Методические указания по дисциплине
«Администрирование информационных систем»
Лабораторная работа № 3

Развертывание и запуск серверной части системы. Приёмочное тестирование разработанной информационной системы

Цель: развернуть и запустить серверную часть системы; протестировать приложение; оформить итоговые документы и презентацию.

Задание

1. Ознакомиться с теоретической частью по основам тестирования программного обеспечения (Приложение 1).

2. Задокументировать полученный опыт:

а) Завершить проект с написанием итогового отчета: соединить все отчеты в одну папку, исходные файлы проекта.

б) Добавить информацию о вашем проекте на ваш *WordPress*-сайт (лабораторная работа № 1) или загрузить ваше приложение на *GitHub* (Приложение 2), указав информацию в README. Отобразить данный процесс в отчете.

3. Оформить презентацию по проекту:

а) Продемонстрировать основные этапы и результаты разработки приложения (можно использовать материалы отчетов).

б) Описать профессиональные навыки и основные обязанности студентов рабочей группы. Необходимо подробно описать вклад каждого студента в общий проект.

4. Развернуть и запустить ваше приложение на виртуальной машине (можно использовать машину, созданную при выполнении лабораторной работы №1).

5. Предоставить материалы преподавателю (итоговая папка, презентация). Папку может быть передана через систему СЭО УУНИТ/почту/ облачное хранилище.

6. Продемонстрировать работу разработанной информационной системы.

Требования к презентации: титульный лист; слайд с фотографиями авторов и контактной информацией (почта, телефон); содержание с кнопками навигации; основные пункты презентации; список источников.

Криерии презентаций: полнота раскрытия темы; структуризация информации; наличие и удобство навигации; отсутствие грамматических, орфографических и речевых ошибок; отсутствие фактических ошибок, достоверность представленной информации; наличие и правильность оформления обязательных слайдов (титульный, о проекте, список источников, содержание); оригинальность оформления презентации; обоснованность и рациональность использования средств мультимедиа и анимационных эффектов; применимость презентации для выбранной целевой аудитории; грамотность использования цветового оформления; использование авторских иллюстраций, фонов, фотографий, видеоматериалов; наличие дикторской речи, ее грамотность и целесообразность; наличие, обоснованность и грамотность использования фонового звука; размещение и комплектование объектов; единый стиль слайдов.

Порядок демонстрации проекта: Для демонстрации вашего клиент-серверного приложения, воспользуйтесь программой для нагрузочного тестирования **Locust**.

Создайте функции, изменяющие необходимые параметры в вашей БД, запустите тест Locust с небольшим количеством пользовательских запросов и продемонстрируйте изменение состояния интерфейса вашего клиентского приложения.

Требования к оформлению отчета:

Способ выполнения текста должен быть единым для всей работы. Шрифт – Times New Roman, кегль 14, межстрочный интервал – 1,5, размеры полей: левое – 30 мм; правое – 10 мм, верхнее – 20 мм; нижнее – 20 мм. Сокращения слов в тексте допускаются только общепринятые.

Абзацный отступ (1,25) должен быть одинаковым во всей работе. Нумерация страниц основного текста должна быть сквозной. Номер страницы на титульном листе не указывается. Сам номер располагается внизу по центру страницы или справа.

Методические рекомендации

1. Развертывание и запуск серверной части системы

При выполнении Лабораторной работы №2 основные этапы развёртывания приложения должны быть выполнены (скопированы исполняемые файлы, создана среда выполнения и установлены необходимые зависимости).

Для запуска приложения в production (продуктивном, рабочем) режиме необходима настройка дополнительной инфраструктуры на сервере. В первую очередь встроенный веб-сервер приложения НЕ должен использоваться напрямую, так как, как правило, данные веб-сервера не соответствуют требованиям безопасности. Таким образом, запросы к REST API должны перенаправляться (проксироваться) через веб-сервер специального назначения. Одним из таких веб-серверов является **Nginx**, типовая схема использования Nginx в качестве прокси-сервера представлена на рис. 1:

