|  |  |  |  |
| --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  ФГБОУ ВО «Пермский государственный национальный  исследовательский университет» | | | |
|  | | | |
| ОТЧЕТ  по лабораторной работе №2  «Разработка распределенного приложения,  использующего технологию передачи сообщений и сокеты»  по дисциплине «Технологии разработки распределенных приложений» | | | |
| Работу выполнили студенты группы ФИТ-4-НБ 4 курса:  Ботова Карина  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2020 г. |  | | Проверил к.ф.-м.н., доцент кафедры ПМИ  Деменев Алексей Геннадьевич  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2020 г. |
|  | |  | |
| Пермь 2020 | | | |

1. Постановка задачи

***Цель***: изучение возможностей технологии передачи сообщений и сокетов для создания распределенных приложений. Сравнение методов реализации взаимодействия компонент распределенной системы.. ***Формируемая компетенция:*** способность применять на практике теоретические основы и общие принципы разработки распределенных систем; способность использовать на практике стандарты сетевого взаимодействия компонент распределенной системы..

***Организация выполнения работы:*** Каждый студент выполняет индивидуальное задание.

***Требования к выполнению работы:***

Разработать распределенное приложение, в котором:

1. Сервис обмена данными должен выполнять прием данных в нормализованную реляционную БД (например, спроектированную при выполнении входного контроля) из как минимум пять таблиц в 3-й нормальной форме.
2. Должно быть создано приложение, посылающее данные сервису при помощи сокетов и системы очередей сообщений, со свободной лицензией (Apache ActiveMQ, Apache Kafka или RabbitMQ), а при отсутствии такой возможности (соответствующих умений) допустимо использование импортных с бесплатной лицензией для университета в образовательных целях (например, MSMQ).
3. Данные перед передачей должны сжиматься и шифроваться при помощи ключа симметричного шифрования (DES).
4. Ключ симметричного шифрования должен передаваться сервису импорта для выполнения дешифрации данных.
5. При этом ключ симметричного шифрования должен в свою очередь шифроваться при помощи ключа асимметричного шифрования (RSA).
6. Ключ асимметричного шифрования должен генерироваться сервисом импорта и приложению должна передаваться открытая часть ключа.
7. Сервис импорта при получении данных должен импортировать их в БД при помощи механизма, реализованного при выполнении входного контроля.

По окончании выполнения задания каждый студент должен подготовить отчет.

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). Подходы к реализации. Алгоритмы работы.
4. Методы коммуникаций компонентов системы (клиент→сервер, сервер→клиент и т.д.).
5. Структура передаваемых данных.
6. Исходный код.

Распределение баллов за выполнение работы представлено в следующей таблице:

|  |  |
| --- | --- |
| **Критерий оценивания** | **Оценка** |
| Приложение позволяет выполнять прием и передачу данных из ненормализованной БД в нормализованную без модификации данных. | 2 |
| Приложение позволяет передавать информацию с помощью очередей сообщений. | 3 |
| Приложение позволяет передавать информацию с помощью сокетов. | 3 |
| При передаче данных они шифруются с помощью симметричного ключа. | 2 |
| При передаче симметричного ключа шифрования данные шифруются с помощью ассиметричного ключа. | 2 |
| Приложение, написанное студентом, работает в сети без сбоев. | 2 |

1. Описание приложения. Введение в предметную область

Предметная область – студенты университета.

Имеется ненормализованная таблица, содержащая информацию о студентах: номер зачетной книжки, группа, факультет, специальность, кафедра, телефон кафедры, адрес. С помощью приложения можно переслать эти данные с клиента на сервер с использованием протоколов REST, WebSocket или очереди сообщений (Google PubSub).

1. Лицензия

Приложение разработано на языке программирования Go и распространяется с лицензией Creative Commons Attribution 3.0 License (URL: https://creativecommons.org/licenses/by/3.0/ (дата обращения: 25.06.20)), BSD license (URL: https://golang.org/LICENSE (дата обращения: 25.06.20)).

1. Архитектура системы

На рисунке 1 изображена архитектура системы.

