ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | А. В. Фомин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| РАЗРАБОТКА REST API |
| по курсу: СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4232М |  |  |  | В. Ф. Губайдулин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**Цель** **работы:**

Изучение representation state transfer (REST) подхода к построению API для Web приложений.

**Вариант:**

Разрабатываемое приложение ­– автоматизированная система управления составом футбольной команды.

Система будет позволять работать со списком команд, с игроками, играющими за определённую команду, а также с формациями – схемами игры.

**Ход работы:**

1) Был разработан REST API интерфейс в виде набора URI и методов вызова.

Таблица 1 – URI и методы

|  |  |
| --- | --- |
| **URI** | **Метод** |
| GET: /api/teams | Iterable<TeamsDto> getAllTeams() |
| POST: /api/teams | ResponseEntity<Teams> addNewTeam(@RequestBody TeamsDto dto) |
| DEL: /api/teams/{id} | ResponseEntity deleteTeamById(@PathVariable Integer id) |
| GET: /api/positions | Iterable<PositionListDto> getAllPositions() |
| PUT: /api/players/ | ResponseEntity updatePlayerData(@RequestBody PlayersDto dto) |
| POST: /api/players/ | ResponseEntity<Players> addPlayerData(@RequestBody PlayersDto dto) |

2) Была реализована back-end часть REST API.

Листинг 1 ­– Реализация контроллера «Команда»

@RestController

@CrossOrigin(origins = "http://localhost:3000", allowedHeaders = "\*", exposedHeaders = "\*")

@RequestMapping("/api/teams")

public class TeamsController {

private final ITeamsService \_teamsService;

public TeamsController (ITeamsService teamsService) {

\_teamsService = teamsService;

}

@GetMapping

public Iterable<TeamsDto> getAllTeams() {

return \_teamsService.getAllTeams();

}

@PostMapping()

public ResponseEntity<Teams> addNewTeam(@RequestBody TeamsDto dto) {

try {

Teams teams = \_teamsService.AddNewTeam(dto);

return new ResponseEntity<Teams>(HttpStatus.OK);

} catch (Error e) {

return new ResponseEntity<Teams>(HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

@DeleteMapping("/{id}")

public ResponseEntity deleteTeamById(@PathVariable Integer id) {

try {

\_teamsService.DeleteTeamById(id);

return new ResponseEntity<>(HttpStatus.OK);

} catch (EmptyResultDataAccessException e) {

ErrorApiResponse error = new ErrorApiResponse();

error.setErrorMsg(e.getMessage());

return new ResponseEntity<>(error, HttpStatus.NOT\_FOUND);

}

}

}

Листинг 2 – Сервис «Команды»

@Service

public class TeamsService implements ITeamsService {

private final ITeamsRepos \_teamsRepos;

public TeamsService(ITeamsRepos teamsRepos) {

this.\_teamsRepos = teamsRepos;

}

public Iterable<TeamsDto> getAllTeams() {

return TeamsMapper.toDtoIterable(\_teamsRepos.findAll());

}

public Teams AddNewTeam(TeamsDto dto) {

Teams team = TeamsMapper.toModel(dto);

return \_teamsRepos.save(team);

}

public void DeleteTeamById(Integer id) {

\_teamsRepos.deleteById(id);

return;

}

}

3) Разработанный REST API протестирован с помощью Postman.

