Практическая работа Java

Настроем бд:

Java > demo > src > main > resources > ! application.yml

```yaml
spring:
  application:
    name: demo

  security:
    basic:
      enabled: true
    user:
      name: "admin"
      password: "admin123"
      roles: "ADMIN"

  datasource:
    url: "jdbc:h2:mem:taskdb"
    driver-class-name: "org.h2.Driver"
    username: "sa"
    password: ""

  jpa:
    database-platform: "org.hibernate.dialect.H2Dialect"
    show-sql: true
    hibernate:
      ddl-auto: update

  h2:
    console:
      enabled: true
      path: /h2-console
      settings:
        web-allow-others: false
        trace: false
```

Создание модели задачи:

```xml
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

```java
package com.example.demo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

import java.time.LocalDateTime;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Task {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String description;
    private String status;
    private LocalDateTime createdAt;
}
```

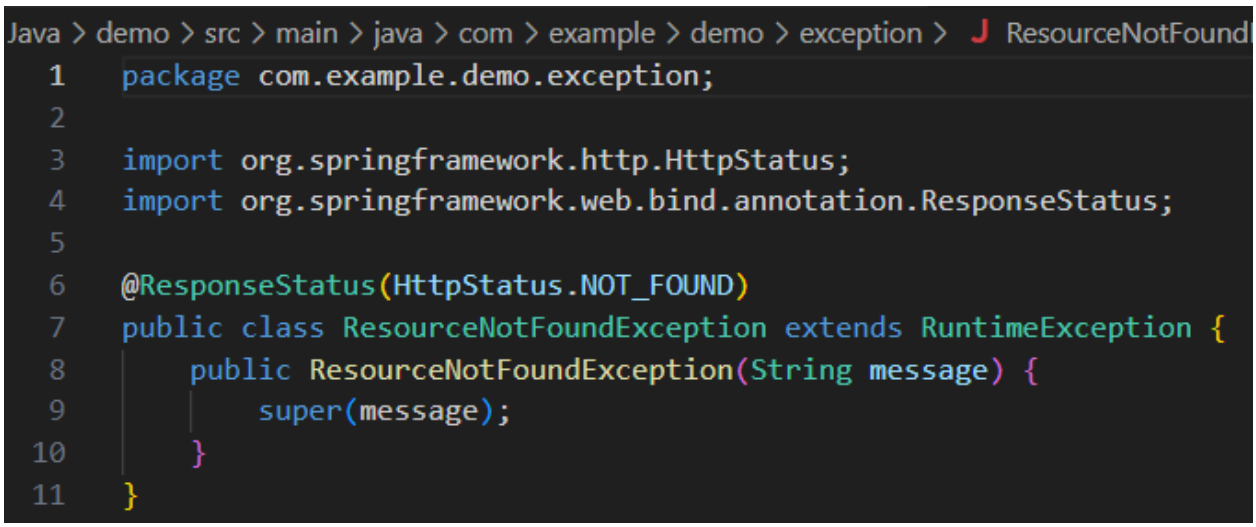
Созданиие репозитория задач:

Java > src > main > java > com > example > demo > repository > J TaskRepository.java

```java
package com.example.demo.repository;

import com.example.demo.model.Task;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface TaskRepository extends JpaRepository<Task, Long> {
}
```

Создание REST контроллера

Класс для обработки ошибок:


Java > demo > src > main > java > com > example > demo > exception > ResourceNotFound
1 package com.example.demo.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 @ResponseStatus(HttpStatus.NOT_FOUND)
7 public class ResourceNotFoundException extends RuntimeException {
8 public ResourceNotFoundException(String message) {
9 super(message);
10 }
11 }

Контроллер:

ava > demo > src > main > java > com > example > demo > controller > J TaskController.java

```java
package com.example.demo.controller;

import com.example.demo.exception.ResourceNotFoundException;
import com.example.demo.model.Task;
import com.example.demo.repository.TaskRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.time.LocalDateTime;
import java.util.List;

@RestController
@RequestMapping("/api/tasks")
public class TaskController {

    @Autowired
    private TaskRepository taskRepository;

