

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего
образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ ЗАЩИЩЕН С
ОЦЕНКОЙ: _____

ПРЕПОДАВАТЕЛЬ:

ассистент / / А.Э.Зянчурин
(должность, учёная степень, звание) (подпись) (дата защиты) (инициалы, фамилия)

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

«Применение и конфигурирование общего системного
программного обеспечения. Разработка специального
программного обеспечения»

ПО КУРСУ: «Основы программной инженерии»

РАБОТУ ВЫПОЛНИЛ СТУДЕНТ:

Z9431 / Д.И.Андреев
(номер группы) (инициалы, фамилия)

/ / 29.05.2022
(подпись студента) (дата отчета)

Санкт-Петербург 2022

Цель работы

Целью работы является формирование практических навыков разработки специального программного обеспечения с учетом специфики клиент-серверной архитектуры.

Задание на лабораторную работу:

Вариант 1

Разработка программного обеспечения в соответствии с требованиями, предъявленными в первой лабораторной работе и поставленными задачами во второй лабораторной работе. Установка, конфигурирование и применение общего и системного программного обеспечения. Выполнение третьей лабораторной работы предполагает коллективное взаимодействие в группе из трех человек в ролях: «Специалист по дизайну графических пользовательских интерфейсов», «Разработчик Web и мультимедийных приложений», «Администратор баз данных».

Вариант 1

Разработка программного обеспечения для автоматизированной/информационной системы железной дороги

Выполнение работы:

Выполнение лабораторной работы проходило на виртуальной машине с развернутым образом Debian 9 в режиме сети bridge

```
it@debian:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:cc:a6:7c brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.222/24 brd 192.168.88.255 scope global dynamic enp0s3
        valid_lft 591sec preferred_lft 591sec
    inet6 fe80::a00:27ff:fecc:a67c/64 scope link
        valid_lft forever preferred_lft forever
it@debian:~$
```

1. Установим необходимые для выполнения работы пакеты apache2 и Postgresql, а также библиотеки libpqxx и libcgicc для работы с Posgresql из под C++ для последовательно выполнив команду:

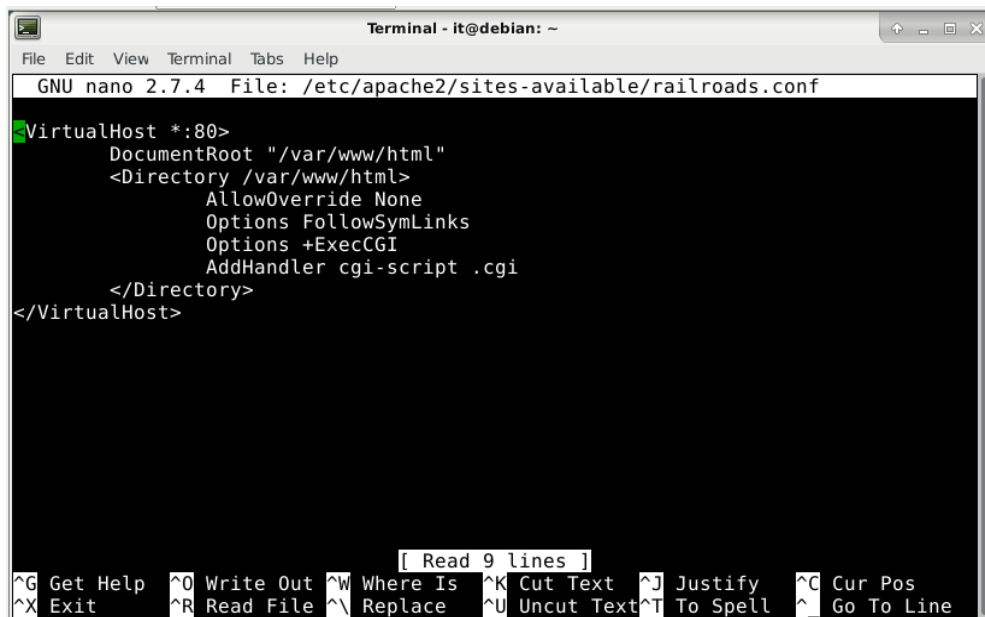
```
$ sudo apt update && sudo apt install apache2 postgresql postgresql-contrib libpqxx-4.0v5 libpqxx-dev libcgicc3 libcgicc-dev
```

```
Terminal - it@debian: ~
File Edit View Terminal Tabs Help
it@debian:~$ sudo apt update && sudo apt install apache2 postgresql postgresql-c
ontrib libpqxx-4.0v5 libpqxx-dev libcgicc3 libcgicc-dev
Hit:1 http://security.debian.org/debian-security stretch/updates InRelease
Ign:2 http://ftp.debian.org/debian stretch InRelease
Hit:3 http://ftp.debian.org/debian stretch-updates InRelease
Hit:4 http://ftp.debian.org/debian stretch Release
Reading package lists... Done
Building dependency tree
Reading state information... Done
360 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
postgresql is already the newest version (9.6+181+deb9u3).
postgresql-contrib is already the newest version (9.6+181+deb9u3).
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libpq-dev pkg-config
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom libcgicc-doc
  postgresql-doc-9.6 libpqxx4-doc
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcgicc-dev libcgicc3 libpq-dev
```

