|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА– Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**Отчет по практическим работам №1-4**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы** ИКБО-03-21 Хречко Сергей Викторович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Благирев М.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

Оглавление

[Практическая работа № 1 3](#_Toc148445038)

[Цель работы 3](#_Toc148445039)

[Ход работы 4](#_Toc148445040)

[Вывод 6](#_Toc148445041)

[Ответы на вопросы к практической работе 6](#_Toc148445042)

[Использованные источники 6](#_Toc148445043)

[Ссылка на удаленный репозиторий 7](#_Toc148445044)

[Практическая работа № 2 8](#_Toc148445045)

[Цель работы 8](#_Toc148445046)

[Ход работы 9](#_Toc148445047)

[Вывод 13](#_Toc148445048)

[Ответы на вопросы к практической работе 13](#_Toc148445049)

[Использованные источники 13](#_Toc148445050)

[Ссылка на удаленный репозиторий 14](#_Toc148445051)

[Практическая работа № 3 15](#_Toc148445052)

[Цель работы 15](#_Toc148445053)

[Ход работы 16](#_Toc148445054)

[Вывод 23](#_Toc148445055)

[Ответы на вопросы к практической работе 23](#_Toc148445056)

[Использованные источники 23](#_Toc148445057)

[Ссылка на удаленный репозиторий 24](#_Toc148445058)

# Практическая работа № 1

# Цель работы

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера.

Docker контейнер должен включать:

* ОС Linux
* Веб-сервер Apache

Docker-compose:

* Docker-контейнер, созданный ранее
* Примонтированные тома
* Настройки портов

Конечной задачей студента будет корректный запуск приложенного php скрипта генерации страницы с характеристиками веб-сервера и его работа на созданном веб-сервере.

# Ход работы

В первую очередь был найден контейнер с сервером Apache и встроенным в него PHP (рис. 1). Также в докер-контейнере присутствует ОС Linux.

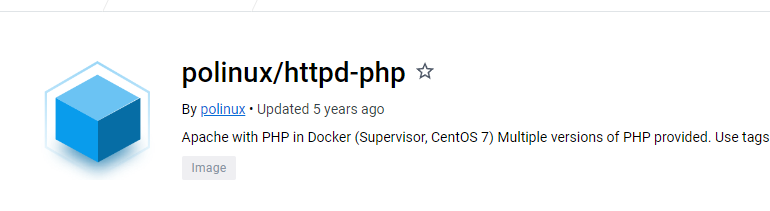


Рисунок 1 – Контейнер с Apache и PHP

На рисунке 2 представлен файл docker-compose. Видно примонтированный том и настроенные порты.

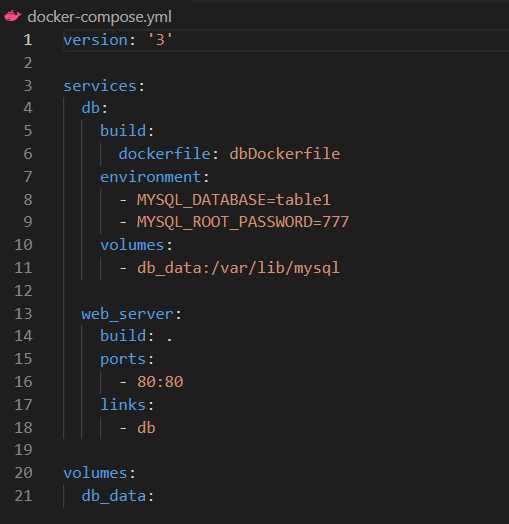


Рисунок 2 – Docker-compose проекта

На рисунке 3 представлена отображаемая страница на запущенном локальном сервере.

