МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Пермский государственный национальный исследовательский университет”

Лабораторная работа №2

по дисциплине «Технологии разработки распределенных приложений»

Работу выполнила студентка
группы КМБ-16 5 курса
механико-математического
факультета

Кузнецова Александра
Дмитриевна

«12» ноября 2020 г.

Проверил
доцент кафедры
математического обеспечения
вычислительных систем,
кандидат
физико-математических наук

Деменев Алексей Геннадьевич

«__» _____ 2020 г.

г. Пермь, 2020

СОДЕРЖАНИЕ

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.

Организация выполнения работы: Каждый студент выполняет индивидуальное задание.

Требования к выполнению работы:

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.

Технологии и структура приложения

В качестве основы было взято приложение [1], созданное в ходе выполнения входного контроля.

Описание предметной области

База данных была спроектирована для регистратуры Городской клинической больницы №7 г. Перми. Деятельность регистратуры – это сбор и хранение информации о врачах и пациентах, формирование расписаний посещений врачей и хранение информации о прошедших приемах.

Информация, хранимая в базе данных информационной системы регистратуры:

Сведения о врачах:

- ФИО;
- дата рождения;
- пол;
- образование;
- категория;
- должность;
- департамент и специальный департамент, в котором работает врач.

Сведения о пациентах:

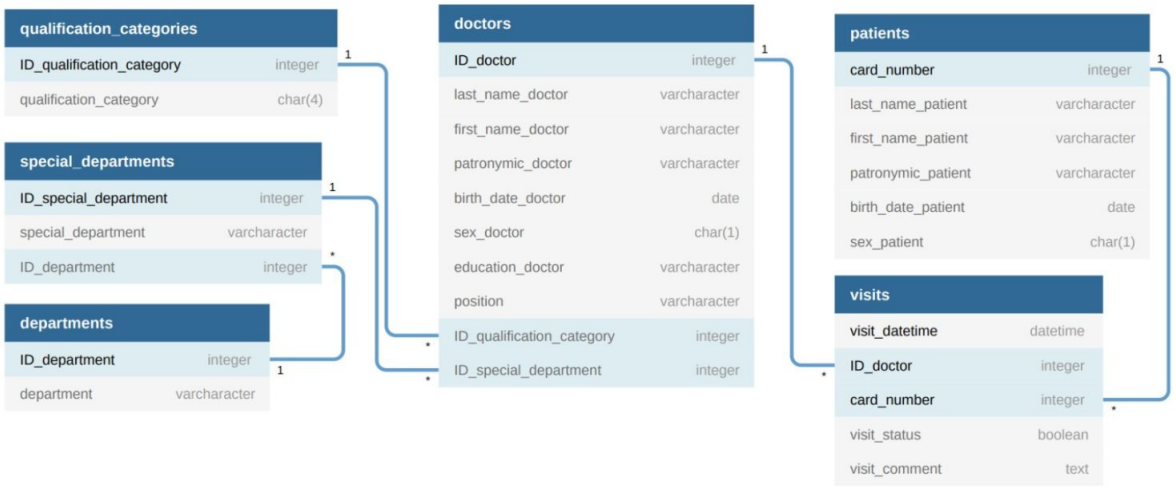
- ФИО;
- дата рождения;
- пол;

Сведения о приемах:

- время приема;
- статус (пациент пришел/не пришел);
- комментарий врача.

Структура не нормализованной БД:

Структура нормализованной БД:



Использованные технологии

Разработка и сборка приложения производилась в среде ОС Fedora 30 [2], приложение совместимо только с семейством ОС Linux.

- Язык - Java 8 [3];
- Среда разработки - IntelliJ IDEA Ultimate 2019.2.4 [4], предоставленная по индивидуальной студенческой лицензии JetBrains [5].
- Настольная СУБД SQlite 3.28.0 [6];
- Корпоративная СУБД PostgreSQL 12.4 [7];
- Фреймворк для автоматизации сборки Maven 3.6.0 [8];
- Брокер сообщений ActiveMQ 5.7.0 [9];
- Дополнительные библиотеки:
    - jooq 3.13 [10] для построения запросов к БД;

Структура приложения, сборка и запуск

Итоговое приложение состоит из двух рабочих модулей, db_sender и db_receiver для отправки и получения данных соответственно, и вспомогательного модуля main, в котором определяются все компоненты и библиотеки, используемые одновременно db_sender и db_receiver.

db_sender получает данные из не нормализованной БД из SQLite и отправляет их db_receiver, который в свою очередь помещает данные в нормализованную БД в PostgteSQL.

