



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**

по дисциплине «Проектирование и разработка серверных частей интернет-ресурсов»

**Студент группы ИКБО-21-23**

**Зеленков Н.А.**

---

(подпись студента)

**Руководитель практической работы**

**Благирев М.М.**

---

(подпись руководителя)

Работа представлена

«\_\_\_» \_\_\_\_\_ 2025 г.

Допущен к работе

«\_\_\_» \_\_\_\_\_ 2025 г.

Москва 2025

## **ЦЕЛЬ РАБОТЫ**

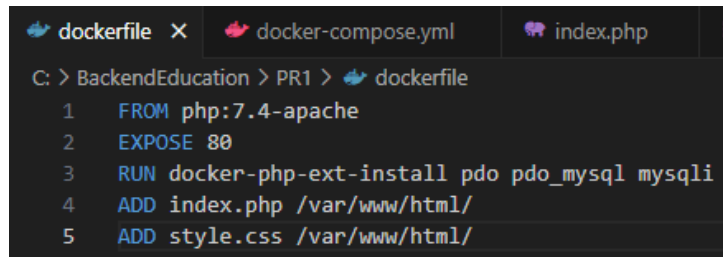
Предлагается создать свою конфигурацию серверного программного обеспечения, в которой должны присутствовать веб-сервер, операционная система, язык программирования и база данных. Данная конфигурация будет использоваться для выполнения следующих практических работ по данной дисциплине и для выполнения курсового проектирования.

Дается рекомендация использовать ОС Linux, язык программирования PHP, веб-сервер Apache и СУБД MySQL.

Для проверки работоспособности вашей конфигурации требуется инициализировать базу данных: создать отдельного пользователя для работы с ней, создать базу данных, в которой создать таблицу пользователи с полями: идентификационный номер, имя, фамилия. Также для проверки вашей конфигурации требуется сгенерировать тестовую страничку, содержащую выборку из созданной таблицы и информационное сообщение о версии языка программирования, его настройках и конфигурации.

## ХОД РАБОТЫ

Для создания образа необходимого веб-сервера, был использован Dockerfile, изображенный на рисунке 1.



```
dockerfile X docker-compose.yml index.php
C: > BackendEducation > PR1 > dockerfile
1 FROM php:7.4-apache
2 EXPOSE 80
3 RUN docker-php-ext-install pdo pdo_mysql mysqli
4 ADD index.php /var/www/html/
5 ADD style.css /var/www/html/
```

Рисунок 1 – Dockerfile задания

Здесь в качестве основы для нашего образа мы используем официальный образ PHP, затем копируем содержимое сервера, находящееся в текущей директории, в файловую систему веб-сервера и устанавливаем `mysqli` для корректной работы приложения.

Для связи сервера и его базы данных мы будем использовать `docker compose`. Его содержимое показано на рисунке 2.



```
dockerfile X docker-compose.yml X index.php init.sql
C: > BackendEducation > PR1 > docker-compose.yml
1 version: "3"
2 name: sspd_work1
3
4 services:
5   db:
6     image: mysql
7     restart: always
8     environment:
9       MYSQL_ROOT_PASSWORD: rootpass
10      MYSQL_DATABASE: appdb
11      MYSQL_USER: appuser
12      MYSQL_PASSWORD: apppass
13     ports:
14       - "3306:3306"
15     volumes:
16       - sspd_work1:/var/lib/mysql
17       - ./init.sql:/docker-entrypoint-initdb.d/init.sql
18
19   web_server:
20     build:
21       dockerfile: dockerfile
22     restart: always
23     depends_on:
24       - db
25     ports:
26       - "8000:80"
27
28 volumes:
29   sspd_work1:
```

Рисунок 2 – docker-compose.yml

Здесь есть два сервиса: веб-сервер и база данных. При этом веб-сервер зависит от базы данных. В настройках сервера мы указываем Dockerfile, который создаст нужный нам образ, а также порты и тома для работы с нужными нам файлами сервера.

В сервисе базы данных мы, в свою очередь, задаем ее базовый образ и указываем переменные окружения: пользователя, название базы данных, корневой и пользовательский пароли. Помимо этого, мы указываем том для корректной инициализации базы данных на основе файла init.sql.

На рисунке 3 показана итоговая файловая структура проекта.