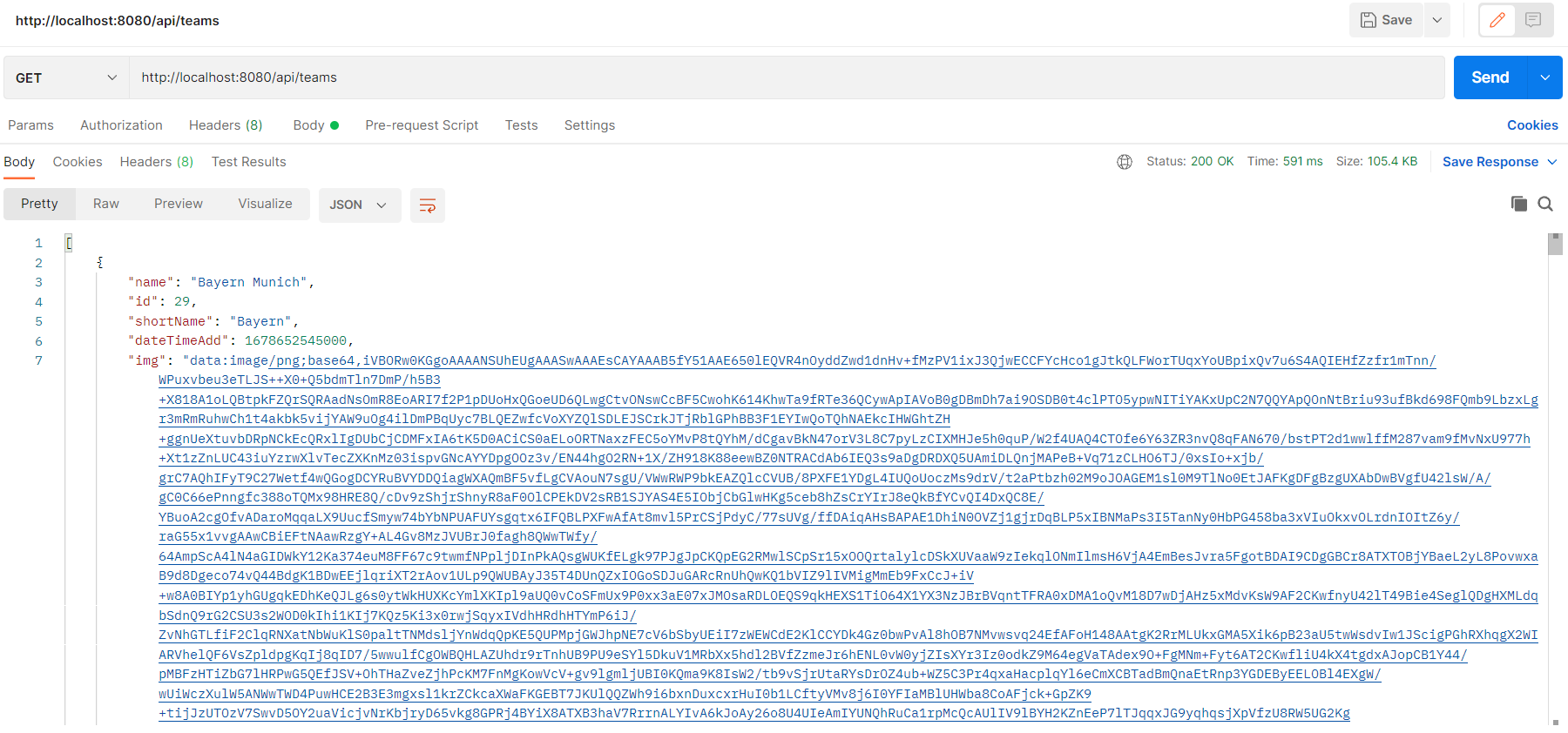
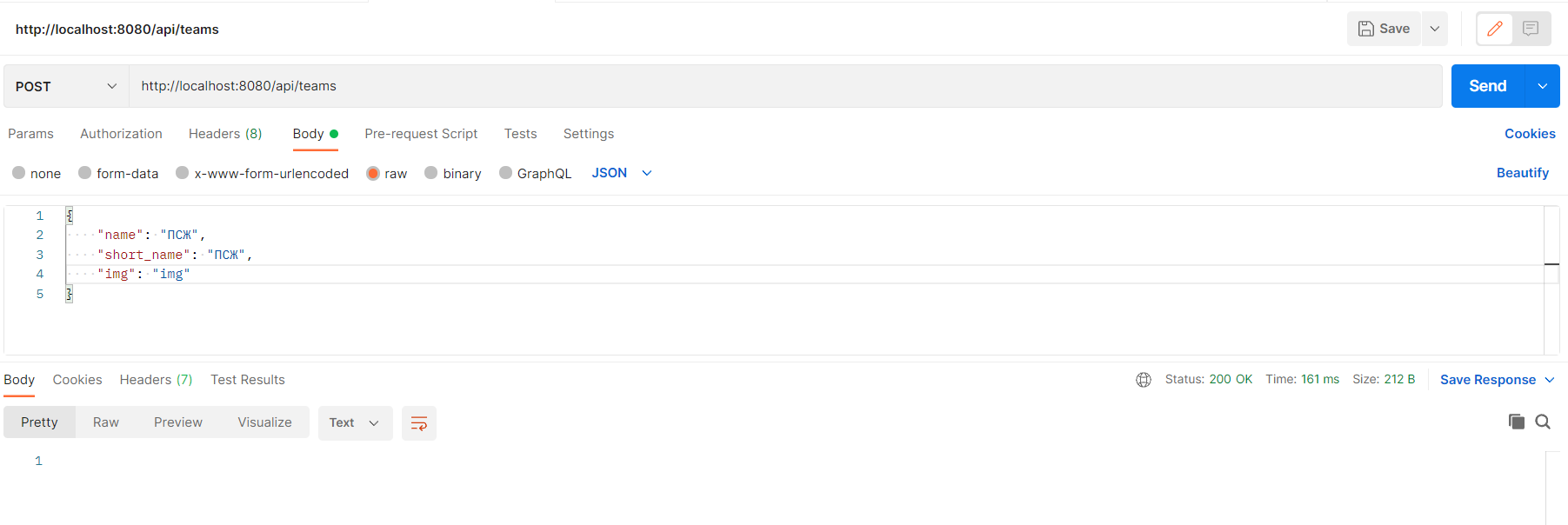


