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

Институт Информационных технологий

Кафедра Интрументального и прикладного программного обеспечения

**Отчет по практической работе**

по дисциплине «Разработка серверных частей интернет ресурсов»

|  |  |
| --- | --- |
| **Выполнили:**  Студенты группыИКБО-03-21 | Теремязев А.О. |
| **Проверил:**  Ассистент | Благирев М.М. |

Москва 2023 г.

**Содержание**

[Практическая работа №1 2](#_Toc147914359)

[Теоретическое введение 2](#_Toc147914360)

[Задание 2](#_Toc147914361)

[Ход работы 3](#_Toc147914362)

[Вывод 6](#_Toc147914363)

[Список источников 6](#_Toc147914364)

[Ответы на вопросы 8](#_Toc147914365)

[Практическая работа №2 9](#_Toc147914366)

[Теоретическое введение 10](#_Toc147914367)

[Задание 10](#_Toc147914368)

[Ход работы 10](#_Toc147914369)

[Вывод 10](#_Toc147914370)

[Список источников 10](#_Toc147914371)

[Ответы на вопросы 10](#_Toc147914372)

[Практическая работа №3 10](#_Toc147914373)

[Теоретическое введение 10](#_Toc147914374)

[Задание 10](#_Toc147914375)

[Ход работы 10](#_Toc147914376)

[Вывод 10](#_Toc147914377)

[Список источников 10](#_Toc147914378)

[Ответы на вопросы 10](#_Toc147914379)

# Практическая работа №1

## Теоретическое введение

Backend – это серверная часть веб-приложения, которая обычно отвечает за обработку запросов пользователя, взаимодействие с базой данных, доступ к файлам и другим ресурсам. Она обеспечивает работу веб-приложения с общими ресурсами, включая базы данных, сервера и другие системы. Backend веб-приложения могут быть написаны на различных языках программирования, включая PHP, Java, Golang, Python и т.д.

Backend является одной из ключевых компонентов веб-приложения. Он обеспечивает логику, необходимую для обработки запросов, а также управление данными и их хранение. Backend веб-приложения работает в связке с frontend (клиентской частью), которая отображает пользовательский интерфейс и обрабатывает пользовательский ввод.

Таким образом, backend является неотъемлемой частью веб-приложения и обеспечивает его надежную работу и безопасность. Разработка backend части веб-приложения — это высоко востребованный навык в сфере IT.

На данном курсе вам необходимо будет выполнить 16 практических работ, которые разделены на 4 блока по языкам программирования. У первого блока практик язык программирования PHP. В ходе работы по данной практике вы должны будете создать первый простейший веб-сервер.

## Задание

Необходимо создать конфигурацию docker-compose для вашего веб-сервера.

Docker контейнер должен включать:

* ОС Linux
* Веб-сервер Apache

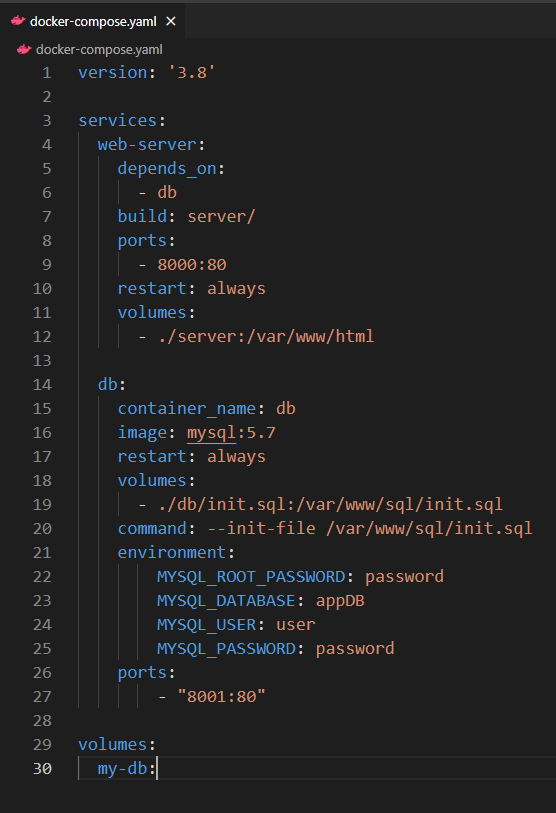
Docker-compose:

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

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

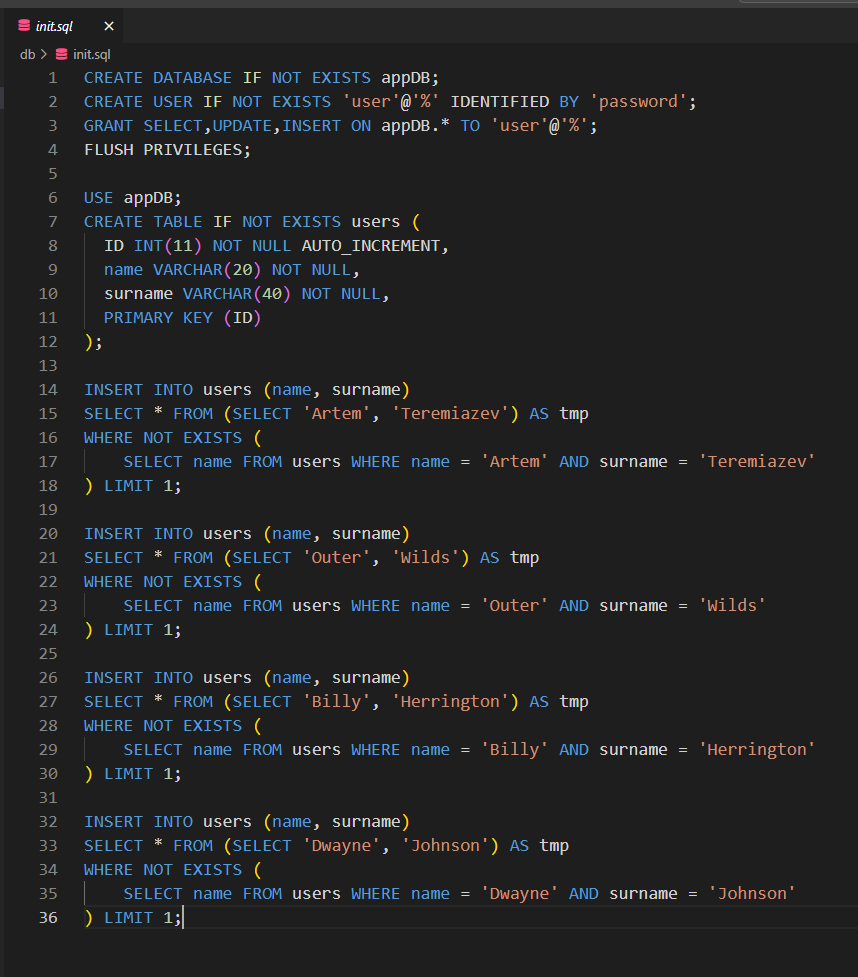
## Ход работы

Создадим файл docker-compose в который впишем сервисы, которые будут развернуты: web-server (веб сервер), bd (база данных) (рисунок 1).



