МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО

«Пермский государственный национальный исследовательский университет»

Механико-математический факультет

Кафедра информационной безопасности и систем связи

**Отчёт**

по лабораторной работе №2 «Разработка распределенного приложения, использующего технологию передачи сообщений и сокеты»

по дисциплине «Технологии разработки распределенных приложений»

Работу выполнил Проверил

студент гр. КМБ-16 доцент кафедры

Якин Никита прикладной математики

Николаевич и информатики

29.10.2020 Деменев Алексей

Геннадьевич

Пермь, 2020 г

**Содержание**

[**Введение** 3](#_Toc54955317)

[**Основная часть** 4](#_Toc54955318)

[**Постановка задачи** 4](#_Toc54955319)

[**Выбор предметной области и инструментов** 4](#_Toc54955320)

[**Сценарии использования** 5](#_Toc54955321)

[**Описание приложения** 6](#_Toc54955322)

[**Заключение** 9](#_Toc54955323)

[**Литература** 10](#_Toc54955324)

# **Введение**

***Цель***: изучение возможностей технологии передачи сообщений и сокетов для создания распределенных приложений. Сравнение методов реализации взаимодействия компонент распределенной системы.

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

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.

# **Основная часть**

## **Постановка задачи**

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

***Таблица 1. Распределение баллов за выполнение работы (***Максимальное количество баллов — 14)***:***

|  |  |
| --- | --- |
| **Требование к заданию** | **Максимальное количество баллов** |
| 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 асимметричного ключа. | 2 |
| Приложение, написанное студентом, работает в сети без сбоев. | 2 |

## **Выбор предметной области и инструментов**

Была выбрана предметная область "Сеансы сети кинотеатров". Предметная область предполагает создание базы данных для хранения информации о сеансах определённой сети кинотеатров, о месте проведения сеанса, информация о фильме, детали сеанса типа: цена, время, тип зала и формат проводимого сеанса.

В качестве СУБД для ненормализованной базы данных была выбрана SQLite [1], которая не требует наличия лицензии при некоммерческом использовании.

В качестве СУБД для нормализованной базы данных была выбрана MySQL [2], которая в случае некоммерческого использования распространяется как под GNU General Public License.

Код программы написан на python 3.7 [3]. В качестве IDE использовался PyCharm Community Edition [4]. PyCharm Community Edition является бесплатной версией, обладающей усеченным набором возможностей.

Для работы с SQLite используется стандартный модуль sqlite3, который является частью Python 3.

Для работы с MySQL используется open-source библиотека под лицензией MIT PyMySQL[5].

Для ассиметричного шифрования (RSA) использовалась библиотека rsa [6].

Для симметричного шифрования (DES) использовалась библиотека PyCryptodome [7].

Для работы с очередями сообщений был выбран RabbitMQ [8], распространяемый под публичной лицензией Mozilla.

Для работы с RabbitMQ используется библиотека pika [9].

## **Сценарии использования**

Приложение может использоваться в следующих сценариях.

1. **Передача данных с помощью сокета**

**Шаги выполнения:**

1. Заполнить файл connection.txt, ввести ip-адрес и порт
2. Пользователь1 запускает приложение передатчик
3. Пользователь2 запускает приложение получатель

**Ожидаемый результат:** Приложения обменяются ключами шифрования с уведомлением и выводом ключей на экран.

**Способ тестирования:**

При успешной передаче данных пользователям будет сообщено, что ключи получены, а сами ключи будут выведены в консоль. (рис. 1-4).

1. Передача данных с помощью очереди сообщений

**Шаги выполнения:**

1. Пользователь1 запускает приложение передатчик
2. Пользователь2 запускает приложение получатель

**Ожидаемый результат:** Приложения автоматически обменяются данными

**Способ тестирования:**

При успешной передаче данных можно пронаблюдать сначала заполнение, а потом освобождение очереди сообщений (рис. 6-7). А также факт заполнения нормализованной базы данных можно проверить в самой СУБД (рис. 8 и 10).

## **Описание приложения**

Приложение разделено на две части: передатчик (transmitter) и приёмник (receiver).

Приложение приёмник формирует пару открытый/закрытый ключ RSA, затем приёмник инициализирует обмен сообщениями с передатчиком, отправляя с помощью сокета свой открытый ключ (рис. 1).



Рис. 1 Формирование и отправка ключа

Приложение передатчик тем временем создаёт прослушивающий сокет и привязывает его к порту, прописанному в файле connection.txt. Получая сообщение от приёмника (рис. 2), он шифрует ключ симметричного шифрования DES и отправляет его приёмнику (рис. 3).