Рисунок 1 – Проверка метода getAllTeams()



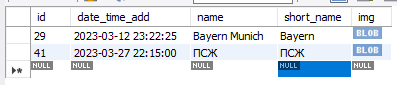


Рисунок 2 – Проверка метода addNewTeam(@RequestBody TeamsDto dto)

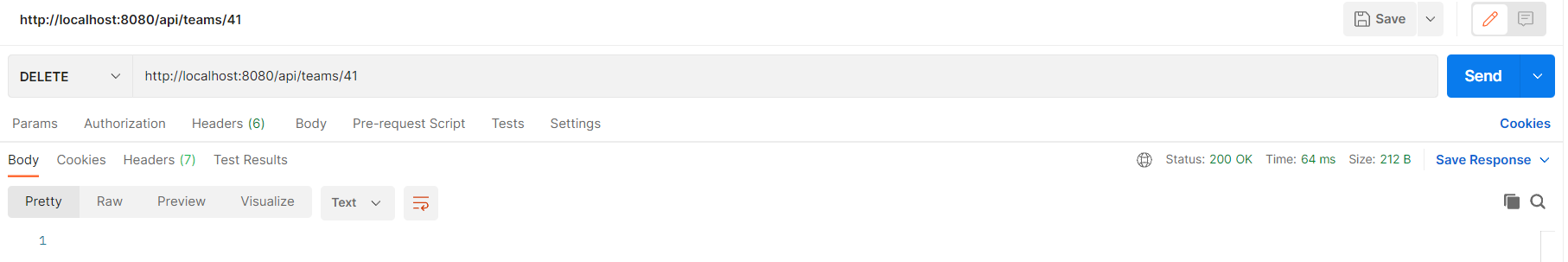


Рисунок 3 – Проверка метода deleteTeamById(@PathVariable Integer id)

