rs

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-01-20 Дегтярев В.С.

_____
(подпись студента)

Руководитель практической работы Волков М.Ю.

_____
(подпись руководителя)

Работа представлена «____»_____ 2022 г.

Допущен к работе «___» _____ 2022 г.

Москва 2022

ОГЛАВЛЕНИЕ

Цель работы

Реализация взаимодействия клиента и сервера с использованием технологии API.

Задание

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 работы

Тема практической работы была взята из работы №3. Создадим два php файла: users.php и menu.php. Эти файлы будут выполнять роль эндпоинтов. Функционал api представлен в таблицах 1-2.

Т

Таблица 1 - Api для menu

Тип запроса	url	Функционал
GET	/sum	Суммирование цен всех блюд
POST	/sale	Изменение цен для всех продуктов

Таблица 2 - Api для users

Тип запроса	url	Функционал
GET	/read	Вывести информацию об одном пользователе
POST	/create	Создать нового пользователя
POST	/update	Обновить пользователя
POST	/delete	Удалить пользователя

Каждый запрос параметризован по схеме x-www-form-urlencoded. Работа запросов была протестирована с использованием ПО Postman, проверки представлены на рисунках 1-6. Файлы проекта расположены в удаленном репозитории на платформе GitHub.com, расположенном по ссылке: https://github.com/Degtyarev02/server-parts