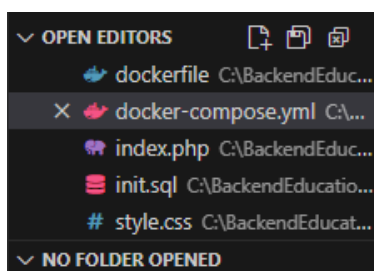


Рисунок 3 – Корневая папка

Проверим работу нашего сервера, выполнив команду “docker-compose up”, что показано на рисунках 4 и 5.

```
PS C:\WINDOWS\system32> cd C:\BackendEducation\PR1
PS C:\BackendEducation\PR1> docker-compose up -d
time="2025-09-11T19:14:28+03:00" level=warning msg="C:\BackendEducation\PR1\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 11/11
✔ db Pulled
✔ 1421f5b704d5 Pull complete
✔ 01859c60b4e2 Pull complete
✔ 7d70b564625b Pull complete
✔ 87565b56b57e Pull complete
✔ 9b2f3769f0be Pull complete
✔ d210a5b69bfe Pull complete
✔ 500d7b2546c4 Pull complete
✔ 1d289f7d1ed9 Pull complete
✔ 95f5cacia9e9 Pull complete
✔ 98045e6cd572 Pull complete
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 33.3s (11/11) FINISHED
=> [web_server internal] load build definition from dockerfile
=> => transferring dockerfile: 178B
=> [web_server internal] load metadata for docker.io/library/php:7.4-apache
=> [web_server auth] library/php:pull token for registry-1.docker.io
=> [web_server internal] load .dockerignore
=> => transferring context: 2B
=> [web_server 1/4] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d190 18.8s
=> => resolve docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2 0.0s
=> sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 2.46kB / 2.46kB 0.3s
=> sha256:05e465aaa99a358add4acccdade8f39843089069f31fea0201533d3a09a98c9a 892B / 892B 0.4s
=> sha256:80692ae2d067c8358112c56490a2a97f69ef395f8f7662a31498644c9a813ef 246B / 246B 0.6s
=> sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 10.20MB / 10.20MB 1.6s
=> sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81c8b1c40cb 491B / 491B 0.4s
=> sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 10.76MB / 10.76MB 6.8s
=> sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 514B / 514B 0.8s
=> sha256:9b233e420ac7bbc0a645bb82c213029762acf1742400c076360dc303213c309d5 475B / 475B 0.4s
=> sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 19.25MB / 19.25MB 5.8s
=> sha256:f05a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 270B / 270B 0.3s
=> sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376ad9169351a1265382e39de 91.63MB / 91.63MB 12.3s
=> sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 226B / 226B 1.1s
=> sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 31.41MB / 31.41MB 13.0s
=> extracting sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 0.6s
=> extracting sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 0.0s
=> extracting sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376ad9169351a1265382e39de 1.3s
=> extracting sha256:f05a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 0.0s
=> extracting sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 0.3s
=> extracting sha256:9b233e420ac7bbc0a645bb82c213029762acf1742400c076360dc303213c309d5 0.0s
=> extracting sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 0.0s
```

Рисунок 4 – Запуск сервера

```

=> => extracting sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 0.0s
=> => extracting sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 0.0s
=> => extracting sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 0.0s
=> => extracting sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 0.2s
=> => extracting sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 0.0s
=> => extracting sha256:80692ae2d067c8358112c56490a2a97f69ef395fd8f7662a31498644c9a813ef 0.0s
=> => extracting sha256:05e465aaa99a358add4acecdade8f39843089069f31fea0201533d3a09a98c9a 0.0s
=> [web_server internal] load build context 0.1s
=> => transferring context: 1.13kB 0.0s
=> [web_server 2/4] RUN docker-php-ext-install pdo pdo_mysql mysqli 11.8s
=> [web_server 3/4] ADD index.php /var/www/html/ 0.0s
=> [web_server 4/4] ADD style.css /var/www/html/ 0.1s
=> [web_server] exporting to image 0.3s
=> => exporting layers 0.2s
=> => exporting manifest sha256:ac22dcdb88166d11cf910f032719fcd023a95ae52ca5f9fb0e4f1943a9ea760b 0.0s
=> => exporting config sha256:8b3ae933d6d23c4ece638a3e3ac3883777f9a3358781a32844688d2e46e77c56 0.0s
=> => exporting attestation manifest sha256:4cb77b13c8f01ae9a5005caeb8c3a204c36fb38d3cda5a0b09e4d3e803145562 0.0s
=> => exporting manifest list sha256:1fa8e04ea9bd71d7e32289f1d6c577207781e50711163306028d6e8e57e5e44b 0.0s
=> => naming to docker.io/library/sspd_work1-web_server:latest 0.0s
=> => unpacking to docker.io/library/sspd_work1-web_server:latest 0.1s
=> [web_server] resolving provenance for metadata file 0.0s
[+] Running 5/5
✔ web_server Built 0.0s
✔ Network sspd_work1_default Created 0.0s
✔ Volume "sspd_work1_ssdp_work1" Created 0.0s
✔ Container sspd_work1-db-1 Started 0.6s
✔ Container sspd_work1-web_server-1 Started 0.6s
PS C:\BackendEducation\PR1>

```

Рисунок 5 – Запуск сервера

Сервер в docker запущен и работает. Результат запуска на рисунке 6.