Для сборки с помощью maven был создан общий файл модели проекта pom.xml в корне:

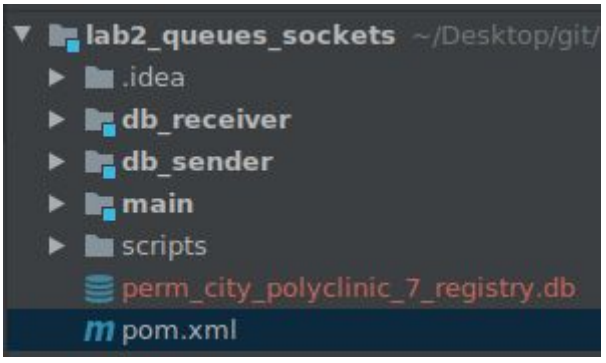


Рис. 1, общий файл pom.xml в корне проекта

Корневой pom.xml содержит в себе определение всех модулей:

```
<groupId>edu.psu</groupId>
<artifactId>lab2_queues_sockets</artifactId>
<packaging>pom</packaging>
<version>1.0-SNAPSHOT</version>
<modules>
   <module>main</module>
   <module>db_sender</module>
   <module>db_receiver</module>
</modules>
```

Для каждого из модулей main, db_sender и db_receiver есть свои собственные файлы моделей проекта pom.xml:

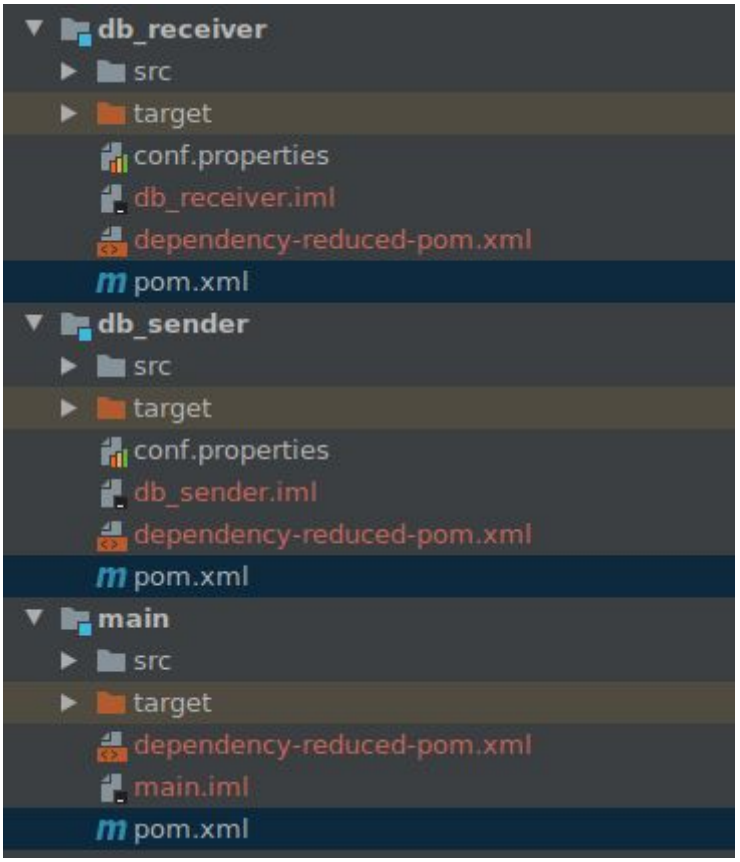


Рис. 2, индивидуальные файлы pom.xml модулей

Сборку приложения необходимо запускать в корневом каталоге проекта с помощью команды `mvn clean install`: команда запускает последовательную сборку всех модулей в том порядке, в котором они указаны в корневом pom.xml.

