

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
1. Introdução	3
2. Cronograma do Trabalho	5
3. Especificação Arquitetural da solução	6
3.1 Restrições Arquiteturais	7
3.2 Requisitos Funcionais	7
3.3 Requisitos Não-funcionais	8
3.4 Mecanismos Arquiteturais	9
4. Modelagem Arquitetural	9
4.1 Diagrama de Contexto	10
4.2 Diagrama de Container	11
4.3 Diagrama de Componentes	12
5. Análise das Abordagens Arquiteturais	13
5.1 Cenários	13
5.2 Evidências da Avaliação	18
5.3 Resultados Obtidos	21
6. Avaliação Crítica dos Resultados	21
7. Conclusão	23
Referências	24

1. Introdução

Existem diversas maneiras de desenvolver softwares, desde técnicas simples até técnicas sofisticadas, e ambas demandam constante revisão e atualização para que os sistemas se mantenham funcionais e ativos. Portanto, o desenvolvimento de softwares e os métodos envolvidos nesse processo estão em permanente evolução.

Ademais, não existe certo ou errado ao criar um sistema, entretanto, é possível aplicar algumas técnicas de desenvolvimento para extrair mais performance, usabilidade, escalabilidade do objetivo proposto, como por exemplo o método de avaliação Architecture Tradeoff Analysis Method (ATAM), que delimita parâmetros de qualidade que se deseja obter, a partir da análise de pontos críticos do processo de desenvolvimento e a posterior solução dos conflitos que podem surgir.

Contudo, alguns sistemas acabam sendo desenvolvidos como monolito (“obra construída em uma só pedra”), e enfrentam algumas dificuldades por ter essa arquitetura de projeto já enraizada. Em contrapartida, 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.

Além disso, as complexidades envolvendo a utilização de micro serviços são poucas, e sua arquitetura pode ser acoplada a *design-pattern* distintos, com objetivo de melhorar ainda mais a entrega do software, possibilitando seu emprego em aplicações para prestação de serviços, como por exemplo, a locação online de veículos.

Nesse viés, atualmente, a locação de veículos está em crescimento, seja para fins de lazer, uso pessoal ou até mesmo trabalho. O processo habitual para locar um veículo é burocrático e oneroso, pois consiste em ir até uma locadora, escolher o veículo, preencher longos formulários e, por fim, locar o veículo. Para flexibilizar e tornar mais dinâmico o processo para o locatário, a Loca-Car facilita a locação de veículos, possibilitando a visualização rápida dos automóveis disponíveis, agilidade na escolha e a posterior contratação.

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).

2. 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	2. Definição dos requisitos Arquiteturais	Listagem dos requisitos Arquiteturais
01 / 01 / 22	15 / 03 / 22	3. Definição dos requisitos Funcionais	Listagem dos requisitos Funcionais
01 / 01 / 22	15 / 03 / 22	4. Definição dos requisitos Não-Funcionais	Listagem dos requisitos Não Funcionais
02 / 02 / 22	15 / 03 / 22	5. Definição dos Mecanismos Arquiteturais	Listagem dos Mecanismos Arquiteturais
15 / 02 / 22	15 / 04 / 22	6. Apresentação do relatório técnico	Envio do relatório técnico completo.
15 / 02 / 22	10 / 04 / 22	7. Desenvolvimento da apresentação visual do projeto.	Criação dos <i>wireframes</i> do projeto.
10 / 04 / 22	15 / 04 / 22	8. Apresentação do visual do projeto	Apresentação dos <i>wireframes</i> criados.
20 / 03 / 22	10 / 04 / 22	9. Desenvolvimento da prova de Conceito (POC) API - Usuário.	Criação do projeto inicial.
20 / 03 / 22	10 / 04 / 22	10. Desenvolvimento da prova de Conceito (POC) API - Veículo.	Criação do projeto inicial e sua arquitetura com padrão CQRS
20 / 03 / 22	10 / 04 / 22	11. 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	12. Modelagem da arquitetura do projeto API - Usuário	Adicionar o <i>design-pattern</i> CQRS ao projeto.
25 / 03 / 22	15 / 04 / 22	13. Modelagem da arquitetura do projeto API – Aluguel de Veículo.	Adicionar o <i>design-pattern</i> CQRS ao projeto.
25 / 03 / 22	15 / 04 / 22	14. Modelagem da arquitetura do projeto API - Veículo.	Adicionar o <i>design-pattern</i> CQRS ao projeto.

3. 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 apresentado na figura 1.

