МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО

«Пермский государственный национальный исследовательский университет»

Механико-математический факультет

Кафедра информационной безопасности и систем связи

**Лабораторная работа №4 по дисциплине «Технологии разработки распределенных приложений»**

|  |  |
| --- | --- |
|  | Работу выполнили:  Тарарков Андрей Викторович,  Чернов Павел Константинович,  Якин Никита Николаевич,  КМБ-1-16  Работу принял:  Доцент кафедры математического обеспечения вычислительных систем, к. ф.-м.н., доц.  Деменев Алексей Геннадьевич |

Пермь, 2020 г

Оглавление

[Введение 3](#_Toc57978429)

[1. Основная часть 4](#_Toc57978430)

[1.1 Постановка задачи 4](#_Toc57978431)

[1.2 Проектирование 5](#_Toc57978432)

[1.3 Разработка 8](#_Toc57978433)

[1.4 Сценарии использования 13](#_Toc57978434)

[1.5 Тестирование 14](#_Toc57978435)

[Заключение 17](#_Toc57978436)

[Список источников 18](#_Toc57978437)

# Введение

**Цель**: изучение методов коммуникации процессов в сети, а также средств динамического конфигурирования распределенных приложений.

Требования к выполнению работы:

* Приложение должно обеспечивать параллельную работу нескольких клиентов и серверов. Дополнительное требование: возможность запуска нескольких серверов на одном компьютере.
* Клиентские приложения должны автоматически находить серверы для обслуживания и выполнения заданных функций.
* Серверы системы могут выполнять различные функции.
* При разрыве сеанса приложения должны автоматически восстанавливать свою работоспособность.
* Для хранения данных и доступа к ним применить ADO и/или ADO.NET (или их аналоги).
* Приложения должны поддерживать возможность взаимодействия в различных режимах.
* Для организации взаимодействия нужно использовать различные средства коммуникации (именованные каналы, мейлслоты, сокеты, MSMQ, .Net Remoting, web-сервисы, WCF-сервисы), сравнив их возможности.

В программе необходимо использовать открытые данные.

По окончании выполнения задания необходимо подготовить отчёт. Отчет по выполнению задания должен включать:

1. Общее описание приложения. Постановка задачи, введение в предметную область.
2. Архитектура системы. Обоснование выбора данного типа архитектуры распределенного приложения. Алгоритм работы приложения в целом.
3. Архитектура каждого из логических компонент системы (серверы, клиенты, диспетчеры). Подходы к реализации. Алгоритмы работы. Многопоточность, обоснование.
4. Методы коммуникаций компонентов системы (клиент→сервер, сервер→клиент и т.д.). Обоснование выбора этих методов коммуникации.
5. Способ передачи данных (синхронная / асинхронная, однонаправленная / двунаправленная и т.д.). Обоснование.
6. Структура передаваемых данных. Вид протоколов, обоснование выбора.
7. Отказоустойчивость системы. Как система поведет себя, если «исчезнет» один или несколько ее компонент. Что произойдет с системой, если «исчезнувший» компонент будет восстановлен на другом узле сети.
8. Работа с базой данных (если используется). Обоснование.
9. Исходный код приложений с комментариями.

# Основная часть

## Постановка задачи

Обеспечить проектирование, реализацию, проверку созданного приложения требованиям в таблице:

***Распределение баллов за выполнение работы (***Максимальное количество баллов — 30)***:***

|  |  |
| --- | --- |
| **Критерий оценивания** | **Оценка** |
| Распределенное приложение спроектировано с учетом особенностей предметной области. Выбрана наиболее подходящая модель распределенной системы. Если используется распределенная база данных, то тиражирование данных имеет подходящий для данной предметной области механизм и архитектуру. | 3 |
| Архитектура системы является оптимальной для заданных при разработке критериев. В отчете присутствует обоснование выбора данного типа архитектуры. | 3 |
| Приложение обеспечивает параллельную работу нескольких клиентов и серверов, в том числе на одном компьютере. Серверы распределенной системы выполняют различные функции. | 2 |
| Приложение является масштабируемым, позволяет добавлять новых участников взаимодействия без переписывания кода и перезапуска приложений. | 2 |
| Существует возможность динамического реконфигурирования системы. | 4 |
| Для организации взаимодействия компонент распределенной системы используется не менее четырех различных средств коммуникации. В отчете присутствует четкое обоснование выбора средств взаимодействия для каждого конкретного случая. | 4 |
| Система является отказоустойчивой. В случае если один и/или несколько компонент системы аварийно завершают свою работу. | 2 |
| Распределенное приложение продолжает работать и в случае, если после аварийного завершения некоторого компонента, он восстановлен на другом узле вычислительной сети. | 3 |
| Отчет содержит подробное описание архитектуры каждого компонента распределенного приложения. | 3 |
| В отчете описана структура передаваемых данных, формат сообщений и вид протокола, используемого для этого. | 2 |
| В отчете представлено описание способа передачи сообщений при коммуникации компонентов распределенной системы с обоснованием. | 2 |

## Проектирование

Предметная область – определение синонимов для английских слов.

