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

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №1-4**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы** ИКБО-10-21 Никонова Е. А.

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

(подпись студента)

**Руководитель практической работы** асистентБлагирев М. М.

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

(подпись руководителя)

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

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

Москва 2023

Содержание

[Практическая работа № 1 3](#__RefHeading___Toc835_1872065658)

[Практическая работа № 2 6](#__RefHeading___Toc835_18720656581)

[Практическая работа № 3 10](#__RefHeading___Toc835_18720656582)

[Практическая работа № 4 14](#__RefHeading___Toc835_187206565821)

[Список использованной литературы: 17](#__RefHeading___Toc926_4009055764)

# 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.

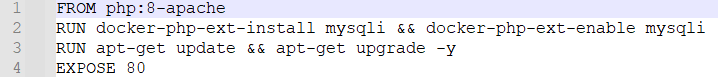
Backend является одной из ключевых компонентов веб-приложения. Он обеспечивает логику, необходимую для обработки запросов, а также управление данными и их хранение. Backend веб-приложения работает в связке с frontend (клиентской частью), которая отображает пользовательский интерфейс и обрабатывает пользовательский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

**Цель работы:** создать конфигурацию docker-compose для веб-сервера.

**Ход работы:**

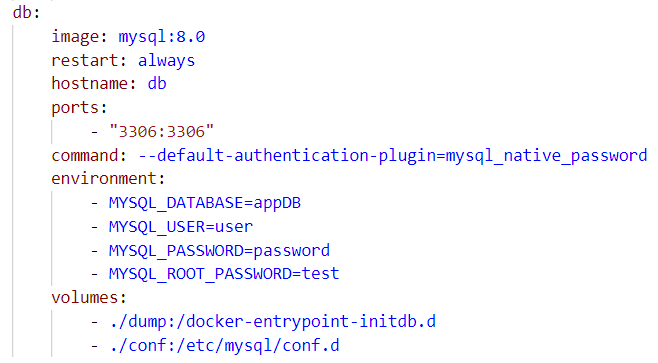
1. Напишем Dockerfile для конфигурации нашего контейнера (рис. 1).

Рисунок 1 — Dockerfile

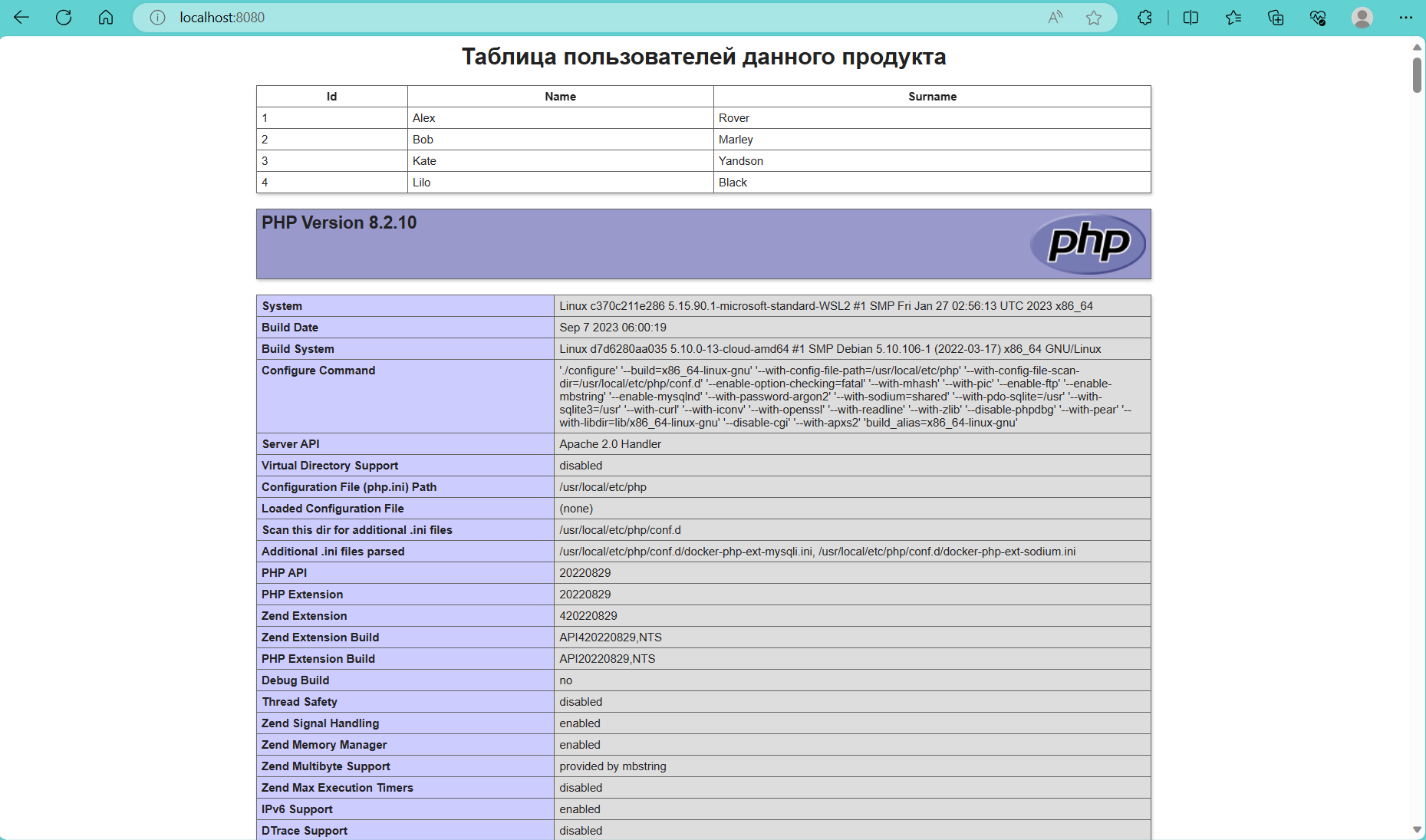
2. Создадим сервис для веб-сервера (рис. 2).

