análise de REQUISITOS .com.br

Engenharia de Software

ESPECIFICAÇÃO DE CASOS DE USO

DOCUMENTO X-0002

Wesley Filemon Rocha Rodrigues

ÚLTIMA ATUALIZAÇÃO: 31/03/2024

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

HISTÓRICO DE REVISÕES DO DOCUMENTO

DATA	VERSÃO	DESCRIÇÃO DA ALTERAÇÃO	AUTOR
14/03/2024	1	CRIAÇÃO DESTE DOCUMENTO	Wesley filemon
31/03/2024	2	CRIAÇÃO DAS PRIMEIRAS FUNCIONALIDADES	Wesley filemon

IDENTIFICAÇÃO DOS ENVOLVIDOS

PAPEL	NOME	EMAIL
ANALISTA DE REQUISITOS	Wesley Filemon	Wesleyfilemon@gmail.com
PRODUCT OWNER	Cristiano de Macêdo	
STAKEHOLDER	Cristiano de Macêdo	
PATROCINADOR	Cristiano de Macêdo	

DESCRIÇÃO DO CASO E USO

O nosso autor principal terá o primeiro contato com um sistema de login onde será necessário informar um usuário e senha, logo após será redirecionado a uma página onde terá sua primeira interação de decisão onde terá as opções de ver tarefas, Excluir tarefas, criar tarefas e Editar tarefas, na opção Ver tarefas o usuário terá as opções de ver suas tarefas passadas, Tarefas do dia e Tarefas Futuras, e ao escolher umas das opções ele terá s lista de tarefas, contendo descrição, importância, data e hora e Tarefas Compartilhada, Na opção de exclusão ele vera essas mesmas opções mas ao entrar nas tarefas ele poderá fazer as exclusões de tarefas do dia, Tarefas Futuras e Tarefas Compartilhada.

Na opção de Criar Tarefa, ele poderá criar uma tarefa, contendo Descrição, Importância e Data e Hora, em Data e hora pode haver um conflito então ele será alertado sobre o possível conflito, em Editar Tarefa ele vera essas mesmas opções só que agora podendo fazer edições nas tarefas.

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

DOCUMENTOS RELACIONADOS

- **SiglaProjeto_Caso_de_Uso_Autenticacao** 1.0

Autenticação do usuário no sistema.

- **SiglaProjeto_Caso_de_Uso_Gerenciamento_de_Tarefas** 1.0

Gerenciamento de tarefas, incluindo criação, edição e exclusão.

- **SiglaProjeto_Modelo_de_Dados_Tarefas** 1.0

Modelo de dados que define a estrutura das informações das tarefas, incluindo descrição, importância, data e hora, e se é compartilhada.

- **SiglaProjeto_Diagrama_de_Caso_de_Uso** 1.0

Diagrama que visualiza as interações entre o usuário e o sistema para autenticação e gerenciamento de tarefas.

- **SiglaProjeto_Interface_de_Usuário** 1.0

Esboços ou protótipos da interface de usuário para as telas de login, listagem de tarefas, criação de tarefas etc.

- **SiglaProjeto_Regras_de_Negócio** 1.0

Documento que descreve as regras de negócio relacionadas ao gerenciamento de tarefas, como validações de dados, tratamento de conflitos de datas, etc.

