

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 №4 «Разработка распределенного приложения для
локальной сети»
по дисциплине «Технологии разработки распределенных приложений»

Работу выполнили
студенты гр. КМБ-16
Кузнецова Александра
Дмитриевна
Мартышенко Сергей
Вадимович
Пепеляев Павел Михайлович
«3» декабря 2020

Проверил
доцент кафедры
прикладной математики
и информатики,
к.ф.-м.н., доц.
Деменев Алексей
Геннадьевич

Пермь, 2020 г

Содержание

Постановка задачи	4
1. Описание функциональных характеристик системы.....	5
1.1 Назначение системы	5
1.2 Описание компонентов системы.....	8
1.2.1 Резервирование данных	8
1.2.2 Хранение данных.....	9
1.2.3 Диспетчер.....	10
1.2.4 Веб-сервер и его модули	10
1.2.5 Клиент	11
1.3 Методы коммуникации и способы передачи данных компонентов системы	12
1.3.1 Коммуникации между клиентом и сервером.....	12
1.3.2 Коммуникации между компонентами серверной части	12
1.3.3 Коммуникации между модулями веб-сервера	13
1.3.4 Структура и протоколы, используемые при передаче сообщений.....	14
1.4 Средства отказоустойчивости и масштабируемости Системы	15
1.5 Объектно-ориентированный доступ к БД.....	15
2 Соблюдение требований	17
2.1 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.....	17
2.2 Архитектура системы является оптимальной для заданных при разработке критериев. В отчете присутствует обоснование выбора данного типа архитектуры.....	18
2.3 Приложение обеспечивает параллельную работу нескольких клиентов и серверов, в том числе на одном компьютере. Серверы распределенной системы выполняют различные функции.....	18
2.4 Приложение является масштабируемым, позволяет добавлять новых участников взаимодействия без переписывания кода и перезапуска приложений.....	19
2.5 Существует возможность динамического реконфигурирования системы.....	22
2.6 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.	24
2.7 Система является отказоустойчивой. В случае если один и/или несколько компонент системы аварийно завершают свою работу.	25
2.8 Распределенное приложение продолжает работать и в случае, если после аварийного завершения некоторого компонента, он восстановлен на другом узле вычислительной сети.	26

2.9	Отчет содержит подробное описание архитектуры каждого компонента распределенного приложения.....	26
2.10	В отчете описана структура передаваемых данных, формат сообщений и вид протокола, используемого для этого.	26
2.11	В отчете представлено описание способа передачи сообщений при коммуникации компонентов распределенной системы с обоснованием.....	26
	Список источников.....	28

Постановка задачи

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

Проверяемые компетенции: способность работы с информацией из различных источников, включая сетевые ресурсы сети Интернет, для решения профессиональных задач; способность применять на практике теоретические основы и общие принципы разработки распределенных систем; уверенное знание теоретических и практических основ построения распределенных баз данных; способность использовать на практике стандарты сетевого взаимодействия компонент распределенной системы.

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.

Лабораторная работа выполнялась в группе из трех человек: Кузнецова А.Д., Мартышенко С.В., Пепеляев П.М.

1. Описание функциональных характеристик системы

1.1 Назначение системы

Разработанная информационная система (далее Система) является информационной системой клиент-серверного типа для хранения и предоставления конечному количеству сотрудников Баука информации о клиентах Банка на основе информации из Единого государственного реестра юридических лиц (ЕГРЮЛ) [1]. Деятельность Системы направлена на достижение следующих целей:

- создание единого сервиса для сотрудников Банка при помощи информационного взаимодействия со свободными данными из ЕГРЮЛ [2];
- получение информации о зарегистрированных в ЕГРЮЛ видах деятельности клиентов Банка в структурированном и удобном для чтения виде;
- сохранность собранных о клиентах данных.

Для достижения поставленных целей решаются следующие задачи:

- создание отказоустойчивой и расширяемой инфраструктуры;
- создание многопользовательского клиент-серверного приложения в локальной сети Банка, имеющего следующие характеристики:
 - создание графического пользовательского интерфейса;
 - организация взаимодействия с АРІ-ФНС [3];
 - автоматизация проверки формата введенного номера ОГРН [4];
 - автоматизация процесса получения документов из ЕГРЮЛ о клиентах по номеру ОГРН;
 - автоматизация процесса синтаксического анализа полученных документов с целью определения видов деятельности;

Для выполнения данной лабораторной работы был получен доступ к АРІ-ФНС по бесплатному тарифу «АРІ Старт» [5].

Функциональная структура Системы представлена на рисунке 1:

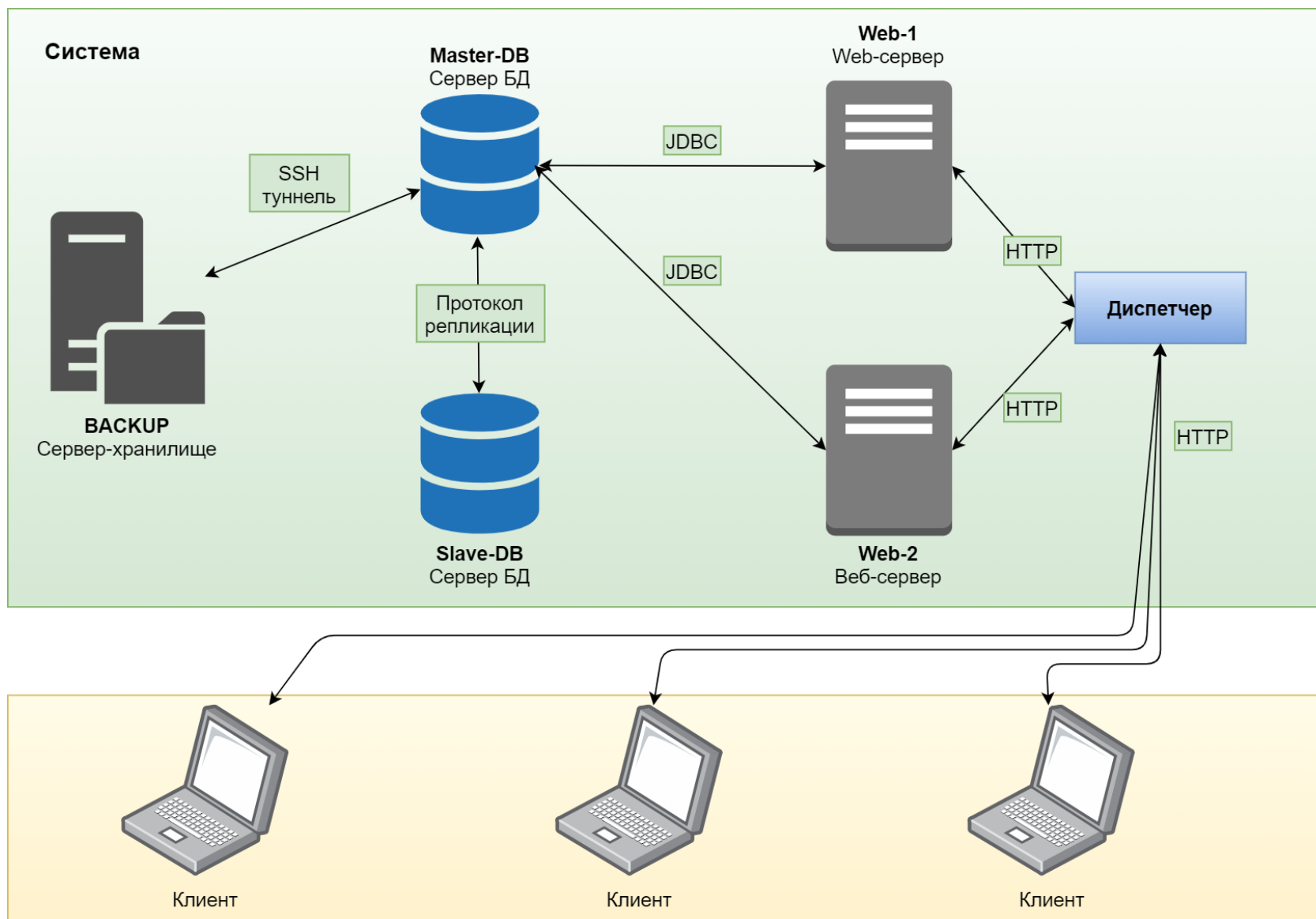


Рис. 1, функциональная схема Системы

Рисунок 1 включает в себя следующие компоненты:

1. BACKUP (сервер-хранилище) – компонент, предназначенный для хранения резервных копий базы данных.
2. Master-DB (сервер баз данных) – компонент, предназначенный для функционирования основного экземпляра базы данных. В этом компоненте происходят все изменения информации, хранящейся в базе данных. Компонент доступен для чтения, записи, модификации, удаления данных.
3. Slave-DB (сервер баз данных) – компонент, предназначенный для функционирования реплицирующего экземпляра базы данных. В этом компоненте все изменения информации появляются асинхронно после свершения операций на основном экземпляре. Компонент доступен только для чтения.
4. Диспетчер – компонент, служащий в качестве надстройки над веб-серверами и отвечающий за распределение пользовательской нагрузки между ними.
5. Web-1 (веб-сервер) - компонент, содержащий основные системные функции. В структуру входит следующий состав функциональных модулей:
 - 5.1. Модуль взаимодействия с данными – предназначен для взаимодействия с ЕГРЮЛ и локальной базой данных.
 - 5.2. Модуль синтаксического анализа документов – предназначен для получения необходимой о клиенте информации из документов ЕГРЮЛ.
 - 5.3. Модуль «Графический интерфейс» - предназначен для реализации взаимодействия с графической оболочкой приложения.
6. Web-2 (веб-сервер) – компонент, дублирующий содержание и поведение компонента Web-1.
7. Клиент – компонент, предоставляющий конечным пользователям графическую оболочку приложения.

1.2 Описание компонентов системы

1.2.1 Резервирование данных

Хранилище данных BACKUP Системы обеспечивает хранение всей необходимой информации для восстановления функционирования серверов баз данных Системы. Хранилище данных в Системе спроектировано с учетом целей системы, определенных в пункте 1.1:

- Высокая скорость передачи данных;
- Полнота и достоверность передаваемых данных.

В качестве операционной системы для хранилища данных выбрана ОС Centos 8 в минимальной редакции [6] и со стандартной файловой системой xfs [7] – минимальная редакция легковесна и обладает всем необходимым для хранилища данных функционалом, стандартная файловая система xfs является быстрой и при этом минимизирует риски, связанные с ошибками совместимости. В качестве инструмента резервирования и хранения базы данных был выбран barman 2.15 [8], так как инструмент позволяет снимать физические резервные копии и проверять их целостность за счет подсчета контрольных сумм.

barman имеет свой конфигурационный файл */etc/barman.conf* со своим определенным синтаксисом [23]. Для настройки резервного копирования необходимо настроить связь только с Master-DB:

```
[master-db-egrul]
description = "Master DB Server"
ssh_command = sshpass -p postgres ssh postgres@10.135.0.3 -o BatchMode=no
conninfo = host=10.135.0.5 user=postgres password=postgres
dbname=egrul_info
backup_method = rsync
retention_policy_mode = auto
retention_policy = RECOVERY WINDOW OF 7 days
wal_retention_policy = main
archiver = on
streaming_archiver=off
```

barman отдельно сохраняет копии файлов журналов СУБД и резервные копии в каталогах */var/lib/barman/master-db-egrul/wals* и */var/lib/barman/master-db-egrul/base* соответственно:

```
[barman@trrp-centos-fs wals]$ pwd
/var/lib/barman/master-db-egrul/wals
[barman@trrp-centos-fs wals]$ ll
total 4
drwxr-xr-x. 2 barman barman 38 Dec 10 17:59 0000000100000000
-rw-rw-r--. 1 barman barman 55 Dec 10 17:59 xlog.db
```

Рис. 2, содержимое каталога для копий журналов


```
[barman@trrp-centos-fs base]$ pwd
/var/lib/barman/master-db-egrul/base
[barman@trrp-centos-fs base]$ ll
total 0
drwxrwxr-x. 3 barman barman 37 Dec 10 17:34 20201210T173403
drwxrwxr-x. 3 barman barman 37 Dec 10 17:48 20201210T174836
[barman@trrp-centos-fs base]$
```

Рис. 3, содержимое каталога резервных копий

1.2.2 Хранение данных

Машины для хранения данных в Системе построены на операционной системе Centos 8 также в минимальной редакции, само хранение данных на основе современной свободной объектно-реляционной СУБД PostgreSQL 12 [9] с активированной технологией репликации [10]. Состав данных, подлежащих сбору и хранению в СУБД:

- Номер ОГРН клиента;
- Фамилия, имя и отчество клиента;
- Информация о видах его деятельности:
 - Код деятельности;
 - Наименование деятельности.

Схема данных, созданная с помощью бесплатного сервиса dbdiagram [22], представлена на рисунке 4:

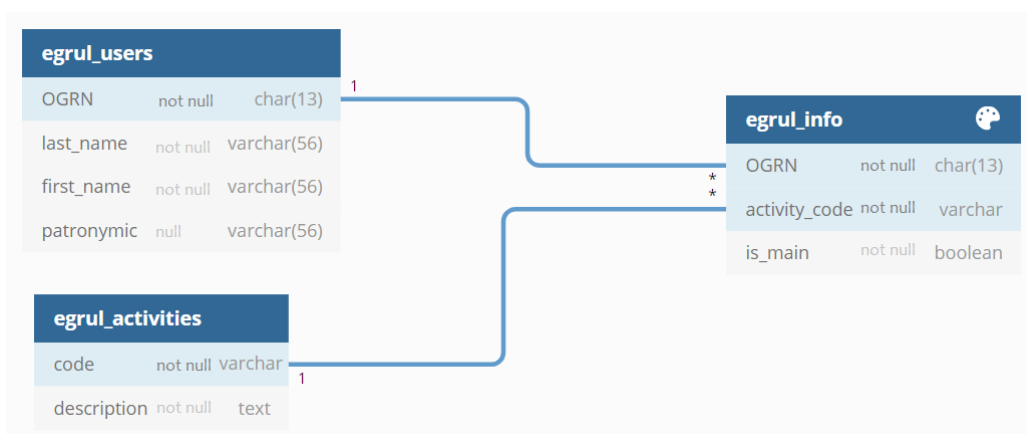


Рис. 4, схема данных Системы

Хранилище данных выполняет обеспечение следующих общих требований:

- соответствие информационных ресурсов Системы принятым отраслевым национальным и международным классификаторам и нормативным актам (для полей, хранящих ОГРН, задано ограничение формата);

- использование форматов представления данных, исключающие дублирование и ввод избыточной информации (созданы первичные ключи, внешние ключи);
- поддержание полноты информации для обеспечения достижения целей создания Системы.

1.2.3 Диспетчер

Компонент «Диспетчер» представляет собой надстройку над веб-серверами и отвечает за балансирование нагрузки на веб-серверы, также повышает отказоустойчивость Системы. В качестве операционной системы сервера-диспетчера выбрана Centos 8 в минимальной редакции, для исполнения функционала диспетчера выбран сервер nginx [11]. nginx является наиболее подходящим вариантом диспетчера для Системы, так как удовлетворяет всем поставленным в пункте 1.1 целям с минимальными затратами:

- Простое управление нагрузкой;
- Предоставляет средства отказоустойчивости.

