|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №4**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы:**

**Студент группы** ИКБО-20-19 Анваржонов Ж Т

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Волков М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «20» ноября 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Цель работы**

Предполагается реализация интерфейса прикладного программирования для доступа к некоторым данным по варианту. Предполагается реализация серверной части обработки запросов и тестирование данного интерфейса с использованием программы Postman. Для реализации данного сервиса предлагается использовать серверную конфигурацию, модернизированную в течение первых трех практических работ. Важной частью данной практической работы является сохранение функциональности реализованной в практической работе №3. То есть интерфейс предлагается создать уже в существующем веб-приложении. Также предполагается использование темы практической работы №3 для продолжения модернизирования собственной системы. Изменение темы согласовывается отдельно с преподавателем. Хранение данных предполагается уже в существующей базе данных. Технические требования к реализации интерфейса:

1. Доступ как минимум к 2 независимым сущностям

2. Реализация как минимум операций группы CRUD(создание, чтение, обновление, удаление). Приветствуется реализация дополнительной функциональности

3. Тестирование всех функциональных возможностей созданного интерфейса с использованием программы Postman

**Выполнение задания**

Будем использовать две независимые сущности: Toy, где хранится название игрушки и её цена и Store, где хранится название магазина и её адрес.

Покажем реализацию операций группы CRUD:

**Interesting\_facts.php:**

<?php   
 include 'convert\_result\_to\_json.php';  
 include 'get\_body.php';  
 include 'get\_params.php';  
  
 header("Access-Control-Allow-Origin: \*");  
 header("Content-Type: application/json; charset=UTF-8");  
  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
  
 $body = getBody();  
 $params = getParams();  
  
 switch ($\_SERVER['REQUEST\_METHOD']) {  
 case 'GET':  
 $result = $mysqli->query("SELECT \* FROM interesting\_facts");  
   
 echo convertRestultToJSON($result);  
 break;  
 case 'POST':  
 $stmt = $mysqli->prepare("INSERT INTO interesting\_facts (title, text, url\_video) VALUES (?, ?, ?)");  
 $res = $stmt->bind\_param('sss', $body['title'], $body['text'], $body['url\_video']);  
 $stmt->execute();  
  
 echo 1;  
 break;  
 case 'PUT':  
 $stmt = $mysqli->prepare("UPDATE interesting\_facts SET title=?, text=?, url\_video=? WHERE ID=?");  
 $res = $stmt->bind\_param('sssi', $body['title'], $body['text'], $body['url\_video'], $body['ID']);  
 $stmt->execute();  
  
 break;  
 case 'DELETE':  
 $stmt = $mysqli->prepare("DELETE FROM `interesting\_facts` WHERE `ID`=?");  
 $res = $stmt->bind\_param('i', $params['ID']);  
 $stmt->execute();  
  
 echo 1;  
 break;  
 default:  
 header("HTTP/1.1 404 Not Found");  
 break;  
 }  
?>

**users.php - для коннекта к базе данных**

<?php   
 include 'convert\_result\_to\_json.php';  
 include 'get\_body.php';  
 include 'get\_params.php';  
  
 header("Access-Control-Allow-Origin: \*");  
 header("Content-Type: application/json; charset=UTF-8");  
  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
  
 $body = getBody();  
 $params = getParams();  
  
 switch ($\_SERVER['REQUEST\_METHOD']) {  
 case 'GET':  
 $result = $mysqli->query("SELECT \* FROM users");  
   
 echo convertRestultToJSON($result);  
 break;  
 case 'POST':  
 $stmt = $mysqli->prepare("INSERT INTO users (name, password) VALUES (?, ?)");  
 $res = $stmt->bind\_param('ss', $body['name'], $body['password']);  
 $stmt->execute();  
  
 echo 1;  
 break;  
 case 'PUT':  
 $stmt = $mysqli->prepare("UPDATE users SET name=?, password=? WHERE ID=?");  
 $res = $stmt->bind\_param('ssi', $body['name'], $body['password'], $body['ID']);  
 $stmt->execute();  
  
 break;  
 case 'DELETE':  
 $stmt = $mysqli->prepare("DELETE FROM `users` WHERE `ID`=?");  
 $res = $stmt->bind\_param('i', $params['ID']);  
 $stmt->execute();  
  
 echo 1;  
 break;  
 default:  
 header("HTTP/1.1 404 Not Found");  
 break;  
 }  
?>