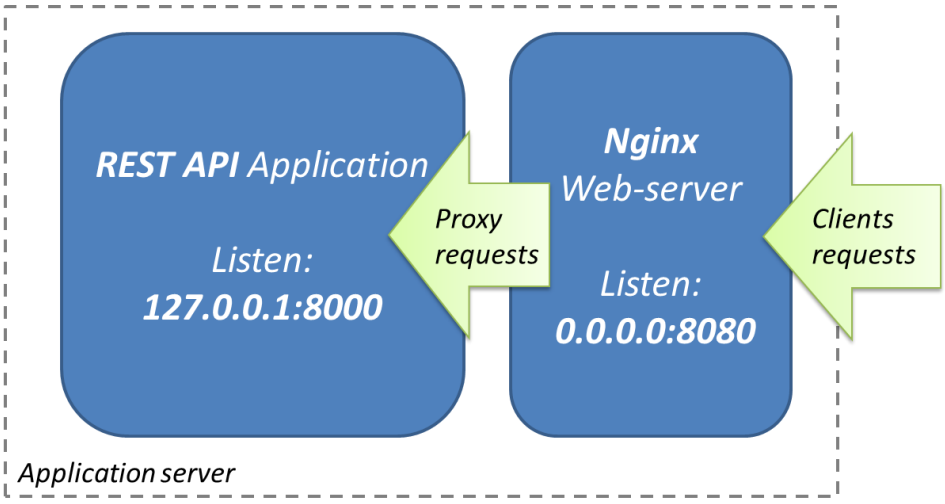


Рис. 1 – Типовая схема использования Nginx в качестве прокси-сервера

Такой способ перенаправления клиентских запросов из внешней сети на сервер (или несколько серверов) приложений, логически расположенных во внутренней сети называется **обратное проксирование (reverse proxy).**

ВАЖНО! Обратите внимание, что для того, чтобы избежать прямого сетевого доступа к REST API приложению, его запуск осуществляется на локальном сетевом интерфейсе (127.0.0.1), на который Nginx будет перенаправлять (проксировать) клиентские запросы.

Для реализации данной схемы взаимодействия устанавливаем Nginx, предварительно остановив веб-сервер Apache (если установлен), чтобы избежать конфликта доступа на порт 80:

```
systemctl stop apache2
```

Отключаем автозапуск при загрузке системы для Apache2:

```
systemctl disable apache2
```

Проверим статус Apache (должен быть disabled):

```
root@aislab22:/opt# systemctl disable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable apache2
Removed /etc/systemd/system/multi-user.target.wants/apache2.service.
root@aislab22:/opt# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: enabled)
     Active: inactive (dead)
       Docs: https://httpd.apache.org/docs/2.4/

ноя 27 14:42:29 aislab22 systemd[1]: Starting The Apache HTTP Server...
ноя 27 14:42:33 aislab22 systemd[1]: Started The Apache HTTP Server.
ноя 27 15:14:55 aislab22 systemd[1]: Stopping The Apache HTTP Server...
ноя 27 15:14:55 aislab22 systemd[1]: apache2.service: Succeeded.
ноя 27 15:14:55 aislab22 systemd[1]: Stopped The Apache HTTP Server.
root@aislab22:/opt#
```

Устанавливаем Nginx:

```
apt install nginx
```

Как и Apache2 Nginx использует директорию **/etc/nginx/sites-available** для конфигурации сайтов (virtual hosts). Создадим новую конфигурацию командой:

```
nano /etc/nginx/sites-available/backend-app.conf
```

Для базовой настройки проксирования (по рис. 1) содержание конфигурации должно быть следующим:

```
server {
        listen      8080;
        server_name _;
        location / {
            include proxy_params;
            proxy_pass http://127.0.0.1:8000;
```

```
        }
}
```

server_name – должно быть указано доменное имя сервера, если доменного имени нет, то ‘_’ (любое).

listen – порт доступа к приложению для проксирования через Nginx.

location – endpoint (адрес) вашего приложения для проксирования через Nginx. Символ ‘/’ означает, что REST API будет доступно по оригинальным адресам, т.е.:

<адрес_сервера>:8080/api/weatherforecast

Если, например, установить **location /proxy/** , то приложение будет доступно по адресу:

<адрес_сервера>:8080/proxy/api/weatherforecast

include proxy_params – использование специальных http-заголовков при проксировании.

proxy_pass – адрес для перенаправления запросов.

