Практическая работа Java

Настройка БД

```yaml
spring:
  datasource:
    url: jdbc:h2:mem:taskdb
    driverClassName: org.h2.Driver
    username: sa
    password: password
  h2:
    console:
      enabled: true
  jpa:
    hibernate:
      ddl-auto: update
```

Создание модели задачи

```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;
        Ctrl+L to chat, Ctrl+K to generate
    private String title;
    private String description;
    private String status;
    private LocalDateTime createdAt;
}
```

Создание репозитория для задач

```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> {
}
```

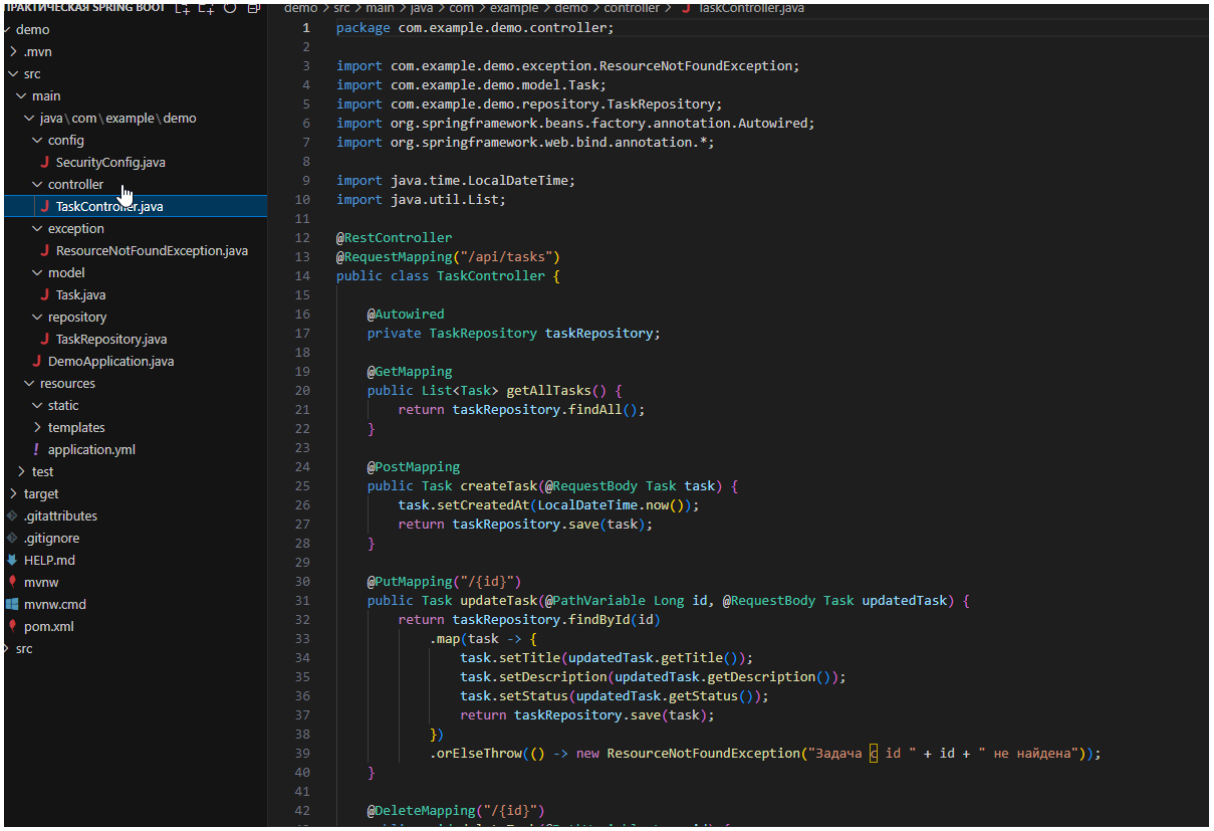
REST контроллер

класс для обработки ошибок

```java
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

и контроллер:



Добавление безопасности

добавим зависимость в pom.xml

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

обновляем application.yml