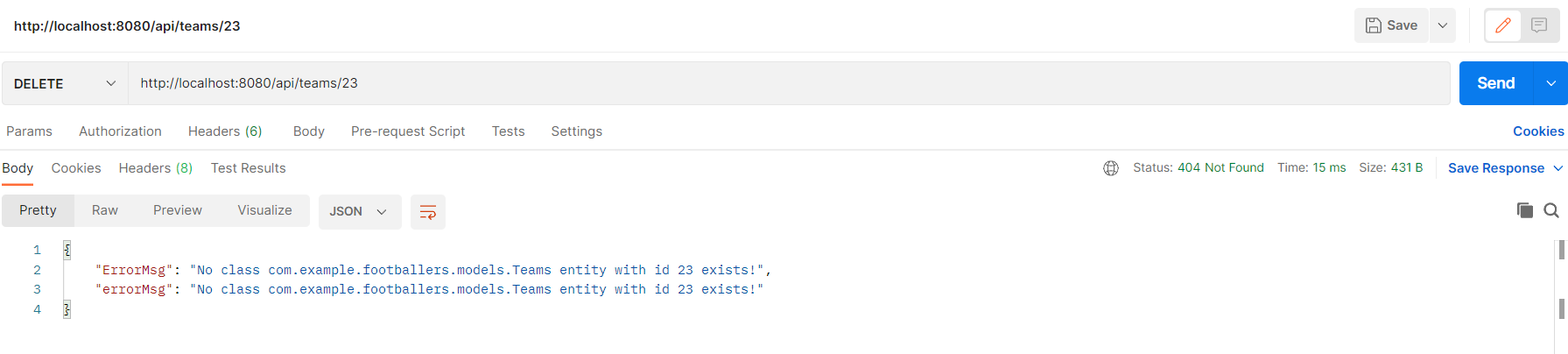
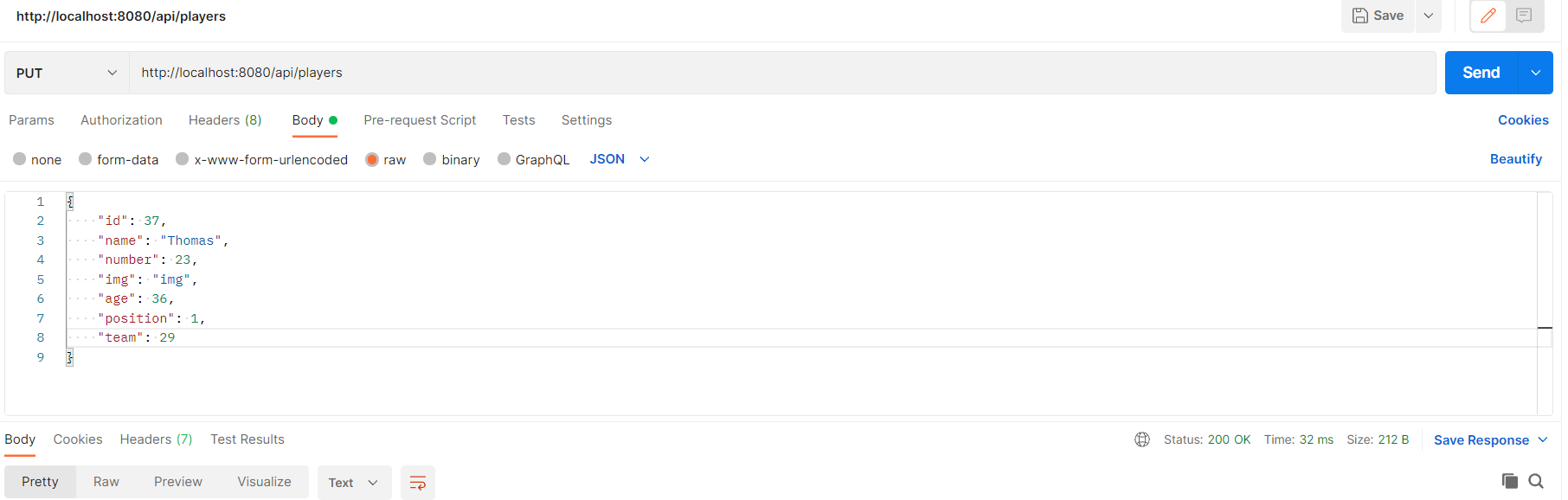


Рисунок 4 – Проверка статуса 404 метода deleteTeamById(@PathVariable Integer id)



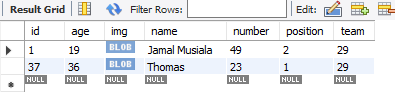


Рисунок 5 – Проверка метода updatePlayerData(@RequestBody PlayersDto dto)

3) Был реализован CrudService веб клиента.