Семантический анализ текста – актуальная и востребованная задача в сфере современных информационных технологий: оценка качества и релевантности текстовых статей, создание чат-ботов для работы с клиентами, анализ открытых данных и т.д.

Разрабатываемое приложение будет позволять оперативно получать список синонимов для заданного слова английского языка, что может послужить для реализации функции выделения семантических отношений в тексте.

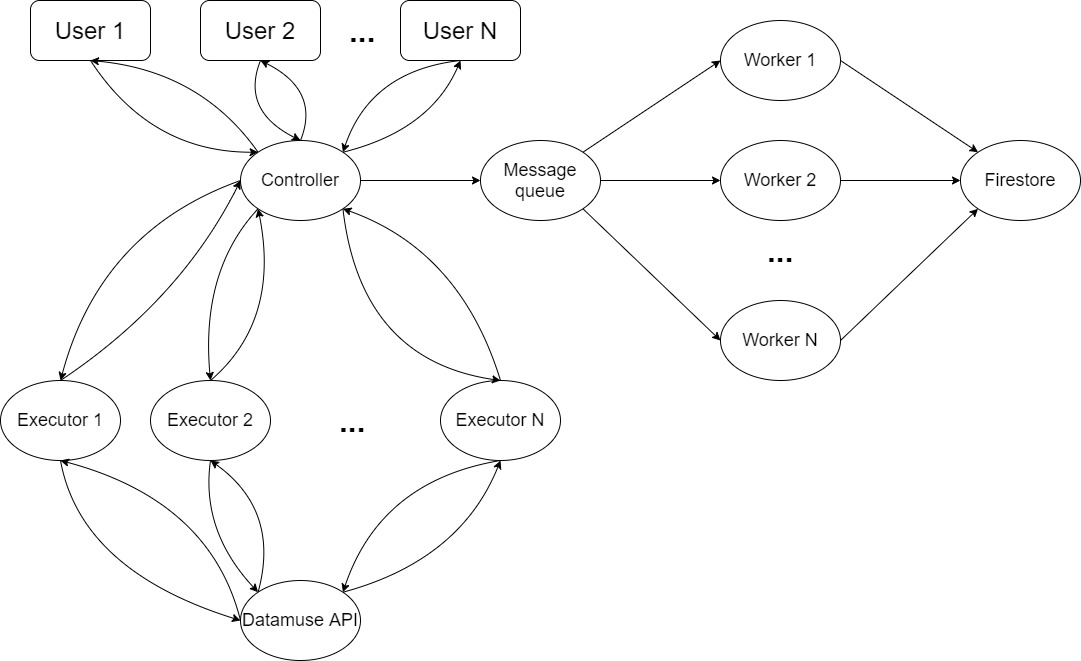
Для разработки модулей распределённого приложения был выбран язык программирования «Go» версии 1.15 [1]. Для реализации приложения применялась среда разработки GoLand 2020.2.3 [2,3]. Данная среда разработки может бесплатно использоваться студентами и преподавателями для учебных целей (некоммерческая разработка) [4].

Для работы определённых модулей в облаке используется ряд сервисов от Google. Сервисы Cloud Run, Firestore, Pub/Sub, App Engine могут использоваться бесплатно с рядом ограничений. Также новым пользователям предоставляются возможности для масштабирования ресурсов в применяемых сервисах [5].