Активируем созданную конфигурацию, создав символическую ссылку (условно – «ярлык») на файл конфигурации в директории **/etc/nginx/sites-enabled**

```
ln -s /etc/nginx/sites-available/backend-app.conf
/etc/nginx/sites-enabled/
```

Перезапускаем Nginx:

```
systemctl restart nginx
```

и проверяем текущий статус веб-сервера:

```
systemctl status nginx
```

```
root@aislab22:/etc/nginx# systemctl restart nginx
root@aislab22:/etc/nginx# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-11-27 16:00:16 +05; 6s ago
       Docs: man:nginx(8)
    Process: 1321 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 1322 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 1323 (nginx)
      Tasks: 2 (limit: 1132)
     Memory: 2.5M
        CPU: 14ms
     CGroup: /system.slice/nginx.service
             ├─1323 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─1325 nginx: worker process

ноя 27 16:00:16 aislab22 systemd[1]: nginx.service: Succeeded.
ноя 27 16:00:16 aislab22 systemd[1]: Stopped A high performance web server and a reverse proxy server.
ноя 27 16:00:16 aislab22 systemd[1]: Starting A high performance web server and a reverse proxy server...
ноя 27 16:00:16 aislab22 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
ноя 27 16:00:16 aislab22 systemd[1]: Started A high performance web server and a reverse proxy server.
```

Далее **запускаем** REST API сервер.

В рамках выполнения лабораторной работы №4 запуск приложения на сервере (особенно приложения Python) происходил с помощью ввода последовательности команд, например, для Django:

активация виртуальной среды

```
source /opt/venv39/bin/activate
```

переход в директорию приложения:

```
cd /opt/lab-app-python-django/application
```

непосредственно запуск:

```
gunicorn -b 0.0.0.0:8000 -w 3 application.wsgi >>
debug.log &
```

Данную последовательность действий можно автоматизировать, воспользовавшись возможностью создания скриптов для командной оболочки Linux (command shell).

Создаем скрипт запуска **run.sh** в папке с вашим приложением:

```
nano /opt/lab-app-python-django/application/run.sh
```

Содержание скрипта (**WORKDIR** – переменная, содержащая путь к рабочей директории приложения):

```
#!/usr/bin/env bash

WORKDIR=/opt/lab-app-python-django/application
source /opt/venv39/bin/activate
cd $WORKDIR
gunicorn -b 127.0.0.1:8000 -w 3 application.wsgi >>
$WORKDIR/debug.log &
deactivate
```

Определяем данный файл в системе как исполняемый:

```
chmod +x /opt/lab-app-python-django/application/run.sh
```

Аналогичным образом создаем скрипт для останова вашего приложения (**stop.sh**) со следующим содержанием:

```
#!/usr/bin/env bash
```

```
pkill -f gunicorn
```

После создания скриптов запускать приложение можно одной командой:

```
/opt/lab-app-python-django/application/run.sh
```

Остановка, соответственно:

```
/opt/lab-app-python-django/application/stop.sh
```

Дополнительную информацию по написанию скриптов в Linux можно посмотреть здесь:

https://losst.pro/napisanie-skriptov-na-bash

ВАЖНО! Веб-приложения на Linux-сервере КРАЙНЕ НЕ РЕКОМЕНДУЕТСЯ запускать от лица суперпользователя (root), таким образом, необходимо предварительно выдать права на использование папки с вашим приложением обычному пользователю, например, для пользователя ais выполняем от лица суперпользователя:

```
chown -R ais /opt/lab-app-python-django
```

Также для Python-приложений необходимо выдать пользователю права на использование директории с виртуальной средой

Переходим на аккаунт **ais**:

```
su ais
```

И запускаем приложение.

