Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа № 6

По дисциплине «OS Linux»

Контейнеризация

Студент Бахмутский М.В.

Группа АС-18

Руководитель Кургасов В.В.

Липецк 2020 г.

Цель работы

Изучить современные методы разработки ПО в динамических и распределенных средах на примере контейнеров Docker.

Ход работы

Клонируем проект Symfony Demo Application:

```
asd@asd:~$ git clone https://github.com/symfony/demo
Клонирование в «demo»…
remote: Enumerating objects: 9831, done.
remote: Total 9831 (delta 0), reused 0 (delta 0), pack-reused 9831
Получение объектов: 100% (9831/9831), 16.42 MiB | 3.05 MiB/s, готово.
Определение изменений: 100% (5916/5916), готово.
asd@asd:~$
```

Рисунок 1 – Клонирование проекта

Запустим проект с помощью команды php bin/console server:start предварительно скачав необходимые пакеты

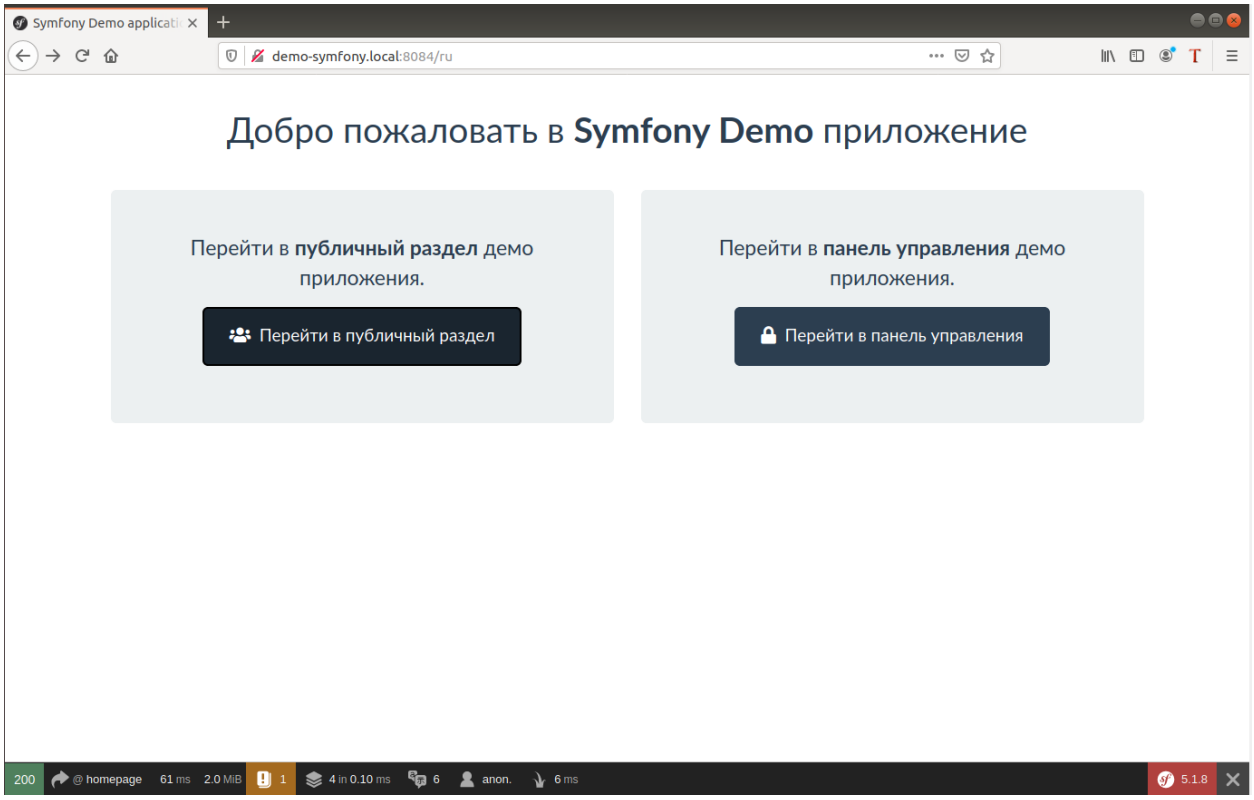
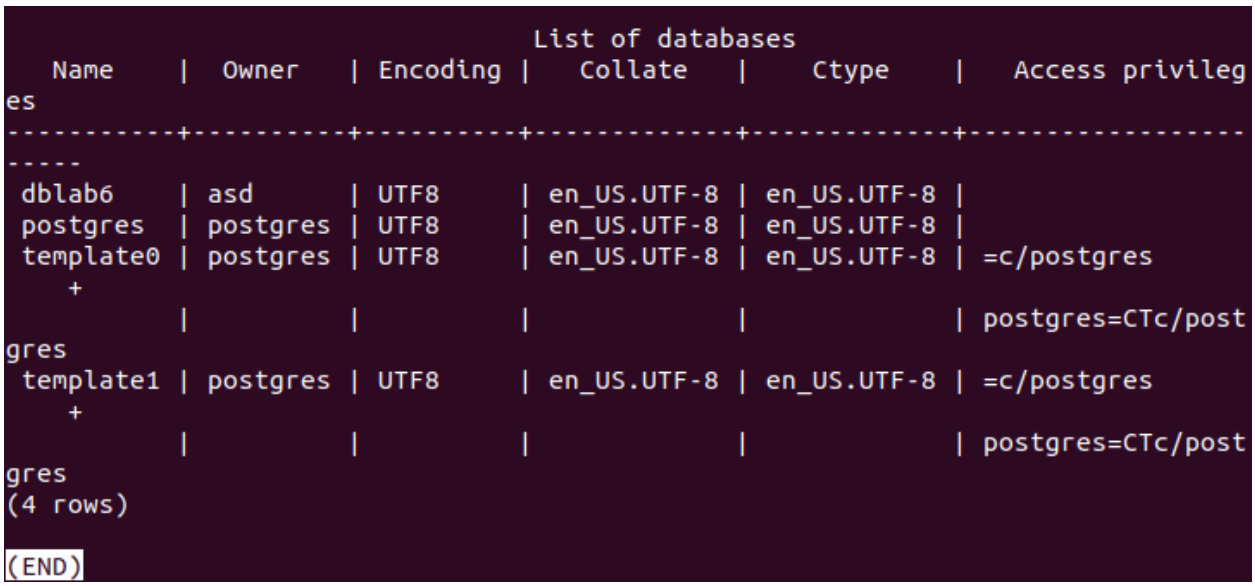


