

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

Pós-graduação Lato Sensu em Arquitetura de Software Distribuído

Projeto Integrado

Relatório Técnico

Sistema Balcão de Atendimento - Aluga Carros

Carlos Eduardo Mattos Barreto Junior

Belo Horizonte

Junho de 2022

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	7
3.1 Restrições Arquiteturais	7
3.2 Requisitos Funcionais	7
3.3 Requisitos Não-funcionais	9
3.4 Mecanismos Arquiteturais	10
4. Modelagem Arquitetural	11
4.1 Diagrama de Contexto	11
4.2 Diagrama de Container	12
4.3 Diagrama de Componentes	13
5. Prova de Conceito (PoC)	15
5.1 Integrações entre Componentes	15
5.2 Código da Aplicação	15
6. Avaliação da Arquitetura (ATAM)	17
6.1. Análise das abordagens arquiteturais	17
6.2. Cenários	17
6.3. Evidências da Avaliação	18
6.4. Resultados Obtidos	19
7. Avaliação Crítica dos Resultados	20
8. Conclusão	21
Referências	22

1. Introdução

O setor de locação de veículos vem crescendo muito nos últimos anos, segundo a ABLA – Associação Brasileira de Locadoras de Automóveis, o setor teve um aumento de 33,5% em 2021. O faturamento bruto do setor saltou de R\$ 17,6 bilhões em 2020 para R\$ 23,5 bilhões em 2021. Com a recente alta dos valores dos seminovos devido a inflação e época em que vivemos, empresas do ramo estão aumentando seus ganhos com locação de veículos e venda da frota como seminova, diante disto, a empresa fictícia Aluga Carros decidiu iniciar a modernização de suas aplicações, utilizando tecnologias mais recentes e migrando-os para a nuvem.

A empresa está passando por uma transformação digital, hoje em seu parque de sistemas, a Aluga Carros LTDA dispõe de vários sistemas para suportar suas diversas operações, que vão desde ferramentas de mercado como SAP, até sistemas de desenvolvimento próprio. Devido ao grande aumento da oferta de diversos produtos de locação, atendendo diferentes públicos, e seu grande crescimento nos últimos anos, a empresa observou a necessidade de migrar seus sistemas para uma tecnologia mais recente para acompanhar a evolução do mercado. Uma destas necessidades é a criação de um sistema exclusivo para seu Balcão de Atendimento nas agências, visto que o sistema legado não dispõe de recursos para acessibilidade, é pouco amigável, lento, além do que é um monolito complexo que dispõe de muitas funcionalidades não necessárias ao balcão de atendimento e com várias funcionalidades espalhadas, tornando o processo de abertura de locações moroso e complexo.

O novo sistema agrupará de forma prática se intuitiva todas as necessidades dos atendentes que o operam, e este irá dispor de acessibilidade, boa usabilidade, e será desenvolvido em uma plataforma moderna. Algumas das funcionalidades serão por exemplo, acompanhar a situação de veículos, realização de novas locações, verificação de futuras reservas, informações estas que no sistema atual estão espalhadas passarão a estar centralizadas e serem utilizadas de maneira objetiva.

O objetivo deste trabalho é apresentar a descrição do projeto arquitetural do Sistema Balcão de Atendimento para a empresa Aluga Carros LTDA, que após um estudo de mercado sobre os produtos já existentes, definiu 3 principais objetivos para o seu novo sistema que são:

Sistema Balcão de Atendimento - Aluga Carros

- Deve ser uma solução totalmente integrada aos demais sistemas já existentes na empresa e compatibilizar as novas informações nos sistemas legados.
- Deve utilizar uma interface amigável, com acessibilidade e de fácil entendimento e uso.
- Deve estar apta a funcionar em ambientes *Cloud*, ser segura, observável, escalável, resiliente e tolerante a falhas, para suportar a expansão da empresa no decorrer dos anos.

2. Cronograma do Trabalho

No Quadro 1 é apresentado o cronograma proposto para as etapas deste trabalho.