1.2.4 Веб-сервер и его модули

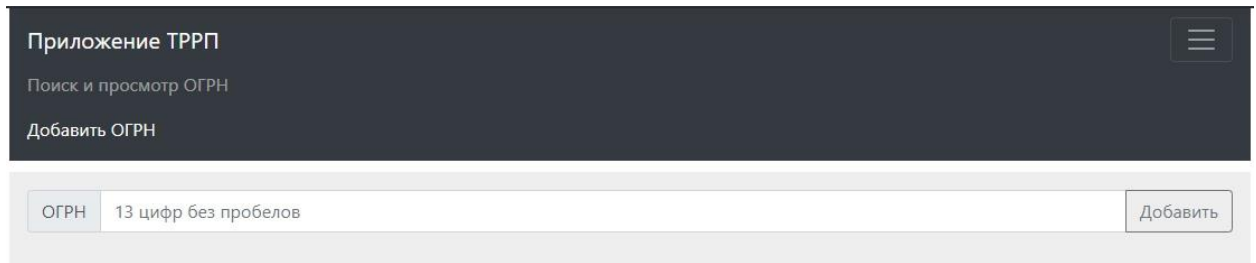
Компонент «Веб-сервер» состоит из двух серверных машин, на каждой из которых расположены и работают экземпляры серверной части приложения. В качестве операционной системы сервера выбрана Centos 8 в минимальной редакции. Распределение нагрузки между машинами происходит с помощью диспетчера. Веб-сервер состоит из трех модулей:

- Модуль взаимодействия с данными – предназначен для взаимодействия с ЕГРЮЛ и локальной базой данных. Модуль написан на языке Java 11 [12]. Взаимодействие с базой данных осуществляется за счет свободной библиотеки jOOQ 3.14.4 [13], предоставляющей функционал для объектно-реляционного отображения данных.
- Модуль синтаксического анализа документов – предназначен для получения необходимой о клиенте информации из документов ЕГРЮЛ. Взаимодействие с API-ФНС происходит с помощью стандартных библиотек языка Python через токен, выданный при регистрации в программе предоставления доступа к API. Модуль реализован на языке Python 3.8 [14] с использованием дополнительных
- Модуль «Графический интерфейс» - предназначен для взаимодействия с графической оболочкой приложения. Написан на языке Java 11 с использованием фреймворка Spring Framework 5.2.7 [17].

1.2.5 Клиент

Клиентский компонент приложения написан на языке Java 11 и является кроссплатформенным, так как клиент представляет собой веб-интерфейс. Графический интерфейс реализован с использованием фреймворка Bootstrap 4. Возможности интерфейса показаны ниже:

Стартовая страница, ввод номера ОГРН:



Приложение ТРРП

Поиск и просмотр ОГРН

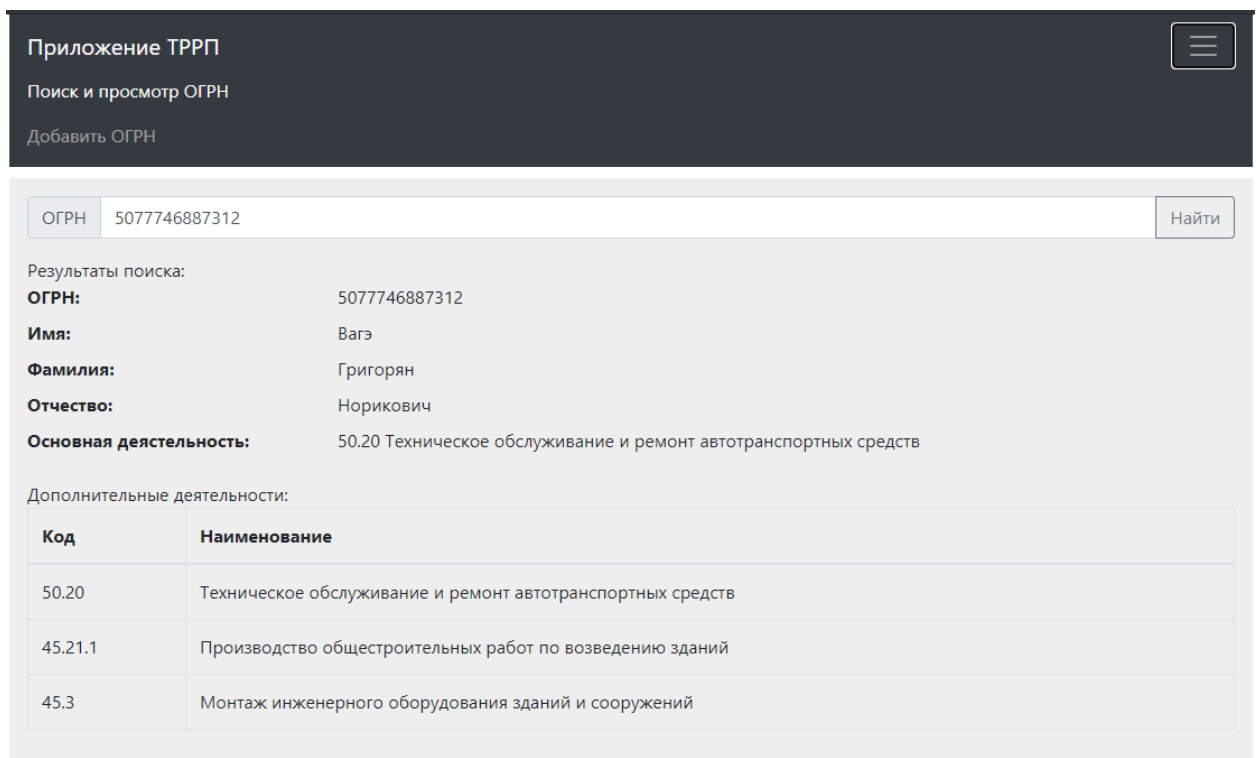
Добавить ОГРН

ОГРН 13 цифр без пробелов

Добавить

Рис. 5, стартовая страница Системы

Вывод информации по введенному ОГРН:



Приложение ТРРП

Поиск и просмотр ОГРН

Добавить ОГРН

ОГРН 5077746887312

Найти

Результаты поиска:

ОГРН: 5077746887312

Имя: Вагэ

Фамилия: Григорян

Отчество: Норикович

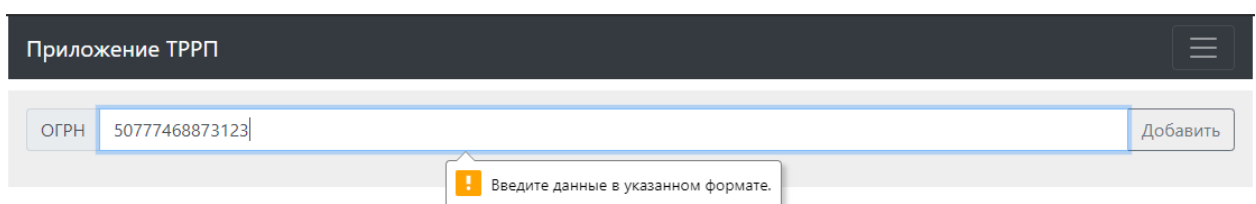
Основная деятельность: 50.20 Техническое обслуживание и ремонт автотранспортных средств

Дополнительные деятельности:

Код	Наименование
50.20	Техническое обслуживание и ремонт автотранспортных средств
45.21.1	Производство общестроительных работ по возведению зданий
45.3	Монтаж инженерного оборудования зданий и сооружений

Рис. 6, вывод информации в Системе

Ошибка ввода ОГРН:



Приложение ТРРП

ОГРН 50777468873123

Добавить

Введите данные в указанном формате.