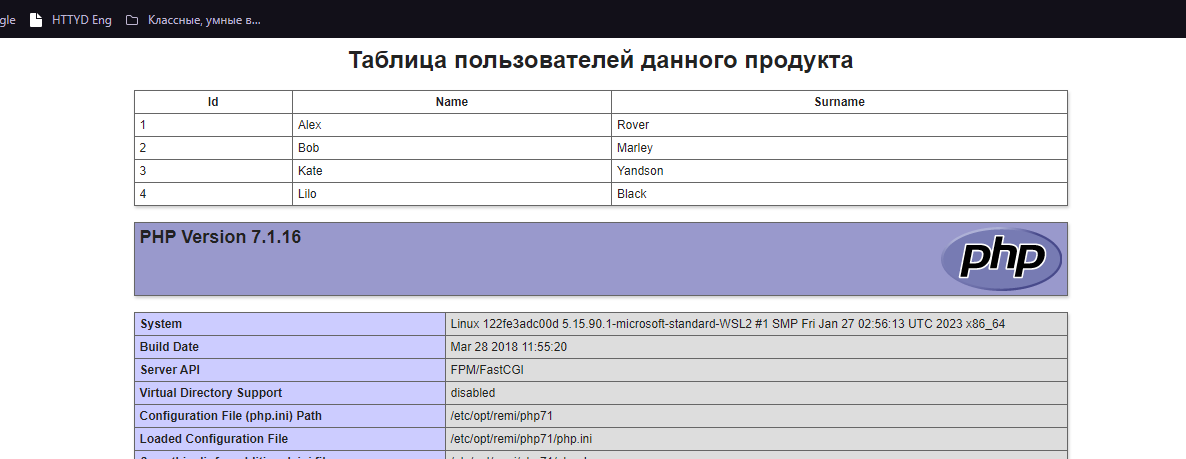


Рисунок 3 – Отображенная страница

# Вывод

В процессе выполнения работы были восстановлены знания о docker’е. Настроен контейнер с сервером на Apache. Был запущен тестовый скрипт предоставленный по заданию.

# Ответы на вопросы к практической работе

Что такое Apache и для чего он нужен?

Apache это веб-сервер, вернее ПО для его создание, бесплатное, открытое.

Что такое оркестрация контейнеров и для чего это нужно?

Это программная технология, которая позволяет управлять несколькими контейнерами автоматически. Позволяет организовать работу над проектом с большим количеством контейнеров.

Что такое сервер?

Сервер – это машина, которая организовывает работы интернет ресурса. (Не обязательно интернет ресурса, может быть просто особенным выделенным компьютером, в компьютерной сети).

# Использованные источники

1. Видео “Введение в Докер” на английском языке от создателя: [Introduction to Docker](https://www.youtube.com/watch?v=Q5POuMHxW-0) (https://www.youtube.com/watch?v=Q5POuMHxW-0)
2. Статья о назначении докера простыми словами: <https://habr.com/ru/post/309556/>
3. Более сложная и подробная статья про докер: <https://habr.com/ru/post/277699/>
4. Хорошая статья с пингвинами для прочтения после туториала по докеру: <https://habr.com/ru/post/250469/>
5. Официальная документация докера: <https://docs.docker.com/>
6. Статья о конкретном опыте использования докер контейнеров: <https://habr.com/ru/post/247969/>
7. Туториал по докеру: <https://badcode.ru/docker-tutorial-dlia-novichkov-rassmatrivaiem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/>
8. Туториал по докеру с Хабра: <https://habr.com/ru/post/310460/>
9. Шпаргалка с командами Docker: <https://habr.com/ru/company/flant/blog/336654/>
10. Ссылка на скачивание докера с официального сайте: <https://www.docker.com/products/docker-desktop>
11. Отличная статья про dockerfile: <https://habr.com/ru/company/ruvds/blog/439980/>
12. Установка и настройка PHP: <https://www.php.net/manual/ru/install.php>
13. Настройка среды PhpStorm и полезные фичи: <https://habr.com/ru/post/282003/>
14. Про docker compose: <https://habr.com/ru/company/ruvds/blog/450312/>
15. Docker hub: <https://hub.docker.com/>

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>

# Практическая работа № 2

# Цель работы

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера.

Docker контейнер веб сервера должен включать:

* ОС Linux
* Веб-сервер Apache

Docker контейнер базы данных должен включать:

* ОС Linux
* Базуданных Mysql

Docker-compose:

* Docker-контейнеры, созданные ранее
* Примонтированные тома
* Настройки портов
* Связь между контейнерами
* Переменные внешнего окружения
* Настройки базы данных

1. Создать sql скрипт инициализации, где студенту необходимо реализовать любую одну сущность (например товары или студенты).
2. Реализовать связь между базой данных и веб-сервером.

Требуется создать 4 страницы php, где на каждой странице будет реализована одна из CRUD операций с выбранной вами сущностью.

# Ход работы

Создание сущности «персонаж представлено на рисунке 4.

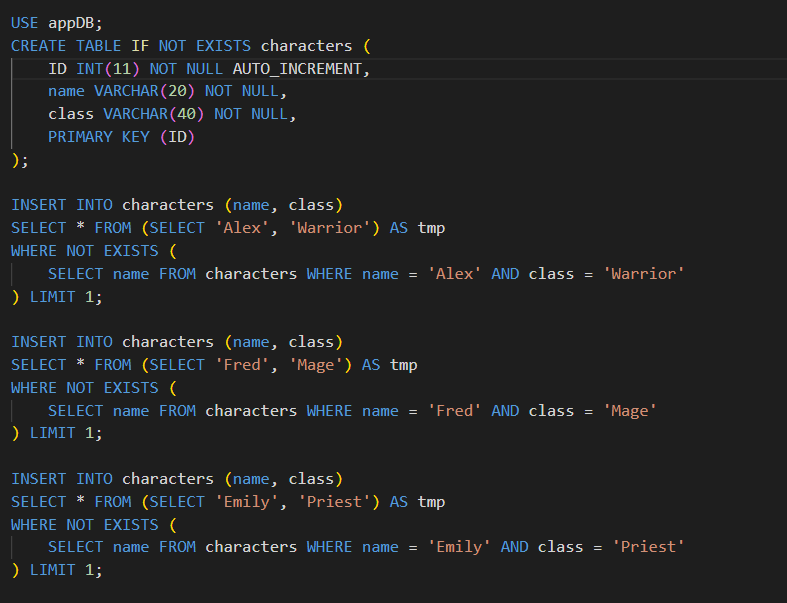


Рисунок 4 – Создание сущности «персонаж»

