|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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-02-20 Богайчук Илья Сергеевич

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

(подпись студента)

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

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

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Москва 2022

СОДЕРЖАНИЕ

[1. Цель работы 4](#_Toc119405060)

[2. Ход работы 4](#_Toc119405061)

[2.1. Добавление дисциплины 4](#_Toc119405062)

[2.2. Удаление дисциплины 5](#_Toc119405063)

[2.3. Получение информации о дисциплине 6](#_Toc119405064)

[2.4. Добавление пользователя 7](#_Toc119405065)

[2.5. Удаление пользователя 7](#_Toc119405066)

[2.6. Смена пароля у пользователя 9](#_Toc119405067)

[2.7. Просмотр имени пользователя 10](#_Toc119405068)

[3. Ответы на вопросы к практической работе 11](#_Toc119405069)

[3.1. Что такое HTTP-запрос? 11](#_Toc119405070)

[3.2. Опишите существующие HTTP-запросы. 11](#_Toc119405071)

[3.3. Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса? 12](#_Toc119405072)

[3.4. Опишите создание HTML-форм на PHP. 12](#_Toc119405073)

[3.5. Что такое API? 12](#_Toc119405074)

[3.6. Опишите API как средство интеграции приложений. 12](#_Toc119405075)

[3.7. Что такое Web API? 13](#_Toc119405076)

[3.8. Приведите пример API. 13](#_Toc119405077)

[3.9. Что такое REST? 13](#_Toc119405078)

[3.10. Как организована передача данных в архитектуре REST? 14](#_Toc119405079)

[3.11. Как организована работа REST? 14](#_Toc119405080)

[3.12. Что такое SOAP? 14](#_Toc119405081)

[3.13. Чем SOAP отличается от REST? 14](#_Toc119405082)

[3.14. Для чего нужен SOAP-процессор? 15](#_Toc119405083)

[3.15. Опишите общую структуру SOAP-сообщения. 15](#_Toc119405084)

[3.16. Что такое и что содержит Конверт (SOAP Envelope)? 15](#_Toc119405085)

[3.17. Что такое и что содержит Заголовок SOAP (SOAP Header)? 16](#_Toc119405086)

[3.18. Что такое и что содержит Тело SOAP (SOAP Body)? 16](#_Toc119405087)

[3.19. Опишите SOAP-сообщение с вложением. 16](#_Toc119405088)

[3.20. Что такое graphql? 17](#_Toc119405089)

[3.21. Что такое Распознаватели (resolvers) в graphql? 17](#_Toc119405090)

[3.22. Из чего состоит экосистема graphql, что нужно, чтобы использовать данную технологию? 17](#_Toc119405091)

[3.23. Что такое валидация данных и для чего она нужна? 18](#_Toc119405092)

[3.24. Где и когда выполнять валидацию данных? 18](#_Toc119405093)

[3.25. Как выполнять валидацию данных? 19](#_Toc119405094)

[3.26. Приведите пример с поэтапной валидацией данных. 20](#_Toc119405095)

[3.27. Что такое запрос и мутация в graphql и чем они отличаются? 20](#_Toc119405096)

[4. Ссылка на удаленный репозиторий проекта 22](#_Toc119405097)

[ЗАКЛЮЧЕНИЕ 22](#_Toc119405098)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 22](#_Toc119405099)

1. 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.
* Технические требования к реализации интерфейса:
  + Доступ как минимум к 2 независимым сущностям.
  + Реализация как минимум операций группы CRUD (создание, чтение, обновление, удаление). Приветствуется реализация дополнительной функциональности.
  + Тестирование всех функциональных возможностей созданного интерфейса с использованием программы Postman.