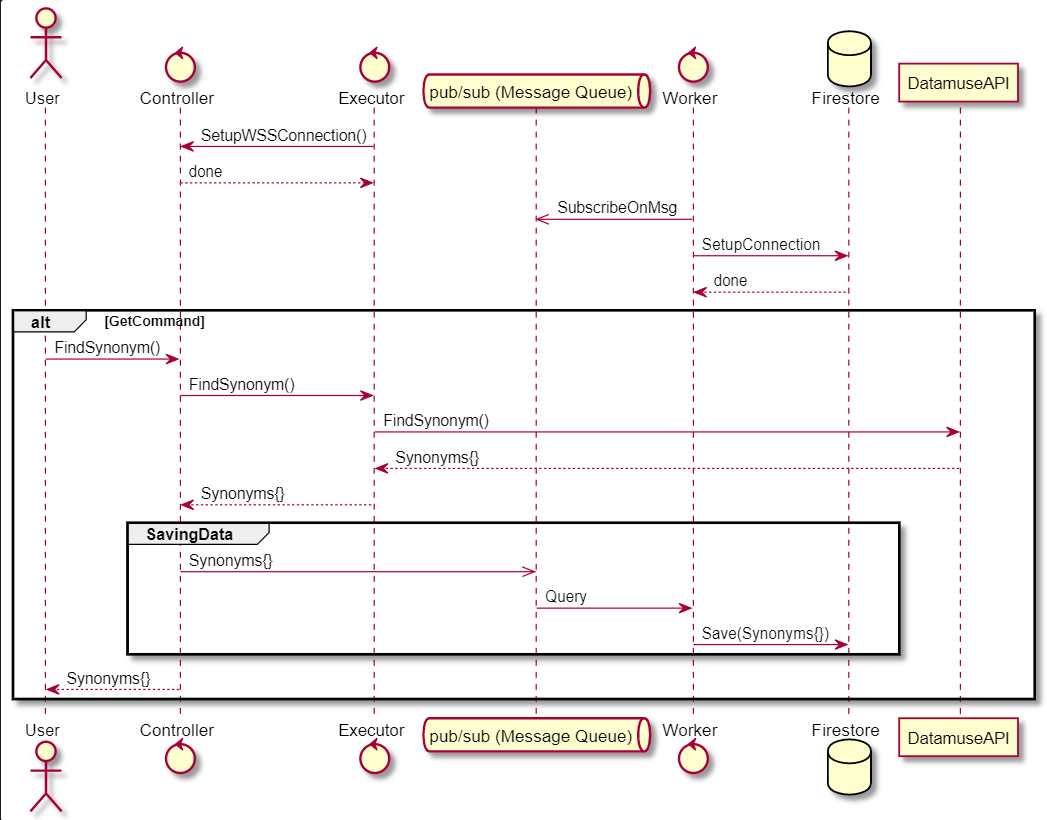
Для тестирования применялся бесплатный браузер для ПК «Яндекс.Браузер» [6,7].

**Описание работы распределённого приложения**

1. Пользователем формируется и посылается запрос
2. Запрос принимается «Контроллером» (Controller), затем запрос передаётся доступному «Исполнителю» (Executor)
3. «Исполнителем» посылается запрос к источнику открытых данных (datamuse.com) [8]
4. Получив результат от источника данных, «Исполнитель» возвращает результат «Контроллеру»
5. Принятый от «Исполнителя» результат запроса отправляется пользователю, а также посылается в очередь сообщений (Message queue)
6. Слушатели очереди (Workers) ожидают появления в очереди данных и при их появлении записывают данные в БД



**Рис. 1.** Схема архитектуры распределённого приложения



**Рис. 2.** Схема работы (UML-диаграмма последовательностей) распределённого приложения

«Контроллер» необходим для распределения пользовательских запросов, что позволяет клиентскому приложению (браузеру) не решать самостоятельно проблему выбора доступного «Исполнителя».

Множество «Исполнителей» обеспечивает системе дополнительную отказоустойчивость, так как при выходе из строя одного «Исполнителя» запросы будут передаваться другому - доступному – «Исполнителю».

Применение очереди сообщений позволяет балансировать нагрузку на отдельные экземпляры модуля слушателя сообщений, а также ускорить загрузку данных в базу за счёт параллельной работы слушателей очереди.

Использование очереди сообщений (Message queue) и нескольких слушателей очереди (Workers) позволяет снизить вероятность потери данных в случае возникновения сбоев в работе элементов системы, так как даже в случае выхода из строя одного или всех слушателей очереди данные, получаемые от «Контроллера», будут сохраняться в очереди до восстановления работы слушателей.

Использование базы данных позволяет создавать локальную версию запрашиваемых данных, сочетая это с функцией журналирования и предоставления доступа к логам.

## 1.3 Разработка

**Пользователь**

В качестве клиентского приложения пользователь использует браузер. Пользователем на REST API «Контроллера» посылается GET-запрос следующего вида:

https://refined-byte-297215.ew.r.appspot.com/command?word=[слово],

где [слово] – слово английского языка, для которого требуется получить синонимы. Таким образом, для связи пользователя с «Контроллером» применяется протокол HTTPS.