После сборки проекта появятся два итоговых jar-файла для db_sender и db_receiver соответственно:

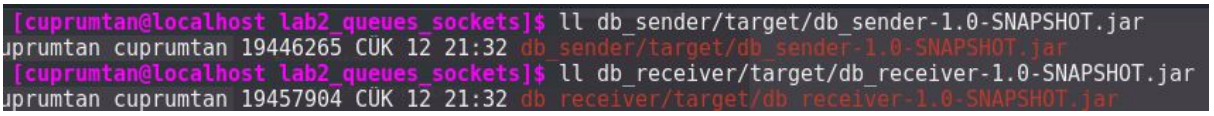


Рис. 3, итоговые jar-файлы

Для модулей db_sender и db_receiver существуют конфигурационные файлы conf.properties, расположеные рядом с jar-файлами:

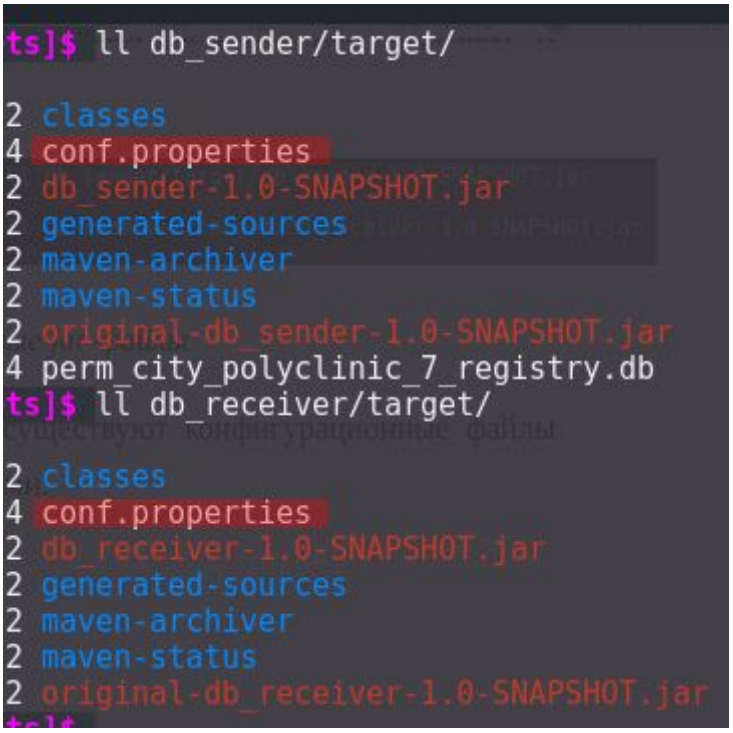


Рис. 4, конфигурационные файлы модулей

Конфигурационные файлы содержат следующую информацию:

db_sender:

- sqlite.jdbc - строка подключения к SQLite;
- activemq.host - IP-адрес хоста, на котором работает ActiveMQ;
- activemq.port - порт, на котором работает ActiveMQ;
- socket.host - IP-адрес хоста, на котором происходит работа через сокеты;
- socket.port - порт, на котором происходит работа через сокеты;

- Tdes.key - ключ шифрования для 3DES;

db_receiver:

- pg.jdbc - строка подключения к PostgreSQL;
- pg.user - пользователь PostgreSQL;
- pg.password - пароль пользователя PostgreSQL;
- activemq.host - IP-адрес хоста, на котором работает ActiveMQ;
- activemq.port - порт, на котором работает ActiveMQ;
- socket.host - IP-адрес хоста, на котором происходит работа через сокеты;
- socket.port - порт, на котором происходит работа через сокеты;

Итоговая последовательность запуска приложения:

1. Запуск ActiveMQ с помощью команды

```
$ACTIVEMQ_HOMEDIR/activemq start
```

2. Запуск db_sender с помощью команд

```
cd .../db_sender/target
java -jar db_sender-1.0-SNAPSHOT.jar
```

3. Запуск db_receiver с помощью команд

```
cd .../db_receiver/target
java -jar db_receiver-1.0-SNAPSHOT.jar
```

Визуально работа приложения видна или через графический интерфейс ActiveMQ:

Рис. 5, графический интерфейс ActiveMQ

Или в консоли, где были запущены модули:

