УДК 004

Создание REST службы с помощью Spring Boot и MongoDB

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается как REST сервисы будут разрабатываться с использованием технологий простым способом, не перетасовывая бизнес-логику. В этой статье используются IntelliJ IDEA, Maven и Postman для кодирования, сборки и тестирования REST API.
Ключевые слова: Java, Rest, MongoDB

Building a REST Service with Spring Boot and MongoDB

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Erovleva Regina Victorovna
Sholom-Aleichem Priamursky State University
Student

Abstract
This article looks at how REST services will be developed using technologies in a simple way without shuffling the business logic. This article uses IntelliJ IDEA, Maven, and Postman to code, build, and test the REST API.
Keywords: Java, Rest, MongoDB

1 Введение

1.1 Актуальность

REST означает «REpresentational State Transfer» - это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Архитектурный стиль – это набор согласованных ограничений и принципов проектирования, позволяющий добиться определённых свойств системы. REST является самым популярным архитектурным подходом для создания API в современном мире. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP.

1.2 Обзор исследований

В.Л. Волушкова провела обзор технологий на примере языка java и привела примеры с подробным описанием для использования фреймворка SpringBoot [1]. М.А. Потовиченко, М.В. Привалов и С.В. Корнев рассмотрели разработку программного продукта, который обеспечивает учет данных посещения занятий студентов, а также защиту их работ [2]. А.Д. Нарижный и Н.Е. Губенко провели сравнительный анализ технологий, которые имеют схожую функциональность и предназначены для одинаковых задач на технологии стеков JavaEE и Spring [3]. Д.В. Козырев, Л.А. Володченкова разработали программный интерфейс серверной части для облачного хранилища данных [4]. А.С. Волков и К.А. Волкова произвели краткий обзор элементов трехуровневой архитектуры для современных Web-приложений [5].

1.3 Цель исследования

Цель исследования – показать, как REST сервисы будут разрабатываться с использованием технологий простым способом, не перетасовывая бизнес-логику.

2 Материалы и методы

Для реализации будет использоваться язык программирования Java, IntelliJ IDEA, Maven и Postman для кодирования, сборки и тестирования REST API.

3 Результаты и обсуждения

REST служба предоставляет информацию о студентах и позволяет добавлять новых студентов в систему или удалять студентов из системы.

- Операция POST используется для добавления учащихся в систему.
- Операции GET используются для получения студентов по номеру студента или электронной почте.
- Коллекция студентов в базе данных MongoDB содержит данные об имени каждого студента, номере студента, адресе электронной почты, списке курсов и среднем балле.
- Все, что хранится, принимается и возвращается службой, форматируется как JSON .

Создаем новое приложение Spring в Spring Initalizr.