    @GetMapping
    public List<Task> getAllTasks() {
        return taskRepository.findAll();
    }

    @PostMapping
    public Task createTask(@RequestBody Task task) {
        task.setCreatedAt(LocalDateTime.now());
        return taskRepository.save(task);
    }

    @PutMapping("/{id}")
    public Task updateTask(@PathVariable Long id, @RequestBody Task updatedTask) {
        return taskRepository.findById(id)
            .map(task -> {
                task.setTitle(updatedTask.getTitle());
                task.setDescription(updatedTask.getDescription());
                task.setStatus(updatedTask.getStatus());
                return taskRepository.save(task);
            })
            .orElseThrow(() -> new ResourceNotFoundException("Задача с id " + id + " не найдена"));
    }

    @DeleteMapping("/{id}")
    public void deleteTask(@PathVariable Long id) {
        taskRepository.deleteById(id);
    }
}
```

Добавление безопасности:

В pom.xml добавим зависимость:

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>

    <build>
```

Теперь обновляем application.yml:

demo > src > main > resources > ! application.yml

```yaml
spring:
  application:
    name: demo

  security:
    basic:
      enabled: true
    user:
      name: "admin"
      password: "admin123"
      roles: "ADMIN"

  datasource:
    url: "jdbc:h2:mem:taskdb"
    driver-class-name: "org.h2.Driver"
    username: "sa"
    password: ""

  jpa:
    database-platform: "org.hibernate.dialect.H2Dialect"
    show-sql: true
    hibernate:
      ddl-auto: update

  h2:
    console:
      enabled: true
      path: /h2-console
      settings:
        web-allow-others: false
        trace: false
```

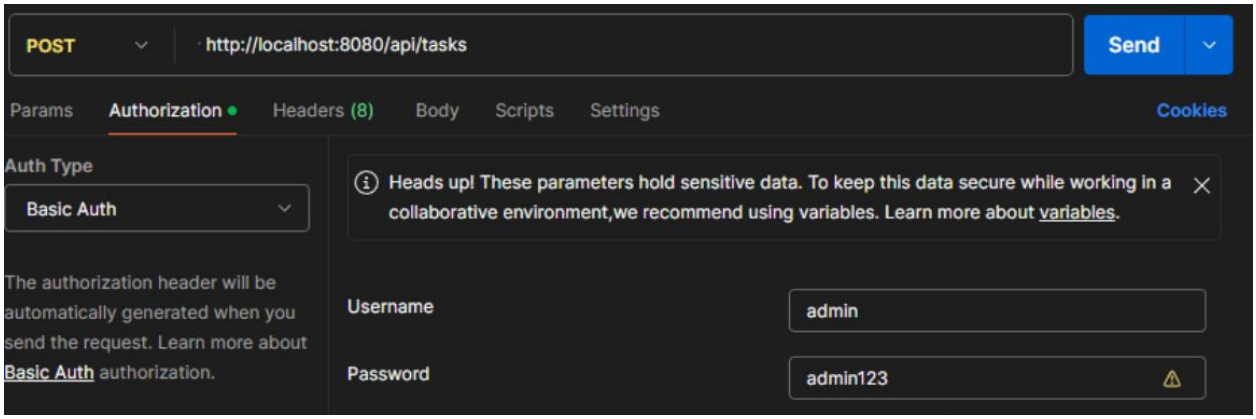
Конфигурация безопасности:

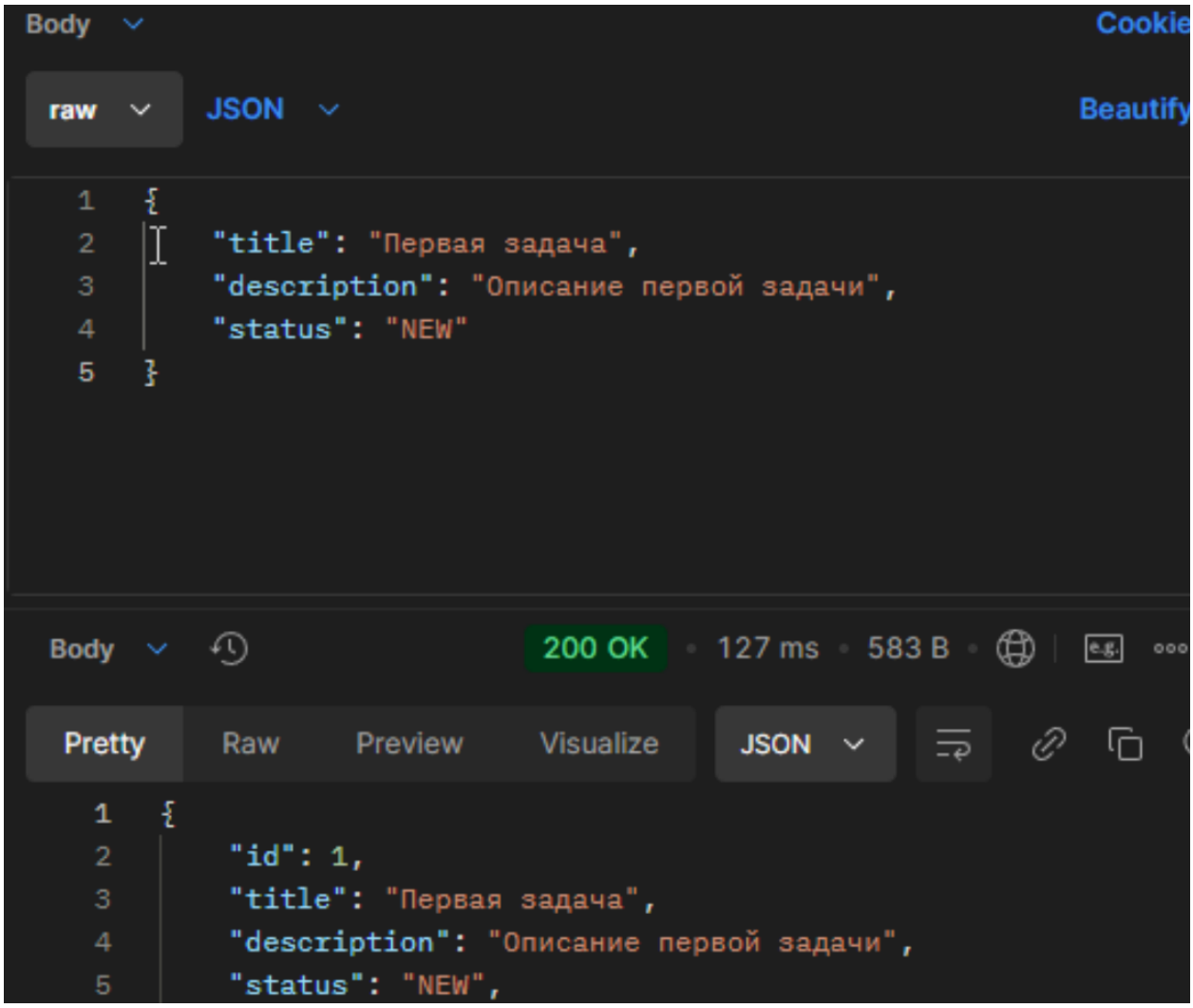
Java > demo > src > main > java > com > example > demo > config > J SecurityConfig.java

```java
package com.example.demo.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .headers(headers -> headers
                .frameOptions()
                .sameOrigin()
            )
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/h2-console/**").permitAll()
                .requestMatchers("/api/tasks/**").authenticated()
                .anyRequest().authenticated()
            )
            .httpBasic(Customizer.withDefaults());

        return http.build();
    }
}
```

