|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №3**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы: Разработка REST API на PHP**

**Студент группы** ИКБО-01-21 Васильев Виктор Владимирович

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

(подпись студента)

**Руководитель практической работы** ассистент Благирев М.М.

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

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

**ОГЛАВЛЕНИЕ**

[ОСНОВНАЯ ЧАСТЬ 3](#_Toc145360019)

[ВЫВОД 22](#_Toc145360020)

[ОТВЕТЫ НА ВОПРОСЫ 23](#_Toc145360021)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 24](#_Toc145360022)

# ОСНОВНАЯ ЧАСТЬ

**Цель работы:** научиться создавать REST API на языке программирования PHP.

**Ход работы:**

Для выполнения работы был использован Docker-образ “php:7.4.30-apache”, включающий в себя как среду выполнения для языка PHP, так и сервер Apache. На рисунке 1 показан Dockerfile, использованный для создания контейнера сервера.

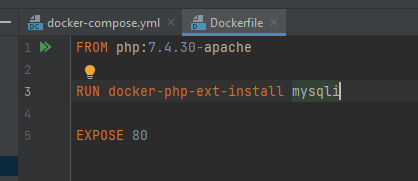


Рисунок 1 – Dockerfile сервера

На рисунке 2 показан файл docker-compose.yml, используемый для оркестрации контейнеров.

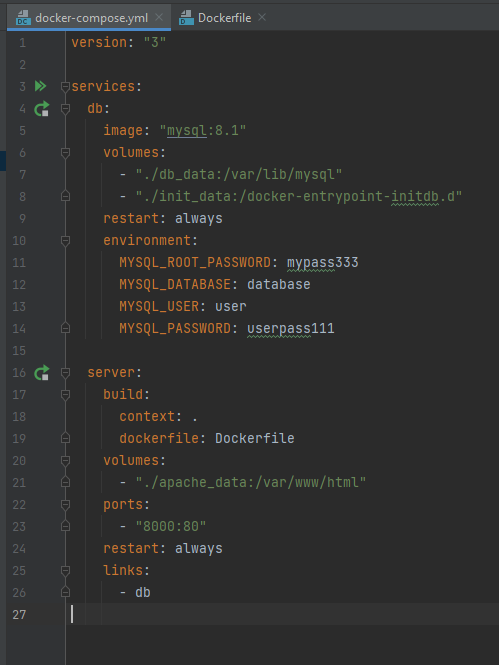


Рисунок 2 – Файл docker-compose.yml

Далее был создан файл common.php, хранящий в себе код подключения к БД и функцию для возврата HTTP статуса. Это было сделано для того, чтобы минимизировать повторение кода и подключать общий код при помощи команды require\_once. На рисунке 3 показан листинг данного файла.

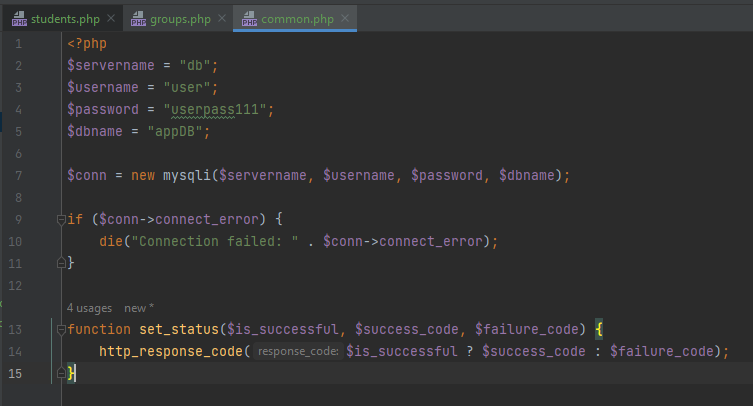


Рисунок 3 – Листинг кода common.php

Для реализации конечной точки взаимодействия с сущностью «студент» был создан файл students.php. В нём конструкция switch-case в зависимости от типа запроса передаёт полученные данные в один из методов. На рисунке 4 показана управляющая конструкция, а на рисунках 5-6 показан листинг методов, осуществляющих непосредственно изменения в базе данных.