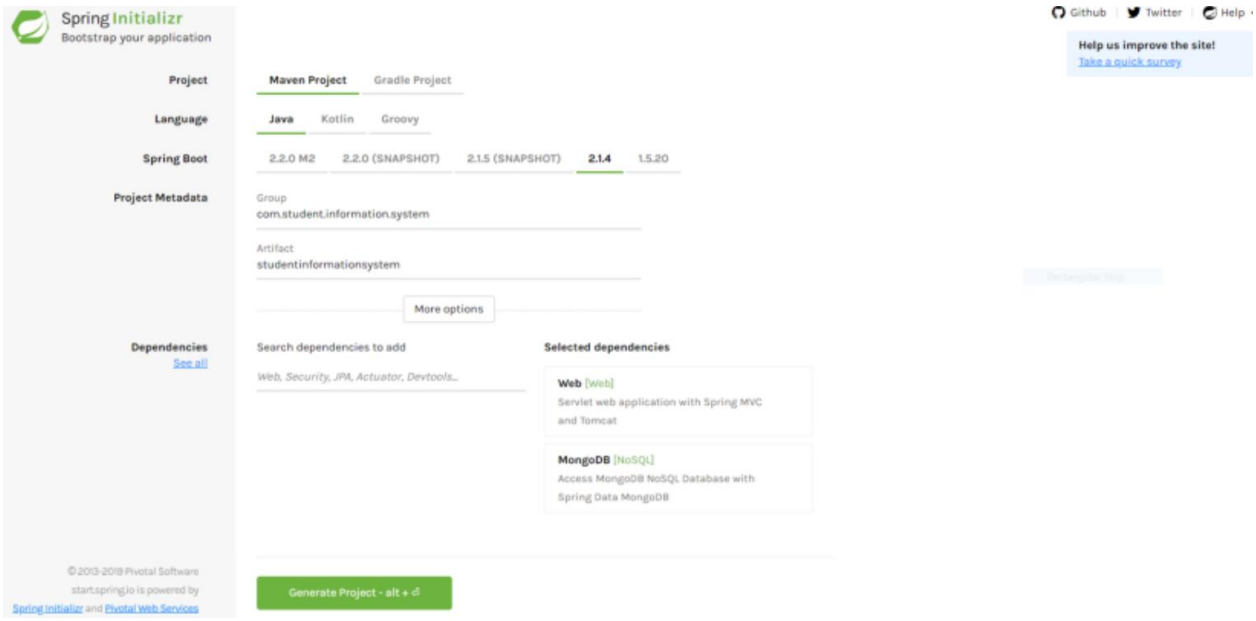


Рисунок 1 – Создание проекта

После создания проекта, его запуска, создадим первую модель Student.

У студента будет шесть полей: идентификатор, имя, номер студента, электронная почта, список занятий и средний балл.

```
@Document(collection = "students")
public class Student {
    3 usages
    @Id
    private String id;
    4 usages
    private String name;
    4 usages
    private long studentNumber;
    4 usages
    private String email;
    4 usages
    private List<String> courseList;
    4 usages
    private float gpa;

    5 usages
    public Student() {
    }
    1 usage
    public Student(String name, long studentNumber, String email, List<String> courseList, float gpa) {
        this.name = name;
        this.studentNumber = studentNumber;
        this.email = email;
        this.courseList = courseList;
        this.gpa = gpa;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id='" + id + '\'' +
                ", name='" + name + '\'' +
                ", studentNumber=" + studentNumber +
                ", email='" + email + '\'' +
                ", courseList=" + courseList +
                ", gpa=" + gpa +
                '}';
    }
```

Рисунок 2 – Создание модели Student

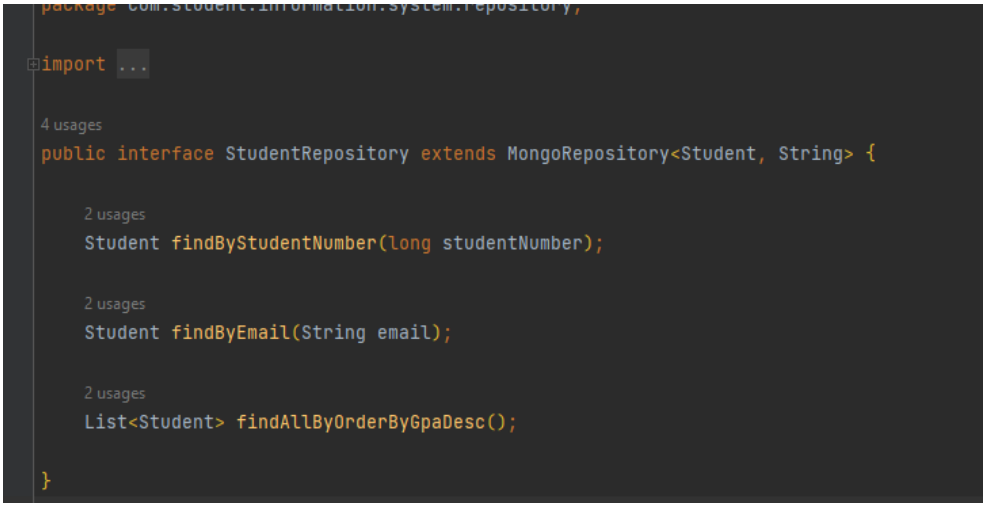
Аннотация «@Document» используется для тех же целей, что и «@Entity» аннотация в JPA.