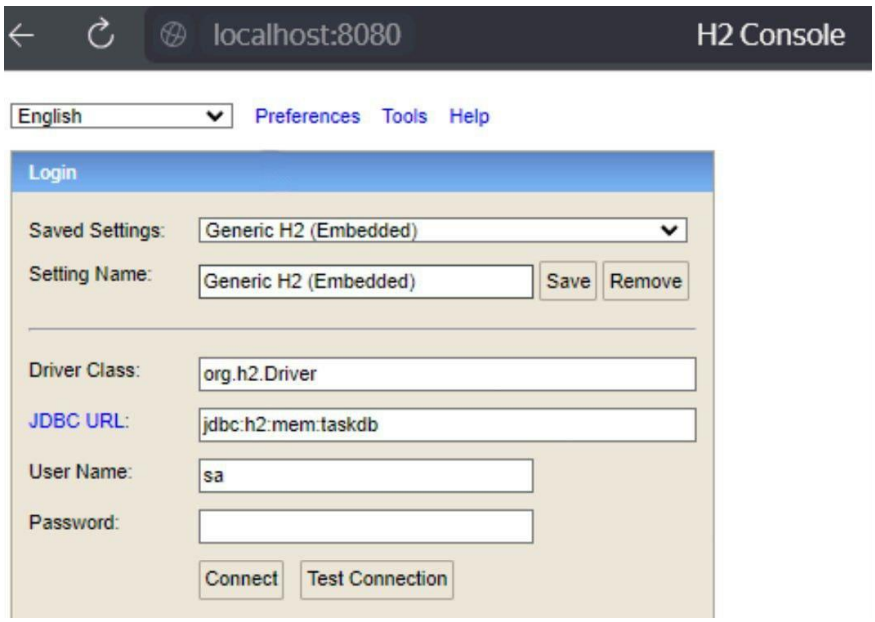
Тестируем приложение:

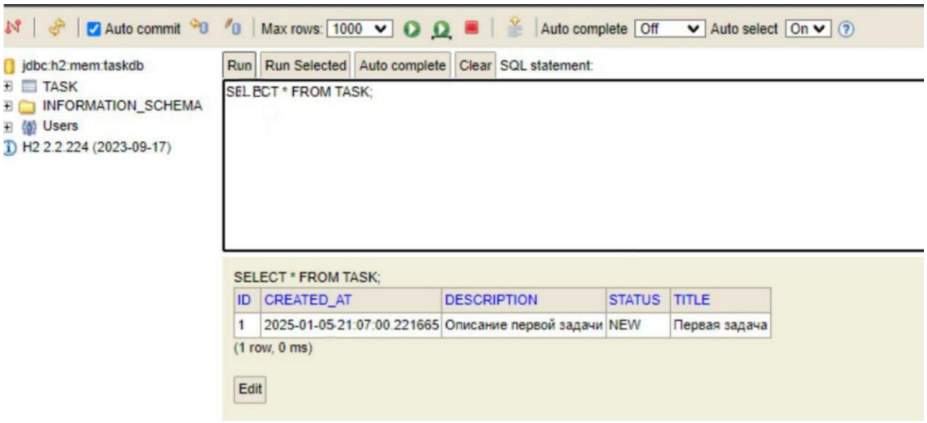
POST: создание задачи:



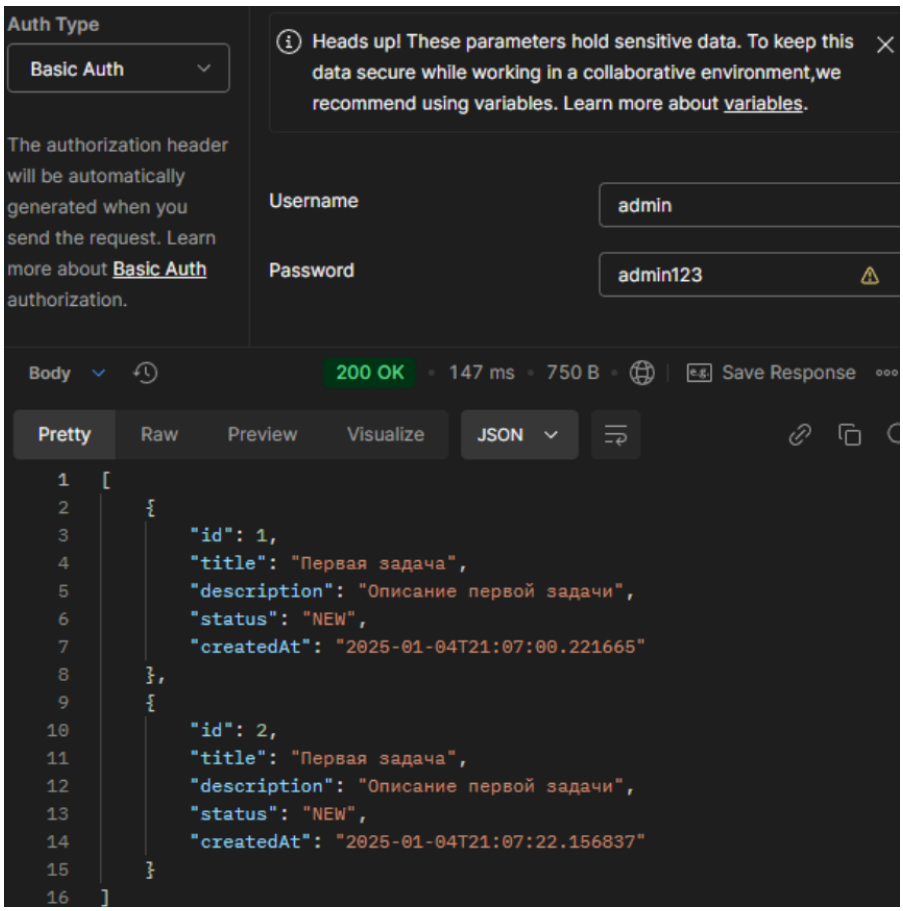


Проверка в H2:

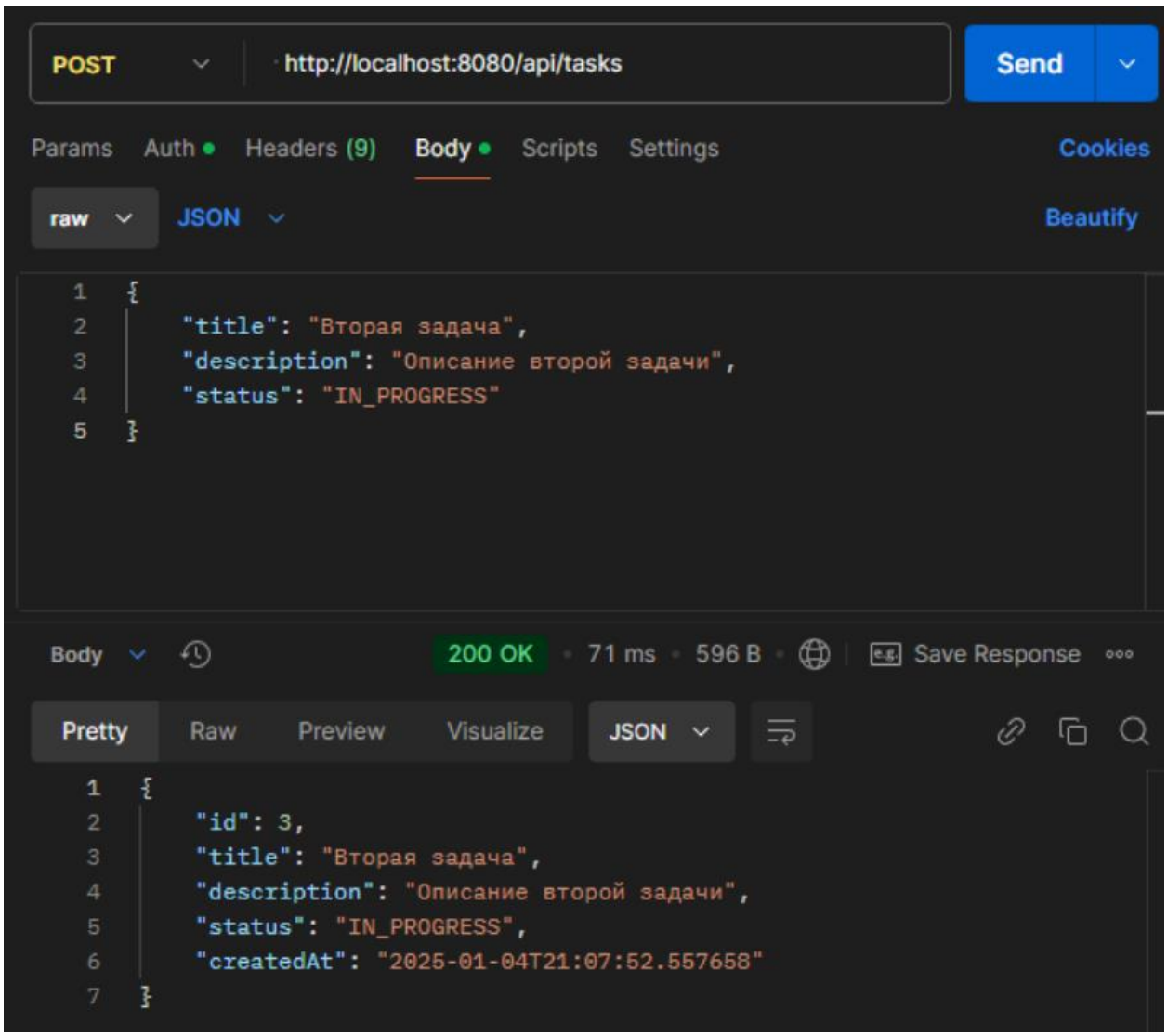




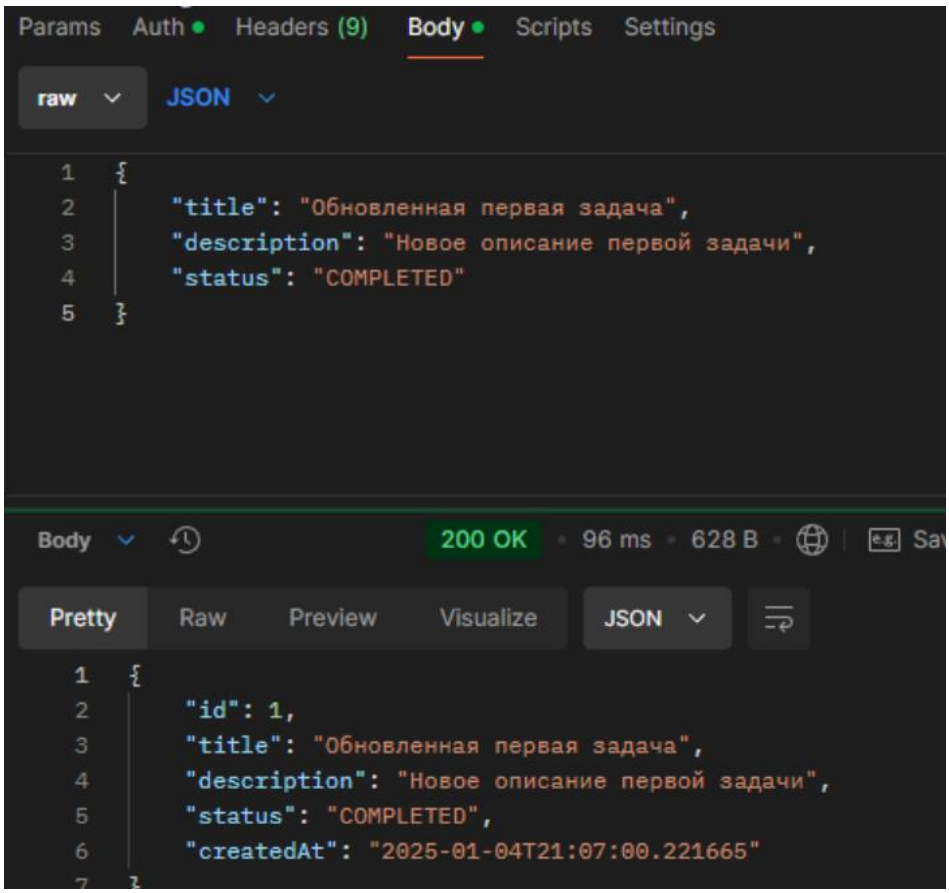
GET: получения списка задач



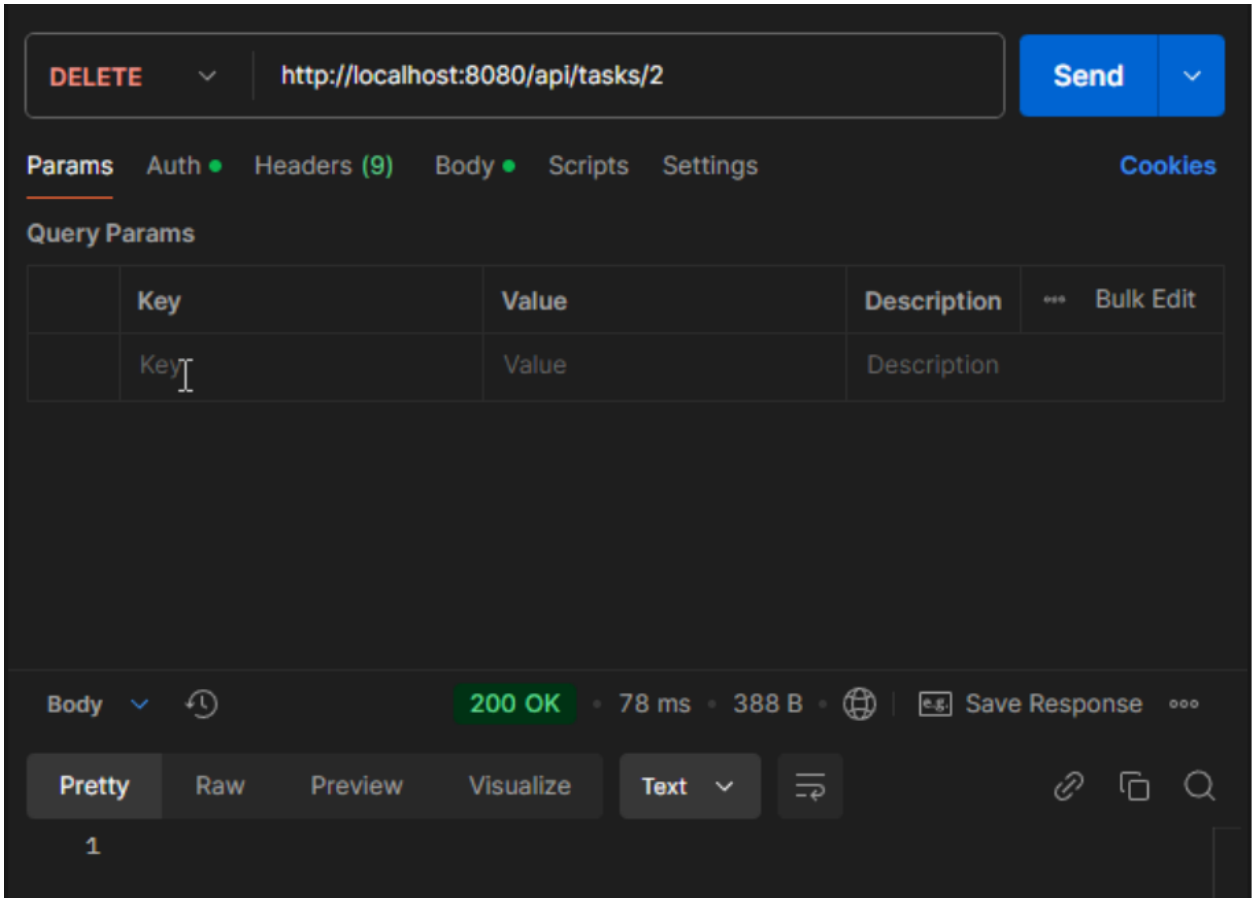
POST: создание второй задачи



PUT: обновим задачу



DELETE: удаление задачи:



```
SELECT * FROM TASK;
```

SELECT * FROM TASK;

ID	CREATED_AT	DESCRIPTION	STATUS	TITLE
1	2025-01-04 21:07:00.221665	Новое описание первой задачи	COMPLETED	Обновленная первая задача
3	2025-01-04 21:07:52.557658	Описание второй задачи	IN_PROGRESS	Вторая задача

(2 rows, 0 ms)

Edit