Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по Операционной системе Linux

Контейнеризация

Студент Лобов М.Ю.

Группа АИ-18

Руководитель Кургасов В.В.

Доцент, к.п.н.

Липецк 2021 г.

Оглавление

Цель работы

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

Ход работы

Перед тем, как переходить к выполнению лабораторной работы, нужно установить необходимое ПО, а именно:

- PHP и необходимые расширения (PDO-SQLite PHP, PHP-XML и PHP-pgsql);

- Composer;

- Symfony.

Затем создадим новый демо-проект с помощью команды **symfony new --demo my_project**. Теперь можем посмотреть структуру проекта:

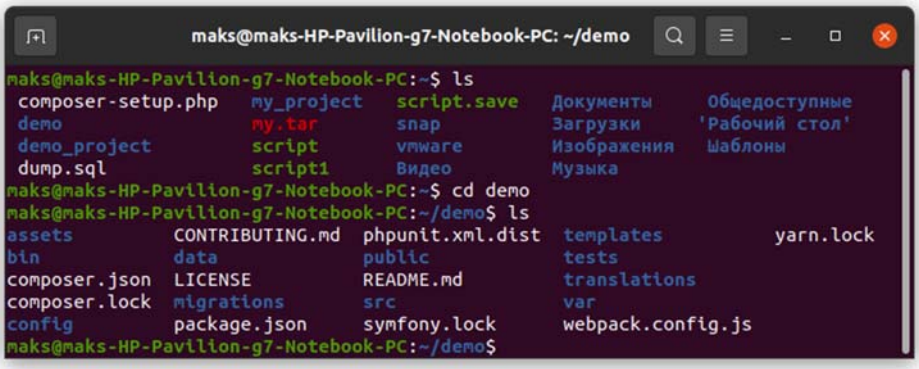

Рисунок 1 – Структура демо-проекта

После этого зайдём в папку с проектом и выполним команду **symfony serve**, чтобы запустить локальный сервер:

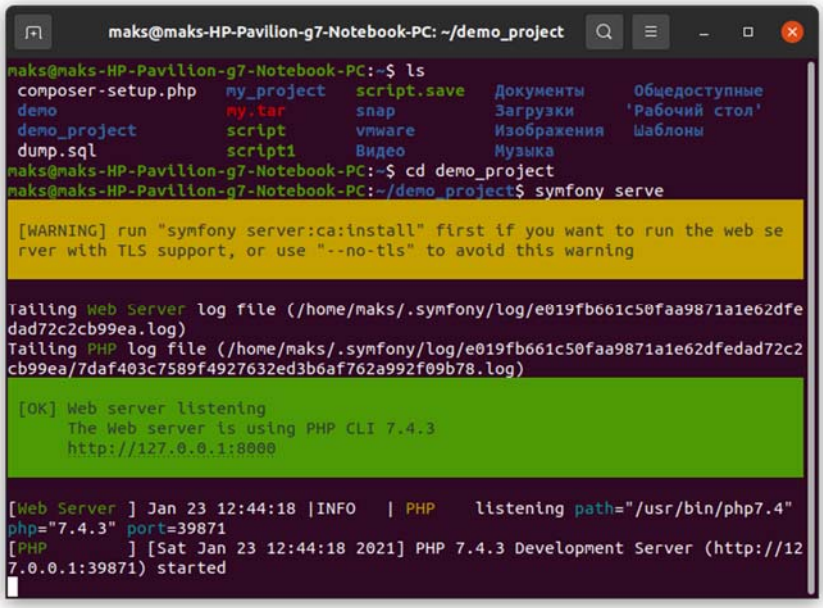

Рисунок 2 – Поднимаем локальный хост

Проверим работоспособность нашего проекта, для этого перейдём по адресу localhost:8000:

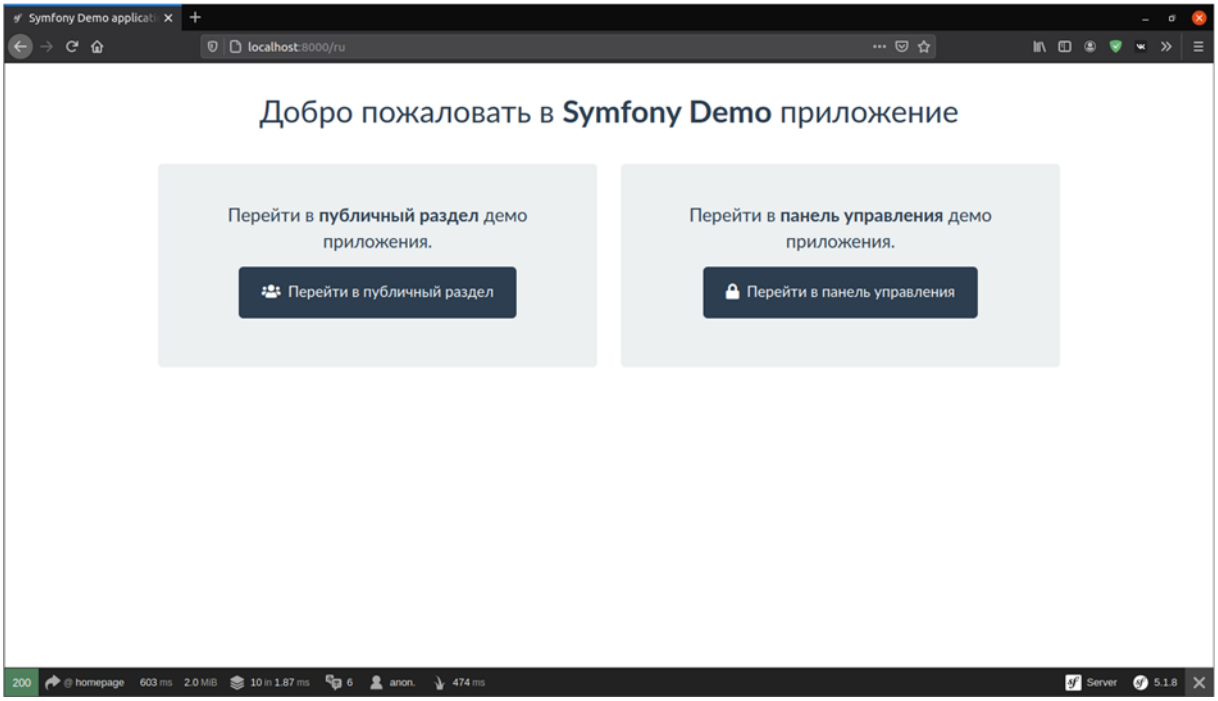

Рисунок 3 – Стартовая страница

Следующим шагом будет создание базы данных в postgres и подключение её к проекту:

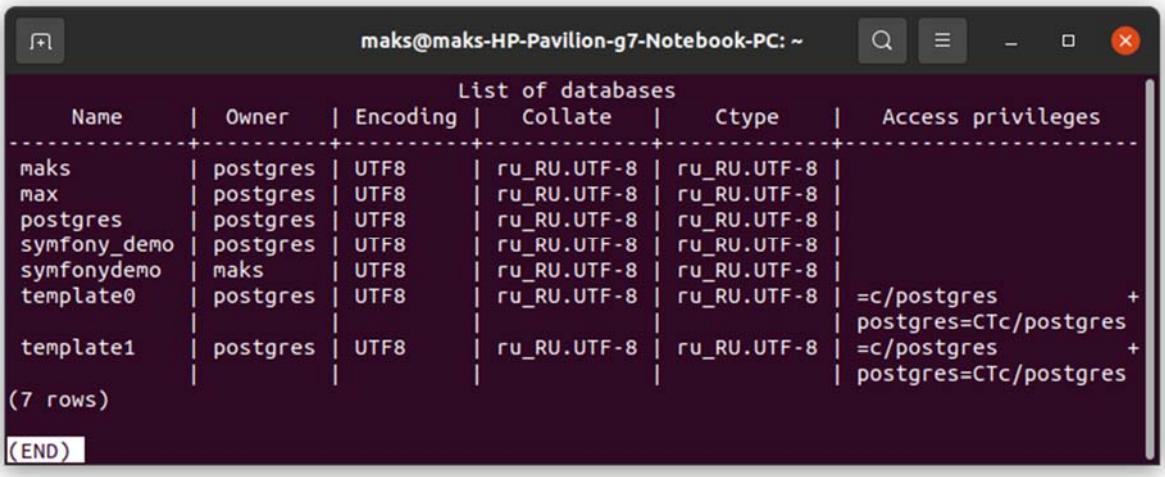

Рисунок 4 – Создание БД symfony_demo

После создания базы данных изменим файл окружения .env, где изменим строку подключения к БД:

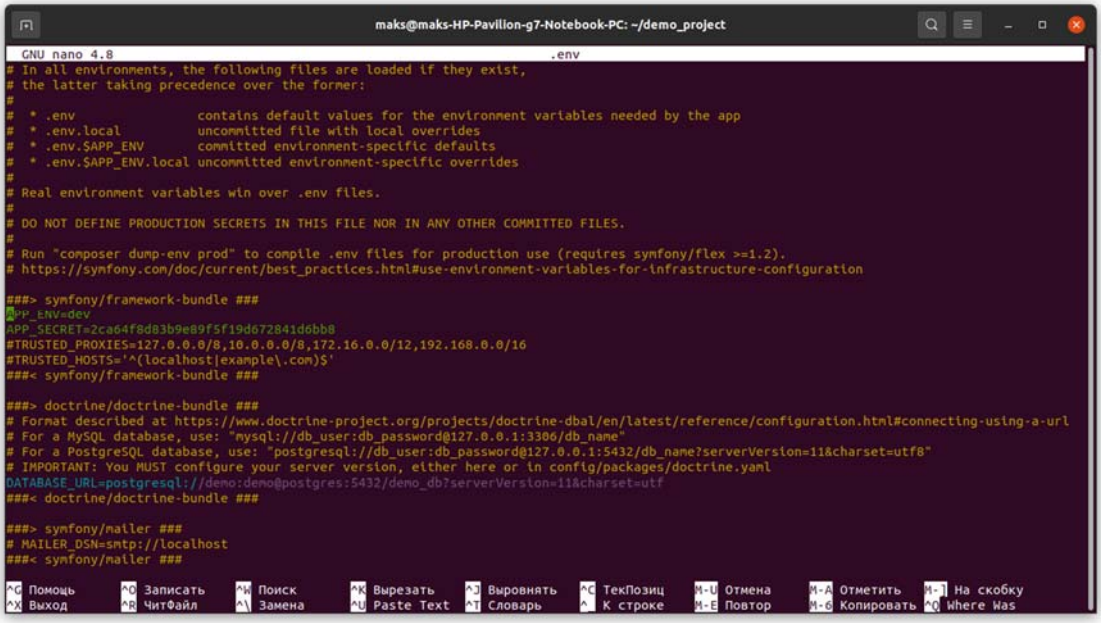

Рисунок 5 – Изменение файла .env

Теперь необходимо загрузить схему БД с помощью команды **php bin/console doctrine:schema:create** и заполняем БД с помощью команды **php bin/console doctrine:fixtures:load**.