В качестве результата пользователь принимает JSON следующей структуры:

|  |
| --- |
| [{"word":"examination","tags":["n"]},{"word":"evaluation","tags":["n"]},{"word":"tester","tags":["n"]},{"word":"laboratory","tags":["n","adj"]},{"word":"validation","tags":["n"]},{"word":"samples","tags":["n"]},{"word":"sampling","tags":["n","v"]},{"word":"inspections","tags":["n"]},{"word":"inspection","tags":["n"]},{"word":"verification","tags":["n"]}] |

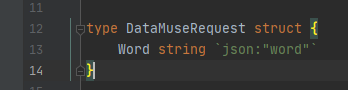
Поле «word» содержит строку – синоним исходного слова в английском языке. Поле «tags» содержит список меток, обозначают принадлежность слова к частям речи: «n» означает существительное, «v» означает глагол, «adj» означает прилагательное, «adv» означает наречие, а «u» означает, что часть речи не относится ни к одному из них или не может быть определена.

Благодаря простоте формирования GET-запросов распределённое приложение может как использоваться пользователями напрямую (посредством браузера), так и использоваться другими приложениями, которые будут формировать запросы автоматически. Также подобное решение позволяет обеспечить возможность использования системы вне зависимости от пользовательской платформы.

**Контроллер**

Модуль «Контроллера» выполняется на Google App Engine [9] – хостинге web-приложений на серверах Google. Это гарантирует надёжность работы данного модуля за счёт применения ресурсов и сервисов Google [10], а также за счёт возможности масштабирования используемых для выполнения приложения ресурсов [11].

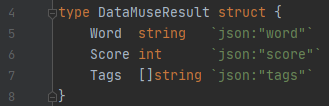
«Контроллер» принимает от пользователей синхронные GET-запросы, структура которых описана ранее. При его получении «Контроллер» выбирает доступного «Исполнителя» и передаёт ему пользовательский запрос на исполнение. Связь с «Исполнителем» осуществляется посредством WebSocket с шифрованным соединением (WSS) [12], «Исполнителям» передаётся JSON следующей структуры:



**Рис. 3.** Структура передаваемых «Исполнителям» данных

Поле «word» содержит слово английского языка, для которого требуется получить синонимы.

Результаты запросов «Контроллер» так же принимает посредством WebSocket, структура JSON с данными результата запроса имеет следующий вид:

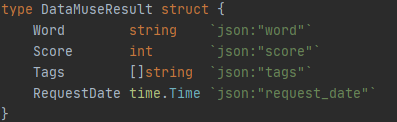


**Рис. 4.** Структура данных, принимаемых от «Исполнителей»

Данная структура полностью соответствует структуре результата запроса к API источника открытых данных.

При получении от «Исполнителя» результата «Контроллер» посылает асинхронный запрос на добавление данных в очередь сообщений. В качестве очереди сообщений использовался сервис Google Pub/Sub - служба обмена сообщениями в реальном времени от Google [13]. Использование данного сервиса позволяет повысить уровень отказоустойчивости системы за счёт авто-масштабирования пропускной способности очереди [14]. «Контроллер» передаёт данные в очередь сообщений с помощью клиентских библиотек Google, которые взаимодействуют с очередью посредством веб-сервиса gRPC.

В очередь сообщений передаются данные, имеющие следующую структуру:

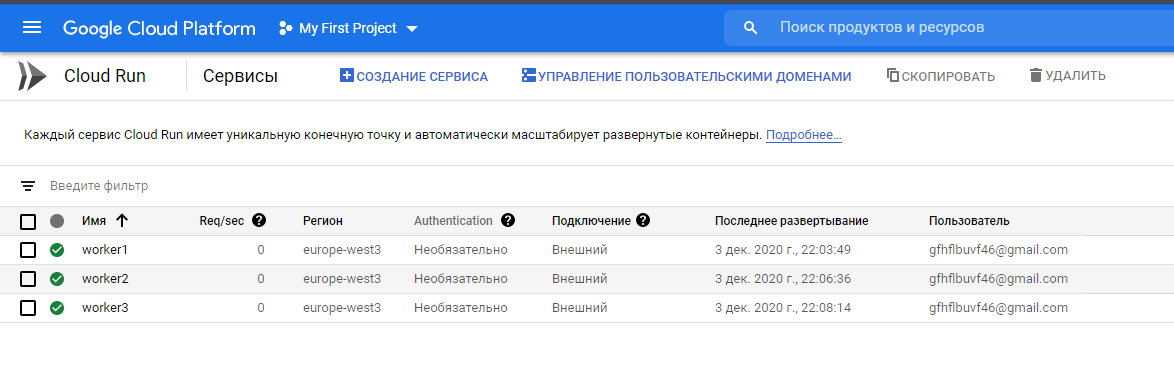


**Рис. 5.** Структура данных, передаваемых в очередь сообщений

Данная структура дублирует структуру данных, получаемую «Контроллером» от «Исполнителей», при этом дополняя ему полем RequestDate, содержащим дату и время получения результата запроса «Контроллером».