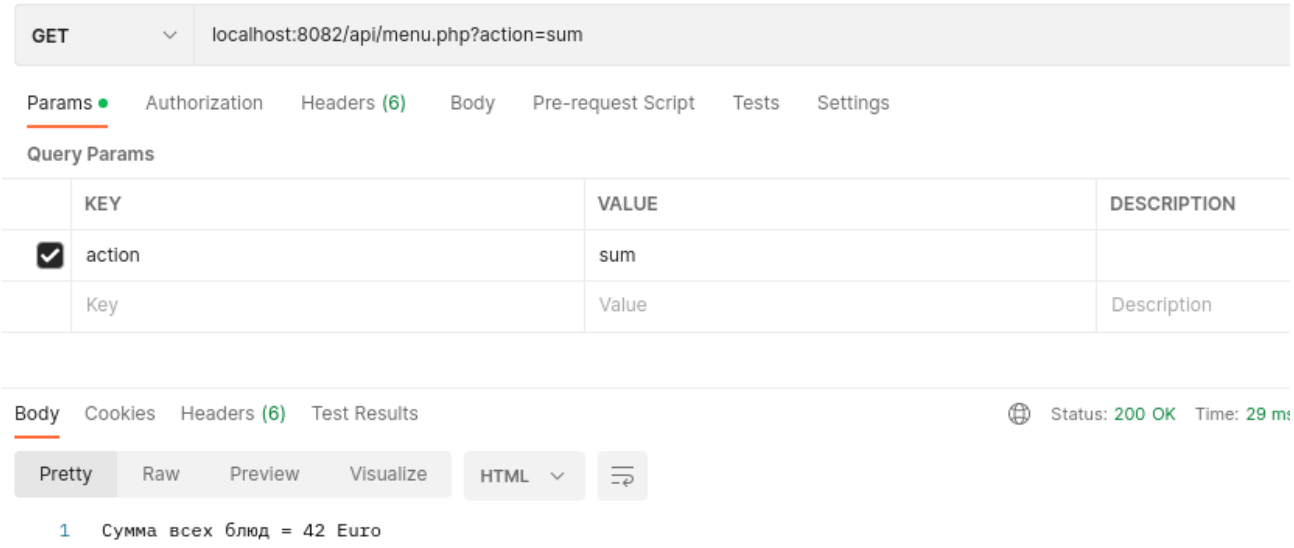

Рисунок 1 - Сумма всех блюд

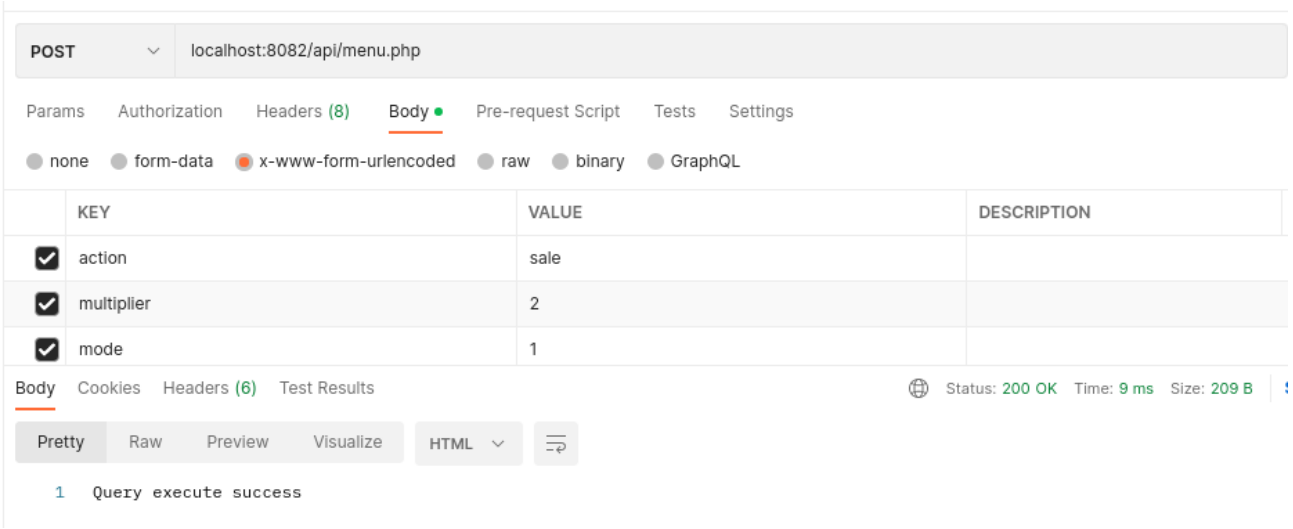

Рисунок 2 - Уменьшение цены на все блюда в два раза

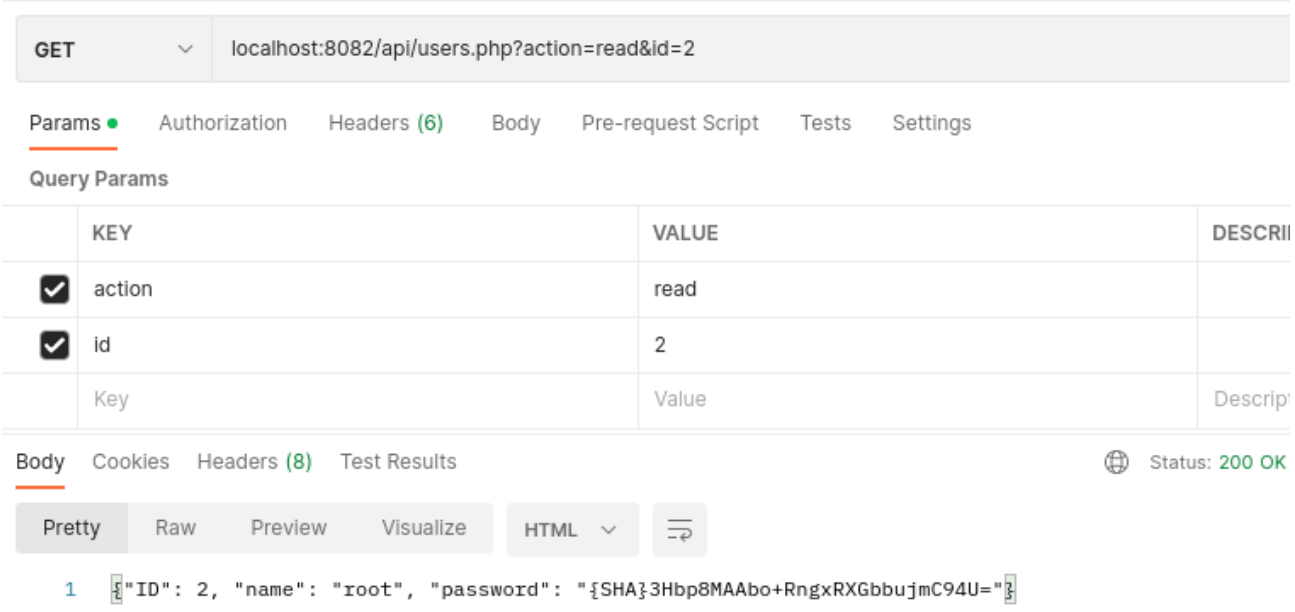

