UNIVERSIDADE DE FORTALEZA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: PROGRAMAÇÃO FUNCIONAL

Integrantes da equipe:

BRUNO MENESES DO NASCIMENTO - 2326703
ERICK LUIZ PEDRO DE FRANCA - 2326577
GEVERSON ARAUJO FERNANDES - 2326700
LUCIVALDO VIANA SILVA - 2326168

Link do projeto no github:

https://github.com/geversonfernandes/todo-list

Resumo do projeto:

O presente trabalho tem como objetivo desenvolver um Sistema de Gerenciamento de Tarefas (To-Do List) utilizando Java 21 e Spring Boot 3, com banco de dados em memória H2.

O sistema permite cadastrar, listar, concluir, excluir e buscar tarefas por palavra-chave, sendo acessado através de uma API REST.

Durante o desenvolvimento foram aplicados conceitos de programação funcional em Java, tais como lambdas, streams, closures e funções de alta ordem, conforme solicitado na proposta da atividade.

Além disso, foram elaborados testes automatizados (JUnit) e testes manuais (Postman) para garantir o correto funcionamento do sistema e o atendimento aos requisitos.

Requisitos do projeto:

Requisitos Funcionais

- RF01 – O sistema deve permitir cadastrar tarefas com descrição e prioridade.
  Implementado em TaskController.createTask() → chama TaskService.saveTask().

- RF02 – O sistema deve listar todas as tarefas cadastradas.
  Implementado em TaskController.getAllTasks() → chama TaskService.getAllTasks().

- RF03 – O sistema deve permitir marcar uma tarefa como concluída.
  Implementado em TaskController.completeTask(Long id) → chama TaskService.markAsCompleted(Long id).

- RF04 – O sistema deve permitir listar apenas tarefas concluídas.
  Implementado em TaskController.getCompletedTasks() → chama TaskService.getCompletedTasks().

- RF05 – O sistema deve permitir buscar tarefas por palavra-chave.
  Implementado em TaskController.searchTasks(String keyword) → chama TaskService.searchByKeyword(String keyword).

- RF06 – O sistema deve permitir excluir tarefas.
  Implementado em TaskController.deleteTask(Long id) → chama TaskService.deleteTask(Long id).

Requisitos Não Funcionais

- RNF01 – O sistema deve utilizar banco de dados em memória H2.
  Configurado em application.properties (spring.datasource.url=jdbc:h2:mem:todolistdb).

- RNF02 – O sistema deve ser implementado em Java 21 com Spring Boot.
  Projeto criado no Spring Initializr com Java 21 + dependências Spring Web e Spring Data JPA.

- RNF03 – O sistema deve expor endpoints REST para integração externa.
  Implementado em TaskController usando @RestController e mapeamentos @GetMapping, @PostMapping, @PutMapping, @DeleteMapping.

- RNF04 – O sistema deve adotar conceitos de programação funcional (uso de lambdas, streams).
  Lambda: tasks.stream().filter(task -> task.isCompleted()).
  List comprehension (Streams): tasks.stream().map(...).collect(Collectors.toList()).
  Closure: searchByKeyword(String keyword) → usa variável externa keyword dentro de Predicate<Task>.
  Função de alta ordem: filterTasks(Predicate<Task> filter) em TaskService.

- RNF05 – O sistema deve ser testado via Postman e possuir testes automatizados JUnit.
  Testes unitários implementados em TaskServiceTest (JUnit).
  Testes manuais realizados com Postman conforme seção de Casos de Teste.

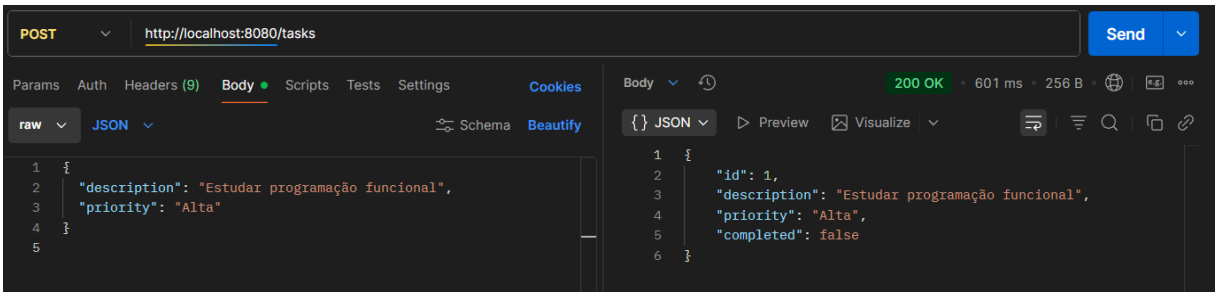
Casos de Teste Automatizados (JUnit)

- CT01 – Criar tarefa: verificar se a tarefa é salva corretamente.
- CT02 – Listar tarefas: garantir que a lista retornada não seja nula e contenha os itens esperados.
- CT03 – Marcar como concluída: verificar se o status completed muda para true.
- CT04 – Filtrar por palavra-chave: retornar apenas tarefas que contenham o termo pesquisado.