Далее создадим php скрипты для выполнения CRUD операций над созданной сущностью, скрипты представлены на рисунках 5 – 7. Скрипт для вывода сущности не нужно создавать отдельно, так как он аналогичен тому, что использовался ранее для вывода информации и находится в коде соответствующей страницы.



Рисунок 5 – Скрипт добавления персонажа

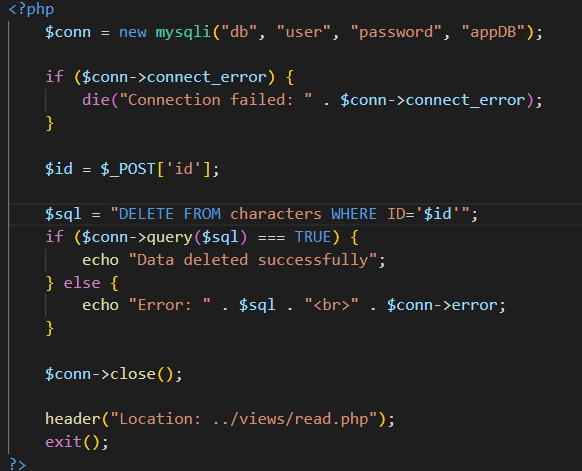


Рисунок 6 – Скрипт удаления персонаа

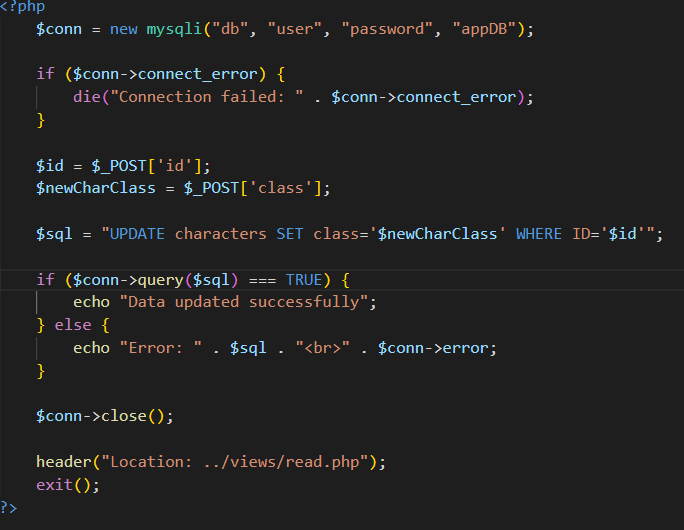
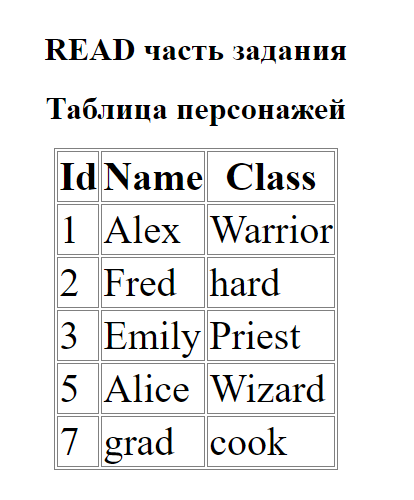
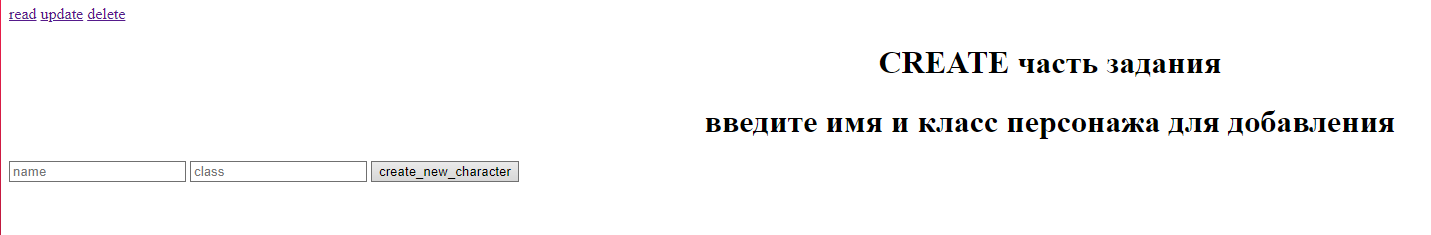
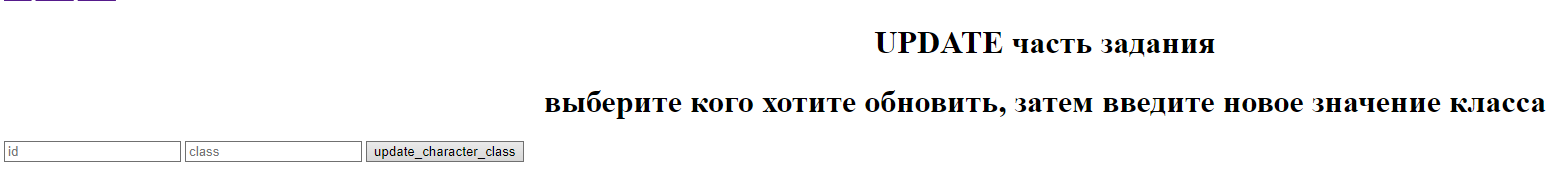


