

# **DOCUMENTAÇÃO DO SISTEMA TaskFlow API**

**Luiz Otávio Portes Avelar**

## **Introdução**

Este documento apresenta a análise de requisitos, arquitetura e tecnologias utilizadas no desenvolvimento da TaskFlow API, um sistema simples para gerenciamento de tarefas integrado com autenticação JWT.

O objetivo é fornecer uma visão clara sobre como o sistema funciona, quais problemas resolve e como foi projetado.

## **Objetivo do Sistema**

O sistema permite que usuários:

- Criem contas.
- Façam login recebendo um token JWT.
- Gerenciem suas tarefas pessoais.
- Tenham segurança, isolamento de dados e facilidade de uso.

É um sistema backend preparado para aplicações futuras (web, mobile ou desktop).

## **Público Alvo**

- Equipes de desenvolvimento
- Professores e avaliadores
- Estudantes analisando arquitetura de APIs

## Escopo do Projeto

O sistema tem:

- API REST
- Autenticação via JWT
- Persistência de dados com Entity Framework Core
- Uso de SQL Server rodando em *container Docker*

## Análise de Requisitos

### Requisitos Funcionais (RF)

		Descrição
<b>RF01</b>	Criar usuário	O sistema deve permitir criar usuários com email e senha.
<b>RF02</b>	Login	O usuário deve conseguir fazer login e receber um token JWT.
<b>RF03</b>	Listar tarefas	Permitir listar todas as tarefas do usuário autenticado.
<b>RF04</b>	Criar tarefa	Permitir criar uma nova tarefa associada ao usuário logado.
<b>RF05</b>	Atualizar tarefa	Permitir editar título, descrição e status.
<b>RF06</b>	Excluir tarefa	Permitir deletar uma tarefa existente.
<b>RF07</b>	Buscar tarefa por ID	Permitir buscar uma tarefa específica.
<b>RF08</b>	Garantir isolamento	O usuário só pode acessar suas próprias tarefas.

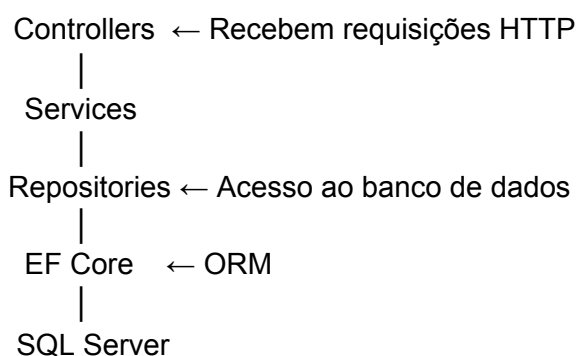
## Requisitos Não Funcionais (RNF)

		Descrição
<b>RNF01</b>	Segurança	A autenticação deve usar JWT com chave segura.
<b>RNF02</b>	Integridade	Banco de dados deve armazenar dados de forma consistente.
<b>RNF03</b>	Manutenibilidade	Código organizado em camadas (Controllers, Repositórios, Models).
<b>RNF04</b>	Desempenho	API deve responder rapidamente a comandos CRUD.
<b>RNF05</b>	Escalabilidade	O sistema deve poder crescer no número de usuários.
<b>RNF06</b>	Confiabilidade	API deve registrar erros adequadamente.
<b>RNF07</b>	Padronização	Endpoints REST obedecem boas práticas de nomes e rotas.
<b>RNF08</b>	Portabilidade	Fácil implantação em outros computadores via Docker.

## Arquitetura do Sistema

A arquitetura segue um padrão simples:

### Arquitetura em Camadas



## Fluxo de Autenticação JWT

1. Usuário envia email + senha
2. API valida credenciais
3. Gera um token JWT contendo o ID do usuário
4. Em cada nova requisição, o token é enviado no header
5. Middleware valida o token

## Componentes Principais

Componente	Função
<b>Controller</b>	Recebe e responde às requisições
<b>Repository</b>	Lida com o banco de dados
<b>Models</b>	Representam as entidades
<b>DTOs</b>	Estruturas para entrada/saída
<b>AppDbContext</b>	Conexão com BD via EF Core
<b>Swagger</b>	Documentação automática
<b>JWT Service</b>	Geração e validação de tokens

## Tecnologias Utilizadas

Tecnologia	Uso
<b>.NET 8 Web API</b>	Estrutura principal da aplicação
<b>C#</b>	Linguagem
<b>Entity Framework Core</b>	Acesso ao banco
<b>SQL Server (Docker)</b>	Banco de dados
<b>JWT Authentication</b>	Segurança

<b>BCrypt.Net</b>	Hash de senhas
<b>Swagger</b>	Testes e documentação
<b>Git + GitHub</b>	Controle de versão
<b>Docker Desktop</b>	Infraestrutura do SQL

## Modelos do Sistema

### Entidade Usuário

Id  
Nome  
Email  
SenhaHash  
CriadoEm

### Entidade Tarefa

Id  
Usuarioid  
Titulo  
Descricao  
Status  
CriadoEm

## Fluxo Geral de Uso

- 1) Criar conta
- 2) Fazer login e pegar token
- 3) Usar token no Swagger
- 4) Criar, listar, editar ou excluir tarefas

## Conclusão

A TaskFlow API é uma aplicação simples, escalável e segura, seguindo boas práticas de desenvolvimento de APIs REST.