Рисунок 3 - Получение пользователя по ID

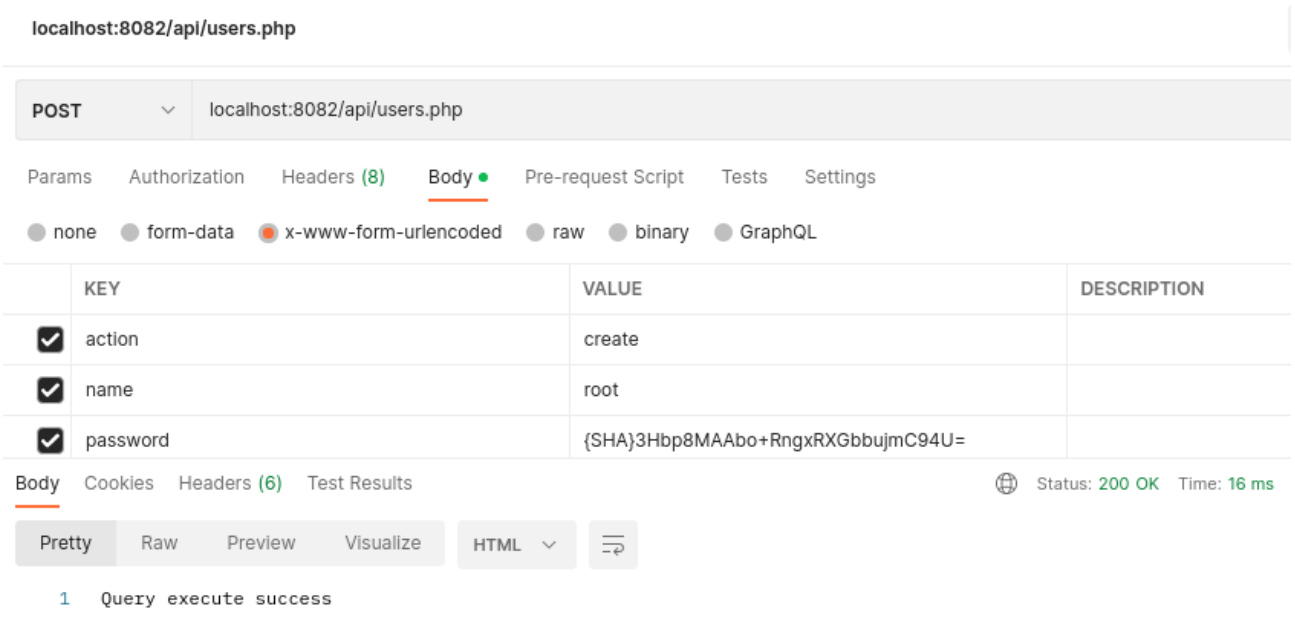

Рисунок 4 - Создание пользователя

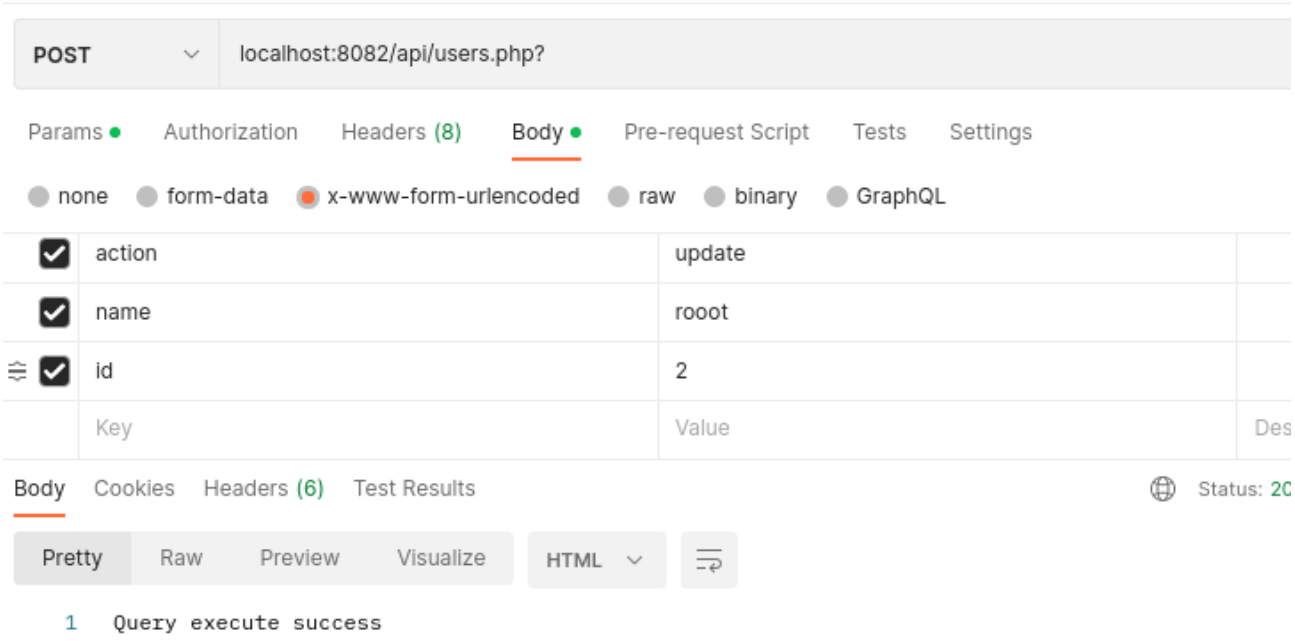

Рисунок 5 - Обновление пользователя

