



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

Муравьев А.О.

(подпись студента)

Руководитель практической работы

Благирев М.М.

(подпись руководителя)

Работа представлена

«___» _____ 2025 г.

Допущен к работе

«___» _____ 2025 г.

Москва 2025

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	1
ВВЕДЕНИЕ.....	ОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.
ВЫПОЛНЕНИЕ РАБОТЫ	4
ВЫВОД.....	ОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ	10

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, изображённый на рисунке 1.

```
1 FROM php:8.2-apache
2
3 RUN apt-get update && apt-get install -y \
4     libpng-dev \
5     libjpeg-dev \
6     libfreetype6-dev \
7     zlib1g-dev \
8     libzip-dev \
9     default-mysql-client \
10    && rm -rf /var/lib/apt/lists/*
11
12 RUN docker-php-ext-configure gd --with-freetype --with-jpeg \
13     && docker-php-ext-install -j$(nproc) gd \
14     && docker-php-ext-install pdo pdo_mysql mysqli zip
15
16 RUN a2enmod rewrite
17
18 COPY index.php /var/www/html/
19 COPY style.css /var/www/html/
20
21 RUN chown -R www-data:www-data /var/www/html \
22     && chmod -R 755 /var/www/html
23
24 EXPOSE 80
25
26 CMD ["apache2-foreground"]
```

Рисунок 1 – Dockerfile для 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 `dockercompose`. Его содержимое показано на рисунке 2.

```
1  version: '3.8'
2  name: sspd_work1
3  services:
4
5      db:
6          image: mysql:8.0
7          container_name: mysql_db
8          volumes:
9              - mysql_data:/var/lib/mysql
10             - ./init.sql:/docker-entrypoint-initdb.d/init.sql
11          restart: always
12          environment:
13              MYSQL_ROOT_PASSWORD: toor
14              MYSQL_DATABASE: appDB
15              MYSQL_USER: user
16              MYSQL_PASSWORD: password
17          ports:
18              - "3306:3306"
19          networks:
20              - app_network
21
22      web_server:
23          build: .
24          container_name: php_apache
25          ports:
26              - "8000:80"
27          restart: always
28          depends_on:
29              - db
30          volumes:
31              - ./index.php:/var/www/html/index.php
32              - ./style.css:/var/www/html/style.css
33          networks:
34              - app_network
35
36  volumes:
37      mysql_data:
38
39  networks:
40      app_network:
41          driver: bridge
```

Рисунок 2 – Файл 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.