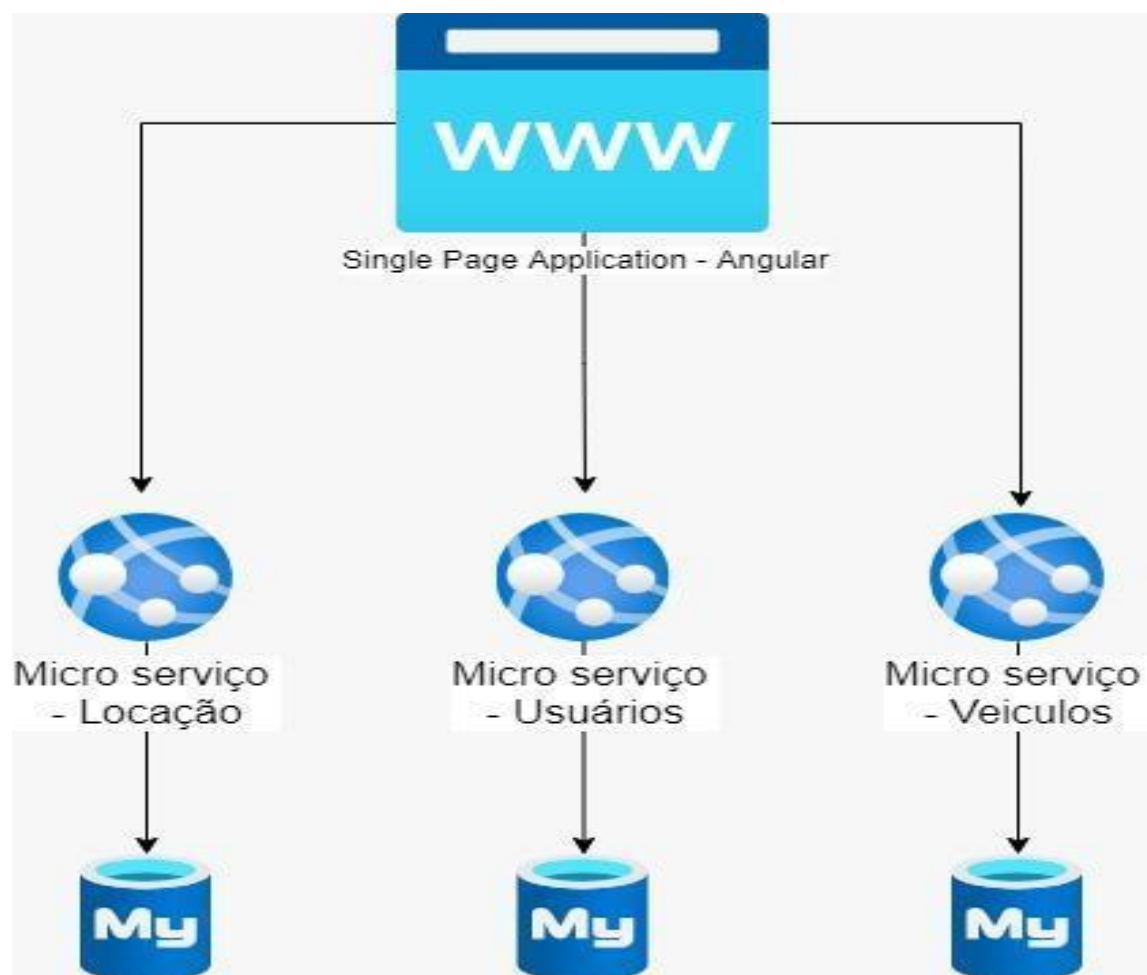


Figura 1- Diagrama do Projeto

3.1 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

3.2 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 <i>tag</i> 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.