Снова запустим локальный сервер командой **symfony serve** и проверим работоспособность проекта по адресу localhost:8000:

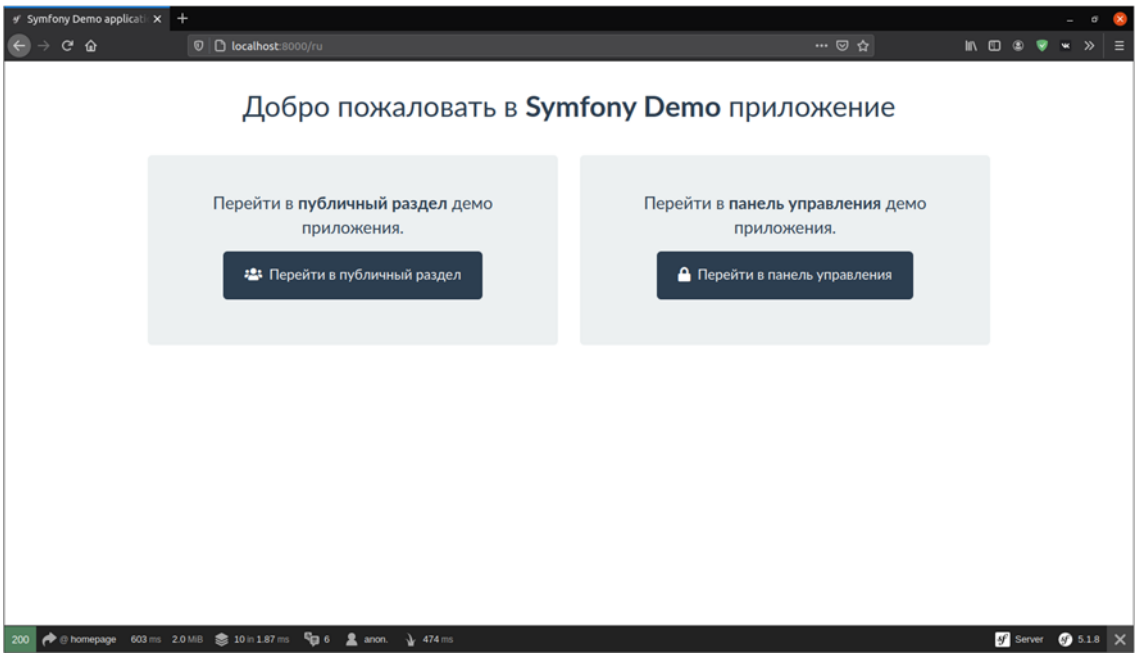

Рисунок 6 – Стартовая страница на postgres

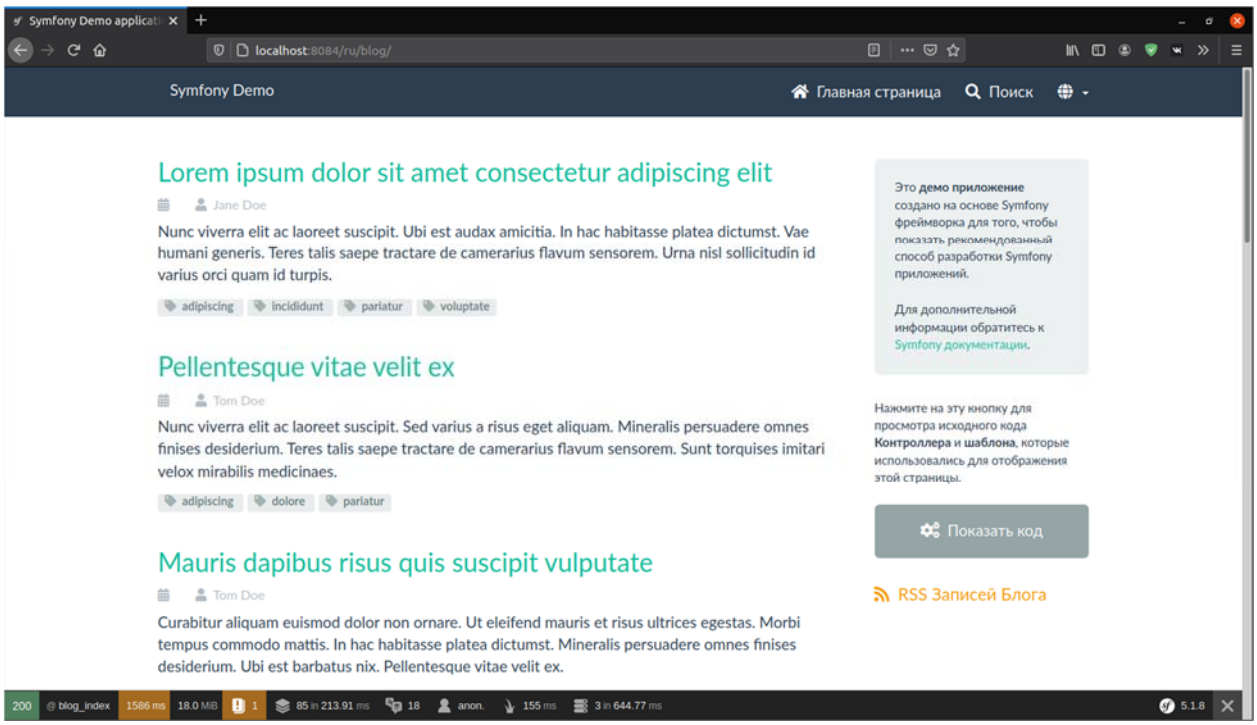

Рисунок 7 – Публичный раздел демо-приложения

Перейдём к настройке контейнеров. Для начала создадим файл docker-compose.yml и заполним его следующим образом:

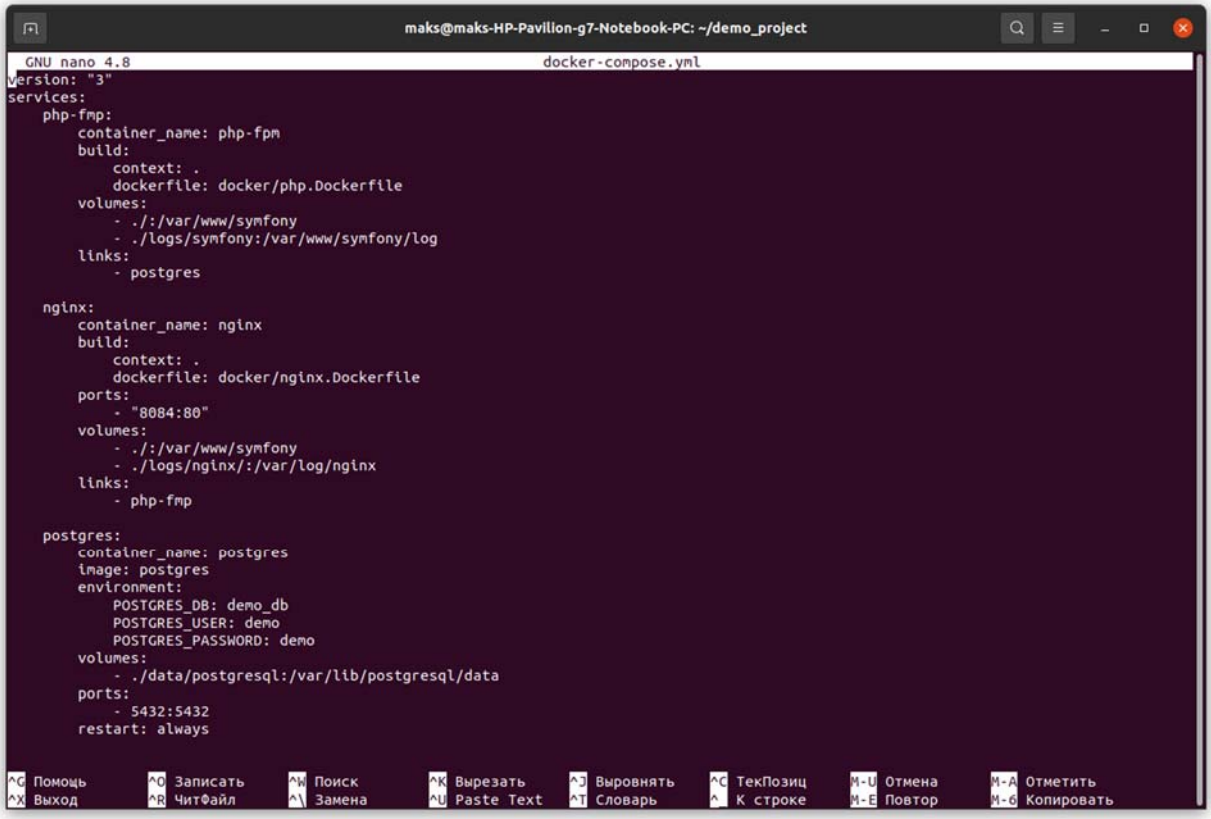

Рисунок 8 – Содержимое файла docker-compose.yml

После этого необходимо создать папку docker внутри нашего проекта и в этой папке создать файлы nginx.Dockerfile и php.Dockerfile и папку config, внутри которой будет располагаться файл vhost.conf. Перечисленные файлы имеют следующее содержание:

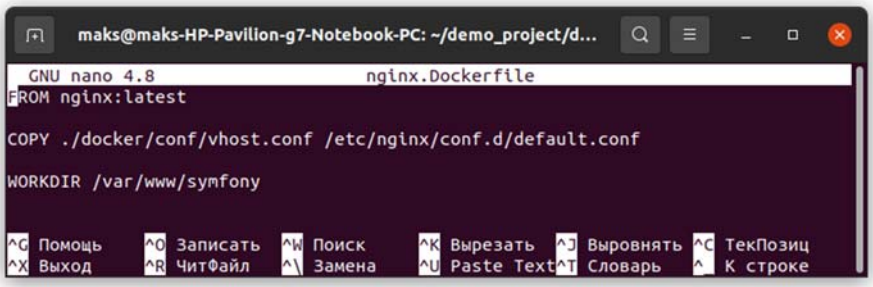

Рисунок 9 – Содержимое файла nginx.Dockerfile

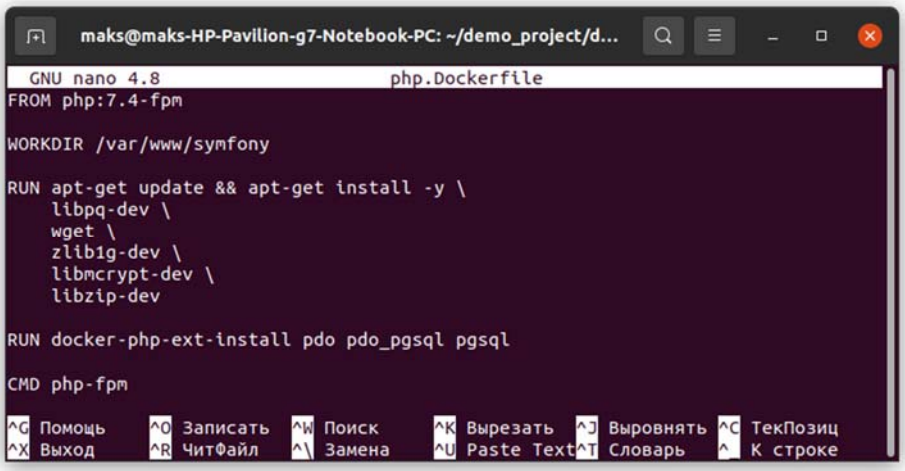

Рисунок 10 – Содержимое файла php.Dockerfile

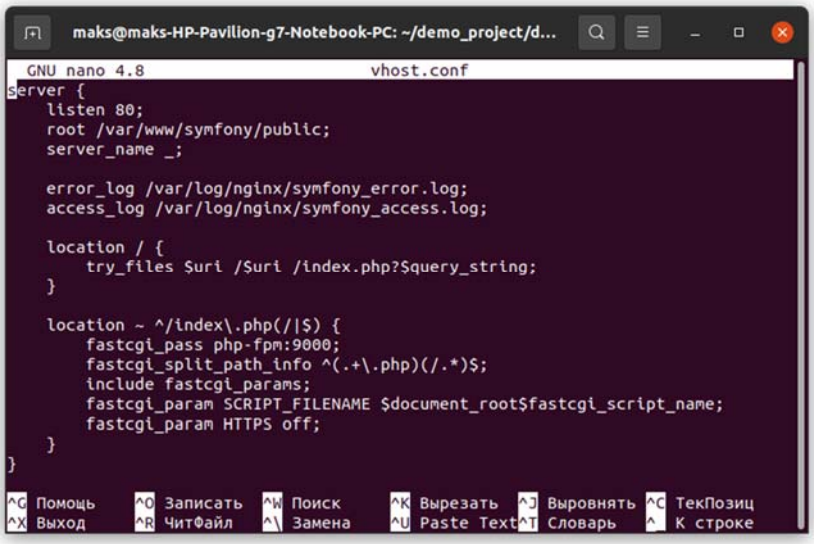

Рисунок 11 – Содержимое файла vhost.conf

Теперь запустим проект с помощью команды **docker-compose up –d** и проверим работоспособность (проект теперь находится по адресу localhost:8084):

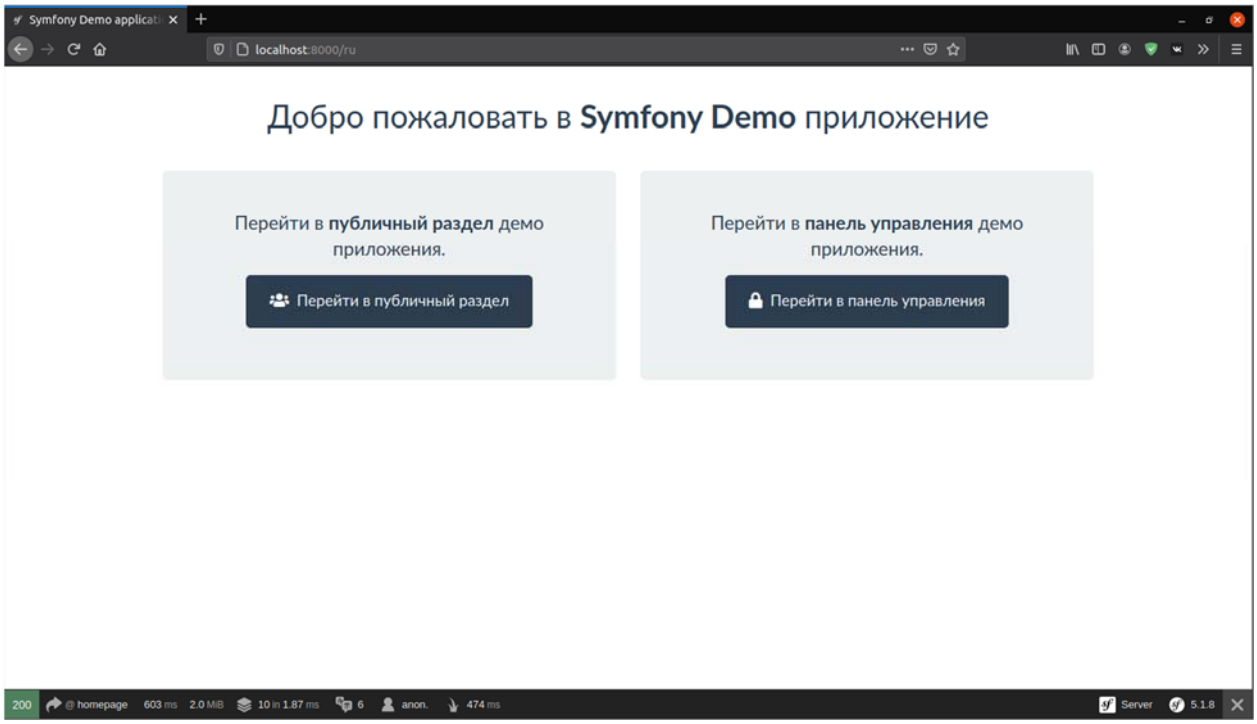

Рисунок 12 – Стартовая страница (запуск через контейнеры)