Рисунок 2 – Запуск демо проекта

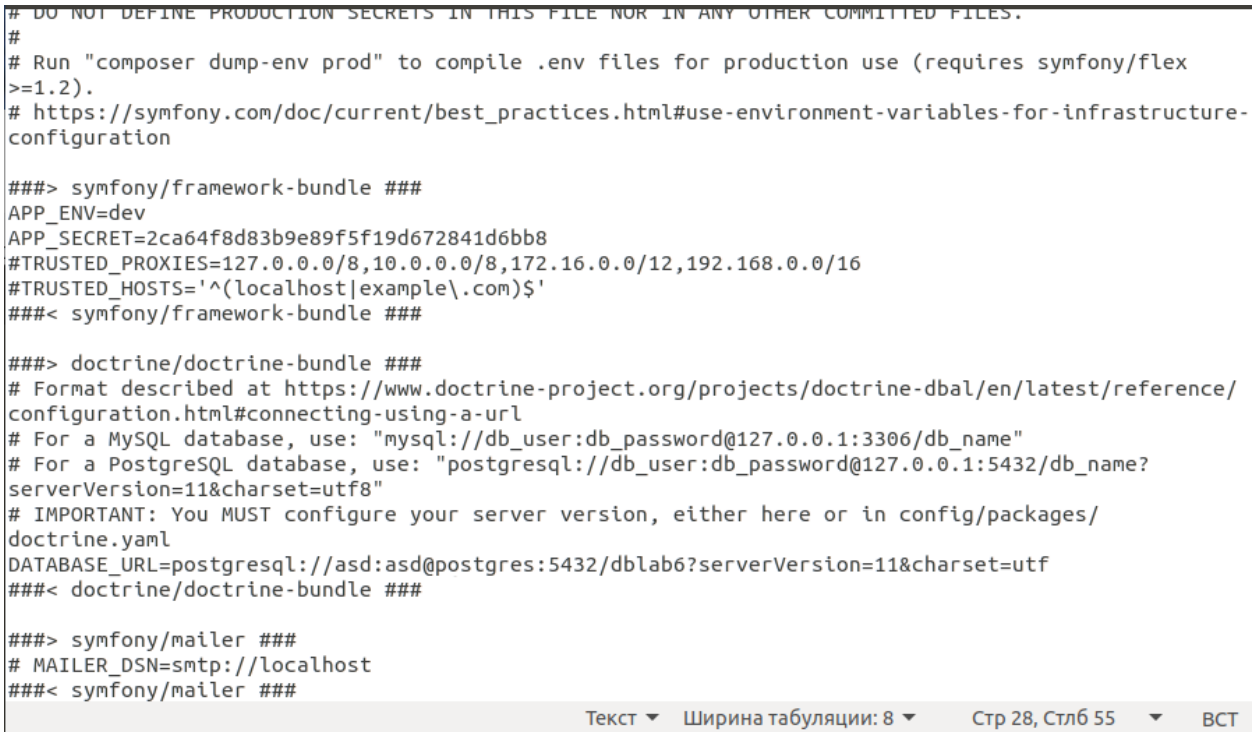
Далее установим postgresql и создадим базу данных tmpdb:



```
                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileg
es
-----------+----------+----------+-------------+-------------+-----------------
-----
 dblab6    | asd      | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
    +
           |          |          |             |             | postgres=CTc/post
gres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
    +
           |          |          |             |             | postgres=CTc/post
gres
(4 rows)

(END)
```

Рисунок 3 – Создание базы данных



```
# DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
#
# Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex
>=1.2).
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-
configuration

###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=2ca64f8d83b9e89f5f19d672841d6bb8
#TRUSTED_PROXIES=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
#TRUSTED_HOSTS='^(localhost|example\.com)$'
###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/
configuration.html#connecting-using-a-url
# For a MySQL database, use: "mysql://db_user:db_password@127.0.0.1:3306/db_name"
# For a PostgreSQL database, use: "postgresql://db_user:db_password@127.0.0.1:5432/db_name?
serverVersion=11&charset=utf8"
# IMPORTANT: You MUST configure your server version, either here or in config/packages/
doctrine.yaml
DATABASE_URL=postgresql://asd:asd@postgres:5432/dblab6?serverVersion=11&charset=utf
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###
# MAILER_DSN=smtp://localhost
###< symfony/mailer ###
```


Рисунок 4 – Подключение базы данных

Загружаем схему БД командой php bin/console doctrine:schema:create и заполняем данными с помощью команды php bin/console doctrine:fixtures:load. Проверяем работоспособность проекта:

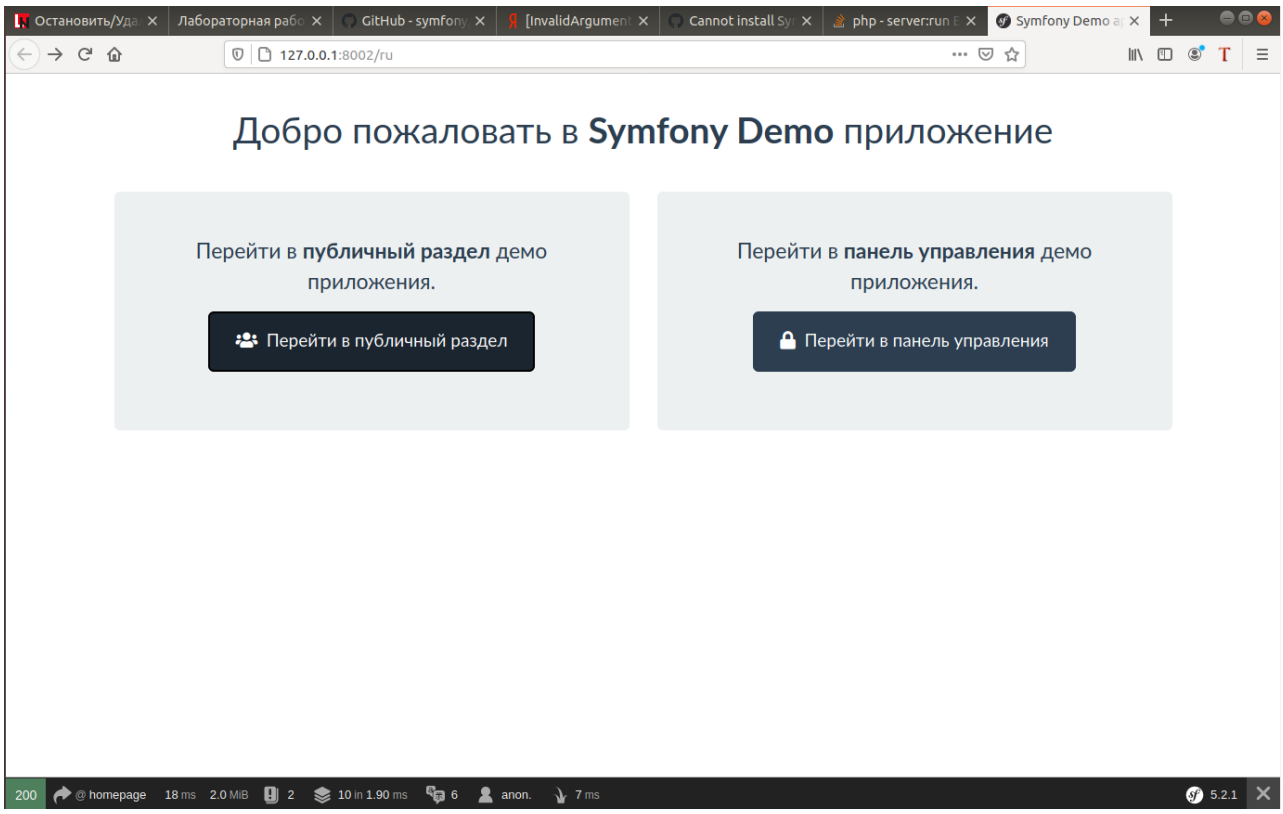


Рисунок 5 – Стартовая страница проекта

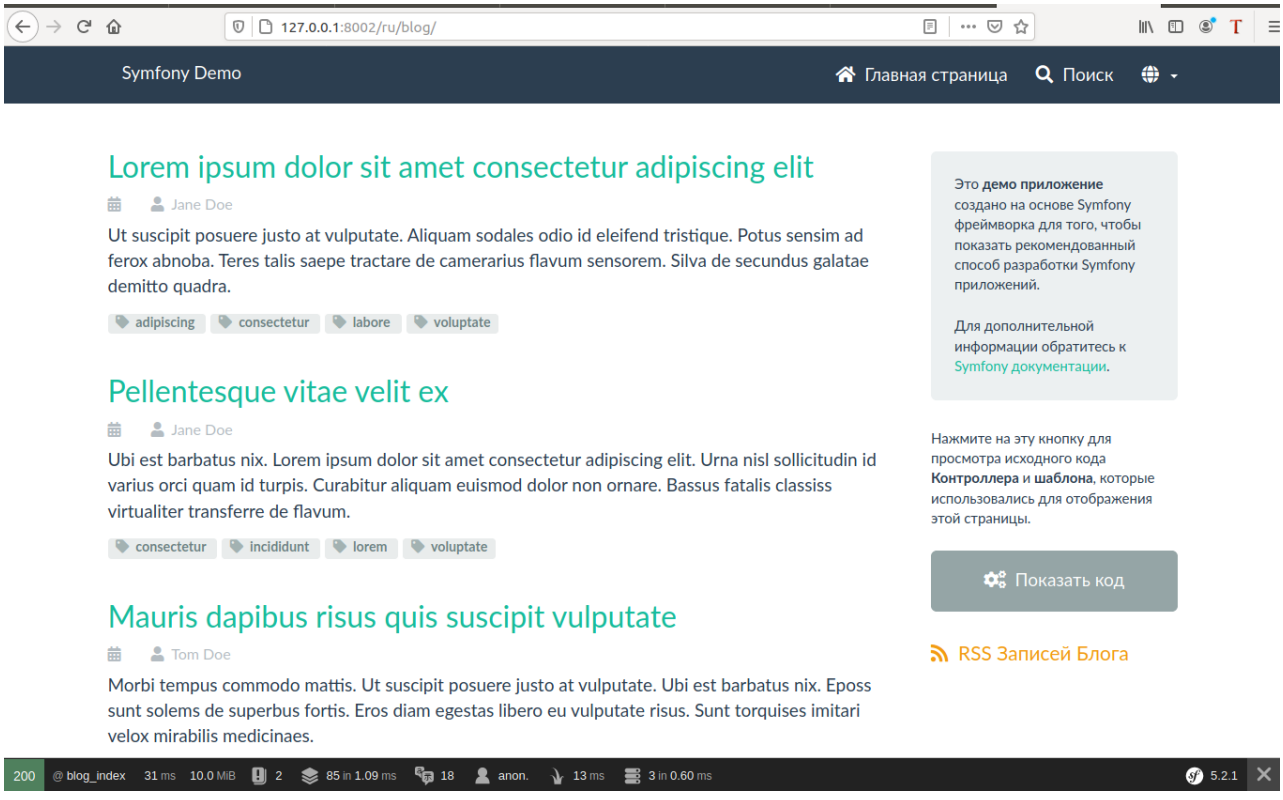


Рисунок 6 – Проверка работоспособности базы данных

Перейдем к настройке контейнеров. Первым делом создадим файл docker-compose.yml и заполним его следующим содержимым:

version: ”3”

services:

php-fmp:

container_name: php-fpm

build: context: .

dockerfile: docker/php.Dockerfile

volumes:

- ./:/var/www/symfony

- ./logs/symfony:/var/www/symfony/log

links:

- postgres

nginx:

container_name: nginx

build: context: .

dockerfile: docker/nginx.Dockerfile

ports: - ”8084:80”

volumes:

- ./:/var/www/symfony

- ./logs/nginx/:/var/log/nginx

links:

- php-fmp

postgres:

container_name: postgres

image: postgres

environment:

POSTGRES_DB: bdlab6

POSTGRES_USER: asd

POSTGRES_PASSWORD: asd

volumes: - ./data/postgresql/demo_db:/var/lib/postgresql/data

ports:

- 5432:5432

Теперь создадим папку docker и внутри нее создади 2 файла: nginx.Dockerfile и php.Dockerfile, а также каталог conf, содержащий файл конфигурации nginx vhost.conf.

Файл nginx.Dockerfile

```
FROM nginx:latest
COPY ./docker/conf/vhost.conf /etc/nginx/conf.d/default.conf
WORKDIR /var/www/symfony
```

