|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра Инструментального и  прикладного программного обеспечения (ИиППО) |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6** | |
| **по дисциплине** |  |
| **«Архитектура клиент-серверных приложений»**  **Тема:**  **Введение в GraphQL API и графовые модели** | |
| Выполнил студент группы ИКБО-20-19 | Московка А.А. |
| Принял ассистент кафедры ИиППО | Степанов П.В. |

|  |  |  |
| --- | --- | --- |
| Практическая работы выполнены | «\_\_»\_\_\_\_\_\_\_2021г. |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2021 г. |  |

Москва 2021

**Постановка задачи**

Целью данной практической работы является знакомство обучающихся с набирающим популярность современным подходом к проектированию и реализации API на основе графовых моделей и с реализующей данный подход технологией на основе спецификации GraphQL.

## Ход работы

Спецификация:

1. метод GET - localhost:4545/graphql.php получить данные из бд
2. метод POST localhost:4545/graphql.php - добавить данные в бд
3. метод DELETE localhost:4545/graphql.php - удалить данные из бд
4. метод PATCH localhost:4545/graphql.php - обновить данные в бд

**Программный код**

**src/graphql.php:**

<?php  
require\_once *\_\_DIR\_\_* . '/vendor/autoload.php';  
require\_once 'Bd/Types.php';  
require\_once 'Bd/DB.php';  
use Bd\Types;  
use Bd\DB;  
use GraphQL\GraphQL;  
use GraphQL\Type\Definition\ObjectType;  
use GraphQL\Type\Schema;  
try {  
 $config = [  
 'host' => 'bd',  
 'database' => 'appDB',  
 'username' => 'root',  
 'password' => 'qwerty'  
 ];  
 DB::init($config);   
 $mutationType = new ObjectType([  
 'name' => 'Mutation',  
 'fields' => [  
 'sum' => [  
 'type' => Types::int(),  
 'args' => [  
 'x' => ['type' => Types::int()],  
 'y' => ['type' => Types::int()],  
 ],  
 'resolve' => static function ($calc, array $args): int {  
 return $args['x'] + $args['y'];  
 },  
 ],  
 ],  
 ]);  
 $schema = new Schema([  
 'query' => Types::query(),  
 'mutation' => $mutationType,  
 ]);  
 $rawInput = file\_get\_contents('php://input');  
 $input = json\_decode($rawInput, *true*);  
 $query = $input['query'];  
 $variableValues = $input['variables'] ?? *null*;  
 $rootValue = ['prefix' => 'You said: '];  
 $result = GraphQL::executeQuery($schema, $query, $rootValue, *null*, $variableValues);  
 $output = $result->toArray();  
} catch (Throwable $e) {  
 $output = [  
 'error' => [  
 'message' => $e->getMessage(),  
 ],  
 ];  
}  
header('Content-Type: application/json; charset=UTF-8');  
echo json\_encode($output);