1. Ход работы

Вариант: «Электронная зачетная книжка»

Для сущности «elap» были созданы следующие API функции:

* 1. Добавление дисциплины

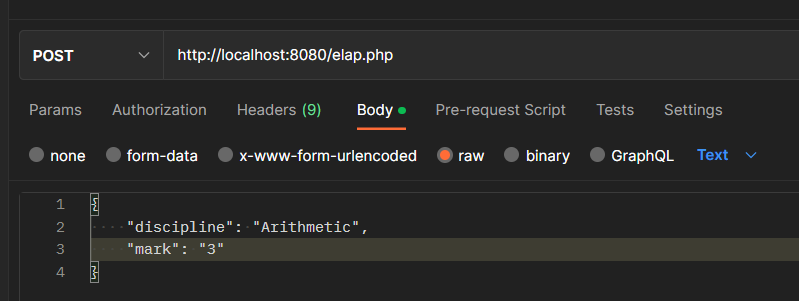


Рисунок 2.1.1 – Добавление зайца

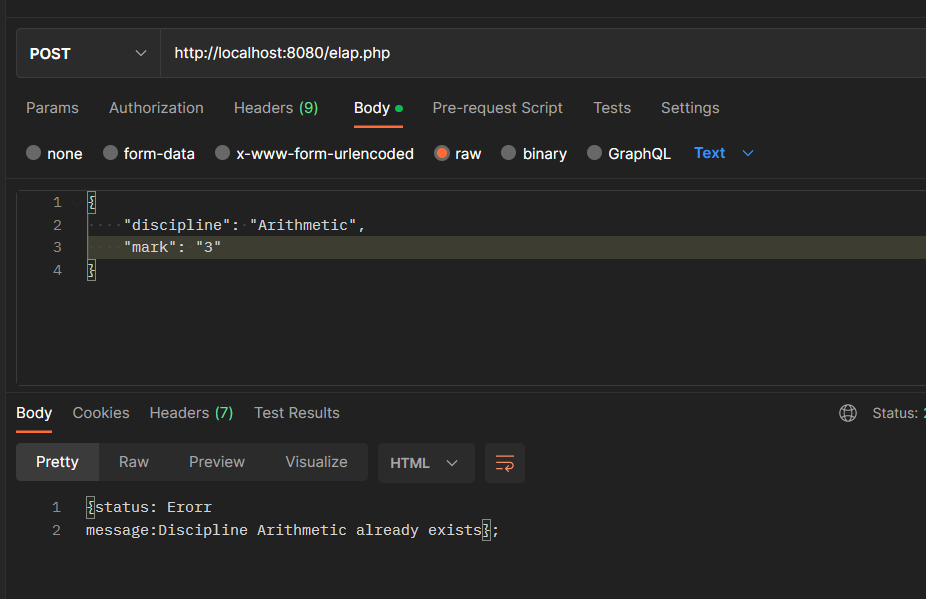


Рисунок 2.1.2 – Добавление дисциплины

* 1. Удаление дисциплины

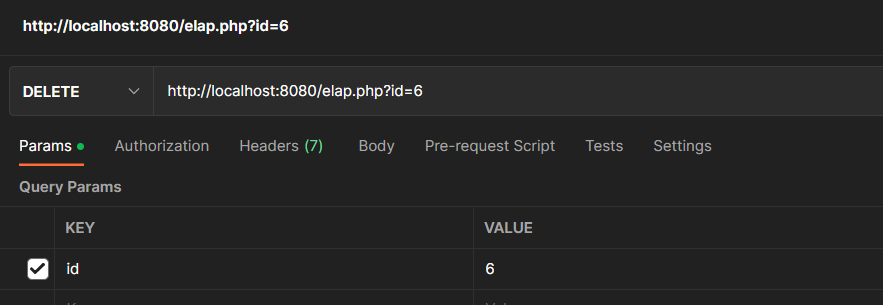


Рисунок 2.2.1 – Удаление дисциплины

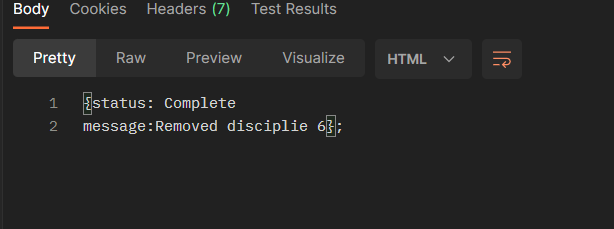


Рисунок 2.2.2 – Удаление дисциплины

* 1. Получение информации о дисциплине

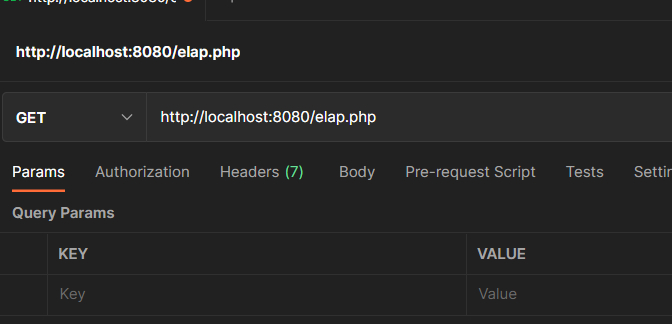


Рисунок 2.3.1 – Получение информации дисциплине

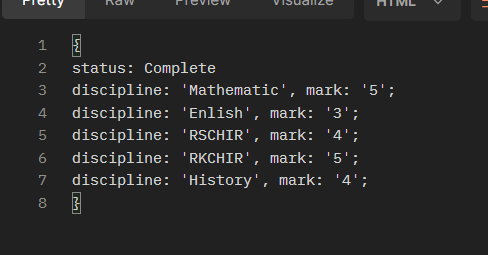


Рисунок 2.3.2 – Получение информации о дисциплине

Для сущности «Пользователи» были созданы следующие API функции:

