

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

ОТЧЁТ  
по индивидуальной практической работе №1  
по дисциплине  
«Интернет-технологии и веб-программирование»  
Вариант 15

Выполнил

ст. гр. 220604  
А.И. Михнюк

Проверила

Н.В. Хаджинова

Минск 2025

# 1 ЦЕЛЬ РАБОТЫ

Основной целью данной практической работы являлось приобретение комплексных навыков разработки полнофункционального веб-сервиса с использованием современных интернет-технологий. Работа была направлена на освоение полного цикла создания веб-приложения - от проектирования пользовательского интерфейса до организации безопасного взаимодействия с базой данных и развертывания решения в контейнеризованной среде. В рамках данного проекта требовалось разработать надежное и безопасное веб-приложение, соответствующее промышленным стандартам качества, что включало в себя реализацию многоуровневой системы валидации данных, защиту от распространенных веб-уязвимостей и обеспечение отказоустойчивости системы при обработке пользовательских запросов.

В работе ставилась задача глубокого изучения принципов разделения клиентской и серверной частей приложения, где фронтенд-компонент отвечает за предоставление интуитивно понятного пользовательского интерфейса для ввода данных, а бэкенд-обработчик обеспечивает комплексную валидацию, обработку и надежное хранение информации в реляционной базе данных. Особое внимание уделялось вопросам безопасности, включая защиту от *SQL*-инъекций, межсайтового скриптинга (*XSS*) и других распространенных атак, что достигалось за счет использования подготовленных выражений для работы с базой данных и тщательной валидацией всех пользовательских входных данных перед обработкой.

Важным аспектом работы стало освоение современных методик контейнеризации приложений с использованием *Docker* и *Docker Compose*, что позволяет создавать переносимые, масштабируемые и легко развертываемые решения, не зависящие от среды выполнения. Этот подход обеспечивает стандартизацию процесса разработки и значительно упрощает развертывание приложения в различных окружениях, а также способствует лучшему пониманию *DevOps*-практик в контексте веб-разработки.

В результате выполнения работы были получены практические навыки, необходимые для создания современных веб-сервисов, способных эффективно обрабатывать пользовательские данные, обеспечивать их целостность и безопасность, а также предоставлять понятную обратную связь пользователям о результатах выполнения операций. Освоенные технологии и методологии позволяют в дальнейшем разрабатывать более сложные веб-приложения с использованием микросервисной архитектуры, *RESTful API* и современных фреймворков, что составляет основу профессиональной деятельности в области веб-программирования и интернет-технологий.

## 2 ХОД РАБОТЫ

### 2.1 Анализ технического задания и проектирование системы

На начальном этапе выполнения работы был проведен детальный анализ технического задания, в рамках которого изучались требования индивидуального варианта №15 «Подача объявления».



15	Подача объявления	ad_title (VARCHAR), ad_category (VARCHAR), price (DECIMAL), contact_email (VARCHAR)
----	-------------------	---

Рисунок 1 – Индивидуальный вариант задания

Данный анализ позволил определить ключевые функциональные возможности будущего веб-сервиса, включая необходимость предоставления пользователям возможности создания объявлений с указанием заголовка, категории, цены, контактной информации и подробного описания предлагаемых товаров или услуг. Особое внимание уделялось пониманию бизнес-логики приложения и определению оптимальных способов организации взаимодействия между пользователем и системой, что являлось фундаментом для последующих этапов проектирования и разработки.

На основе проведенного анализа были определены структура данных и состав полей формы, которые должны были обеспечить полноценное функционирование сервиса подачи объявлений. Для каждого поля формы проводилась оценка необходимого типа данных, допустимой длины и требований к валидации, что впоследствии нашло отражение как в структуре базы данных, так и в клиентской и серверной частях приложения. Проектирование архитектуры веб-сервиса осуществлялось с учетом принципов разделения ответственности, где клиентская часть отвечает за представление данных и взаимодействие с пользователем, а серверная - за обработку бизнес-логики, валидацию и хранение информации. В качестве технологического стека были выбраны *HTML* для создания пользовательского интерфейса, *PHP* для реализации серверной логики, *MySQL* для хранения данных и *Docker* для контейнеризации приложения, что в совокупности обеспечивало надежную, масштабируемую и легко развертываемую платформу для решения поставленных задач.

## 2.2 Проектирование и создание базы данных

Проектирование базы данных являлось критически важным этапом разработки веб-сервиса, поскольку от корректности структуры хранения информации напрямую зависела надежность и производительность всего приложения. Была создана специализированная база данных *advertisement\_db*, предназначенная исключительно для работы с системой подачи объявлений, что обеспечивало изоляцию данных и упрощало администрирование. При выборе имени базы данных использовался принцип уникальности, что соответствует требованиям индивидуальной работы и позволяет избежать конфликтов в рабочей среде.

