**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Micro Serviços - Gerenciamento de Locação de Automóveis

Caio Gomes

Lucas Tondo Sendeski

Curitiba, Paraná

Dezembro 2021.

# Projeto Integrado – Arquitetura de Software Distribuído

Sumário

[● Projeto Integrado – Arquitetura de Software Distribuído 2](#_Toc100762963)

[1. Introdução 3](#_Toc100762964)

[2. Cronograma do Trabalho 5](#_Toc100762965)

[3. Especificação Arquitetural da solução 6](#_Toc100762966)

[3.1 Restrições Arquiteturais 7](#_Toc100762967)

[3.2 Requisitos Funcionais 7](#_Toc100762968)

[3.3 Requisitos Não-funcionais 8](#_Toc100762969)

[3.4 Mecanismos Arquiteturais 9](#_Toc100762970)

[4. Modelagem Arquitetural 9](#_Toc100762971)

[4.1 Diagrama de Contexto 10](#_Toc100762972)

[4.2 Diagrama de Container 10](#_Toc100762973)

[4.3 Diagrama de Componentes 11](#_Toc100762974)

## Introdução

Existem diversas maneiras de desenvolver softwares, desde técnicas simples até técnicas sofisticadas. O desenvolvimento de software está em constante evolução e as técnicas abrangentes para desenvolver também evoluem.

Não existe certo ou errado ao desenvolver um sistema, entretanto é possível aplicar algumas técnicas de desenvolvimento para extrair mais performance, usabilidade, escalabilidade do objetivo proposto.

Alguns sistemas acabam sendo desenvolvidos como monolito (“Obra construída em uma só pedra”) e enfrenta algumas dificuldades por ter essa arquitetura de projeto já enraizada.

Utilizar micro serviços na arquitetura de desenvolvimento traz alguns benefícios que auxiliam a equipe de desenvolvimento envolvida no projeto. É possível entregar mais em menos tempo com a facilidade e a separação do projeto em pequenos blocos o que proporciona melhor manutenibilidade.

As complexidades de utilizar micro serviços são poucas e sua arquitetura pode ser acoplada a *design-pattern* distintos com objetivo de melhorar ainda mais a entrega do software.

Nos dias atuais a locação de veículos está crescendo, seja para lazer, trabalho ou para uso pessoal. O processo habitual para locar um veículo é burocrático e oneroso, consiste em ir até uma locadora, escolher o veículo, preencher longos formulários e por fim locar o veículo.

A Loca-Car facilita o processo de locação, com a visualização rápida dos veículos disponíveis, agilidade na escolha e contratação do veicular.

O objetivo deste trabalho é separar as funcionalidades em micro serviços, ganhando performance, agilidade no desenvolvimento e maior desempenho em seu funcionamento atrelado com a facilidade de locar veículos. Com o particionamento dos serviços, cada micro serviço suporta uma carga maior comparado a um monolito e seu tempo de resposta é menor. Assim o sistema de locação garante maior escalabilidade e agilidade em seu funcionamento.

Será demonstrado o funcionamento de um micro serviço para locação de veículos, suas vantagens e desvantagens para o mercado de locação veicular.

Os objetivos específicos propostos são:

* Desenvolver uma aplicação para o controle de locação de veículos com conexão em três micro serviços.
* Desenvolver o micro serviço para controlar o cadastro de usuário existentes ou novos usuários via API com um *design-pattern* sofisticado *(CQRS).*
* Desenvolver o micro serviço para controlar o cadastro de veículos existentes ou novos via API com um *design-pattern* sofisticado *(CQRS).*
* Desenvolver o micro serviço para controlar o cadastro de aluguel de veículos novos ou existentes via API com um *design-pattern* sofisticado *(CQRS).*

## Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 01 / 01 / 22 | 10 / 04 / 22 | 1. Desenvolvimento do relatório técnico | Relatório técnico completo. |
| 01 / 01 / 22 | 15 / 03 / 22 | 1. Definição dos requisitos Arquiteturais | Listagem dos requisitos Arquiteturais |
| 01 / 01 / 22 | 15 / 03 / 22 | 1. Definição dos requisitos Funcionais | Listagem dos requisitos Funcionais |
| 01 / 01 / 22 | 1. 03 / 22 | 1. Definição dos requisitos  Não-Funcionais | Listagem dos requisitos Não Funcionais |
| 02 / 02 / 22 | 15 / 03 / 22 | 1. Definição dos Mecanismos Arquiteturais | Listagem dos Mecanismos Arquiteturais |
| 15 / 02 / 22 | 15 / 04 / 22 | 1. Apresentação do relatório técnico | Envio do relatório técnico completo. |
| 15 / 02 / 22 | 10 / 04 / 22 | 1. Desenvolvimento da apresentação visual do projeto. | Criação dos *wireframes* do projeto. |
| 10 / 04 / 22 | 15 / 04 / 22 | 1. Apresentação do visual do projeto | Apresentação dos *wireframes* criados. |
| 20 / 03 / 22 | 10 / 04 / 22 | 1. Desenvolvimento da prova de Conceito (POC) API - Usuário. | Criação do projeto inicial. |
| 20 / 03 / 22 | 10 / 04 / 22 | 1. Desenvolvimento da prova de Conceito (POC) API - Veículo. | Criação do projeto inicial e sua arquitetura com padrão CQRS |
| 20 / 03 / 22 | 10 / 04 / 22 | 1. Desenvolvimento da prova de Conceito (POC) API – Aluguel de Veículo. | Criação do projeto inicial e sua arquitetura com padrão CQRS |
| 25 / 03 / 22 | 15 / 04 / 22 | 1. Modelagem da arquitetura do projeto API - Usuário | Adicionar o *design-pattern* CQRS ao projeto. |
| 25 / 03 / 22 | 15 / 04 / 22 | 1. Modelagem da arquitetura do projeto API – Aluguel de Veículo. | Adicionar o *design-pattern* CQRS ao projeto. |
| 25 / 03 / 22 | 15 / 04 / 22 | 1. Modelagem da arquitetura do projeto API - Veículo. | Adicionar o *design-pattern* CQRS ao projeto. |

## Especificação Arquitetural da solução

O projeto será feito com três micro serviços:

* Serviço de controle de usuários.
* Serviço de controle de veículos.
* Serviço de controle de aluguel de veículos.

Ambos os micros serviços serão desenvolvidos com o *design-pattern* CQRS(*Command Query Responsibility Segregation*) com o objetivo de separar a responsabilidade de escrita e leitura dos dados do projeto.

Diagrama do projeto:



Figura 1- Diagrama do Projeto

## Restrições Arquiteturais

A arquitetura de micro serviço possui algumas restrições para não deixar de ser um micro serviço.

O micro serviço não pode ter diversas responsabilidades ou ser um projeto grande, tirando o objetivo de ser pequeno e objetivo.

R1: O micro serviço não deve ter muitas funcionalidades.

R2: O micro serviço não deve ter muitas responsabilidades

R3: O sistema será apenas para plataforma WEB

## Requisitos Funcionais

Cada micro serviço terá seu funcionamento apartado.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O Micro serviço de usuários, deve permitir cadastrar novos usuários no sistema validando se o usuário já não existe na plataforma. | B | A |
| RF02 | O Micro serviço de usuários, deve permitir editar os usuários já cadastrados no sistema. | B | A |
| RF03 | O Micro serviço de usuários, deve permitir realizar a exclusão dos usuários. | M | A |
| RF04 | O Micro serviço de usuários, deve permitir o usuário realizar o login na aplicação. | B | A |
| RF05 | O Micro serviço de usuários, deve autenticar o usuário por meio de um token JWT | M | A |
| RF06 | O Micro serviço de usuários, deve permitir que o usuário administrador bloqueie acessos de um usuário. | B | A |
| RF07 | O Micro serviço de usuários, deve permitir que o usuário administrador altere as permissões de outros usuários. | B | A |
| RF08 | O Micro serviço de usuários, deve validar os dados do usuário (e-mail, CPF e número de contato) | B | A |
| RF09 | O Micro serviço de veículos, deve permitir a inserção de novos veículos | M | A |
| RF10 | O Micro serviço de veículos, verificar se um novo veículo já existe na base de dados. | M | A |
| RF11 | O Micro serviço de veículos, deve permitir a edição dos veículos existentes | B | A |
| RF12 | O Micro serviço de veículos, deve permitir a exclusão lógica dos veículos existentes. Caso o veículo esteja alocado será apenas colocado uma tag com excluído para não aparecer em futuras alocações | B | A |
| RF13 | O Micro serviço de veículos, deve permitir a inserção do aluguel de veículos. O sistema deve verificar se o veículo ou usuário já possui uma locação ativa. Caso exista uma locação ativa não deve ser permitido uma nova locação | M | A |
| RF14 | O Micro serviço de veículos, deve permitir a exclusão lógica do aluguel de veículos. Apenas colocar uma data de exclusão para fins de histórico. | M | A |
| RF15 | O Micro serviço de veículos, deve permitir consultar quais veículos estão alocados | M | A |
| RF16 | O Micro serviço de veículos, deve validar se o veículo está precisando de manutenção preventiva | A | A |
| RF17 | O Micro serviço de veículos, deve validar se o usuário logado tem permissão para aprovar a locação de algum veículo, apenas usuários administradores podem aprovar uma solicitação de locação. | M | A |
| RF18 | O Micro serviço de veículos, deve notificar ao administrador que existe uma solicitação de locação. | M | A |
| RF19 | O Micro serviço de veículos, deve gerar relatórios dos veículos que estão sendo mais utilizados. | B | B |
| RF20 | O Micro serviço de veículos, deve validar um veículo que já está alocado e não permitir a locação do mesmo. Na consulta de veículos exibir ícone em vermelho de veículos alocados | B | A |
| RF21 | O sistema deve exibir apenas os menus de acordo com as permissões do usuário logado | B | M |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

