|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического Обеспечения и Стандартизации Информационных Технологий

**ПРАКТИЧЕСКАЯ РАБОТА №4**

по дисциплине «Интерфейсы прикладного программирования»

**Студент группы** ИКБО-01-19 Кузин Д.О.

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

(подпись студента)

**Руководитель практической работы** Коновалова С.В.

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

(подпись руководителя)

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

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

Москва 2021

# Цель

# Целью данной практической работы является знакомство с принципами

# REST и создание простейшего REST API.

# Ход работы

1. В процессе работы реализованы 6 ендпойнтом:

* /user/add. При обращении к нему возвращаются данные добавленного пользователя (POST-запрос)
* /user/get. При обращении к нему возвращаются данные пользователя (GET-запрос)
* /user/edit. При обращении к нему возвращаются данные измененного пользователя (PUT-запрос)
* /user/delete. При обращении к нему возвращается информация об успехе удаления пользователя из БД (DELETE-запрос)
* /joke/add. При обращении к нему возвращаются данные добавленной шутки и её автора (POST-запрос)
* /joke/get. При обращении к нему возвращаются данные шутки и её автора (GET-запрос)

1. Запросы, выполненные для тестирования API представлены на рисунках

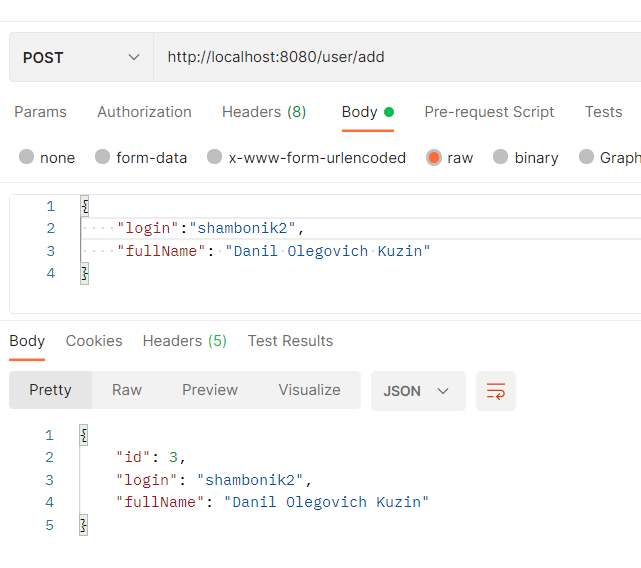


Рисунок 1 – POST-запрос к эндпоинту /user/add

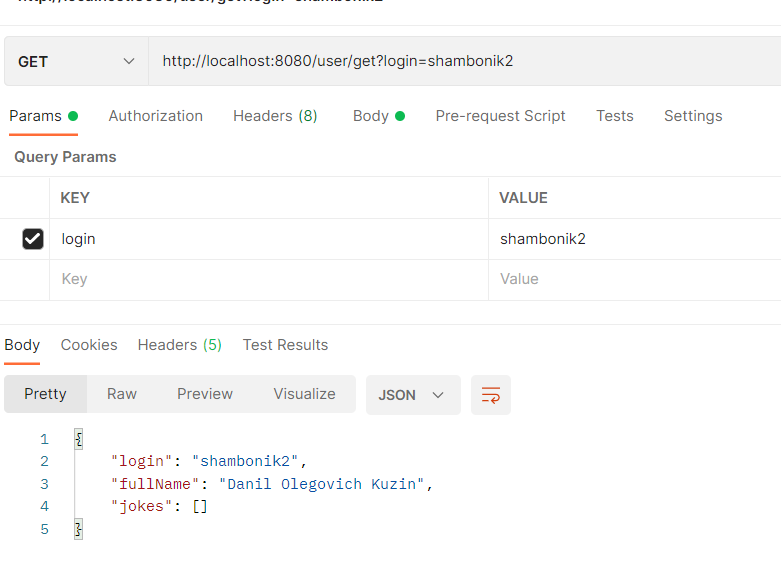


Рисунок 2 – GET-запрос к эндпоинту /user/get

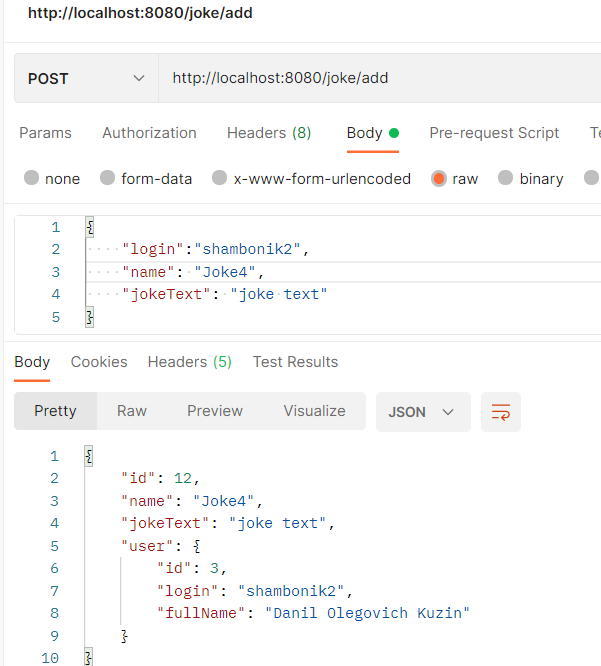


Рисунок 3 – POST-запрос к эндпоинту /joke/add

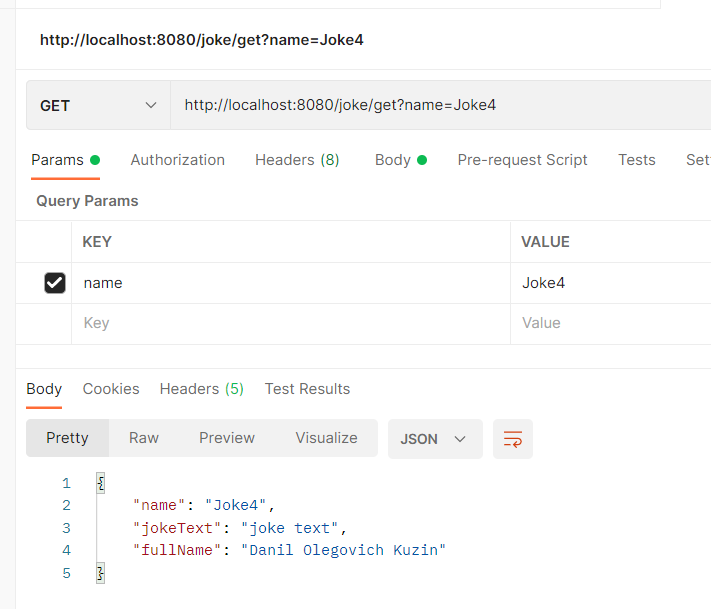


Рисунок 4 – GET-запрос к эндпоинту /joke/get

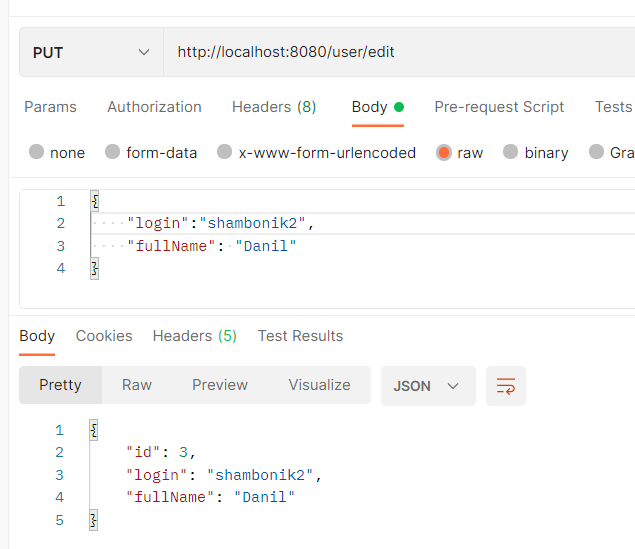


Рисунок 5 – PUT-запрос к эндпоинту /user/edit

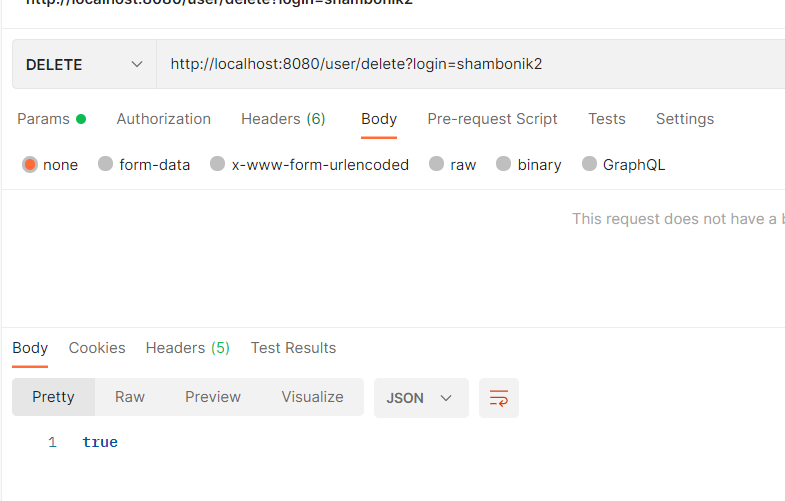


Рисунок 6 – DELETE-запрос к ендпойнту /user/delete

1. В процессе работы были созданы классы-контроллеры (Листинги 1-2).