2. Отключим работу дефолтного сайта с конфигом /etc/apache2/sites-available/000-default.conf и включим модуль cgi
- ```
$ sudo a2dissite 000-default.conf
$ sudo a2enmod cgi
```

```
Terminal - it@debian: ~
File Edit View Terminal Tabs Help
it@debian:~$ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
 systemctl reload apache2
it@debian:~$ sudo a2enmod cgi
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Your MPM seems to be threaded. Selecting cgid instead of cgi.
Enabling module cgid.
To activate the new configuration, you need to run:
 systemctl restart apache2
it@debian:~$
```

3. Добавим конфиг нашего сайта и включим его:
- ```
$ sudo nano /etc/apache2/sites-available/railroads.conf
$ sudo a2ensite railroads.conf
$ sudo systemctl reload apache2.service
```



```
GNU nano 2.7.4 File: /etc/apache2/sites-available/railroads.conf

VirtualHost *:80>
    DocumentRoot "/var/www/html"
    <Directory /var/www/html>
        AllowOverride None
        Options FollowSymLinks
        Options +ExecCGI
        AddHandler cgi-script .cgi
    </Directory>
</VirtualHost>

[ Read 9 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

4. Изменим права для папки /var/www/html для возможности перезаписи в ней файлов без root прав:

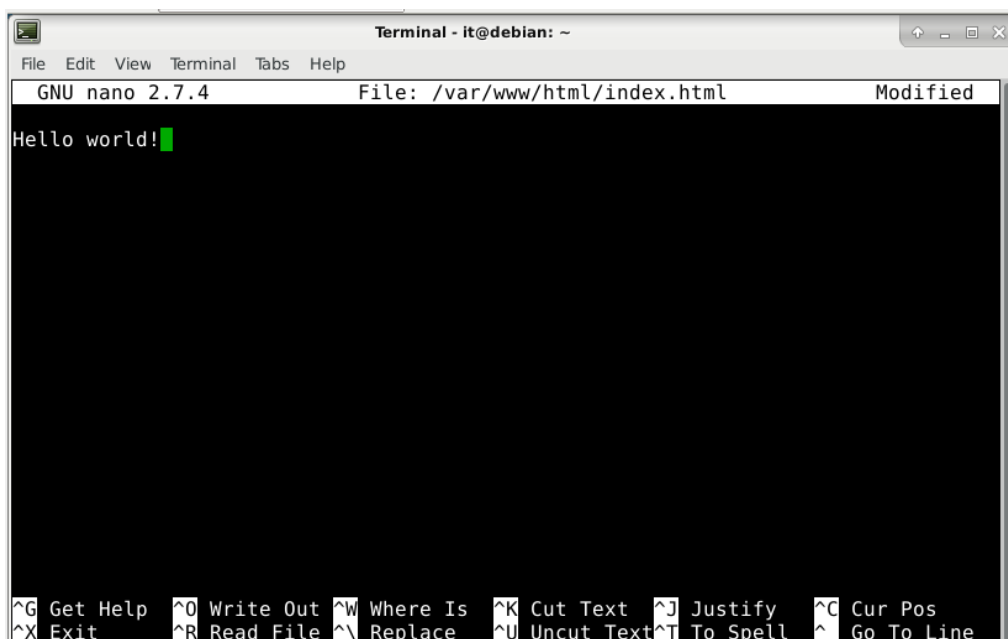
```
$ sudo chown -R it:it /var/www/html
$ sudo chmod -R 755 /var/www/html
```



```
it@debian:~$ sudo chown -R it:it /var/www/html/
it@debian:~$ sudo chmod -R 755 /var/www/html
it@debian:~$
```

4. Заменяем файл index.html в папке /var/www/html на тестовый