**Read.php – чтение всех строк из базы данных.**

<?php  
include\_once 'database.php';  
include\_once 'toy.php';  
$database = new Database();  
  
$db = $database->getConnection();  
$items = new Employee($db);  
$records = $items->getEmployees();  
$itemCount = $records->num\_rows;  
echo json\_encode($itemCount);  
if($itemCount > 0){  
 $employeeArr = array();  
 $employeeArr["body"] = array();  
 $employeeArr["itemCount"] = $itemCount;  
 while ($row = $records->fetch\_assoc())  
 {  
 array\_push($employeeArr["body"], $row);  
 }  
 echo json\_encode($employeeArr);  
}  
else{  
 http\_response\_code(404);  
 echo json\_encode(  
 array("message" => "No record found.")  
 );  
}  
?>

**Convert\_result\_to\_json.php – для чтения информации о одной сущности.**

<?php   
 function convertRestultToJSON($result)  
 {  
 $facts = [];  
 while($row = $result->fetch\_array(*MYSQLI\_ASSOC*)) {  
 $facts[] = $row;  
 }  
 return json\_encode($facts);  
 }  
?>

**Get\_body.php – Для обновления информации о сущности**

<?php   
 function getBody()  
 {  
 return json\_decode(file\_get\_contents('php://input'), true);  
 }  
?>

**getParams.php – класс, отвечающий за все операции над сущностью.**

<?php  
 function getParams()  
 {  
 $query = $\_SERVER['QUERY\_STRING'];  
  
 $delete\_args = [];  
  
 if ( !empty( $query ) )  
 {  
 foreach( explode('&', $query ) as $param )  
 {  
 list($k, $v) = explode('=', $param);  
 $k = urldecode($k);  
 $v = urldecode($v);  
 $delete\_args[$k] = $v ;  
 }  
 }  
  
 return $delete\_args;

**Delete\_interesting\_fact.php:**

<?php  
 include 'login.php';  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
 $stmt = $mysqli->prepare("DELETE FROM `interesting\_facts` WHERE `ID`=?");  
 $res = $stmt->bind\_param('i', $\_GET['ID']);  
 $stmt->execute();  
   
 header('Location: /admin/index.php');

**Delete\_users.php:**

<?php  
 include 'login.php';  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
 $stmt = $mysqli->prepare("DELETE FROM `users` WHERE `ID`=?");  
 $res = $stmt->bind\_param('i', $\_GET['ID']);  
 $stmt->execute();  
   
 header('Location: /admin/index.php');

