|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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»

**Студент группы** ИКБО-01-19 Кузин Данил Олегович

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

(подпись студента)

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

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

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

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

Москва 2021

**Оглавление**

[Цель работы 3](#_Toc86496211)

[Ход работы 3](#_Toc86496212)

[Спецификация API 11](#_Toc86496213)

[Вывод 11](#_Toc86496214)

[Ответы на вопросы к практической работе 11](#_Toc86496215)

[Список использованной литературы 20](#_Toc86496216)

# Цель работы

Используя серверную конфигурацию, разработанную в первой работе, научиться создавать API к 2 независимым сущностям.

# Ход работы

1. Был создана сложная серверная конфигурация, имеющая файловую систему представленную на Рисунке 1.

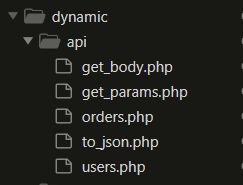


Рисунок - Файловая система проекта

Содержимое php-файлов представлено на листингах 1-8.

Листинг 1 – Файл toJson.php

<?php

function toJson($result)

{

$response = [];

while($row = $result->fetch\_array(MYSQLI\_ASSOC)) {

$response[] = $row;

}

return json\_encode($response);

}

?>

Листинг 2 – Файл get\_body.php

<?php

function getBody()

{

return json\_decode(file\_get\_contents('php://input'), true);

}

?>

Листинг 3 – Файл get\_params.php

<?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;

}

?>

Листинг 4 – Файл orders.php

<?php

include '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", "chocovoDB");

$body = getBody();

$params = getParams();

switch ($\_SERVER['REQUEST\_METHOD']) {

case 'GET':

$result = $mysqli->query("SELECT \* FROM orders");

echo toJson($result);

break;

case 'POST':

$stmt = $mysqli->prepare("INSERT INTO orders (title, count, address, status) VALUES (?, ?, ?, 'В обработке')");

$res = $stmt->bind\_param('sss', $body['title'], $body['count'], $body['address']);

$stmt->execute();

echo 1;

break;

case 'PUT':

$stmt = $mysqli->prepare("UPDATE orders SET title=?, count=?, address=?, status=? WHERE ID=?");

$res = $stmt->bind\_param('ssssi', $body['title'], $body['count'], $body['address'], $body['status'], $body['ID']);

$stmt->execute();

break;

case 'DELETE':

$stmt = $mysqli->prepare("DELETE FROM `orders` WHERE `ID`=?");

$res = $stmt->bind\_param('i', $params['ID']);

$stmt->execute();

echo 1;

break;

default:

header("HTTP/1.1 404 Not Found");

break;

}

?>

Листинг 5 – Файл users.php

<?php

include '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", "chocovoDB");

$body = getBody();

$params = getParams();

switch ($\_SERVER['REQUEST\_METHOD']) {

case 'GET':

$result = $mysqli->query("SELECT \* FROM users");

echo toJson($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;

}

?>

1. Программа была запущена и протестирована (Рисунок 2 - 9).