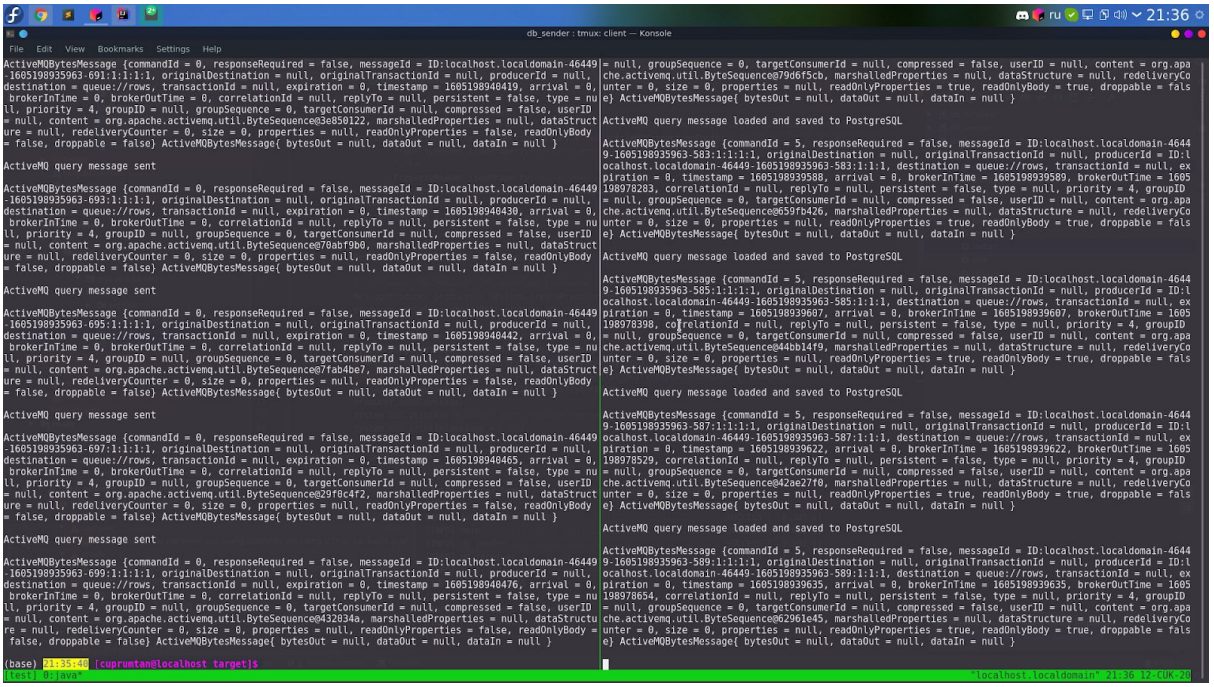


Рис. 6, консольный вывод db_sender и db_receiver

Требования к приложению и их выполнение

Требование 1: Приложение позволяет выполнять прием и передачу данных из ненормализованной БД в нормализованную без модификации данных.

Базы данных в обеих СУБД, SQLite и PostgreSQL, были созданы с настройками по умолчанию. В обеих СУБД по умолчанию используется кодировка UTF-8.en_EN, поэтому проблем с передачей данных строкового типа нет.

В SQLite нет встроенного типа boolean, поэтому поле *visit_status* было создано типа Integer. True и False заполнялись как 1 и 0 соответственно. В базе PostgreSQL поле *visit_status* было создано уже с типом boolean, как и задумывалось при создании модели БД. PostgreSQL обладает встроенным механизмом распознавания целочисленных значений 1 и 0 и перевода их в тип boolean в True и False соответственно. Модуль db_sender передает в db_receiver поле *visit_status* как целочисленное, db_receiver также отправляет в PostgreSQL целочисленное значение и в конечном итоге в БД попадают корректные значения поля *visit_status* без модификации.

Также в SQLite нет встроенного типа для представления времени и даты. Для всех полей типа date и datetime в SQLIte были созданы текстовые поля, а данные были введены заполнены определенным образом:

- “YYYY-MM-DD” для полей типа date;
- “YYYY-MM-DD hh:mm:ss” для полей типа datetime;

Такой формат был выбран не случайно: PostgreSQL обладает встроенным механизмом распознавания строк такого формата и перевода их в типы date и datetime соответственно. Таким образом db_sender получает из SQLite строки, передает из db_receiver, который в свою очередь строки в этом специальном формате отправляет в PostgreSQL. В итоге в нормализованную БД данные типа date и datetime попадают без модификации.

