**Практическая работа №3.**

**Теория**

**REST API**

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. Учитывая, что web-приложения обычно состоят из клиентской и серверной частей, REST API находится в backend серверной части. REST API определяет набор ограничений и ограничения на взаимодействие между клиентом и сервером. Клиенты отправляют запросы с помощью определенных HTTP-методов (POST, GET, PUT, PATCH, DELETE), чтобы получить, обновить, создать или удалить определенную информацию или ресурс в приложении.

Backend обрабатывает запросы и отправляет обратно соответствующие ответы в соответствии с принятыми стандартами HTTP. REST API обычно используются вместе с форматом данных JSON или XML для передачи информации между клиентом и сервером. Они позволяют разработчикам легко интегрировать и расширять приложения, создавать новые функции и предоставлять доступ к данным другим приложениям и сервисам.

REST API также придерживается принципов, называемых "ограничениями REST", такими как унифицированный интерфейс, отсутствие состояния, кэширование, многослойная архитектура и код по запросу. Эти принципы обеспечивают удобство использования и универсальность приложения, простоту его реализации и масштабируемость. REST API является неотъемлемой частью backend разработки и используется для взаимодействия между различными приложениями, а также для реализации различных сервисов и функций в приложении.

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.

GET-запросы

Метод запроса GET используется для получения данных от сервера. С помощью GET-запроса клиент может передавать параметры в URL-адресе, и сервер будет использовать эти параметры для отображения информации на странице. GET-запросы не могут изменять данные на сервере, они только могут получать данные.

POST-запросы

Метод запроса POST используется для отправки данных на сервер. POST-запросы часто используются при отправке форм на сервер. В отличие от GET-запросов, POST-запросы могут изменять данные на сервере.

PUT-запросы

Метод запроса PUT используется для обновления данных на сервере. PUT-запрос должен содержать обновленные данные, которые нужно заменить в базе данных на сервере. PUT-запросы часто используются при обновлении отдельных элементов на странице.

DELETE-запросы

Метод запроса DELETE используется для удаления данных на сервере. DELETE-запрос должен содержать идентификатор данных, которые нужно удалить. Он удаляет целые записи из базы данных на сервере.

Все запросы состоят из двух частей: заголовка и тела запроса. Заголовок содержит метаданные, такие как тип запроса, тип документа и другую дополнительную информацию. Тело запроса содержит данные, которые отправляются на сервер.

Для того чтобы реализовать нормальный rest api, потребуется описать базовые сущности.

В backend разработке сущность (Entity) — это объект, который представляет собой некоторый вид данных, которые могут быть сохранены и извлечены из базы данных. Сущность в backend содержит поля и методы, которые позволяют работать с данными в базе данных.

Примером сущности в backend может быть таблица "Пользователь" в базе данных. Сущность "Пользователь" содержит поля, такие как "id", "имя", "адрес электронной почты" и "пароль". Кроме того, эта сущность может иметь методы для добавления, удаления, обновления и чтения пользователей из базы данных.

Сущности в backend должны быть полностью абстрагированы от любой логики приложения и слоя представления. Это означает, что изменения в базе данных или логике приложения не должны приводить к изменению сущности. Это позволяет уменьшить связанность между различными частями приложения и повышает его модульность.

Пример сущности:

Допустим, мы хотим создать сущность "Пользователь" для работы с базой данных MySQL. Главной задачей будет правильно определить поля этой сущности - они должны соответствовать полям таблицы "users" в базе данных.

|  |
| --- |
| class User {  private $id;  private $name;  private $email;  private $password;  public function \_\_construct($name, $email, $password) {  $this->name = $name;  $this->email = $email;  $this->password = $password;  }  public function setId($id) {  $this->id = $id;  }  public function getId() {  return $this->id;  }  public function getName() {  return $this->name;  }  public function getEmail() {  return $this->email;  }  public function getPassword() {  return $this->password;  }  } |

Листинг 1. Пример сущности

Здесь мы определяем класс "User" с четырьмя приватными полями: "id", "name", "email" и "password". Метод "\_\_construct" используется для инициализации свойств объекта при создании, методы "getId", "getName", "getEmail" и "getPassword" используются для получения значений этих полей.

Это пример простой сущности, которая может быть расширена, например, добавлением методов сохранения и удаления записей из базы данных, методов валидации данных и других методов необходимых для работы с пользователями.

**Задание**

Вам необходимо создать конфигурацию docker-compose для вашего веб-сервера или взять из предыдущей практики.

Docker контейнер веб серверва 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 любые две связанные сущности.

2. Требуется реализовать REST API для двух сущностей, где на запросы ответы должны быть в формате JSON.

Пример запроса: `[GET] localhost:8080/api/orders.php?id=3`

3. Провести ручное тестирование с помощью POSTMAN, CURL или Insomnia.

**Полезные ссылки**

1. Пример создания rest api на php: https://cms.by/blog/php/primer-prostogo-sozdaniya-rest-api-na-php/
2. Примеры rest api приложений: https://github.com/topics/php-rest-api
3. Гайд как пользоваться postman: https://testengineer.ru/gajd-po-testirovaniyu-v-postman/
4. Статья на хабре про принципы построения REST JSON API: https://habr.com/ru/articles/447322/
5. Шпаргалка по безопасности rest api: https://habr.com/ru/companies/acribia/articles/453384/
6. Гайд по использованию curl: https://megahub.me/cheat-sheets/curl
7. Лучшие практики rest api: https://habr.com/ru/companies/piter/articles/511382/

**Вопросы**

* Что такое сущность?
* Какие методы существуют в протоколе http?
* Что такое контроллер?