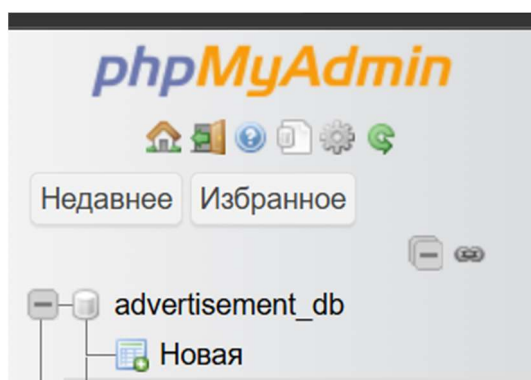


Рисунок 2 – Новая БД в MySQL

В рамках созданной базы данных разработана таблица *advertisements*, структура которой тщательно продумана для эффективного хранения всех необходимых данных объявлений. Поле *ad\_title* типа *VARCHAR* предназначено для хранения заголовков объявлений с ограничением длины, что обеспечивает оптимальное использование дискового пространства и соответствует требованиям к отображению кратких заголовков в интерфейсе. Для категорий объявлений использовано поле *ad\_category* также типа *VARCHAR*, что позволяет гибко управлять категориальной системой без необходимости частого изменения структуры базы данных. Особое внимание уделено полю *price* типа *DECIMAL*, которое обеспечивает точное хранение денежных значений с фиксированной точностью, исключая ошибки округления, характерные для типов с плавающей точкой.

Для хранения контактной информации предусмотрено поле *contact\_email* типа *VARCHAR* с соответствующими ограничениями длины, что соответствует стандартам хранения электронных адресов. Наиболее объемные текстовые описания размещаются в поле *ad\_text* типа *TEXT*, которое

специально разработано для хранения больших объемов текстовой информации без жестких ограничений по длине. Дополнительно в таблицу включены служебные поля: уникальный идентификатор *id* с автоинкрементом, обеспечивающий однозначную идентификацию каждой записи, и поле *created\_at* с временной меткой создания объявления, что позволяет в дальнейшем реализовать функции сортировки и фильтрации по дате.

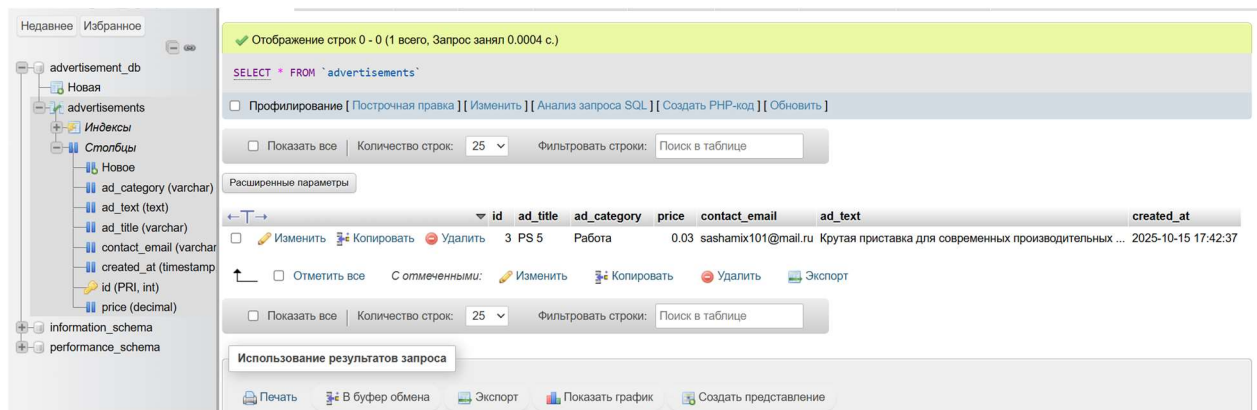


Рисунок 3 – Структура рабочей страницы и данные

Для автоматизации процесса развертывания базы данных был разработан *SQL*-скрипт инициализации *init.sql*, содержащий все необходимые команды для создания структуры базы данных и таблиц.

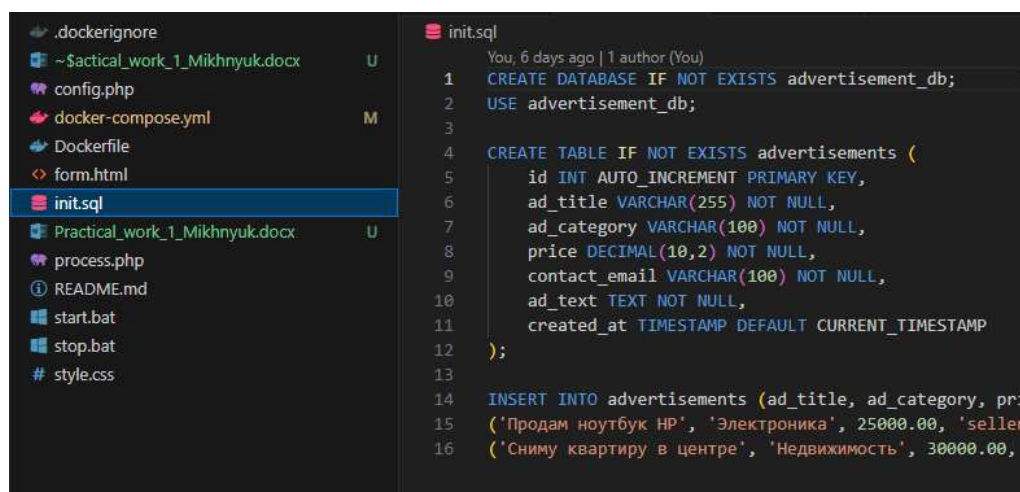


Рисунок 4 – Скрипт автоматической инициализации

Такой подход значительно упрощает процесс *deployment* в различных окружениях, включая *Docker*-контейнеры.