**Insert\_interesting\_fact.php**

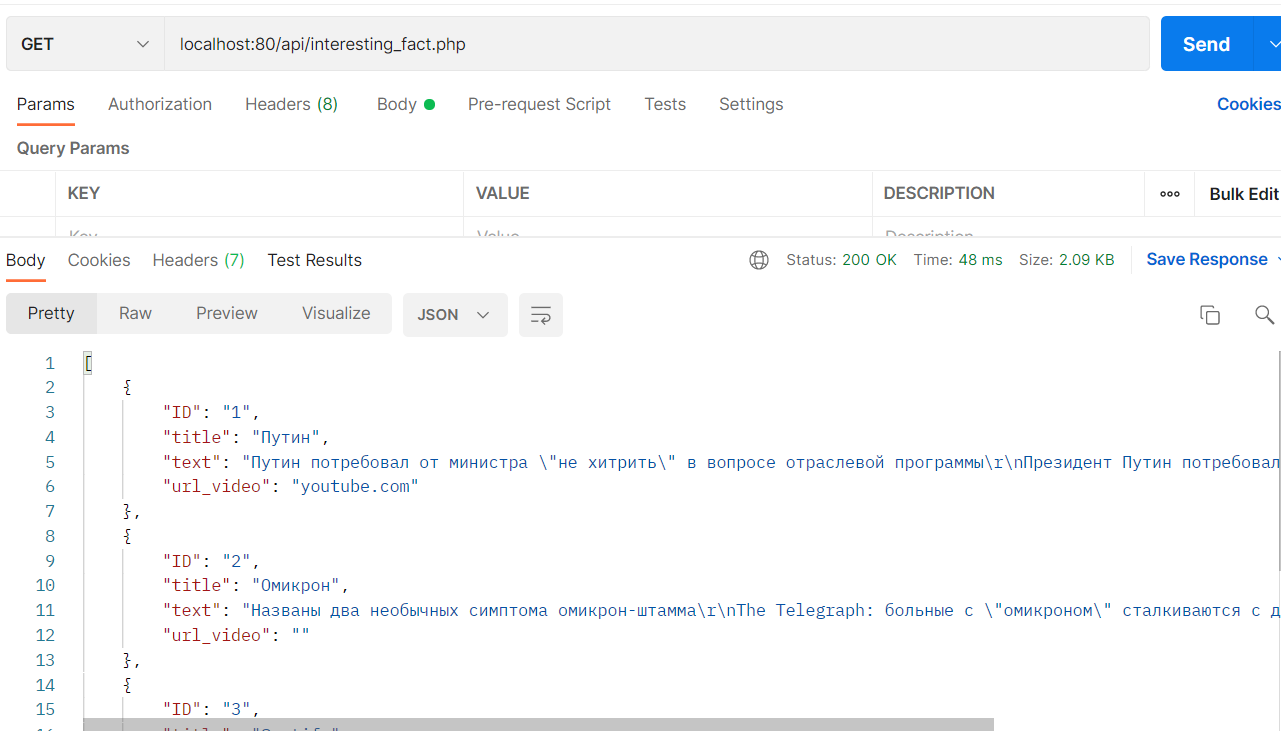
<?php  
 include 'login.php';  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
 $stmt = $mysqli->prepare("DELETE FROM `users` WHERE `ID`=?");  
 $res = $stmt->bind\_param('i', $\_GET['ID']);  
 $stmt->execute();  
   
 header('Location: /admin/index.php');

**Insert\_users.php**

<?php  
 include 'login.php';  
   
 $mysqli = new mysqli("db", "user", "password", "appDB");  
 $stmt = $mysqli->prepare("INSERT INTO users (name, password) VALUES (?, ?)");  
 $res = $stmt->bind\_param('ss', $\_POST['name'], $\_POST['password']);  
 $stmt->execute();  
  