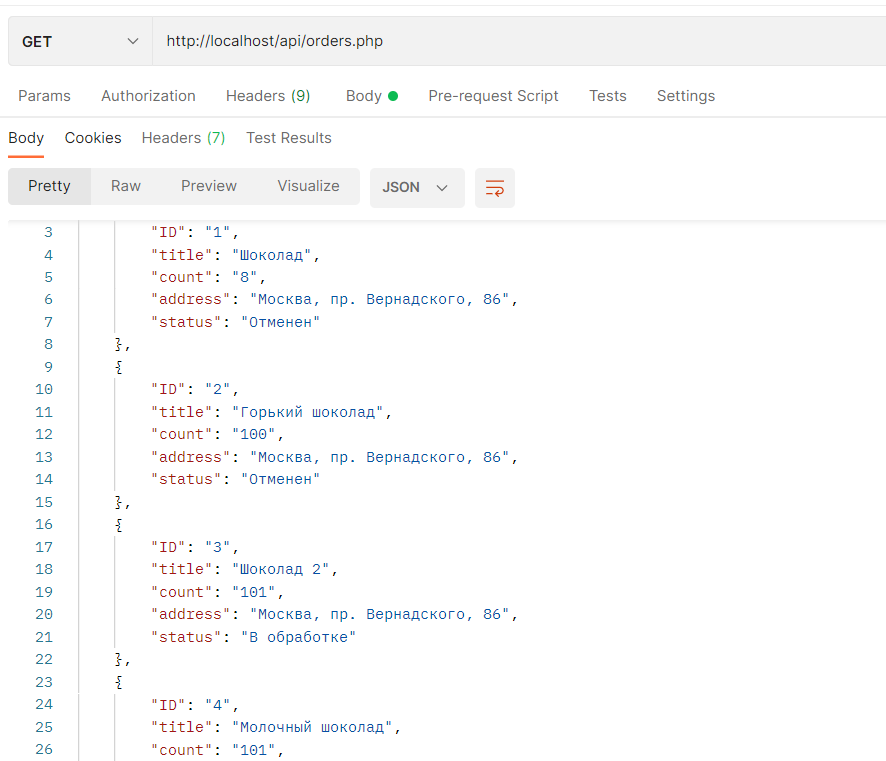


Рисунок – получение заказов

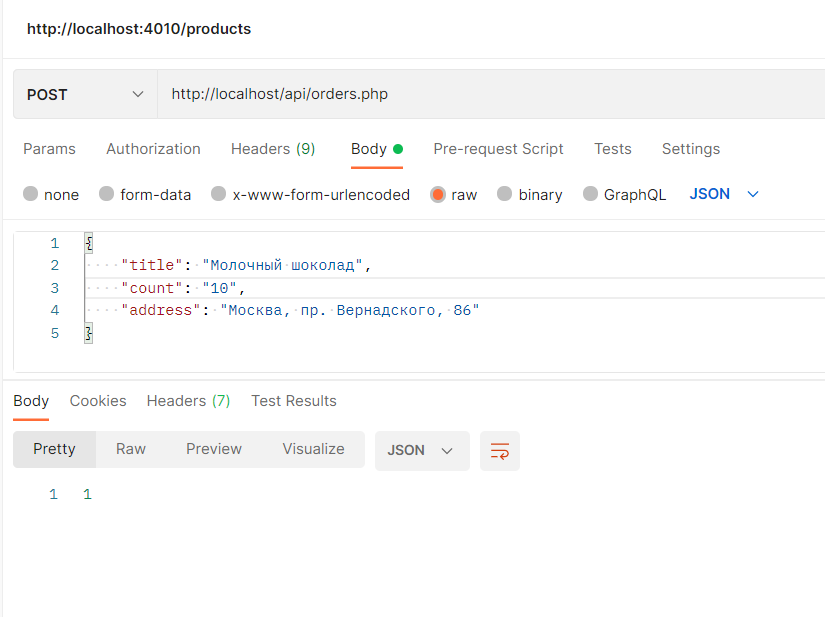


Рисунок – создание заказа

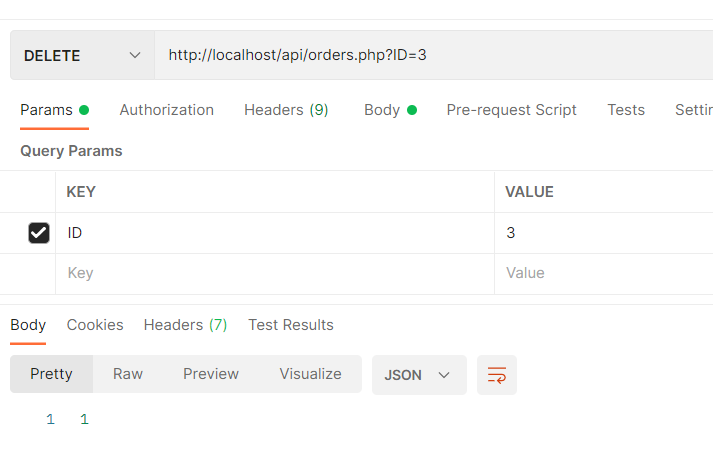


Рисунок – удаление заказа

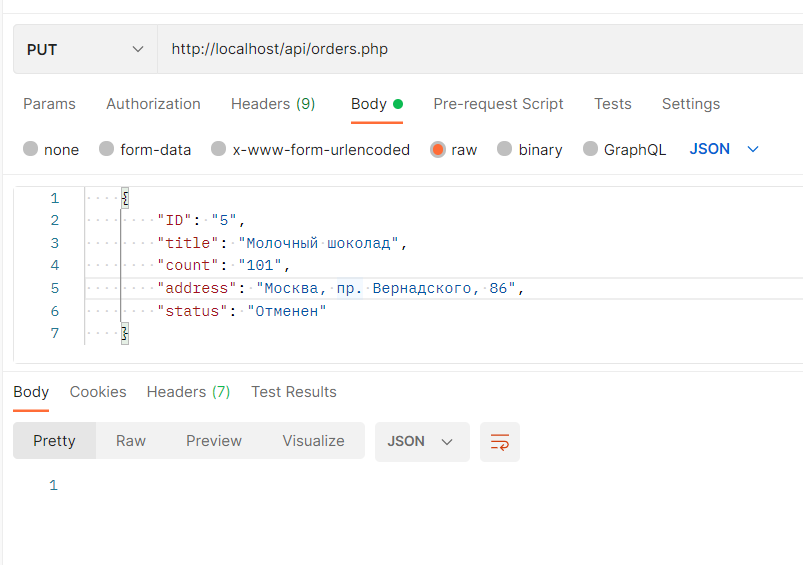


Рисунок – изменение данных о заказе

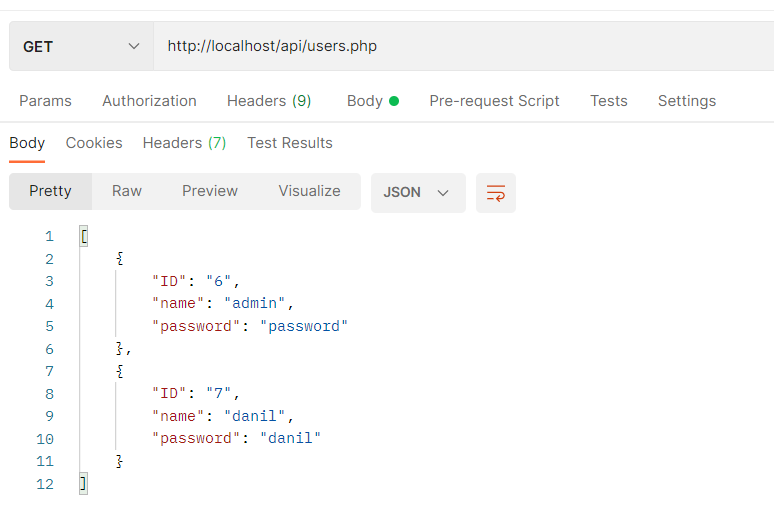


Рисунок 6 – получение данных об администраторах

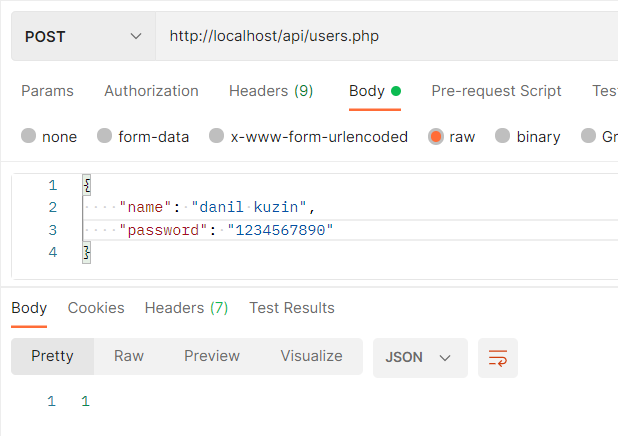


Рисунок 7 – создание администратора

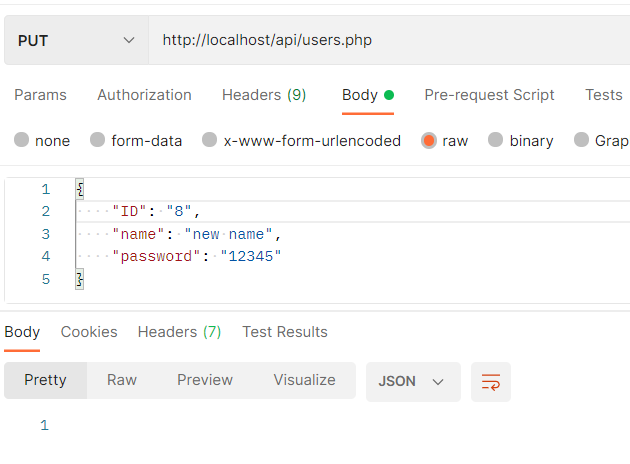


Рисунок 8 – изменение данных об администраторе

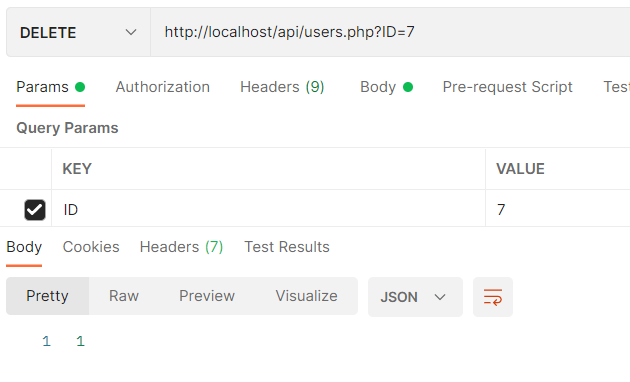


Рисунок 9 – удаление данных об администраторе

# Спецификация API

* GET запросы:  
  1) [/api/users](http://localhost/php/api/author).php Получение всех пользователей.  
  2) [/api/orders.php](http://localhost/php/api/author/4) Получение всех заказов.
* POST запросы:  
  1) [/api/users](http://localhost/php/api/author).php Создание пользователя.  
  2) [/api/orders.php](http://localhost/php/api/author/4) Создание заказа.
* DELETE запросы:  