Рисунок 7 – Скрипт обновления данных персонажа









# Вывод

В процессе выполнения работы были созданы страницы для отображения CRUD операций. Были созданы php скрипты выполняющие CRUD операции.

# Ответы на вопросы к практической работе

В чём преимущество языка PHP для backend'а?

PHP реализует тонкий клиент, что значит, что вычисления не ложатся на клиент, это полезно, если клиентское оборудование слабое.

Как достигается тонкий клиент при использовании PHP?

PHP выполняет свой код на сервере, формируя страницу, которую и отправляет клиенту. Клиент не получает PHP код.

Что такое SSR?

Server-side-renderin – подход при котором страницы изначально рендерятся на сервере, а затем отправляются клиенту.

# Использованные источники

1. Хабр статья со шпаргалками по MySql: https://habr.com/ru/articles/564390/
2. Официальная документация php по работе с Mysql: https://www.php.net/manual/ru/book.mysql.php
3. Пример настройки конфигурации с использованием MySql: https://miac.volmed.org.ru/wiki/index.php/Docker-compose\_%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0\_%D0%B4%D0%BB%D1%8F\_%D1%81%D0%B0%D0%B9%D1%82%D0%B0\_NGINX\_%2B\_MYSQL\_%2B\_PHP-FPM
4. Статья про использование MySql в php: https://htmlacademy.ru/blog/php/mysql
5. Официальная документация докера: <https://docs.docker.com/>
6. Туториал по докеру: <https://badcode.ru/docker-tutorial-dlia-novichkov-rassmatrivaiem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/>
7. Туториал по докеру с Хабра: <https://habr.com/ru/post/310460/>
8. Шпаргалка с командами Docker: <https://habr.com/ru/company/flant/blog/336654/>
9. Docker hub: <https://hub.docker.com/>

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>

# Практическая работа № 3

# Цель работы

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера или взять из предыдущей практики.

Docker контейнер веб серверва должен включать:

• ОС Linux

• Веб-сервер Apache

Docker контейнер базы данных должен включать:

• ОС Linux

• Базу данных Mysql

Docker-compose:

• Docker-контейнеры, созданные ранее

• Примонтированные тома

• Настройки портов

• Связь между контейнерами

• Переменные внешнего окружения

• Настройки базы данных

1. Создать sql скрипт инициализации, где студенту необходимо реализовать любые две связанные сущности.

2. Требуется реализовать REST API для двух сущностей, где на запросы ответы должны быть в формате JSON.

Примерзапроса:`[GET] localhost:8080/api/orders.php?id=3`

3. Провести ручное тестирование с помощью POSTMAN, CURL или Insomnia.

# Ход работы

Создание сущности «питомец», связанной с персонажем представлено на рисунке 8.

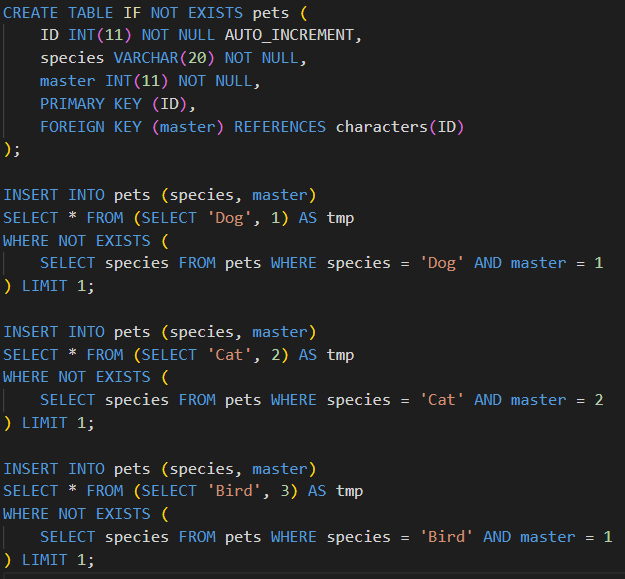


Рисунок 8 – Создание сущности «питомец»

Далее была создана система api php скриптов, файлы представлены на рисунке 9.