```
$ rm -rf /var/www/html/index.html
$ nano /var/www/html/index.html
```

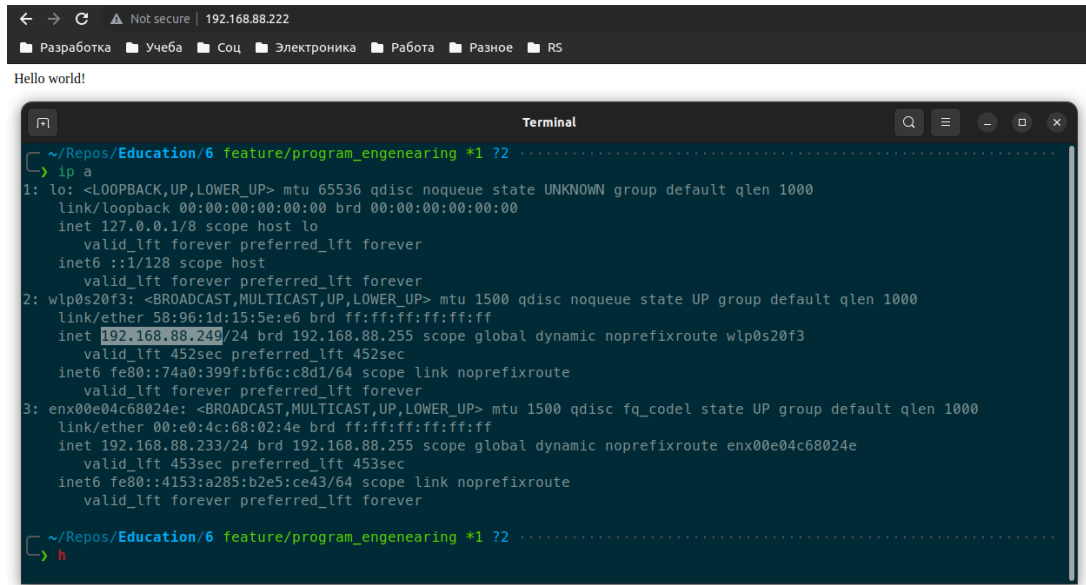


```
GNU nano 2.7.4 File: /var/www/html/index.html Modified

Hello world!

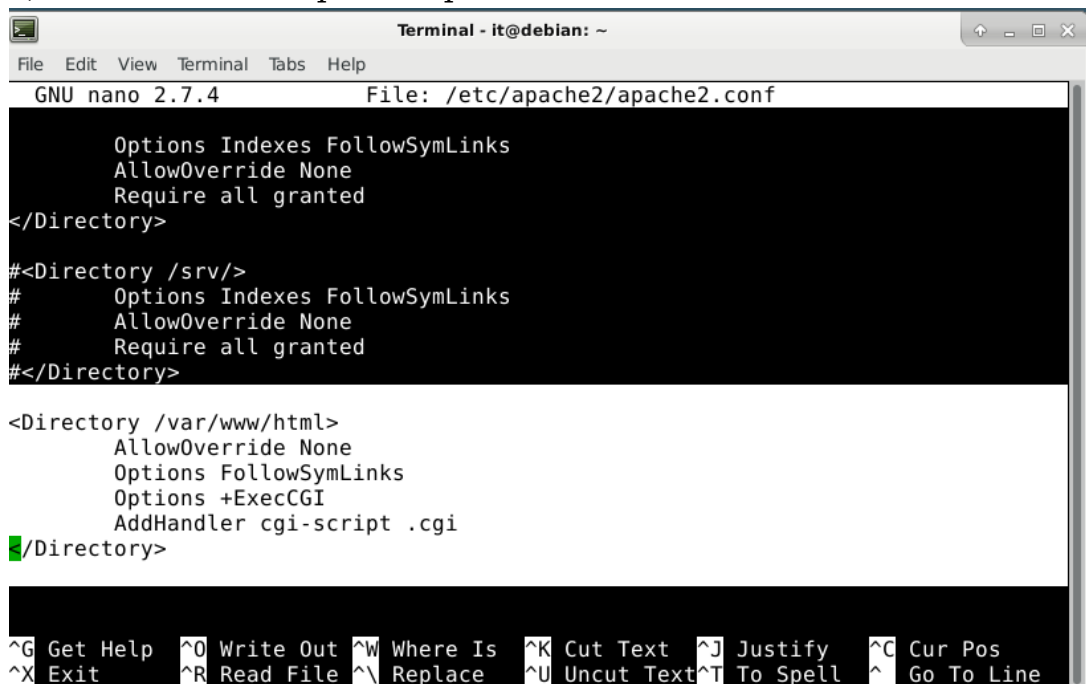
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

5. Убедимся, что сайт работает. В браузере хоста введем ip виртуальной машины:



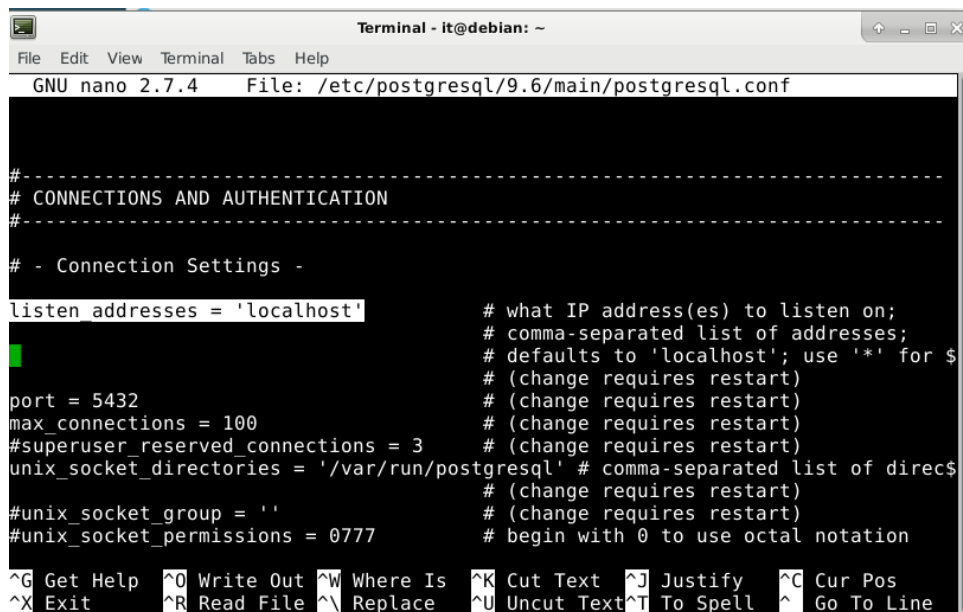
6. Внесем необходимые правки конфиг apache для корректной работы cgi скриптов:

```
$ sudo nano /etc/apache/apache2.conf
```



7. Для того, чтобы Postgres мог принимать входящие соединения, раскомментируем строку listen_addresses = 'localhost':

```
$ sudo nano /etc/postgresql/9.6/main/postgresql.conf
```



```
Terminal - it@debian: ~
File Edit View Terminal Tabs Help
GNU nano 2.7.4 File: /etc/postgresql/9.6/main/postgresql.conf

#-----
# CONNECTIONS AND AUTHENTICATION
#-----

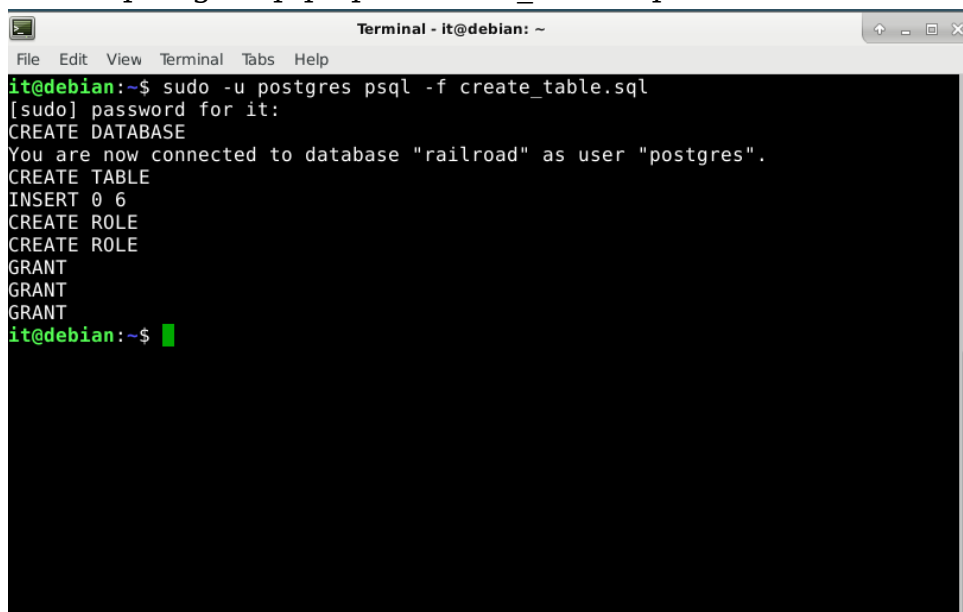
# - Connection Settings -

listen_addresses = 'localhost'          # what IP address(es) to listen on;
                                         # comma-separated list of addresses;
                                         # defaults to 'localhost'; use '*' for all
                                         # (change requires restart)
port = 5432                             # (change requires restart)
max_connections = 100                   # (change requires restart)
#superuser_reserved_connections = 3      # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                         # (change requires restart)
#unix_socket_group = ''                  # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation

^G Get Help  ^O Write Out ^W Where Is ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Uncut Text ^T To Spell  ^_ Go To Line
```

8. Создадим базу данных с помощью заранее подготовленного скрипта:

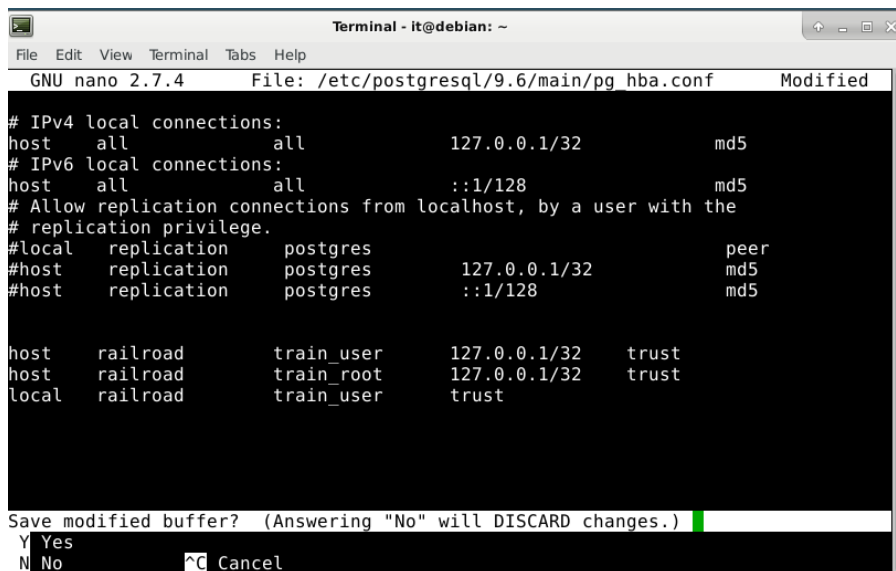
```
$ sudo -u postgres psql -f create_table.sql
```



```
Terminal - it@debian: ~
File Edit View Terminal Tabs Help

it@debian:~$ sudo -u postgres psql -f create_table.sql
[sudo] password for it:
CREATE DATABASE
You are now connected to database "railroad" as user "postgres".
CREATE TABLE
INSERT 0 6
CREATE ROLE
CREATE ROLE
GRANT
GRANT
GRANT
it@debian:~$
```

9. Добавим в конфиг pg_hba.conf конфигурацию подключения для пользователей базы данных



```
Terminal - it@debian: ~
File Edit View Terminal Tabs Help
GNU nano 2.7.4 File: /etc/postgresql/9.6/main/pg_hba.conf Modified

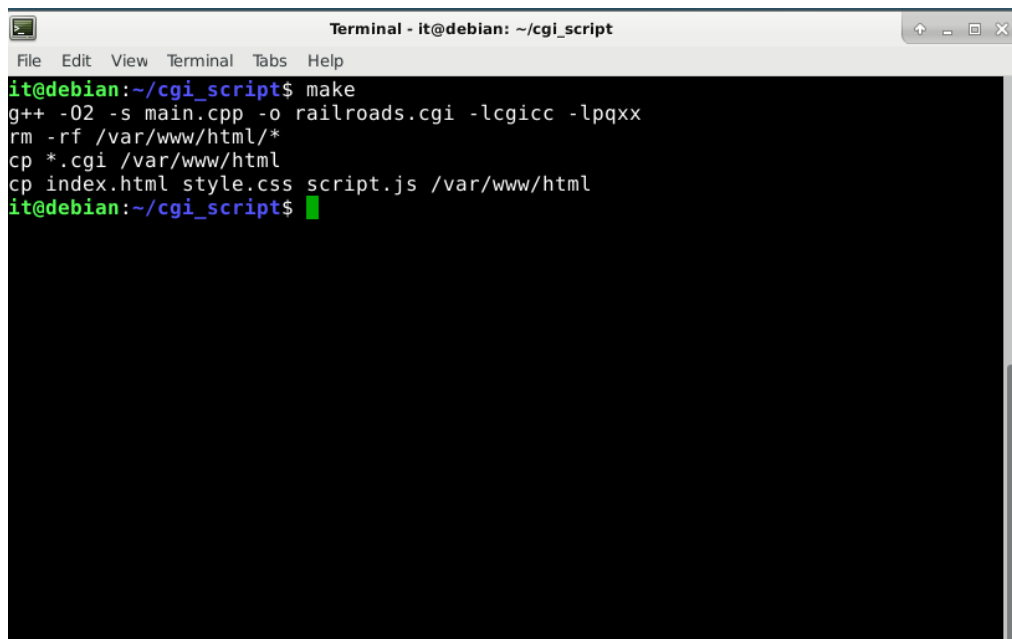
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local replication postgres peer
#host replication postgres 127.0.0.1/32 md5
#host replication postgres ::1/128 md5

host railroad train_user 127.0.0.1/32 trust
host railroad train_root 127.0.0.1/32 trust
local railroad train_user trust

Save modified buffer? (Answering "No" will DISCARD changes.)
Y Yes
N No ^C Cancel
```