 header('Location: /admin/index.php');  
?>

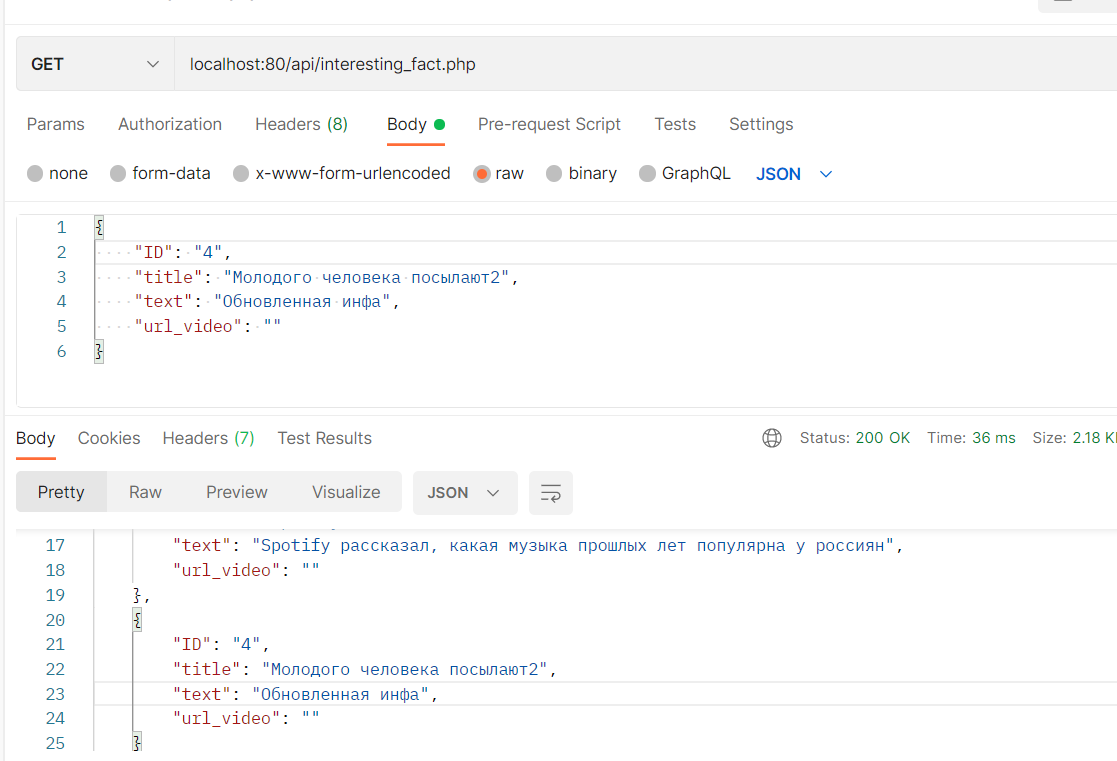
**Тестирование сделанных операций**

Сделаем тестирование операции get для получения всех объектов

****

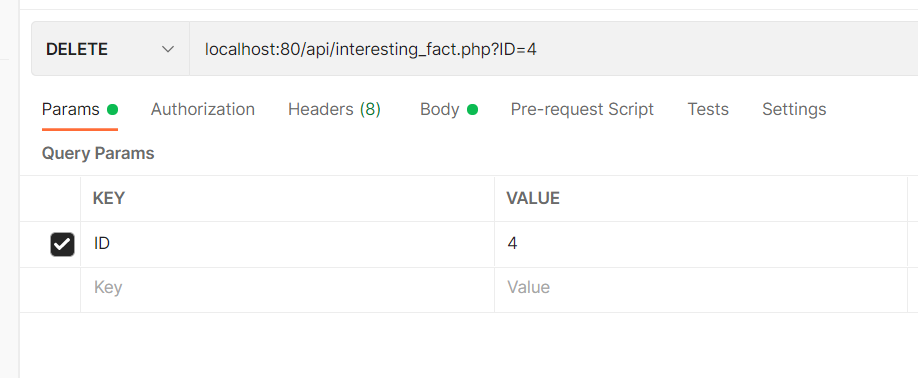
**Рисунок 1 – тестирование**

**Тестирование операции обновления сущности**

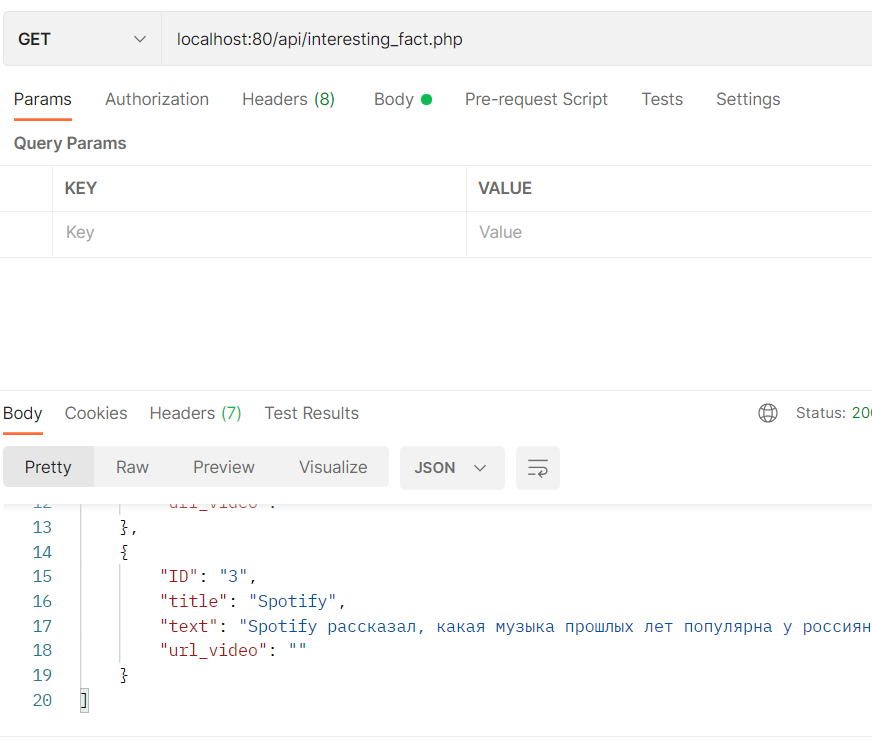
****

**Рисунок 2 – тестирование обновления сущности**

**Покажем операцию удаления:**

****

**Покажем удачное удаление:**

****

**Рисунок 3 – тестирование удаления .**

Вывод

В ходе выполнения практической работы был реализован интерфейс прикладного программирования для доступа к некоторым данным c помощью Rest API, также протестированы реализованные операции с помощью Postman.

**Ответы на вопросы к практической работе**

1. Что такое HTTP-запрос?

HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

2. Опишите существующие HTTP-запросы.

[**GET**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET)

Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

[**HEAD**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

[**POST**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST)

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

[**PUT**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT)

PUT заменяет все текущие представления ресурса данными запроса.

[**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

DELETE удаляет указанный ресурс.

[**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)

CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.

[**OPTIONS**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS)

OPTIONS используется для описания параметров соединения с ресурсом.

[**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)

TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.

[**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH)

PATCH используется для частичного изменения ресурса.

3. Опишите обработку запроса на PHP. Что нужно использовать, как

вычленить параметры запроса?

В PHP есть все возможности для взаимодействия с протоколом HTTP:

* Получение тела запроса;
* Получение заголовков запроса;
* Добавление/изменение заголовков ответа;
* Управление телом ответа.

4. Опишите создание HTML-форм на PHP.