Требование 2: Приложение позволяет передавать информацию с помощью очередей сообщений.

С помощью очередей сообщений происходит передача и получение данных из ненормализованной БД. В качестве брокера сообщений был выбран ActiveMQ.

Работа с ActiveMQ с точки зрения отправки данных описана в следующих классах db_sender:

- QueuesSender;
- SenderMa*n;*

QueuesSender содержит методы:

- pushToQueue(сообщение, пароль) - шифрование сообщения и отправка в очередь ActiveMQ;
- encrypt(сообщение, пароль) - шифрование сообщения.

SenderMain содержит основную логику модуля - в методе main происходит получение списка объектов таблицы ненормализованной БД (объект представляет собой один кортеж этой таблицы) и последовательное помещение каждого объекта в очередь ActiveMQ.

Работа с ActiveMQ с точки зрения получения данных описана в следующих классах db_receiver:

- QueuesReceiver;
- ReceiverMain;

QueuesReceiver содержит методы:

- main() - получает сообщение из ActiveMQ, расшифровывает его и помещает в нормализованную БД;
- decrypt(сообщение, пароль) - расшифрование сообщения.

ReceiverMain организует цикл, в котором происходит ожидание поступления сообщений в ActiveMQ и их последующая обработка методами из QueuesReceiver.

Для ActiveMQ в конфигурационных файлах указаны следующие параметры:

- порт 127.0.0.1 (т.е. локальный запуск);

- стандартный порт 61616;

Проверка отправки и получения сообщений была выполнена с помощью отслеживания работы приложения с помощью графического интерфейса ActiveMQ (рис. 5).

Требование 3: Приложение позволяет передавать информацию с помощью сокетов.

С помощью сокетов передаются ключи шифрования. Для организации передачи публичного ключа асимметричного шифрования в модуле db_receiver создается сокет в классе ReceiverMain. В db_server в свою очередь в классе SenderMain прописан метод подключения к сокету и принятия оттуда сообщений. Работа с сокетами организована через класс стандартной библиотеки java.net.Socket.

Для сокета в конфигурационных файлах указаны следующие параметры:

- порт 127.0.0.1 (т.е. локальный запуск);
- порт 5433;

Проверка работы сокета была выполнена с помощью просмотра статуса заданного порта и процесса, его держащего:

```
# был получен PID процесса, занимающего порт
$ netstat -anp | grep 5433
# PID = 1317

# команда PID
$ ps aux -p 1317
# команда: java -jar db_receiver-1.0-SNAPSHOT.jar
```

Требование 4: При передаче данных они шифруются с помощью симметричного ключа.

В качестве алгоритма симметричного шифрования был выбран 3DES из стандартной библиотеки javax.crypto.

Метод шифрования сообщения encrypt(сообщение, пароль) расположен в модуле db_sender в классе QueuesSender. Метод расшифрования сообщения decrypt(сообщение, пароль) расположен в модуле db_receiver в классе QueuesReceiver. С помощью 3DES шифруются сообщения, передаваемые в ActiveMQ.

Пароль для алгоритма 3DES задается в конфигурационном файле db_sender через параметр Tdes.key.

Требование 5: При передаче симметричного ключа шифрования данные шифруются с помощью асимметричного ключа.

В качестве алгоритма асимметричного шифрования был выбран RSA из стандартной библиотеки javax.crypto. Оба модуля, db_sender и db_receiver, сгенерировали свои пары ключей RSA и обменялись ими через сокет для дальнейшего взаимодействия.

При передаче пароля 3DES db_sender шифрует его с помощью публичного ключа RSA:

```
// метод библиотеки javax.crypto для
шифрования RSA
byte[] desPasswdEncrypted = cipher.doFinal(desKey.getBytes());
// отправка зашифрованного
os.write(desPasswdEncrypted);
os.flush();
```

Требование 6: Приложение, написанное студентом, работает в сети без сбоев.