- **SiglaProjeto_Especificação_de_Interfaces** 1.0

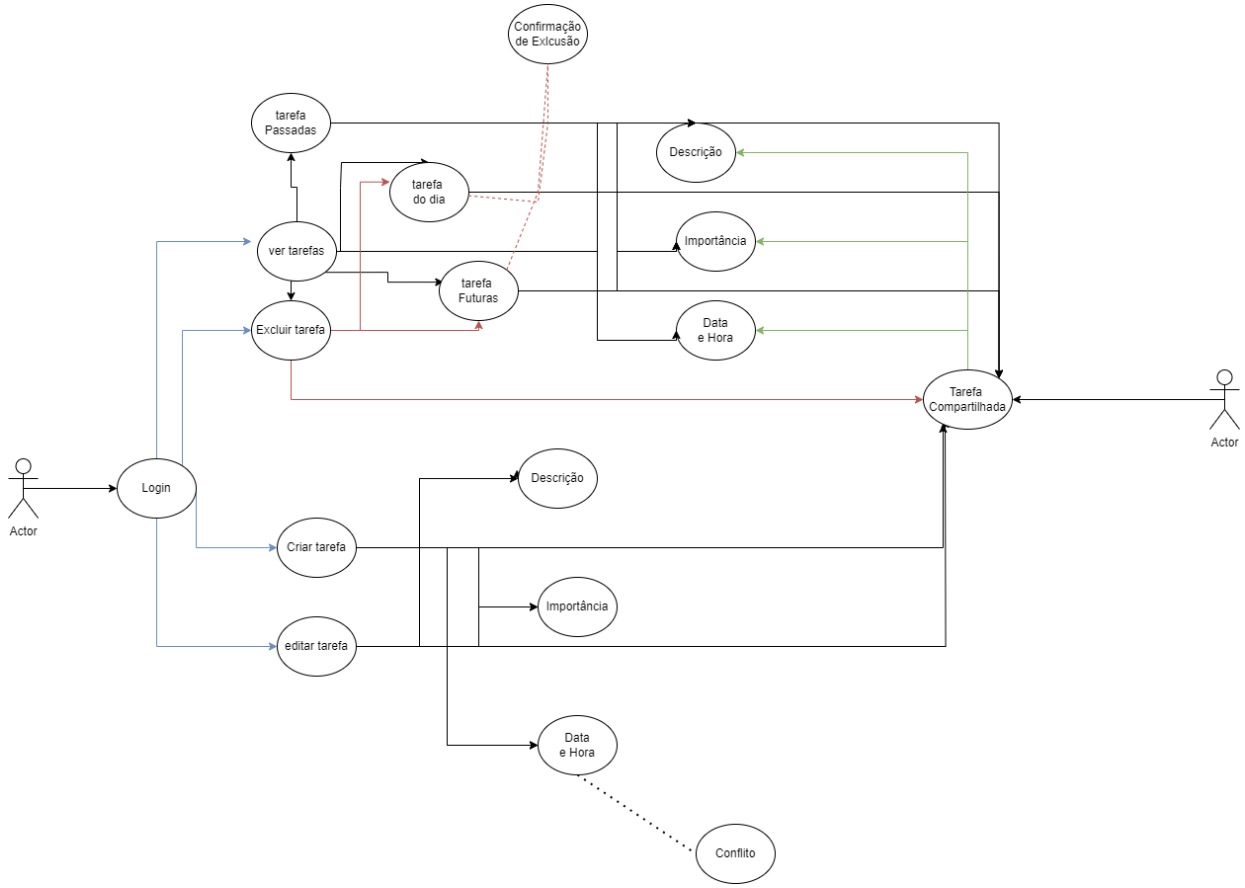
Detalhamento técnico das interfaces de usuário necessárias para a implementação do sistema, como APIs de autenticação e de manipulação de tarefas.

- **SiglaProjeto_Documento_de_Testes** 1.0

Documento que lista os casos de teste relacionados ao caso de uso, incluindo cenários de sucesso e de falha.

DIAGRAMAS DOS CASOS DE USO

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.



ATORES

- **Usuário Principal (Autor)**
  - Este ator representa o usuário principal do sistema, que interage diretamente com as funcionalidades de login e gerenciamento de tarefas. Ele é responsável por autenticar-se no sistema, visualizar, criar, editar e excluir tarefas.
    Além disso, podemos identificar os seguintes casos de uso relacionados aos atores:
- **Autenticação**
  - Este caso de uso envolve o ator "Usuário Principal" interagindo com o sistema para autenticar-se, fornecendo um nome de usuário e senha válidos.
- **Gerenciamento de Tarefas**
  - Este caso de uso envolve o ator "Usuário Principal" interagindo com o sistema para realizar diversas operações de gerenciamento de tarefas, como visualizar, criar, editar e excluir tarefas.

PRÉ-CONDIÇÕES

– O sistema de login deve estar funcionando corretamente.

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

- O usuário deve possuir um cadastro (Login) para poder ter acesso a lista de tarefas, e poder fazer as criações e edições necessárias.
- As operações de criação, edição e exclusão de tarefas devem estar disponíveis para o usuário principal.
- O sistema deve ter registrado as tarefas existentes, caso o usuário escolha visualizar tarefas passadas, tarefas do dia ou tarefas futuras.