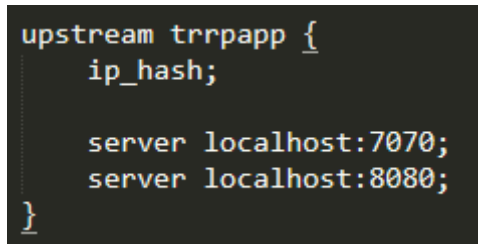
Рис. 7, ошибка ввода ОГРН в Системе

1.3 Методы коммуникации и способы передачи данных компонентов системы

1.3.1 Коммуникации между клиентом и сервером

Коммуникации между клиентом и сервером (конкретно между клиентом и компонентом «Диспетчер») организованы по протоколу HTTP. В конфигурации диспетчера указаны IP адреса веб-серверов. Клиент использует только 1 IP адрес для обращения к серверной части – это IP адрес диспетчера. Диспетчер принимает входящее соединение от клиента и перенаправляет его на один из веб-серверов, описанных в конфигурации. Перенаправление осуществляется по принципу RoundRobin, т.е. диспетчер каждый новый запрос от уникального IP адреса отправляет по кругу к следующему веб-серверу, если предыдущий веб-сервер был последним в списке, то запрос перенаправляется на веб-сервер, который указан первым. Запросы от одного и того же клиента (от одного и того же IP адреса) диспетчер направляет на один и тот же сервер, но если он станет недоступен, то запрос будет перенаправлен на другой, работающий веб-сервер. После того как сервер полностью обработает запрос, ответ отсылается напрямую клиенту.

Пример настройки списка веб-серверов в конфигурационном файле диспетчера:



```
upstream trrpapp {  
    ip_hash;  
  
    server localhost:7070;  
    server localhost:8080;  
}
```

Рис. 8, список веб-серверов в конфигурационном файле диспетчера

1.3.2 Коммуникации между компонентами серверной части

С точки зрения серверной части в Системе существуют следующие коммуникации:

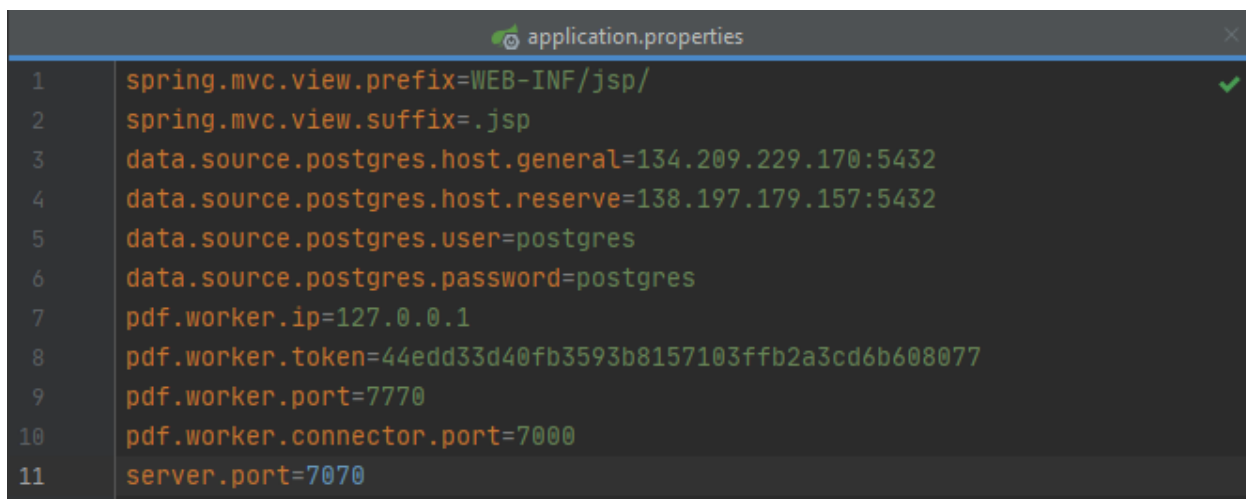
- Коммуникации между Master-DB и Slave-DB – передача данных организована по протоколу потоковой репликации. Передача по протоколу репликации была настроена следующим образом: передача данных происходит асинхронно и двунаправленно. Асинхронность позволяет получить согласованные данные для репликационного сервера, двунаправленность позволяет балансировать нагрузку от входящего потока запросов.
- Коммуникации между Master-DB и BACKUP – передача данных организована по протоколу SSH [18] с помощью утилиты rsync

[19]. Так как передача данных связана с работой протокола репликации через утилиту `barman`, способ передачи тот же – асинхронный. При этом используется также двунаправленная передача данных: утилита `barman` отправляет Master-DB код завершения операции после копирования данных на BACKUP для отслеживания ошибок.

- Коммуникации между Диспетчером и Web-1 и Web-2 – коммуникации организованы по протоколу HTTP [20]. Коммуникация асинхронная и двунаправленная: `nginx` принимает входящий от клиентов трафик и распределяет его между двумя веб-серверами, получая от них ответ о статусе связи.
- Коммуникации между веб-серверами и сервером БД – коммуникации организованы с помощью драйвера JDBC [21]. Канал JDBC – асинхронный и двунаправленный.

1.3.3 Коммуникации между модулями веб-сервера

Коммуникации между модулями веб-сервера реализованы с помощью синхронных двунаправленных сокетов. После запуска Java-приложения, оно автоматически запускает на выполнение `python`-скрипт, предназначенный для обработки pdf-документов, полученных через API-ФНС. Необходимые настройки сокетов задаются в конфигурационном файле `application.properties`:



```
1 spring.mvc.view.prefix=WEB-INF/jsp/
2 spring.mvc.view.suffix=.jsp
3 data.source.postgres.host.general=134.209.229.170:5432
4 data.source.postgres.host.reserve=138.197.179.157:5432
5 data.source.postgres.user=postgres
6 data.source.postgres.password=postgres
7 pdf.worker.ip=127.0.0.1
8 pdf.worker.token=44edd33d40fb3593b8157103ffb2a3cd6b608077
9 pdf.worker.port=7770
10 pdf.worker.connector.port=7000
11 server.port=7070
```

Рис. 9, конфигурационный файл

В этом файле указаны порты, по которым `python`-скрипт и Java-приложение могут взаимодействовать. Также указывается IP-адрес на котором будет работать `python`-скрипт. Для взаимодействия используются 2 порта, в конфигурационном файле они обозначены как `pdf.worker.port` и `pdf.worker.connector.port`.

pdf.worker.port - порт, на котором работает python-модуль.
pdf.worker.connector.port - порт на стороне Java-приложения, к данному порту обращается python-скрипт для того, чтобы установить соединение. После запуска python-скрипт устанавливает соединение с Java-приложением с помощью сокетов и ожидает входящего соединения. Java-приложение обращается к скрипту каждый раз, когда от пользователя поступает запрос на добавление новых данных в систему. В скрипт передаётся введённый пользователем ОГРН, далее с помощью API-ФНС осуществляется попытка загрузки данных из ЕГРЮЛ. Если документ найден, то он скачивается и обрабатывается. Полученные данные передаются обратно в Java-приложение в формате json. Если по заданному ОГРН в ЕГРЮЛ не найдено данных, то в Java-приложение возвращается пустая строка. И в том, и в другом случае Java-приложение продолжает работу в штатном режиме и обрабатывает полученный от скрипта результат.

1.3.4 Структура и протоколы, используемые при передаче сообщений

Клиенты и веб-серверы взаимодействуют по протоколу HTTP. Пользователь пишет в адресную строку браузера адреса, а также выполняет действия на web-страницах приложения. В результате чего на веб-сервер отправляются стандартные HTTP запросы GET и POST. Веб-сервер возвращает результат в виде HTML документа.

Общая структура запроса клиента:

http://<адрес диспетчера>/<требуемая страница>

Также клиент (браузер) пользователя автоматически формирует HTTP заголовки и передаёт данные с форм (тег form), если это POST запрос.

Общая структура ответа сервера:

```
<!DOCTYPE HTML>
<html>
  <head>
    [мета-данные страницы]
  </head>
  <body>
    [тело страницы]
  </body>
</html>
```

Веб-сервер и модуль обработки документов взаимодействуют с помощью сокетов, данные при этом передаются в виде текстовых строк.