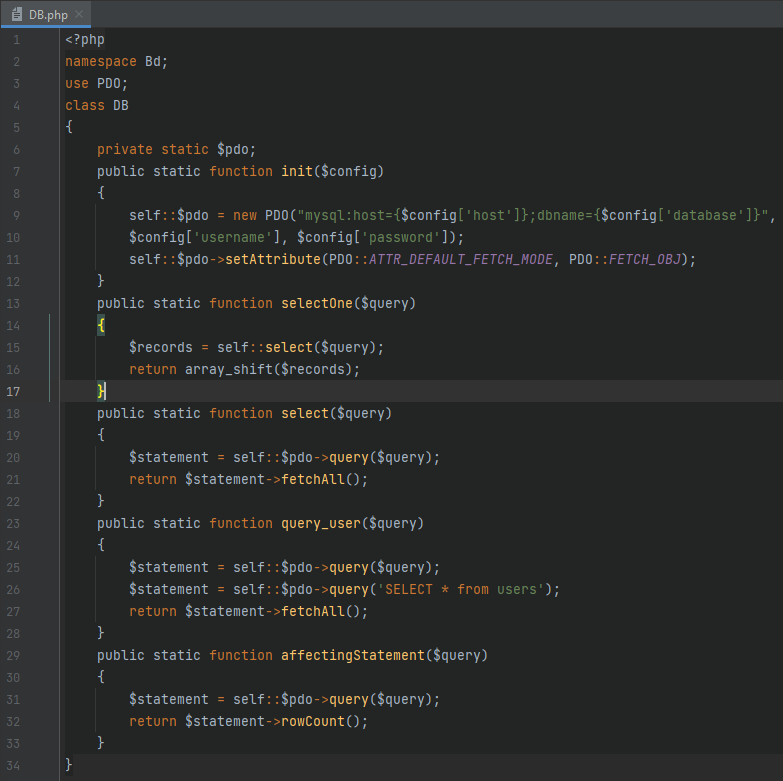


Рис. 1 - Код работы с БД

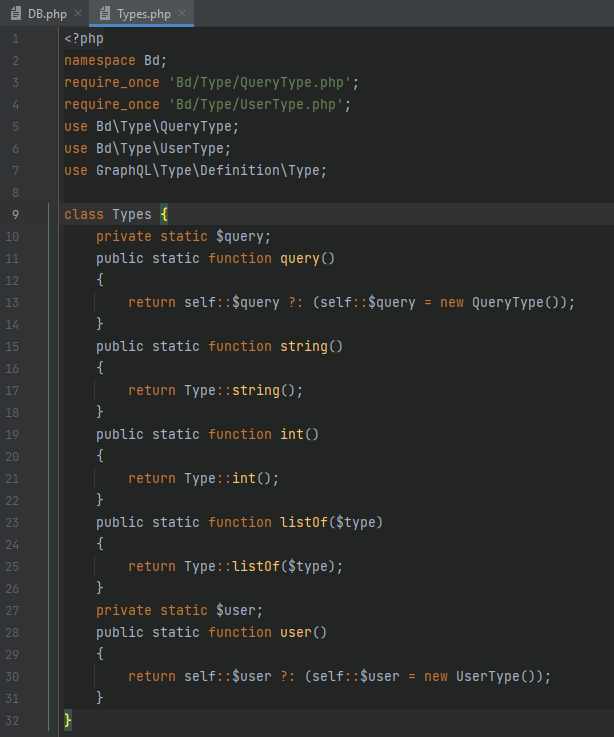


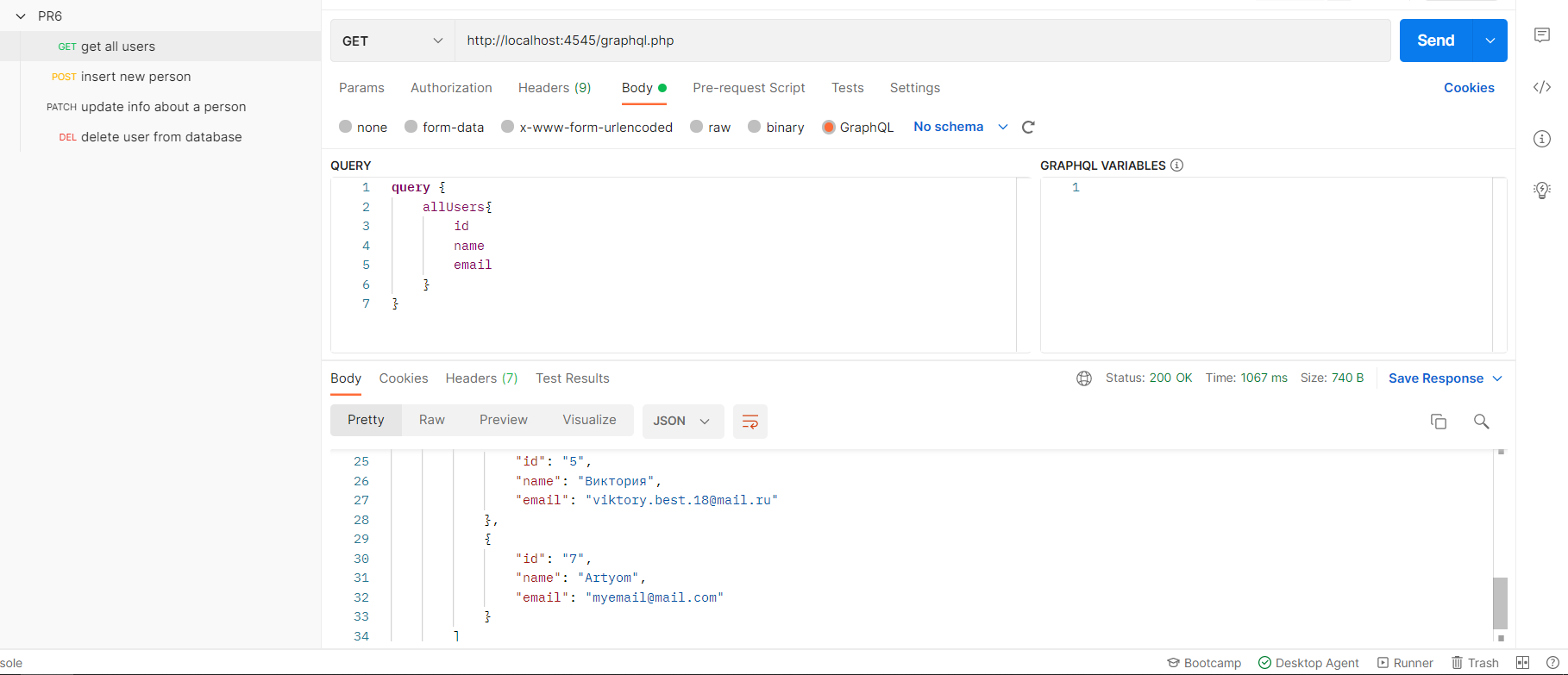
Рис. 2 ‒ Код работы с данными



Рис. 3 ‒ Код работы с пользователем

**src/Bd/Type/QueryType.php:**

<?php  
  
namespace Bd\Type;  
  
require\_once 'Bd/DB.php';  
require\_once 'Bd/Types.php';  
  
use Bd\DB;  
use Bd\Types;  
use GraphQL\Type\Definition\ObjectType;  
  
class QueryType extends ObjectType  
{  
 public function \_\_construct()  
 {  
 $config = [  
 'fields' => function() {  
 return [  
 'user' => [  
 'type' => Types::user(),  
 'description' => 'Возвращает пользователя по id',  
 'args' => [  
 'id' => Types::int()  
 ],  
 'resolve' => function ($root, $args) {  
 if($\_SERVER['REQUEST\_METHOD'] == 'GET')  
 {  
 return DB::selectOne("SELECT \* from users WHERE id = {$args['id']}");  
 }  
 echo 'Only GET method!';  
 }  
 ],  
 'allUsers' => [  
 'type' => Types::listOf(Types::user()),  
 'description' => 'Список пользователей',  
 'resolve' => function () {  
 if($\_SERVER['REQUEST\_METHOD'] == 'GET')  
 {  
 return DB::select('SELECT \* from users');  
 }  
 echo 'Only GET method!';  
 }  
 ],  
 'Post' => [  
 'type' => Types::listOf(Types::user()),  
 'description' => 'Добавление пользователей',  
 'args' => [  
 'id' => ['type' => Types::int()],  
 'name' => ['type' => Types::string()],  
 'email' => ['type' => Types::string()]  
 ],  