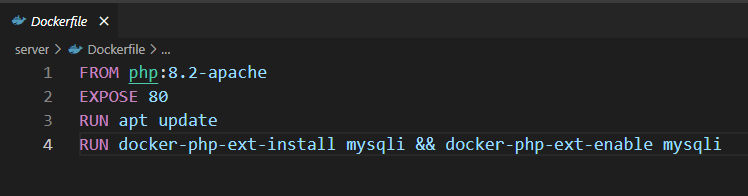
1. Файл docker-compose

Для инициализации базы данных будем использовать инициализирующий файл init.sql, который создает базу данных и записывает в нее информацию (рисунок 2).



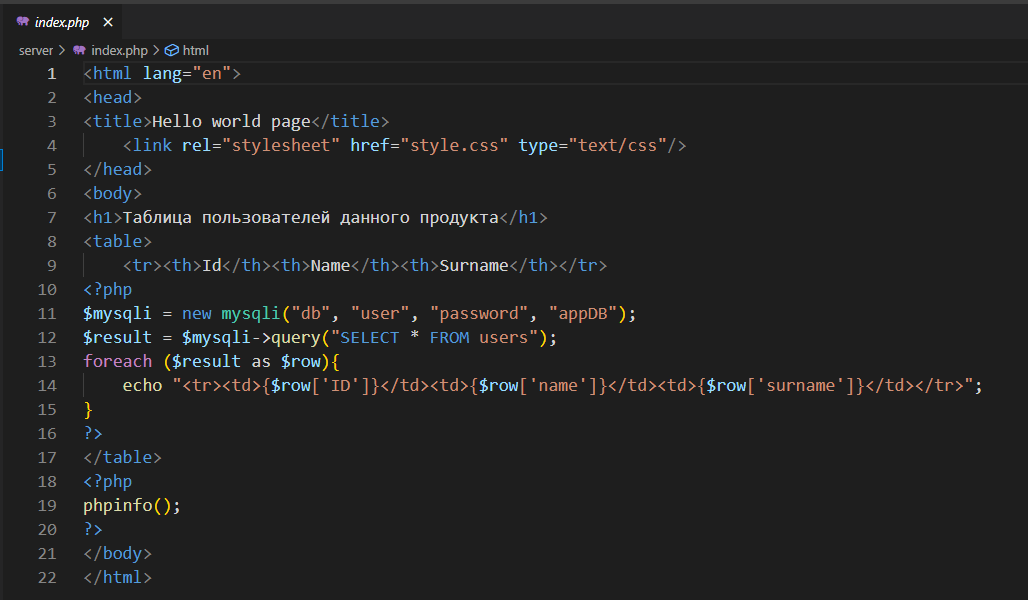
1. Файл init.sql

Для запуска веб-сервера будем использовать dockerfile (рисунок 3).



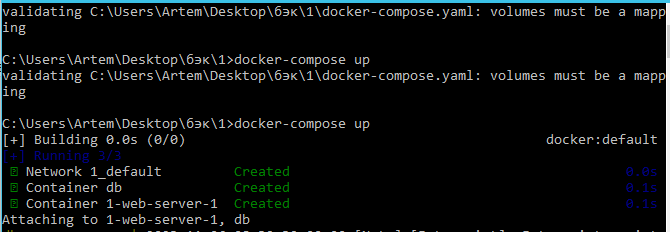
1. Файл dockefile

Веб сервер будет возвращать сгенерированную страницу, которая будет отображать таблицу из базы данных (рисунок 4).



1. Файл index.php

Запустим docker-compose с помощью команды docker-compose up (рисунок 5).



1. Файл docker-compose

Зайдём на веб-сервер и удостоверимся, что страница возвращается корректно (рисунок 6).



1. Скриншот возвращаемой страницы

## Вывод

В результате выполнения практической работы был выполнен корректный запуск приложенного php скрипта генерации страницы с характеристиками веб-сервера и его работа на созданном веб-сервере.

## Список источников

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/>

## Ответы на вопросы

1. Что такое Apache и для чего он нужен?

Apache - это открытый веб-сервер, который используется для обслуживания веб-сайтов. Основными функциями является: обработка HTTP-заросов, хостинг веб-сайтов, обработка динамического содержания, конфигурирование модулей веб-сервера, обеспечение безопасности.

1. Что такое оркестрация контейнеров и для чего это нужно?

Оркестрация контейнеров — это процесс управления, развертывания и управления контейнеризированными приложениями в среде, состоящей из множества контейнеров. Оркестрация обеспечивает автоматизацию и координацию работы контейнеров на кластере серверов, что упрощает масштабирование и обеспечивает надежную и высокодоступную работу приложений.

1. Что такое сервер?

Веб-сервер (web server) — это программное обеспечение или аппаратное устройство, которое принимает запросы от клиентских устройств (например, веб-браузеров) через протокол HTTP (или HTTPS) и отвечает на эти запросы, предоставляя запрашиваемую веб-страницу, файлы, данные или другие ресурсы через Интернет.

# Практическая работа №2

## Теоретическое введение

Server-side rendering (SSR) — это подход к созданию веб-страниц, при котором их начальный рендеринг происходит на сервере, а не на стороне клиента. При использовании SSR веб-страницы выполняются на серверной стороне, где генерируется HTML-код, который затем отправляется браузеру для отображения пользователю.

CRUD — это акроним, который означает Create, Read, Update и Delete - четыре основных операции над данными в системах управления базами данных. Эти операции широко используются в веб-разработке для создания, чтения, обновления и удаления информации в базах данных.

* Create — это операция создания новых записей в базе данных. Например, создание нового пользователя в системе или добавление новой статьи на сайт.
* Read — это операция чтения данных из базы данных. Например, получение списка всех пользователей в системе или всех товаров на сайте.
* Update — это операция обновления данных в базе данных. Например, изменение имени пользователя или обновление цены товара.
* Delete — это операция удаления данных из базы данных. Например, удаление пользователя из системы или удаление статьи с сайта.

## Задание

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
* Базу данных Mysql

Docker-compose:

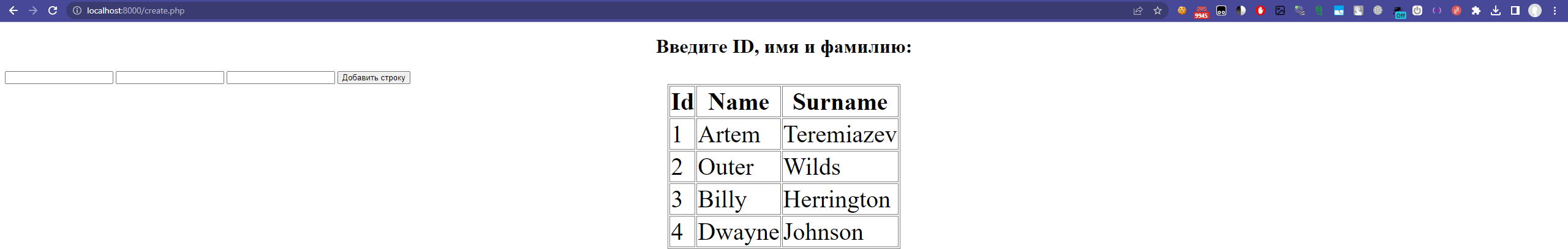
* Docker-контейнеры, созданные ранее
* Примонтированные тома
* Настройки портов
* Связь между контейнерами
* Переменные внешнего окружения
* Настройки базы данных

1. Создать sql скрипт инициализации, где студенту необходимо реализовать 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.