Типичный запрос веб-сервера содержит лишь ОГРН, передаваемая строка выглядит следующим образом:

“<ОГРН из 13 цифр>”

Модуль отвечает пустой строкой, если по данному ОГРН не удалось ничего найти. В обратном случае формирует ответ в формате json. Структура ответа в формате json выглядит следующим образом:

```
{
  "lastName": "<фамилия>",
  "firstName": "<имя>",
  "patronymic": "<имя>",
  "activities": [
    {
      "activityCode": "<код деятельности>",
      "description": "<описание деятельности>",
      "isMain": "<0 или 1 - признак основного вида деятельности>"
    },
    ...
  ]
}
```

1.4 Средства отказоустойчивости и масштабируемости Системы

Отказоустойчивость Системы реализована за счет введения отдельного сервера для хранения резервных копий, дублирующих серверов для сервера баз данных и веб-сервера. Вся необходимая для настройки информация вынесена в конфигурационные файлы.

Система легко масштабируется: существует диспетчер-балансировщик нагрузки на веб-серверы, настраиваемый с помощью конфигурационного файла. В этом конфигурационном файле можно задать адреса добавляемых веб-серверов. Также возможно простое добавление новых серверов в кластер СУБД через конфигурационные файлы самой СУБД и утилиты barman. Для добавления новых клиентов не требуется перезапуска Системы или изменения ее инфраструктуры или кода приложений, достаточно запустить новую сессию клиентского приложения через браузер.

1.5 Объектно-ориентированный доступ к БД

Для взаимодействия с базой данных используется библиотека Jooq. Jooq - это легкая библиотека программного обеспечения для отображения базы

данных в Java, которая реализует шаблон активной записи. Таким образом, Jooq является аналогом ADO – технологии доступа к данным используемой для разработки приложений на платформе Microsoft.

С помощью Jooq в приложении генерируются все необходимые классы, отображающие данные в БД. Также Jooq генерирует специальные DAO (Data Access Object), которые предоставляют интерфейс для взаимодействия с сущностями БД как с объектами.

2 Соблюдение требований

2.1 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.

Требование выполнено полностью. Особенности предметной области, определенной в пункте 1.1, и их учет следующие:

- создание единого сервиса для сотрудников Банка при помощи информационного взаимодействия со свободными данными из ЕГРЮЛ – было создано многопользовательское клиент-серверное приложение, использующее в своей работе официальный API сервиса – API-ФНС;
- получение информации о зарегистрированных в ЕГРЮЛ видах деятельности клиентов Банка в структурированном и удобном для чтения виде:

В клиентском приложении реализован графический интерфейс. Также работа клиентского приложения организована максимально просто для конечных пользователей – не требуется установка дополнительных инструментов на клиентские рабочие станции, для работы с приложением достаточно наличия стандартных средств отображения веб-страниц (например, браузер).

- сохранность собранных о клиентах данных:

Сохранность и целостность данных в момент работы приложения обеспечивается штатными средствами СУБД – была выбрана объектно-реляционная СУБД PostgreSQL, соблюдающая во время своей работы принципы ACID. Сохранность и целостность данных по завершению работы с ними организована с помощью выделения отдельного сервера-хранилища, на котором организован прием и хранение резервных копий базы данных и журналов через утилиту barman.

База данных распределенная – есть два сервера СУБД с настроенной между ними асинхронной репликацией в реальном времени. Репликация в СУБД PostgreSQL работает по шаблону Master-Slave, возможностей данного шаблона достаточно для предметной области: Система не подразумевает распределенные вычисления, необходимо только надежное хранение данных и их быстрое получение – шаблон Master-Slave позволяет равномерно распределять нагрузку от входящего потока пользовательских запросов на

выборку данных, в случае падения Master возможно быстрое и простое переключение на другую машину.

2.2 Архитектура системы является оптимальной для заданных при разработке критериев. В отчете присутствует обоснование выбора данного типа архитектуры.

Требование выполнено полностью.

Критерии, заданные при разработке:

- отказоустойчивость и масштабируемость;
- создание графического пользовательского интерфейса;
- организация взаимодействия с API-ФНС;
- автоматизация проверки формата введенного номера ОГРН;
- автоматизация процесса получения документов из ЕГРЮЛ о клиентах по номеру ОГРН;
- автоматизация процесса синтаксического анализа полученных документов с целью определения видов деятельности;

Архитектура Системы является оптимальной, так как для реализации приложения были выбраны наиболее подходящие языки программирования в достаточном и не избыточном количестве: синтаксический анализ документов был реализован как отдельный модуль на наиболее подходящем для этой задачи языке – на языке Python. Этот язык определен как самый подходящий по причине того, что Python имеет множество бесплатных простых и производительных библиотек для работы с текстом и документами. Для реализации модуля была выбрана библиотека rumpdf. Остальные задачи из критериев были реализованы в отдельном модуле на языке Java, так как Java является одним из наиболее подходящих языков для реализации бизнес-логики.

Обоснование выбора данного типа архитектуры также приведено в разделах 1.1 и 1.2.

2.3 Приложение обеспечивает параллельную работу нескольких клиентов и серверов, в том числе на одном компьютере. Серверы распределенной системы выполняют различные функции.

Требование выполнено полностью. Система была создана как многопользовательская, в том числе возможен запуск нескольких клиентов с одного адреса. Входящий на серверную часть трафик поступает асинхронно, поэтому возможен одновременный запуск клиентов и работа с одной машины:

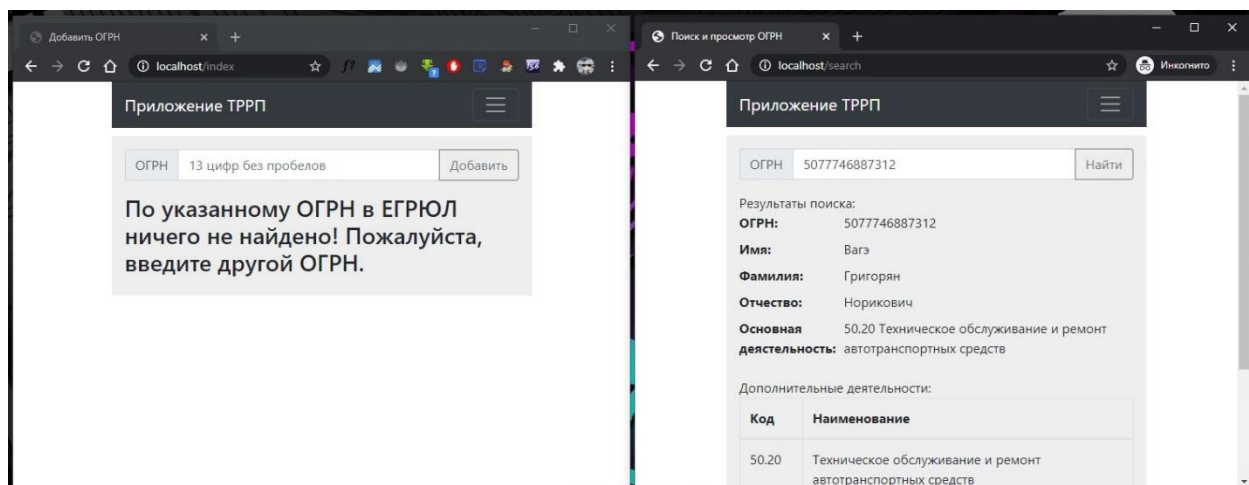


Рис. 9, одновременная работа нескольких клиентов на одной машине

Помимо одновременного запуска клиентов на одной машине можно запустить также и несколько серверов. Например:

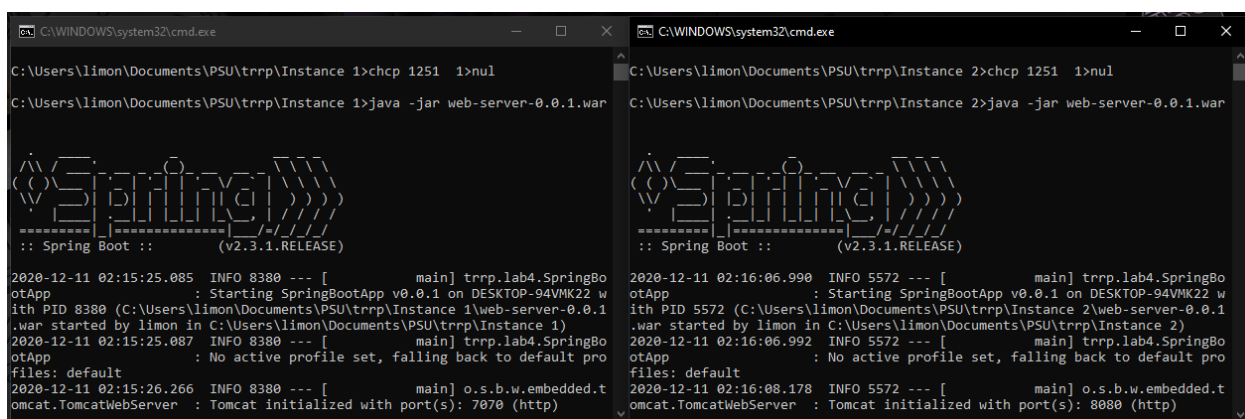


Рис. 10, одновременная работа нескольких веб-серверов на одной машине

Серверы распределенной системы выполняют различные функции. Подробное описание их функций приведено в разделе 1.2, краткое описание:

- BACKUP – функция хранения резервных копий;
- Master-DB – функция основного сервера баз данных;
- Slave-DB – функция реплицируемого сервера баз данных;
- Диспетчер – функция балансировки нагрузки;
- Web-1 – функция веб-сервера;
- Web-2 – функция веб-сервера (дубликат Web-1);

2.4 Приложение является масштабируемым, позволяет добавлять новых участников взаимодействия без переписывания кода и перезапуска приложений.

Требование выполнено полностью.

Приложение позволяет добавлять новых клиентов без переписывания кода и перезапуска приложений, так как за взаимодействие с клиентским трафиком отвечает nginx. nginx принимает все входящие соединения. Пример зафиксированных nginx обращений к Системе (видны различные IP-адреса и устройства, с которых отправлялись запросы):

```
127.0.0.1 - - [11/Dec/2020:00:34:42 +0500] "GET /search HTTP/1.1" 200 2860
"http://localhost/index" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36"

127.0.0.1 - - [11/Dec/2020:00:34:55 +0500] "GET /search HTTP/1.1" 200 2860
"http://localhost/index" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36"

192.168.0.2 - - [11/Dec/2020:00:38:29 +0500] "GET /favicon.ico HTTP/1.1"
404 124 "http://192.168.0.7/index" "Mozilla/5.0 (Linux; Android 10; Mi A2
Lite) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Mobile
Safari/537.36"

192.168.0.5 - - [11/Dec/2020:00:38:42 +0500] "GET /index HTTP/1.1" 200 2804
"- " "Mozilla/5.0 (Linux; Android 10; LRA-LX1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/86.0.4240.99 Mobile Safari/537.36"
```

Также возможно масштабирование веб-серверов и серверов баз данных. Для добавления нового сервера достаточно ввести новую машину в локальную сеть, провести ее конфигурирование (запустить на ней экземпляр приложения с настроенным application.properties) и добавить ее в nginx с помощью конфигурационного файла, расположенного на машине-диспетчере по пути `/etc/nginx/conf/nginx.conf`. Для активации машины необходимо динамически перечитать конфигурацию nginx без перезапуска Системы:



The image shows two side-by-side screenshots of nginx configuration files, illustrating the process of adding a new server to an upstream group. Both screenshots show the same configuration structure, but the second one includes an additional server entry.

```
http {
    include      mime.types;
    default_type application/octet-stream;

    sendfile      on;
    keepalive_timeout 65;

    upstream trrpapp.local {
        ip_hash;

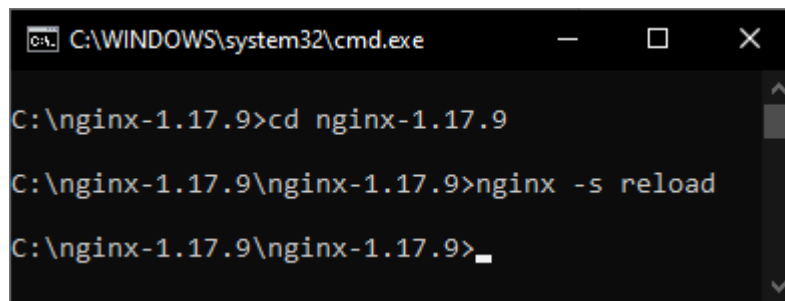
        server localhost:7070;
        server localhost:8080;
    }

    server {
```

In the second screenshot, an additional server entry is added to the `upstream trrpapp.local` block:

```
        server 92.255.139.213:8080;
```

Рис. 11.1, добавление в конфигурацию nginx нового сервера



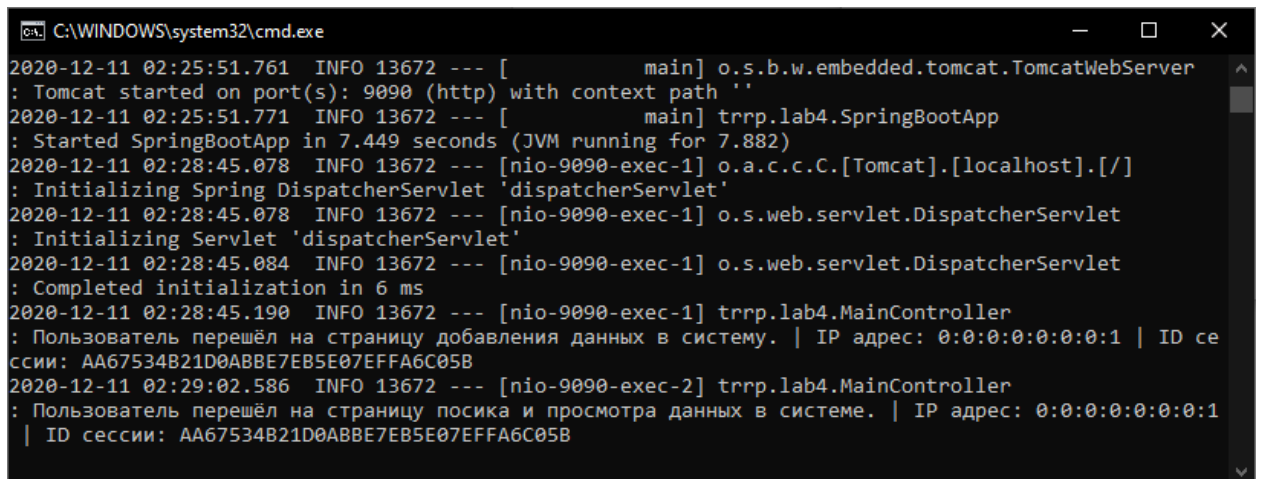
```
C:\WINDOWS\system32\cmd.exe

C:\nginx-1.17.9>cd nginx-1.17.9

C:\nginx-1.17.9\nginx-1.17.9>nginx -s reload

C:\nginx-1.17.9\nginx-1.17.9>_
```

Рис. 14, обновление конфигурации nginx



```
C:\WINDOWS\system32\cmd.exe

2020-12-11 02:25:51.761 INFO 13672 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat started on port(s): 9090 (http) with context path ''
2020-12-11 02:25:51.771 INFO 13672 --- [main] trrp.lab4.SpringBootApplication
: Started SpringBootApplication in 7.449 seconds (JVM running for 7.882)
2020-12-11 02:28:45.078 INFO 13672 --- [nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-12-11 02:28:45.078 INFO 13672 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet
: Initializing Servlet 'dispatcherServlet'
2020-12-11 02:28:45.084 INFO 13672 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet
: Completed initialization in 6 ms
2020-12-11 02:28:45.190 INFO 13672 --- [nio-9090-exec-1] trrp.lab4.MainController
: Пользователь перешёл на страницу добавления данных в систему. | IP адрес: 0:0:0:0:0:0:1 | ID се
ссии: AA67534821D0ABBE7EB5E07EFA6C05B
2020-12-11 02:29:02.586 INFO 13672 --- [nio-9090-exec-2] trrp.lab4.MainController
: Пользователь перешёл на страницу поиска и просмотра данных в системе. | IP адрес: 0:0:0:0:0:0:1
| ID сессии: AA67534821D0ABBE7EB5E07EFA6C05B
```

Рис. 15, проверка работоспособности

Как видно по рисунку 15, веб-сервер принимает и обрабатывает запросы клиентов.

