное количество баллов** |
| Приложение, написанное студентом, работает в сети Интернет без сбоев. | 2 |
| Веб-сервис и приложение, его использующее, написаны на разных языках программирования. | 1 |
| Веб-сервис принимает от приложения исходные данные и возвращает результат. | 1 |
| В качестве параметров веб-сервис принимает объекты классов. | 2 |

## **1.2 Выбор предметной области и инструментов**

Для разработки серверного и клиентского приложения был выбран язык программирования Python 3.9[1][2]. В качестве IDE использовался PyCharm Community Edition[3]. PyCharm Community Edition является бесплатной версией, обладающей усеченным набором возможностей[4].

Веб-сервис реализуется с помощью «gRPC» [5][6] - это высокопроизводительный фреймворк для удаленного вызова процедур, разработанный компанией Google. В качестве языка описания веб-сервиса используется «Protocol Buffers (Protobuf)» - язык описания интерфейсов (IDL) [6], который подразумевает определение сервиса через указание методов, которые могут быть вызваны удалённо, с их параметрами и типами принимаемых и возвращаемых данных. Это осуществляется с помощью текстового файла с расширением *.proto*. В этом файле описываются «сообщения» – объекты, содержащие набор полей, представляющих собой пары {тип данных – имя}. Также в *протофайле* описываются сервисы, каждый из которых содержит набор методов с описанием типа принимаемых и возвращаемых данных.

В качестве утилиты для генерации кода использовалась утилита *protoc* [7]. Поставляется вместе с Protocol Buffers.

Был сформирован следующий «протофайл»:

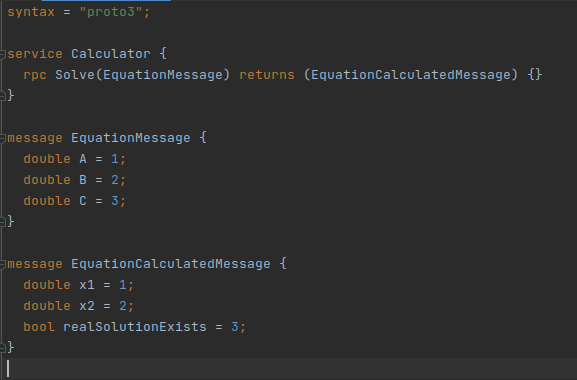


Рис. 1. Структура протофайла

Мы видим, что было описано два объекта: квадратное уравнение (EquationMessage) и решение квадратного уравнения (EquationCalculatedMessage)

Так же в файле описан сервис «Calculator», в котором есть только 1 метод «Solve», который принимает EquationMessage и возвращает EquationCalculatedMessage

## **Описание программ сервера и клиента**

Программа клиент представляет CLI приложение, написанное на JavaScript[9] использованием NodeJS[10]. Из-за выбранной задачи у нас существует единственный вариант использования: решение квадратного уравнения. В данном варианте использования происходят следующие шаги:

1. Устанавливаются переменные окружения host и port. В случае, если они не установленны, то задаются значения по умолчанию «127.0.0.1» и «8008» соответственно
2. Запускаем сервер
3. Запускаем клиента
4. Пользователь начинает ввод
5. Пользователь может завершить работу в приложении клиенте
6. В случае, если пользователь выбрал решение уравнения, то он вводит 3 числа - коэффициенты в квадратном уравнении
7. На основе этих данных формируется объект класса EquationMessage
8. Данный объект отправляется на сервер
9. Сервер обрабатывает полученные данные (решает уравнение)
10. Формирует объект класса EquationCalculatedMessage и отправляет его клиенту

## **1.4 Тестирование**

При разработке с помощью инструмента gRPC на основе файла service.proto было получено два автоматически сгенерированных файла – service\_pb2.py и service\_pb2\_grpc.py – для работы с спроектированным сервисом. Для генерации была выполнена команда: *python3 -m grpc\_tools.protoc -I. --python\_out=./server --grpc\_python\_out=./server service.proto*

Из исходного протофайла были сгенерированы файлы и для клиентского приложения с использованием команды: grpc\_tools\_node\_protoc -I=. --js\_out=import\_style=commonjs,binary:./client --grpc\_out=grpc\_js:./client service.proto