Рис. 2 Получение ключа



Рис. 3 Отправка зашифрованного ключа DES

Приёмник расшифровывает ключ DES с помощью своего закрытого ключа и теперь оба приложения готовы к обмену данными.

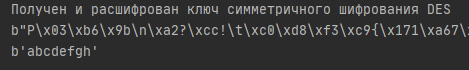


Рис. 4 Получение и расшифровка ключа DES

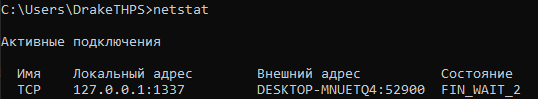


Рис. 5 Использование сокета

**Данное взаимодействие отражает выполнение следующих требований лабораторной работы:**

* Приложение позволяет передавать информацию с помощью сокетов.
* При передаче симметричного ключа шифрования данные шифруются с помощью асимметричного ключа.
* Приложение, написанное студентом, работает в сети без сбоев.

Приложение передатчик открывает соединение с базой данных SQLite по средством обращения к файлу БД, и считывает все данные из ненормализованной таблицы sessions. Затем передатчик открывает соединение с сервисом RabbitMQ и создаёт очередь сообщений database и отправляет в очередь сообщений каждую строку считанной таблицы зашифрованную DES.

Приложение приёмник также открывает соединение с сервисом RabbitMQ и подключается к очереди сообщений database, ожидая появления в нём сообщений, получает сообщения из очереди и расшифровывает их.

Для демонстрации работы очереди сообщений, работа приёмника была приостановлена до получения сообщений (рис. 5).

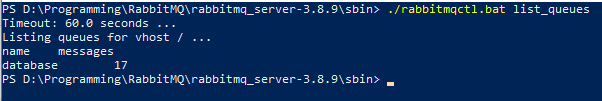


Рис. 6 Заполненная очередь сообщений

После продолжения работы приёмника, он поглощает все сообщения и обрабатывает их.

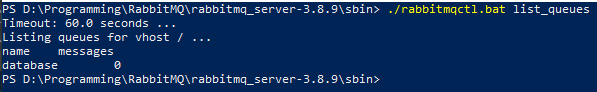


Рис. 7 Очередь после считывания.

Факт обработки и записи данных в нормализованную БД на примере таблице sessions.

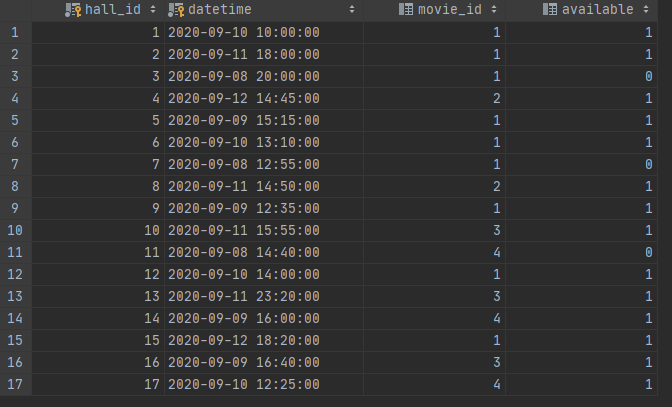


Рис. 8 Таблица sessions

**Данное взаимодействие отражает выполнение следующих требований лабораторной работы:**

* Приложение позволяет выполнять прием и передачу данных из ненормализованной БД в нормализованную без модификации данных.
* Приложение позволяет передавать информацию с помощью очередей сообщений.
* При передаче данных они шифруются с помощью симметричного ключа.

Факт отсутствия модификации данных можно пронаблюдать ниже на рис. 9 изображена таблица sessions из ненормализованной БД, а на рис. 10 результат запроса из всех таблиц нормализованной БД, видно, что данные идентичны, а значит не модифицированы.