Масштабирование серверов баз данных происходит по тому же принципу. Необходимо изменить конфигурационные файлы СУБД на добавляемой машине и на машине-мастере для настройки работы Системы по протоколу потоковой репликации (на мастере добавить добавляемую машину в *archive_command*, на добавляемой машине добавить *primary_conninfo* по аналогии с уже существующим Slave-DB). Перечитывание конфигурации серверов баз данных происходит также без перезапуска Системы.

2.5 Существует возможность динамического реконфигурирования системы.

Требование выполнено полностью.

Для каждого компонента Системы существуют свои конфигурационные файлы:

- BACKUP (сервер-хранилище): */etc/barman.conf* — определяются параметры для связи с главным сервером баз данных и параметры резервного копирования, перечитывание измененных настроек

происходит автоматически при изменении файла без перезапуска системы;

- Master-DB (сервер баз данных): `/var/log/pgsql/data/postgresql.conf` – определяют настройки сервера баз данных, репликации и связи с хранилищем резервных копий [24]. Перечитывание конфигураций происходит без перезапуска системы с помощью системной функции СУБД `pg_reload_conf()`;
- Slave-DB (сервер баз данных) – аналогично Master-DB;

```
[postgres@trrp-centos-db-03 ~]$ systemctl status postgresql | grep Active
Active: active (running) since Thu 2020-12-10 20:02:01 UTC; 1min 58s ago
[postgres@trrp-centos-db-03 ~]$ psql -c "show temp_file_limit"
temp_file_limit
-----
1GB
(1 row)

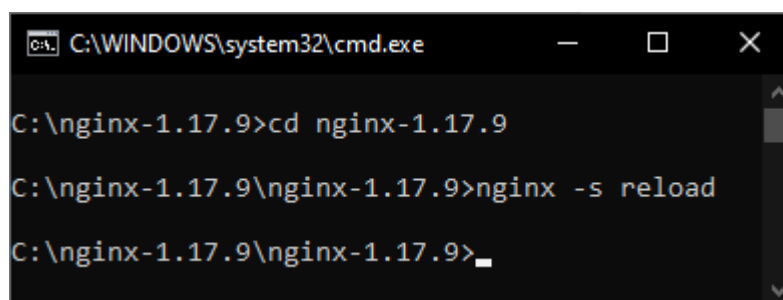
[postgres@trrp-centos-db-03 ~]$ echo "temp_file_limit = 2GB" >> /var/lib/pgsql/data/postgresql.conf
[postgres@trrp-centos-db-03 ~]$ psql -c "select pg_reload_conf()"
pg_reload_conf
-----
t
(1 row)

[postgres@trrp-centos-db-03 ~]$ psql -c "show temp_file_limit"
temp_file_limit
-----
2GB
(1 row)

[postgres@trrp-centos-db-03 ~]$ systemctl status postgresql | grep Active
Active: active (running) since Thu 2020-12-10 20:02:01 UTC; 2min 28s ago
[postgres@trrp-centos-db-03 ~]$
```

Рис. 16, динамическое изменение параметров СУБД без перезапуска системы

- Д
и
с
П
е
Т
ч
е
р
:



```
C:\WINDOWS\system32\cmd.exe
C:\nginx-1.17.9>cd nginx-1.17.9
C:\nginx-1.17.9\nginx-1.17.9>nginx -s reload
C:\nginx-1.17.9\nginx-1.17.9>_
```

Рис.17, обновление конфигурации nginx без перезапуска

- Веб-сервер использует конфигурационный файл `application.properties`. Данный файл располагается в той же директории, где расположен исполняемый файл веб-сервера. В конфигурации веб-сервера указывается IP адреса серверов баз данных – основной и резервный. Приложение использует основное

п
х
с
о

соединение до тех пор, пока не случится какой-либо сбой. Если возникает ошибка выполнения запроса или соединение с основной базой данных не устанавливается, то веб-сервер автоматически переключается на использование резервной базы данных. На рисунке 17 видно, что в лог выводится информация о том, что используется основное соединение. Затем, база данных была принудительно отключена, после чего на стороне клиента был выполнен запрос. На рисунке 18 показано, что веб-сервер автоматически переключился на резервное соединение.

```
2020-12-11 01:36:19.433 INFO 252 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor
: Initializing ExecutorService 'taskExecutor'
2020-12-11 01:36:19.900 INFO 252 --- [ restartedMain] trrp.lab4.db.DataSourcePostgres
: Используется основное подключение к БД: 134.209.229.170:5432
2020-12-11 01:36:20.283 INFO 252 --- [ restartedMain] o.s.b.a.w.s.WelcomePageHandlerMapping
: Adding welcome page template: index
2020-12-11 01:36:20.348 INFO 252 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer
: LiveReload server is running on port 35729
```

Рис.18, включение основного соединения к БД

```
2020-12-11 01:37:11.613 INFO 252 --- [nio-7070-exec-3] trrp.lab4.MainController
: Пользователь произвёл попытку добавления данных в систему. | IP адрес: 0:0:0:0:0:0:1 | ID
сессии: 50C7C1F3054E8C0544D4AEF64F9646D8
2020-12-11 01:37:16.189 INFO 252 --- [nio-7070-exec-3] trrp.lab4.db.DataSourcePostgres
: Используется резервное подключение к БД: 138.197.179.157:5432
```

Рис.19, переключение на резервное соединение к БД

2.6 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.
Требование выполнено не полностью.

В распределенной системе используются следующие типы коммуникаций:

- **Web-сервис.** Веб-серверы распределённой системы работают с клиентом по протоколу HTTP. Клиент получает данные от веб-сервера в виде HTML-документа:

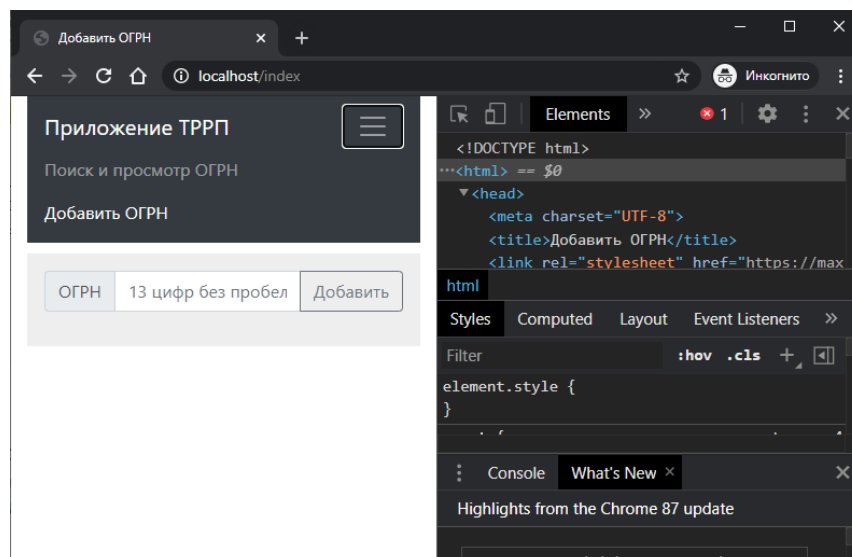


Рис.20, HTML документ полученный по запросу <http://localhost/index>

Взаимодействие по протоколу HTTP – самый очевидный и удобный вариант взаимодействия с использованием клиента в виде браузера и веб-сервера.