Если при запуске приложения возникают ошибки прав доступа, также назначьте пользователю права на использование директории с виртуальной средой:

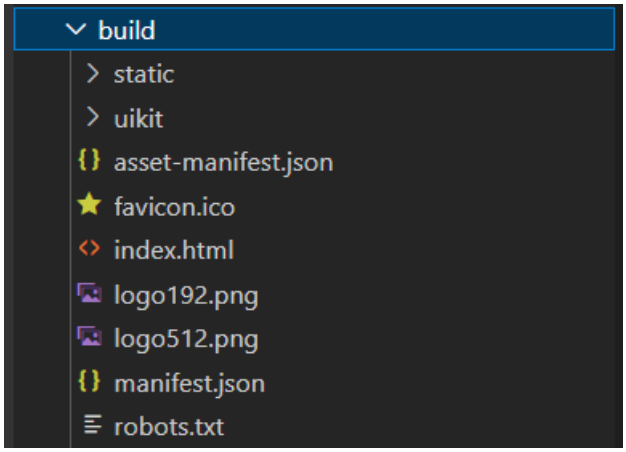
```
chown -R ais /opt/venv39
```

2. Развертывание клиентской части системы (для frontend-приложения на *JavaScript*)

2.1 Сборка проекта. Перед развёртыванием клиента на React (см. пример реализации frontend-приложения в Лабораторной работе №6) необходима сборка проекта. Если React-приложение создано с помощью ***create-react-app***, то в проекте уже будет установлена система сборки ***webpack***. Для сборки такого проекта перейдите в директорию с файлами **`package.json`** (конфигурация для сборки проекта) и выполните:

```
npm run build
```

В корне проекта появится папка **/build** , содержащая ***js***-код и статичные ресурсы приложения (html, css и т.д.).

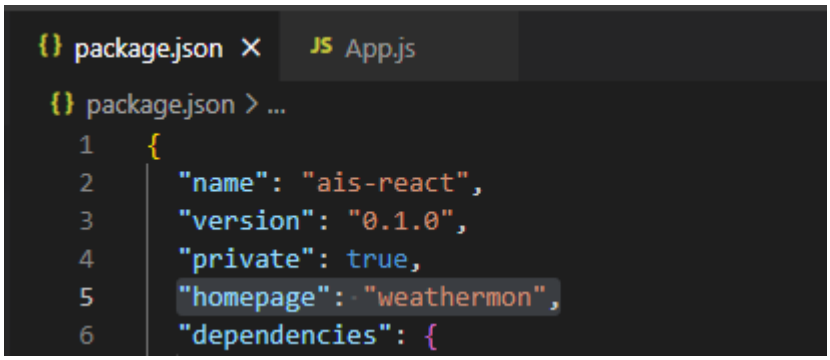


ВАЖНО! Перед сборкой проекта не забудьте поменять необходимые статичные переменные в проекте, например, адрес backend-сервера (API_URL). Также при сборке необходимо учитывать по какому адресу будет доступно ваше frontend-приложение. Если адрес приложения будет, например:

<адрес_сервера>/weathermon

То в **package.json** необходимо указать параметр:

"homepage": "weathermon"



Подробнее о системе сборки webpack можно почитать здесь:

https://metanit.com/web/react/2.9.php

2.2 Развёртывание React-приложения. В рамках текущей схемы развёртывания взаимодействие клиентской и серверной части системы будет выглядеть следующим образом (рис. 2):