FLUXO PRINCIPAL

Gerenciamento de Tarefas					
ID	**Passo**	**Fluxo**	**Regras**	**Msg**	**Tela**
1	O sistema apresenta a tela de login.		-	-	TL001
	O sistema valida as credenciais fornecidas.		RGN001 -	-	-
2	O sistema apresenta as opções: Ver Tarefas, Excluir Tarefas, Criar Tarefa e Editar Tarefa.	-		-	TL002 -
3	O usuário principal seleciona a opção "Ver Tarefas".	FA02			
3.1	O sistema apresenta as opções: Tarefas Passadas, Tarefas do Dia e Tarefas Futuras.		-	-	TL003
3.2	O usuário escolhe uma das opções de visualização de tarefas.	FA03			

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

3.3	O sistema exibe a lista de tarefas conforme a opção selecionada, contendo descrição, importância, data e hora, e se é compartilhada.	-	-	-	*TL004*
4	Caso o usuário opte por excluir uma tarefa:	*FA04* -	-	-	*TL005* -
4.1	O sistema permite ao usuário selecionar a tarefa a ser excluída.	*FA05*	*RGN002*	-	-
4.2	O sistema confirma a exclusão da tarefa selecionada.		-	*MSGS001*	-
4.3	O sistema remove a tarefa da lista.		-		-
5	Caso o usuário opte por criar uma tarefa:	*FA06*	-		*TL006* -
5.1	O sistema apresenta um formulário para inserção da descrição, importância e data e hora da nova tarefa.		-		*TL006* -
5.2	O usuário preenche os campos do formulário.	*FA07*	-		-
5.3	O sistema valida os dados inseridos.			*MSGS002*	
5.4	Caso haja um conflito de datas, o sistema alerta o usuário.	*FA08*		*MSGS003*	
5.5	O usuário confirma a criação da nova tarefa.			*MSGS004*	

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

5.6	O sistema adiciona a nova tarefa à lista.				
6	Caso o usuário opte por editar uma tarefa existente:	*FA09*			*TL007*
6.1	O sistema permite ao usuário selecionar a tarefa a ser editada.	*FA09*			*TL007*
6.2	O sistema apresenta um formulário pré-preenchido com os dados da tarefa selecionada.	*FA10*			*TL008*
6.3	O usuário realiza as edições necessárias nos campos do formulário.				
6.4	O sistema valida as alterações realizadas.	*FA11*		*MSGS004*	
6.5	Caso haja um conflito de datas, o sistema alerta o usuário.			*MSGS005*	
6.6	O usuário confirma as alterações.			*MSGS006*	
6.7	O sistema atualiza os dados da tarefa na lista.				
7	O usuário principal encerra a interação com as tarefas.				
8	O sistema exibe a lista de tarefas atualizada.				*TL004*

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

9	O usuário finaliza o caso de uso.				

FLUXOS ALTERNATIVOS E EXCEÇÕES

Gerenciamento de Tarefas					
ID	***Passo***	***Fluxo***	***Regras***	***Msg***	***Tela***
4	Caso o usuário opte por excluir uma tarefa:	*FA04*	-	-	*TL005*
4.1	O sistema permite ao usuário selecionar a tarefa a ser excluída.	*FA05*	*RGN002*	-	-
4.2	O sistema confirma a exclusão da tarefa selecionada.		-	*MSGS001*	-
4.3	O sistema remove a tarefa da lista.		-		-
5	Caso o usuário opte por criar uma tarefa:	*FA06*	-		*TL006* -

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

5.1	O sistema apresenta um formulário para inserção da descrição, importância e data e hora da nova tarefa.		-		TL006 -
5.2	O usuário preenche os campos do formulário.	FA07	-		-
5.3	O sistema valida os dados inseridos.			MSGS002	
5.4	Caso haja um conflito de datas, o sistema alerta o usuário.	FA08		MSGS003	
5.5	O usuário confirma a criação da nova tarefa.			MSGS004	
5.6	O sistema adiciona a nova tarefa à lista.				

6	Caso o usuário opte por editar uma tarefa existente:	FA09			TL007
6.1	O sistema permite ao usuário selecionar a tarefa a ser editada.	FA09			TL007
6.2	O sistema apresenta um formulário pré-preenchido com os dados da tarefa selecionada.	FA10			TL008
6.3	O usuário realiza as edições necessárias nos campos do formulário.				
6.4	O sistema valida as alterações realizadas.	FA11		MSGS004	

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

6.5	Caso haja um conflito de datas, o sistema alerta o usuário.			*MSGS005*	
6.6	O usuário confirma as alterações.			*MSGS006*	
6.7	O sistema atualiza os dados da tarefa na lista.				