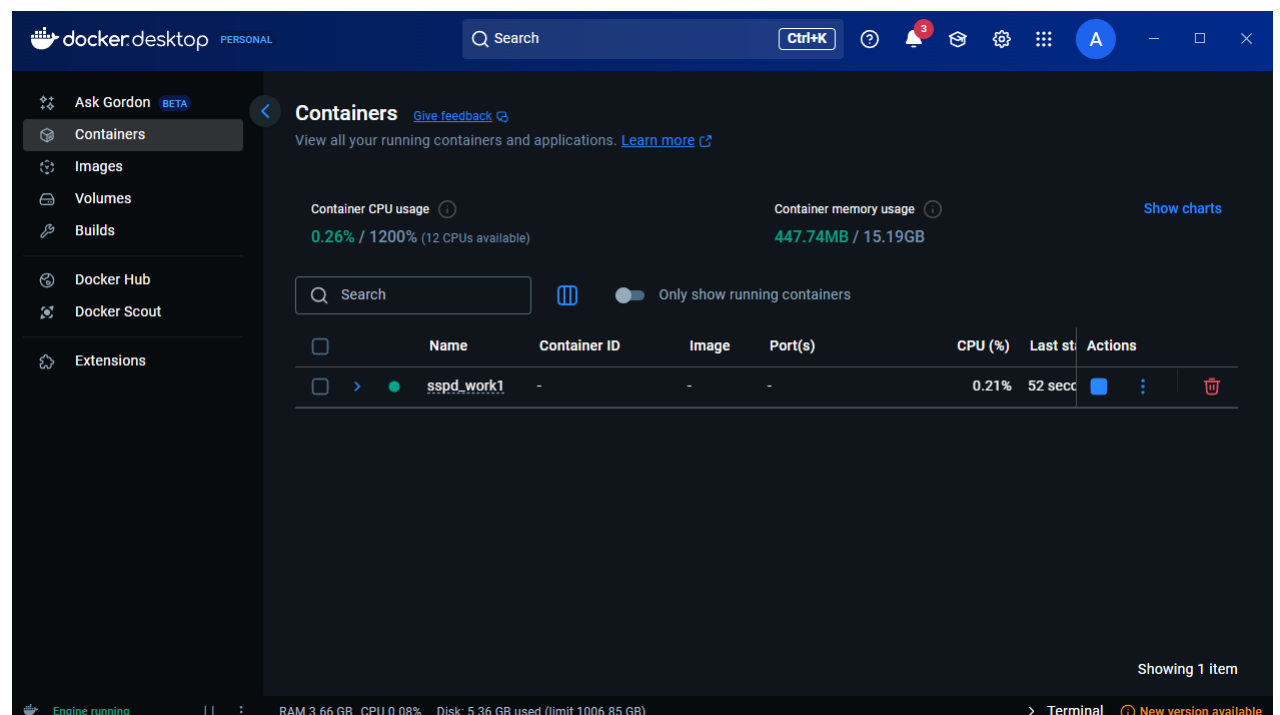


Рисунок 6 – Запущенный сервер в Docker

Теперь можем перейти на localhost и убедиться, что все работает корректно, как показано на рисунке 7.

←localhost:8000

Таблица пользователей данного продукта

Идентификатор	Имя	Фамилия
1	Алекс	Соталац
2	Отвес	Марин
3	Кейт	Янсон
4	Лео	Черный