 'resolve' => function ($uzer, array $args) {  
 $id = $args['id'];  
 $name = $args['name'] . '';  
 $email = $args['email'] . '';  
 if($\_SERVER['REQUEST\_METHOD'] == 'POST')  
 {  
 return DB::query\_user("INSERT INTO users (id, name, email) VALUES ({$id}, '{$name}', '{$email}')");  
 }  
 echo 'Only POST method!';  
 }  
 ],  
 'Delete' => [  
 'type' => Types::listOf(Types::user()),  
 'description' => 'Добавление пользователей',  
 'args' => [  
 'id' => ['type' => Types::int()]  
 ],  
 'resolve' => function ($uzer, array $args) {  
 $id = $args['id'];  
 if($\_SERVER['REQUEST\_METHOD'] == 'DELETE')  
 {  
 return DB::query\_user("DELETE FROM users WHERE id = {$id}");  
 }  
 echo 'Only DELETE method!';  
 }  
 ],  
 'Patch' => [  
 'type' => Types::listOf(Types::user()),  
 'description' => 'Добавление пользователей',  
 'args' => [  
 'old\_id' => ['type' => Types::int()],  
 'new\_id' => ['type' => Types::int()],  
 'name' => ['type' => Types::string()],  
 'email' => ['type' => Types::string()]  
 ],  
 'resolve' => function ($uzer, array $args) {  
 $old\_id = $args['old\_id'];  
 $new\_id = $args['new\_id'];  
 $name = $args['name'] . '';  
 $email = $args['email'] . '';  
 if($\_SERVER['REQUEST\_METHOD'] == 'PATCH')  
 {  
 return DB::query\_user("UPDATE users SET id={$new\_id}, name='{$name}', email='{$email}' WHERE id={$old\_id};");  
 }  
 echo 'Only PATCH method!';  
 }  
 ]  
 ];  
 }  
 ];  
 parent::\_\_construct($config);  
 }  
}