Рисунок 2 — Docker-compose.yaml

3. Создадим сервис для базы данных (рис. 3).

Рисунок 3 — Docker-compose.yaml

Убедимся в работоспособности созданного решения (рис. 4).

Рисунок 4 — Итог работы веб-сервера

**Вывод:** была создана конфигурация docker-compose для веб-сервера. Конфигурация включает 2 сервиса: сервер и база данных. Итогом стал успешный запуск php-скрипта, который генерирует базу данных и выводит информацию о сервере.

**Ответы на вопросы:**

1. Что такое Apache и для чего он нужен?

Apache — это самый популярный веб-сервер, который запускается на миллионах веб-сайтов во всем мире. Apache является свободной и открытой программой, которая доступна для распространения и использования бесплатно.

Основной целью Apache является обслуживание веб-страниц пользователям, обрабатывая запросы от веб-браузеров и отображая содержимое веб-страниц в ответ на эти запросы.

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

Оркестрация контейнеров — это управление отдельными веб-серверами, работающими в контейнерах, с помощью виртуальных разделов оборудования ЦОД. Программы-оркестраторы управляют жизненными циклами контейнеров микросервисных приложений. Но не только. Так как в разработке приложений все взаимосвязано, то системы оркестрации помогают еще автоматизировать различные производственные процессы

3. Что такое сервер?

Сервер — устройство, которое хранит данные и даёт доступ к ним большому числу пользователей. Сервер включает мощный компьютер и программное обеспечение.

# Практическая работа № 2

**Теоретическое введение**

При использовании тонкого клиента веб-сервер выполняет роль контролера, отвечая за координацию работы сервера и приложения на клиентских устройствах. Это позволяет минимизировать размер и сложность кода JavaScript, а также улучшить производительность приложения и общее качество работы.

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.

**Цель работы:** подключить БД к веб-серверу, реализовать операции CRUD на php.

**Ход работы:**

1. Cоздадим конфигурацию docker-compose для веб-сервера.

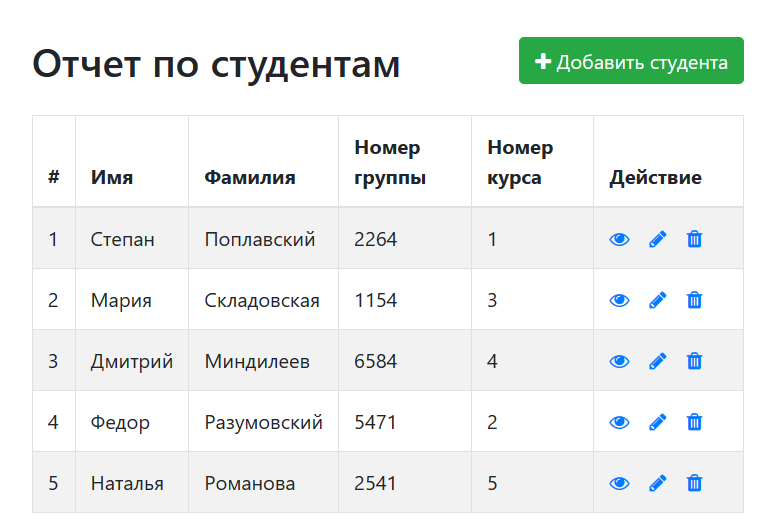
Docker-compose будет включать:

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

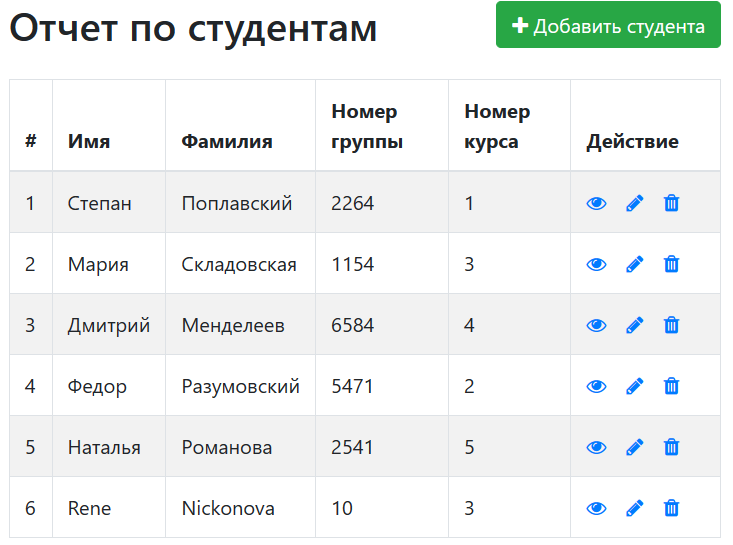
Соответствующая конфигурация docker-compose представлена на рисунке 5.

Рисунок 5 — Файл docker-compose

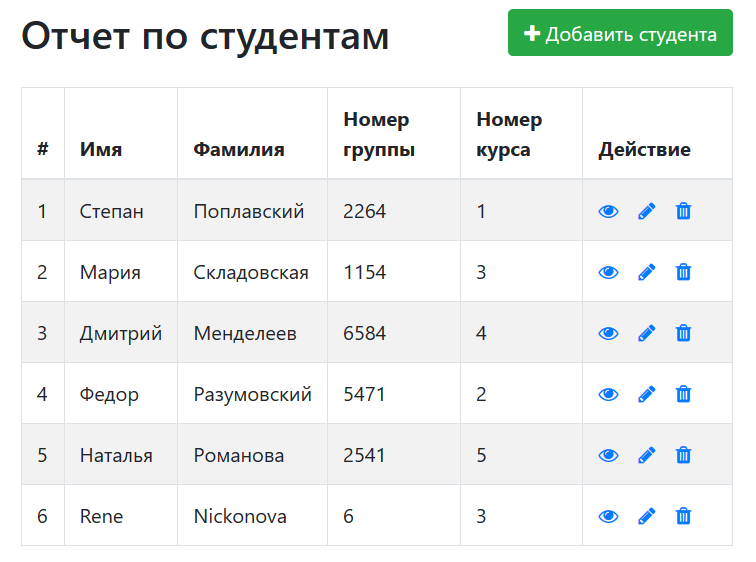
2. Реализуем операции CRUD на языке php и проверим их правильность. Для этого запустим созданный веб-сервер (рис. 6).

Рисунок 6 — Результат запуска веб-сервера

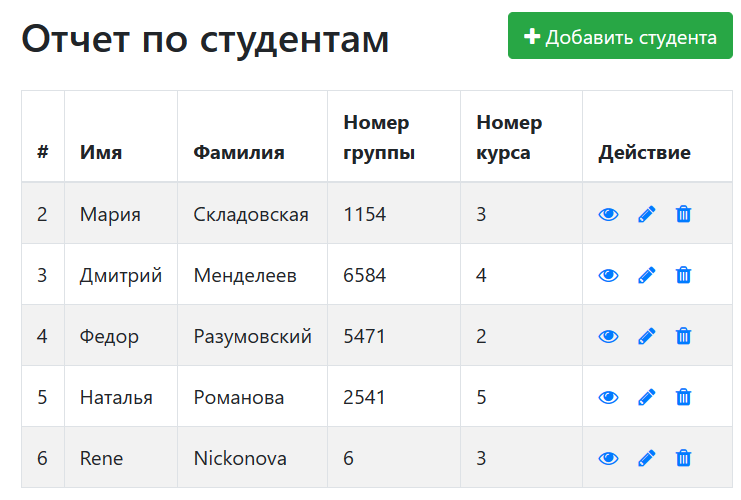
Проверим правильность выполнения операции Create: создадим нового студента (рис. 7).

Рисунок 7 — Результат добавления нового студента

Проверим правильность операции Update: обновим созданному студенту номер группы (рис. 8).

Рисунок 8 — Результат обновления данных

Проверим правильность выполнения операции Delete: удалим одну из записей о студентах (рис. 9).

Рисунок 9 — Результат удаления студента

**Вывод:** была создана конфигурация docker-compose для веб-сервера, подключена БД к веб-серверу, реализованы операции CRUD на php.

**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.

2. Как достигается тонкий клиент при использовании PHP?

При запросе на сервер за страницей выполняется PHP код запрошенной страницы.

3. 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.