3.3 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

3.4 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

4. 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

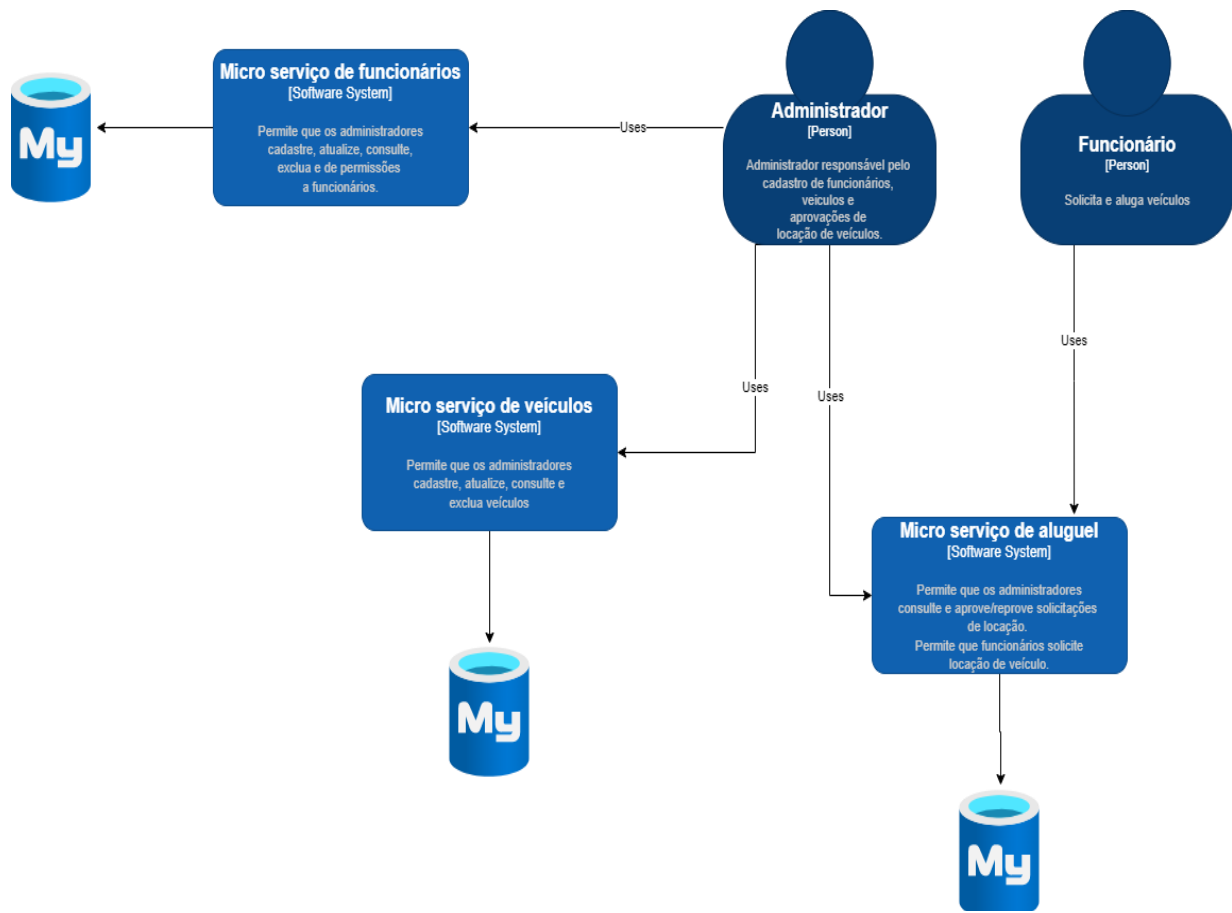


Figura 2 – Diagrama de Contexto

A figura 2 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 