После этого можно добавить класс «StudentDTO». Будем использовать это на уровне контроллера для взаимодействия с интерфейсом или любыми другими потребителями.

```
10 usages
public class StudentDTO {

    3 usages
    private String id;
    3 usages
    private String name;
    3 usages
    private long studentNumber;
    3 usages
    private String email;
    3 usages
    private List<String> courseList;
    3 usages
    private float gpa;

    public StudentDTO() {
    }

    public StudentDTO(String id, String name, long studentNumber, String email,
                      List<String> courseList, float gpa) {
        this.id = id;
        this.name = name;
        this.studentNumber = studentNumber;
        this.email = email;
        this.courseList = courseList;
        this.gpa = gpa;
    }
```

Рисунок 3 – Создание класса StudentDTO

Теперь приступим к созданию интерфейса, который будет расширять MongoRepository. После чего, можно будет написать запросы, которые будут выполняться при обращении к Mongo.



```
import ...

4 usages
public interface StudentRepository extends MongoRepository<Student, String> {

    2 usages
    Student findByStudentNumber(long studentNumber);

    2 usages
    Student findByEmail(String email);

    2 usages
    List<Student> findAllByOrderByGpaDesc();

}
```

Рисунок 4 – Создание интерфейса StudentRepository

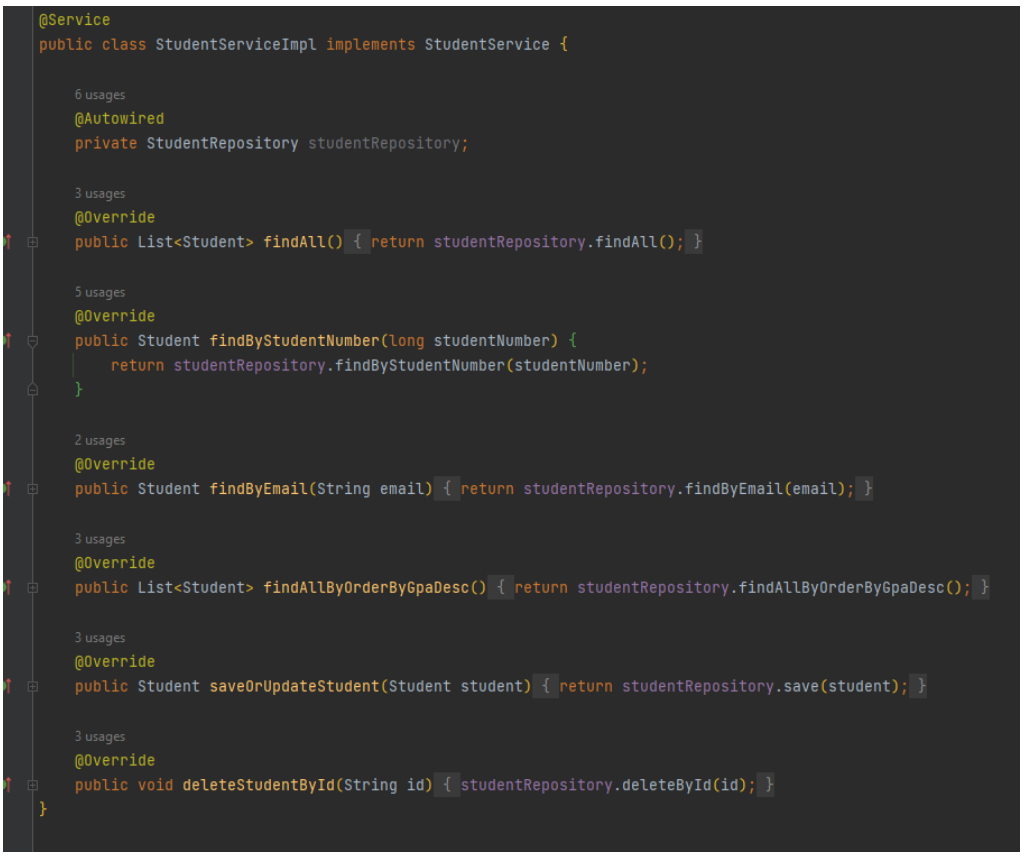
Интерфейс студенческого репозитория имеет три метода. Первые два метода находят студента по его номеру или электронной почте. Последний метод сортирует студентов по их среднему баллу.