Листинг 1 – JokeController

@RestController  
@RequestMapping("/joke")  
@RequiredArgsConstructor  
public class JokeController {  
  
 private final JokeService jokeService;  
  
 @GetMapping("/get")  
 public GetJokeDto getJoke(@RequestParam("name") String name) {  
 return jokeService.getJoke(name);  
 }  
  
 @PostMapping("/add")  
 public Joke addJoke(@RequestBody AddJokeDto dto) {  
 return jokeService.addJoke(dto);  
 }  
}

Листинг 2 – UserController

@RestController  
@RequestMapping("/user")  
@RequiredArgsConstructor  
public class UserController {  
  
 private final UserService userService;  
  
 @GetMapping("/get")  
 public GetUserDto getUser(@RequestParam("login") String login) {  
 return userService.getUser(login);  
 }  
  
 @PostMapping("/add")  
 public User addUser(@RequestBody AddUserDto dto) {  
 return userService.addUser(dto);  
 }  
  
 @PutMapping("/edit")  
 public User editUser(@RequestBody EditUserDto dto){  
 return userService.editUser(dto);  
 }  
  
 @DeleteMapping("/delete")  
 public Boolean deleteUser(@RequestParam("login") String login){  
 return userService.deleteUser(login);  
 }  
}

Классы-сервисы (Листинги 3-4).

Листинг 3 – JokeService

@Service  
@RequiredArgsConstructor  
public class JokeService {  
  
 private final JokeRepo jokeRepo;  
 private final UserRepo userRepo;  
  
  
 public GetJokeDto getJoke(String name){  
 Joke joke = jokeRepo.findByName(name);  
 if(joke!=null) {  
 if(joke.getUser() == null){  
 return new GetJokeDto(name, joke.getJokeText(), null);  
 }  
 return new GetJokeDto(name, joke.getJokeText(), joke.getUser().getFullName());  
 }  
 return null;  
 }  
  
 public Joke addJoke(AddJokeDto dto){  
 Joke joke = new Joke();  
 joke.setJokeText(dto.getJokeText());  
 joke.setName(dto.getName());  
 User user = userRepo.findByLogin(dto.getLogin());  
 joke.setUser(user);  
 return jokeRepo.save(joke);  
 }  
}

Листинг 4 – UserService

@Service  
@RequiredArgsConstructor  
public class UserService {  
  
 private final JokeRepo jokeRepo;  
 private final UserRepo userRepo;  
  
 public GetUserDto getUser(String login){  
 User user = userRepo.findByLogin(login);  
 List<Joke> jokes = jokeRepo.findAllByUser(user);  
 if(user!=null) {  
 return new GetUserDto(login, user.getFullName(), jokes.stream()  
 .map(joke -> new GetJokeDto(joke.getName(), joke.getJokeText(), user.getFullName()))  
 .collect(Collectors.*toList*())  
 );  
 }  
 return null;  
 }  
  
 public User addUser(AddUserDto dto){  
 User user = new User();  
 user.setJokes(new HashSet<>());  
 user.setLogin(dto.getLogin());  
 user.setFullName(dto.getFullName());  
 return userRepo.save(user);  
 }  
  
 public User editUser(EditUserDto dto){  
 User user = userRepo.findByLogin(dto.getLogin());  
 if(user!=null){  
 user.setFullName(dto.getFullName());  
 return userRepo.save(user);  
 }  
 return null;  
 }  
  
 public Boolean deleteUser(String login){  
 User user = userRepo.findByLogin(login);  
 if(user==null){  
 return false;  
 }  
 List<Joke> jokes = jokeRepo.findAllByUser(user);  
 for(Joke joke : jokes){  
 jokeRepo.delete(joke);  
 }  
 userRepo.delete(user);  
 return true;  
 }

Также были созданы классы-сущности, DTO-классы и JPA репозитории

# Вывод

В процессе работы были закреплены знания о принципах REST, а также был создан простейший REST API.