reprovando as solicitações de locação de veículo.

4.2 Diagrama de Container

O diagrama de container representa as formas de interações 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 conforme a figura 3.

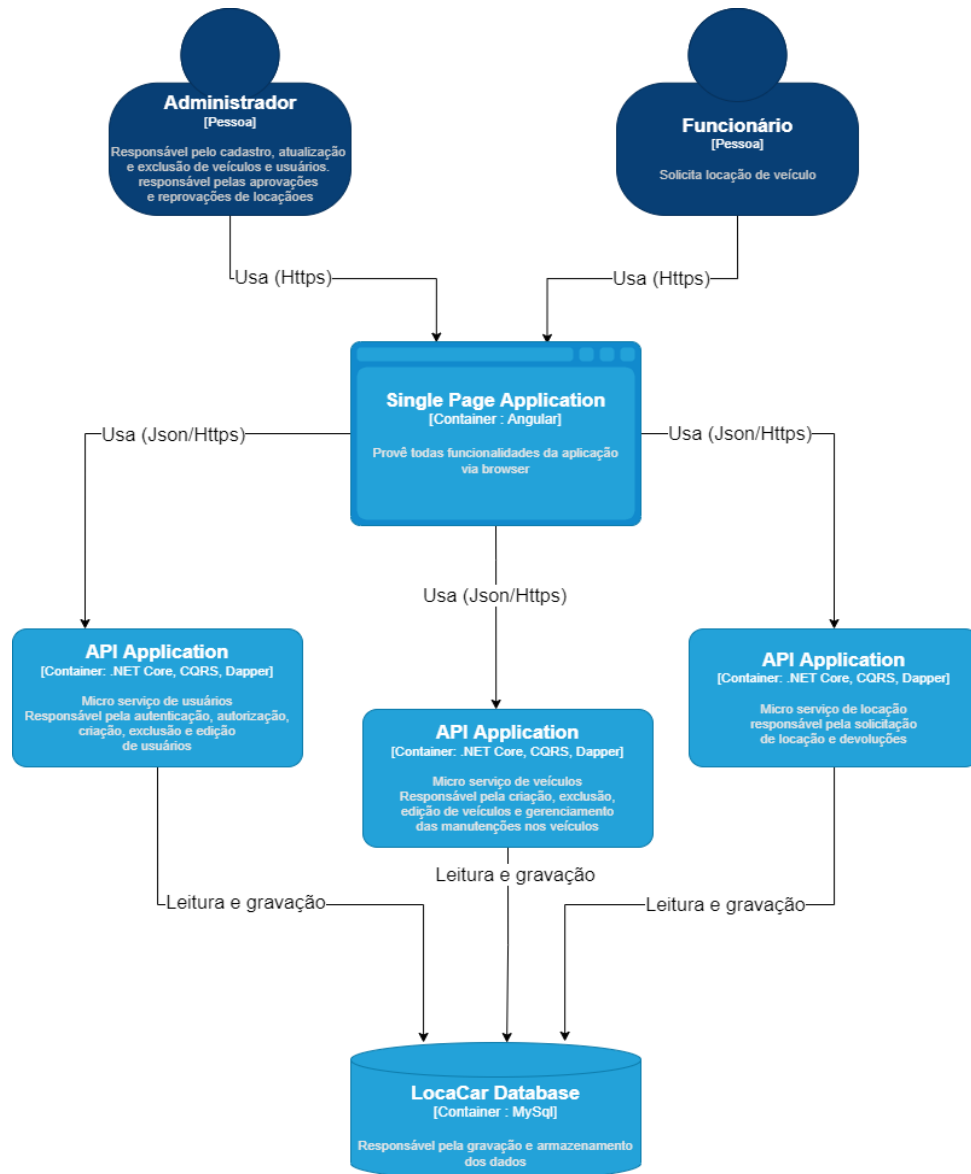
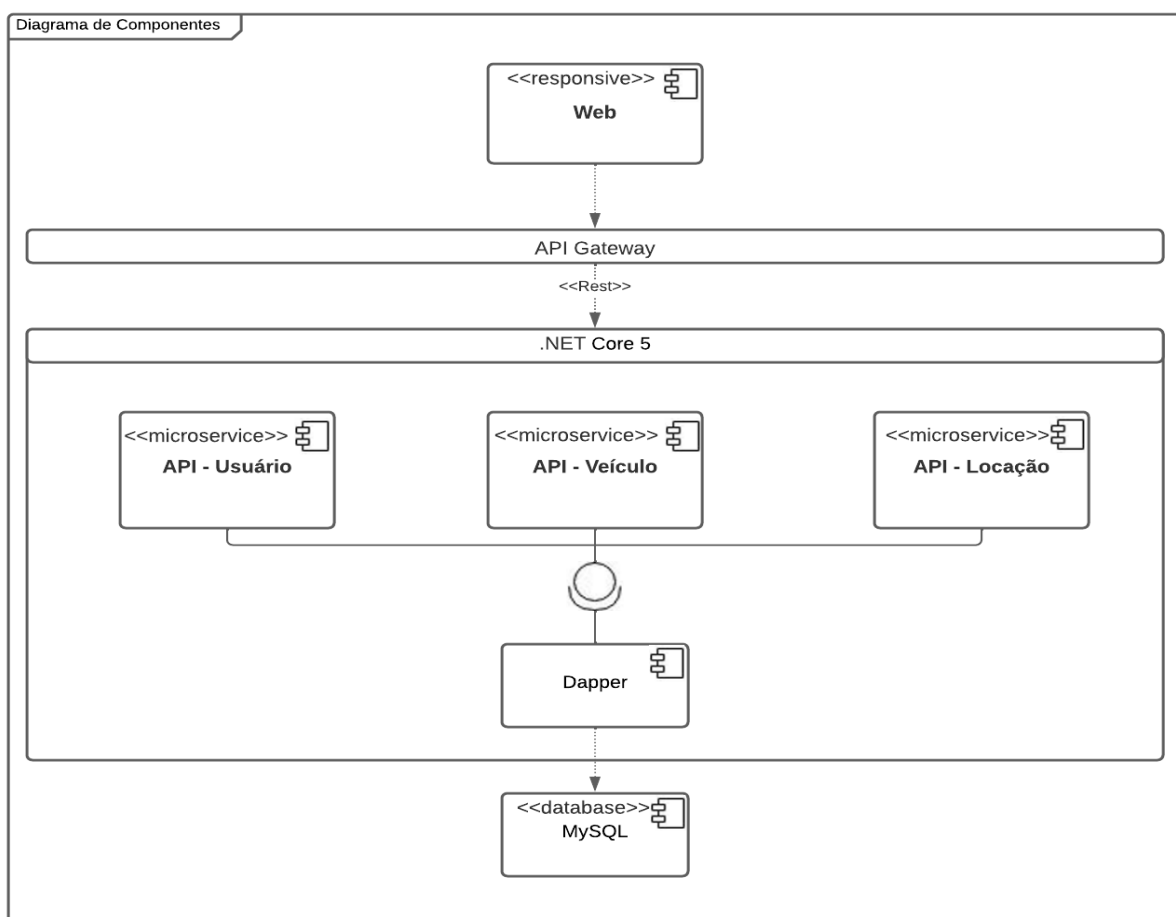