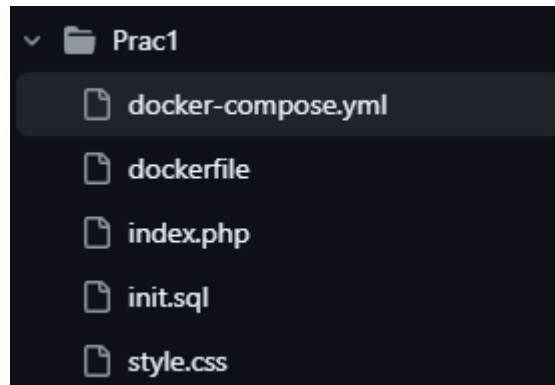


Рисунок 3 – Структура проекта

Соберём проект с помощью команды docker compose build (рисунок 4).

```
=> => extracting sha256:ce1261c6d56/ef6a8e1b4576/3eeeb4/74a0a8066df6bb95ca9a6a94a31e219dd3 0.7s
=> => extracting sha256:48ebd04f4983bbe65e67c4185e1ec3bcde92727bb40e759c0c908c67da8061b5 0.0s
=> => extracting sha256:70bb6b41f165fc132a739c330a5a9c36d7cf2802afa513c5022f271faed87e96 2.2s
=> => extracting sha256:945024922144282c29a6aa88270e089fd811e36be4e488ef28c166cc98d359a2 0.0s
=> => extracting sha256:2e432d6184b2b78c83ff23e2cb792f037e3b01d66f09cfff2051d47877d15794b 0.2s
=> => extracting sha256:d4f032b51ed9b861f00e3782d8dd2892abc8da6af9cfa7d21a0ee2bca2db8c17 0.0s
=> => extracting sha256:c4f07164d4bff49b210440820f12de98d43b09b593147f17bd8b457886c9eefd 0.0s
=> => extracting sha256:7ba7adab300af10e637c47db3381561c12f92843a165c99aa16dca20a7f087df 0.1s
=> => extracting sha256:08a8f3c4034a4d970c604a004455d8e4ec3032a9fe9a9ee9204d360fec19e4fc 0.0s
=> => extracting sha256:aecd457091159dd6b03d3fd06cc2b313344e97d5aeb416dab36ae45ae4551a 0.3s
=> => extracting sha256:80da6eb38538300c90fc603f0f20b91644aa344ce7ddda7b4d196c6ffaf7bc11 0.0s
=> => extracting sha256:5a7c38055062139483605d08b890258311fec7f9b52d3b190c80b56439e913ae 0.0s
=> => extracting sha256:9dcc750708704ba428714146884f1f200117fe7cd72bfc09d8247acc56a9980f 0.0s
=> => extracting sha256:1f8d884fef54398fda475516682678bd45554ba28b2f3cc5efa4ec421bae67f2 0.0s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 0.0s
=> [web_server internal] load build context
=> => transferring context: 1.13kB 0.0s
=> [web_server 2/7] RUN apt-get update && apt-get install -y libpng-dev libjpeg-dev libfreetype6-de 29.0s
=> [web_server 3/7] RUN docker-php-ext-configure gd --with-freetype --with-jpeg && docker-php-ext-install - 17.8s
=> [web_server 4/7] RUN a2enmod rewrite 0.5s
=> [web_server 5/7] COPY index.php /var/www/html/ 0.1s
=> [web_server 6/7] COPY style.css /var/www/html/ 0.1s
=> [web_server 7/7] RUN chown -R www-data:www-data /var/www/html && chmod -R 755 /var/www/html 0.5s
=> [web_server] exporting to image 0.4s
=> => exporting layers 0.4s
=> => writing image sha256:8352ebbc3186984fbbec3275864899d13e54331d2a07a6406d618465579ce0ff 0.0s
=> => naming to docker.io/library/sspd_work1-web_server 0.0s
=> [web_server] resolving provenance for metadata file 0.0s
C:\Users\Alexomur\Desktop\projects\MireaBackend\Prac1>
```

Рисунок 4 – Сборка проекта

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

```
Командная строка - docker compose up
mysql_db | 2025-09-11T16:15:41.845630Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.43) MySQL Community Server - GPL.
mysql_db | 2025-09-11 16:15:41+00:00 [Note] [Entrypoint]: Temporary server stopped
mysql_db | 2025-09-11 16:15:41+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
mysql_db | 2025-09-11T16:15:41.781012Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
mysql_db | 2025-09-11T16:15:41.781747Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.43) starting as process 1
mysql_db | 2025-09-11T16:15:41.787889Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql_db | 2025-09-11T16:15:42.287938Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql_db | 2025-09-11T16:15:42.618997Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql_db | 2025-09-11T16:15:42.619029Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql_db | 2025-09-11T16:15:42.632620Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql_db | 2025-09-11T16:15:42.646691Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql_db | 2025-09-11T16:15:42.646751Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.43' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
php_apache | 172.19.0.1 - - [11/Sep/2025:16:16:51 +0000] "GET / HTTP/1.1" 200 25150 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
php_apache | 172.19.0.1 - - [11/Sep/2025:16:16:51 +0000] "GET /style.css HTTP/1.1" 200 587 "http://127.0.0.1:8000/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
php_apache | 172.19.0.1 - - [11/Sep/2025:16:16:51 +0000] "GET /favicon.ico HTTP/1.1" 404 489 "http://127.0.0.1:8000/" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
View in Docker Desktop View Config Enable Watch
```

Рисунок 5 – Запуск проекта

Оба сервиса запущены и работают, что видно в Docker Desktop (рисунок 6).