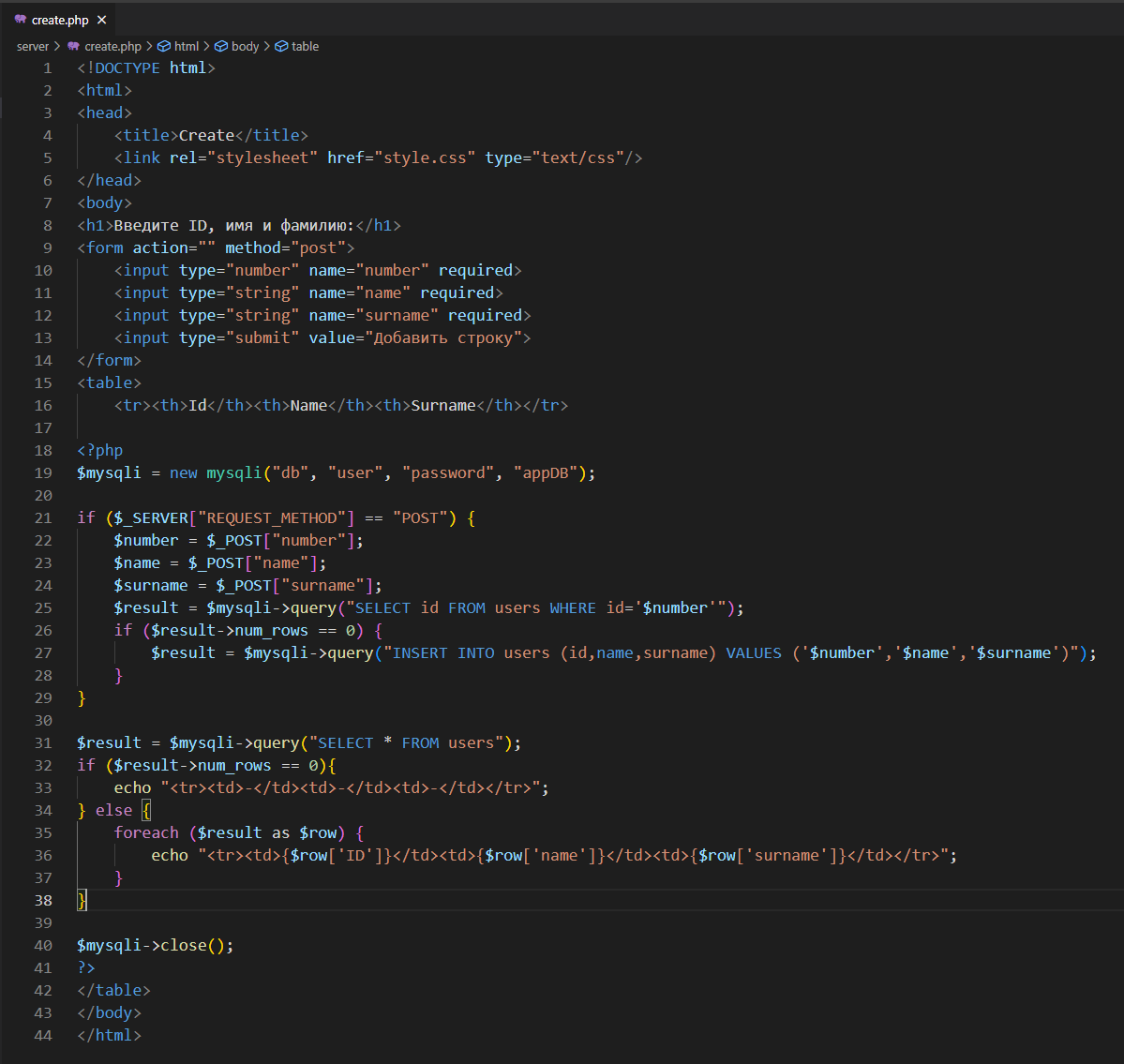
## Ход работы

Создадим страницу create.php. На ней даётся возможность добавить строку в базу данных (рисунок 1)



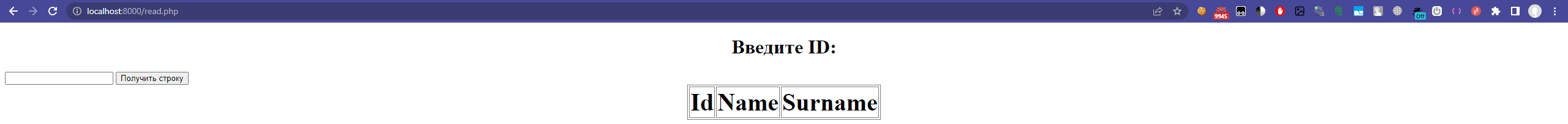
1. Скриншот страницы create.php

Php код этой страницы на сервере проверяет наличие строки с указанным id в базе данных и, если такой строки нет, то добавляет её (рисунок 2).



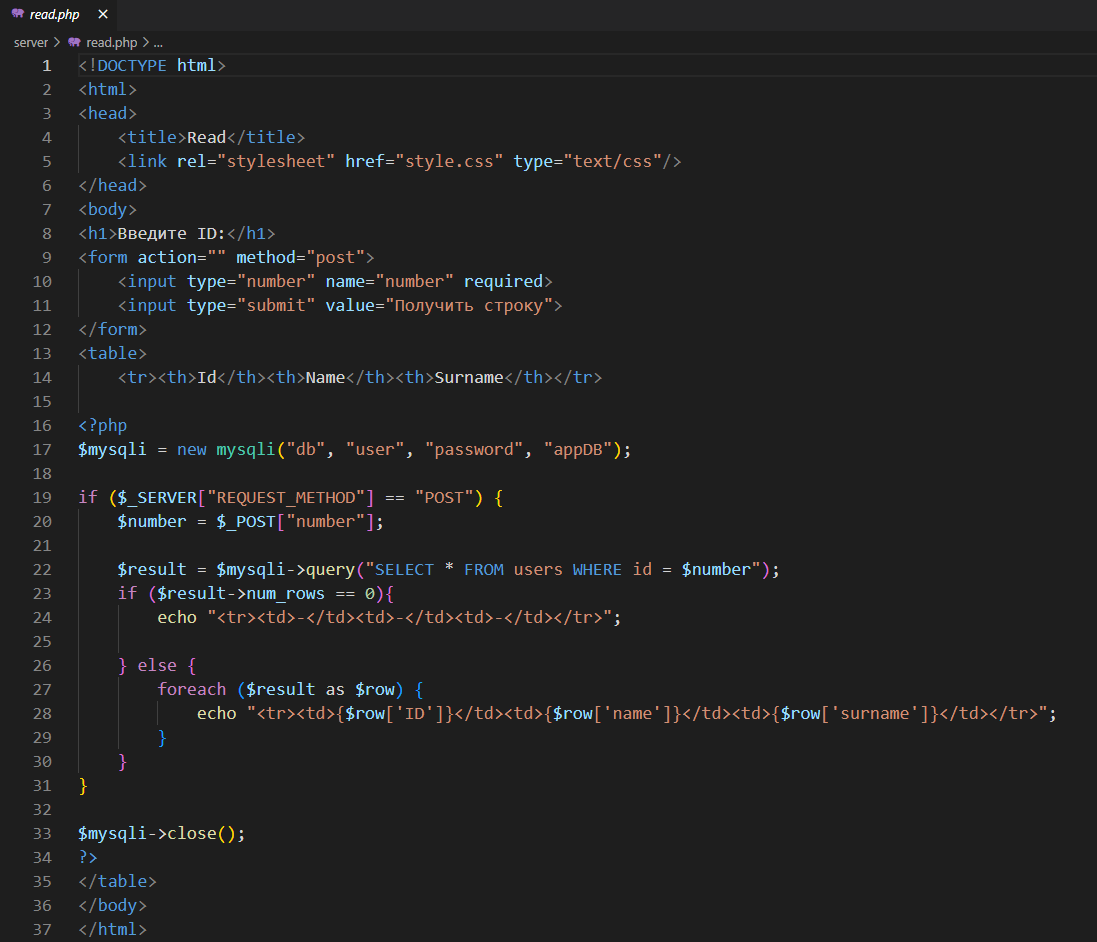
1. Код страницы create.php

Создадим страницу read.php. На ней даётся возможность получить строку из базы данных по id (рисунок 3)