Теперь создаем студенческий сервис, который будет вызывать студенческий контроллер. У студенческого сервиса есть шесть методов. Просто эти методы перечисляют студентов в соответствии с некоторыми критериями, сохраняют студента, обновляют студента и удаляют студента.

```
8 usages  1 implementation
public interface StudentService {

    3 usages  1 implementation
    List<Student> findAll();

    5 usages  1 implementation
    Student findByStudentNumber(long studentNumber);

    2 usages  1 implementation
    Student findByEmail(String email);

    3 usages  1 implementation
    List<Student> findAllByOrderByGpaDesc();

    3 usages  1 implementation
    Student saveOrUpdateStudent(Student student);

    3 usages  1 implementation
    void deleteStudentById(String id);

}
```

Рисунок 5 – Создание струденческого сервиса StudentService

Обычно классы контроллеров взаимодействуют напрямую с методами классов репозитория. Но в данном случае нужна какая-то бизнес-логика, нет необходимости писать этот блок кода в контроллере. Из-за этого необходим сервисный уровень.



```java
@Service
public class StudentServiceImpl implements StudentService {

    6 usages
    @Autowired
    private StudentRepository studentRepository;

    3 usages
    @Override
    public List<Student> findAll() { return studentRepository.findAll(); }

    5 usages
    @Override
    public Student findByStudentNumber(long studentNumber) {
        return studentRepository.findByStudentNumber(studentNumber);
    }

    2 usages
    @Override
    public Student findByEmail(String email) { return studentRepository.findByEmail(email); }

    3 usages
    @Override
    public List<Student> findAllByOrderByGpaDesc() { return studentRepository.findAllByOrderByGpaDesc(); }

    3 usages
    @Override
    public Student saveOrUpdateStudent(Student student) { return studentRepository.save(student); }

    3 usages
    @Override
    public void deleteStudentById(String id) { studentRepository.deleteById(id); }
}
```

Рисунок 6 – Создание струденческого сервиса StudentServiceImpl

Теперь можно создать REST контроллер. Этот контроллер будет иметь четыре «@GetMapping», один «@PostMapping» и один «@DeleteMapping».

```java
@RestController
@RequestMapping("/students")
public class StudentRestController {

    7 usages
    @Autowired
    private StudentService studentService;

    @GetMapping(value = "/")
    public List<StudentDTO> getAllStudents() {
        return ObjectMapperUtils.mapAll(studentService.findAll(), StudentDTO.class);
    }

    @GetMapping(value = "/byStudentNumber/{studentNumber}")
    public StudentDTO getStudentByStudentNumber(@PathVariable("studentNumber") Long studentNumber) {
        return ObjectMapperUtils.map(studentService.findByStudentNumber(studentNumber), StudentDTO.class);
    }

    @GetMapping(value = "/byEmail/{email}")
    public StudentDTO getStudentByEmail(@PathVariable("email") String email) {
        return ObjectMapperUtils.map(studentService.findByEmail(email), StudentDTO.class);
    }

    @GetMapping(value = "/orderByGpa")
    public List<StudentDTO> findAllByOrderByGpaDesc() {
        return ObjectMapperUtils.mapAll(studentService.findAllByOrderByGpaDesc(), StudentDTO.class);
    }

    @PostMapping(value = "/save")
    public ResponseEntity<?> saveOrUpdateStudent(@RequestBody StudentDTO studentDTO) {
        studentService.saveOrUpdateStudent(ObjectMapperUtils.map(studentDTO, Student.class));
        return new ResponseEntity( body: "Студент успешно добавлен", HttpStatus.OK);
    }

    @DeleteMapping(value = "/delete/{studentNumber}")
    public ResponseEntity<?> deleteStudentByStudentNumber(@PathVariable long studentNumber) {
        studentService.deleteStudentById(studentService.findByStudentNumber(studentNumber).getId());
        return new ResponseEntity( body: "Студент удален", HttpStatus.OK);
    }

}
```