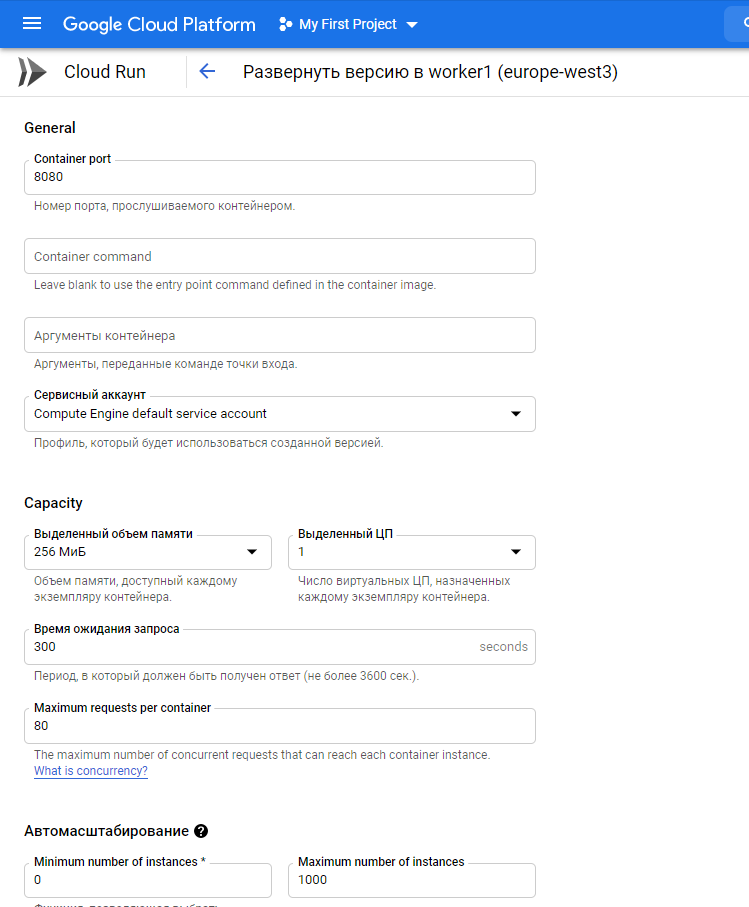
**Слушатель очереди**

Код модуля «Слушатель очереди» (Worker) исполняется на Google Cloud Run [15] – управляемой платформе для развертывания и масштабирования приложений. Использования этого сервиса позволяет повысить надёжность работы данного модуля, а также предоставляет возможность масштабирования ресурсов, используемых для исполнения экземпляра модуля [10,16]. Последнее реализуется за счёт одновременного запуска и использования нескольких экземпляров модуля слушателя. Это демонстрируется на рисунке 6.



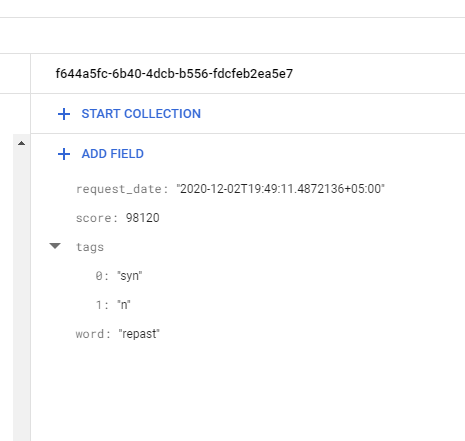
**Рис. 6.** Одновременная работа нескольких экземпляров модуля слушатель очереди

Также имеется возможность динамического реконфигурирования экземпляров данного модуля:



**Рис. 7.** Динамическое реконфигурирование модуля

Модуль использует клиентские библиотеки Google, которые взаимодействуют с очередью посредством веб-сервиса gRPC [17]. Посредством REST API [18] полученная из очереди структура данных посылается в облачную Google Cloud Firestore – облачную NoSQL базу данных от Google [19]. Использование данного сервиса позволяет повысить уровень масштабируемости и надёжности работы распределённого приложения, так как Google Cloud Firestore является бессерверной легко масштабируемой БД [10,20]. Данные в самой БД имеет следующий вид:



**Рис. 8.** Структура данных в Cloud FireStore

**Исполнители**

Эксплуатация системы подразумевает запуск «Исполнителей» на выделенных машинах. Также допускается их запуск на пользовательских машинах. Подобная схема размещения данного модуля позволяет избегать ограничений в обращениях к API источника открытых данных. К примеру, если бы обращения к API источника данных производились «Контроллером» и его IP-адрес по каким-либо причинам был бы занесён в чёрный список ресурса (например, по причине превышения допустимого количества запросов в единицу времени), то работа распределённого приложения была бы парализована.