  1) [/api/users](http://localhost/php/api/author).php . Удаление администратора по параметру ID

2) [/api/orders.php](http://localhost/php/api/author/4) Удаление заказа по параметру ID

* PUT запросы:

1. [/api/users](http://localhost/php/api/author).php. Обновление данных о пользователе
2. [/api/orders.php](http://localhost/php/api/author/4) Обновление данных о заказе

# Вывод

В результате работы было создано API к 2 независимым сущностям.

# Ответы на вопросы к практической работе

1. Что такое HTTP-запрос?

HTTP запросы - это сообщения, отправляемые клиентом, чтобы инициировать реакцию со стороны сервера.

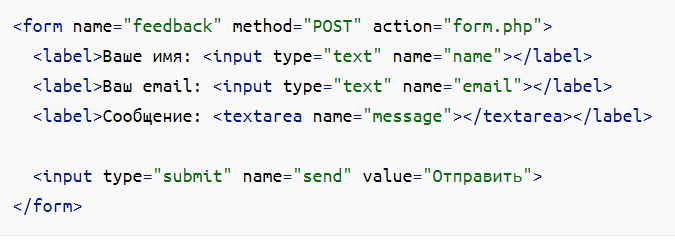
1. Опишите существующие HTTP-запросы.

GET – получение данных  
POST – добавление данных  
PATH – обновление данных  
DELETE – удаление данных

1. Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса?

Чтобы получить метод запроса мы используем глобальную переменную “$\_SERVER['REQUEST\_METHOD'];”, далее нам надо узнать url с помощью глобальной переменной $\_SERVER['REQUEST\_METHOD'], а чтобы получить данные мы используем метод file\_get\_contents('php://input'), а далее преобразуем данные в массив.

1. Опишите создание HTML-форм на PHP.



Почти весь приведённый код описывает внешний вид и содержание формы, но следует обратить внимание на два атрибута тега <form>, которые нужны для указания на способ обработки данных:

method — этот атрибут используется для определения метода HTTP, который будет использован для передачи данных на сервер.

Метод POST сообщает о намерении передать на сервер некоторую информацию, что, впрочем, не отменяет последующее получение контента.

action — содержит адрес PHP-скрипта, который должен обработать эту форму.

Все данные из формы находятся в глобальном ассоциативном массиве $\_POST.

1. Что такое API?

Api – это описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой

1. Опишите API как средство интеграции приложений.

Так как api – это описание способов взаимодействия между компьютерными программами, то мы можем создать свой собственный api, например, Rest api, который выдает нам данные в формате json, а далее в нашем веб приложении подключиться к этому api и манипулировать данными.