Рисунок 7 – Создание REST сервиса

Чтобы подключиться к базе данных MongoDB, отредактируем настройки в файле «application.properties», который является частью «resources». Следующие конфигурации достаточны, поскольку в настоящее время в базе данных нет аутентификации.

```
#server
server.port=8081

#mongodb
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=test

#logging
logging.level.org.springframework.data=debug
logging.level.=errors
```

Рисунок 8 – Данные для подключения к Mongo

После запуска проекта можно протестировать REST службу. Для этого откроем Postman, введем URL-адрес и выберем метод GET.

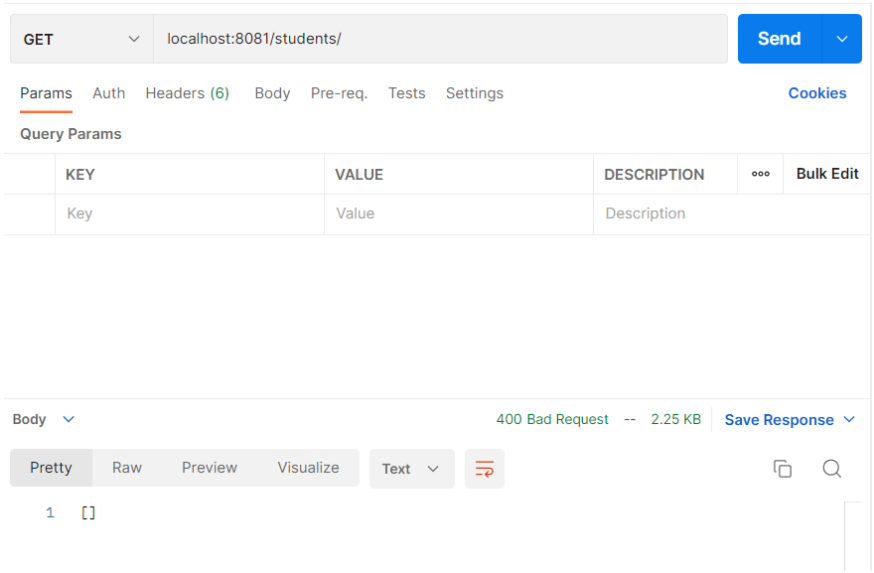


Рисунок 9 – Вызов GET метода

Можно добавить ученика в систему методом POST в Postman. Во-первых, необходимо ввести «Content-Type: application/json» пара ключ-значение заголовка. Во-вторых, на вкладке «Body» нужно ввести информацию о студенте, которую хотим сохранить в базе данных в формате JSON.