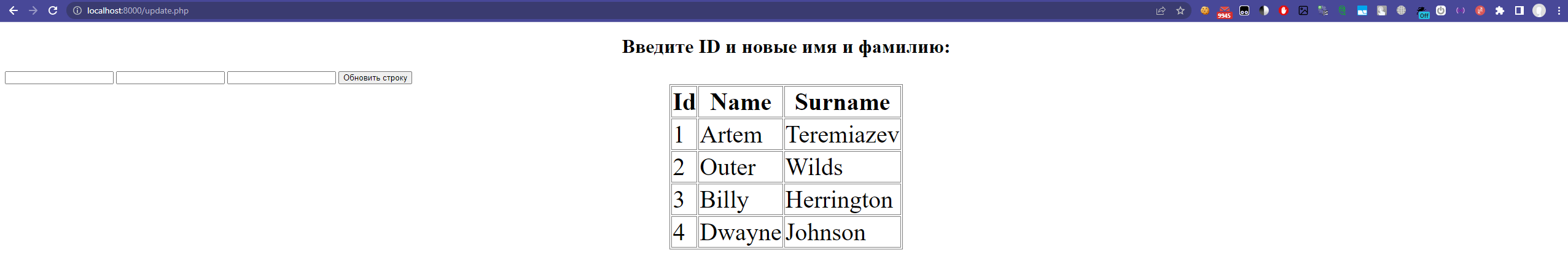
1. Скриншот страницы read.php

Php код этой страницы на сервере проверяет наличие строки с указанным id в базе данных и, если такая строка есть, то возвращает её (рисунок 4).



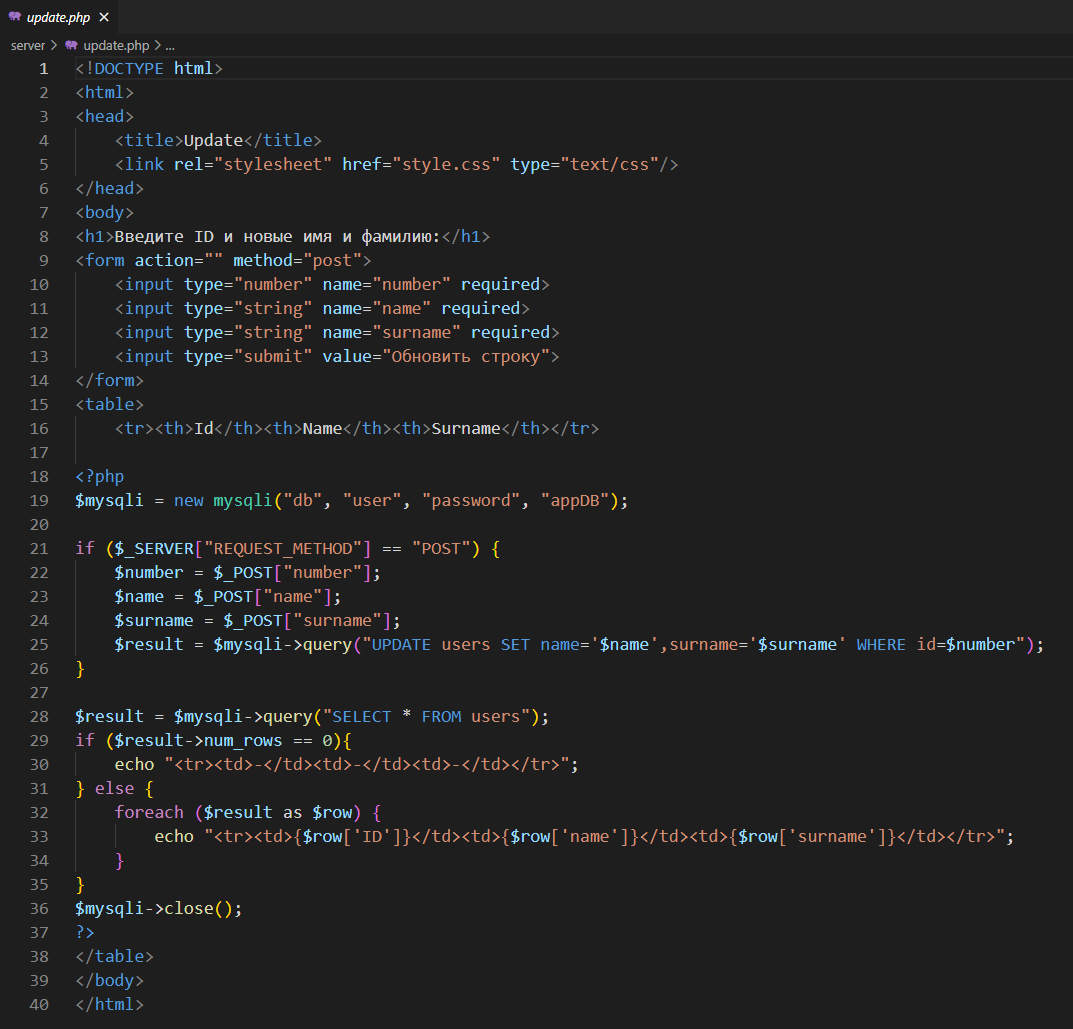
1. Код страницы read.php

Создадим страницу update.php. На ней даётся возможность получить строку из базы данных по id (рисунок 5)

x

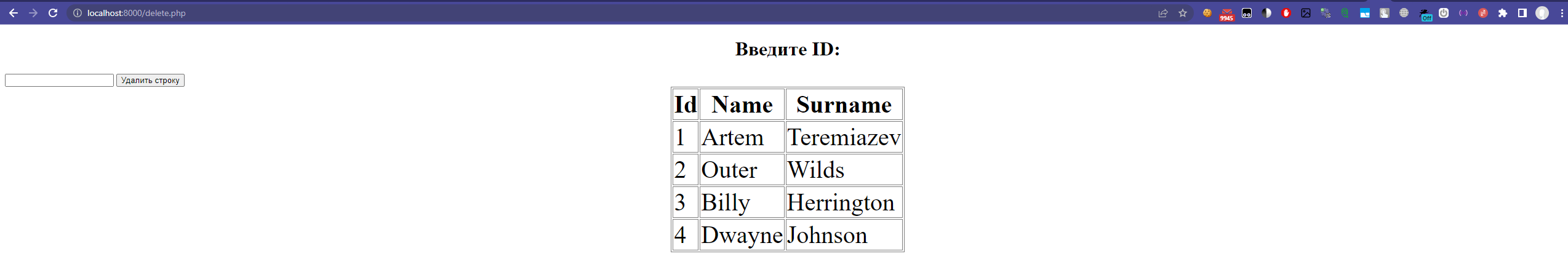
1. Скриншот страницы update.php

Php код этой страницы на сервере обновляет строку с указанным id (рисунок 6).



