|  |  |  |  |
| --- | --- | --- | --- |
| 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 ЗАОФИТ-4-2016 НБ 4 курса:  Бродников Вадим  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г. |  | | Проверил к.ф.-м.н., доцент кафедры ПМИ  Деменев Алексей Геннадьевич  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г. |
|  | |  | |
| Пермь 2021 | | | |

1. Постановка задачи

***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.

Каждый студент выполняет индивидуальное задание.

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.

Максимальное количество баллов, которые студент может получить за выполнение работы равно четырнадцати. Распределение баллов за выполнение работы представлено в следующей таблице:

|  |  |
| --- | --- |
| **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 ассиметричного ключа. | 2 |
| Приложение, написанное студентом, работает в сети без сбоев. | 2 |

**2. Описание приложения. Введение в предметную область**

Предметная область сотрудники в IT-компании.

3.Архитектура системы

На рисунке 1 изображена архитектура системы.

Предварительно надо включить сервер

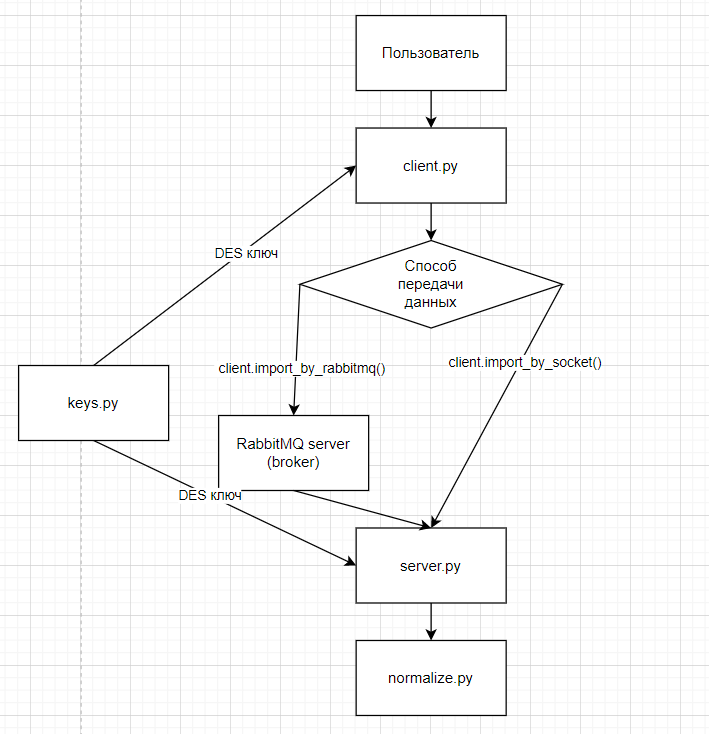


Рисунок 1 – Архитектура системы

**Алгоритм работы системы следующий:**

* 1. Пользователь запускает server.py. Затем пользователю дают по текстовой инструкции выбор способ передачи данных(базы данных), введя через клавиатуру:  
     -“**1”** для передачи через RabbitMQ  
     **-“2”** для передачи через сокеты
  2. При передаче через RabbitMQ сначала отправляется сообщение с размером отправляемого файла на сервер RabbitMQ, затем файл, который мы собираемся отправить, открываем и читаем его битовые данные (по 8 байт) и в то же время создаем поток сообщений, кодируем по DES технологии и отправляем на сервер. Сервер принимает данные
  3. При передаче через сокеты происходит подключение к заранее открытому порту на сервере благодаря server.py и передаются данные также, как и в случае с RabbitMQ, но передаются только зашифрованные данные.
  4. После получения данных с помощью RabbitMQ. server.py “прослушивает” данные. 1 сообщение, это размер передаваемого файла, остальной поток сообщений расшифровывается и используется для создания файла база данных «new\_db.db» в текущей папке. После использует функции нормализации базы из normalize.py, создает новую нормализованную базу данных(normalize\_db.db) в текущей папке, а базу данных, полученную через сервер удаляет(“new\_db.db”) из текущей папки.
  5. После получения данных с помощью сокетов данные дешифровываются и создают файл с базами данными «new\_db.db» в текущей папке. После используется функция нормализации из normalize.py, создается новая нормализованная базы данных(normalize\_db.db) и удаляется “new\_db.db”

1. Описание БД

Для хранения ненормализованной таблицы используется субд sqlite3

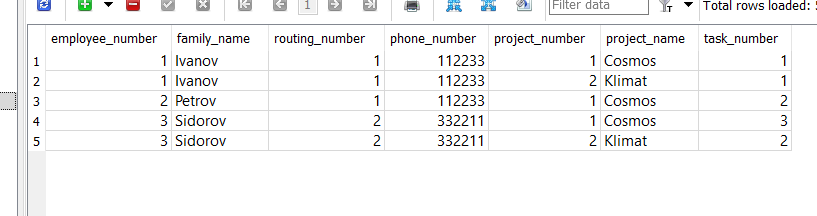


Рисунок 2 – структура ненормализованной базы данных