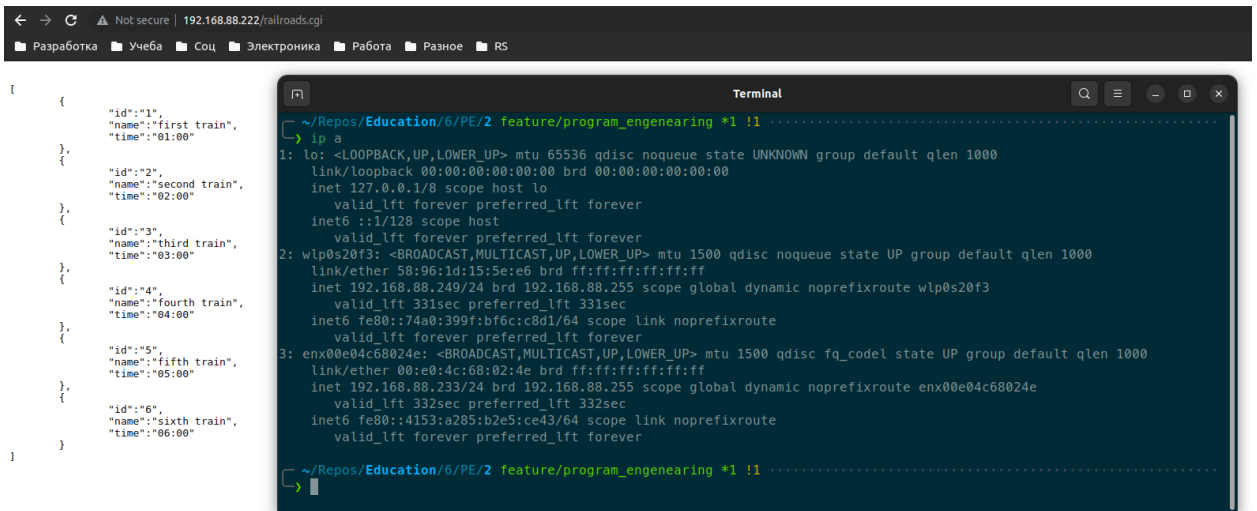
10. Соберем подготовленные cgi скрипты. Makefile написан таким образом, что все необходимые для работы сайта файлы (index.html, style.css, script.js, railroads.cgi), будут перемещены в директорию /var/www/html/:

\$ make

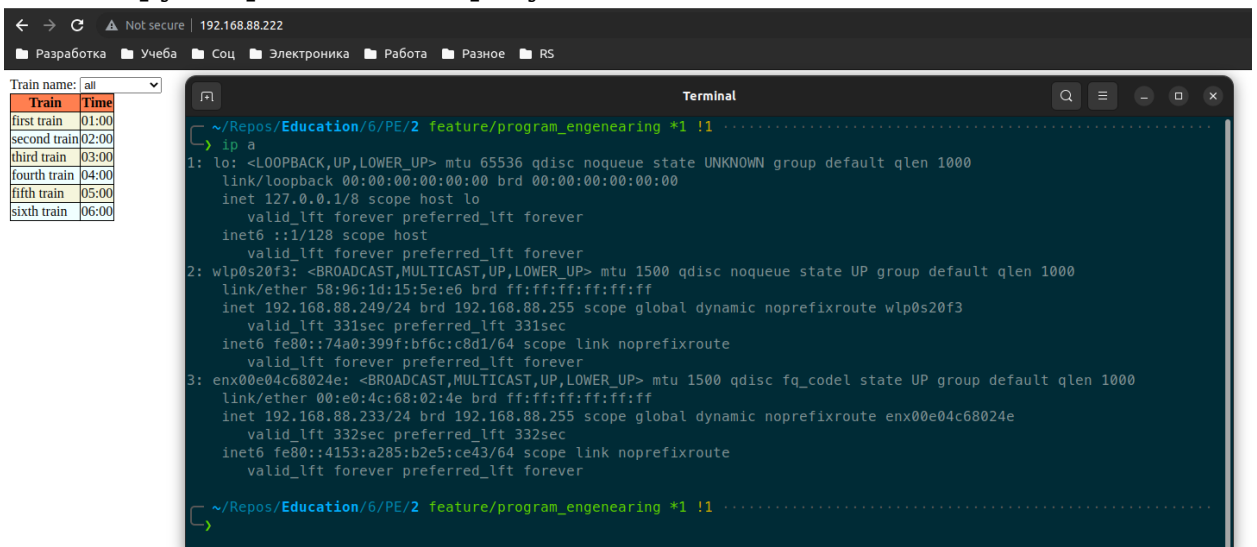


```
Terminal - it@debian: ~/cgi_script
File Edit View Terminal Tabs Help
it@debian:~/cgi_script$ make
g++ -O2 -s main.cpp -o railroads.cgi -lcgicc -lpqxx
rm -rf /var/www/html/*
cp *.cgi /var/www/html
cp index.html style.css script.js /var/www/html
it@debian:~/cgi_script$
```