* 1. Добавление пользователя

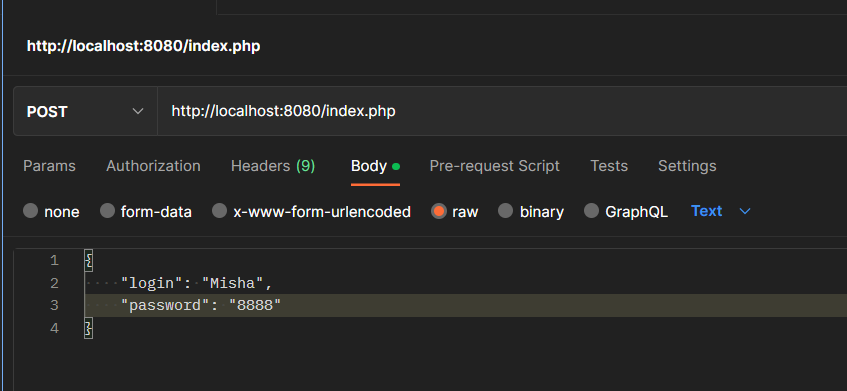


Рисунок 2.4.1 – Добавление пользователя

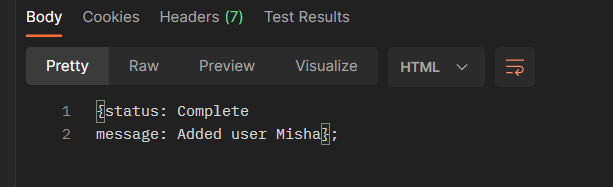


Рисунок 2.4.2 – Добавление пользователя

* 1. Удаление пользователя

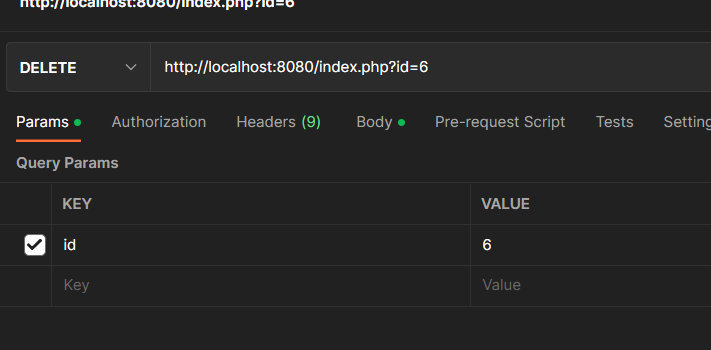


Рисунок 2.5.1 – Удаление пользователя

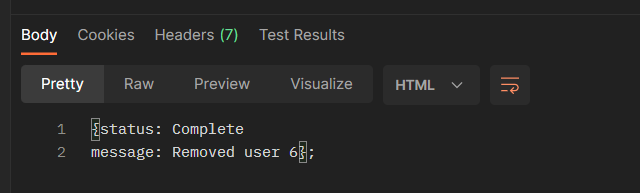


Рисунок 2.5.2 – Удаление пользователя

* 1. Смена пароля у пользователя

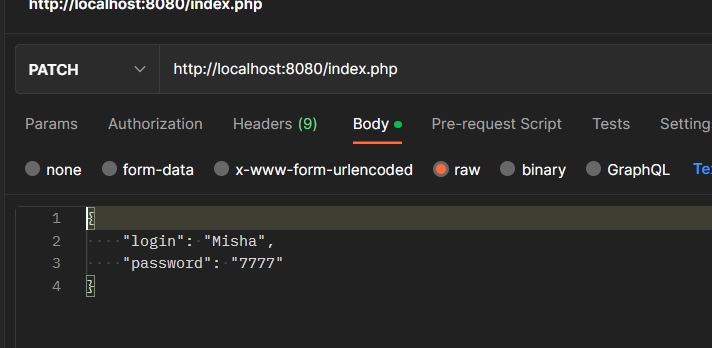


Рисунок 2.6.1 – Смена пароля у пользователя

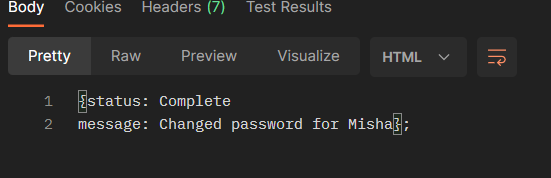


Рисунок 2.6.2 – Смена пароля у пользователя

* 1. Просмотр имени пользователя

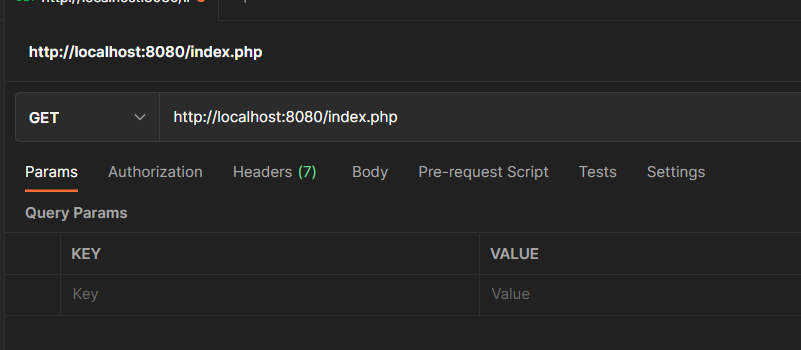


Рисунок 2.7.1 – Просмотр пользователя

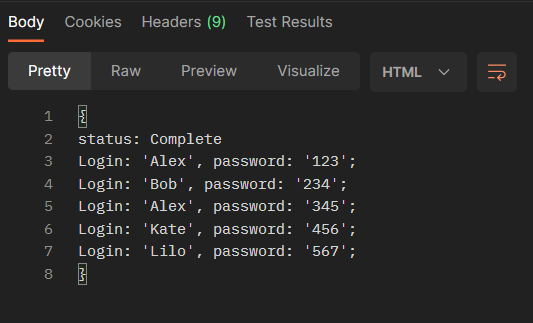


Рисунок 2.7.2 – Просмотр пользователя

* 1. Спецификация API.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Путь | Действие | Вход | Выход |
| GET | http://localhost:8080/index.php  http://localhost:8080/elap.php | Получение всех данных из сущности. | - | Данные в формате JSON. |
| GET | http://localhost:8080/index.php?id=1  http://localhost:8080/elap.php?id=1 | Получение данных из сущности по индексу. | Параметр ID в формате int | Данные в формате JSON. |
| POST | http://localhost:8080/index.php  http://localhost:8080/elap.php | Добавление записей в сущности. | Данные в формате JSON | Данные в формате JSON. |
| PATCH | http://localhost:8080/index.php  http://localhost:8080/elap.php | Изменение записей в сущностях. | Данные в формате JSON | Данные в формате JSON. |
| DELETE | http://localhost:8080/index.php?id=1  http://localhost:8080/elap.php?id=1 | Удаление записей из сущностей. | Параметр ID в формате int | Данные в формате JSON. |

1. Ответы на вопросы к практической работе
   1. Что такое HTTP-запрос?