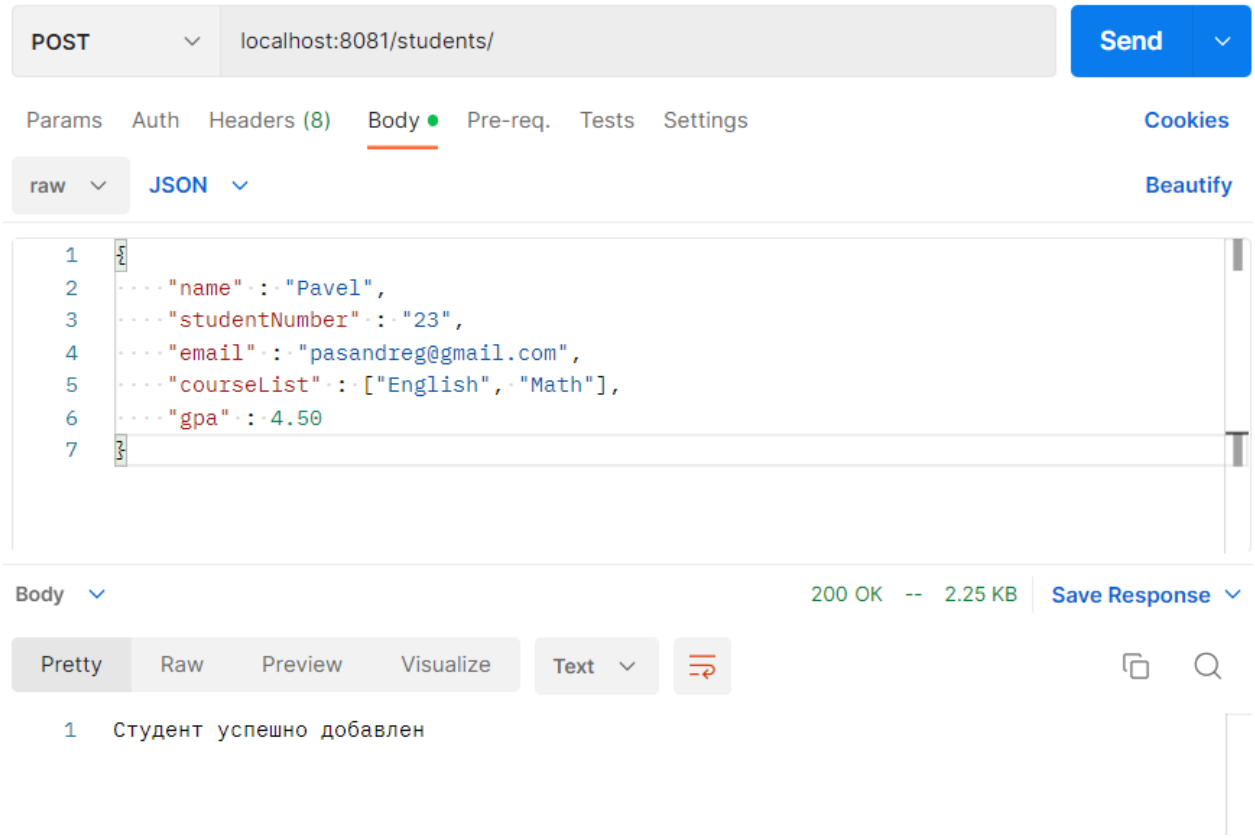


Рисунок 10 – Добавление студента

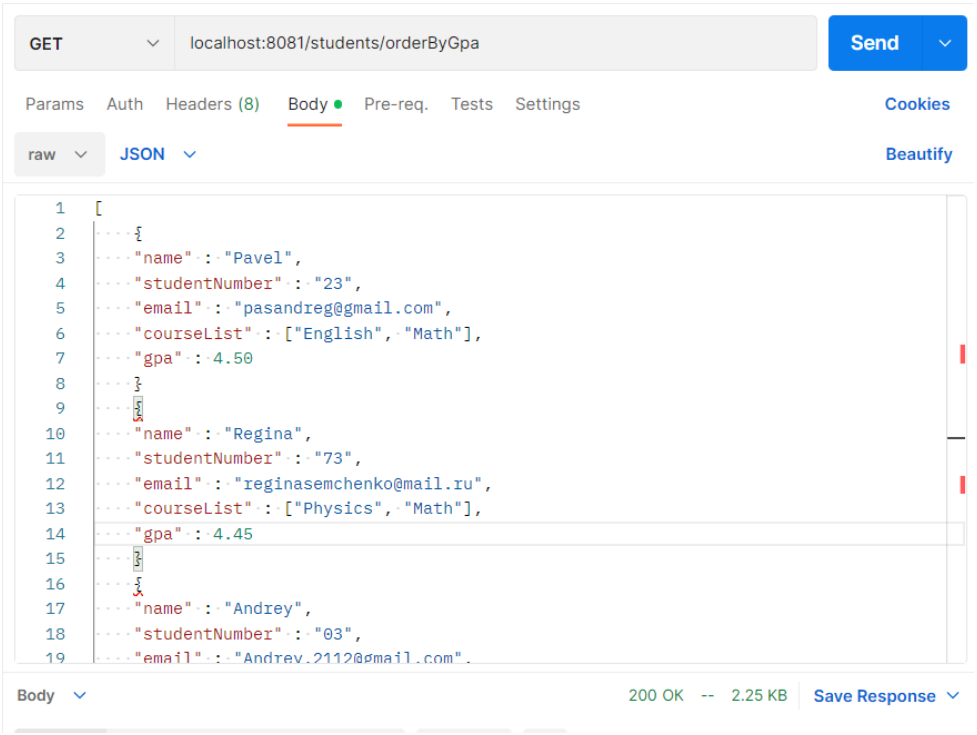


Рисунок 11 – Список студентов по среднему баллу

Если нужно удалить студента, то необходимо выбрать метод DELETE в Postman и добавить номер студента, которого нужно удалить, в конец ссылки.

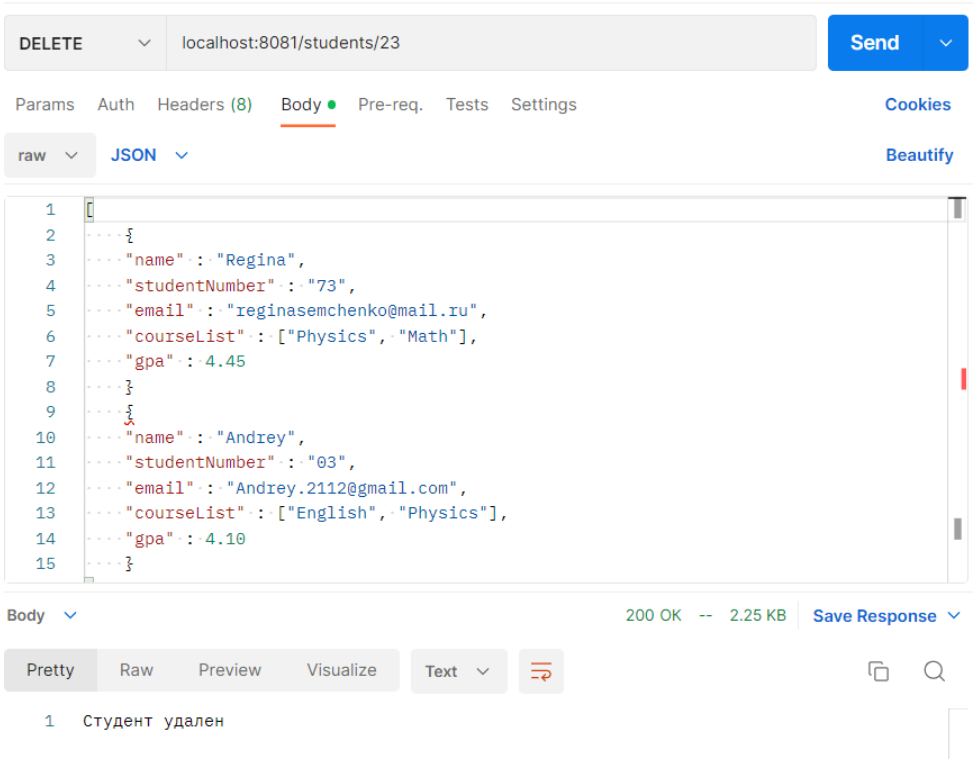


Рисунок 12 – Удаление студента

Выводы

В данной статье рассматривался процесс создания и работы REST сервиса с использованием технологий MongoDB, Postman и языка программирования Java.

Библиографический список

1. Волушкова В.Л. Архитектурные решения java для доступа к данным // Тверской государственный университет. 2019. С. 137-140.
2. Потовиченко М.А., Привалов М.В., Корнев С.В. Компьютеризированная подсистема учета текущей успеваемости студента в условиях вуза // Информатика, управляющие системы, математическое и компьютерное моделирование. 2019. № 2. С. 71-75.
3. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование. 2020. № 17. С. 459-462.
4. Володченкова Л.А., Козырев Д.В. Разработка серверной части программного приложения для удаленного хранения данных// Математические структуры и моделирование. 2020. № 1 (53). С. 108-138.
5. Волков А.С., Волкова К.А. Обзор архитектурных компонентов современного веб-приложения // Аллея науки. 2019. № 1(28). С. 958-961.