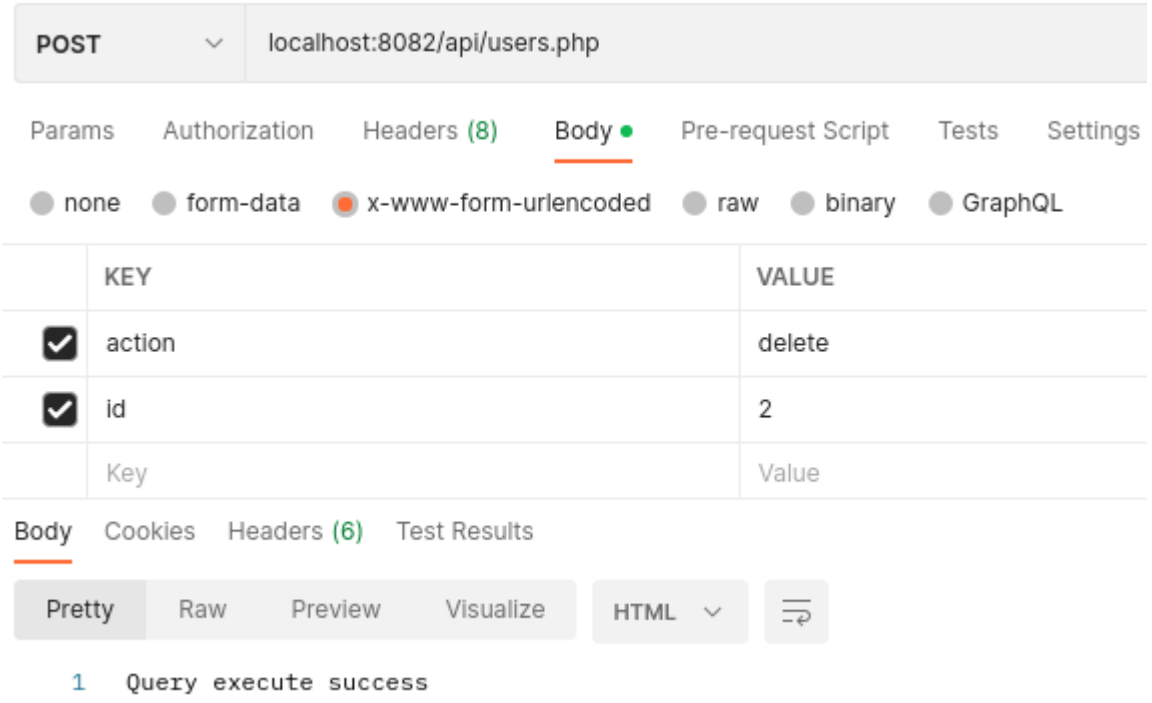

Рисунок 6 - Удаление пользователя

Выводы

В ходе данной практической работы, были получены навыки в создании REST-full API реализующий CRUD на языке программирование PHP.

