

- 1 ☐ Vehicle Resale API
 - 1.1 Clean Architecture & Kubernetes Implementation
 - 1.2 ☐ Links Principais
 - 1.2.1 ☐ Repositório GitHub
 - 1.2.2 ☐ Vídeo Demonstrativo
 - 1.3 ☐ Arquitetura
 - 1.3.1 ☐ Clean Architecture
 - 1.3.2 ☐ Princípios SOLID
 - 1.4 ☐ Execução Local
 - 1.4.1 ☐ Docker Compose (Recomendado)
 - 1.5 ☐ Estrutura do Repositório
 - 1.6 ☐ Tecnologias Utilizadas
 - 1.7 ☐ Endpoints da API
 - 1.7.1 ☐ Veículos
 - 1.7.2 ☒ Health Check
 - 1.8 ☐ Demonstração em Vídeo
 - 1.9 ☐ Características da Solução
 - 1.9.1 ☐ Clean Architecture
 - 1.9.2 ☐ Princípios SOLID
 - 1.9.3 ☐ Containerização Completa
 - 1.9.4 ☐ Pronto para Produção

1 ☐ Vehicle Resale API

1.1 Clean Architecture & Kubernetes Implementation

API RESTful em .NET 8 implementando Clean Architecture e princípios SOLID para gerenciamento de veículos, com infraestrutura completa em Docker e Kubernetes.

1.2 ☐ Links Principais

1.2.1 ☐ Repositório GitHub

<https://github.com/ohntrebor/vehicle-resale>

1.2.2 ☐ Vídeo Demonstrativo

<https://youtu.be/ehMrxDCCR5k> (15 min)

1.3 ☐ Arquitetura

1.3.1 ☐ Clean Architecture

- **Domain:** Entidades, Value Objects, Interfaces
- **Application:** Use Cases, DTOs, Validações
- **Infrastructure:** EF Core, Repositórios, Serviços
- **Presentation:** Controllers, API, Middlewares

1.3.2 ☐ Princípios SOLID

- **S:** Single Responsibility Principle

- **O:** Open/Closed Principle
 - **L:** Liskov Substitution Principle
 - **I:** Interface Segregation Principle
 - **D:** Dependency Inversion Principle
-

1.4 Execução Local

1.4.1 Docker Compose (Recomendado)

```
git clone https://github.com/ohntrebor/vehicle-resale
cd vehicle-resale
docker compose up -d --build
```

Acesso: <http://localhost:5000/swagger>

1.5 Estrutura do Repositório

```
vehicle-resale/
├── README.md                # Documentação do projeto
├── Dockerfile               # Build da aplicação
├── docker-compose.yml       # Orquestração local
├── Makefile                 # Automação de comandos
├── VehicleResale.API/        # Controllers & Config
├── VehicleResale.Application/ # Use Cases & DTOs
├── VehicleResale.Domain/     # Entidades & Interfaces
├── VehicleResale.Infrastructure/ # EF Core & Repositories
├── k8s/                     # Manifestos Kubernetes
│   ├── namespace.yaml       # Namespace
│   ├── configmap.yaml       # Configurações
│   ├── secret.yaml          # Dados sensíveis
│   ├── api-deployment.yaml  # API Deployment
│   ├── api-service.yaml     # API Service
│   ├── postgres-deployment.yaml # DB Deployment
│   ├── postgres-service.yaml # DB Service
│   └── postgres-pvc.yaml    # Storage persistente
└── tests/                   # Testes automatizados
```

1.6 Tecnologias Utilizadas

- **.NET8** - Framework principal
 - **Entity Framework Core** - ORM
 - **PostgreSQL** - Banco de dados
 - **Docker** - Containerização
 - **Kubernetes** - Orquestração
 - **Swagger** - Documentação da API
-

1.7 Endpoints da API

1.7.1 Veículos






- GET /api/vehicles - Listar veículos
- GET /api/vehicles/{id} - Obter veículo por ID
- POST /api/vehicles - Criar novo veículo
- PUT /api/vehicles/{id} - Atualizar veículo
- DELETE /api/vehicles/{id} - Remover veículo

1.7.2 Health Check

- GET /health - Status da aplicação
 - GET /health/live - Liveness probe
 - GET /health/ready - Readiness probe
-

1.8 Demonstração em Vídeo

Link do YouTube: <https://www.youtube.com/watch?v=ehMrxDCCR5k>

O vídeo demonstra: -  Execução local com Docker Compose -  Deploy no Kubernetes com Minikube -  Funcionalidades da API -  Clean Architecture implementada -  Infraestrutura funcionando

1.9 Características da Solução

1.9.1 Clean Architecture

- Separação clara de responsabilidades
- Independência de frameworks
- Testabilidade facilitada
- Manutenibilidade aprimorada

1.9.2 Princípios SOLID


- Código bem estruturado
- Baixo acoplamento
- Alta coesão
- Facilidade de extensão

1.9.3 Containerização Completa

- Dockerfile otimizado
- Docker Compose para ambiente local
- Manifestos Kubernetes completos
- Alta disponibilidade

1.9.4 Pronto para Produção

- Health checks implementados
 - Configurações externalizadas
 - Logs estruturados
 - Monitoramento preparado
-

 **Solução completa implementando as melhores práticas de Arquitetura no desenvolvimento e DevOps**