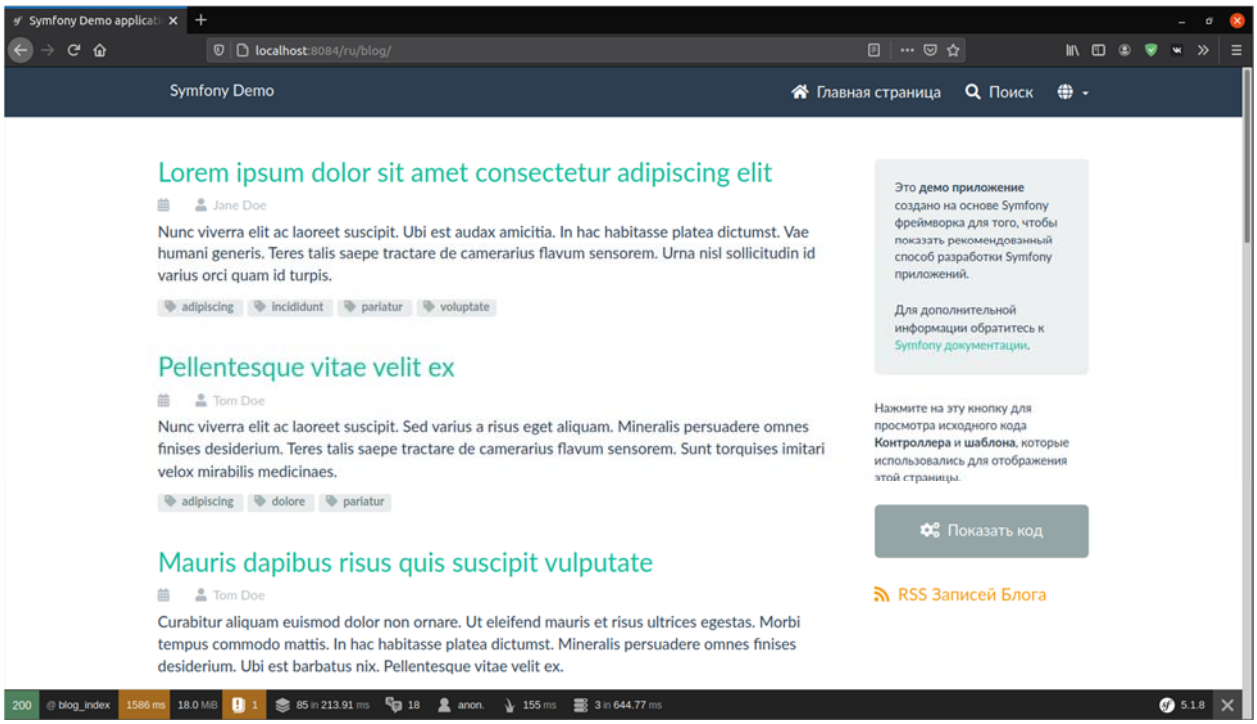

Рисунок 13 – Публичный раздел (запуск через контейнеры)

Далее необходимо подключить проект к локальной базе данных. Изменяем содержимое файла docker-compose.yml:

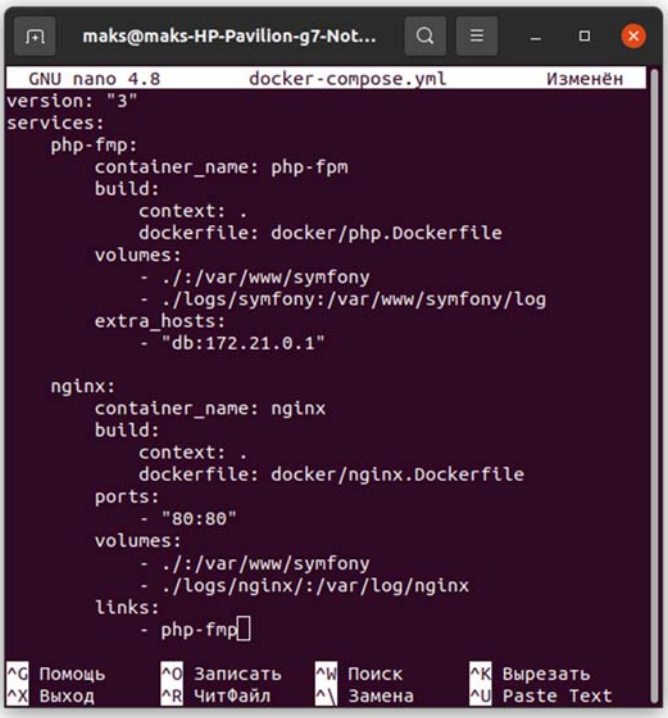

Рисунок 14 – Содержимое файла docker-compose.yml при переносе на локальную базу данных

Теперь изменим конфигурационные файлы /etc/postgresql/10/main/ postgresql.conf и pg_hba.conf, чтобы локальная база данных поддерживала запуск из контейнеров:

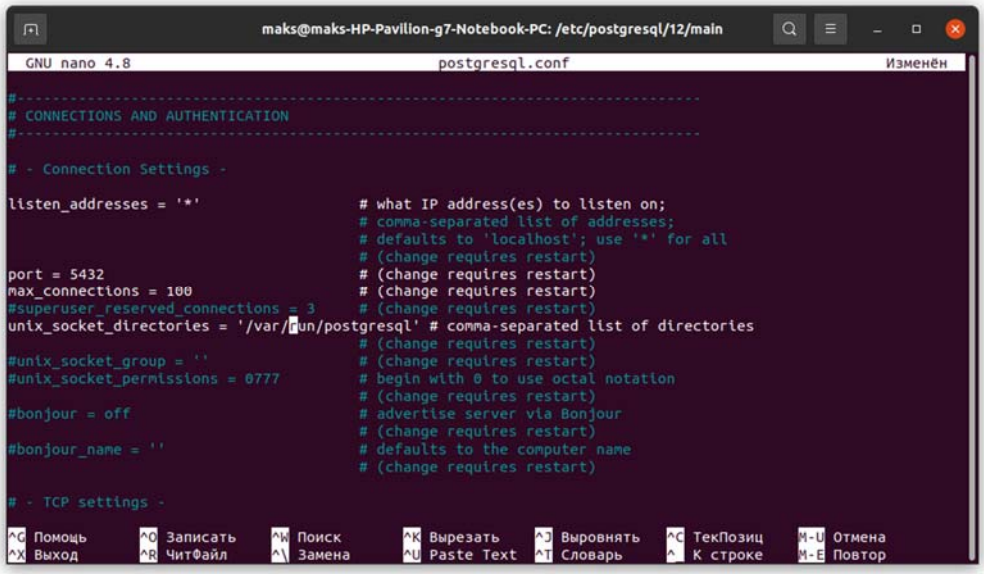

Рисунок 15 – Содержимое файла postgresql.conf

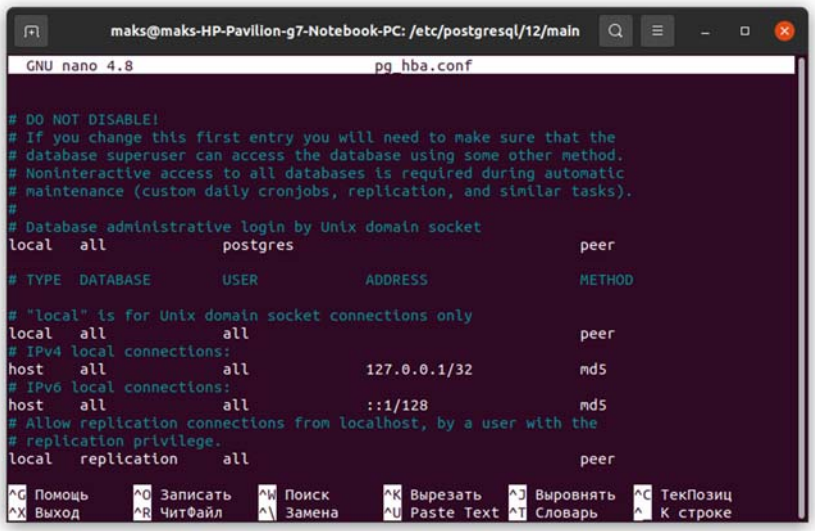

Рисунок 16 – Содержимое файла pg_hba.conf

Теперь перезапустим postgres с помощью команды **sudo service postgres restart** и проверим работоспособность проекта:

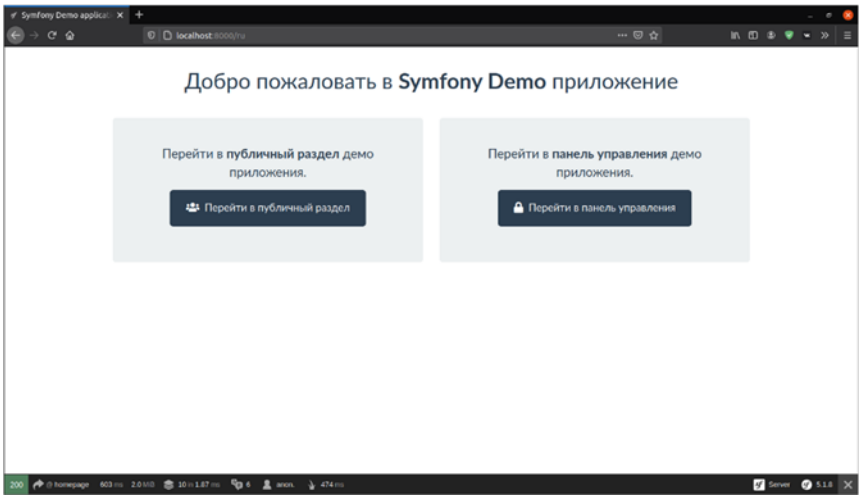

Рисунок 17 – Стартовая страница при запуске на локальной базе данных

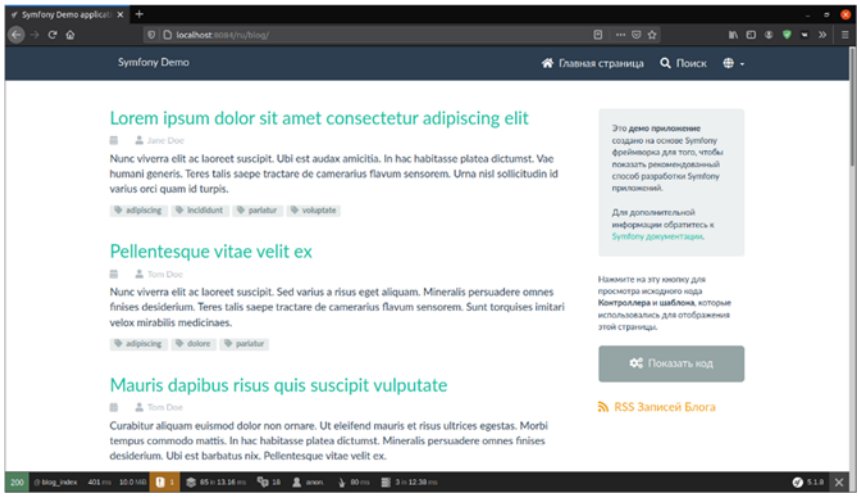

Рисунок 18 – Публичный раздел при запуске на локальной базе данных

Вывод

В ходе выполнения лабораторной работы были изучены современные методы разработки ПО в динамических и распределенных средах на примере контейнеров Docker.

Контрольные вопросы

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.

A. Меньшие накладные расходы на инфраструктуру

2. Назовите основные компоненты Docker.

B. Контейнеры

3. Какие технологии используются для работы с контейнерами?

C. Контрольные группы (cgroups)

4. Найдите соответствие между компонентом и его описанием:

— образы – доступные только для чтения шаблоны приложений;

— контейнеры – изолированные при помощи технологий операционной системы пользовательские окружения, в которых выполняются приложения;

— реестры (репозитории) – сетевые хранилища образов.

5. В чем отличие контейнеров от виртуализации?

Виртуальная машина – программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой целевой и исполняющая программы для гостевой платформы на платформе-хозяине (хосте) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы.

Виртуальные машины запускают на физических машинах, используя гипервизор.

В отличие от виртуальной машины, обеспечивающей аппаратную виртуализацию, контейнер обеспечивает виртуализацию на уровне операционной системы с помощью абстрагирования пользовательского пространства.

В целом контейнеры выглядят как виртуальные машины. Например, у них есть изолированное пространство для запуска приложений, они позволяют выполнять команды с правами суперпользователя, имеют частный сетевой интерфейс и IP-адрес, пользовательские маршруты и правила межсетевого экрана и т. д.

Одна большая разница между контейнерами и виртуальными машинами в том, что контейнеры разделяют ядро хоста с другими контейнерами.

6. Перечислите основные команды утилиты Docker с их кратким описанием.

— docker ps — показывает список запущенных контейнеров;

— docker pull — скачать определённый образ или набор образов (репозиторий);

— docker build — эта команда собирает образ Docker из Dockerfile и «контекста»;

— docker run — запускает контейнер, на основе указанного образа;

— docker logs — эта команда используется для просмотра логов указанного контейнера;

— docker volume ls — показывает список томов, которые являются предпочитаемым механизмом для сохранения данных, генерируемых и используемых контейнерами Docker;

— docker rm — удаляет один и более контейнеров;

— docker rmi — удаляет один и более образов;

— docker stop — останавливает один и более контейнеров;

— docker exec –it ... - выполняет команду в определенном контейнере

7. Каким образом осуществляется поиск образов контейнеров?

Сначала проверяется локальный репозиторий на наличия нужного контейнера, если он не найден локально, то поиск производится в репозитории Docker Hub.

8. Каким образом осуществляется запуск контейнера?

Для запуска контейнера его необходимо изначально создать из образа, поэтому изначально контейнер собирается с помощью команды docker build,а уже затем запускается с помощью команды docker run.

9. Что значит управлять состоянием контейнеров?

Это означает, что в любой момент времени есть возможность запустить, остановить или выполнить команды внутри контейнера.

10. Как изолировать контейнер?

Контейнеры уже по сути своей являются изолированными единицами, поэтому достаточно без ошибок сконфигурировать файлы Dockerfile и/или docker-compose.yml.

11. Опишите последовательность создания новых образов, назначение Dockerfile?

Производится выбор основы для нового образа на Docker Hub, далее производится конфигурация Dockerfile, где описываются все необходимые пакеты, файлы, команды и т.п.

Dockerfile — это текстовый файл с инструкциями, необходимыми для создания образа контейнера. Эти инструкции включают идентификацию существующего образа, используемого в качестве основы, команды, выполняемые в процессе создания образа, и команду, которая будет выполняться при развертывании новых экземпляров этого образа контейнера.

12. Возможно ли работать с контейнерами Docker без одноименного движка?

Да, если использовать Kubernetes

13. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes.

Kubernetes — открытое программное обеспечение для автоматизации развёртывания, масштабирования контейнеризированных приложений и управления ими. Поддерживает основные технологии контейнеризации, включая Docker, rkt, также возможна поддержка технологий аппаратной виртуализации.

— Nodes: Нода это машина в кластере Kubernetes.

— Pods: Pod это группа контейнеров с общими разделами, запускаемых как единое целое.

— Replication Controllers: replication controller гарантирует, что определенное количество «реплик» pod’ы будут запущены в любой момент времени.

— Services: Сервис в Kubernetes – это абстракция, которая определяет логический объединённый набор pod и политику доступа к ним.

— Volumes: Volume(раздел) это директория, возможно, с данными в ней, которая доступна в контейнере.

— Labels: Label’ы это пары ключ/значение которые прикрепляются к объектам, например pod’ам. Label’ы могут быть использованы для создания и выбора наборов объектов.

— Kubectl Command Line Interface: kubectl интерфейс командной строки для управления Kubernetes.