Figura 3 - Diagrama de container

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.



5. Análise das Abordagens Arquiteturais

A proposta arquitetural da locação de veículos tem como objetivo a escalabilidade e desempenho dos serviços utilizados. A utilização dos micros serviços fragmenta as requisições e mantém o desempenho caso tenha uma requisição ou milhares de requisições simultâneas.

Atributos de Qualidade	Cenários	Importância	Complexidade
Desempenho	Cenário 1: O Sistema deve ter desempenho nas requisições para qualquer micro serviço vinculado.	A	M
Interoperabilidade	Cenário 2: O sistema deve se comunicar com todos os micros serviços.	A	B
Usabilidade	Cenário 3: O sistema deve prover boa usabilidade.	M	B
Manutenibilidade	Cenário 4: O sistema deve ter a manutenção facilitada devido a divisão dos projetos.	M	M

5.1 Cenários

Cenário 1 – Desempenho: Qualquer requisição feita para os micros serviços de Usuário, Veículo ou Locação. As requisições são realizadas via HTTP e possuem o retorno em uma velocidade alta.

Uma requisição via HTTP Get em todos os usuários por exemplo possui um retorno de menos de 1 segundo, mesmo com sua base de dados estando em um servidor simples de teste conforme a figura 4 abaixo.

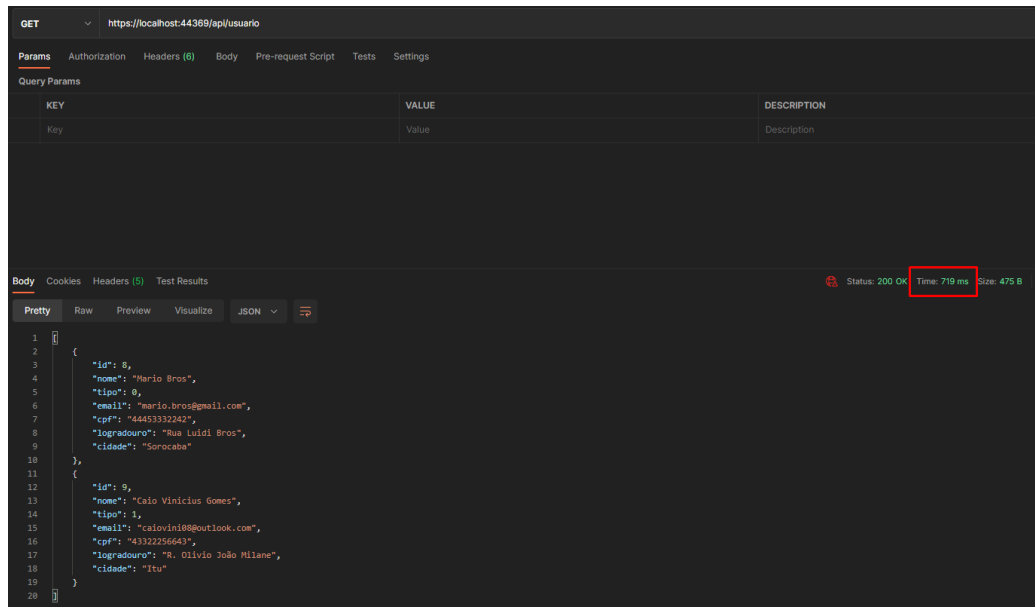


Figura 4 - Requisição HTTP Get em usuários

Cenário 2 – Interoperabilidade: A aplicação se comunica com todos os micros serviços via HTTP e essas comunicações podem ser de forma simultânea e assíncrona. Os micros serviços podem se comunicar entre si caso haja necessidade.

Cenário 3 – Usabilidade: A aplicação web fornece aos usuários uma boa usabilidade do sistema. A navegação e o acesso as funcionalidades do sistema são objetivas e apresentam clareza para o usuário, os formulários de cadastros estão simplificados e objetivos, facilitando o dia a dia do usuário final.

A tela inicial da aplicação exibe relatórios para o usuário com informações sobre o sistema conforme a figura 5.

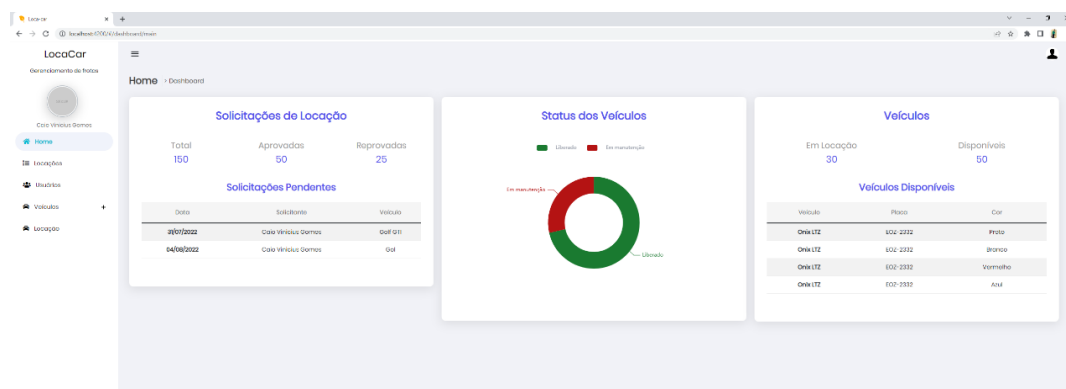


Figura 5 - Tela inicial da LocaCar

O Menu lateral do sistema é compacto e pode ser expandido conforme a necessidade do projeto. A figura 6 mostra um dos menus expandido para exemplificação.

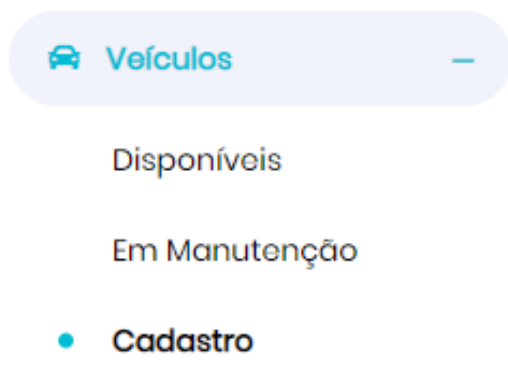


Figura 6 - Estilo do Menu da aplicação

As telas de cadastro estão padronizadas na exibição dos registros existente e no cadastro de novos registros. As figuras 7 e 8 mostram a tela de usuário e seu formulário de cadastro. A figura 9 mostra o formulário de locação no mesmo formato do usuário.