Листинг 3 – CrudService веб клиента

import axios from "axios"

import ApiPath from "../constans/ApiPath";

export default class CrudService {

static async getAll(url) {

const response = await axios.get(String(url));

return response;

}

static async getAllByTeam(url) {

const response = await axios.get(String(url));

return response;

}

static async add(url, obj) {

const response = await axios.post(String(url), obj, {

headers: ApiPath.Headers

});

return response;

}

static async deleteById(url, id) {

const response = await axios.delete(String(url) + `/${id}`);

return response;

}

static async update(url, obj) {

const response = await axios.put((String)(url), obj, {

headers: ApiPath.Headers

});

return response;

}

}

4) Была реализована спецификация REST API в виде Swagger.

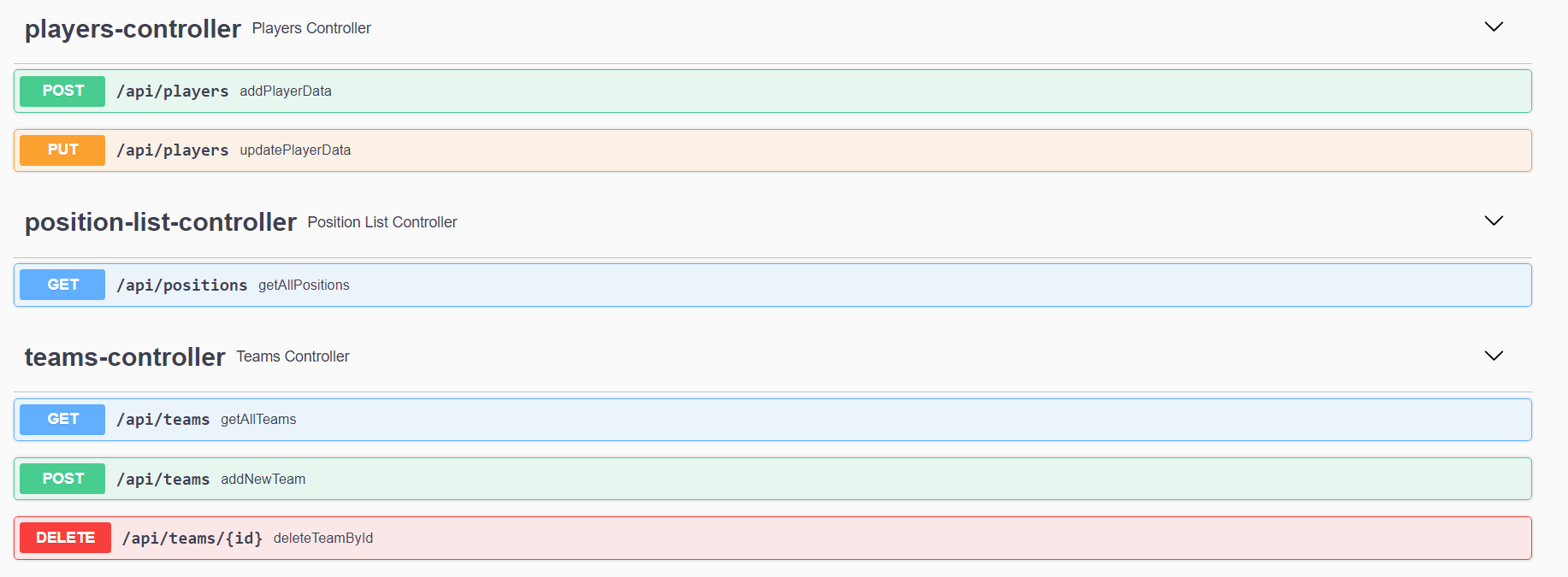


Рисунок 6 – Список контроллеров

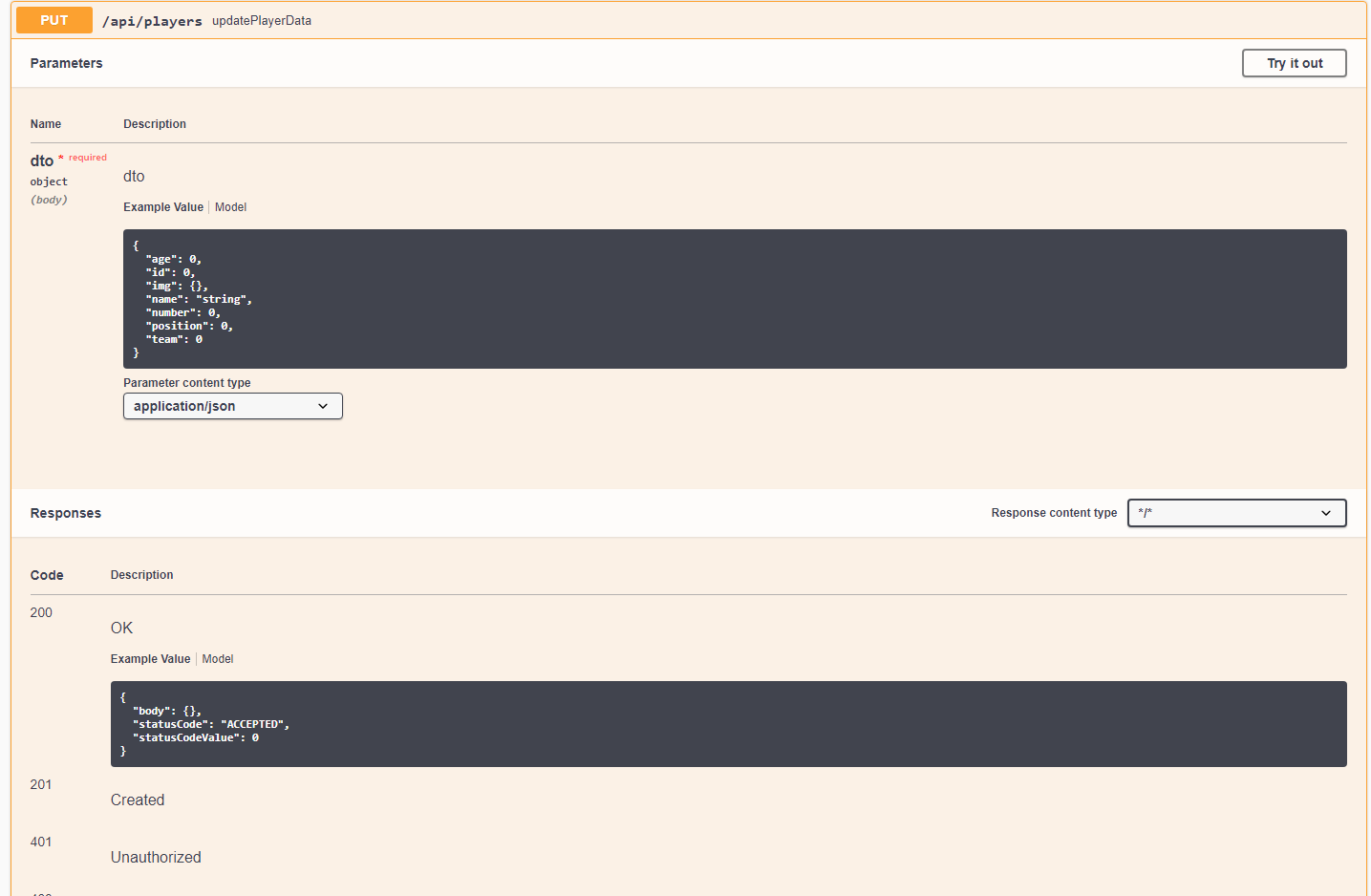


Рисунок 7 – Пример описания метода PUT

**Вывод:**

Был разработан и протестирован REST API с помощью Postman. Была сформирована спецификация REST API с помощью Swagger.