Ответы на вопросы

1. Запросы (HTTP Requests) — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок. Ответы (HTTP Responses) — сообщения, которые сервер отправляет в ответ на клиентский запрос.

2. Существуют следующие методы запросов: метод GET запрашивает представление ресурса, запросы с использованием этого метода могут только извлекать данные; HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа; POST используется для отправки сущностей к определённому ресурсу, часто вызывает изменение состояния или какие-то 9 побочные эффекты на сервере; PUT заменяет все текущие представления ресурса данными запроса; DELETE удаляет указанный ресурс; CONNECT устанавливает "туннель" к серверу, определённому по ресурсу; OPTIONS используется для описания параметров соединения с ресурсом; TRACE выполняет вызов возвращаемого тестового сообщения с ресурса; PATCH используется для частичного изменения ресурса

3. Для обработки запроса достаточно создать файл-обработчик, который будет служить эндпоинтом, внутри для доступа к параметрам можно использовать предопределенные массивы _GET и _POST.

4. Чтобы создать форму используется тег с двумя атрибутами, который можно вывести на страницу используя функцию echo: атрибут action. С его помощью указывается адрес, на который отправятся введенные на форме данные, атрибут method. С его помощью указывается HTTP-метод отправки формы (get, post, put, и т.д.).

5. API — описание способов, взаимодействия одной компьютерной программы на другую. Обычно входит в описание какого-либо

интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы.

6. API как средство интеграции приложений Если программу (модуль, библиотеку) рассматривать как чёрный ящик, то API - это множество «ручек», которые доступны пользователю данного ящика, которые он может вертеть и дёргать. Программные компоненты взаимодействуют друг с другом посредством API.

7. Web API или Web Service API -это интерфейс обработки приложений между веб-сервером и веб-браузером. Все веб-сервисы являются API, но не все API являются веб-сервисами. REST API - это особый тип Web API, в котором используется стандартный архитектурный стиль, описанный выше.

8. Примером API может служить данная работа.

9. 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. Передача данных осуществляется через запросы, входящие параметры могут передаваться через url, заголовки или тело (например json объекты), запросы также возвращают значения от API. Передача данных обычно осуществляется через протоколы http и https, но могут быть использованны и другие.

11. Для веб-сайта это код, который позволяет двум программам взаимодействовать друг с другом. API предлагает разработчикам правильный способ написания программы, запрашивающей услуги у операционной системы или другого приложения. Проще говоря, это своего рода стандарт, который позволяет программам и приложениям понимать друг друга; это язык интернета, который необходим для работы практически всех сайтов и приложений.

12. 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. REST — это архитектурный стиль. SOAP — это формат обмена сообщениями. На верхнем уровне SOAP ограничивает структуры ваших сообщений, тогда как REST — это архитектурный подход, ориентированный на использование HTTP в качестве транспортного протокола.

14.SOAP-процессор, совместимый с v1.1, генерирует ошибку при получении сообщения, содержащего пространство имен конверта v1.2. SOAPпроцессор, совместимый с v1.2, генерирует ошибку VersionMismatch, если он получает сообщение, которое не включает пространство имен конверта v1.2. Каждое сообщение SOAP имеет корневой элемент Envelope.

15.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. Fault — необязательный элемент, который предоставляет информацию об ошибках, которые произошли при обработке сообщений.