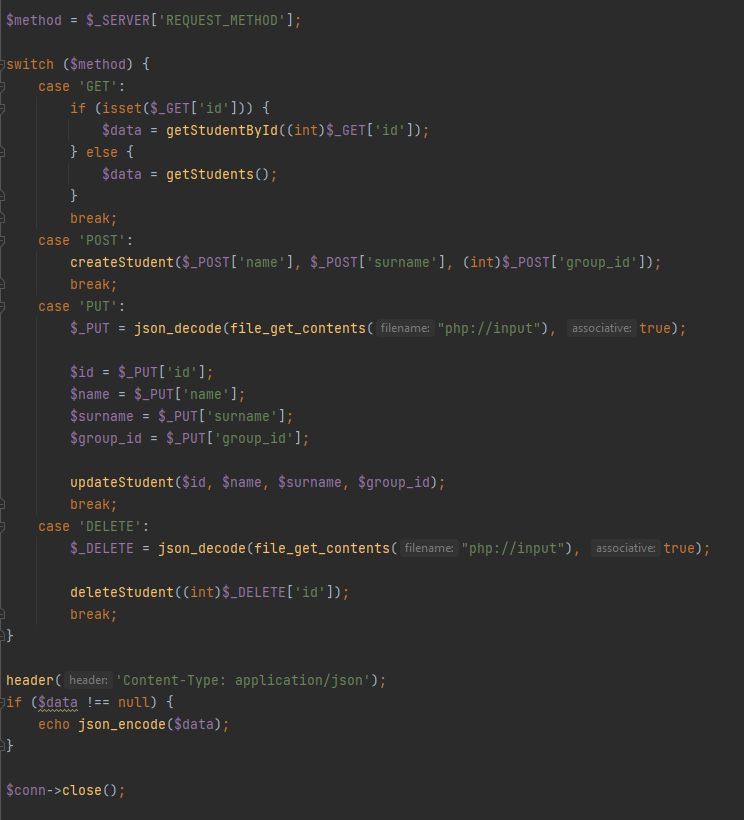


Рисунок 4 – Листинг кода управляющей конструкции

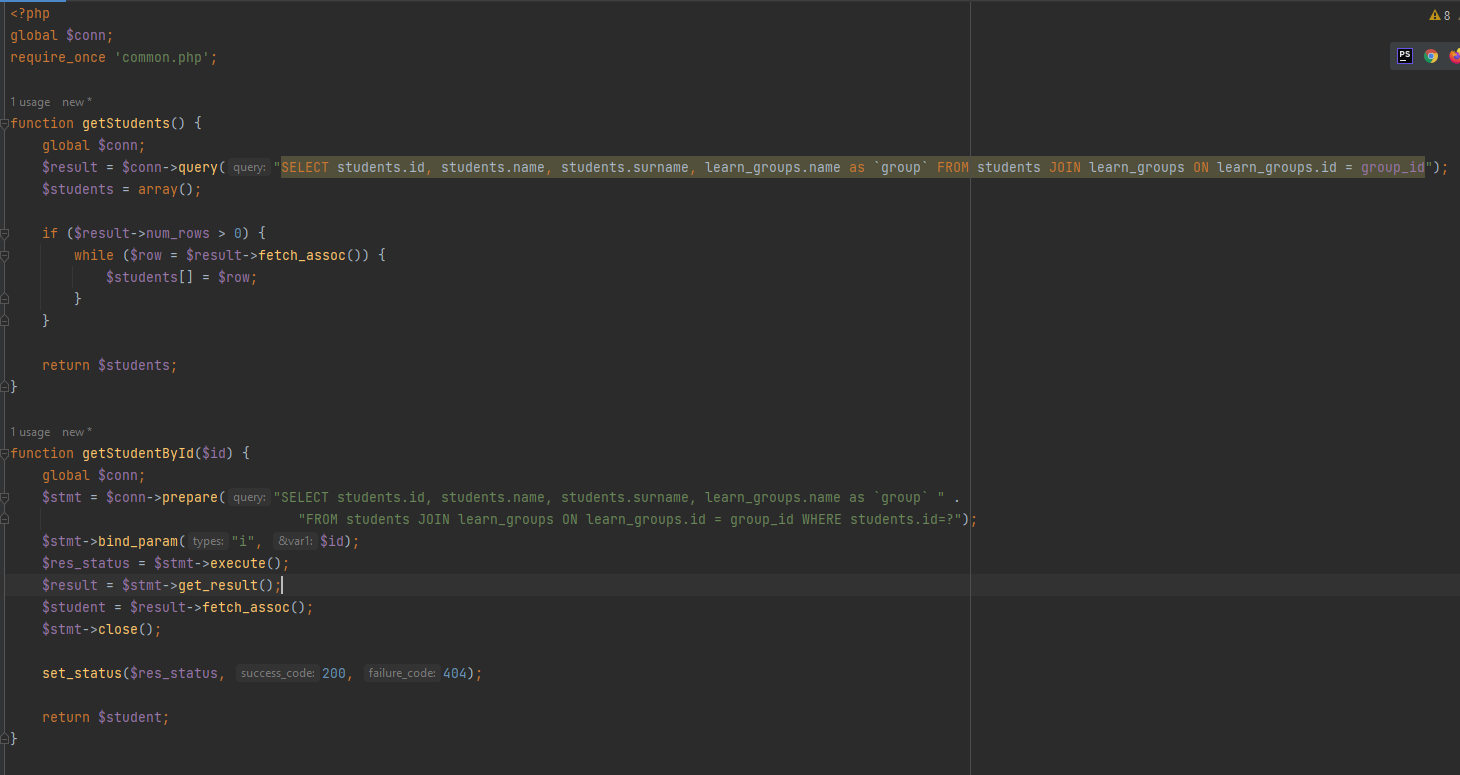


Рисунок 5 – Первый фрагмент листинга кода методов CRUD

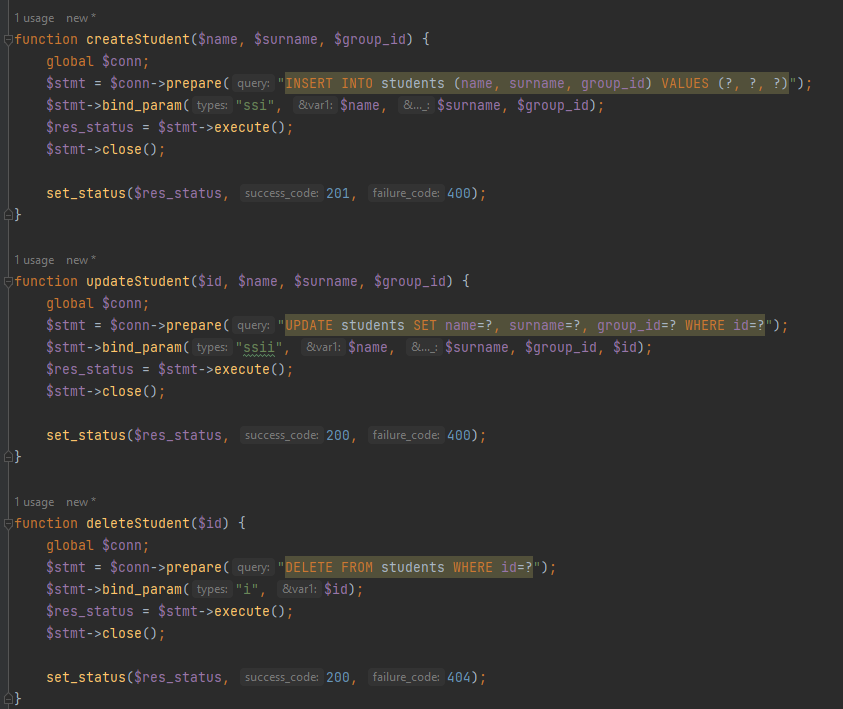


Рисунок 6 – Вторая часть листинга кода методов CRUD

Аналогично реализована конечная точка для учебных групп. На рисунках 7-9 показаны управляющая конструкция и методы CRUD для конечной точки groups.php.