Os requisitos não funcionais de ambos os micros serviços serão os mesmos e são:

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | O sistema deve ser apresentar disponibilidade 24 X 7 X 365 | A |
| RNF02 | O sistema deve utilizar o banco de dados MySql | A |
| RNF03 | O sistema deve ser intuitivo | A |
| RNF04 | O sistema deve ser fácil de utilizar | A |
| RNF05 | O sistema deve ser responsivo | A |
| RNF06 | O sistema deve utilizar o padrão de orientação a objetos sob a plataforma .NetCore | A |

**Obs**: acrescente quantas linhas forem necessárias.

## Mecanismos Arquiteturais

O sistema será dividido em três micro serviços, todos desenvolvidos em .Net Core com o *design-pattern* CQRS. O primeiro é responsável pelas informações que envolvem os usuários do sistema e a autenticação. O segundo será responsável pelas informações que envolvem os veículos do sistema. O terceiro será responsável pelas informações relacionadas aos veículos do sistema.

Todos os serviços partilharam do mesmo banco de dados. Cada micro serviço não irá acessar informações de outro serviço.

A usabilidade e o visual do sistema serão apresentados em Angular, consumindo as informações disponibilizadas pelos micros serviços. As funcionalidades e união das funções disponibilizadas pelos micros serviços serão imperceptíveis para o usuário.

Todos os serviços serão disponibilizados na AWS, utilizando o ECS e o S3 para o armazenamento e publicação dos serviços em um servidor.

|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | Banco de dados | MySql |
| Persistência | ORM | Dapper |
| Front end | Single Page Application | Angular |
| Back end | Micro serviços | .Net Core |
| Deploy | EC2/S3 | AWS |

## Modelagem Arquitetural

A modelagem arquitetural apresentada na seção 4.1 será utilizada em ambos os micros serviços.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para a documentação de arquitetura de software junto com componentes de serviços cloud no caso do banco de dados a ser utilizado.

## 4.1 Diagrama de Contexto

Figura 1 – Diagrama de Contexto

*Figura*

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais módulos e suas interfaces. Há duas personas:

* Administrador – Pode gerenciar funcionários e permite manipular os veículos.
* Funcionário – O usuário pode aprovar ou reprovar as solicitações de locação de veículo.

## 4.2 Diagrama de Container

O diagrama de container representa as formas de iterações dos usuários chaves da aplicação com o sistema. Nele podemos ter uma visão de como o sistema irá interagir entre si. Com esse diagrama temos a representação dos containers utilizados na aplicação e a interação entre eles.