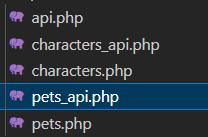


Рисунок 9 – Файлы api

На Рисунках 10 – 14 продемонстрированы операции над сущностью «персонаж». На 14-ом рисунке показан результа, который можно сравнить с рисунком 10.

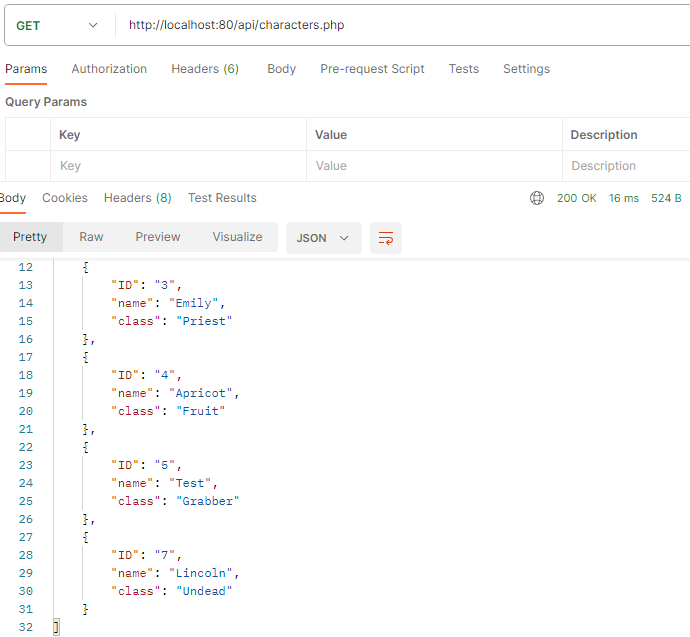


Рисунок 10 – Получение персонажей

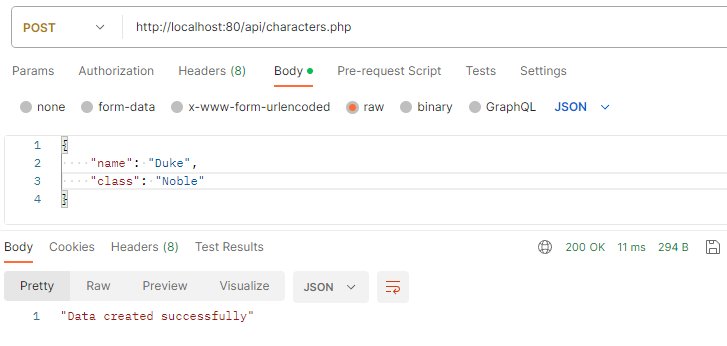


Рисунок 11 – Создание персонажа

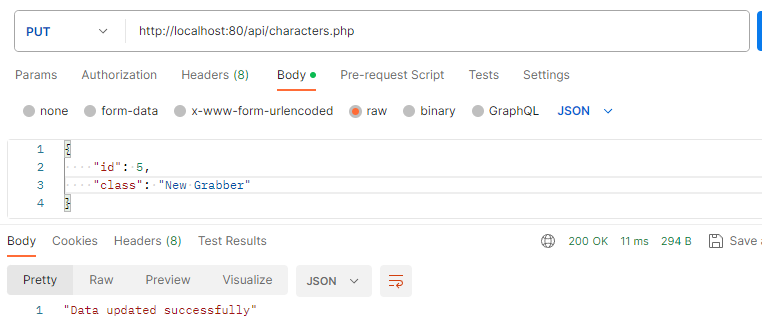


Рисунок 12 – Обновление персонажа

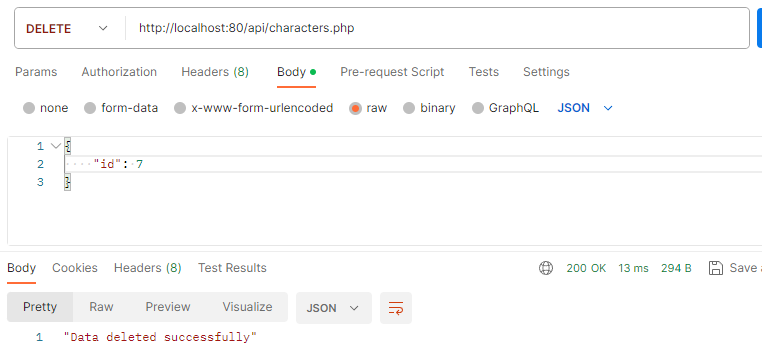


Рисунок 13 – Удаление персонажа

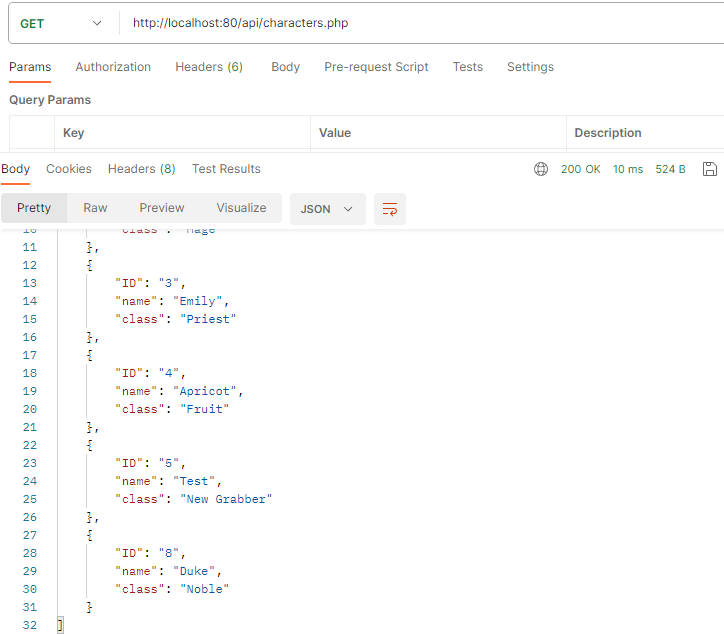


Рисунок 14 – Получение персонажей после проделанных операций

Проделаем то же самое с сущностью «Питомец», результаты представлены на рисунках 15 – 19.