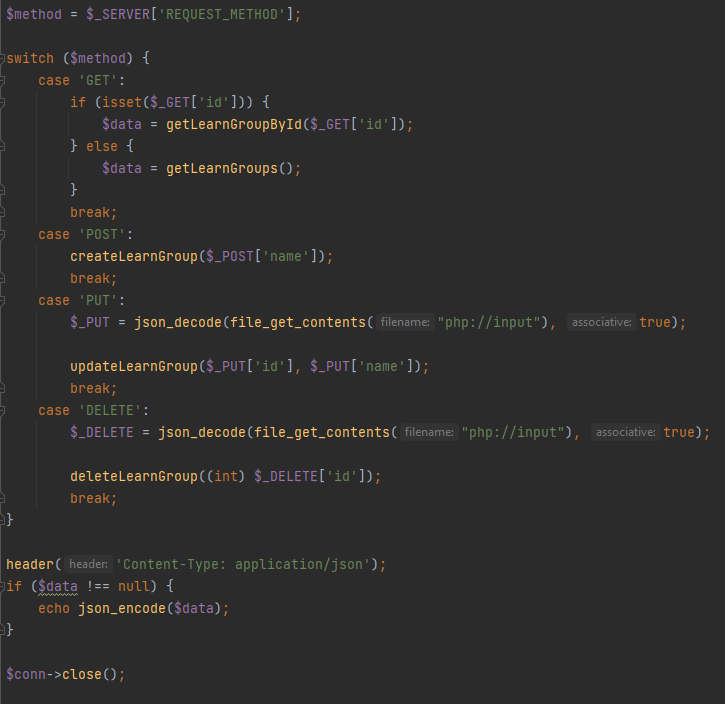


Рисунок 7 – Управляющая конструкция



Рисунок 8 – Первая часть листинга кода методов CRUD

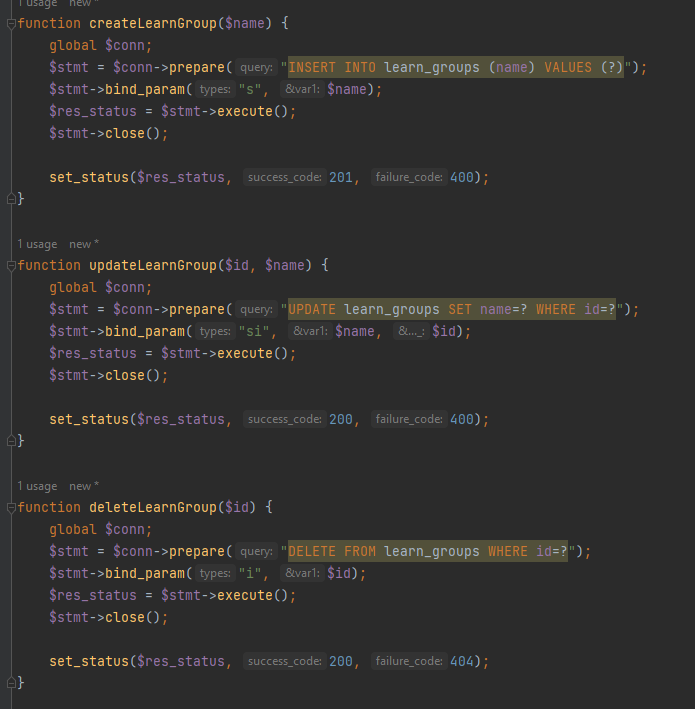


Рисунок 9 – Вторая часть листинга кода методов CRUD

На рисунках 10-13 показаны результаты тестирования всех GET-методов созданного REST API.