## 2.3 Разработка клиентской части

Разработка клиентской части веб-сервиса началась с создания *HTML*-формы *form.html*,

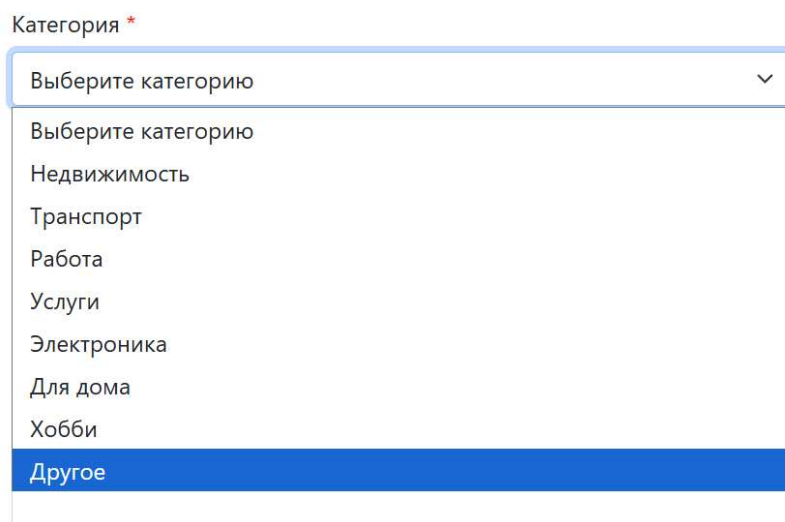
Рисунок 5 – Визуализация страницы *form.html*

которая представляет собой основной интерфейс взаимодействия пользователя с системой подачи объявлений. Форма была тщательно спроектирована для обеспечения максимальной удобства использования и включала все необходимые поля для ввода информации об объявлении. Первым элементом формы стало поле ввода заголовка объявления, для которого были установлены соответствующие ограничения по длине и обязательность заполнения,

Рисунок 6 – Поле заголовка объявления

что гарантирует сохранение целостности данных и соответствие структуре базы данных. Особое внимание было уделено *usability* - полю добавлен *placeholder* с примером содержимого, что помогает пользователю понять ожидаемый формат ввода.


Для выбора категории объявления реализован выпадающий список с predetermined вариантами, включающими наиболее распространенные направления: недвижимость, транспорт, работа, услуги, электроника и другие.



The image shows a web form element labeled "Категория \*" in red. Below the label is a dropdown menu with a light blue border. The menu's header is "Выберите категорию" with a small downward arrow on the right. The menu is open, showing a list of options: "Выберите категорию", "Недвижимость", "Транспорт", "Работа", "Услуги", "Электроника", "Для дома", "Хобби", and "Другое". The "Другое" option is highlighted with a solid blue background.

Рисунок 7 – Выпадающий список выбора категории

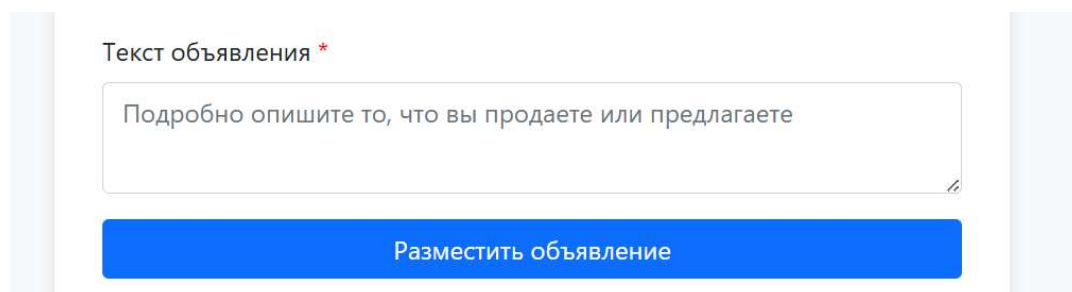
Такой подход не только упрощает процесс заполнения формы для пользователя, но и обеспечивает стандартизацию данных, что впоследствии облегчает их обработку и категоризацию. Поле ввода цены было оснащено клиентской валидацией числовых значений с установкой минимального порога, что предотвращает ввод некорректных данных на самом раннем этапе. Для *email*-адреса использована встроенная браузерная валидация формата, дополненная серверной проверкой для обеспечения максимальной надежности.



The image shows a web form element labeled "Цена (руб.) \*" in red. Below the label is a text input field with a light blue border. The field contains the text "0,03". To the right of the input field is a vertical spinner control with up and down arrows.

Рисунок 8 – Строка установки цены со стрелкой-счётчиком

Текстовая область для описания объявления была спроектирована с учетом необходимости размещения развернутого контента - установлена достаточная высота поля и реализована поддержка многострочного ввода.