# Практическая работа № 3

**Теоретическое введение**

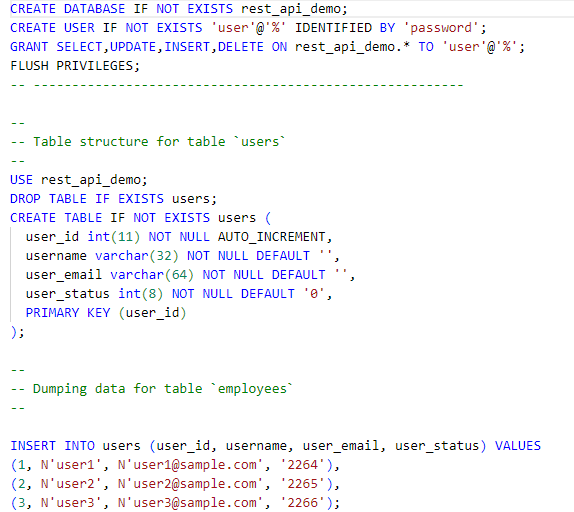
REST API (Representational State Transfer Application Programming Interface) - это архитектурный стиль, используемый в веб-разработке, который предоставляет доступ к приложению через интернет-протоколы, такие как HTTP и HTTPS.

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.

**Цель работы:** реализовать REST API для двух сущностей, где на запросы ответы должны быть в формате JSON. Провести ручное тестирование с помощью POSTMAN.

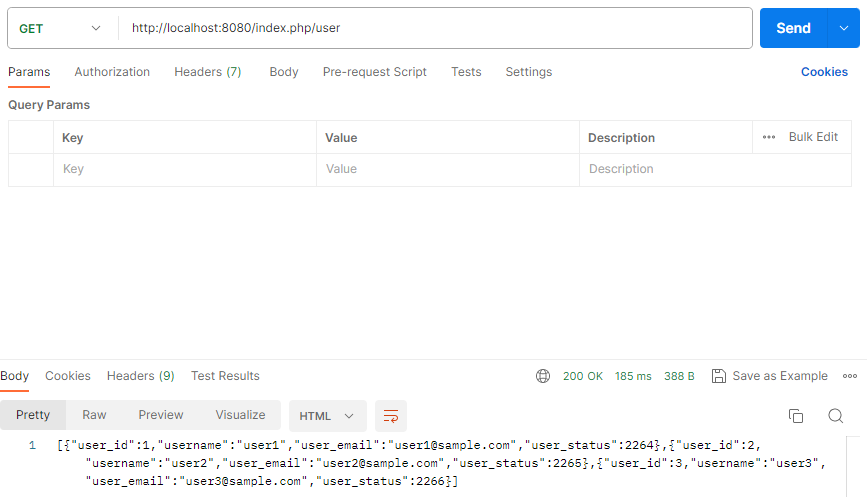
**Ход работы:**

1. Создадим sql-скрипт инициализации (рис. 10).

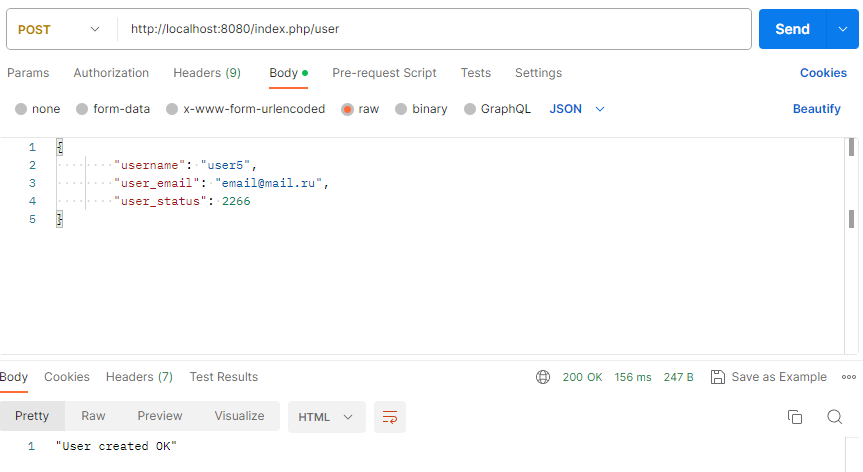
Рисунок 10 – Sql-скрипт инициализации

Реализуем REST API для сущности, где на запросы ответы должны быть в формате JSON и проведем ручное тестирование в POSTMAN.