PÓS-CONDIÇÃO OU RESULTADO ESPERADO

Ao final do caso de uso, o usuário visualiza a lista de tarefas atualizada conforme as operações realizadas durante a interação, incluindo adições, edições ou exclusões de tarefas. O sistema permanece disponível para que o usuário possa continuar interagindo com outras funcionalidades, como visualizar relatórios, realizar outras operações de gerenciamento, ou sair do sistema, conforme desejado.

Códigos Fontes

Controller

```
package com.labdesoft.roteiro01.controller;

import java.time.LocalDate;
import java.util.List;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

```
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.labdesoft.roteiro01.entity.Task;
import com.labdesoft.roteiro01.service.TaskService;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.parameters.RequestBody;

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

    @Autowired
    private final TaskService taskService;

    @Autowired
    public TaskController(TaskService taskService) {
        this.taskService = taskService;
    }

    @SuppressWarnings("null")
    @GetMapping("/past")
@Operation(summary = "Lista as tarefas passadas")
public ResponseEntity<List<Task>> getPastTasks() {
    try {
        List<Task> pastTasks = taskService.getAllTasks().stream()
                .filter(task -> task.getDate().isBefore(LocalDate.now()))
                .collect(Collectors.toList());

        if (pastTasks.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }
        return new ResponseEntity<>(pastTasks, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}


@SuppressWarnings("null")
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
@GetMapping("/today")
@Operation(summary = "Lista as tarefas do dia")
public ResponseEntity<List<Task>> getTodayTasks() {
    try {
        List<Task> todayTasks = taskService.getAllTasks().stream()
                .filter(task -> task.getDate().isEqual(LocalDate.now()))
                .collect(Collectors.toList());

        if (todayTasks.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }
        return new ResponseEntity<>(todayTasks, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}


@SuppressWarnings("null")
@GetMapping("/future")
@Operation(summary = "Lista as tarefas futuras")
public ResponseEntity<List<Task>> getFutureTasks() {
    try {
        List<Task> futureTasks = taskService.getAllTasks().stream()
                .filter(task -> task.getDate().isAfter(LocalDate.now()))
                .collect(Collectors.toList());

        if (futureTasks.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }
        return new ResponseEntity<>(futureTasks, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}


    @DeleteMapping("/{id}")
@Operation(summary = "Exclui uma tarefa pelo ID")
public ResponseEntity<HttpStatus> deleteTask(@PathVariable Long id) {
    try {
        taskService.deleteTask(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    } catch (Exception e) {
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
    }
}

@SuppressWarnings("null")
@PostMapping
@Operation(summary = "Cria uma nova tarefa")
public ResponseEntity<Task> createTask(@RequestBody Task task) {
    try {
        Task createdTask = taskService.createTask(task);
        return new ResponseEntity<>(createdTask, HttpStatus.CREATED);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}

@SuppressWarnings("null")
@PutMapping("/{id}")
@Operation(summary = "Atualiza uma tarefa existente pelo ID")
public ResponseEntity<Task> updateTask(@PathVariable Long id, @RequestBody
Task task) {
    try {
        Task updatedTask = taskService.updateTask(id, task);
        if (updatedTask == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(updatedTask, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}


}
```

Entity

```
package com.labdesoft.roteiro01.entity;

import io.swagger.v3.oas.annotations.media.Schema;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.Size;

import java.time.LocalDate; // Importe a classe LocalDate se ainda não
estiver importada

@Entity
@Getter
@Setter
@AllArgsConstructor
public class Task {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Schema(name = "Descrição da tarefa deve possuir pelo menos 10
caracteres")
    @Size(min = 10, message = "Descrição da tarefa deve possuir pelo menos
10 caracteres")
    private String description;

    private Boolean completed;

    // Adicione um atributo de data
    private LocalDate date;

    // Construtor vazio
    public Task() {}

    // Construtor com descrição, status de conclusão e data
    public Task(String description, Boolean completed, LocalDate date) {
        this.description = description;
        this.completed = completed;
        this.date = date;
    }

    // Método getter para a data
    public LocalDate getDate() {
        return date;
    }

    // Método setter para a data
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: "Como especificar casos de uso em 5 passos", "Como fazer um diagrama de casos de uso" e "Como documentar requisitos de software".

```
    public void setDate(LocalDate date) {
        this.date = date;
    }

    @Override
    public String toString() {
        return "Task [id=" + id + ", description=" + description + ",
completed=" + completed + "]";
    }
}
```