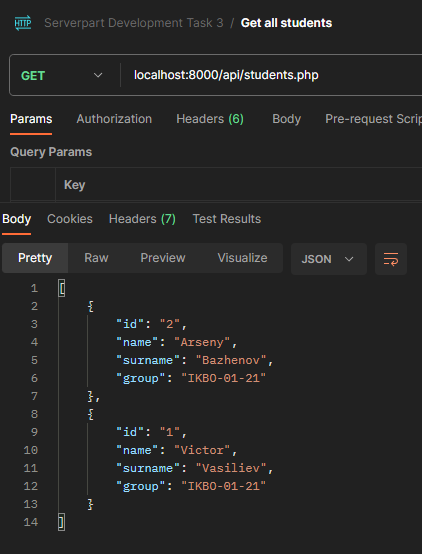


Рисунок 10 – Тестирование GET students.php

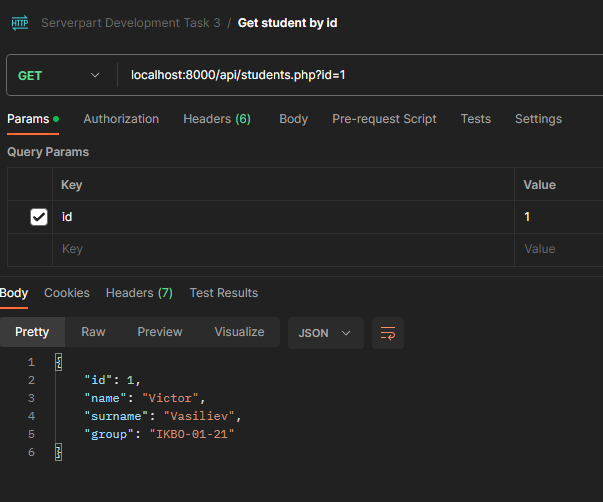


Рисунок 11 – Тестирование GET students.php?id=1

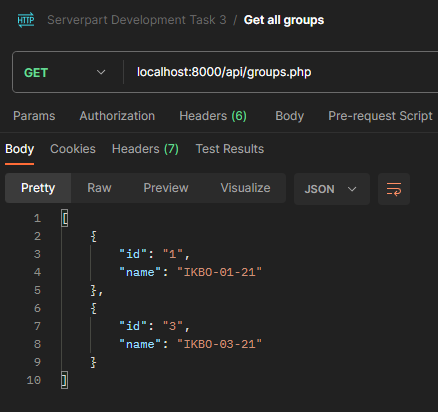


Рисунок 12 – Тестирование GET groups.php

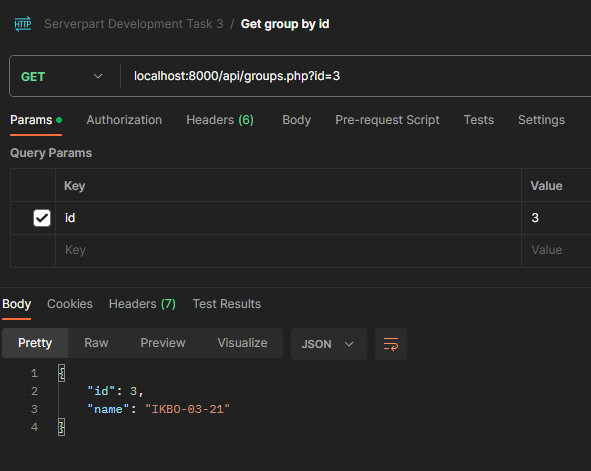


Рисунок 13 – Тестирование GET groups.php?id=3

На рисунках 14-15 показано тестирование POST-методов. Дополнительно на рисунках 16-17 показаны произошедшие после этого изменения.