1. Что такое Web API?

Веб-API - это интерфейс прикладного программирования для веб-сервера или веб-браузера.

1. Приведите пример API

Обмен информацией о рейсах между авиакомпаниями и туристическими сайтами

-Использование Google Maps в приложении для совместных поездок (райдшеринга)

1. Что такое REST?

Rest - Это архитектурный стиль взаимодействия компонентов распределенной системы в компьютерной сети. Проще говоря, REST определяет стиль взаимодействия (обмена данными) между разными компонентами системы, каждая из которых может физически располагаться в разных местах.

1. Как организована передача данных в архитектуре REST?

В REST архитектуре клиенты отправляют на сервер запросы для получения или модификации данных, а сервера отправляют клиентам ответы на их запросы

1. Как организована работа REST?

Суть работы алгоритма заключается в паре действий, в зависимости от типа запроса. От работы сервера зависит функционал и способности архитектуры. В качестве пакета обычно отправляется JSON массив на указанный конкретный URL. Там срабатывает так называемая функция, а в зависимости от уже отправленных данных и текущего запроса начинается определенное действие. При этом не имеет значения, с какого устройства выслана информация — мобильное приложение или браузер компьютера.

1. Что такое 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.

1. Чем SOAP отличается от REST?

SOAP обозначает простой протокол доступа к объектам, тогда как REST обозначает передачу представительного состояния  
\* SOAP — это протокол, тогда как REST — это архитектурный паттерн  
\* SOAP использует сервисные интерфейсы для предоставления своих функций клиентским приложениям, а REST использует унифицированные локаторы сервисов для доступа к компонентам на аппаратном

1. Для чего нужен SOAP-процессор?

SOAP-процессоры, совместимые с версией 1.1, при получении сообщения, содержащего конверт с пространством имен версии 1.2, будут генерировать сообщение об ошибке.

1. Опишите общую структуру SOAP-сообщения

Сообщение SOAP состоит из заголовка — элемент SOAP-ENV:Header и основной части — элемент SOAP-ENV:Body. Заголовок может содержать метаданные, относящиеся к сообщению в целом.

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

Конверт (SOAP Envelope) Является самым «верхним» элементом SOAP сообщения. Содержит корневой элемент XML-документа. Описывается с помощью элемента Envelope с обязательным пространством имен

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

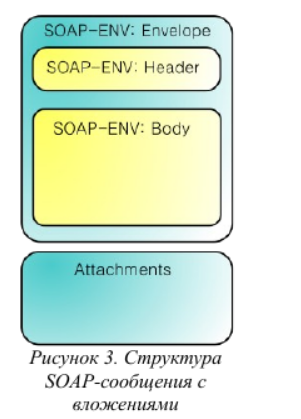
Header — содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации

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

Body — содержит сообщение, которым обмениваются приложения

1. Опишите SOAP-сообщение с вложением.

Существуют ситуации, когда клиент и сервер должны обмениваться данными в формате, отличном от текстового. Это могут быть данные в форматах каких-то приложений, мультимедийные данные и т.п. С точки зрения обмена данными все нетекстовые данные рассматриваются как данные в двоичных кодах.



Первая часть MIME-сообщения сообщения - часть SOAP -содержит XML: конверт SOAP с вложенными в него заголовком и телом сообщения. Остальные части - вложения - содержат данные в любом формате, двоичном или текстовом. Каждая часть предваряется MIME-заголовком, описывающим формат данных части и содержащим идентификатор части (Content-ID). По этому идентификатору тело SOAP-сообщения может ссылаться на вложения (href). Приведенный ниже фрагмент должен дать представление о структуре SOAP-сообщения с вложениями

1. Что такое graphql?

GraphQL – это язык запросов, используемый клиентскими приложениями для работы с данными, независимо от источников их изначального хранения: СУБД, файловое или объектное хранилище и пр. Можно сказать, что GraphQL (GQL) развивает REST-технологию, используя для передачи данных клиенту и получения их от него, простые GET или POST-запросы.

1. Что такое Распознаватели (resolvers) в graphql?