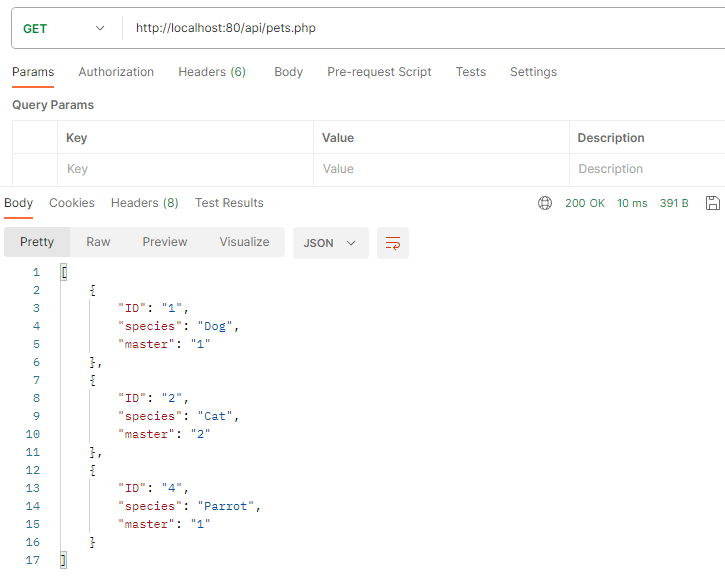


Рисунок 15 – Получение питомцев

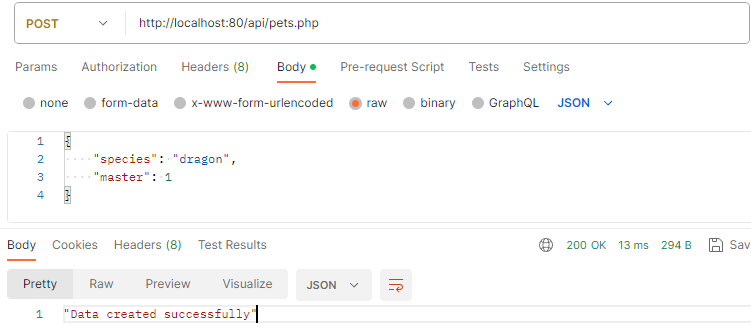


Рисунок 16 – Создание питомца

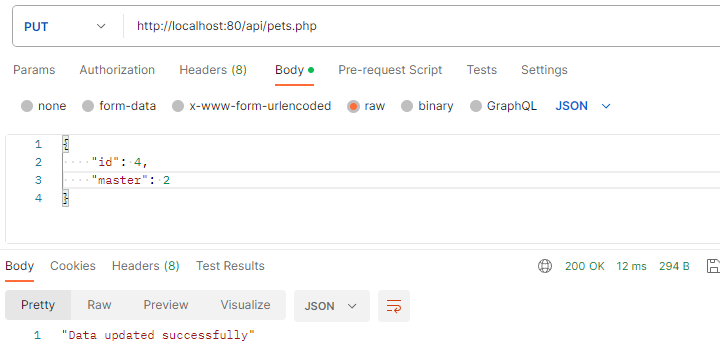


Рисунок 17 – Обновление питомца

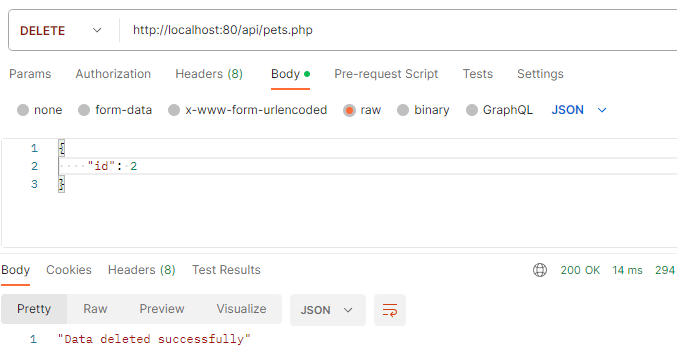


Рисунок 18 – Удаление питомца

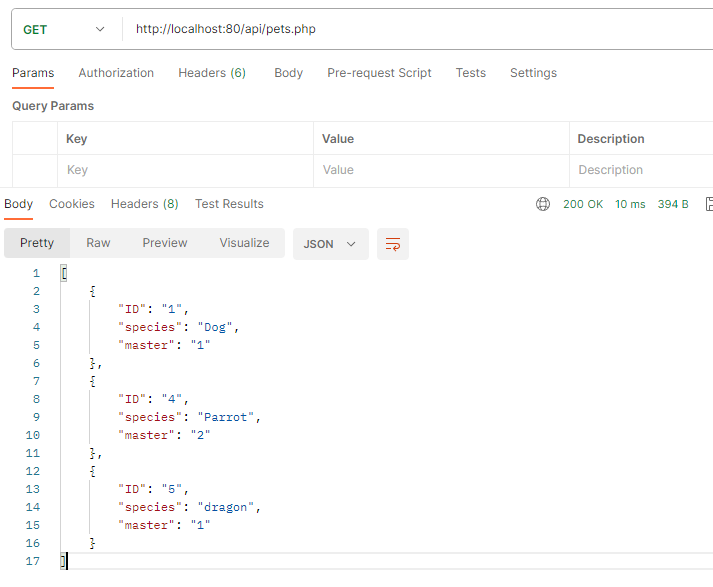


Рисунок 19 – Получение питомцев после операций

# Вывод

В результате выполнения работы были созданы две связанных сущности. Также была создана api для работы с ними. Было проведено тестирование с помощью postman.

# Ответы на вопросы к практической работе

Что такое сущность?

Это объект, который представляет собой некоторый вид данных, которые могут быть сохранены и извлечены из базы данных. Сущность в backend содержит поля и методы, которые позволяют работать с данными в базе данных.

Какие методы существуют в протоколе http?

OPTIONS, GET, HEAD, POST, PUT, PATCH, DELETE, TRACE, CONNECT.

Что такое контроллер?

Функции для получения запрашиваемых данных из моделей, создание HTML страницы, отображающей данные, и возращение их пользователю для просмотра в браузере.

# Использованные источники

1. Пример создания restapiна php:https://cms.by/blog/php/primer-prostogo-sozdaniya-rest-api-na-php/
2. Примеры restapiприложений: https://github.com/topics/php-rest-api
3. Гайд как пользоваться postman:https://testengineer.ru/gajd-po-testirovaniyu-v-postman/
4. Статьяна хабре про принципы построения REST JSON API: https://habr.com/ru/articles/447322/
5. Шпаргалка по безопасности restapi: https://habr.com/ru/companies/acribia/articles/453384/