Также подобная схема размещения данного модуля повышает уровень отказоустойчивости – за счёт запуска экземпляров модуля на множестве независимых выделенных машин – и масштабируемости – за счёт возможности запуска дополнительных экземпляров модуля без необходимости внесения изменений в код или перезапуска каких-либо модулей распределённого приложения.

При получении от «Контроллера» запроса посредством web-сокетов (WSS) «Исполнитель» на REST-API источника открытых данных посылается GET-запрос следующего вида:

*https://api.datamuse.com/words?max=10&ml=[слово]*,

где [слово] – слово английского языка, для которого требуется получить синонимы. Таким образом, для связи «Исполнителя» с источником открытых данных применяется протокол HTTPS.

Данные результата, получаемые от источника данных в ответ на пользовательский запрос, представляют собой JSON и имеют следующую структуру:

|  |
| --- |
| [{"word":"instance","score":102125,"tags":["syn","n"]},{"word":"illustration","score":93669,"tags":["syn","n"]},{"word":"exemplar","score":92778,"tags":["syn","n"]},{"word":"lesson","score":88191,"tags":["syn","n"]},{"word":"model","score":87649,"tags":["syn","n"]},{"word":"case","score":84753,"tags":["syn","n","adj"]},{"word":"representative","score":81775,"tags":["syn","n","adj"]},{"word":"object lesson","score":81581,"tags":["syn","n"]},{"word":"exercise","score":81330,"tags":["syn","n"]},{"word":"good example","score":77136,"tags":["syn","n"]}] |

Данная структура включает [8]:

* Поле «Word» - содержит синоним слова из исходного запроса
* Поле «score» - не имеет интерпретируемого значения и используется ресурсом для ранжирования результатов
* Поле «tags» - содержит метку «syn», которая означает, что слова из поля «Word» является синонимом к слову из исходного запроса, а также одну или несколько меток принадлежности слова из поля «Word» к различным частям речи («n» означает существительное, «v» означает глагол, «adj» означает прилагательное, «adv» означает наречие, а «u» означает, что часть речи не относится ни к одному из них или не может быть определена)

Полученные таким образом данные с сохранением структуры передаются «Контроллеру» посредством WebSocket (WSS).

## 1.4 Сценарии использования

**Сценарий №1**

Предусловия: отсутствуют запущенные и доступные экземпляры модуля «Исполнитель»

1. Формирование и отправка запроса пользователем
2. Ожидание получения ответа

Ожидаемый результат: Получение соответствующей ошибки.

**Сценарий №2**

Предусловие: Запущен и доступен хотя бы один экземпляр модуля «Исполнитель», а также остальные модули

1. Формирование и отправка запроса пользователем
2. Ожидание получения ответа

Ожидаемый результат: Получен ответ в виде соответствующей структуры.

## 1.5 Тестирование

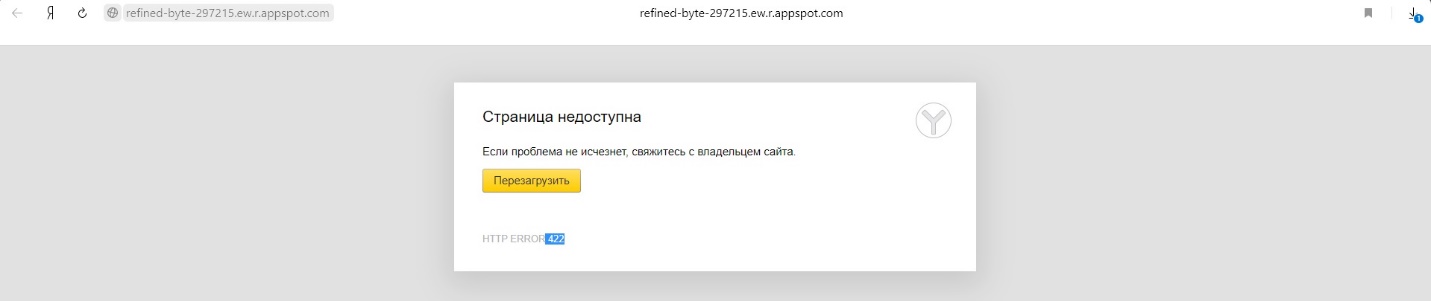
Сценарий №1

Формирование запроса:

D:\Учебная\5 КУРС\1 триместр\ТРРП\лаб4\Скрины\screen_2.jpg

**Рис. 9.** Формирование запроса

Получение ответа:



**Рис. 10.** Получение соответствующей ошибки

Результат: Получена соответствующая ошибка. Тест пройден успешно.