```
demo > src > main > resources > ! application.yml
1   spring:
2     datasource:
3       url: jdbc:h2:mem:taskdb
4       driverClassName: org.h2.Driver
5       username: sa
6       password: password
7     h2:
8       console:
9         enabled: true
10    jpa:
11      hibernate:
12        ddl-auto: update
13
14    security:
15      user:
16      name: admin
17      password: admin123
```

Создадим конфигурацию безопасности

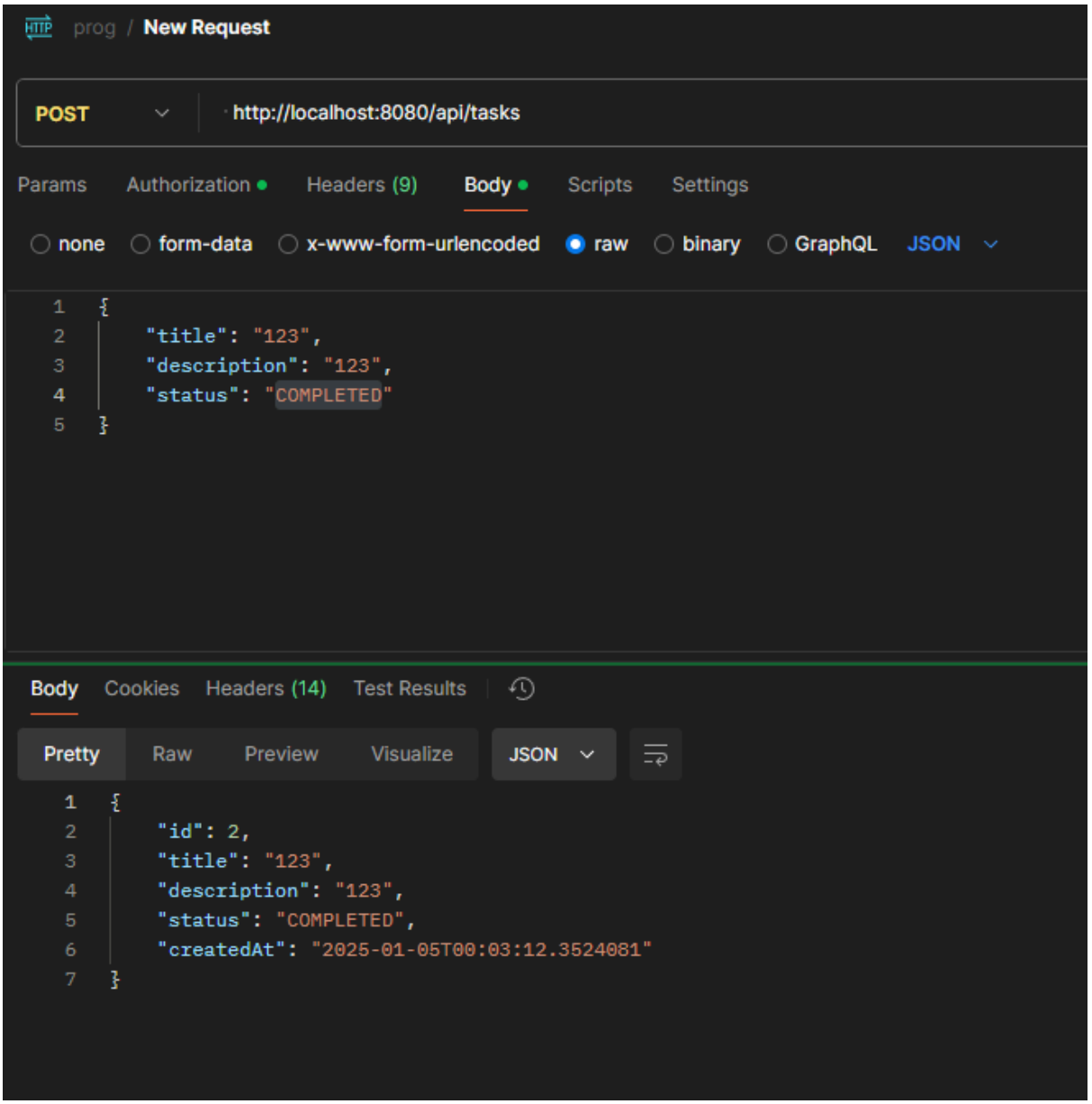
```
demo
  .mvn
  src   E:\Repositories\vladik\Практическая Spring Boot\demo\.mvn
    main
      java \ com \ example \ demo
        config
          SecurityConfig.java
        controller
          TaskController.java
        exception
          ResourceNotFoundException.java
        model
          Task.java
        repository
          TaskRepository.java
          DemoApplication.java
      resources
        static
        templates
        application.yml
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  src
```