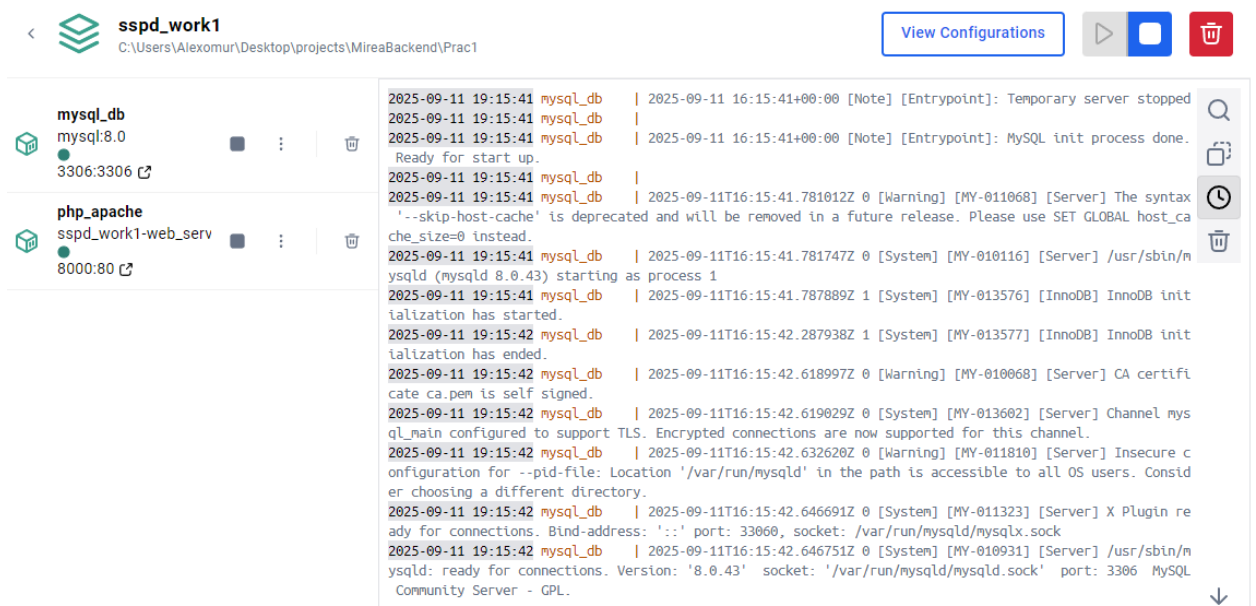


Рисунок 6 – Сервисы в Docker Desktop

Теперь по адресу localhost:8000 (или 127.0.0.1:8000) можно увидеть сайт, продемонстрированный на рисунке 7.


127.0.0.1:8000

УтилитаExled9УчебаSZDiscohookОфициальный сайт...ПеревестиGmailМузыкаSpeedtestТаблица tarkovWordTHМой МТС - Личны...МТС 8920FTHiLinkВывод Eth

Таблица пользователей данного продукта

	Id	Name	Surname
1	Alex	Rover	
2	Bob	Marley	
3	Kate	Yandson	
4	Lito	Black	

PHP Version 8.2.29



System	Linux 38b461f06240 5.15.153.1-microsoft-standard-WSL2 #1 SMP Fri Mar 29 23:14:13 UTC 2024 x86_64
Build Date	Sep 8 2025 21:21:19
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	'/configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-opcache' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-gd.ini, /usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-opcache.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini, /usr/local/etc/php/conf.d/docker-php-ext-zip.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829.NTS
PHP Extension Build	API20220829.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled

Рисунок 7 – Проверка работы проекта

ВЫВОД

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

Исходный код проекта расположен по адресу:

<https://github.com/alexomur/MireaBackend/tree/master/Prac1>

ОТВЕТЫ НА КОНТРОЛЬНЫЕ 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 (десктопприложение).
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. 7
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.