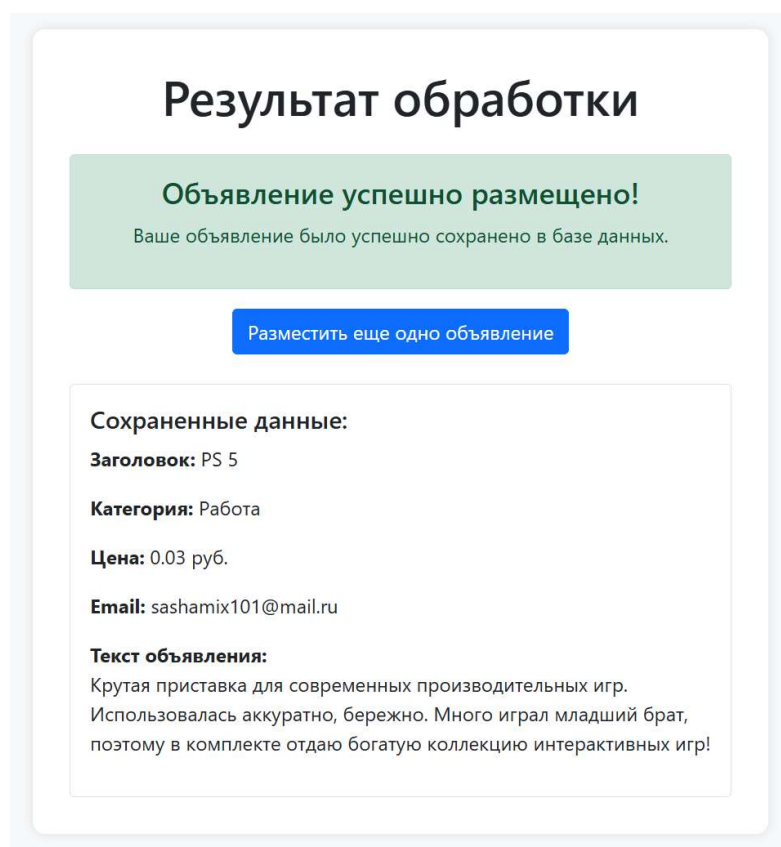
Текст объявления \*

Подробно опишите то, что вы продаете или предлагаете

Разместить объявление

Рисунок 9 – Поле для ввода текста объявления с расширением

Для обеспечения адаптивного дизайна, гарантирующего корректное отображение формы на устройствах с различными размерами экранов, был использован фреймворк *Bootstrap*. Это позволило создать современный, визуально привлекательный интерфейс с грамотно организованными отступами, гармоничной цветовой схемой и интуитивно понятной структурой.



## Результат обработки

**Объявление успешно размещено!**

Ваше объявление было успешно сохранено в базе данных.

Разместить еще одно объявление

**Сохраненные данные:**

**Заголовок:** PS 5

**Категория:** Работа

**Цена:** 0.03 руб.

**Email:** sashamix101@mail.ru

**Текст объявления:**

Крутая приставка для современных производительных игр. Использовалась аккуратно, бережно. Много играл младший брат, поэтому в комплекте отдаю богатую коллекцию интерактивных игр!

Рисунок 10 – Ответ об успешной записи в БД



Ключевые атрибуты формы, такие как *method="POST"* и *action="process.php"*, были настроены для обеспечения безопасной передачи данных на серверную часть приложения, что составляет основу корректного взаимодействия между клиентской и серверной компонентами веб-сервиса.

## 2.4 Разработка серверной части

Разработка серверной компоненты веб-сервиса осуществлялась через создание *PHP*-обработчика *process.php*, который выступает в качестве центрального узла для приема, обработки и сохранения данных, поступающих от клиентской части приложения. Данный скрипт был спроектирован как многофункциональный обработчик, способный не только взаимодействовать с базой данных, но и выполнять комплексную проверку целостности и корректности поступающей информации. Архитектура обработчика построена по модульному принципу, где каждый блок кода отвечает за конкретную задачу, что обеспечивает легкость сопровождения и возможности для дальнейшего масштабирования функциональности.

Безопасное подключение к базе данных было организовано через отдельный конфигурационный файл *config.php*, в котором вынесены все параметры подключения, включая хост, имя пользователя, пароль и название базы данных. Такой подход не только повышает безопасность за счет изоляции критически важных данных, но и упрощает процесс миграции приложения между различными окружениями. В рамках разработки системы валидации данных была реализована многоуровневая проверка, включающая контроль заполнения обязательных полей, валидацию формата *email* с использованием встроенных *PHP*-функций, проверку корректности числовых значений на предмет соответствия ожидаемому диапазону и формату, а также контроль длины строковых полей для предотвращения переполнения соответствующих колонок в базе данных.

Особое внимание было уделено вопросам безопасности, в частности, защите от *SQL*-инъекций, для чего повсеместно применялись подготовленные выражения (*prepared statements*) вместо конкатенации строк при формировании *SQL*-запросов. Этот подход исключает возможность выполнения злонамеренного кода через поля ввода данных, обеспечивая надежную защиту базы данных. Организация обработки ошибок была реализована через систему сбора сообщений об ошибках в массив с последующим выводом пользователю в структурированном виде, что позволяет одновременно информировать пользователя о всех обнаруженных проблемах и предотвращает прерывание процесса выполнения скрипта при

обнаружении первой же ошибки. Для обеспечения лучшего пользовательского опыта были разработаны понятные сообщения об ошибках на русском языке, которые точно указывают на характер проблемы и возможные способы ее устранения.

## 2.5 Реализация безопасности и валидации данных

Обеспечение безопасности веб-сервиса являлось одним из наиболее критичных аспектов разработки, поскольку система обработки пользовательских данных потенциально подвержена различным видам кибератак. Комплексный подход к безопасности был реализован на нескольких уровнях, начиная с базового экранирования пользовательского ввода и заканчивая защитой от сложных уязвимостей. Экранирование входных данных выполнялось с помощью специализированных функций *PHP*, которые преобразуют потенциально опасные символы в безопасные для обработки эквиваленты, что предотвращает непреднамеренное выполнение кода и сохраняет целостность информации на всех этапах её обработки.