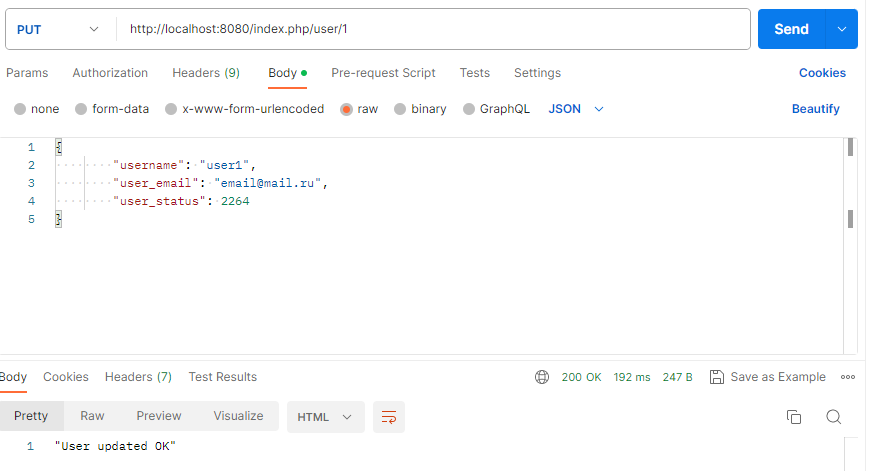
Проверим работу GET-запроса (рис. 11).

Рисунок 11 — GET-запрос

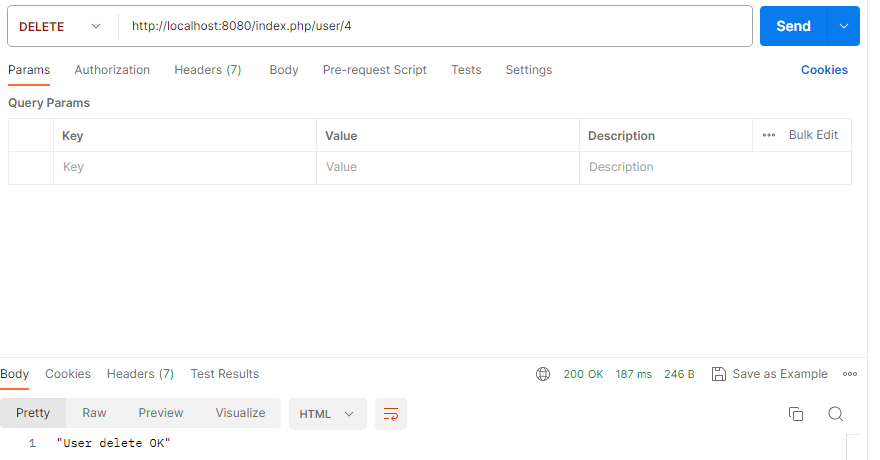
Проверим работу POST-запроса (рис. 12).

Рисунок 12 — POST-запрос

Проверим работу PUT-запроса (рис. 13).

Рисунок 13 — PUT-запрос

Проверим работу DELETE-запроса (рис. 14).

Рисунок 14 — DELETE-запрос

**Вывод:** был реализован REST API для двух сущностей, где на запросы ответы должны быть в формате JSON, проведено ручное тестирование с помощью POSTMAN.

**Ответы на вопросы:**

1. Что такое сущность?

Сущность — это конфигурация хранимого и обрабатываемого объекта, обеспечивающая корректность и целостность данных, обязывающая систему проверять формат данных и вхождение в диапазон допустимых значений.

2. Какие методы существуют в протоколе http?

GET, POST, PUT, PATCH, DELETE.

3. Что такое контроллер?

Программный контроллер — это часть программы, отвечающая за управление и координацию работы программы.

# 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.

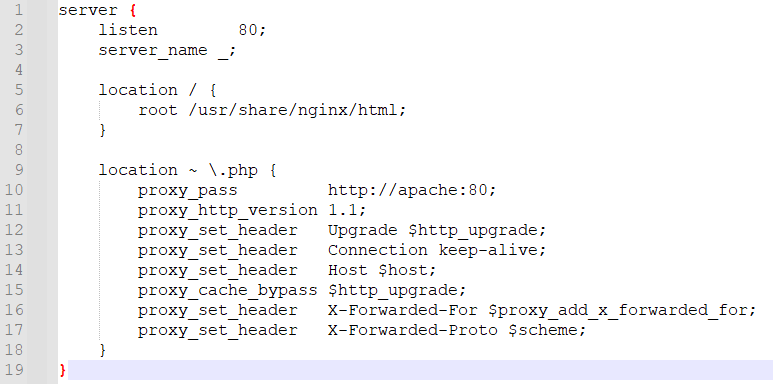
Важным преимуществом Nginx является его способность к эффективной обработке статических файлов, которые обслуживаются непосредственно из кэш-памяти, что значительно ускоряет процесс обработки запросов.

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.