HTTP requests – сообщения, которые отправляются клиентом на сервер чтобы вызвать выполнение некоторых действий

* 1. Опишите существующие HTTP-запросы.
* GET (запрос содержимого ресурса) - можно передать с параметрами в URI. Существует 3 типа: обычный, условный, частичный, - различающихся дополнительными заголовками
* POST (передача данных ресурсу) - передача данных осуществляется в теле запроса, поэтому, возможно, передача не только текстовых параметров, но и файлов
* HEAD - (запрос ресурса) - аналогичен GET, но для ответа на этот запрос тело отсутствует, т.е. присутствует только строка состояния и заголовки. Используется для запроса метаданных и проверки наличия ресурса
* PUT (загрузка содержимого запроса на ресурс) - в отличие от POST, где предполагается обработка содержимого, то для метода PUT клиент предполагает соответствие содержимого ресурсу, также повторные идентичные запросы будут давать тот же результат, в отличие от POST
* PATCH (частичное изменение ресурса) - можно грубо назвать данный метод обновляющим ресурс, но его нельзя использовать в HTML-формах
* OPTIONS (определение возможностей веб-сервера или параметров соединения для конкретного ресурса) - в ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях
* DELETE (удаляет указанный ресурс) - метод не разрешен к использованию в HTML-формах
  1. Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса?

Обработка запроса предполагает: получение HTTP-метода запроса, выделение параметров запроса, совершение предполагаемой функциональности и отправка результата. Параметры запроса распложены в $\_GET.

* 1. Опишите создание HTML-форм на PHP.

Инициализируется сеанс, данному сеансу задаются параметры: URL и значение для возврата результата передачи в качестве строки, после выполняется запрос и закрывается сеанс.

* 1. Что такое API?

API (Application Programming Interface) − прикладной программный интерфейс) − набор функций и подпрограмм, обеспечивающий взаимодействие клиентов и серверов.

API (в клиент-сервере) − 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.

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.

* 1. Что такое Web API?

Web API − используется в веб-разработке, содержит, как правило, определённый набор HTTP-запросов, а также определение структуры HTTP-ответов, для выражения которых используют XML− или JSON−форматы.

Web API является практически синонимом для веб-службы, хотя в последнее время за счёт тенденции Web 2.0 осуществлен переход от SOAP к REST типу коммуникации. Веб-интерфейсы, обеспечивающие сочетание нескольких сервисов в новых приложениях, известны как гибридные

* 1. Приведите пример API.

API — это спецификация возможных взаимодействий с компонентом ПО. Например, если бы машина была компонентом ПО, в ее API содержалась бы информация о том, что машина может ускоряться, тормозить и включать радио. Ее API также бы включало информацию о том, как ускоряться.

* 1. Что такое REST?

REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.[[1]](https://ru.wikipedia.org/wiki/REST#cite_note-1) REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

* 1. Как организована передача данных в архитектуре REST?

Филдинг описал концепцию построения распределенного приложения, при которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе сервера (желаемом представительном состоянии), и сервер не обязан сохранять информацию о состоянии клиента («клиентской сессии»)

При реализации API, особенно для работы с данными в RESTful стиле предполагается реализации группы основных операций CRUD(создание, чтение, обновление, удаление).

* 1. Как организована работа REST?

В REST API код запросов остается на стороне клиента, а код для доступа к данным — на стороне сервера. Это упрощает организацию API, позволяет легко переносить пользовательский интерфейс на другую платформу и дает возможность лучше масштабировать серверное хранение данных.

* 1. Что такое SOAP?

Протокол SOAP − набор правил для обмена структурированными сообщениями в распределенной вычислительной среде. Это набор правил для обмена сообщениями с использованием протоколов прикладного уровня, чаще всего HTTP. Данный протокол поддерживается консорциумом Всемирной паутины - организацией, разрабатывающей и внедряющей технологические стандарты для Всемирной паутины. Если коротко, то SOAP может расширить HTTP для обмена сообщениями в формате XML.

* 1. Чем SOAP отличается от REST?

REST — на самом деле архитектурный стиль, а не протокол. В отличие от SOAP, REST не подкреплен официальным стандартом. Фактически, он основывается на соглашениях.

* 1. Для чего нужен SOAP-процессор?

Важной частью протокола является SOAP-процессор. В общем случае это обработчик на сервере, отвечающий за обработку SOAP-сообщения в XML формате.

* 1. Опишите общую структуру SOAP-сообщения.