Для защиты от *XSS*-атак (межсайтового скриптинга) была систематически применена функция *htmlspecialchars()*, которая преобразует специальные *HTML*-символы в их сущности, делая невозможным выполнение вредоносных скриптов в браузере пользователя. Это особенно важно при выводе данных, полученных от пользователей, поскольку нейтрализует попытки внедрения *JavaScript*-кода через текстовые поля формы. Предотвращение *SQL*-инъекций обеспечивалось за счет повсеместного использования подготовленных выражений (*prepared statements*) при работе с базой данных, которые разделяют данные и команды *SQL*, полностью исключая возможность интерпретации пользовательского ввода как исполняемого кода базы данных.

Валидация данных на стороне сервера была организована как многоуровневый процесс, включающий проверки на различных этапах обработки запроса. Каждое поле формы подвергалось тщательному анализу на соответствие ожидаемому типу данных, формату и бизнес-логике приложения. Серверная валидация дополняла клиентскую, обеспечивая надежную защиту даже в случае обхода проверок на стороне браузера. Процесс санитизации входных данных включал не только удаление потенциально опасных конструкций, но и нормализацию данных к ожидаемому формату, что гарантировало их корректное хранение и дальнейшую обработку. Все эти меры в совокупности создали *robust*-систему, способную эффективно противостоять основным векторам атак на веб-

приложения и обеспечивающую надежную защиту как пользовательских данных, так и самой инфраструктуры приложения.

## 2.6 Развёртывание приложения

Контейнеризация приложения стала ключевым этапом в разработке веб-сервиса, обеспечивающим воспроизводимость среды выполнения и упрощающим процесс развёртывания на различных платформах. Основой для создания контейнера веб-сервера послужил *Dockerfile*, который был разработан на базе официального образа *php:8.1-apache*. В данном файле последовательно описаны все необходимые действия по настройке среды: установка системных зависимостей для работы с графикой и архивами, настройка расширений *PHP* для поддержки *MySQL*, включение модуля *mod\_rewrite* для *Apache*, а также копирование файлов приложения и настройка соответствующих прав доступа. Такой подход гарантирует, что веб-сервер будет содержать все необходимые компоненты для корректной работы *PHP*-приложения и взаимодействия с базой данных.

Оркестрация многокомпонентной архитектуры приложения была реализована через *docker-compose.yml*, который описывает три взаимосвязанных сервиса: веб-сервер на базе *Apache* с *PHP*, базу данных *MySQL* и административный интерфейс *PhpMyAdmin*. Каждый сервис был настроен с учетом его специфических требований: для веб-сервера определены переменные окружения для подключения к БД, настроены порты для доступа извне и обеспечена зависимость от сервиса базы данных; для *MySQL* заданы надежные пароли, настроены тома для сохранения данных и подключен скрипт инициализации; *PhpMyAdmin* сконфигурирован для автоматического подключения к базе данных. Сетевые порты были тщательно подобраны для избежания конфликтов с другими службами системы: 8080 для веб-приложения, 8081 для *PhpMyAdmin* и 3308 для прямого доступа к *MySQL*, что обеспечивает удобство отладки и администрирования.

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last start...	↓	Actions
<input type="checkbox"/>	itawp-individual-practical-work-1	-	-	-	0.35%	22 minutes ago		
<input type="checkbox"/>	phpmyadmin-1	f99cdc4ae339	<a href="#">phpmyadmin/phpmyadmin</a>	<a href="#">8081:80</a>	0%	22 minutes ago		
<input type="checkbox"/>	web-1	5b15f4760ef	<a href="#">itawp-individual-practical-work-1-web</a>	<a href="#">8080:80</a>	0%	22 minutes ago		
<input type="checkbox"/>	db-1	6d222bdd0eb3	<a href="#">mysql:8.0</a>	<a href="#">3308:3306</a>	0.35%	22 minutes ago		

Рисунок 11 – Все контейнеры и их порты

Для автоматизации процессов управления жизненным циклом приложения были разработаны скрипты *start.bat* и *stop.bat*, предназначенные для операционных систем *Windows*. Эти скрипты не только упрощают запуск и остановку всей системы одной командой, но и выполняют предварительные проверки на наличие установленного *Docker*, корректность его работы и доступность требуемых портов. Внутренняя логика скриптов включает обработку ошибок и предоставление понятных сообщений пользователю, что делает работу с приложением комфортной даже для неопытных пользователей.

```
PS E:\BSUIR\ITaWP\DistanceEducation2semester\ITaWP-Individual-practical-work-1> .\start.bat

=====
Запуск веб-сервиса для подачи объявлений
=====

Проверка зависимостей... OK

Запуск контейнеров (это может занять несколько минут при первом запуске)...
time="2025-10-15T20:53:35+03:00" level=warning msg="E:\BSUIR\ITaWP\DistanceEducation2semester\ITaWP-Individual-practical-work-1\start.bat is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
✓ Container itawp-individual-practical-work-1-db-1      Running
✓ Container itawp-individual-practical-work-1-phpmyadmin-1 Running
✓ Container itawp-individual-practical-work-1-web-1     Running

=====
Приложение успешно запущено!
=====

Доступные сервисы:
- Веб-приложение: http://localhost:8080/form.html
- PhpMyAdmin:    http://localhost:8081
- База данных:   localhost:3308

Данные для входа в PhpMyAdmin:
- Сервер: db
- Пользователь: root
- Пароль: rootpassword

Для остановки приложения запустите stop.bat

Press any key to continue . . .
```