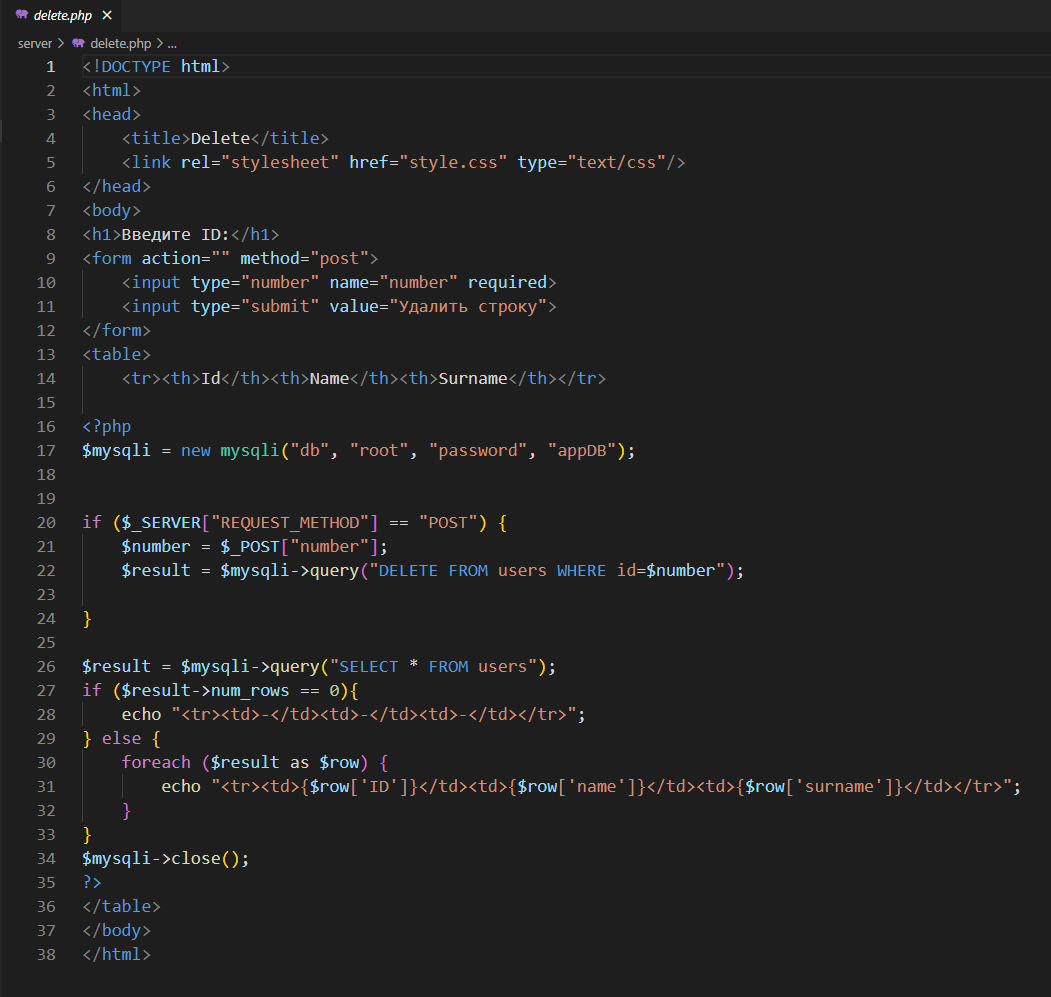
1. Код страницы update.php

Создадим страницу delete.php. На ней даётся возможность получить строку из базы данных по id (рисунок 7)



1. Скриншот страницы delete.php

Php код этой страницы на сервере ищет в базе данных строку с указанным id и удаляет её (рисунок 8).



1. Код страницы delete.php

## Вывод

В результате выполнения практической работы были созданы страницы с четырьмя основными операциями над данными в системах управления базами данных (CRUD).

## Список источников

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/

## Ответы на вопросы

1. В чём преимущество языка PHP для backend'а?

PHP идеально подходит для SSR. Когда в других язык необходимо создавать шаблоны и подвязывать всё с помощью библиотек, то в PHP это изначально зашито. Вам достаточно в html страничке открыть тег <?php> и писать необходимый вам код на php. И при запросе на сервер за страницей будет выполняться PHP код запрошенной страницы.

1. Как достигается тонкий клиент при использовании PHP?

Тонкий клиент достигается путем создания веб-приложений, в которых большая часть бизнес-логики и обработки данных выполняется на стороне сервера, а клиентский браузер служит для отображения пользовательского интерфейса и отправки запросов на сервер для обработки.

1. Что такое SSR?

Server-side rendering (SSR) — это подход к созданию веб-страниц, при котором их начальный рендеринг происходит на сервере, а не на стороне клиента. При использовании SSR веб-страницы выполняются на серверной стороне, где генерируется HTML-код, который затем отправляется браузеру для отображения пользователю.

# Практическая работа №3

## Теоретическое введение

REST API (Representational State Transfer Application Programming Interface) - это архитектурный стиль, используемый в веб-разработке, который предоставляет доступ к приложению через интернет-протоколы, такие как HTTP и HTTPS. Учитывая, что web-приложения обычно состоят из клиентской и серверной частей, REST API находится в backend серверной части. REST API определяет набор ограничений и ограничения на взаимодействие между клиентом и сервером. Клиенты отправляют запросы с помощью определенных HTTP-методов (POST, GET, PUT, PATCH, DELETE), чтобы получить, обновить, создать или удалить определенную информацию или ресурс в приложении.

Backend обрабатывает запросы и отправляет обратно соответствующие ответы в соответствии с принятыми стандартами HTTP. REST API обычно используются вместе с форматом данных JSON или XML для передачи информации между клиентом и сервером. Они позволяют разработчикам легко интегрировать и расширять приложения, создавать новые функции и предоставлять доступ к данным другим приложениям и сервисам.

REST API также придерживается принципов, называемых "ограничениями REST", такими как унифицированный интерфейс, отсутствие состояния, кэширование, многослойная архитектура и код по запросу. Эти принципы обеспечивают удобство использования и универсальность приложения, простоту его реализации и масштабируемость. REST API является неотъемлемой частью backend разработки и используется для взаимодействия между различными приложениями, а также для реализации различных сервисов и функций в приложении.

HTTP-запросы являются основой взаимодействия между клиентом и сервером в интернете. Мы будем использовать 4 типа запросов в этой практике: GET, POST, PUT и DELETE, которые используются для получения, отправки, обновления и удаления данных.

GET-запросы

Метод запроса GET используется для получения данных от сервера. С помощью GET-запроса клиент может передавать параметры в URL-адресе, и сервер будет использовать эти параметры для отображения информации на странице. GET-запросы не могут изменять данные на сервере, они только могут получать данные.

POST-запросы

Метод запроса POST используется для отправки данных на сервер. POST-запросы часто используются при отправке форм на сервер. В отличие от GET-запросов, POST-запросы могут изменять данные на сервере.

PUT-запросы

Метод запроса PUT используется для обновления данных на сервере. PUT-запрос должен содержать обновленные данные, которые нужно заменить в базе данных на сервере. PUT-запросы часто используются при обновлении отдельных элементов на странице.