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

17.Header — содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации. Содержится в сообщении. 18.Body — содержит сообщение, которым обмениваются приложения. Содержится в сообщении. 19.Можно посылать вложением различные форматы: PDF, изображения или другие двоичные данные. Сообщения SOAP работают вместе с расширением MIME, в котором предусмотрено multipart/related

20.GraphQL — это язык запросов для API -интерфейсов и среда, в

которой они выполняются. С помощью GraphQL можно получить данные из API и передать их в приложение (от сервера к клиенту). Официальная документация GraphQL есть только на английском языке, на русский язык пока ещё не переведена. GraphQL разработали в 2012 году как альтернативу REST.

21.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.

22.Необходимо всего два компонента чтобы начать: Сервер GraphQL для обработки запросов к API, Клиент GraphQL, который будет подключаться к endpoint.

23.Валидация данных — это процесс проверки данных различных типов по критериям корректности и полезности для конкретного применения. Валидация данных проводится, как правило, после выполнения операций ETL и для подтверждения корректности результатов работы моделей машинного обучения.

24.Валидация данных является одним из методов, позволяющих исключить поступление на вход информационной системы или её компонент заведомо ошибочных, неполных или неточных данных, которые могут привести к ошибочным результатам работы, утрате данных и сбоям в работе систем. Причинами появления таких ошибочных данным могут быть ошибки в процессе ручного ввода данных, в результате ошибок в алгоритмах и 13 программах, в процессе хранения и передачи данных, а также при создании данных датчиками и устройствами различного оборудования и IoT. В процессе валидации могут осуществляется корректировка либо исключение данных, файлов, пакетов и записей, информирование оператора, изменение алгоритма работы информационной системы. если всё-таки проверка нужна, логика подсказывает, что удобно проверять данные в том месте, где они попадают в программу из внешнего мира. После такой проверки можно быть уверенным,

что в программу попадают правильные данные и в дальнейшем они могут использоваться без дополнительных проверок

25.Валидация данных может осуществляться следующими методами: посимвольная проверка, проверка отдельных значений, совокупность входных значений, проверка состояния системы после обработки данных.

26.Валидацию данных можно и нужно выполнять в несколько этапов, усложняя проверки. Сначала, по мере ввода, следим за тем, чтобы данные не содержали недопустимых символов. Например, для числового поля пользователю может быть запрещён ввод нецифровых символов. После того, как ввод завершён, можно проверить всё значение целиком. Для введённого числа могут быть какие-то ограничения, например, оно не должно превышать определённого максимального допустимого значения. Если наше числовое поле представляет собой возраст, оно должно находиться в пределах от 0 до, скажем, 120. Когда заполнены все поля, можно проверить, согласованы ли введённые значения друг с другом. Например, если в форме кроме поля для указания возраста есть поле для ввода номера паспорта, приложение может проверить, что при заполнении номера паспорта возраст должен быть не менее 14 лет. Наконец, если всё введено корректно, можно попытаться начать обработку, выполняя проверки по ходу дела, а также в самом конце, и если что-то пошло не так, выполнить откат к исходному состоянию. Ну и, конечно же, проверки на следующем уровне могут 14 подстраховывать проверки предыдущих уровней. Скажем, для вебприложений обязательной является проверка данных, пришедших на сервер в HTTP-запросе, независимо от того, выполнялась ли перед этим предварительная валидация в браузере или нет. Причина этого в том, что проверку на клиентской стороне можно обойти. Для других видов приложений обойти проверки не так просто, но иногда тоже вполне возможно, как показано в примере чуть ниже.

27.Запрос является интерактивным. Это означает, что Вы можете изменить его, как Вам нравится, и увидеть новый результат. Попробуйте добавить поле appearsIn в объект hero в запросе, и увидите новый результат. В

запросе указываются те данные, которые необходимо получить, возможно указание параметров. Запрос GraphQL используется для чтения или извлечения значений, в то время как мутация используется для записи или публикации значений

Список информационных источников:

1. Документация по языку PHP, электронный ресурс URL: https://php.net (дата обращения: 11.09.2022) – Текст: электронный;
2. Документация по системе Docker, электронный ресурс URL: https://docs.docker.com (дата обращения: 11.09.2022) – Текст: электронный.
3. Документация по HTML, CSS, JavaScript, электронный ресурс URL: https://developer.mozilla.org (дата обращения 11.09.2022) – Текст: электронный.