Одно из главнейших достоинств PHP - то, как он работает с формами HTML. Здесь основным является то, что каждый элемент формы автоматически становится доступным вашим программам на PHP

<form action="action.php" method="post">

<p>Ваше имя: <input type="text" name="name" /></p>

<p>Ваш возраст: <input type="text" name="age" /></p>

<p><input type="submit" /></p>

</form>

5. Что такое API?

*API (Application programming interface)* — это контракт, который предоставляет программа. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».  
  
Если переводить на русский, это было бы слово «договор». Договор между двумя сторонами, как договор на покупку машины:

* мои обязанности — внести такую то сумму,
* обязанность продавца — дать машину.

6. Опишите API как средство интеграции приложений.

Если программу (модуль, библиотеку) рассматривать как [чёрный ящик](https://ru.wikipedia.org/wiki/%D0%A7%D1%91%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D1%89%D0%B8%D0%BA), то API — это набор «ручек», которые доступны пользователю данного ящика и которые он может вертеть и дёргать.

Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

По такому принципу построены [протоколы передачи данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) по [Интернету](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Стандартный стек протоколов ([сетевая модель OSI](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI)) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам [HTTP](https://ru.wikipedia.org/wiki/HTTP) и [IMAP](https://ru.wikipedia.org/wiki/IMAP)). Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню.

Понятие протокола близко по смыслу к понятию API. И то, и другое является абстракцией функциональности, только в первом случае речь идёт о передаче данных, а во втором — о взаимодействии приложений.

API библиотеки функций и классов включает в себя описание *сигнатур* и *семантики функций*.

7. Что такое Web API?

Веб-API - это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской стороной веб-приложения (включая любые используемые веб-фреймворки), и поэтому обычно не включает детали реализации веб-сервера или браузера, такие как SAPI или API, если они не доступны для общего доступа через удаленное веб-приложение.

