**Практическая работа №2.**

**Теория**

1. **SSR**

В контексте веб-разработки существуют тонкие и толстые клиенты.

Тонкий клиент — это подход, при котором большая часть логики приложения выполняется на сервере, а клиентский браузер выполняет только стандартные задачи, такие, как отображение контента, манипуляции с DOM и взаимодействие с пользователем. При таком подходе веб-сервер является "мозгом" приложения, что означает, что все вычисления и обработка запросов происходит на серверной стороне. Примерами тонких клиентов могут служить приложения, основанные на PHP, Node.js или Ruby on Rails.

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.

Толстый клиент - это подход, при котором большая часть логики выполнена на клиентской стороне, а сервер выступает преимущественно в роли поставщика данных. При этом клиент загружает не только HTML, CSS и JavaScript, но и дополнительные библиотеки и фреймворки, которые выполняют все основные функции приложения на стороне браузера пользователя. Примерами толстых клиентов могут служить SPA (Single-page applications), созданные с использованием фреймворков React, Vue.js или Angular. При использовании толстого клиента сервер выполняет роль поставщика данных, что позволяет клиенту загружать и обрабатывать данные самостоятельно, без постоянной связи с сервером. Это создает более интерактивный и быстрый интерфейс, улучшает возможности отладки и повышает надежность работы приложения в целом.

В рамках нашего курса будет использоваться тонкий клиент, достигаться это будет разными способами, например в данной практике мы будем использовать 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.

SSR может быть использован различными языками программирования, включая PHP. Для SSR в PHP могут использоваться фреймворки, такие, как Laravel или Symfony, а также библиотеки для рендеринга пользовательских интерфейсов.

Когда пользователь запрашивает страницу, HTTP-запрос передается на сервер, который отвечает содержимым страницы, передавая исходный код HTML, CSS и JavaScript на клиентскую сторону браузера. Далее, браузер рендерит этот HTML-код на стороне клиента и отображает страницу.

Однако, при использовании SSR сервер оказывает более активную роль в этом процессе, из-за чего веб-сервер более нагружен, чем при иных подходах, например, как при REST API. Пользовательское взаимодействие остается на стороне клиента и выполняется JavaScript в браузере. В итоге SSR может улучшить производительность и SEO-оптимизацию веб-страниц, так как страницы более быстро загружаются и более доступны для поисковых роботов. С использованием SSR, также, удается предоставить лучший пользовательский опыт веб-сайта в целом.

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. **CRUD**

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.

Все операции CRUD могут быть реализованы с использованием языка SQL, который предоставляет широкий набор команд для работы с базами данных. В веб-разработке CRUD-операции могут быть реализованы с помощью языка программирования (например, PHP), фреймворков или CMS, которые предоставляют специальные функции и библиотеки для работы с базами данных.

CRUD-операции являются основными элементами веб-разработки. Разработчики используют их для создания интерфейса для управления данными, который позволяет пользователям выполнять операции над информацией. Например, панель управления администратора, где он может создавать, изменять, просматривать и удалять данные в базе данных.

Важно понимать, что CRUD-операции могут оказаться уязвимостью приложения. Вредоносные пользователи могут использовать запросы, чтобы получить несанкционированный доступ к данным или изменить их. Поэтому безопасность приложения является важным фактором при разработке систем, которые работают с базами данных.

1. **MySql**

Вспомните из прошлой практики теорию про конфигурацию, мы ведь не добавили базу данных в нашу конфигурацию. В данной практике необходимо будет создать новый контейнер с базой данных MySql.

PHP обладает различными функциями для работы с MySQL, которые позволяют легко создавать соединения с базой данных, выполнение SQL-запросов и обработку результатов запросов.

* Используйте функцию `mysqli\_connect()` для подключения к базе данных MySQL.

|  |
| --- |
| $conn = mysqli\_connect('localhost', 'username', 'password', 'database'); |

Здесь параметры — это имя хоста, имя пользователя, пароль и имя базы данных.

* Используйте функцию `mysqli\_query()` для выполнения SQL-запроса в базе данных.

|  |
| --- |
| $result = mysqli\_query($conn, "SELECT \* FROM users"); |

Эта функция вернет результат запроса, который может быть обработан далее в коде.

* Используйте функции `mysqli\_fetch\_array()` или `mysqli\_fetch\_assoc()` для обработки результатов запроса.

|  |
| --- |
| while($row = mysqli\_fetch\_array($result)) {  echo $row['name'] . '<br>';  } |

Эта функция выполняет обход всех строк в результате запроса и выдает каждую из них в виде ассоциативного массива или обычного массива.

* Не забудьте закрыть соединение с базой данных после того, как закончите работу с ней.

|  |
| --- |
| mysqli\_close($conn); |

**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.
3. Требуется создать 4 страницы php, где на каждой странице будет реализована одна из CRUD операций с выбранной вами сущностью.
4. **Полезные ссылки**
5. Хабр статья со шпаргалками по MySql: https://habr.com/ru/articles/564390/
6. Официальная документация php по работе с Mysql: https://www.php.net/manual/ru/book.mysql.php
7. 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
8. Статья про использование MySql в php: https://htmlacademy.ru/blog/php/mysql
9. Официальная документация докера: <https://docs.docker.com/>
10. Туториал по докеру: <https://badcode.ru/docker-tutorial-dlia-novichkov-rassmatrivaiem-docker-tak-iesli-by-on-byl-ighrovoi-pristavkoi/>
11. Туториал по докеру с Хабра: <https://habr.com/ru/post/310460/>
12. Шпаргалка с командами Docker: <https://habr.com/ru/company/flant/blog/336654/>
13. Docker hub: <https://hub.docker.com/>

**Вопросы**

* В чём преимущество языка PHP для backend'а?
* Как достигается тонкий клиент при использовании PHP?
* Что такое SSR?