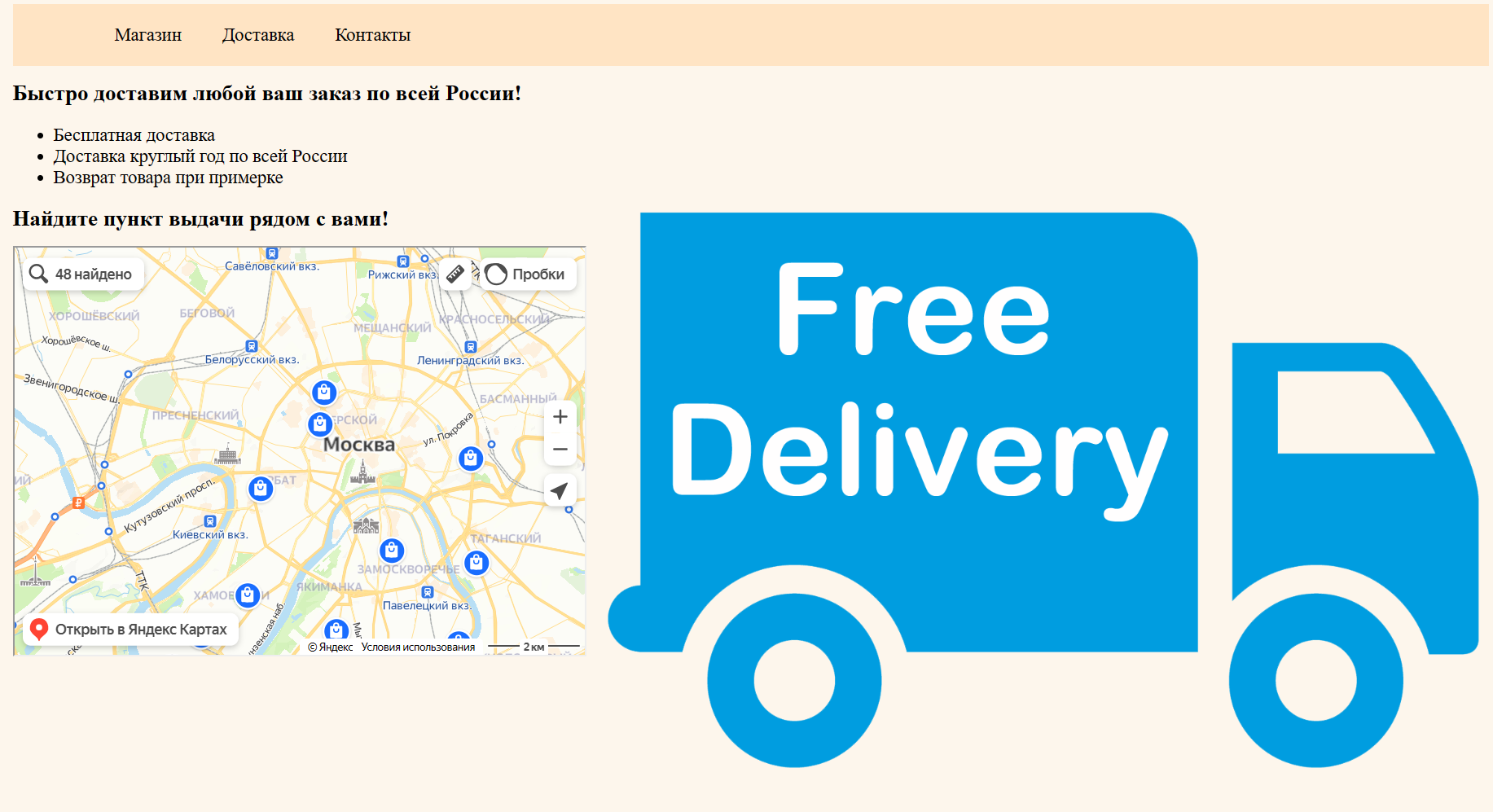
**Цель работы:** Создать 2 статичные html страницы для маркетплейса, где на одной странице будет информация по доставке, а на другой будут контакты и реквизиты вашего маркетплейса. Нужно сделать таким образом, чтобы статические страницы хранились в контейнере вместе с сервисом обратного прокси-сервера, а также настроить NGINX так, чтобы он мог их выдавать по запросу. Необходимо сделать так, чтобы запросы и ответы на динамические страницы проходили через NGINX.