Рисунок 12 – Лог исполнения скрипта *start.bat*

Использование томов данных для хранения состояния базы данных гарантирует сохранность информации между перезапусками контейнеров,

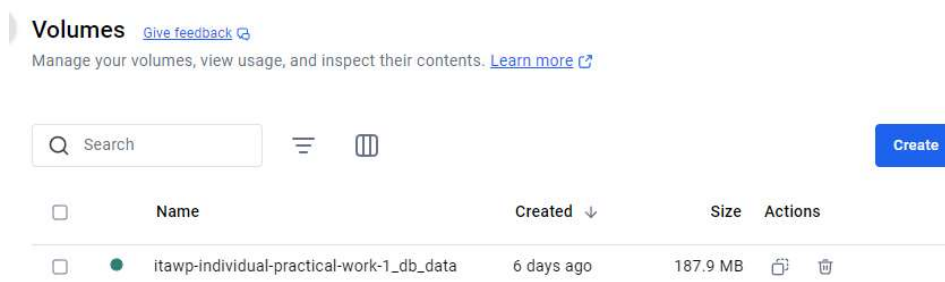


Рисунок 13 – Пример *volume*-а с базой данных

а продуманная структура *docker-compose* файла позволяет легко масштабировать приложение и добавлять новые сервисы в будущем.

## 2.7 Ответ на контрольный вопрос: Чем отличается *require* от *include*

Основное отличие, которое непосредственно влияет на надежность приложения, заключается в поведении при отсутствии подключаемого файла. Директива *require* при невозможности найти указанный файл немедленно прекращает выполнение скрипта с фатальной ошибкой, что гарантирует целостность критически важных компонентов системы. В отличие от этого, *include* лишь генерирует предупреждение и позволяет скрипту продолжить выполнение, что может быть полезно для опциональных компонентов.

Это различие в обработке ошибок определяет сферы применения каждой директивы в архитектуре веб-приложения. Для файлов, без которых функционирование системы невозможно, таких как конфигурация базы данных или основные библиотеки, необходимо использовать *require*. В разработанном веб-сервисе подключение параметров базы данных осуществлялось исключительно через *require*, поскольку отсутствие этих настроек делало бессмысленной дальнейшую работу приложения. Такой подход обеспечивает немедленное выявление проблем на ранних этапах выполнения скрипта.

Для дополнительных модулей и компонентов, которые не являются жизненно важными для работы системы, целесообразно применять *include*. Это позволяет создавать более гибкую и отказоустойчивую архитектуру, где отсутствие некоторых функций не приводит к полному отказу всего приложения. В процессе разработки я оценил важность этого различия для построения стабильных веб-приложений, особенно в производственной среде.

Особое значение имеют варианты этих директив с суффиксом *\_once*, которые предотвращают многократное подключение одного и того же файла. Это особенно актуально в крупных проектах со сложной структурой, где существует риск повторного объявления функций или констант. Использование *require\_once* для критических компонентов и *include\_once* для дополнительных модулей является хорошей практикой, обеспечивающей стабильность приложения.

Практический опыт показал, что выбор между *require* и *include* — это не просто техническая особенность языка, а важный архитектурный инструмент. Правильное применение этих директив позволяет строить более надежные системы с четкой структурой зависимостей.

## ЗАКЛЮЧЕНИЕ

В результате выполнения индивидуальной практической работы был успешно разработан и внедрен полнофункциональный веб-сервис для подачи объявлений, который полностью соответствует поставленным целям и задачам. Все запланированные этапы разработки были последовательно реализованы - от проектирования базы данных и создания пользовательского интерфейса до организации безопасной обработки данных и контейнеризации приложения. Практическая значимость работы подтверждается тем, что полученное решение представляет собой законченный продукт, готовый к использованию в образовательных и коммерческих целях, демонстрирующий применение современных веб-технологий в реальных условиях.

Анализ полученных результатов показывает, что разработанный веб-сервис успешно справляется с возложенными на него функциями: обеспечивает удобный интерфейс для ввода данных, выполняет комплексную валидацию информации, безопасно сохраняет объявления в базе данных и предоставляет пользователям понятную обратную связь. Особенно важно отметить достигнутый уровень безопасности приложения, который включает защиту от *SQL*-инъекций, *XSS*-атак и других распространенных уязвимостей, что соответствует современным стандартам веб-разработки. Использование *Docker*-контейнеризации доказало свою эффективность, обеспечив простоту развертывания и воспроизводимости среды выполнения.

Оценка функциональности разработанного веб-сервиса позволяет утверждать, что все основные требования технического задания выполнены в полном объеме. Сервис демонстрирует стабильную работу, корректно обрабатывает различные сценарии ввода данных и обеспечивает надежное хранение информации. Среди преимуществ реализации можно выделить модульную архитектуру, которая облегчает поддержку и расширение кода, использование подготовленных выражений для работы с базой данных, обеспечивающих высокий уровень безопасности, а также продуманный пользовательский интерфейс с адаптивным дизайном. Однако в процессе работы были выявлены и некоторые ограничения, такие как отсутствие механизма редактирования уже размещенных объявлений и системы модерации контента, что может быть критично для промышленной эксплуатации.