Файл vhost.conf

```
server {
listen 80;
root /var/www/symfony/public;
server_name _;
error_log /var/log/nginx/symfony_error.log;
access_log /var/log/nginx/symfony_access.log;
location / {
try_files $uri /$uri /index.php?$query_string;
}
location ~ ^/index\.php(/|$) {
fastcgi_pass php-fpm:9000;
fastcgi_split_path_info ^(.+\.php)(/.*)$;
include fastcgi_params;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
fastcgi_param HTTPS off;
}
}
```

Файл php.Dockerfile

```
FROM php:7.4-fpm
WORKDIR /var/www/symfony
RUN apt-get update && apt-get install -y \
```

libpq-dev \

wget \

zlib1g-dev \

libmcrypt-dev \

libzip-dev RUN docker-php-ext-install pdo pdo_pgsql pgsql

CMD php-fpm

Далее запускаем все контейнеры в фоне, с помощью команды docker-compose up -d, переходим в контейнер с postgresql (команда docker exec -it postgres bash), внутри контейнера переходим в консоль psql и создаем БД demo_bd (команда create database demo_db;). Дальше переходим в контейнер с php и загрузить схему БД (команда php bin/console doctrine:schema:create) и данные для БД (команда php bin/console doctrine:fixtures:load).

После этого редактируем файл /etc/hosts и добавляем псевдоним адресу 127.0.0.1 demo-symfony.local.