SOAP-сообщение состоит из:

1. Envelope(конверт) - корневой элемент, который определяет сообщение и пространство имен, используемое в документе.

2. Header(заголовок) - содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации.

3. Body(тело) - содержит сообщение, которым обмениваются приложения.

4. Fault(Неисправность) - необязательный элемент, который предоставляет информацию об ошибках, которые произошли при обработке сообщений.

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

SOAP-конверт — это механизм упаковки. SOAP-конверт указывает на начало и конец сообщения, чтобы получатель знал, когда было получено все сообщение. Каждый конверт содержит ровно одно SOAP-тело. Описывается с помощью элемента Envelope с обязательным пространством имен http://www.w3.org/2003/05/soap-envelope для версии 1.2 и http://schemas.xmlsoap.org/soap/ для версии 1.1. Важно понимать, что SOAPпроцессор, совместимый с v1.1, генерирует ошибку при получении сообщения, содержащего пространство имен конверта v1.2. Envelope может иметь необязательный дочерний элемент Header c тем же пространством имен − заголовок. Если этот элемент присутствует, то он должен быть первым прямым дочерним элементом конверта. Элементы Header и Body могут содержать элементы из различных пространств имен.

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

SOAP-заголовок − это механизм расширения, который обеспечивает способ передачи информации в сообщениях SOAP, которая не является полезной нагрузкой приложения. Такая "контрольная" информация включает, например, директивы о передаче или контекстуальную информацию, связанную с обработкой сообщения. Это позволяет расширять сообщение SOAP в зависимости от конкретного приложения.

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

Элемент SOAP-Body обязательно записывается сразу за элементом Header, если он есть в сообщении, или первым в SOAP-сообщении, если заголовок отсутствует. В элемент Body можно вложить произвольные элементы, спецификация никак не определяет их структуру. Определен только один стандартный элемент, который может быть в теле сообщения - элемент сообщения Fault, содержащий сообщение об ошибке.

* 1. Опишите SOAP-сообщение с вложением.

Ограничение на использование только текстовых данных, заключенных в теги, отбрасывало технологию в стек неиспользуемых по причине ограниченной функциональности. Это могут быть данные в форматах каких-то приложений, мультимедийные данные и т.п. Но в SOAP возможна передача данных нетекстовых типа, рассматриваемых в двоичном виде.

Данные в двоичном представлении включаются в сообщение в виде «вложения». В спецификации описаны правила включения SOAP-сообщения в MIME-сообщение типа multipart/related и правила пересылки его по протоколу HTTP.

Этот протокол определяет пересылку SOAP-сообщения внутри MIMEсообщения, состоящего из нескольких частей. 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).

* 1. Что такое graphql?

GraphQL — это язык запросов и манипулирования данными для API с открытым исходным кодом. Рядом преимуществ данной технологии является:

1. Позволяет клиенту точно указать, какие данные ему нужны.

2. Облегчает агрегацию данных из нескольких источников.

3. Использует систему типов для описания данных. Из недостатков стоит отметить сложности в ограничении доступа к данным с клиента.

* 1. Что такое Распознаватели (resolvers) в graphql?

Для получения данных используются распознаватели: для каждого типа данных создается свой распознаватель, который «знает», как получить объект этого типа.

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

Экосистема GraphQL растет. Речь не только об интеграциях с редакторами и IDE, связанных с сильно типизированной природой GraphQL; для GraphQL как такового появляются новые полноценные варианты применения. Например, можно припомнить Postman, применявшийся при работе с REST API, а теперь для этой же цели, но с GraphQL API применяется GraphiQL или GraphQL Playground. Также для вас найдутся различные библиотеки, например, Gatsby.js, генератор статических веб-сайтов для React, использующий GraphQL. Например, Gatsby.js позволяет написать движок для блога, наполняющий ваш блог контентом во время сборки через GraphQL API. Следовательно, у вас также будут CMS без клиентской части (напр., GraphCMS), предоставляющие контент (для блога) через GraphQL. API. Однако, в этой области развиваются не только технологические компоненты. Как грибы после дождя растут конференции, митапы и сообщества, посвященные GraphQL, также не составляет труда найти по нему новостные рассылки и подкасты.

Использование технологии GraphQL способна значительно уменьшить количество запросов, выполняемых со стороны клиента. Позволяет избежать денормализации данных, проводимой лишь ради того, чтобы оптимизировать работу механизмов построения пользовательского интерфейса.