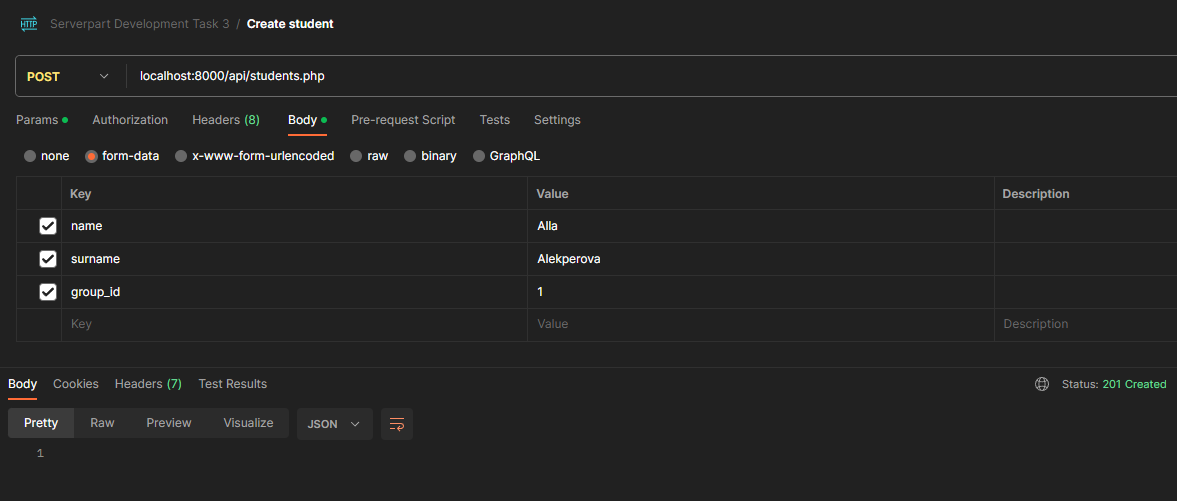


Рисунок 14 – Тестирование POST students.php

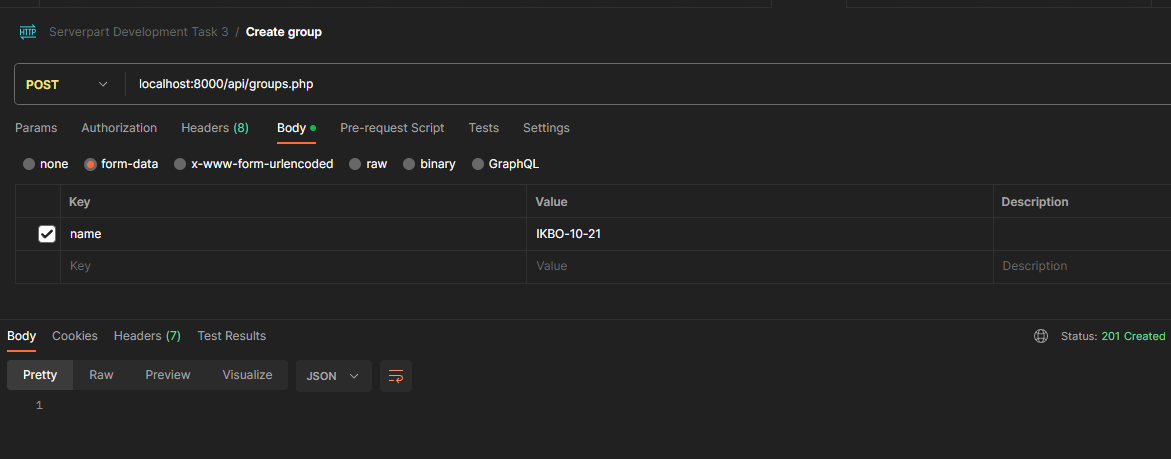


Рисунок 15 – Тестирование POST groups.php

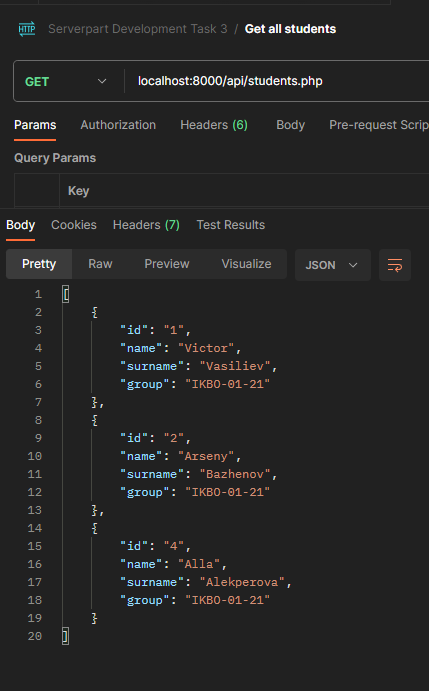


Рисунок 16 – Изменения, произошедшие после добавления студента

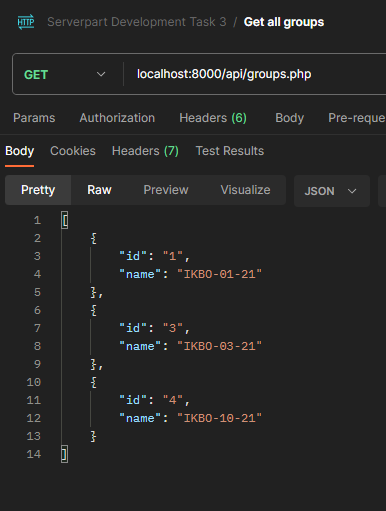


Рисунок 17 – Изменения, произошедшие после добавления группы

На рисунках 18-19 показаны PUT-запросы на изменение существующих студентов и групп. На рисунках 20-21 показаны произошедшие изменения.