Запускаем контейнеры командой docker-compose up -d и проверяем работу проекта:

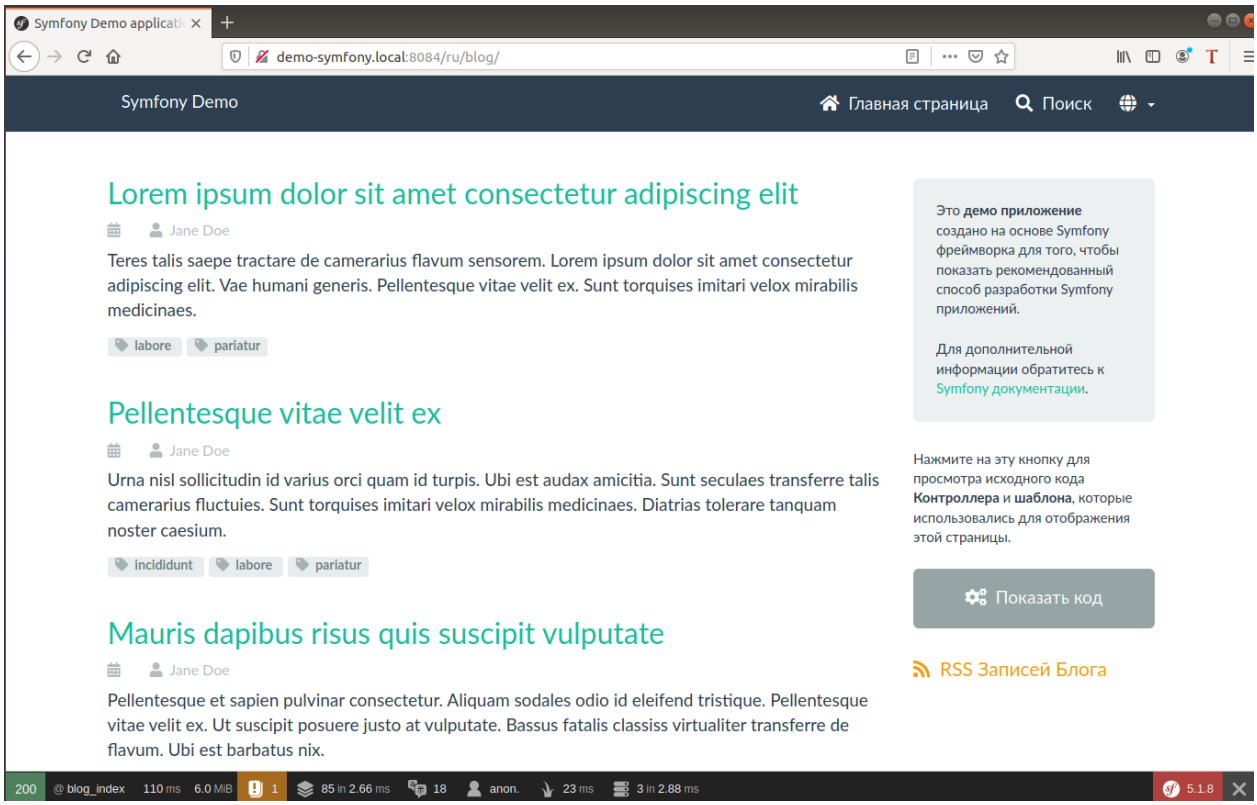


Рисунок 7 – Запуск проекта в контейнерах

Часть 2

Теперь удаляем конфигурацию контейнера с postgresql и подключим проект к локальной базе данных. Для этого узнаем ip локальной машины с помощью команды hostname -I | cut -d ’ ’ -f1 и добавим этот ip под псевдонимом bd в наш файл docker-compose.yml, получим следующее содержимое:

version: ”3”

services:

php-fmp:

container_name: php-fpm

build:

context: .

dockerfile: docker/php.Dockerfile

volumes:

- ./:/var/www/symfony

- ./logs/symfony:/var/www/symfony/log

extra_hosts:

- ”db:172.127.0.1”

nginx:

container_name:

nginx

build: context: .

dockerfile: docker/nginx.Dockerfile

ports: - ”80:80”

volumes:

- ./:/var/www/symfony

- ./logs/nginx/:/var/log/nginx

links: - php-fmp

Также изменим этот адрес в файле .env.