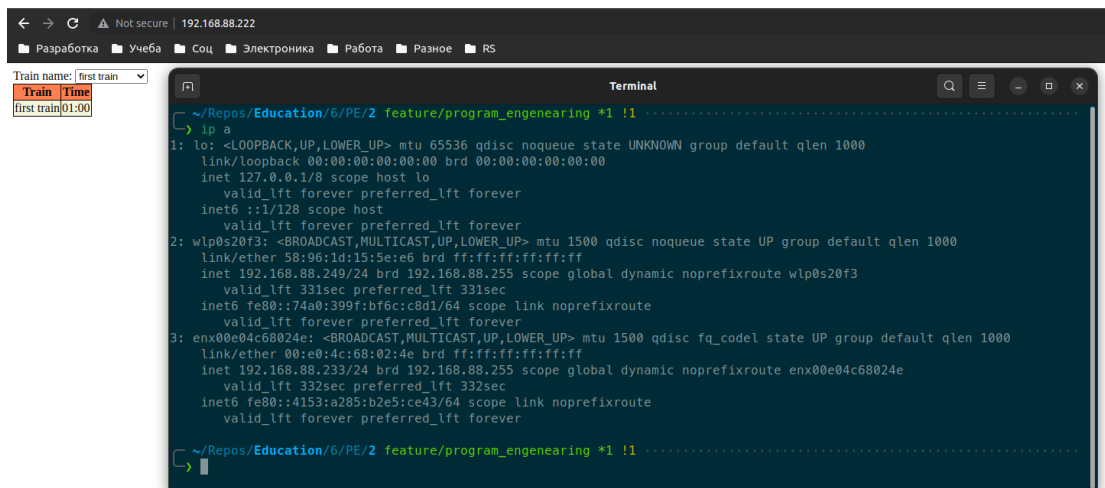
11. Убедимся, что cgi отправляет данные на хост. Перейдем в браузере по адресу 192.168.88.222/railroads.cgi:



12. Убедимся, что сервер обрабатывает входящие запросы. Для этого в созданном index.html файле предусмотрен контрол фильтрации по номеру. Перейдем по адресу 192.168.88.222:



Далее отфильтруем полученную таблицу. При этом cgi скрипт должен обработать входящий GET запрос и передать клиенту JSON только с одним набором данных (в таблице будет отображена только одна строка):



Выводы

В рамках лабораторной работы было произведено создание веб-сервиса согласно варианту задания. Была реализована трёхзвенная архитектура, включающая СУБД, веб-сервер (с CGI-скриптами в качестве замены выделенного сервера приложений), и веб-браузер в качестве клиента.

В СУБД была создана БД, два пользователя с правами доступа к ней (один с административными правами, другой с правами уровня приложения), а также таблица и небольшой набор тестовых данных. Для самой СУБД была сконфигурирована возможность сетевого подключения с локальной машины. Для веб-сервера был сконфигурирован сайт (виртуальный хост) для раздачи файлов из директории веб-сервиса, с возможностью исполнения CGI-скриптов.

Затем были разработаны сами CGI-скрипты, реализующие логику получения тестовых данных из базы данных, в виде JSON-документов. Также реализована возможность фильтрации

Наконец, был реализован небольшой Front-end приложения (с помощью

технологий HTML+CSS+JavaScript с применением библиотеки jQuery), который позволяет отображать данные в виде таблицы, с возможностью для пользователя динамически фильтровать записи.

Файл Makefile

```
all:
    g++ -O2 -s main.cpp -o railroads.cgi -lcgicc -lpqxx
    rm -rf /var/www/railroad/*
    cp *.cgi /var/www/railroad
    cp index.html style.css script.js /var/www/railroad
clean:
    rm -rf *.cgi
```

Файл index.html

```
<!doctype html>
<html>
  <head>
    <meta charset="utf8">
    <title>Train schedule</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></
script>
    <script src="script.js"></script>
    <link rel="stylesheet" href="style.css" type="text/css">
  </head>
  <body>
    <div>
      Train name:
      <select id="selector">
        <option value="0">all</option>
      </select>
    </div>
    <div id="tab">
      <table>
        <thead>
          <tr>
            <th>Train</th>
            <th>Time</th>
          </tr>
        </thead>
        <tbody id="body"></tbody>
      </table>
    </div>
  </body>
</html>
```