Рис.4 ‒ Проверка GET запроса

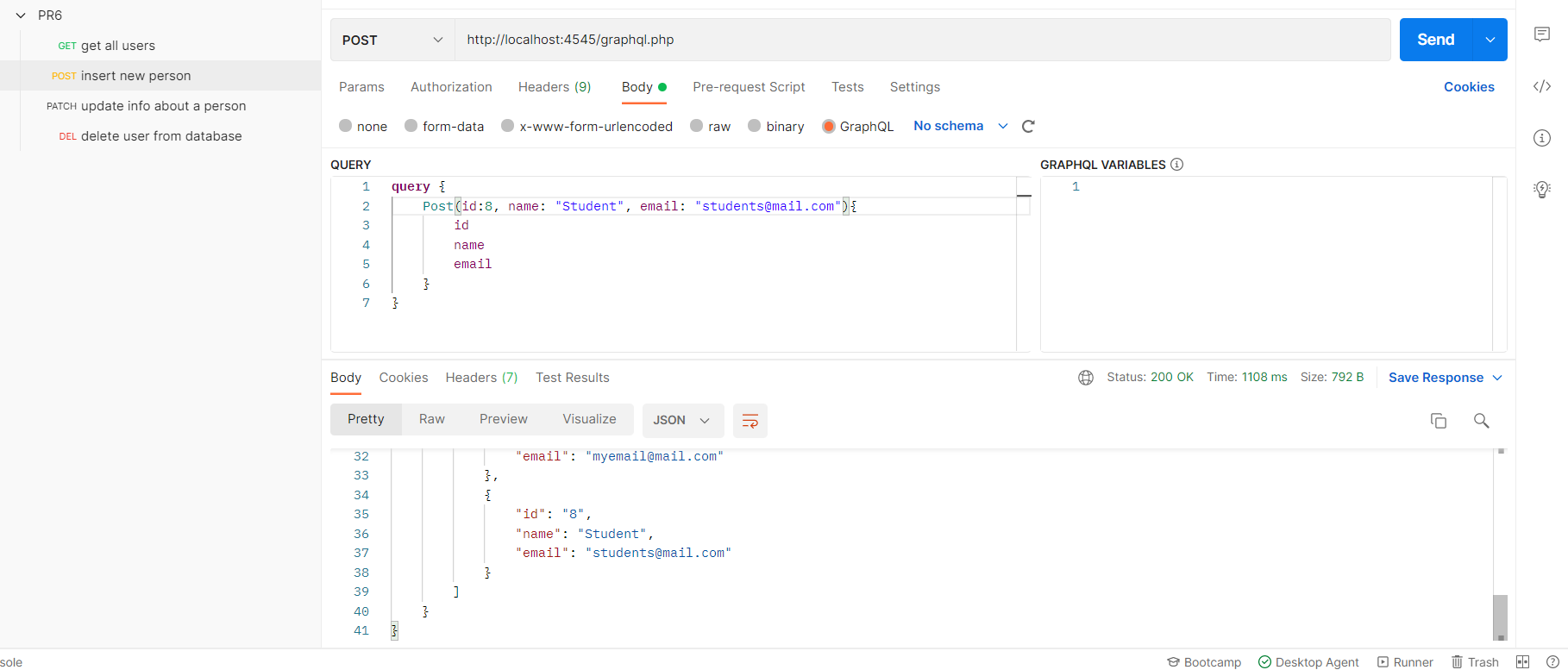


Рис.5 ‒ Проверка POST запроса

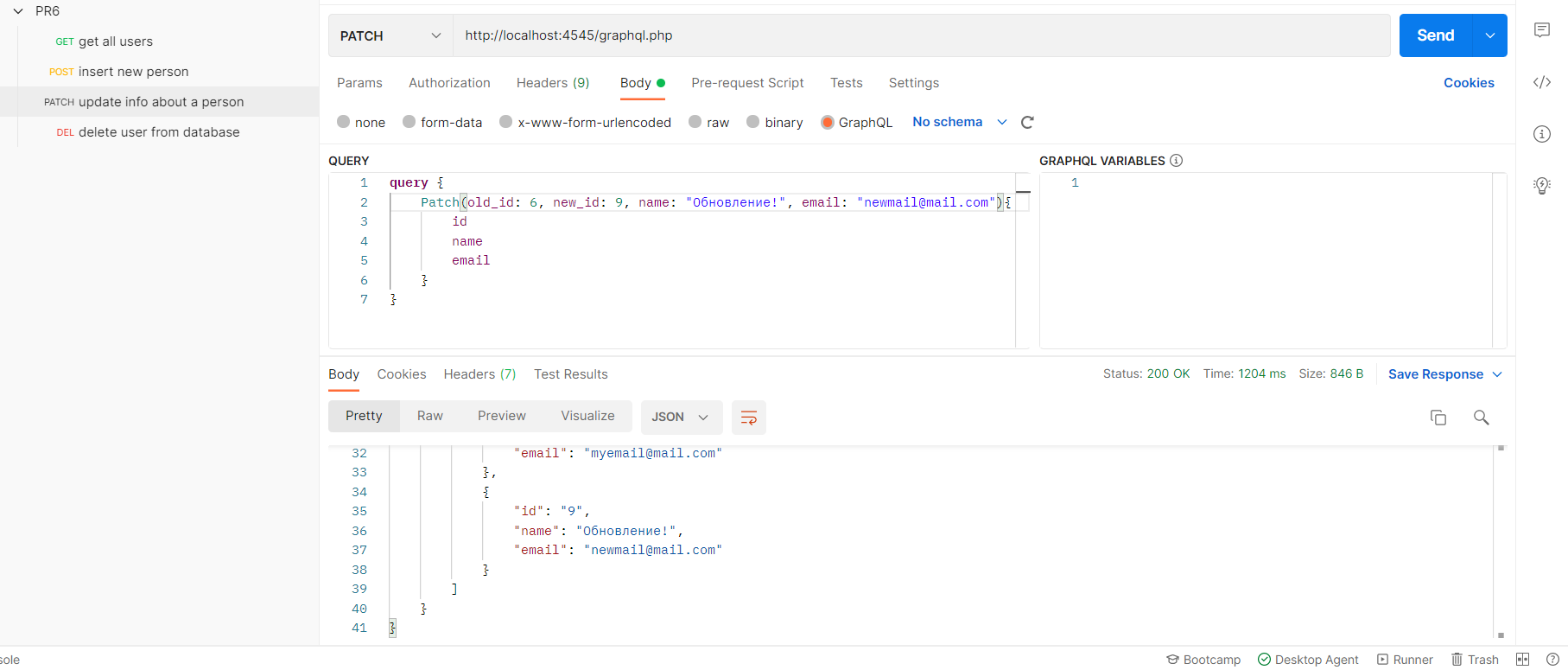


Рис.6 ‒ Проверка PATCH запроса

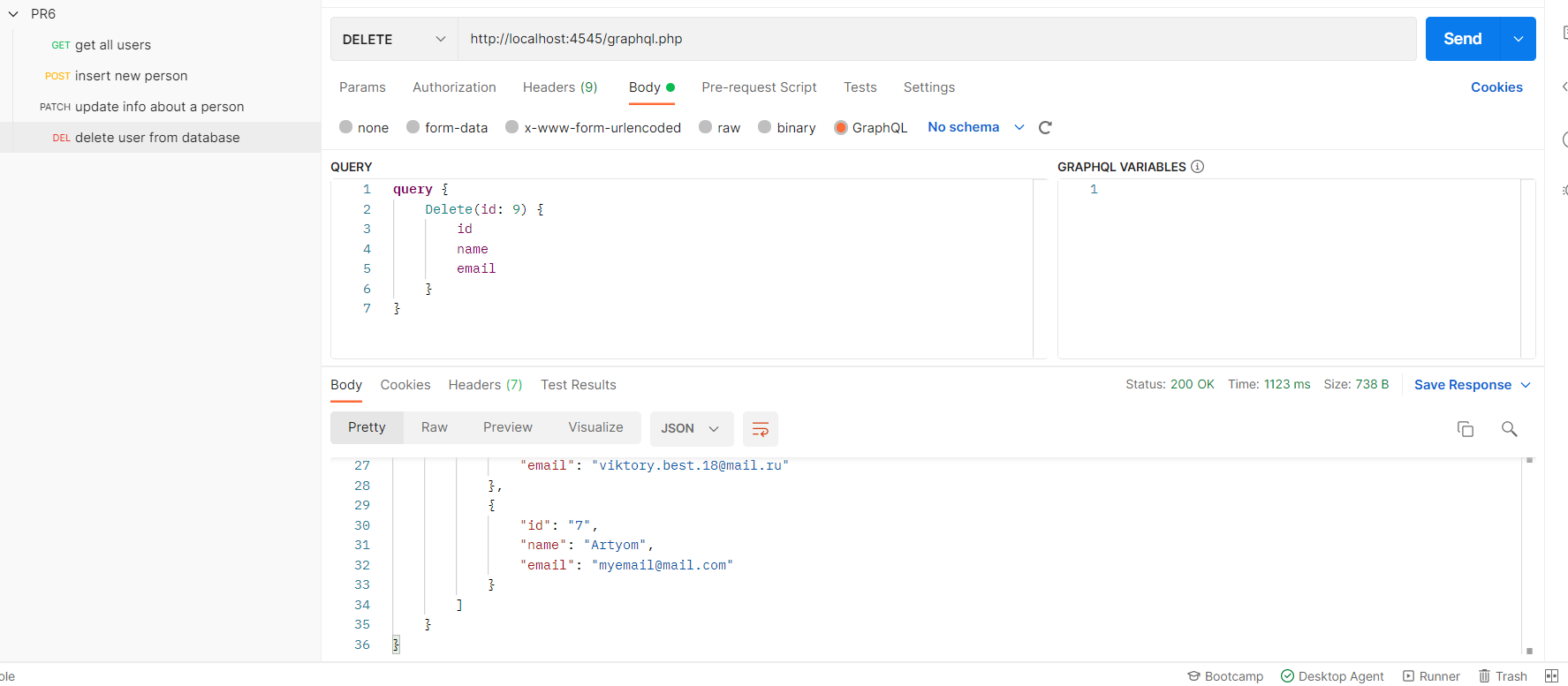


Рис.7 ‒ Проверка DELETE запроса

**Вывод**

В результате выполнения данной практической работы были получены и закреплены практические навыки работы с клиент-серверной архитектурой и было произведено ознакомление с языком запросов и манипулирования данными GraphQL API с применением сервиса по отправке HTTP-запросов Postman.

## Список использованной литературы:

1. Конспект лекций по дисциплине «Разработка сервисных частей интернет ресурсов», РТУ МИРЭА, лектор –преподаватель Куликов А.А.

2. Docker – URL: https://www.docker.com/ (Дата обращения – 17.11.2021)

3. Настройка стека ламп в контейнерах Docker– URL: https://www.centlinux.com/2020/03/configure-lamp-stack-in-docker-containers.html (Дата обращения – 17.11.2021)

4. Полное практическое руководство Docker – URL: https://habr.com/ru/post/310460/ (Дата обращения – 17.11.2021)

5. Руководство по GraphQL – URL: https://habr.com/ru/post/326986/ (Дата обращения – 17.11.2021)