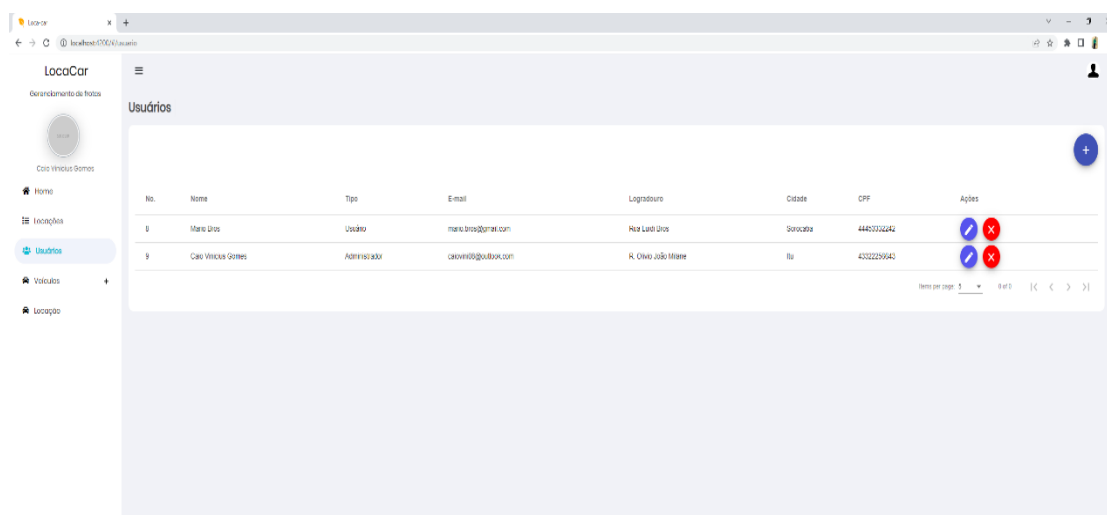


Figura 7 - Tela de Usuários

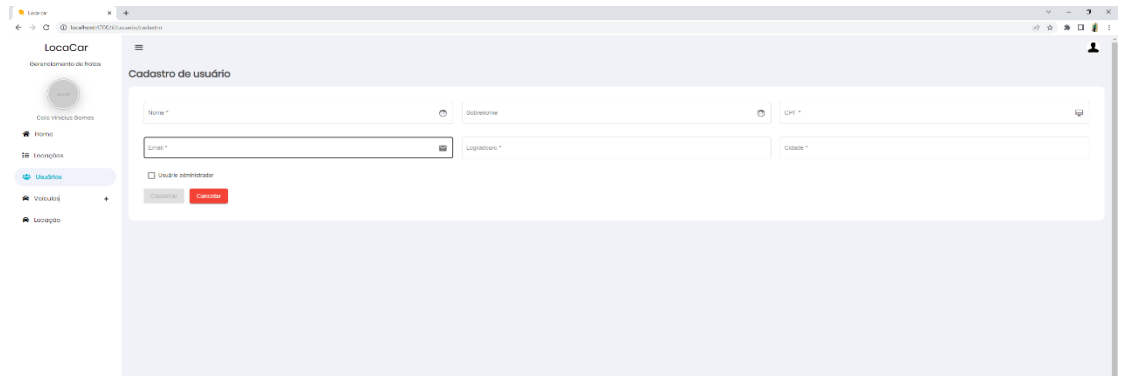


Figura 8 - Formulário de usuário

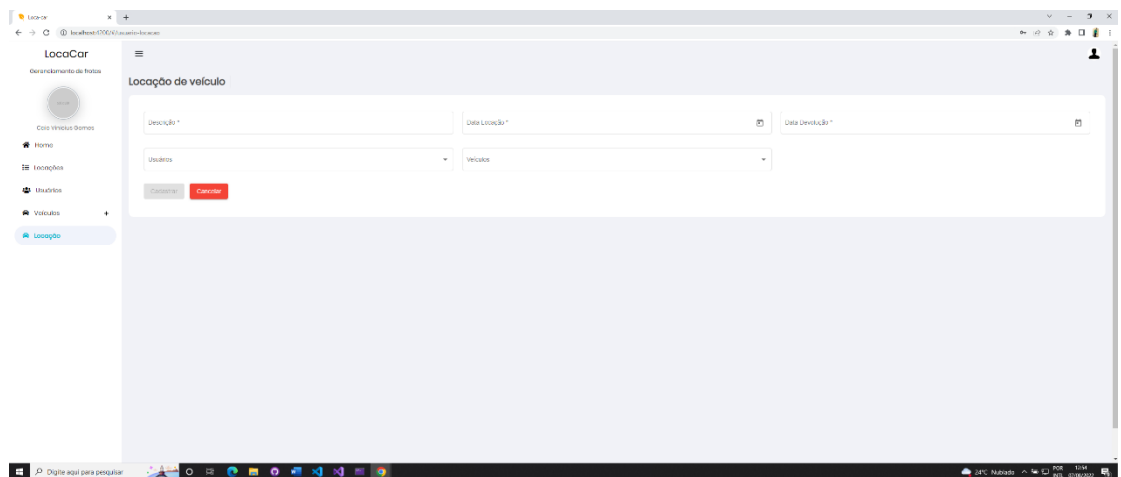


Figura 9 - Formulário de Locação

Cenário 4 – Manutenibilidade: O sistema possui três micros serviços, o que pode ser dividido em três equipes de desenvolvimento e/ou três equipes de sustentação, obtendo muita vantagem na manutenção e desenvolvimento de novas *features* para o projeto.

O sistema é de fácil manutenção e suas correções são ágeis. Pensando nisso o desenvolvimento do sistema web foi elaborado em pequenos componentes onde cada um é responsável por uma parte da tela facilitando a manutenção e futuras modificações.