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

Валидация или проверка достоверности модели – это проверка пользовательских данных. Эти данные можно (более того нужно) проверить дважды и более того при непосредственном вводе их пользователем или перед отправкой формы с данными на сервер (клиентская валидация) и после связывания уже на сервере с помощью специального объекта - валидатора (серверная валидация). Клиентская валидация — это принятая норма при написании клиентской части интернет-ресурса. Она реализуется с помощью JavaScript. Но её может быть недостаточно, поскольку JavaScript на клиенте может быть отключен по соображению безопасности. Поэтому прежде, чем сохранять получение данные на сервере в БД их лучше проверить еще раз.

* 1. Где и когда выполнять валидацию данных?

Валидацию данных можно и нужно выполнять в несколько этапов, усложняя проверки. Первоначально, по мере ввода, следим за тем, чтобы данные не содержали недопустимых символов. Например, для поля типа даты может быть запрещён ввод нецифровых символов. После того, как ввод завершен, можно проверить все значение целиком. Для введенной даты могут существовать какие-то ограничения, например, она не должна быть больше или меньше какихто определённых дат. Когда заполнены все поля, можно проверить, согласованы ли введенные значения друг с другом. Например, если в форме кроме поля для указания даты доставки есть поле для ввода адреса, приложение может проверить, что доставка по данному адресу может быть осуществлена в указанную дату. Если всё введено корректно, можно попытаться начать обработку, выполняя проверки по бизнес-логике приложения, и, если что-то пошло не так, выполнить откат к исходному состоянию.

* 1. Как выполнять валидацию данных?

Есть несколько типов:

1. Посимвольная проверка.

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.

2. Проверка отдельных значений.

Для пользовательского интерфейса это проверка значения в отдельном поле, причем 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. Такие проверки условно можно назвать «синтаксическими».

3. Совокупность входных значений.

Можно предположить, что в программу сначала передаются какие-то данные, после чего подается некоторый сигнал, который инициирует их обработку. В этот момент можно выполнить так называемые «семантические» проверки, нацеленные на валидацию не только отдельных значений, но и взаимосвязей между ними, взаимных ограничений.

4. Проверка состояния системы после обработки данных.

Это способ, к которому можно прибегнуть, если валидацию непосредственно входных данных выполнить не удается - можно попытаться их обработать, но оставить возможность вернуть всё к исходному состоянию. Такой механизм часто называется транзакционным.

* 1. Приведите пример с поэтапной валидацией данных.

Форма с заполнением личных данных пользователя. Проверяем, не ввел ли пользователь недопустимые символы, после ввода проверяем корректность ввода на логичность, после отправки пользователем сигнала проверяем взаимосвязь введенных данных.

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

Есть типы операций, такие как:

1. Запрос (query) – возвращает запрошенные клиентом данные.

2. Изменение (mutation) – производит манипуляции с данными и возвращает клиенту измененные данные.

3. Подписка (subscription) – при обновлении данных сервер выполняет определенный в подписке запрос и передает данные клиенту.

Важно понимать, что хотя технически любая операция может быть реализована так, чтобы перезаписать данные, по соглашению предполагается, что изменения должны быть отправлены явным образом через мутации. Здесь можно провести аналогию с REST, где предполагается, что GET-запросы не используются для изменения данных. Также между Query и Mutation есть еще одно очень важное различие: поля query выполняются параллельно, поля mutation – последовательно. Они описываются аналогично остальным типам, но при этом с них начинается выполнение каждого запроса.

1. Ссылка на удаленный репозиторий проекта

Полный код проекта можно найти по ссылке:

<https://github.com/IlyaShaparev/server>

ЗАКЛЮЧЕНИЕ

Были реализованы интерфейсы прикладного программирования для доступа к некоторым данным. Для тестирования всех функциональных возможностей созданного интерфейса была использована программа Postman.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Учебное пособие РСЧИР 2021-2022»: [Электронный ресурс]. URL: <https://online-edu.mirea.ru/mod/resource/view.php?id=406048> (дата обращения 23.10.2022)
2. «Руководство по PHP»: [Электронный ресурс]. URL: <https://www.php.net/manual/ru/index.php> (дата обращения 23.10.2022)
3. «Документация Docker»: [Электронный ресурс]. URL: <https://docs.docker.com/> (дата обращения 23.10.2022)