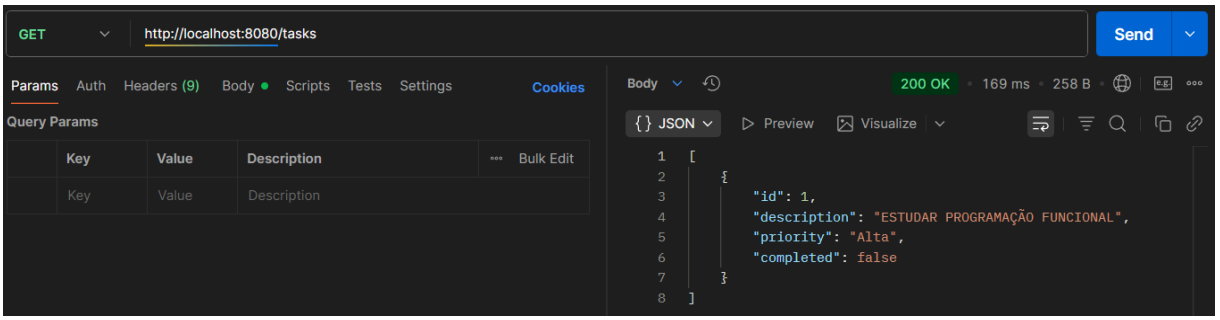
Casos de Teste Manuais (Postman)

Caso de teste	Método	Endpoint	Corpo	Resultado
CT01 – Criar tarefa	POST	/tasks	{"description": "Estudar programação funcional", "priority": "Alta"}	Retorna JSON da tarefa criada com id
CT02 – Listar todas	GET	/tasks	-	Lista todas as tarefas
CT03 – Marcar concluída	PUT	/tasks/1/complete	-	Retorna tarefa com completed=true
CT04 – Buscar concluídas	GET	/tasks/completed	-	Lista apenas as concluídas
CT05 – Buscar por palavra-chave	GET	/tasks/search?keyword=estudar	-	Retorna lista contendo tarefas com "estudar"
CT06 – Excluir tarefa	DELETE	/tasks/1	-	Retorna 204 No Content e tarefa é removida

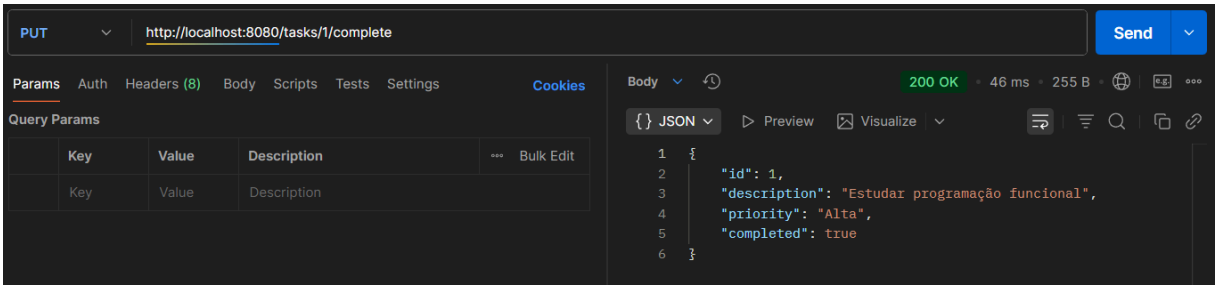
CT01:



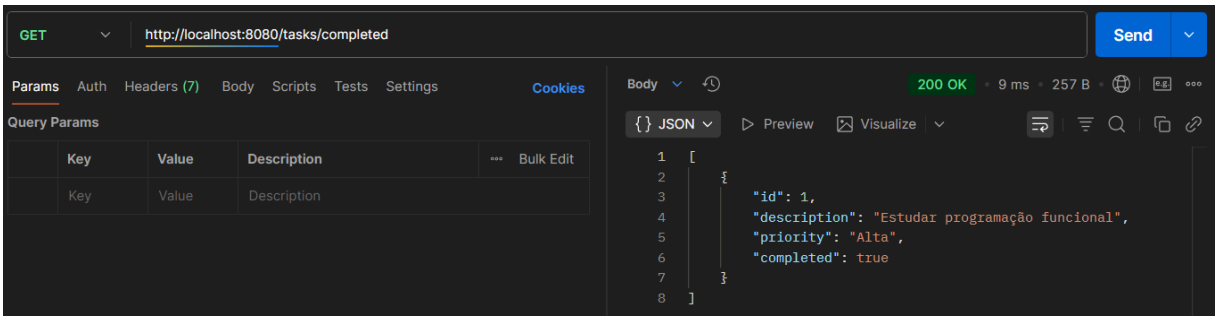
CT02:



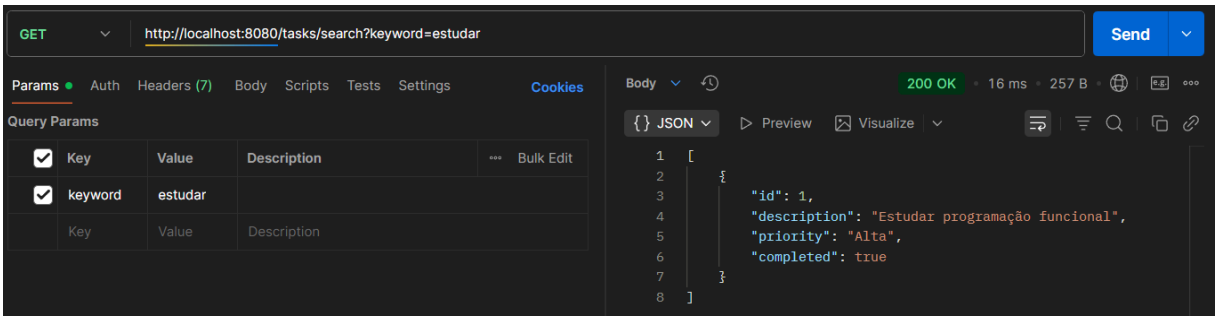
CT03:



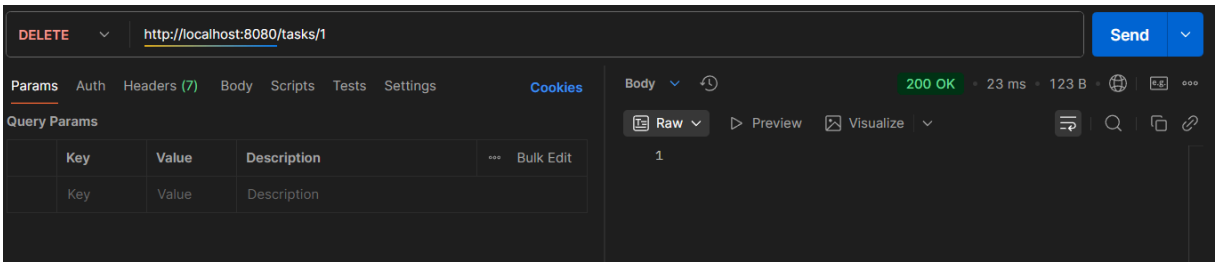
CT04:



CT05:



CT06:



Distribuição de papéis:

Lucivaldo Viana – Documentação dos Requisitos

- Criação do relatório.
- Diferenciou requisitos funcionais e não funcionais.
- Mapeou cada requisito para os métodos do código.
- Garantiu que os conceitos de programação funcional (lambda, stream, closure, função de alta ordem) estivessem descritos no documento.
- Responsável por revisar o documento para o GitHub.

Bruno Meneses – Implementação do Backend (Parte 1)

- Estruturou o projeto no Spring Initializr (Java 21, Gradle, Spring Boot, H2, Spring Data JPA, Web).
- Criou a entidade Task, o repositório TaskRepository e o service TaskService.
- Implementou os endpoints de:
  - Criar tarefa (POST /tasks)
  - Listar todas as tarefas (GET /tasks)
  - Excluir tarefa (DELETE /tasks/{id})

Erick Luis – Implementação do Backend (Parte 2)

- Desenvolveu o controller TaskController.
  Implementou os endpoints de:
  - Marcar tarefa como concluída (PUT /tasks/{id}/complete)
  - Listar concluídas (GET /tasks/completed)
  - Buscar por palavra-chave (GET /tasks/search?keyword=...)

- Aplicou os conceitos de programação funcional:
  - Uso de lambda e streams no filtro de busca.
  - Closure em função auxiliar de prioridade.
  - Função de alta ordem para aplicar filtros dinamicamente.

Geverson Araujo – Testes e Integração

- Implementou os testes automatizados JUnit (criar tarefa, listar, marcar concluída, buscar por palavra-chave).
- Construiu os casos de teste manuais no Postman (tabela).
- Validou a execução do projeto sem erros.
- Configurou o H2 Database no application.properties e verificou a persistência.
- Responsável por subir o projeto no GitHub com todos os arquivos e relatório.