A figura 10 mostra como os componentes estão sendo armazenados na estrutura do projeto da aplicação. A figura 11 mostra a facilidade em chamar um componente existente em qualquer parte do *front-end* e a figura 12 mostra o quão simplificado fica o código de um componente pronto de forma enxuta.

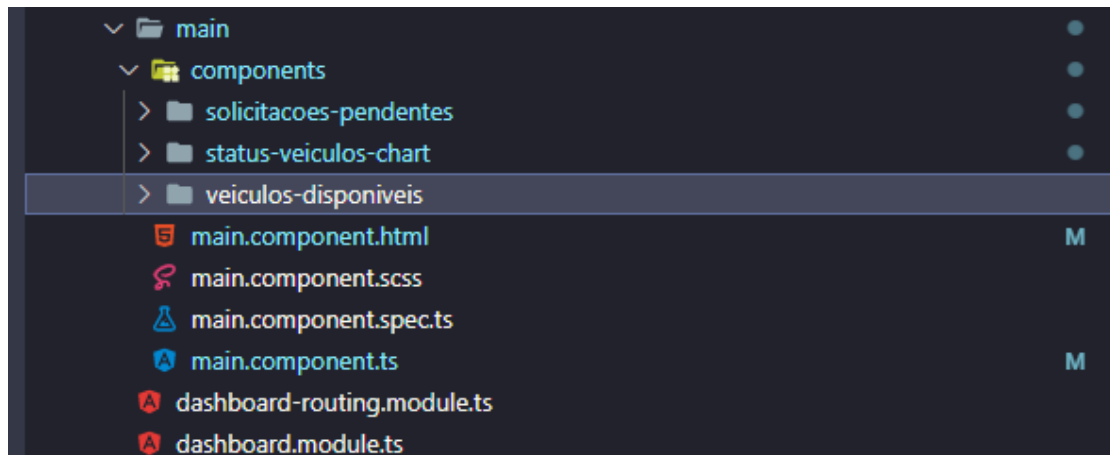


Figura 10 - Componentes do front-end

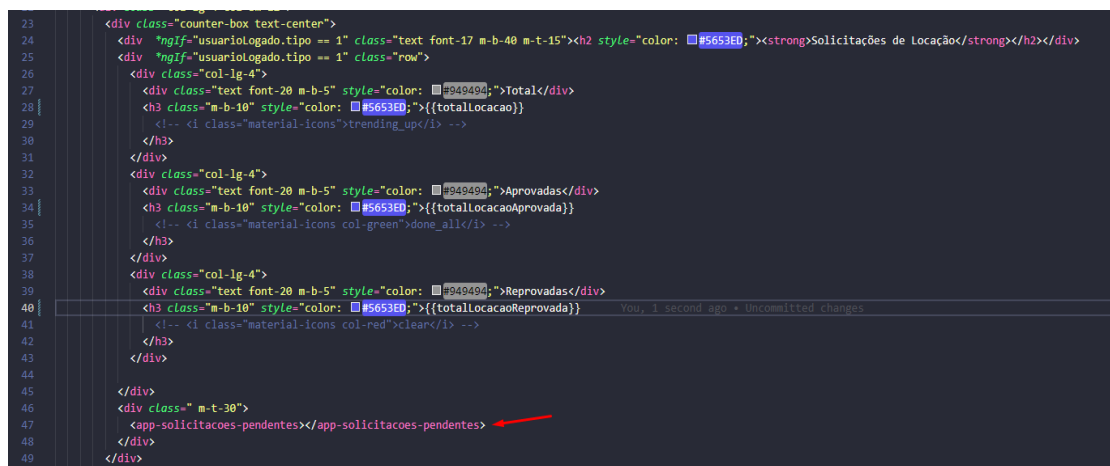


Figura 2 - Chamada de um componente



Figura 12 - Exemplo de componente

5.2 Evidências da Avaliação

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deve ter desempenho e velocidade na resposta as requisições.
Preocupação: Conforme o número de requisições simultâneas aumenta a tendência a velocidade de resposta é diminuir	
O sistema deve ter como resposta a uma ou mais requisições de forma rápida e eficiente.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
A aplicação se comunica com todos os micros serviços e faz as requisições via HTTP conforme sua necessidade.	
Mecanismo:	
Realizar uma requisição aos usuários existentes por exemplo via HTTP GET.	
Medida de resposta:	
Retornar os dados requisitados no formato JSON	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a conexão lenta ou mesmo a perda de pacotes.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve se comunicar com outras tecnologias.