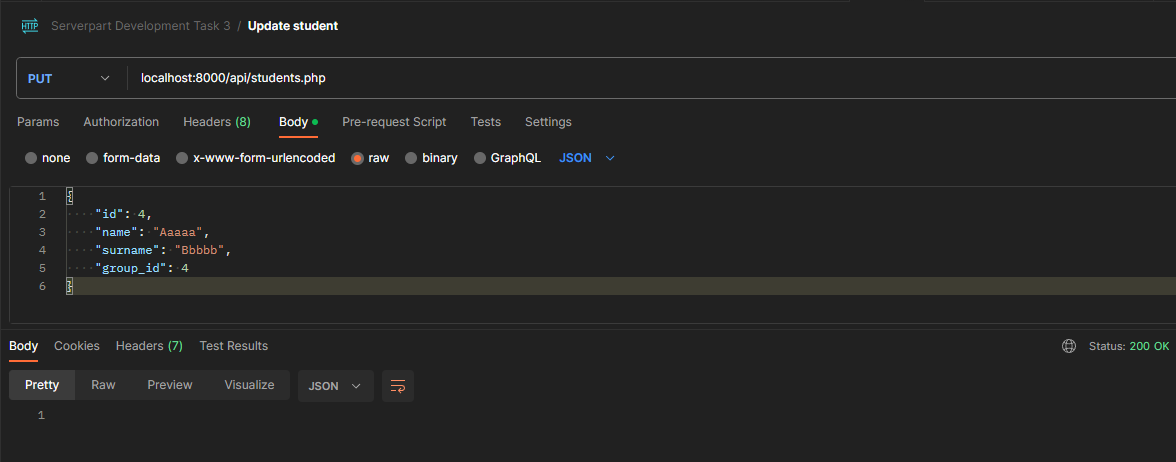


Рисунок 18 – Тестирование запроса PUT students.php

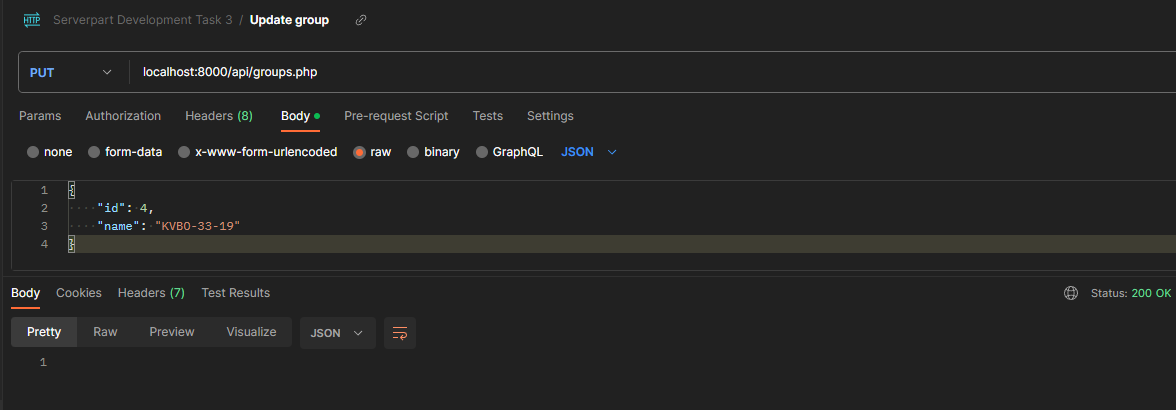


Рисунок 19 – Тестирование запроса PUT groups.php

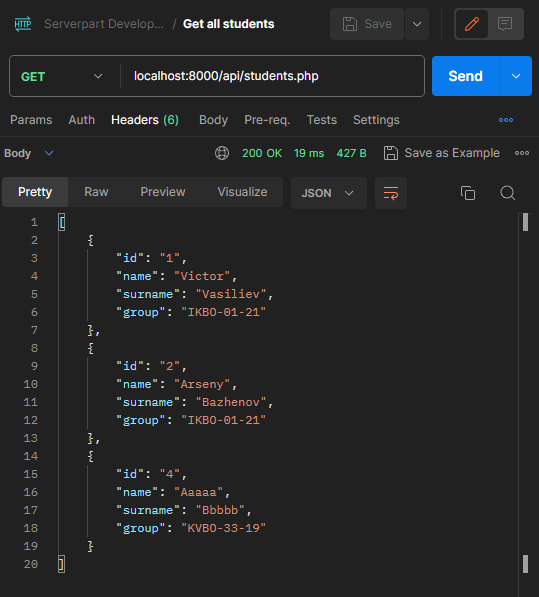


Рисунок 20 – Изменения, произошедшие после PUT-запроса

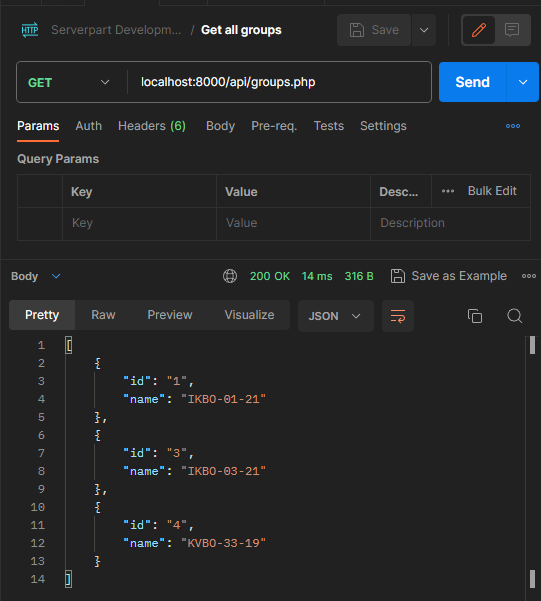


Рисунок 21 – Изменения, произошедшие после PUT-запроса

На рисунках 22-23 показано тестирование DELETE-запросов. На рисунках 24-25 отображены произошедшие изменения.