Сценарий №2

Формирование запроса:

D:\Учебная\5 КУРС\1 триместр\ТРРП\лаб4\Скрины\screen_3.jpg

**Рис. 11.** Формирование запроса

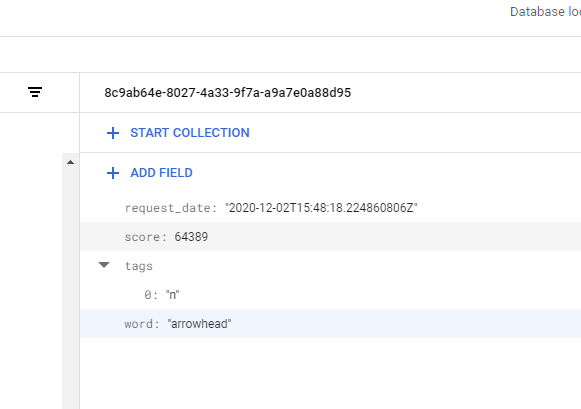
Получение ответа:



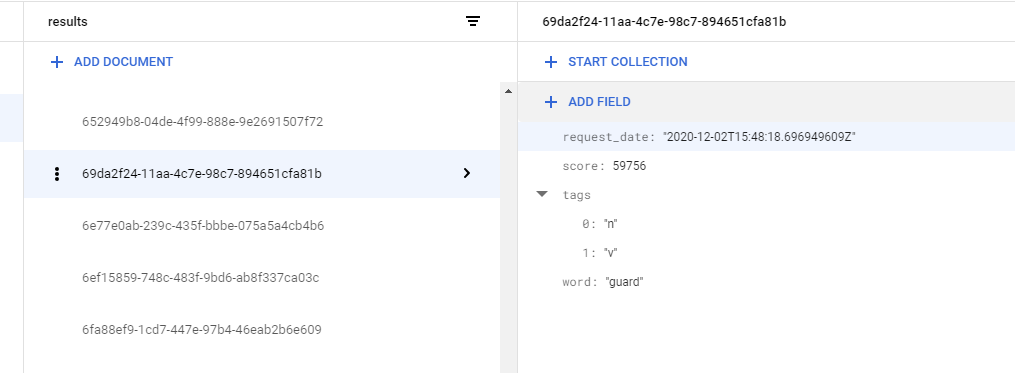
**Рис. 12.** Получение ответа

Результат: Получен ответ в виде соответствующей структуры данных. Тест пройден успешно.

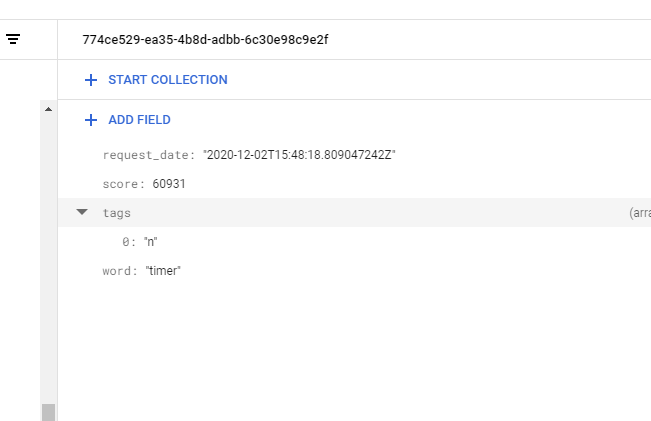
Чтобы убедиться в возможности параллельной работы нескольких пользователей, произведём несколько одновременных запросов и пронаблюдаем появление записей об их выполнении в базе данных:



**Рис. 13.** Запись о первом запросе



**Рис. 14.** Запись о втором запросе



**Рис. 15.** Запись о третьем запросе

# Заключение

В результате работы была достигнута цель и выполнены следующие требования:

* Требование «Распределенное приложение спроектировано с учетом особенностей предметной области. Выбрана наиболее подходящая модель распределенной системы. Если используется распределенная база данных, то тиражирование данных имеет подходящий для данной предметной области механизм и архитектуру» выполнено полностью. Ожидаемое количество баллов: 3.
* Требование «Архитектура системы является оптимальной для заданных при разработке критериев. В отчете присутствует обоснование выбора данного типа архитектуры» выполнено полностью. Ожидаемое количество баллов: 3.
* Требование «Приложение обеспечивает параллельную работу нескольких клиентов и серверов, в том числе на одном компьютере. Серверы распределенной системы выполняют различные функции» выполнено полностью. Ожидаемое количество баллов: 2.
* Требование «Приложение является масштабируемым, позволяет добавлять новых участников взаимодействия без переписывания кода и перезапуска приложений» выполнено полностью. Ожидаемое количество баллов: 2.
* Требование «Существует возможность динамического реконфигурирования системы» не выполнено. Ожидаемое количество баллов: 4.
* Требование «Для организации взаимодействия компонент распределенной системы используется не менее четырех различных средств коммуникации. В отчете присутствует четкое обоснование выбора средств взаимодействия для каждого конкретного случая» выполнено полностью. Ожидаемое количество баллов: 4.
* Требование «Система является отказоустойчивой. В случае, если один и/или несколько компонент системы аварийно завершают свою работу, распределенное приложение продолжает работать и в случае, если после аварийного завершения некоторого компонента, он восстановлен на другом узле вычислительной сети» выполнено полностью. Ожидаемое количество баллов: 5.
* Требование «Отчет содержит подробное описание архитектуры каждого компонента распределенного приложения» выполнено полностью. Ожидаемое количество баллов: 3.
* Требование «В отчете описана структура передаваемых данных, формат сообщений и вид протокола, используемого для этого» выполнено полностью. Ожидаемое количество баллов: 2.
* Требование «В отчете представлено описание способа передачи сообщений при коммуникации компонентов распределенной системы с обоснованием» выполнено полностью. Ожидаемое количество баллов: 2.

Итоговое количество баллов: 30

# Список источников

1. Описании BSD-лицензии, которой лицензируются приложения, написанные на «Go» (дата обращения: 03.12.2020): https://golang.org/LICENSE
2. Ссылка на скачивание дистрибутива GoLand 2020.2.3 (дата обращения: 03.12.2020): https://download.jetbrains.com/go/goland-2020.2.3.exe
3. Ссылка на лицензию, по которой распространяется GoLand в образовательных целях (дата обращения: 03.12.2020): https://www.jetbrains.com/student/license\_educational.html
4. Лицензия на бесплатное использование GoLand в учебных целях (дата обращения: 03.12.2020): https://www.jetbrains.com/community/education/#students
5. Описание бесплатных для использования сервисов Google и их ограничений (дата обращения: 03.12.2020): https://cloud.google.com/free?hl=ru
6. Ссылка на скачивание «Яндекс.Браузер» (дата обращения: 03.12.2020): https://browser.yandex.ru/
7. Лицензионное соглашение на использование программ Браузер «Yandex» (дата обращения: 03.12.2020): https://yandex.ru/legal/browser\_agreement/
8. Документация по ресурсу datamuse API (дата обращения: 03.12.2020): https://www.datamuse.com/api/
9. Документация по Google App Engine (дата обращения: 03.12.2020): https://cloud.google.com/appengine?hl=ru#section-4
10. Соглашение Google по предоставлению услуг облачных сервисов (дата обращения: 03.12.2020): https://cloud.google.com/kubernetes-engine/sla
11. Описание ключевых особенностей Google App Engine (дата обращения: 03.12.2020): https://cloud.google.com/appengine?hl=ru#section-2
12. Спецификация WebSocket (дата обращения: 03.12.2020): https://tools.ietf.org/html/rfc6455
13. Документация по Google Pub/Sub (дата обращения: 03.12.2020): https://cloud.google.com/pubsub/docs
14. Описание ключевых особенностей Google pub/sub (дата обращения: 03.12.2020): https://cloud.google.com/pubsub?hl=ru
15. Документация по Google Cloud Run (дата обращения: 03.12.2020): https://cloud.google.com/run?hl=ru#section-5
16. Описание ключевых особенностей Google Cloud Run (дата обращения: 03.12.2020): https://cloud.google.com/run?hl=ru
17. Google pub/sub API с указанием применения gRPC (дата обращения: 03.12.2020): https://cloud.google.com/pubsub/docs/reference/service\_apis\_overview
18. Google Cloud Firestore API (дата обращения: 03.12.2020): https://firebase.google.com/docs/firestore/reference/rpc
19. Документация по Google Cloud FireStore (дата обращения: 03.12.2020): https://cloud.google.com/run?hl=ru#section-5
20. Описание ключевых особенностей Google Cloud Firestore (дата обращения: 03.12.2020): https://cloud.google.com/firestore?hl=ru