## ПРИЛОЖЕНИЕ А

(обязательное)

### Программный код приложения

С полным кодом веб-приложения можно ознакомиться по ссылке:  
***<https://github.com/POOSTOTEL/ITaWP-Individual-practical-work-1>***

#### **Файл *config.php***

```
<?php
$db_host = getenv('DB_HOST') ?: 'localhost';
$db_user = getenv('DB_USER') ?: 'user';
$db_pass = getenv('DB_PASS') ?: 'password';
$db_name = getenv('DB_NAME') ?: 'advertisement_db';

$mysqli = new mysqli($db_host, $db_user, $db_pass, $db_name);

if ($mysqli->connect_error) {
    die("Ошибка подключения к базе данных: " . $mysqli->connect_error .
        ". Проверьте настройки подключения в docker-compose.yml");
}

$mysqli->set_charset("utf8");

function log_error($message) {
    error_log("Advertisement Form Error: " . $message);
}
?>
```

#### **Файл *process.php***

```
<?php
require_once 'config.php';

$errors = [];

function clean_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```

$ad_title = clean_input($_POST['ad_title'] ?? "");
$ad_category = clean_input($_POST['ad_category'] ?? "");
$price = clean_input($_POST['price'] ?? "");
$contact_email = clean_input($_POST['contact_email'] ?? "");
$ad_text = clean_input($_POST['ad_text'] ?? "");

if (empty($ad_title)) {
    $errors[] = "Заголовок объявления обязателен для заполнения";
} elseif (strlen($ad_title) > 255) {
    $errors[] = "Заголовок не должен превышать 255 символов";
}

if (empty($ad_category)) {
    $errors[] = "Необходимо выбрать категорию";
}

if (empty($price)) {
    $errors[] = "Цена обязательна для заполнения";
} elseif (!is_numeric($price) || $price < 0) {
    $errors[] = "Цена должна быть положительным числом";
}

if (empty($contact_email)) {
    $errors[] = "Email обязателен для заполнения";
} elseif (!filter_var($contact_email, FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Некорректный формат email";
}

if (empty($ad_text)) {
    $errors[] = "Текст объявления обязателен для заполнения";
}

if (empty($errors)) {
    $stmt = $mysqli->prepare("INSERT INTO advertisements (ad_title, ad_category, price, contact_email, ad_text) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("ssdss", $ad_title, $ad_category, $price, $contact_email, $ad_text);

    if ($stmt->execute()) {
        $success_message = "Объявление успешно размещено!";
    } else {
        $errors[] = "Ошибка при сохранении в базу данных: " . $stmt->error;
    }

    $stmt->close();
}

```



```

    $mysqli->close();
}
?>

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Результат обработки</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
            padding: 20px;
        }
        .result-container {
            max-width: 600px;
            margin: 0 auto;
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
    </style>
</head>
<body>
    <div class="result-container">
        <h1 class="text-center mb-4">Результат обработки</h1>

        <?php if (!empty($errors)): ?>
            <div class="alert alert-danger">
                <h4>Обнаружены ошибки:</h4>
                <ul>
                    <?php foreach ($errors as $error): ?>
                        <li><?php echo $error; ?></li>
                    <?php endforeach; ?>
                </ul>
            </div>
            <div class="text-center">
                <a href="form.html" class="btn btn-warning">Вернуться к форме</a>
            </div>
        <?php elseif (isset($success_message)): ?>
            <div class="alert alert-success text-center">

```

```

        <h4><?php echo $success_message; ?></h4>
        <p>Ваше объявление было успешно сохранено в базе данных.</p>
    </div>
    <div class="text-center">
        <a href="form.html" class="btn btn-primary">Разместить еще одно
        объявление</a>
    </div>

    <div class="mt-4 p-3 border rounded">
        <h5>Сохраненные данные:</h5>
        <p><strong>Заголовок:</strong> <?php echo htmlspecialchars($ad_title); ?></p>
        <p><strong>Категория:</strong> <?php echo htmlspecialchars($ad_category);
        ?></p>
        <p><strong>Цена:</strong> <?php echo number_format($price, 2, '.', ' '); ?>
        руб.</p>
        <p><strong>Email:</strong> <?php echo htmlspecialchars($contact_email);
        ?></p>
        <p><strong>Текст объявления:</strong><br><?php echo
        nl2br(htmlspecialchars($ad_text)); ?></p>
    </div>
    <?php endif; ?>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

### Файл form.html

```

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Подача объявления</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
            padding: 20px;
        }
        .form-container {
            max-width: 600px;

```

```

margin: 0 auto;
background: white;
padding: 30px;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
.form-title {
text-align: center;
margin-bottom: 30px;
color: #333;
}
.required::after {
content: " *";
color: red;
}
</style>
</head>
<body>
<div class="form-container">
<h1 class="form-title">Подача объявления</h1>

<form action="process.php" method="POST">
<div class="mb-3">
<label for="ad_title" class="form-label required">Заголовок объявления</label>
<input type="text" class="form-control" id="ad_title" name="ad_title"
required maxlength="255" placeholder="Введите заголовок объявления">
<div class="form-text">Максимум 255 символов</div>
</div>