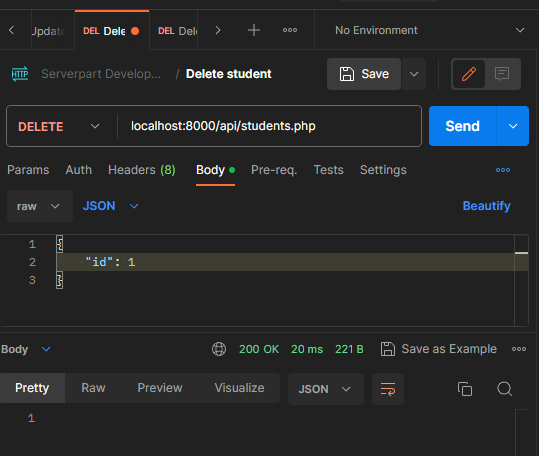


Рисунок 22 – Тестирование DELETE-запроса students.php

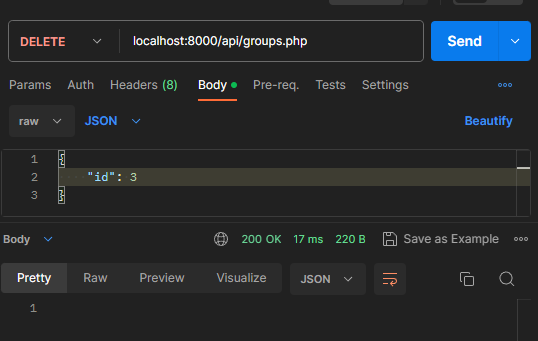


Рисунок 23 – Тестирование DELETE-запроса groups.php

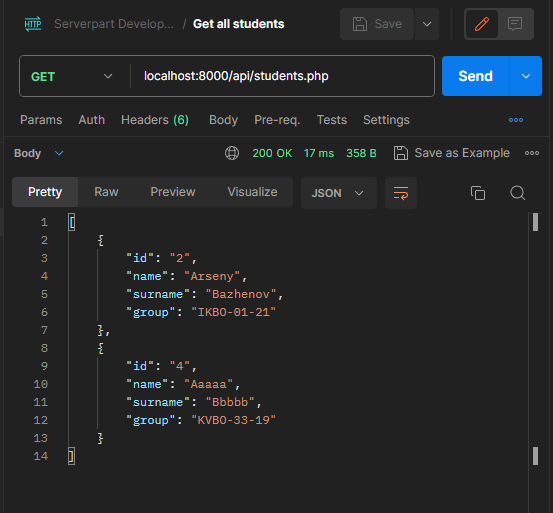


Рисунок 24 – Изменения, произошедшие после DELETE

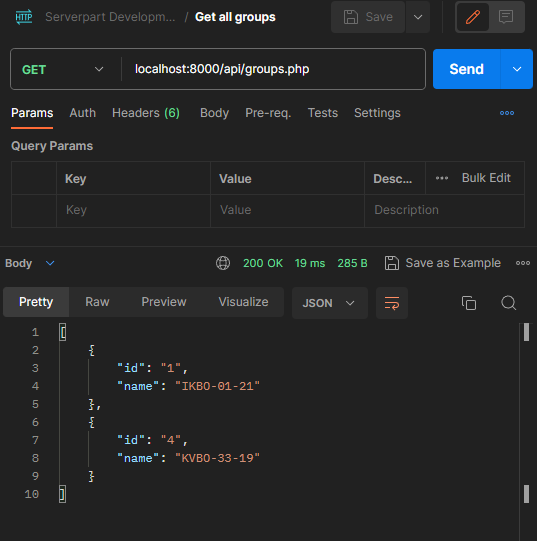


Рисунок 25 – Изменения, произошедшие после DELETE

Ссылка на репозиторий с файлами: <https://github.com/xxavierr404/client-server-arch-mirea/tree/main/pract3>

# ВЫВОД

В результате выполнения данной практической работы мной были получены навыки создания REST API на языке программирования PHP.

# ОТВЕТЫ НА ВОПРОСЫ

* Что такое сущность?

Сущности в REST API — это объекты, которые представляют данные, которые могут быть созданы, изменены, удалены или просмотрены в рамках системы API. Сущности могут быть представлены в различных форматах, таких как JSON или XML, и содержать различные поля, которые содержат информацию о сущности.

* Какие методы существуют в протоколе HTTP?

В протоколе HTTP существуют методы GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH.

* Что такое контроллер?

Контроллер в REST API — это часть приложения, которая обрабатывает запросы, поступающие от клиента, и возвращает нужные данные. В методах контроллера определяются действия, которые нужно выполнить, чтобы вернуть данные. Контроллеры обычно работают с моделями данных и сервисами, чтобы получить необходимые данные.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Простой RESTful-сервис на нативном PHP. [Электронный ресурс] URL: <https://webdevkin.ru/posts/backend/restful-servis-na-nativnom-php>
2. PHP documentation: http\_response\_code [Электронный ресурс] URL: <https://www.php.net/manual/en/function.http-response-code.php>