```
1   package com.example.demo.config;
2
    .springframework.context.annotation.Bean;
    .springframework.context.annotation.Configuration;
5   import org.springframework.security.config.Customizer;
6   import org.springframework.security.config.annotation.web.builders.HttpSecurity;
7   import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
8   import org.springframework.security.web.SecurityFilterChain;
9
10  @Configuration
11  @EnableWebSecurity
12  public class SecurityConfig {
13
14      @Bean
15      public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
16          http
17              .csrf(csrf -> csrf.disable())
18              .headers(headers -> headers
19                  .frameOptions()
20                  .sameOrigin()
21              )
22              .authorizeHttpRequests(auth -> auth
23                  .requestMatchers("/h2-console/**").permitAll()
24                  .requestMatchers("/api/tasks/**").authenticated()
25                  .anyRequest().authenticated()
26              )
27              .httpBasic(Customizer.withDefaults());
28
29          return http.build();
30      }
31  }
```

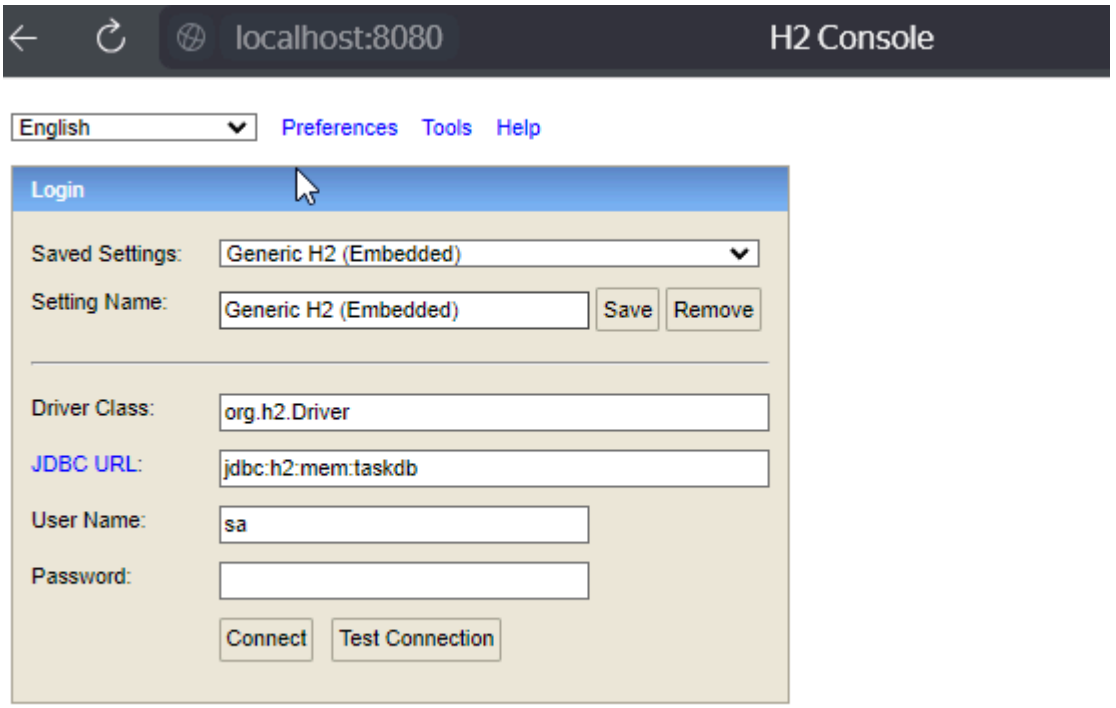
Тестирование приложения

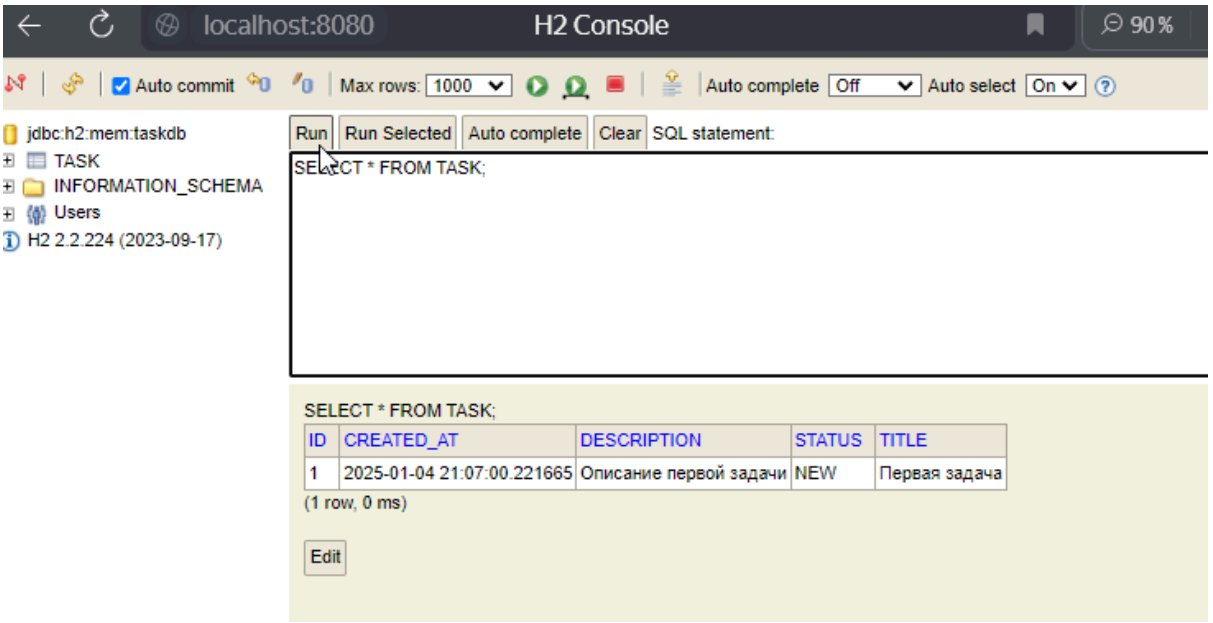
1. Создание задачи (POST)

указываем basic auth в постмане



2. Проверка в H2





3. Получение списка задач (GET)

prog / **New Request**

Save | Share

GET | http://localhost:8080/api/tasks | Send

Params | Auth | Headers (9) | Body | Scripts | Settings | Cookies

Auth Type

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Username: admin

Password: admin123

Body | 200 OK | 147 ms | 750 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "id": 1,
        "title": "Первая задача",
        "description": "Описание первой задачи",
        "status": "NEW",
        "createdAt": "2025-01-04T21:07:00.221665"
    },
    {
        "id": 2,
        "title": "Первая задача",
        "description": "Описание первой задачи",
        "status": "NEW",
        "createdAt": "2025-01-04T21:07:22.156837"
    }
]
```