Версия PHP 7.4.33



Система	Linux afd68b4c3a2 5.15.167.4-microsoft-standard-WSL2 #1 SMP Fri Nov 5 00:21:55 UTC 2024 x86_64
Дата постройки	Nov 15, 2022 06:03:12
Команда Configure	"configure" --build=x86_64-linux-gnu --with-enable-ftp --enable-mbstring --enable-mysqlnd --with-with-password-argon2 --with-sodium-shared --with-pdo-sqlite=lib --with-sqlite3=lib --with-curl --with-iconv --with-openssl --with-readline --with-zlib --disable-openssl --with-pear --with-pear --with-ibdb=libx86_64-linux-gnu --disable-cgi --with-apxs2 --build_alias=x86_64-linux-gnu
API сервера	Обработка Apache 2.0
Поддержка виртуального каталога	нетрудоспособный
Путь к файлу конфигурации (php.ini)	/usr/local/etc/php
Загруженный файл конфигурации	(нет)
Отсканируйте этот каталог на наличие дополнительных файлов .ini	/usr/local/etc/php/conf.d
Проанализированы дополнительные файлы .ini	/usr/local/etc/php/conf.d/docker-php-ext-mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
RHP API	20190902
Расширение RHP	20190902
Расширение Zend	320190902
Сборка расширения Zend	API20190902.NTS
Сборка расширения RHP	API20190902.HTC
Отладочная сборка	Нет
Безопасность натай	нетрудоспособный
Обработка сигналов Zend	Включен
Менеджер памяти Zend	Включен
Поддержка Zend Multibyte	Предоставлено mbstring
Поддержка IPv6	Включен
Поддержка DTase	нетрудоспособный
Зарегистрированные потоки RHP	https, fips, compress.zlib, php, file, glob, data, http, ftp, phar
Транспорты зарегистрированных потоковых соединений	TCP, UDP, Unix, UDQ, SSL, TLS, TLSV1.0, TLSV1.1, TLSV1.2, TLSV1.3
Фильтры зарегистрированных потоков	zlib, "convert.iconv", string.rot13, string.toupper, string.tolower, string.strip_tags, convert, "неподключеный, dechunk

Рисунок 7 – Проверка работы сервера

## **ВЫВОД**

Таким образом, был произведен корректный запуск приложенного к практической работе php скрипта генерации страницы с характеристиками веб-сервера.

# ОТВЕТЫ НА ВОПРОСЫ

1. Сервер и клиент – Сервер предоставляет ресурсы/услуги, клиент их потребляет (например, веб-сервер и браузер).
2. База данных – Организованное хранилище данных (например, MySQL), управляемое СУБД.
3. API – Интерфейс для взаимодействия между программными компонентами (например, REST API).
4. Сервис, отличия от сервера – Сервис выполняет конкретную функцию (например, авторизация), сервер — физическая/виртуальная система, предоставляющая сервисы.
5. Архитектура клиент-сервер – Модель взаимодействия, где клиент запрашивает услуги, а сервер их предоставляет.
6. Виды сервисов – Веб-сервисы, микросервисы, облачные сервисы и т.д.
7. Масштабируемость – Способность системы работать при увеличении нагрузки (горизонтальная/вертикальная).
8. Протоколы передачи данных – Правила обмена данными (HTTP, TCP/IP, FTP).
9. Тонкий и толстый клиенты – Тонкий клиент минимально загружен логикой (браузер), толстый — содержит больше функций (десктоп приложение).
10. Паттерн MVC: общие тезисы – Разделение приложения на Model (данные), View (отображение), Controller (логика).
11. MVC: Model-View-Presenter – Presenter mediates between View and Model, обрабатывает пользовательский ввод.
12. MVC: Model-View-View Model (MVVM) – View Model обеспечивает связь данных с View через привязки.
13. MVC: Model-View-Controller – Controller обрабатывает input, обновляет Model и View.
14. Docker: общие тезисы – Технология контейнеризации для изоляции и развёртывания приложений.
15. Dockerfile – Скрипт для сборки Docker-образа (инструкции: FROM, RUN, COPY и т.д.).



16. Docker Compose – Инструмент для оркестровки многоконтейнерных приложений.

17. LAMP – Стандартный стек серверного ПО: Linux, Apache, MySQL, PHP.

## ССЫЛКА НА GITHUB

Ссылка на практику: <https://github.com/AkitaUI/BackPR1>

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Документация // PHP URL: <https://www.php.net/manual/ru/index.php> (дата обращения: 10.09.2025). – Текст: электронный.
2. Статья о назначении докера простыми <https://habr.com/ru/post/309556/> (дата обращения: 10.09.2025).
3. словами: Методические указания по выполнению практической работы: <https://online-edu.mirea.ru/mod/resource/view.php?id=508421> (дата обращения: 10.09.2025).
4. Официальная документация докера: <https://docs.docker.com/> (дата обращения: 10.09.2025).
5. Более сложная и подробная статья <https://habr.com/ru/post/277699/> (дата обращения: 10.09.2025).