Теперь изменим конфигурацию локальной базы данных, так, чтобы она допускала подключение из контейнера. Для этого изменим файлы конфигурации /etc/postgresql/10/main/postgresql.conf и pg_hba.conf:

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

# - Connection Settings -

#listen_addresses = '*'          # 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
                                 # (change requires restart)
#bonjour = off                   # advertise server via Bonjour
                                 # (change requires restart)
#bonjour_name = ''               # defaults to the computer name
                                 # (change requires restart)

# - Security and Authentication -

#authentication_timeout = 1min          # 1s-600s
ssl = on
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # allowed SSL ciphers
```

Рисунок 8 – Файл Postgresql.conf

```
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
host    all             all             0.0.0.0./0              md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
```

Рисунок 9 – Файл pg_hba.cong

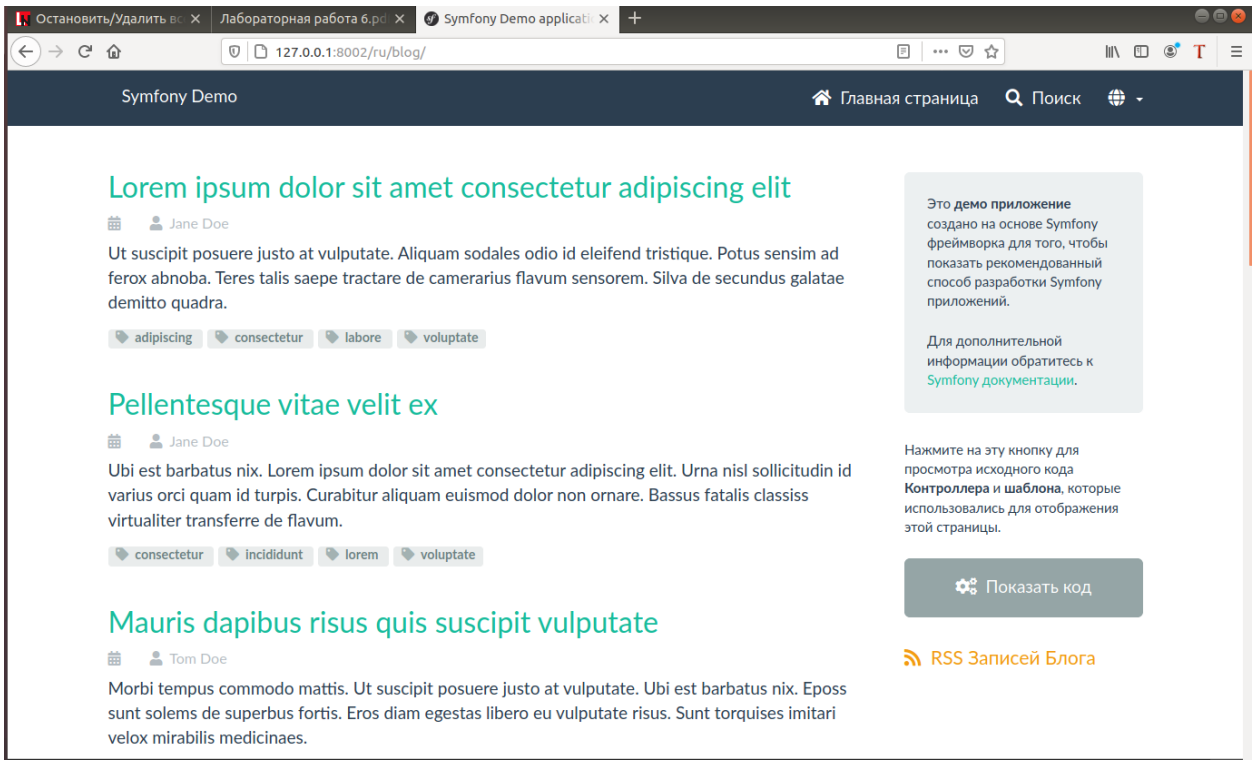


Рисунок 10 – Подключение к локальной базе данных