DELETE-запросы

Метод запроса DELETE используется для удаления данных на сервере. DELETE-запрос должен содержать идентификатор данных, которые нужно удалить. Он удаляет целые записи из базы данных на сервере.

Все запросы состоят из двух частей: заголовка и тела запроса. Заголовок содержит метаданные, такие как тип запроса, тип документа и другую дополнительную информацию. Тело запроса содержит данные, которые отправляются на сервер.

Для того чтобы реализовать нормальный rest api, потребуется описать базовые сущности.

В backend разработке сущность (Entity) — 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.

Примером сущности в backend может быть таблица "Пользователь" в базе данных. Сущность "Пользователь" содержит поля, такие как "id", "имя", "адрес электронной почты" и "пароль". Кроме того, эта сущность может иметь методы для добавления, удаления, обновления и чтения пользователей из базы данных.

Сущности в backend должны быть полностью абстрагированы от любой логики приложения и слоя представления. Это означает, что изменения в базе данных или логике приложения не должны приводить к изменению сущности. Это позволяет уменьшить связанность между различными частями приложения и повышает его модульность.

## Задание

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.

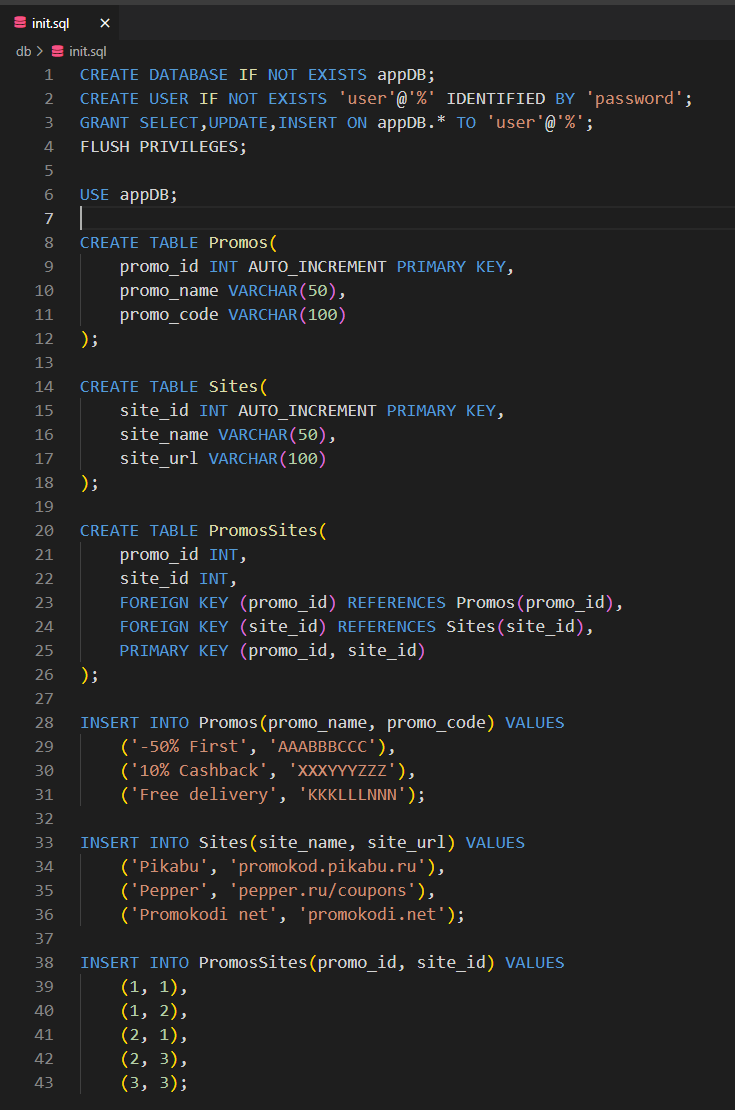
2. Требуется реализовать REST API для двух сущностей, где на запросы ответы должны быть в формате JSON.

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

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

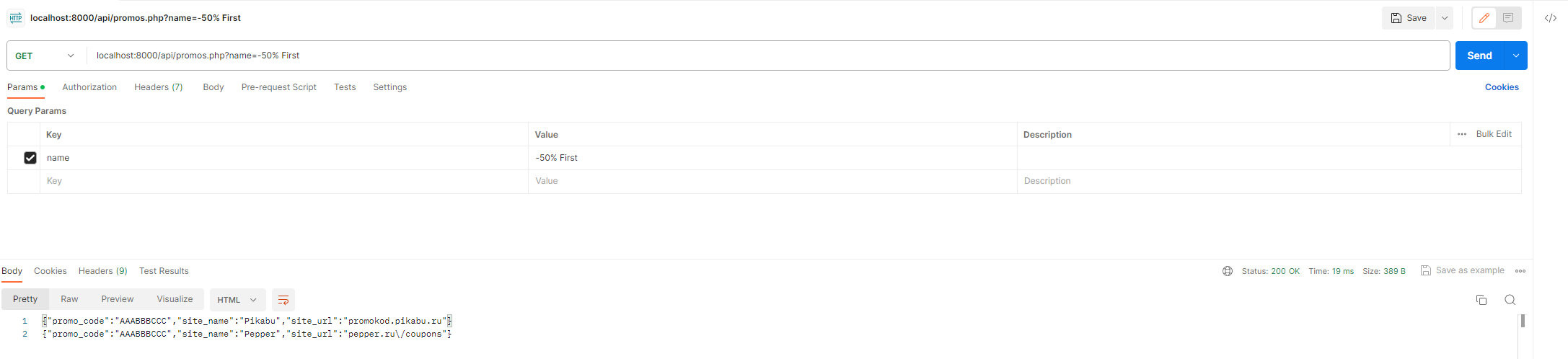
## Ход работы

Обновим скрипт для инициализации базы данных (рисунок 9).



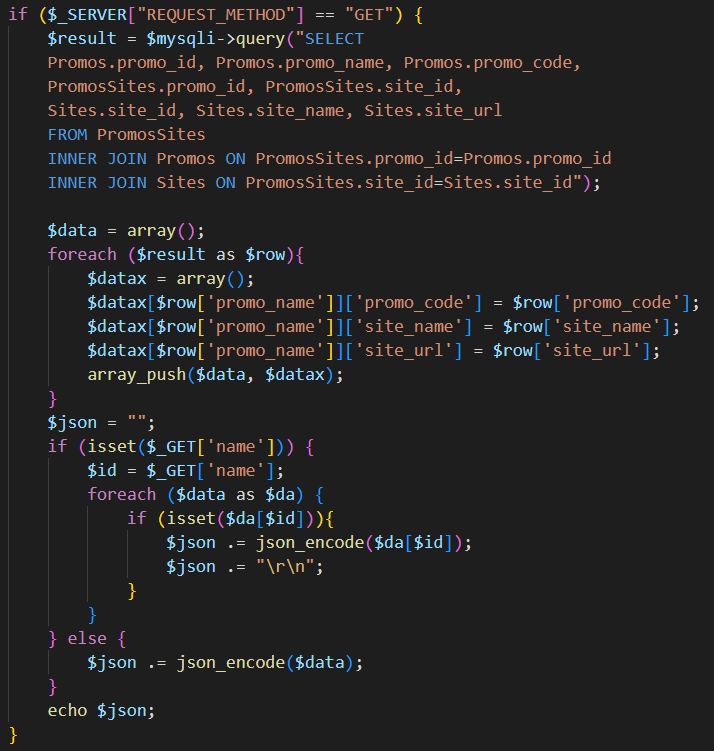
1. Новый файл init.sql

В базе данных теперь есть таблицы Promos и Sites, которые связаны через таблицу PromosSites. С данными из этих таблиц можно взаимодействовать через API. Так через GET запрос можно получить информацию об игре (рисунок 10).