Для обеспечения бесперебойной работы в сети было сделано следующее:

1. Созданы конфигурационные файлы, где можно менять IP-адреса и порты служб;
2. Создан демон автозапуска activemq для systemd (с помощью специализированного синтаксиса [11]):

```
[Unit]
Description=Apache ActiveMQ
After=network-online.target

[Service]
Type=forking
WorkingDirectory=/opt/activemq/bin
ExecStart=/opt/activemq/bin/activemq start
```

```
ExecStop=/opt/activemq/bin/activemq stop
Restart=on-abort
User=activemq
Group=activemq

[Install]
WantedBy=multi-user.target
```

Вывод

Требование	Статус
Приложение позволяет выполнять прием и передачу данных из ненормализованной БД в нормализованную без модификации данных	Выполнено полностью Передача данных из ненормализованной БД SQLite в нормализованную БД PostgreSQL реализовано с помощью методов, написанных при выполнении входного контроля
Приложение позволяет передавать информацию с помощью очередей сообщений	Выполнено полностью Передача и получение данных происходит с помощью брокера сообщений ActiveMQ
Приложение позволяет передавать информацию с помощью сокетов	Выполнено полностью Передача и получение ключей шифрования происходит с помощью сокета
При передаче данных они шифруются с помощью симметричного ключа	Выполнено полностью Перед помещением сообщения в очередь ActiveMQ оно шифруется с помощью алгоритма 3DES из стандартной библиотеки javax.crypto
При передаче симметричного ключа шифрования данные шифруются с помощью асимметричного ключа	Выполнено полностью Перед передачей пароль шифруется с помощью публичного ключа RSA. Алгоритм RSA из стандартной библиотеки javax.crypto
Приложение, написанное студентом, работает в сети без сбоев	Выполнено полностью Обеспечена возможность замены IP-адресов и портов и отслеживание состояния ActiveMQ

Список источников

1. db_import_export [Электронный ресурс]. URL: https://github.com/cuprumtan/db_import_export (дата обращения 12.11.2020).
2. Fedora project Wiki: Fedora Release 30 [Электронный ресурс]. URL: https://fedoraproject.org/wiki/Releases/30/ (дата обращения 12.11.2020).
3. Java Development Kit 8 Update Release Notes [Электронный ресурс]. URL: https://www.oracle.com/java/technologies/javase/8u-relnotes.html (дата обращения 12.11.2020).
4. Jetbrains Confluence: IntelliJ IDEA 2019.2.4 (192.7142.36 build) Release Notes [Электронный ресурс]. URL: https://confluence.jetbrains.com/display/IDEADEV/IntelliJ+IDEA+2019.2.4+(192.7142.36+build)+Release+Notes (дата обращения 12.11.2020).
5. Jetbrains: Бесплатные лицензии для обучения программированию. Индивидуальные лицензии для студентов и преподавателей [Электронный ресурс]. URL: https://www.jetbrains.com/ru-ru/community/education/#students (дата обращения 01.10.2020).
6. SQlite: SQLite Release 3.28.0 On 2019-04-16 [Электронный ресурс]. URL: https://www.sqlite.org/releaselog/3_28_0.html (дата обращения 01.10.2020).
7. PostgreSQL Release Notes: PostgreSQL 12.4 [Электронный ресурс]. URL: https://www.postgresql.org/docs/release/12.4/ (дата обращения 01.10.2020).
8. Apache Maven Project: Release Notes - Maven 3.6.0 [Электронный ресурс]. URL: https://maven.apache.org/docs/3.6.0/release-notes.html (дата обращения 12.11.2020).
9. Apache ActiveMQ: ActiveMQ 5.7.0 Release [Электронный ресурс]. URL: https://activemq.apache.org/activemq-570-release (дата обращения 12.11.2020).
10. jOOQ: The history of jOOQ. From 2009 to 2020: Version 3.13.0 - February 2020 [Электронный ресурс]. URL: https://www.jooq.org/notes#3.13.0 (дата обращения 12.11.2020).
11. Freedesktop Wiki: systemd.service [Электронный ресурс]. URL: https://www.freedesktop.org/software/systemd/man/systemd.service.html (дата обращения 12.11.2020).