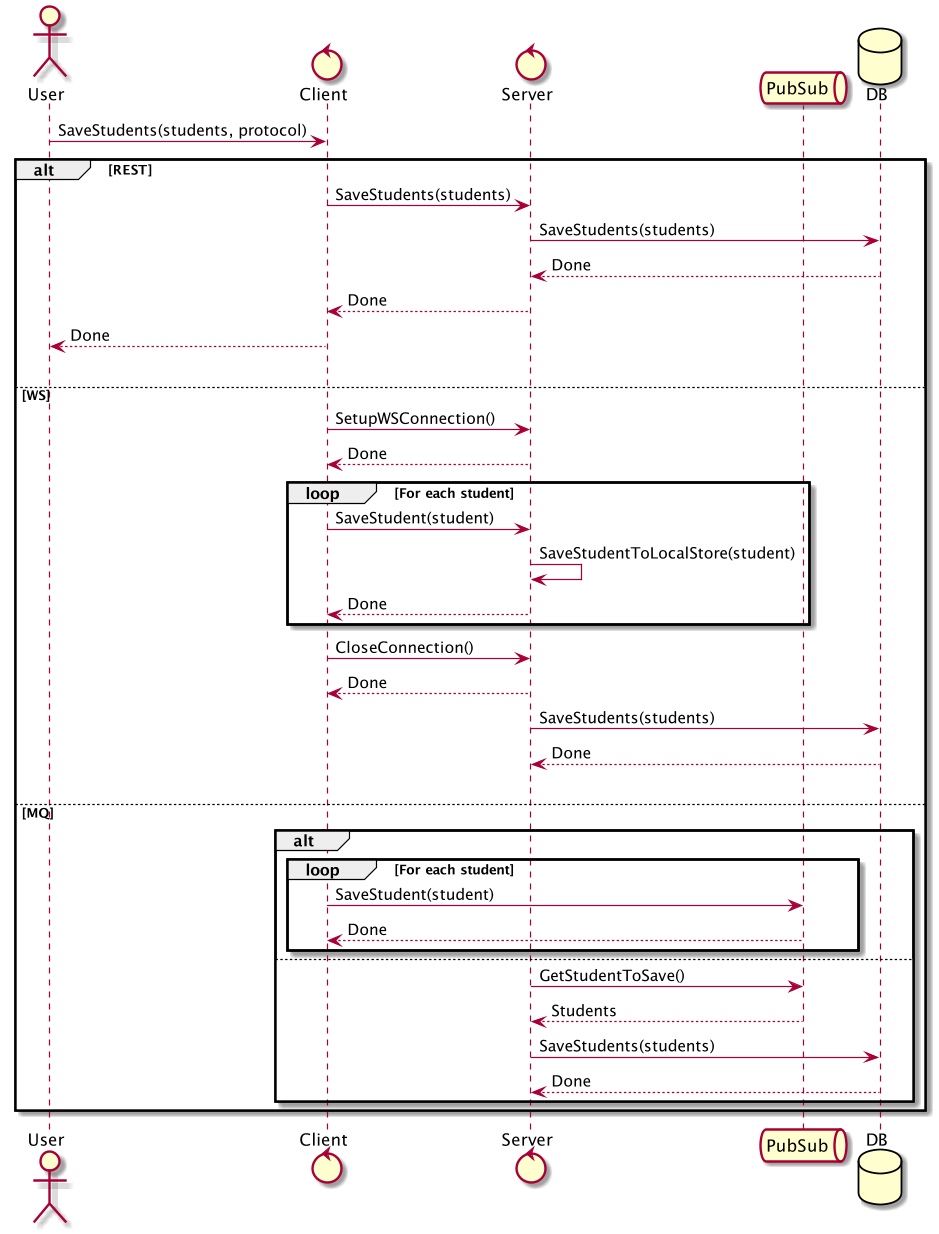


Рисунок 1 – Архитектура системы

**Алгоритм работы системы следующий:**

1. Клиент и сервер представляют собой консольное приложение написанный на языке Golang. Клиент представляет собой web-сервер, принимающий запросы от пользователя и пересылающий данные по выбранному протоколу на сервер.
2. Сервер представляет собой web-сервис, включающий в себя RESTfool web-сервер, web-cocket сервер и consumer для очереди Google PubSub.
3. При отправке данных с помощью протокола REST клиент отправляет HTTP-запрос на сервер. Сервер принимает и обрабатывает запрос, сохраняет данные в БД PostgreSQL.
4. При отправке данных по протоколу web Socket клиент устанавливает Web Socket соединение с сервером и пишет в это соединение данные о студентах. Сервер в свою очередь считывает данные из соединения и сохраняет их в БД PostgreSQL.
5. При отправке данных с помощью очереди клиент пишет данные в очередь, а сервер считывает их и сохраняет в БД PostgreSQL.
6. Архитектура клиента

Архитектура клиента изображена на рисунке 2.

Задачи клиента – получить данные в ненормализованном виде (формат Json) и отправить данные по выбранному протоколу.

Client

User

Server

MQ (очередь сообщений)

Рисунок 2 – Архитектура клиента

6 Архитектура сервера

Архитектура сервера изображена на рисунке 3.

Задача сервера: получить данные по выбранному протоколу и сохранить их в нормализованном виде в PostgreSQL.

Client

Server

DB

MQ (очередь сообщений)

Рисунок 3 – Архитектура сервера

1. Методы коммуникаций компонентов системы

Для коммуникации по протоколам REST и web-socket используется синхронное взаимодействие по модели передачи данных «запрос-ответ». Для коммуникации по протоколу очереди используется асинхронное взаимодействие по модели передачи данных «издатель-подписчик». Передаваемый формат данных - JSON

1. Структура передаваемых данных

[

{

"FIO": "Кашина",

"ZachetBook": "ИТ200013",

"Group":"ИТ111",

"Faculty":"Мехмат",

"Speciality":"ПМИ",

"Cafedry":"ПМИ",

"Tel":"666-666",

"Adress":"Букирева,10"

}

]

9 Исходный код

Реализация приложена рядом с отчетом.

Проект - <https://github.com/KarinaBotova/Normalization>