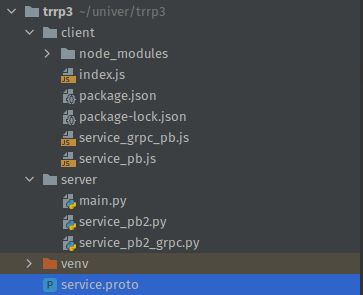


Рис. 2. Автоматически сгенерированные файлы для клиента и сервера

**Протестируем взаимодействие клиента и сервера.**

1. Запустили сервер

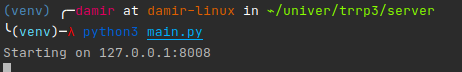


Рис. 3. Запуск сервера

1. В другом терминале запускаем клиента

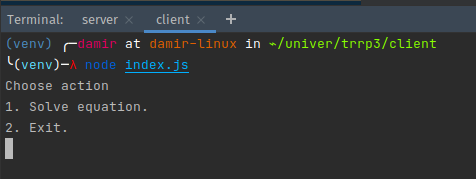


Рис. 4. Запуск клиента

1. Введём коэффициенты

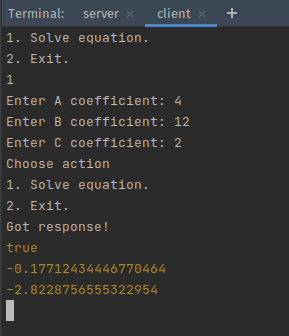


Рис. 5. Ввели коэффициенты, получили результат

1. Мы можем заметить, что запрос следующего действия пользователя происходит до того, как был получен ответ от сервера. Это происходит из-за особенностей NodeJS.
2. Введём 2 для выхода из программы клиента

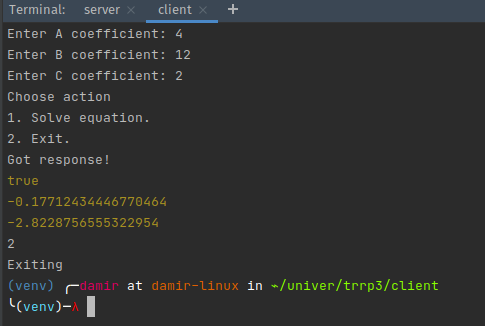


Рис. 6. Завершение работы клиентского приложения

1. Мы можем заметить, что сервер всё ещё продолжает работать. Для его остановки введём сочетание клавиш «Ctrl+C»

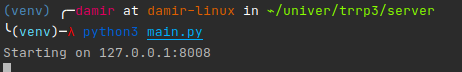


Рис. 7. Сервер продолжает работать после завершения работы клиента

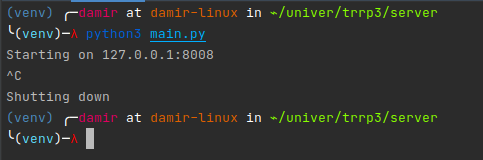


Рис. 8. Завершение работы сервера

# **Заключение**

Было реализовано требование использование различных языков программирования для клиентской и серверной части - 1 балл

Было реализовано требование передачи объектов класса, а не примитивных типов данных - 2 балла

Было реализовано требование принятия данных и возврата результата сервером - 1 балл

Было реализовано требования работы приложения в сети без сбоев - 2 балла

Итого: 6 баллов.

# **Библиографический список**

* + 1. Лицензия Python 3.8, <https://docs.python.org/3.7/license.html> (Дата обращения 18.03.2021)
    2. Python 3.8.0. Ссылка наскачивание URL: <https://www.python.org/downloads/release/python-380/> (дата обращения: 18.03.2021)
    3. PyCharm Community Edition. Распространяется под лицензией Apache 2, URL: <https://www.jetbrains.com/pycharm/features/editions_comparison_matrix.html> (Дата обращения 18.03.2021)
    4. PyCharm Community Edition. Ссылка на скачивание. URL: <https://www.jetbrains.com/pycharm/download/#section=windows> (Дата обращения 18.03.2021)
    5. Ссылка на FAQ по gRPC. URL: <https://grpc.io/faq/> (дата обращения: 18.03.2021)
    6. Ссылка на лицензию gRPC (Apache 2.0). URL: <https://github.com/grpc/grpc/blob/master/LICENSE> (дата обращения: 18.03.2021
    7. protocolbuffers/protobuf. URL: <https://github.com/protocolbuffers/protobuf> (дата обращения: 21.12.2020)
    8. Спецификация IDL. URL: <https://www.omg.org/spec/IDL/4.2/> (дата обращения: 18.03.2021)
    9. Язык программирования JavaScript. URL: https://www.javascript.com/ (дата обращения: 18.03.2021)
    10. Технология NodeJS. URL: https://nodejs.org/ (дата обращения: 18.03.2021)