8. Приведите пример API.

Одним из примеров API в интернет-рекламе является приложение, которое использует «Яндекс. Директ».

9. Что такое REST?

REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST - это набор правил о том, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать

10.Как организована передача данных в архитектуре REST?

По протоколу HTTP

11.Как организована работа REST?

REST API функционируют без сохранения состояния, т. е. каждый запрос должен содержать всю информацию, необходимую для его обработки. Другими словами, для REST API не требуется установление постоянных сеансов с сервером.

12.Что такое SOAP?

SOAP — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур. Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

13.Чем SOAP отличается от REST?

REST и SOAP на самом деле не сопоставимы. REST — это архитектурный стиль. SOAP — это формат обмена сообщениями. Давайте сравним популярные реализации стилей REST и SOAP.

15.Опишите общую структуру SOAP-сообщения.

Сообщение SOAP выглядит так: Envelope — корневой элемент, который определяет сообщение и пространство имен, использованное в документе. Header — содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации. Body — содержит сообщение, которым обмениваются приложения.

16.Что такое и что содержит Конверт (SOAP Envelope)?

Envelope — корневой элемент, который определяет сообщение и пространство имен, использованное в документе.

17.Что такое и что содержит Заголовок SOAP (SOAP Header)?

Header — содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации.

18.Что такое и что содержит Тело SOAP (SOAP Body)?

Body — содержит сообщение, которым обмениваются приложения.

20.Что такое graphql?

GraphQL - это язык запросов и манипулирования данными с открытым исходным кодом для API, а также среда выполнения для выполнения запросов с существующими данными.

21.Что такое Распознаватели (resolvers) в graphql?

Resolver или распознаватель — функция, которая возвращает данные для определённого поля. Resolver'ы возвращают данные того типа, который определён в схеме. Распознаватели могут быть асинхронными. С их помощью можно получать данные из REST API, базы данных или другого источника

23.Что такое валидация данных и для чего она нужна?

Валидация – это проверка продукта, процесса или системы на соответствие требованиям клиента. Если человек покупает смартфон, он ожидает как минимум возможности звонить, выходить в интернет и снимать на встроенную камеру. Всё это может быть прописано в инструкции по эксплуатации, но если при первом включении телефон не видит сим-карту, а снимки не сохраняются в памяти, считается, что валидация не пройдена.

24.Где и когда выполнять валидацию данных?

Для валидации требуется доступ к недоступной части состояния системы. Это особенно характерно для проверки данных, вводимых человеком через графический интерфейс пользователя. ...

Валидация требует полностью повторить логику обработки.

25.Как выполнять валидацию данных?

Валидация на стороне клиента — это первичная проверка введённых данных, которая существенно улучшает удобство взаимодействия с интерфейсом; обнаружение некорректных данных на стороне клиента позволяет пользователю немедленно их исправить. Если же проверка происходит только на сервере, процесс заполнения может быть более трудоёмким, так как требует повторения одних и тех же действий отправки данных на сервер для получения обратного ответа с сообщением о том, что нужно исправить.

27.Что такое запрос и мутация в graphql и чем они отличаются?

К первому виду относятся запросы на чтение данных, которые в терминологии GraphQL называются просто запросами (query) и относятся к букве R (reading, чтение) акронима CRUD. Запросы второго вида — это запросы на изменение данных, которые в GraphQL называют мутациями (mutation)

Список использованных источников

1. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
2. Одиночкина, С. В. Web-программирование PHP : учебно-методическое пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2012. — 79 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/43562 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
3. Ульман, Л. MySQL / Л. Ульман. — Москва : ДМК Пресс, 2008. — 352 с. — ISBN 5-94074-229-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/1241 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.
4. Даева, С. Г. Информационные системы и технологии: работа с веб-сервером nginx : учебно-методическое пособие / С. Г. Даева. — Москва : РТУ МИРЭА, 2020. — 75 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/167619 (дата обращения: 02.10.2021). — Режим доступа: для авториз. пользователей.