**Ход работы:**

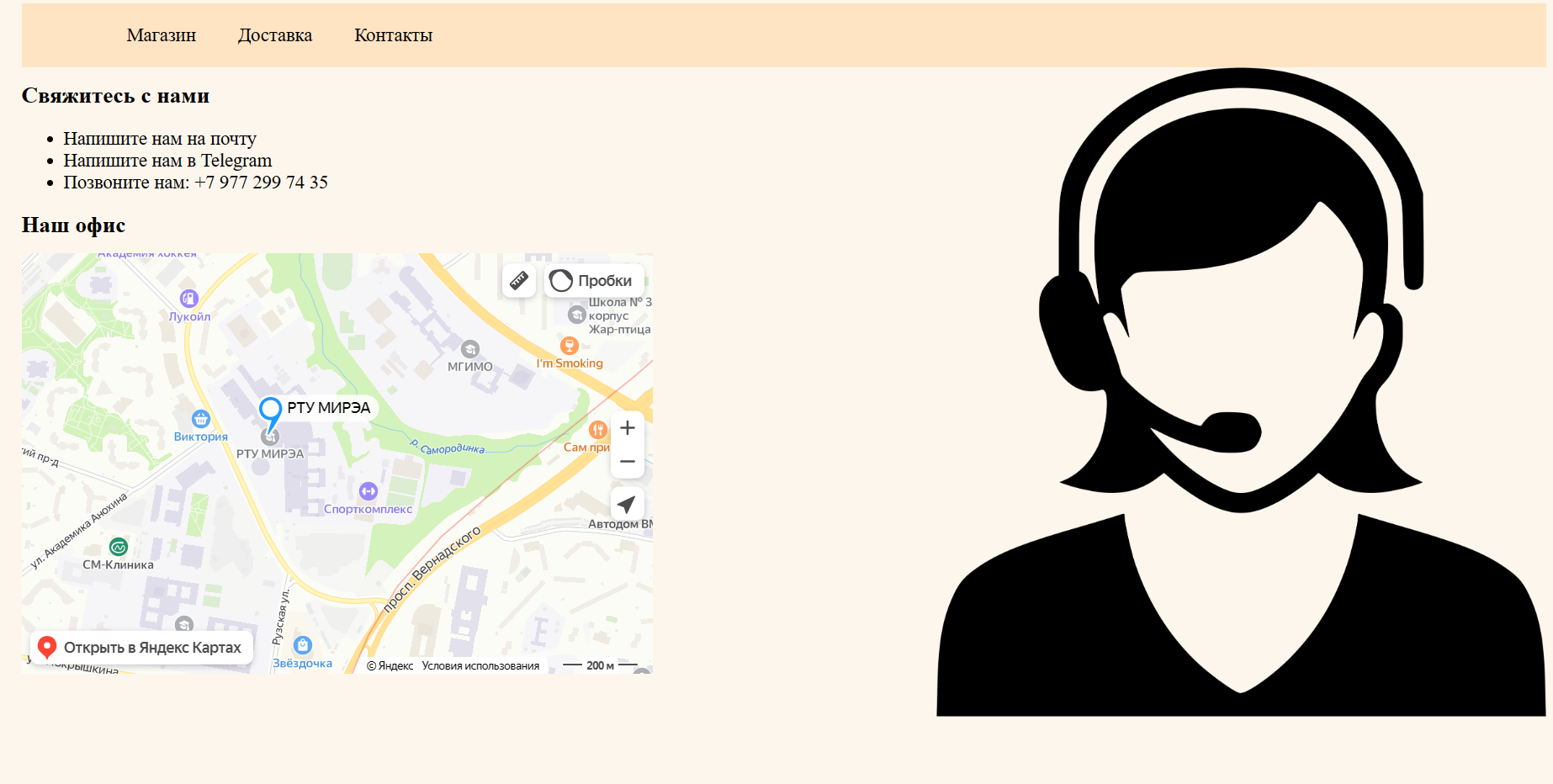
1. Напишем конфигурационный файл (рис. 15).