Repository

```
package com.labdesoft.roteiro01.repository;

import com.labdesoft.roteiro01.entity.Task;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

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

    // Método para salvar uma tarefa
    @SuppressWarnings("null")
    @Override
    <S extends Task> S save(S entity);

    // Método para encontrar uma tarefa por ID
    @SuppressWarnings("null")
    @Override
    Optional<Task> findById(Long id);

    // Método para encontrar todas as tarefas
    @SuppressWarnings("null")
    @Override
    List<Task> findAll();
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
    // Método para contar todas as tarefas
    @Override
    long count();

    // Método para excluir uma tarefa
    @Override
    void delete(@SuppressWarnings("null") Task entity);

    // Método para excluir uma tarefa por ID
    @Override
    void deleteById(@SuppressWarnings("null") Long id);
}
```

Service

```
package com.labdesoft.roteiro01.service;

import com.labdesoft.roteiro01.entity.Task;
import com.labdesoft.roteiro01.repository.TaskRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class TaskService {

    private final TaskRepository taskRepository;

    @Autowired
    public TaskService(TaskRepository taskRepository) {
        this.taskRepository = taskRepository;
    }

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

    public Task getTaskById(Long id) {
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
        Optional<Task> taskOptional = taskRepository.findById(id);
        return taskOptional.orElse(null);
    }

    public Task createTask(Task task) {
        return taskRepository.save(task);
    }

    public Task updateTask(Long id, Task updatedTask) {
        Task existingTask = getTaskById(id);
        if (existingTask != null) {
            updatedTask.setId(id);
            return taskRepository.save(updatedTask);
        }
        return null;
    }

    public void deleteTask(Long id) {
        taskRepository.deleteById(id);
    }
}
```

Roteiro01Application

```
package com.labdesoft.roteiro01;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;

@SpringBootApplication
@EntityScan("com.labdesoft.roteiro01.entity")
public class Roteiro01Application {
    public static void main(String[] args) {
        SpringApplication.run(Roteiro01Application.class, args);
    }
}
```

SQL

```
-- Drop table se existir
DROP TABLE IF EXISTS task;

-- Criação da nova estrutura da tabela task
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
CREATE TABLE task (
    id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(250) NOT NULL,
    completed BOOLEAN
);

-- Inserção dos dados iniciais
INSERT INTO task (description, completed) VALUES
('Primeira tarefa', false),
('Segunda tarefa', false),
('Terceira tarefa', false);
```

Roteiro01Application

```
package com.labdesoft.roteiro01;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.autoconfigure.domain.EntityScan;



@EntityScan("com.labdesoft.roteiro01.entity")
@SpringBootTest
class Roteiro01ApplicationTests {

    @Test
    void contextLoads() {
    }

}
```

SwaggerConfig

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
        return new Docket(DocumentationType.SWAGGER_2)
                .select()

.apis(RequestHandlerSelectors.basePackage("com.labdesoft.roteiro01.control
ler"))
                .paths(PathSelectors.any())
                .build();
    }
}
```

Organização de pastas

```
src
└── main
    ├── java
    │   └── com
    │       └── labdesoft
    │           └── roteiro01
    │               ├── controller
    │               │   └── TaskController.java
    │               ├── entity
    │               │   └── Task.java
    │               ├── repository
    │               │   └── TaskRepository.java
    │               ├── service
    │               │   └── TaskService.java
    │               └── Roteiro01Application.java
    └── resources
        ├── application.properties
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
        └── static
        └── templates
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.labdesoft</groupId>
    <artifactId>roteiro01</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>roteiro01</name>
    <description>roteiro 01 laboratório de Desenvolvimento de
Software</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies><dependency>
    <groupId>javax.activation</groupId>
    <artifactId>javax.activation-api</artifactId>
    <version>1.2.0</version>
</dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-boot-starter</artifactId>
            <version>3.0.0</version>
        </dependency>

        <dependency>
            <groupId>javax.persistence</groupId>
            <artifactId>javax.persistence-api</artifactId>
            <version>2.2</version>
        </dependency>

        <dependency>
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
        <groupId>javax.validation</groupId>
        <artifactId>validation-api</artifactId>
        <version>2.0.1.Final</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>2.2.222</version>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.32</version>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-boot-starter</artifactId>
        <version>3.0.0</version>
    </dependency>
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>2.4.0</version>
    </dependency>
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-ui</artifactId>
        <version>1.8.0</version>
    </dependency>
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>5.0.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>compile</scope>
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.

```
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

[1] Para entender melhor o que são requisitos de software e casos de uso, aconselhamos a leitura dos artigos: “Como especificar casos de uso em 5 passos”, “Como fazer um diagrama de casos de uso” e “Como documentar requisitos de software”.