Preocupação:	
O sistema deve ter como resposta a uma requisição uma saída de fácil leitura por outro componente.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O sistema de monitoramento envia uma requisição para o serviço REST do módulo de informações gerenciais.	
Mecanismo:	
Criar um serviço REST para atender às requisições do sistema de monitoramento	
Medida de resposta:	
Retornar os dados requisitados no formato JSON	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a conexão lenta ou mesmo a perda de pacotes.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve apresentar uma interface de fácil usabilidade para o usuário
Preocupação:	
O sistema deve ter uma interface de fácil utilização e objetiva assim facilitando a usabilidade do sistema	
Cenário(s):	
Cenário 3	
Ambiente:	
Navegador web (Google Chrome)	

Estímulo:	
Não há	
Mecanismo:	
Sistema web desenvolvido em Angular no formato de SPA e responsivo	
Medida de resposta:	
Interface web	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a conexão lenta assim afetando a usabilidade
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Atributo de Qualidade:	Manutenibilidade
Requisito de Qualidade:	O sistema apresenta uma estrutura de fácil manutenção para os desenvolvedores
Preocupação:	
O sistema deve ter uma estrutura de componentes que agrupados irão compor uma página assim ficando cada componente responsável por uma parte da tela. Visando a facilidade na manutenção futura do sistema	
Cenário(s):	
Cenário 4	
Ambiente:	
Visual Studio Code (Ambiente Local)	
Estímulo:	
Não há	
Mecanismo:	
Estrutura de pastas e componentes definida pelo time de desenvolvimento	
Medida de resposta:	
Estrutura de fácil manutenção do projeto	
Considerações sobre a arquitetura:	

Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

5.3 Resultados Obtidos

Requisitos Não Funcionais	Teste	Homologação
RNF01: O sistema deve ter alto desempenho.	OK	OK
RNF02: O sistema deve se comunicar com todos os micros serviços.	OK	OK
RNF03: O sistema deve ter uma boa usabilidade	OK	OK
RNF04: O sistema deve ter uma boa manutenibilidade	OK	OK

6. Avaliação Crítica dos Resultados

Ponto avaliado	Descrição
Desempenho	A divisão de um sistema em micro serviço é também a divisão das requisições realizadas por uma aplicação. Considerando um grande número de requisições, os serviços conseguem entregar respostas de forma mais eficiente.
Agilidade no desenvolvimento	Com a divisão do projeto é possível ter um time para cada parte do sistema, tendo entregas com mais agilidade.
Usabilidade da aplicação	O sistema apresenta uma usabilidade simplificada para facilitar o dia a dia do usuário final.

Agilidade na Locação	O sistema previamente configurado permite que a locação de um veículo seja feita de forma rápida e prática.
Cadastro de informações do sistema	Os formulários do sistema são íntegros e intuitivos, facilitando a população de dados dentro da aplicação.
Visibilidade de informações	A tela inicial apresenta alguns relatórios do sistema, trazendo agilidade para o administrador em coletar dados do dia a dia referente a locações.

7. Conclusão

Ao final deste trabalho foi desenvolvido uma PoC para um sistema de locação de veículos com tecnologias recentes do mercado e metodologias que visaram a eficiência e desempenho do projeto.

Existem diversas maneiras de desenvolver uma aplicação sendo difícil afirmar qual é a mais completa ou eficiente, porém, atribuímos ao projeto apresentado a opção que consideramos mais válida para o cenário utilizado.

Realizar a divisão do *back-end* em micro serviços com o *design pattern* CQRS foi a alternativa encontrada para ter agilidade em desenvolver e desempenho para a aplicação, fragmentando as requisições do sistema e a responsabilidade de cada serviço.

Trabalhar com locação de veículos envolve diversos fatores, normas, regras, contrato e etc... Foi encontrado o essencial e básico para esse cenário com a proposta de simplicidade e fácil usabilidade para os usuários do sistema.

A utilização dessas tecnologias com o *design pattern* escolhido foi desafiante nesse projeto. Aprendemos em como fragmentar os projetos, como arquiteta-los e como podemos realizar a divisão técnica do código ou qual forma ele deve ser estruturado nesse contexto.

Todo projeto pode ser melhorado ou evoluído e este projeto é um deles. Futuramente pode ser implementado funcionalidades no ramo de locação, relatórios mais detalhados, emissão do contrato final, assinatura digital e etc...

Aprendemos com esse projeto em como utilizar micros serviços e qual sua finalidade em um determinado cenário, isso trouxe o questionamento de qual padrão de projeto utilizar, qual a melhor prática de arquitetura desse cenário e como devemos apresentar a aplicação final para o usuário de forma simples.

As principais dificuldades encontradas durante o desenvolvimento do trabalho foram definir qual arquitetura utilizar e qual seria a metodologia que se encaixaria com nosso cenário. Tivemos dificuldade em criar uma aplicação com um visual agradável e de boa interação com o usuário final que se comunicasse com todos os serviços de forma simultânea.

Referências

BALARINE, Oscar Fernando Osorio. **Tecnologia da Informação como Vantagem Competitiva**. Revista de Administração Eletrônica. Vol. 1. N 1. São Paulo. Jan./Jun. 2002. Disponível em: <<http://www.scielo.br/pdf/raeel/v1n1/v1n1a05.pdf> >. Acesso em 12/07/2022.

PONTES, Danielle P. Noronha. ARAKAKI, Reginaldo. **Evolução de software baseada em avaliação de Arquitetura de Software**. In: **XVII Congreso Argentino de Ciencias de la Computación**. Disponível em: <<https://san.uri.br/sites/anais/Stin/trabalhos/04.pdf>>. Acesso em 13/07/2022.

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

Figma - <https://www.figma.com/file/0xSnft04Z6K80pi5OqB54q/LocaCar-Tcc>

Video Final -

<https://drive.google.com/file/d/14GF391LMPwBEP8EzC0WNNn17GBL8-dHeb/view?usp=sharing>

Github - <https://github.com/Sendeskill/loca-car>