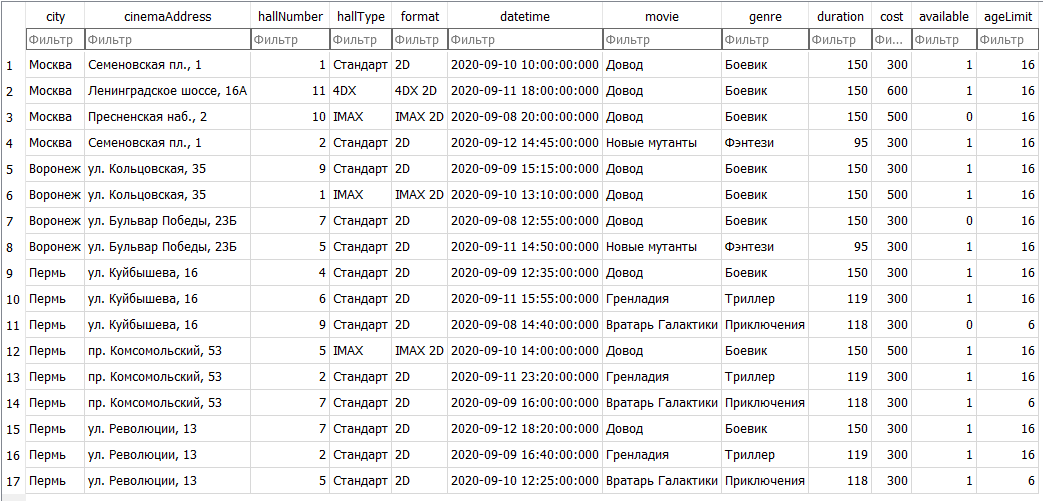


Рис. 9 Таблица sessions

|  |
| --- |
| select c.city, c.cinema\_address, h.hall\_number, ht.type, ht.format, s.datetime, m.name, m.genre, m.duration, ht.cost, s.available, m.age\_limit from sessions s join movies m on s.movie\_id = m.id join halls h on s.hall\_id = h.id join hall\_types ht on h.hall\_type\_id = ht.id join cinemas c on c.id = h.cinema\_id; |

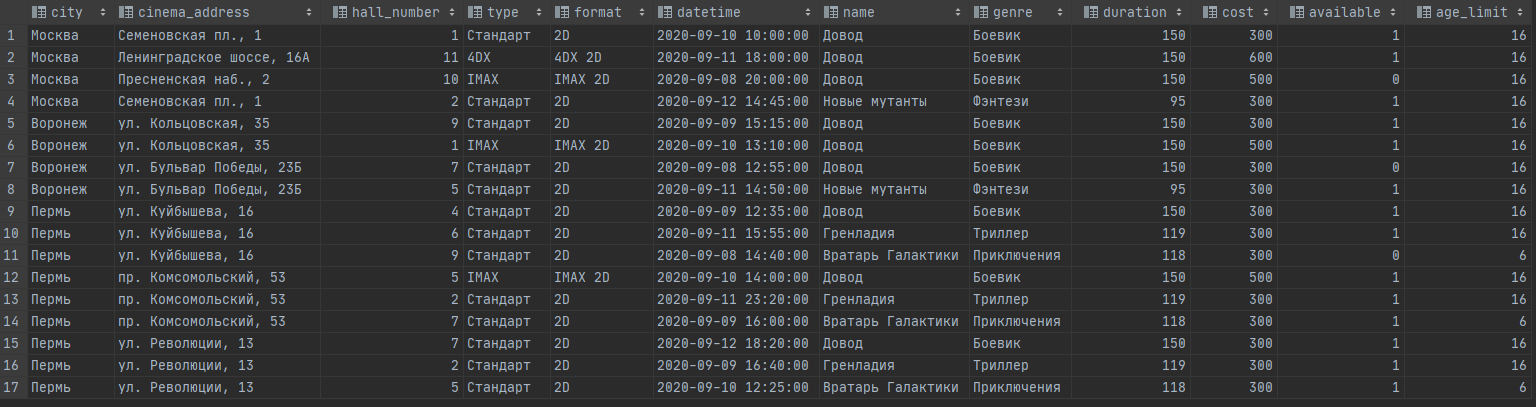


Рис. 10 Результат запроса

# **Заключение**

В ходе работы достигнута цель и выполнены следующие требования:

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

Итого: 14 баллов

# **Литература**

* + 1. SQLite, URL: <https://www.sqlite.org/copyright.html> (Дата обращения 29.10.2020)
    2. MySQL, URL: <https://github.com/mysql/mysql-server/blob/mysql-8.0.3/COPYING> (Дата обращения 29.10.2020)
    3. Лицензия Python 3.7, <https://docs.python.org/3.7/license.html> (Дата обращения 29.10.2020)
    4. PyCharm Community Edition. Распространяется под лицензией Apache 2, URL: <https://www.jetbrains.com/pycharm/features/editions_comparison_matrix.html> (Дата обращения 29.10.2020)
    5. PyMySQL, URL: <https://github.com/PyMySQL/PyMySQL/#license> (Дата обращения 29.10.2020)
    6. Python-RSA, URL: <https://stuvel.eu/python-rsa-doc/licence.html> (Дата обращения 29.10.2020)
    7. Библиотека PyCryptodome, URL: <https://pycryptodome.readthedocs.io/en/latest/src/license.html> (Дата обращения 29.10.2020)
    8. RabbitMQ, URL: <https://www.rabbitmq.com/mpl.html> (Дата обращения 29.10.2020)
    9. Pika, URL: <https://github.com/pika/pika/blob/master/LICENSE> (Дата обращения 29.10.2020)