Diagrama

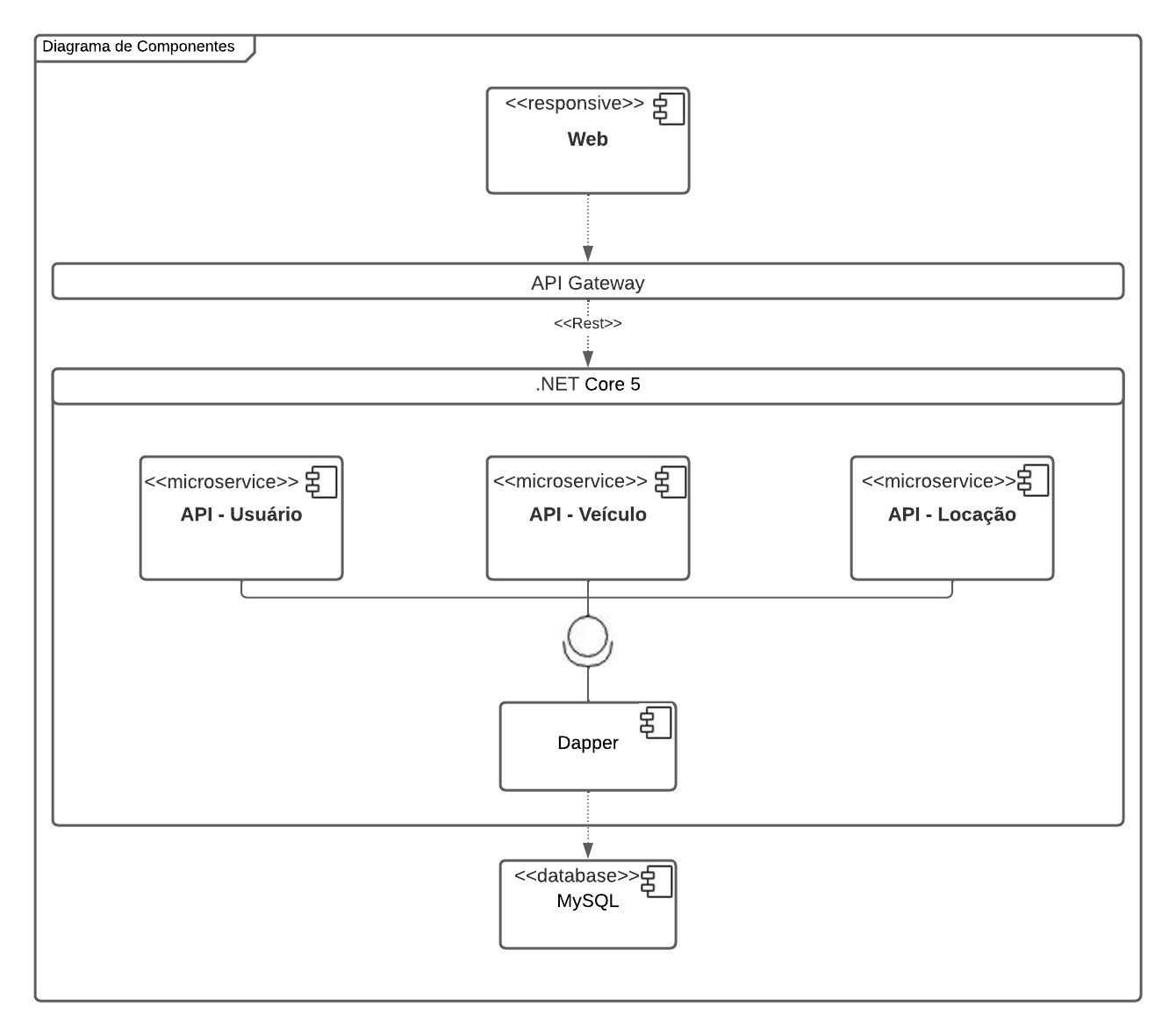
Descrição gerada automaticamente

Figura 2 - Diagrama de container

Figura 3 - Diagrama de Componentes

## 4.3 Diagrama de Componentes

O diagrama de componentes exibe desde a conexão com o banco de dados da aplicação até a exibição final na parte web do sistema e funciona da seguinte forma:

* **MySQL** – O banco de dados pode ser acessado por todos os micros serviços e cada API consome suas devidas informações.
* **API Usuário** – Faz o controle dos usuários do sistema.
* **API Veículo** – Faz o controle dos veículos do sistema.
* **API Locação** – Faz o controle das locações do sistema.

Video 1 - <https://drive.google.com/file/d/1d0_J1rhEUE0RSkxUw4LDVlgpwuiLpizv/view?usp=sharing>

Figma - <https://www.figma.com/file/0xSnft04Z6K80pi5OgB54q/LocaCar-Tcc>