Рисунок 15 — Конфигурационный файл

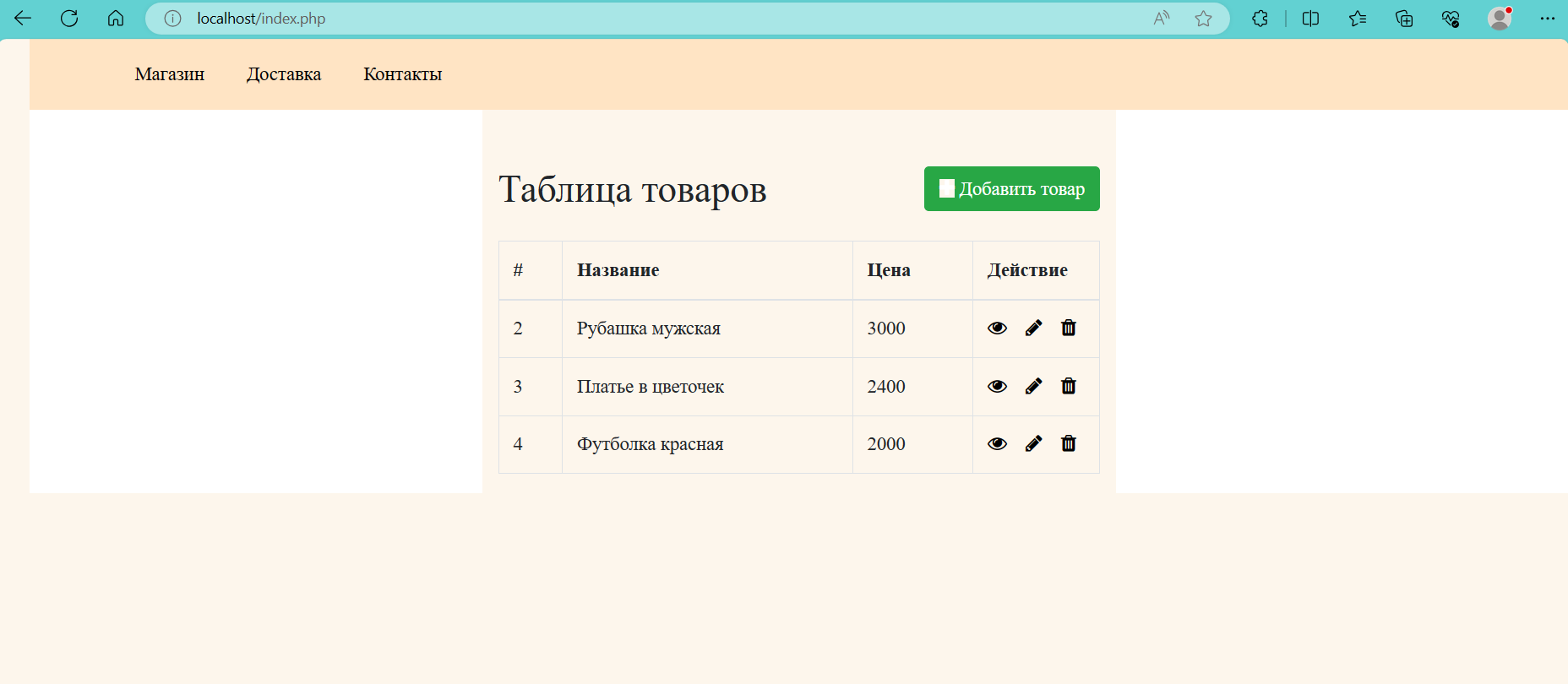
2. Проверим работу программы. Проверим работу страницы с информацией о доставке (рис. 16).

Рисунок 16 — Страница с информацией о доставке

Проверим работу страницы с информацией о контактных данных (рис. 17).

Рисунок 17 — Страница с информацией о контактных данных

Проверим работу динамической страницы с информацией о товарах (рис. 18).

Рисунок 18 — Страница с информацией о товарах

**Вывод:** были созданы статичные html страницы для маркетплейса, где на одной странице будет информация по доставке, а на другой будут контакты и реквизиты вашего маркетплейса. Статические страницы хранятся в контейнере вместе с сервисом обратного прокси-сервера, а также NGINX настроен так, чтобы он мог их выдавать по запросу. Запросы и ответы на динамические страницы проходили через NGINX.

**Ответы на вопросы:**

1. Что такое обратный прокси сервер?

Обратный прокси-сервер — тип прокси-сервера, который ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети.

2. Как можно использовать NGINX?

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.

3. Как работает кеширование в NGINX?

Важным преимуществом Nginx является его способность к эффективной обработке статических файлов, которые обслуживаются непосредственно из кэш-памяти, что значительно ускоряет процесс обработки запросов.

4. Как в конфигурации NGINX добавить несколько веб-серверов динамических страниц?

Можно добавить несколько блоков location с proxy\_pass.

**Ссылка на удаленный репозиторий проекта:** [**Elisaveta-N/rschir (github.com)**](https://github.com/Elisaveta-N/rschir)

# Список использованной литературы:

1. Документация // PHP URL: [https://www.php.net/manual/ru/index.php](https://vk.com/away.php?to=https%3A%2F%2Fwww.php.net%2Fmanual%2Fru%2Findex.php&cc_key=) (дата обращения: 16.10.2021). – Текст: электронный.

2. Дергачев, А. М. Проблемы эффективного использования сетевых сервисов / А. М. Дергачев // Научно-технический вестник СПбГУ ИТМО. –2011. – № 1 (71). С. 83-87.

3. Розенфельд, Л. Информационная архитектура в Интернете / Л. Розенфельд, П. Морвиль, 2 е издание. – Пер. с англ. – СПб: Символ Плюс, 2005 – 544 с.

4. Спинеллис, Д. Идеальная архитектура. Ведущие специалисты о красоте программных архитектур / Д. Спинеллис, Г. Гусиос. – Пер. с англ. –СПб.: Символ Плюс, 2010 – 528 с.

5. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер Пер. с англ. − М.: Издательский дом "Вильяме", 2006 − 544 с.

6. Информационные технологии / О.Л. Голицына, Н.В. Максимов, Т.Л. Партыка, И.И. Попов. − 2-е изд. − Москва: ФОРУМ − ИНФРА-М, 2008. – 395 с.

7. Таненбаум, Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. − СПб.: Питер, 2003. − 93 с.

8. Котеров, Д.В. PHP 7 / Д.В. Котеров, И. В. Симдянов. - СПб.: БХВПетербург, 2021. – 1088 с.