6. Гайд по использованию curl: https://megahub.me/cheat-sheets/curl

Лучшие практики restapi: https://habr.com/ru/companies/piter/articles/511382/

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>

# Практическая работа № 4

# Цель работы

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера.

Docker контейнер веб сервера должен включать:

• ОС Linux

• Веб-сервер Apache

Docker контейнер базы данных должен включать:

• ОС Linux

• Базуданных Mysql

Docker контейнер NGINX должен включать:

• ОС Linux

• NGINX

Docker-compose:

• Docker-контейнеры, созданные ранее

• Примонтированные тома

• Настройки портов

• Связь между контейнерами

• Переменные внешнего окружения

• Настройки базы данных

В рамках данного курса вам предлагается создать один большой проект маркетплейса, некоторые из практик, например, как эта, содержат кусок того проекта.

1. Создать 2 статичные html страницы для маркетплейса, где на одной странице будет информация по доставке, а на другой будут контакты и реквизиты вашего маркетплейса. (Эти страницы будут в финальном проекте)

2. Нужно сделать таким образом, чтобы статические страницы хранились в контейнере вместе с сервисом обратного прокси-сервера, а также настроить NGINX так, чтобы он мог их выдавать по запросу.

3. Создать 2 статичные html страницы. Динамические страницы могут быть взяты из предыдущих практических заданий или же написаны с нуля. Однако необходимо сделать так, чтобы запросы и ответы на динамические страницы проходили через NGINX.

# Ход работы

Сначала перенастроим наш проект, добавив в него nginx и настроив docker-compose. На рисунке 20 представлен новый docker-compose.