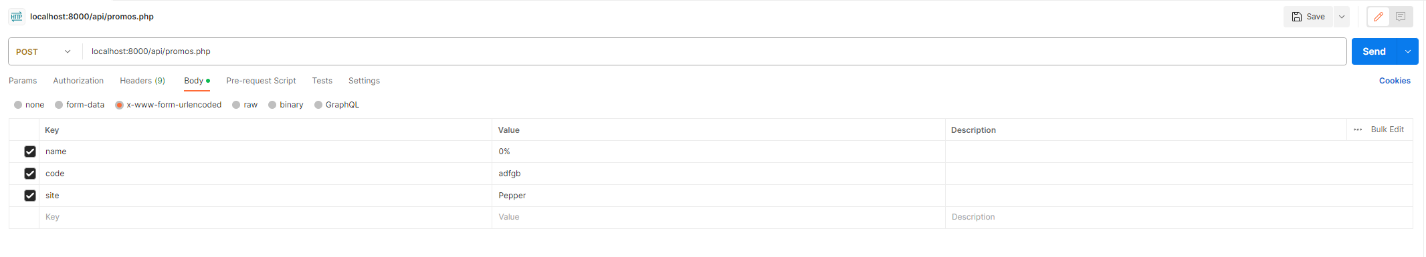
1. GET запрос

Код, отвечающий за GET запрос проводит поиск информации по базе данных или возвращает информацию обо всех промокодах, если в запросе не было указано название (рисунок 11).



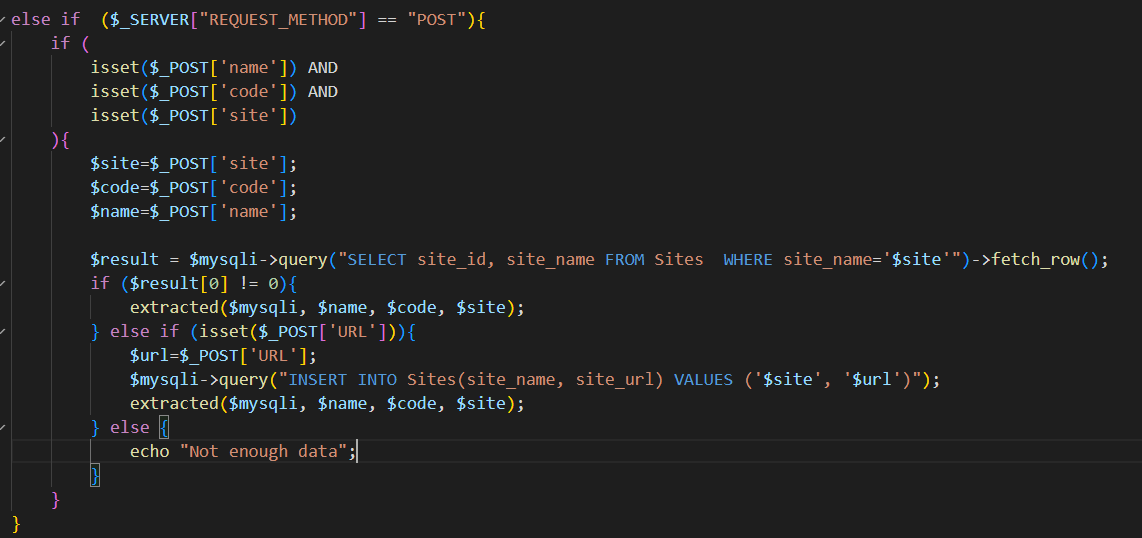
1. Код GET запроса

POST запрос позволяет добавить промокод в базу данных (рисунок 12).



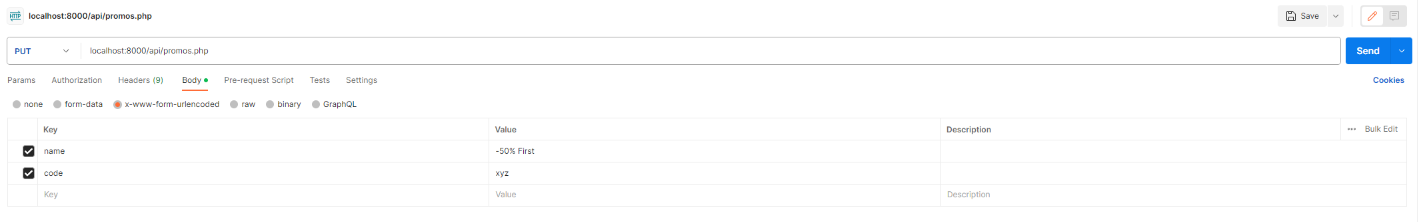
1. POST запрос

Код, отвечающий за POST запрос проводит поиск информации по базе данных на наличие уже такого промокода и проверяет, что в запросе достаточно информации для формирования новой записи (рисунок 13).



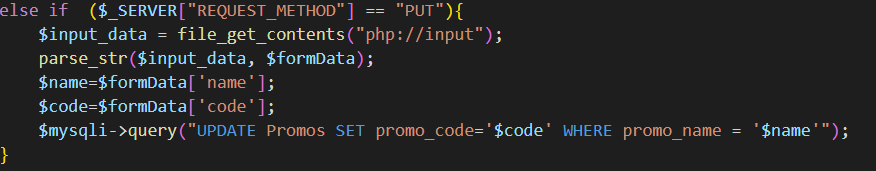
1. Код POST запроса

PUT запрос позволяет обновить запись в базе данных по названию (рисунок 14).



1. PUT запрос

Код, отвечающий за PUT запрос обновляет запись с помощью одной SQL команды (рисунок 15).



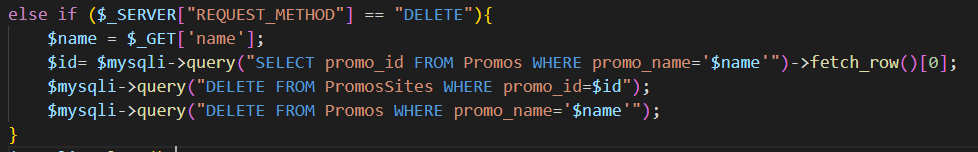
1. Код PUT запроса

DEL запрос удаляет запись в базе данных по названию (рисунок 16).



1. DEL запрос

Код, отвечающий за DEL запрос удаляет упоминания игры из всех таблиц базы данных (рисунок 17).



1. Код DEL запроса

## Вывод

В результате выполнения практической работы было настроено взаимодействие с базой данных через серверную часть с помощью API. Получены навыки работы с основными HTTP запросами.

## Список источников

1. Пример создания rest api на php: https://cms.by/blog/php/primer-prostogosozdaniya-rest-api-na-php/

2. Примеры rest api приложений: <https://github.com/topics/php-rest-api>

3. Гайд как пользоваться postman: <https://testengineer.ru/gajd-po-testirovaniyu-vpostman/>

4. Статья на хабре про принципы построения REST JSON API: <https://habr.com/ru/articles/447322/>

5. Шпаргалка по безопасности rest api: <https://habr.com/ru/companies/acribia/articles/453384/>

6. Гайд по использованию curl: <https://megahub.me/cheat-sheets/curl>

7. Лучшие практики rest api: https://habr.com/ru/companies/piter/articles/511382/

## Ответы на вопросы

1. Что такое сущность?