<div class="mb-3">
<label for="ad_category" class="form-label required">Категория</label>
<select class="form-select" id="ad_category" name="ad_category" required>
<option value="">Выберите категорию</option>
<option value="Недвижимость">Недвижимость</option>
<option value="Транспорт">Транспорт</option>
<option value="Работа">Работа</option>
<option value="Услуги">Услуги</option>
<option value="Электроника">Электроника</option>
<option value="Для дома">Для дома</option>
<option value="Хобби">Хобби</option>
<option value="Другое">Другое</option>
</select>
</div>

<div class="mb-3">
<label for="price" class="form-label required">Цена (руб.)</label>

```

```

        <input type="number" class="form-control" id="price" name="price"
            required min="0" step="0.01" placeholder="0.00">
    </div>

    <div class="mb-3">
        <label for="contact_email" class="form-label required">Email для связи</label>
        <input type="email" class="form-control" id="contact_email" name="contact_email"
            required placeholder="example@mail.ru">
    </div>

    <div class="mb-3">
        <label for="ad_text" class="form-label required">Текст объявления</label>
        <textarea class="form-control" id="ad_text" name="ad_text"
            required rows="5" placeholder="Подробно опишите то, что вы продаете
или предлагаете"></textarea>
    </div>

    <button type="submit" class="btn btn-primary w-100">Разместить
объявление</button>
</form>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

### **Файл *init.sql***

```

CREATE DATABASE IF NOT EXISTS advertisement_db;
USE advertisement_db;

```

```

CREATE TABLE IF NOT EXISTS advertisements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    ad_title VARCHAR(255) NOT NULL,
    ad_category VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    contact_email VARCHAR(100) NOT NULL,
    ad_text TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

```

INSERT INTO advertisements (ad_title, ad_category, price, contact_email, ad_text) VALUES
('Продам ноутбук HP', 'Электроника', 25000.00, 'seller@example.com', 'Отличное
состояние, б/у 1 год.').

```

*('Сниму квартиру в центре', 'Недвижимость', 30000.00, 'rent@example.com', 'Ищу 2-х комнатную квартиру в центре города.');*

### **Файл *docker-compose.yml***

```
# docker-compose.yml
version: '3.8'

services:
  web:
    build: .
    ports:
      - "8080:80"
    volumes:
      - ../var/www/html
    depends_on:
      - db
    environment:
      - DB_HOST=db
      - DB_USER=user
      - DB_PASS=password
      - DB_NAME=advertisement_db
    restart: unless-stopped

  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: advertisement_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    ports:
      - "3308:3306"
    volumes:
      - db_data:/var/lib/mysql
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    restart: unless-stopped

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      PMA_HOST: db
      PMA_PORT: 3308
      MYSQL_ROOT_PASSWORD: rootpassword
    ports:
```

- "8081:80"  
depends\_on:  
- db  
restart: unless-stopped

volumes:  
db\_data:

### **Файл Dockerfile**

*FROM php:8.1-apache*

*RUN apt-get update && apt-get install -y \  
libpng-dev \  
libjpeg-dev \  
libpq-dev \  
libzip-dev \  
zip \  
unzip \  
git \  
curl \  
&& docker-php-ext-configure gd --with-jpeg \  
&& docker-php-ext-install -j\$(nproc) gd \  
&& docker-php-ext-install pdo pdo\_mysql mysqli \  
&& docker-php-ext-install zip*

*RUN a2enmod rewrite*

*COPY . /var/www/html/*

*RUN chown -R www-data:www-data /var/www/html \  
&& chmod -R 755 /var/www/html*

*EXPOSE 80*

*CMD ["apache2-foreground"]*

### **Файл start.bat**

*@echo off  
chcp 65001 > nul  
echo =====  
echo Запуск веб-сервиса для подачи объявлений  
echo =====*

*echo.*

*docker --version >nul 2>&1*

*if errorlevel 1 (*

*echo ОШИБКА: Docker не установлен или не запущен!*

*echo Пожалуйста, установите Docker Desktop и запустите его.*

*echo Скачать можно с: <https://www.docker.com/products/docker-desktop>*

*pause*

*exit /b 1*

*)*

*docker compose version >nul 2>&1*

*if errorlevel 1 (*

*echo ОШИБКА: Docker Compose не доступен!*

*echo Убедитесь, что Docker Desktop запущен.*

*pause*

*exit /b 1*

*)*

*echo Проверка зависимостей... ОК*

*echo.*

*echo Запуск контейнеров (это может занять несколько минут при первом запуске)...*

*docker compose up -d*

*if errorlevel 1 (*

*echo ОШИБКА: Не удалось запустить контейнеры!*

*echo Проверьте, что порты 8080, 8081 и 3308 свободны.*

*pause*

*exit /b 1*

*)*

*echo.*

*echo =====*

*echo Приложение успешно запущено!*

*echo =====*

*echo.*

*echo Доступные сервисы:*

*echo - Веб-приложение: <http://localhost:8080/form.html>*

*echo - PhpMyAdmin: <http://localhost:8081>*

*echo - База данных: [localhost:3308](http://localhost:3308)*

*echo.*

*echo Данные для входа в PhpMyAdmin:*

*echo - Сервер: db*

*echo - Пользователь: root*

*echo - Пароль: rootpassword*

```
echo.  
echo Для остановки приложения запустите stop.bat  
echo.  
pause
```

### **Файл stop.bat**

```
@echo off  
chcp 65001 > nul  
echo Остановка контейнеров...  
docker compose down
```

```
echo.  
echo Все контейнеры остановлены и удалены.  
pause
```