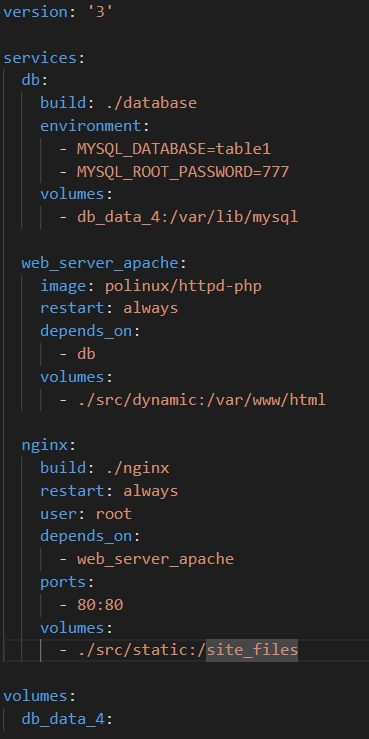


Рисунок 20 – Новый docker-compose

Тепреь настроим nginx, для этого настроим dockerfile в папке nginx, а также файл nginx.conf. Данная настройка представлена на рисунках 21 – 22.

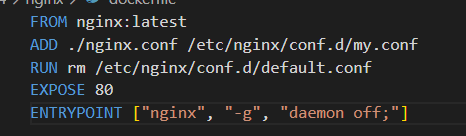


Рисунок 21 – Dockerfile для nginx

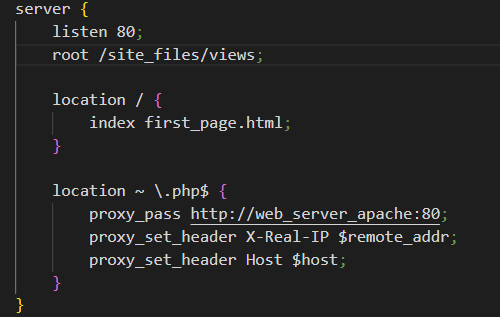


Рисунок 22 – Файл nginx.conf

На рисунке 23 представлены файлы составляющие статичную часть нашего сайта.

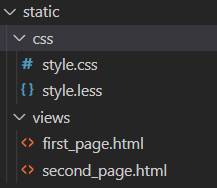


Рисунок 23 – «Статичные» файлы

На рисунках 24 и 25 представлены страницы.



Рисунок 24 – Первая страница



Рисунок 25 – Вторая страница

При тестировании обнаруживались «плавающие» багги, в результате которых была ошибка 502 при переходе на динамические страницы, помогло удаление контейнера и создание заново.

На рисунке 26 показано возвращение на php страницы.

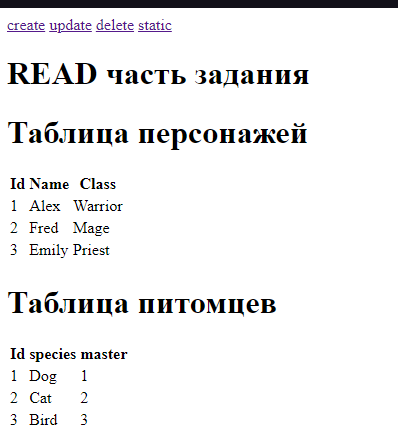


Рисунок 26 – Динамические страницы

# Вывод

В результате выполнения работы был настроен сервер nginx. Были созданы статичные html страницы, выдаваемые сервером nginx. Сервер nginx был настроен как обратный прокси-сервер.

# Ответы на вопросы к практической работе

Что такое обратный прокси сервер?

Это сервер, который действует от имени серверов внутри защищенной сети и принимает запросы от клиентов, направляя их к соответствующим внутренним серверам.

Как можно использовать NGINX?

Широко используется для обслуживания динамических веб-сайтов и приложений. Он также может быть использован в качестве прокси-сервера для передачи запросов на серверы приложений.

Как работает кеширование в NGINX?

Кэширование работает за счёт сохранения HTTP ответа на определенное время, а при повторном обращении к ресурсу, отдачи его из кэша без проксирования запроса на бекенд. Кэширование, даже на непродолжительный период, даст значительный прирост к количеству обрабатываемых запросов сервером.

Как в конфигурации NGINXдобавить несколько веб-серверов динамических страниц?

В конфигурации NGINX можно добавить несколько веб-серверов динамических страниц с помощью блока server. В этом блоке можно указать разные порты для разных сервисов и тогда NGINX будет работать сразу с несколькими ресурсами.

# Использованные источники

1. Официальная документация NGINX:https://nginx.org/ru/
2. Статья по установке NGINX: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04-ru
3. Статья по настройке NGINXв качестве обратного прокси сервера: https://help.reg.ru/support/servery-vps/oblachnyye-servery/ustanovka-programmnogo-obespecheniya/kak-nastroit-nginx-v-kachestve-obratnogo-proksi-dlya-apache
4. Сбор мини статьей по настройке :https://www.dmosk.ru/categories.php?object=nginx
5. Шпаргалка по NGINX: http://idoka.ru/nginx-tips-and-tricks/

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>