В backend разработке сущность (Entity) — это объект, который представляет собой некоторый вид данных, которые могут быть сохранены и 3 извлечены из базы данных. Сущность в backend содержит поля и методы, которые позволяют работать с данными в базе данных.

1. Какие методы существуют в протоколе http?

Клиенты отправляют запросы с помощью определенных HTTP-методов (POST, GET, PUT, PATCH, DELETE), чтобы получить, обновить, создать или удалить определенную информацию или ресурс в приложении.

GET-запросы. Метод запроса GET используется для получения данных от сервера. С помощью GET-запроса клиент может передавать параметры в URL-адресе, и сервер будет использовать эти параметры для отображения информации на странице. GET-запросы не могут изменять данные на сервере, они только могут получать данные.

POST-запросы. Метод запроса POST используется для отправки данных на сервер. POST-запросы часто используются при отправке форм на сервер. В отличие от GET-запросов, POST-запросы могут изменять данные на сервере.

PUT-запросы. Метод запроса PUT используется для обновления данных на сервере. PUT-запрос должен содержать обновленные данные, которые нужно заменить в базе данных на сервере. PUT-запросы часто используются при обновлении отдельных элементов на странице.

DELETE-запросы. Метод запроса DELETE используется для удаления данных на сервере. DELETE-запрос должен содержать идентификатор данных, которые нужно удалить. Он удаляет целые записи из базы данных на сервере.

1. Что такое контроллер?

В контексте backend-разработки, контроллер (или контроллеры) - это часть программного кода, которая обрабатывает входящие запросы от клиентской стороны (например, от веб-браузера) и управляет выполнением соответствующей логики в приложении.

# Практическая работа №4

## Теоретическое введение

Nginx — это сервер веб-сервер и прокси-сервер, который обеспечивает высокую производительность и эффективную обработку запросов. Nginx изначально был создан для обеспечения высокой производительности в условиях высоких нагрузок на сервер, где он используется для обслуживания динамических веб-сайтов и приложений.

Одним из ключевых преимуществ Nginx является его способность обрабатывать тысячи одновременных соединений, что делает его идеальным выбором для высоконагруженных веб-сайтов и приложений. В качестве прокси-сервера Nginx используется для балансировки нагрузки и распределения запросов на различные серверы, что помогает увеличить скорость и устойчивость приложения.

Важным преимуществом Nginx является его способность к эффективной обработке статических файлов, которые обслуживаются непосредственно из кэш-памяти, что значительно ускоряет процесс обработки запросов. Кроме того, Nginx является открытым программным обеспечением, которое позволяет пользователям использовать его бесплатно, уставлять свои доработки и расширения.

В современной веб-разработке 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, таких как Apache, Tomcat и другие.

В общем, Nginx является мощным инструментом для обеспечения высокой производительности и эффективной обработки запросов, что делает его идеальным выбором для высоконагруженных веб-сайтов и приложений.

## Задание

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 контейнер NGINX должен включать:

* ОС Linux
* NGINX

Docker-compose:

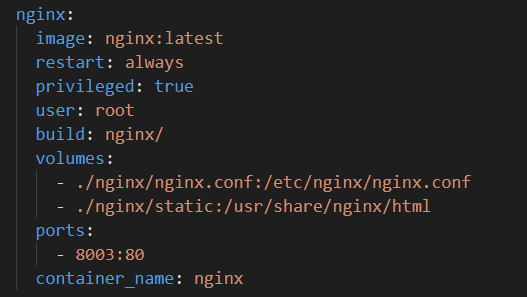
* Docker-контейнеры, созданные ранее
* Примонтированные тома
* Настройки портов
* Связь между контейнерами
* Переменные внешнего окружения
* Настройки базы данных

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

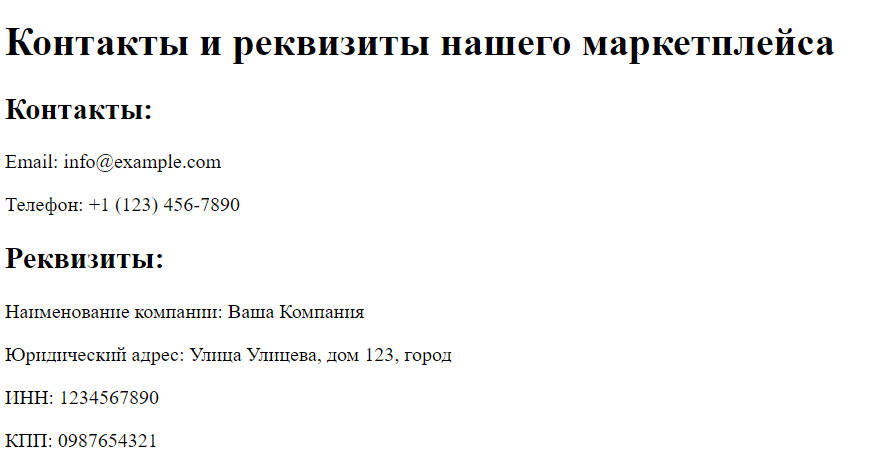
## Ход работы

Изменим файл docker-compose, добавив в него nginx (рисунок 18).



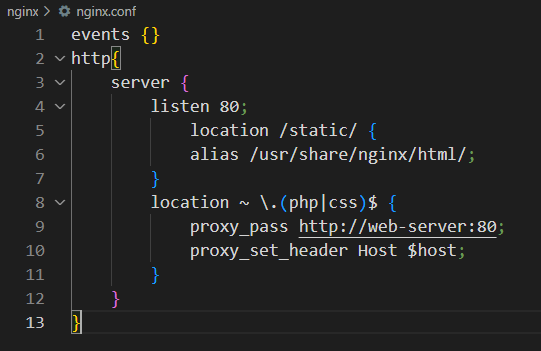
1. Изменения в docker-compose

Создадим статичные html страницы (рисунок 19).



1. Пример статичной страницы

В файле nginx.conf настроим сервер так, чтобы статические контейнеры могли выдаваться по запросу. Там же подключим обратный прокси-сервер, чтобы nginx выдавал php страницы из предыдущих практик (рисунок 20).



1. Конфигурационный файл nginx

## Вывод

В результате выполнения практической работы был настроен веб-сервер и прокси-сервер Nginx. Также был подключен обратный прокси для выдачи страниц из предыдущих практик.

## Список источников

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 NGINX в качестве обратного прокси сервера: <https://help.reg.ru/support/servery-vps/oblachnyye-servery/ustanovka-programmnogoobespecheniya/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/

## Ответы на вопросы

1. Что такое обратный прокси сервер?

Обратный прокси-сервер — это сервер, который действует от имени клиента и осуществляет обратное направление запросов от клиента к серверам. В отличие от прямого прокси-сервера, который обеспечивает доступ клиента к внешним ресурсам, обратный прокси-сервер используется для обработки входящего трафика и доставки его на конечный сервер.

1. Как можно использовать NGINX?

NGINX — это мощный сервер, который может быть использован для различных целей. Его можно использовать как веб-сервер, прокси-сервер и для обслуживания динамических веб-сайтов и приложений.

1. 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.

1. Как в конфигурации NGINX добавить 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.