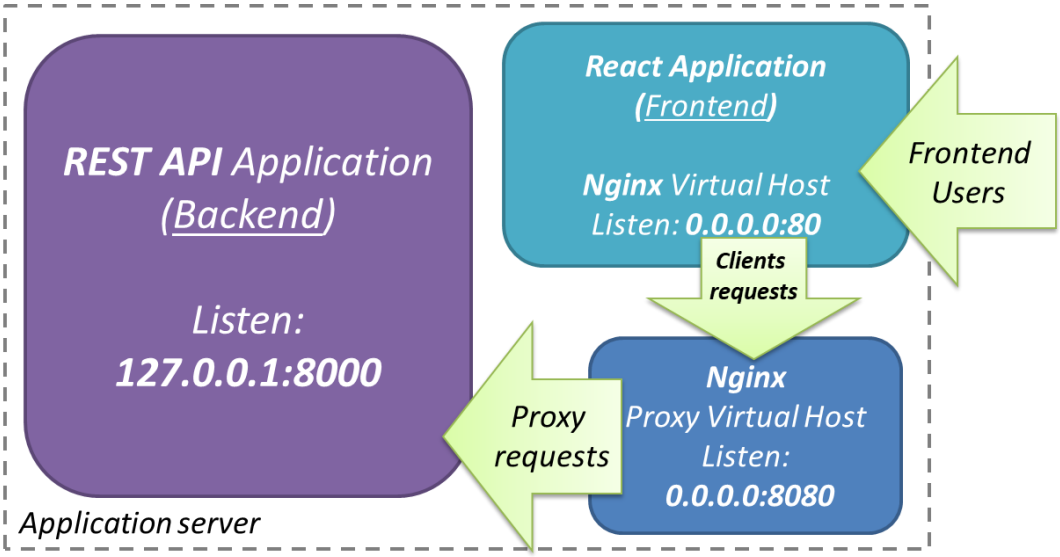


Рис. 2 – Типовая схема взаимодействия компонентов клиент-серверного приложения, развёрнутого на едином сервере

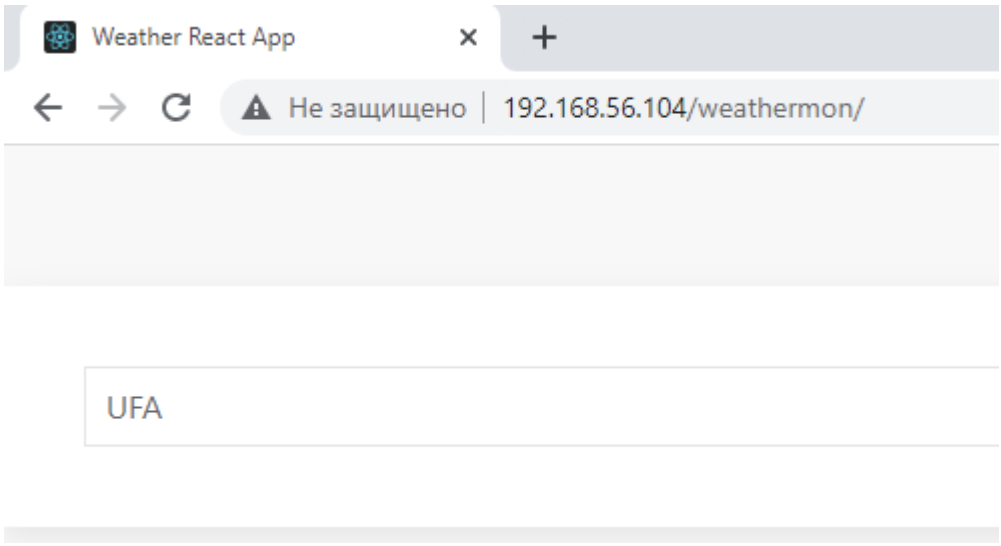
В более крупных проектах клиентская часть (Frontend) и серверная часть (REST API) обычно располагаются на отдельных серверах.

Как и Apache2 веб-сервер Nginx по умолчанию принимает подключения на портах 80 и 443 (для защищенных соединений) и предоставляет контент из директории **/var/www/html**. Таким образом, для предоставления собранного веб-приложения пользователям необходимо создать директорию, указанную в параметре **`"homepage"`** **(в `package.json`)** и скопировать в эту директорию содержимое папки **/build** из проекта. Для сборки из примера, копируем файлы приложения в папку **`/var/www/html/weathermon`**

Также необходимо выдать права системному пользователю, от которого по умолчанию запускается Nginx (пользователь **www-data**):

```
chown -R www-data:www-data /var/www/html/weathermon
```

Frontend-приложение будет доступно по следующему адресу:



Если в предыдущих лабораторных работах были изменены настройки CORS, то не забудьте установить корректные адреса в списке доступа:

```
['http://localhost',  'http://192.168.56.104']
```

ВАЖНО! Данные рекомендации по развертыванию клиент-серверного приложения охватывают только базовую настройку сервера для демонстрации проекта. Данная настройка может быть применена для приложений, которые будут работать только в **безопасной** локальной сети и **НЕ** будут доступны напрямую из сети интернет. Для безопасной работы приложения в глобальной сети необходима дополнительная настройка firewall (с помощью **ufw** или **iptables**), конфигурация проксирующего веб-сервера (Nginx или Apache) для работы через защищенный протокол **https**, а также защита вашего REST API с помощью системы авторизации.