Файл style.css

```
table, tr, td, th {
    border: 1px solid black;
    border-collapse: collapse;
}

th {
    background-color: coral;
}

tr:hover {
    background-color: lightblue;
}

tr.odd {
    background-color: azure;
}

tr.even {
    background-color: beige;
}

#shedule-tab {
    margin-top: 15 px;
}
```

Файл script.js

```
const url="railroads.cgi";
function loadSelector() {
    $.getJSON(url, (data, status) => {
        const select = $("#selector");
        for(i in data)
        {
            const option = $(document.createElement("option"));
            option.text(data[i]["name"]);
            option.val(data[i]["id"]);
            select.append(option);
        }
    });
}
```

```

function redraw(schedule) {
  $("#tab").fadeOut(30, () => {
    $("#body").children("tr").remove();

    for (i in schedule) {
      const row = document.createElement("tr");
      if (i % 2) {
        row.setAttribute("class", "entity odd");
      } else {
        row.setAttribute("class", "entity even");
      }

      const name = document.createElement("td");
      name.innerText = schedule[i].name;
      row.appendChild(name);

      const time = document.createElement("td");
      time.innerText = schedule[i].time;
      row.appendChild(time);

      $("#body").append(row);
    }

    $("#tab").fadeIn();
  })
}

function onSelect() {
  const selection = $("#selector").val();
  let query = url;
  if (selection !== 0) {
    query = url + "?select=" + selection;
  }

  $.getJSON(query, (data, status) => {
    if (status !== "success") {
      return;
    }

    schedule = [];

    for (i in data) {
      console.warn("on select data length:", data.length);
      schedule.push({

```

```

        id: data[i].id,
        name: data[i].name,
        time: data[i].time
    });
}

    redraw(schedule);
});
}

$(document).ready(function(){
    $("#selector").change(() => { onSelect() });
    loadSelector();
    onSelect();
});

```

Файл main.cpp

```

// cgicc
#include <cgicc/Cgicc.h>
#include <cgicc/HTTPContentHeader.h>
#include <cgicc/HTTPStatusHeader.h>

// std
#include <sstream>
#include <iostream>
#include <map>

//pqxx
#include <pqxx/pqxx>

struct Train_info;

using train_info_vec = std::vector<Train_info>;

std::string make_json(train_info_vec &);
train_info_vec load_trains(int);
int parceQuery(cgicc::Cgicc &);

int main(int argc, char ** argv)
{
    try
    {
        cgicc::Cgicc cgi;

```

```

        std::cout << cgicc::HTTPContentHeader("application/json") <<
std::endl;
        auto selection = parceQuery(cgi);
        auto trains = load_trains(selection);
        auto result = make_json(trains);
        std::cout << result;
    }
    catch (std::exception & e)
    {
        std::cerr << "error while execute cgi script: " << e.what();
        std::cout << cgicc::HTTPStatusHeader(500, e.what()) <<
std::endl;
    }
}

struct Train_info
{
    int id;
    std::string name;
    std::string time;
};

int parceQuery(cgicc::Cgicc & cgi)
{
    int selection = -1;
    cgicc::form_iterator id = cgi.getElement("select");
    if (id != cgi.getElements().end())
    {
        selection = id->getIntegerValue(0);
    }

    return selection;
}

train_info_vec load_trains(int selection)
{
    try
    {
        const std::string conn_info =
"postgresql://train_user:user_pass@127.0.0.1/railroad?
connect_timeout=10";

        train_info_vec trains_info;
        pqxx::connection conn(conn_info);

```

```

pqxx::work job(conn);

auto res = job.exec("SELECT * FROM trains");
job.commit();

std::cerr << "filtered id:" << selection;

for (auto const & row : res)
{
    int id = row[0].as<int>();
    if (selection >= 0 && id != selection) {
        continue;
    }

    Train_info ti = {
        id,
        row[1].as<std::string>(),
        row[2].as<std::string>()
    };
    trains_info.push_back(ti);
}

return trains_info;
}
catch (std::exception & e)
{
    std::cerr << "error while execute query: " << e.what();
    throw e;
}
}

std::string make_json(train_info_vec & v)
{
    const std::string begin_array_token = "[\n";
    const std::string end_array_token = "]\n";

    std::ostringstream json_stream;

    json_stream << begin_array_token;
    for (auto it = v.begin(); it != v.end(); ++it)
    {
        json_stream << "\t{\n";
        json_stream << "\t\t\"id\": \"" << it->id << "\",\n";
        json_stream << "\t\t\"name\": \"" << it->name << "\",\n";
    }
}

```

```

    json_stream << "\\t\\t\\\"time\\\":\\\"\" << it->time << "\\\"\\n";

    if (it == --v.end())
    {
        json_stream << "\\t}\\n";
    }
    else
    {
        json_stream << "\\t},\\n";
    }
}
json_stream << end_array_token;

return json_stream.str();
}

```