- **Сокеты.** Веб-сервер и модуль обработки документов взаимодействуют с помощью сокетов. Это обусловлено тем, что эти два компонента напрямую взаимосвязаны, т.к. модуль обработки документов запускается с помощью системного вызова из веб-сервера. Наиболее простой способ организовать связь между ними – это использовать сокет, установив конкретные значения портов для детерминированного определения компонентов и организации взаимодействий.

Таким образом, в системе используется лишь 2 средства коммуникации. Требование выполнено не полностью.

2.7 Система является отказоустойчивой. В случае если один и/или несколько компонент системы аварийно завершают свою работу. Требование выполнено полностью.

Если во время работы системы какой-либо узел завершает работу, то это не повредит работу системы в целом. Все клиенты будут обслуживаться и дальше, т.к. nginx автоматически распределяет запросы пользователей только по тем веб-серверам, которые находятся в рабочем состоянии. Кроме того, в системе присутствует резервная СУБД, с которой будут работать веб-серверы, если основная СУБД аварийно завершит работу.

2.8 Распределенное приложение продолжает работать и в случае, если после аварийного завершения некоторого компонента, он восстановлен на другом узле вычислительной сети.

Требование выполнено полностью.

Например, рассмотрим аварийное завершение работы основного сервера баз данных. В случае аварийного завершения работы основного сервера баз данных будет остановлена репликация. Необходимо обратиться к ведомому серверу - удалить файл `/var/lib/pgsql/data/standby.signal`, являющийся флагом реплицируемости, и перечитать конфигурацию СУБД. После этой операции основным компонентом СУБД становится Slave-DB. В конфигурационном файле приложения указаны адреса всех серверов баз данных, первым выполняется подключение к Master-DB. В случае, если Master-DB не отвечает, приложение автоматически переключается на другой адрес:

```
2020-12-11 01:37:11.613 INFO 252 --- [nio-7070-exec-3] trrp.lab4.MainController
: Пользователь произвёл попытку добавления данных в систему. | IP адрес: 0:0:0:0:0:0:1 | ID
сессии: 50C7C1F3054E8C0544D4AEF64F9646D8
2020-12-11 01:37:16.189 INFO 252 --- [nio-7070-exec-3] trrp.lab4.db.DataSourcePostgres
: Используется резервное подключение к БД: 138.197.179.157:5432
```

Р

2.9 Отчет содержит подробное описание архитектуры каждого компонента распределенного приложения.

Требование выполнено полностью, описание каждого компонента находится в пункте 1.2 отчета.

2

1

2.10 В отчете описана структура передаваемых данных, формат сообщений и вид протокола, используемого для этого.

Требование выполнено полностью, описание структур данных описано в пункте 1.3.4 отчета.

П

е

2.11 В отчете представлено описание способа передачи сообщений при коммуникации компонентов распределенной системы с обоснованием.

Требование выполнено полностью, описание способа передачи сообщений описано в пункте 2.6 отчета.

П

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

а

S

1

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.	2
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

Список источников

1. ФНС России: ПРЕДОСТАВЛЕНИЕ СВЕДЕНИЙ ИЗ ЕГРЮЛ/ЕГРИП [Электронный ресурс] URL: <https://egrul.nalog.ru/> (дата обращения 03.12.2020).
2. ФНС России: ПРЕДОСТАВЛЕНИЕ СВЕДЕНИЙ ИЗ ЕГРЮЛ/ЕГРИП. О сервисе [Электронный ресурс] URL: <https://egrul.nalog.ru/about.html> (дата обращения 03.12.2020).
3. ФНС России: API-ФНС [Электронный ресурс] URL: <https://api-fns.ru/> (дата обращения 03.12.2020).
4. Приказ от 18 февраля 2015 г. № 25н Об утверждении порядка ведения единого государственного реестра юридических лиц и единого государственного реестра индивидуальных предпринимателей, исправления технической ошибки в записях указанных государственных реестров, предоставления содержащихся в них сведений и документов органам государственной власти, иным государственным органам, органам государственных внебюджетных фондов, органам местного самоуправления и судам.
5. ФНС России: ПРЕДОСТАВЛЕНИЕ СВЕДЕНИЙ ИЗ ЕГРЮЛ/ЕГРИП. Тариф API Старт [Электронный ресурс] URL: <https://egrul.nalog.ru/> (дата обращения 03.12.2020).
6. Centos wiki: Centos 8 Release Notes [Электронный ресурс] URL: <https://wiki.centos.org/Manuals/ReleaseNotes/CentOS8.1905> (дата обращения 03.12.2020).
7. Fossies: xfs Release Notes [Электронный ресурс] URL: <https://fossies.org/linux/xfsprogs/doc/CHANGES> (дата обращения 03.12.2020).
8. pgbarman: barman 2.15 [Электронный ресурс] URL: <https://www.pgbarman.org/about/> (дата обращения 03.12.2020).
9. PostgreSQL: PostgreSQL 12 Release Notes [Электронный ресурс] URL: <https://www.postgresql.org/docs/release/12.0/> (дата обращения 03.12.2020).
10. PostgreSQL 12: Replication [Электронный ресурс] URL: <https://www.postgresql.org/docs/12/runtime-config-replication.html> (дата обращения 03.12.2020).

11. nginx: about nginx [Электронный ресурс] URL: <https://nginx.org/en/> (дата обращения 03.12.2020).
12. Oracle: Java 11 Release Notes [Электронный ресурс] URL: <https://www.oracle.com/java/technologies/javase/11-relnotes.html> (дата обращения 03.12.2020).
13. jOOQ Notes: jOOQ 3.14.4 [Электронный ресурс] URL: <https://www.jooq.org/notes> (дата обращения 03.12.2020).
14. Python docs: Python 3.8 Release Notes [Электронный ресурс] URL: <https://docs.python.org/3/whatsnew/3.8.html> (дата обращения 03.12.2020).
15. pypi: pymupdf 1.18.4 [Электронный ресурс] URL: <https://pypi.org/project/PyMuPDF/1.18.4/> (дата обращения 03.12.2020).
16. pypi: requests 2.25.0 [Электронный ресурс] URL: <https://pypi.org/project/requests/2.25.0/> (дата обращения 03.12.2020).
17. Spring: Spring Framework 5.2.7 Release Notes [Электронный ресурс] URL: <https://spring.io/blog/2020/06/09/spring-framework-5-2-7-and-5-1-16-available-now> (дата обращения 03.12.2020).
18. ssh: ssh protocol [Электронный ресурс] URL: <https://www.ssh.com/ssh/protocol/> (дата обращения 03.12.2020).
19. samba: rsync [Электронный ресурс] URL: <https://rsync.samba.org/> (дата обращения 03.12.2020).
20. HTTP/2 [Электронный ресурс] URL: <https://tools.ietf.org/html/draft-ietf-httpbis-http2-17> (дата обращения 03.12.2020).
21. Oracle: JDBC [Электронный ресурс] URL: <https://www.oracle.com/java/technologies/javase/javase-tech-database.html> (дата обращения 03.12.2020).
22. dbdiagram [Электронный ресурс] URL: <https://dbdiagram.io> (дата обращения 03.12.2020).
23. barman: config file syntax [Электронный ресурс] URL: <https://docs.pgbarman.org/release/2.12/barman.5.html> (дата обращения 03.12.2020).
24. PostgreSQL: PostgreSQL config [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql/12/config-setting> (дата обращения 03.12.2020).

25. nginx: nginx config [Электронный ресурс] URL:
<https://nginx.org/ru/docs/> (дата обращения 03.12.2020).