GraphQL Распознаватели (Resolvers). Распознаватели ― это специальные функции, которые возвращают данные для соответствующих полей. GraphQL-сервер без распознавателя не сможет определить, что делать с запросом и как получить данные. Иными словами, распознаватели ― это инструкции по выполнению запросов

1. Из чего состоит экосистема graphql, что нужно, чтобы использовать данную технологию?

На стороне сервера сервер GraphQL может быть реализован на любом языке, который может создавать веб-сервер. В дополнение к JavaScript, Ruby, Python, Scala, Java, Clojure, Go и .NET имеют реализации для справки. Основной алгоритм для выполнения запросов на стороне сервера также очень прост: запрос обходит поле за полем и выполняет преобразователь для каждого поля для обработки операций с данными.

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

Валидация – это процесс проверки данных на соответствие различным критериям. При разработке любого приложения в большинстве случаев разработчику приходится иметь дело с обработкой данных, которые ввел пользователь в соответствующие поля на форме

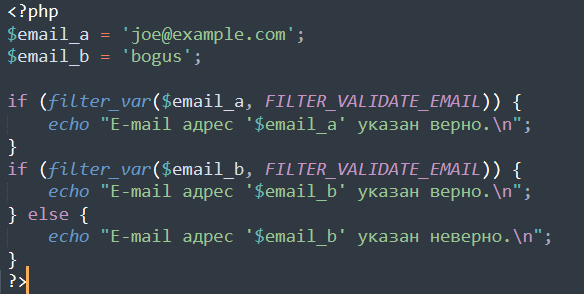
1. Где и когда выполнять валидацию данных?

Это может быть пользовательский интерфейс, через который человек вводит данные. Это может быть файл, содержащий настройки программы или данные, которые программа должна обработать. Это может быть база данных, в которую информация может попадать из других программ. Это может быть сетевой протокол обмена данными с другими программами. Наконец, это может быть программный интерфейс, который использует другая программа, вызывая некоторые функции/процедуры и передавая в них параметры.

1. Как выполнять валидацию данных?

**Посимвольная проверка**. Как правило такие проверки выполняются в пользовательском интерфейсе, по мере ввода данных. Но не только. Например, лексический анализатор компилятора тоже выявляет недопустимые символы непосредственно в процессе чтения компилируемого файла. Поэтому такие проверки можно условно назвать «лексическими»  
**Проверка отдельных значений**. Для пользовательского интерфейса это проверка значения в отдельном поле, причём выполняться она может как по мере ввода (проверяется то неполное значение, которое введено к настоящему моменту), так и после завершения ввода, когда поле теряет фокус. Для программного интерфейса (API) это проверка одного из параметров, переданных в вызываемую процедуру. Для данных, получаемых из файла, это проверка какого-то прочитанного фрагмента файла. Такие проверки, опять-таки по аналогии с компиляторной терминологией, можно назвать «синтаксическими».  
**Проверка состояния системы после обработки данных**. Наконец, есть последний способ, к которому можно прибегнуть, если валидацию непосредственно входных данных выполнить не удаётся — можно попытаться их обработать, но оставить возможность вернуть всё к исходному состоянию. Такой механизм часто называется транзакционным.

1. Приведите пример с поэтапной валидацией данных.



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

Отличаются от запросов тем, что начинаются с ключевого слова mutation. Технически любой запрос GraphQL может быть реализован, чтобы вызвать запись данных. Но существует соглашение о том, что любые операции, вызывающие записи, должны быть отправлены явно через мутацию. Помимо различия в семантике, существует одна важная техническая разница:Поля Query могут выполняться параллельно движком GraphQL, в то время как поля верхнего уровня Mutation ДОЛЖНЫ выполняться последовательно в соответствии со спецификацией.

# Список использованной литературы

1. Документация Docker — Текст : электронный // Docker docs : [Электронный ресурс]. — URL: https://docs.docker.com/ (дата обращения: 04.09.2021).
2. Руководство по php — Текст : электронный // php : [Электронный ресурс]. — URL: https://www.php.net/ (дата обращения: 22.09.2021).
3. Docker Hub — Текст : электронный // Docker Hub : [Электронный ресурс]. — URL: https://hub.docker.com/ (дата обращения: 04.09.2021).
4. Справочник по php — Текст : электронный // Docker Hub : [Электронный ресурс]. — URL: https://php.ru/manual/ (дата обращения: 22.09.2021).