4. Создание второй задачи (POST)

prog / New Request

Save Share

POST http://localhost:8080/api/tasks Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

```
1 {
2     "title": "Вторая задача",
3     "description": "Описание второй задачи",
4     "status": "IN_PROGRESS"
5 }
```

Body 200 OK 71 ms 596 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": 3,
3     "title": "Вторая задача",
4     "description": "Описание второй задачи",
5     "status": "IN_PROGRESS",
6     "createdAt": "2025-01-04T21:07:52.557658"
7 }
```

5. Обновление задачи (PUT)

prog / **New Request**

Save | Share

PUT http://localhost:8080/api/tasks/1 **Send**

Params | Auth | Headers (9) | Body | Scripts | Settings | Cookies

raw | JSON | Beautify

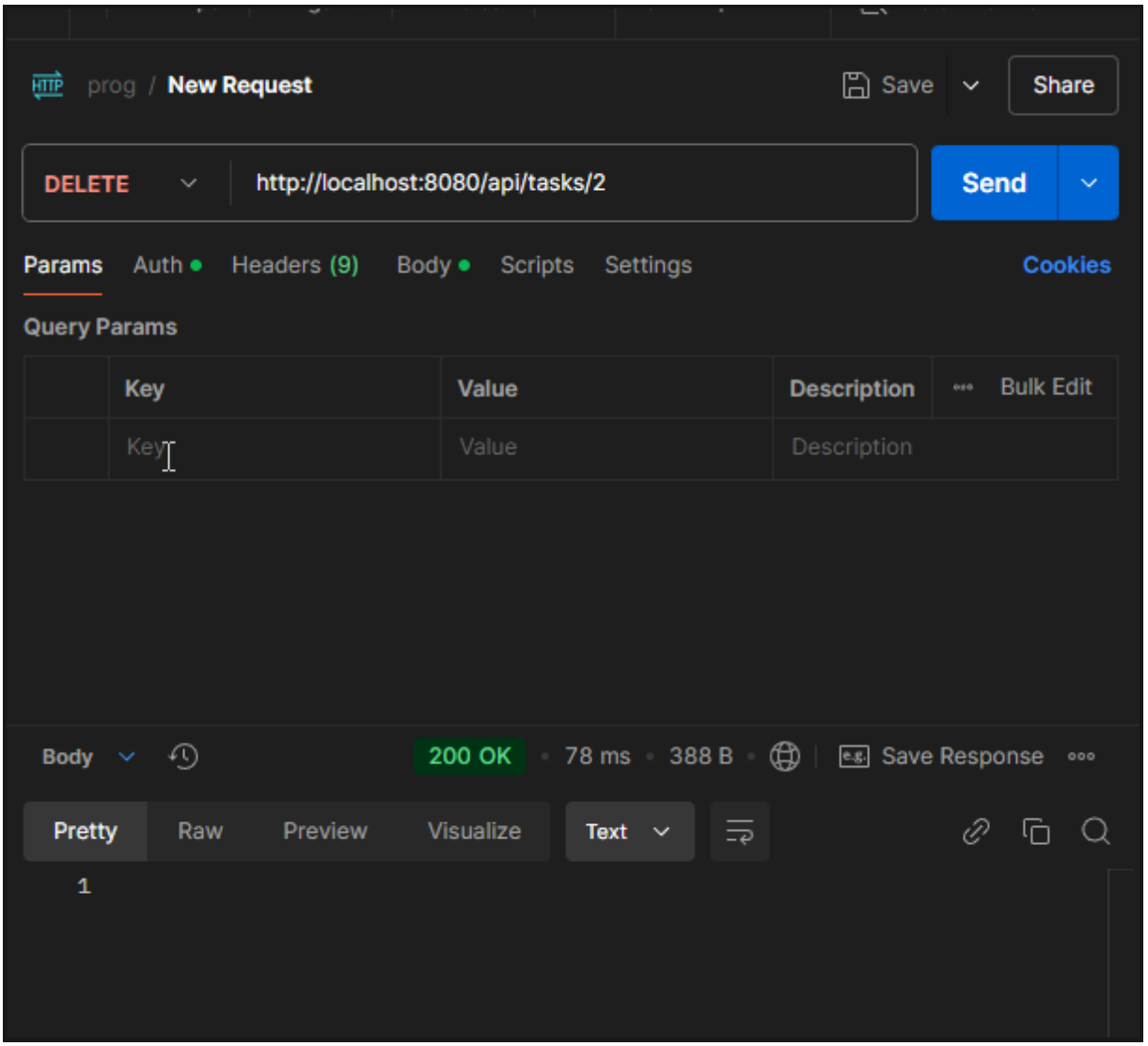
```
1 {
2     "title": "Обновленная первая задача",
3     "description": "Новое описание первой задачи",
4     "status": "COMPLETED"
5 }
```

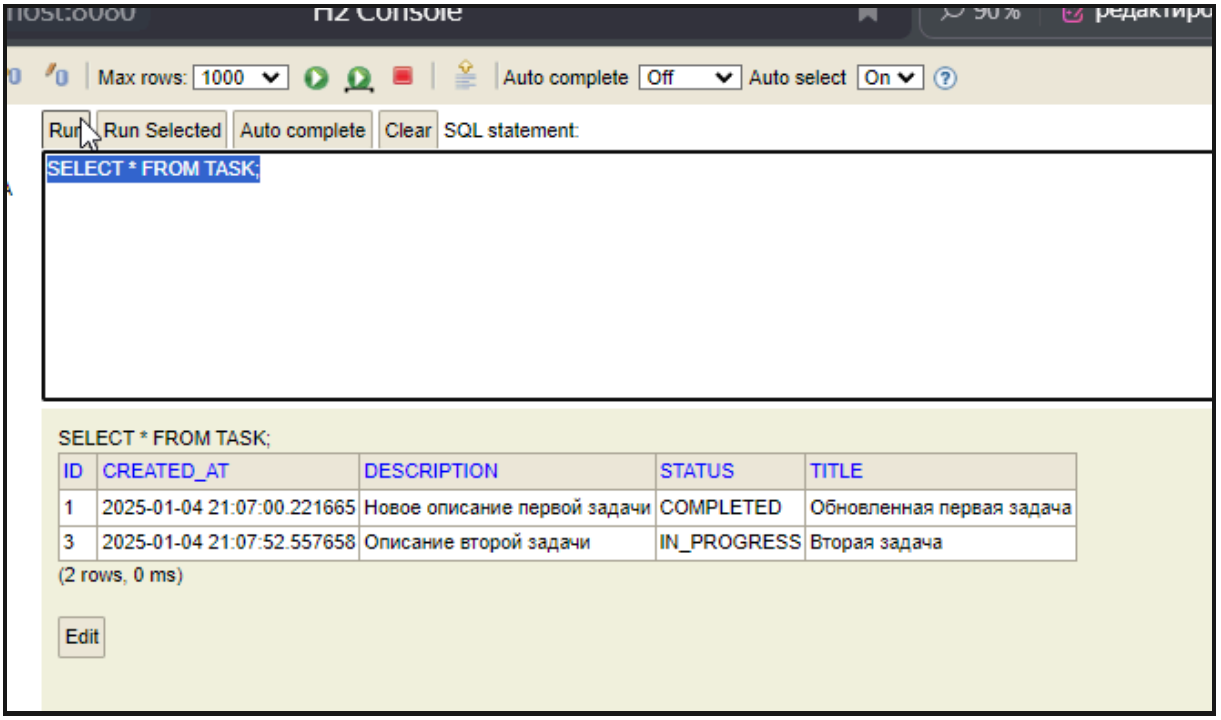
Body | 200 OK • 96 ms • 628 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2     "id": 1,
3     "title": "Обновленная первая задача",
4     "description": "Новое описание первой задачи",
5     "status": "COMPLETED",
6     "createdAt": "2025-01-04T21:07:00.221665"
7 }
```

6. Удаление задачи (DELETE)





GET /api/tasks - Get all tasks
GET /api/tasks/{id} - Get task by ID
POST /api/tasks - Create new task

PUT /api/tasks/{id} - Update task
DELETE /api/tasks/{id} - Delete task