Quadro 1 – Cronograma de Atividades

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
01/05/2022	03/05/2022	1.Introdução	Construção da Introdução
04/03/2022	05/05/2022	2.Cronograma do Trabalho	Criação desta tabela
06/05/2022	09/05/2022	3.Especificação Arquitetural	Levantamento da Especificação Arquitetural
09/05/2022	13/05/2022	4. Restrições Arquiteturais	Levantamento de Restrições Arquiteturais
14/05/2022	17/05/2022	5. Requisitos Funcionais	Levantamento dos requisitos Funcionais
18/05/2022	21/05/2022	6. Requisitos Não Funcionais	Levantamento dos Requisitos Não Funcionais
22/05/2022	31/05/2022	7. Mecanismos Arquiteturais	Levantamento dos Mecanismos Arquiteturais
01/06/2022	05/06/2022	8.Criação do Diagrama de Contexto	Diagrama de Contexto Criado no modelo C4
06/06/2022	10/06/2022	9. Criação do Vídeo de apresentação do projeto	Gravação do Vídeo de apresentação
11/06/2022	15/06/2022	10. Revisão e envio da primeira etapa do projeto	Revisão e postagem da primeira etapa do trabalho
16/06/2022	22/06/2022	11. Criação do diagrama de Container	Criação do Diagrama de Container
23/06/2022	30/06/2022	12. Criação do Diagrama de Componentes	Criação do Diagrama de Componentes
01/07/2022	05/07/2022	13. Criação de Wireframes da POC	Criação dos Wireframes da POC
06/07/2022	06/08/2022	14. Criação do código da aplicação	Criação da aplicação com os 3 requisitos principais implementados
07/08/2022	15/08/2022	15. Revisão e envio da segunda etapa do projeto	Revisão e postagem da segunda etapa do trabalho
16/08/2022	22/08/2022	16. Análise das abordagens arquiteturais	Criação da documentação das análises das abordagens

Sistema Balcão de Atendimento - Aluga Carros

23/08/2022	31/08/2022	17. Cenários	Criação da documentação dos cenários
01/09/2022	07/09/2022	18. Evidências da avaliação	Criação da documentação da avaliação
08/09/2022	15/09/2022	19. Resultados Obtidos	Criação da documentação dos resultados obtidos
16/09/2022	20/09/2022	20. Avaliação Crítica dos resultados	Criação da documentação de avaliação crítica dos resultados do trabalho
21/09/2022	30/09/2022	21. Conclusão	Criação da documentação de conclusão
01/10/2022	07/10/2022	22. Criação do vídeo de apresentação da Etapa 3	Criação do vídeo de apresentação da Etapa 3
08/10/2022	15/10/2022	23. Revisão e envio da terceira etapa do projeto	Revisão e postagem da terceira etapa do trabalho

3. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macroarquitetura da solução.

3.1 Restrições Arquiteturais

Restrições Arquiteturais são requisitos que podem ser Funcionais ou Não Funcionais, mas que impactam diretamente a arquitetura do sistema como um todo. No Quadro 2 é possível ver as restrições arquiteturais do Projeto:

Quadro 2 – Restrições Arquiteturais

Código	Descrição
RA 01	Devem ser utilizadas tecnologias Microsoft para o projeto, Framework .Net a partir da versão 6.0
RA 02	Serviços devem se comunicar preferencialmente através de HTTP ou AMQP, no formato JSON
RA 03	Deve ser utilizado para gestão de identidade a plataforma Microsoft Identity, se integrando ao serviço já existente, gerenciamento todo controle de autenticação, autorização e permissões de acesso de usuários
RA 04	Deve utilizar o provedor Cloud Azure como provedor de infra e demais serviços necessários a aplicação
RA 05	A aplicação deve ser acessível somente na rede interna da empresa ou através de VPN

3.2 Requisitos Funcionais

Os Requisitos Funcionais são aqueles associados as funcionalidades e devem dizer o que a aplicação deve fazer. Abaixo a lista de Requisitos Funcionais identificados para o desenvolvimento inicial do Sistema de Balcão de Atendimento.

Sistema Balcão de Atendimento - Aluga Carros

Quadro 3 – Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A) *	Prioridade (B/M/A) *
RF01	Sistema deve ser integrado ao sistema legado de "Identidade" para controle de usuários e permissões	B	A
RF02	Para acessar o sistema, o usuário deve informar login e senha válidos no serviço de identidade e selecionar a agência desejada. Em caso de usuário e senhas corretos, deverá ser validado se o usuário possui permissão para acessar a agência selecionada. Em caso negativo retornar mensagem "Este usuário não possui permissão para acessar esta agência". Caso o usuário informe um login ou senha incorreta, deve apresentar mensagem "Usuário ou Senha Inválidos"	B	A
RF03	Deve permitir visualizações e movimentações apenas na agência selecionada no login	M	A
RF04	Deve possuir controle de permissão de usuário por menus, oriundas do sistema de Identidade.	A	B
RF05	Deve ser integrado ao sistema legado de "Agências" para listar as agências ativas para seleção no login	B	A
RF06	Deve ser integrado ao sistema legado de "Clientes" para consulta a informações destes	B	B
RF07	Deve listar somente clientes com status "Ativo" para as consultas no sistema	B	B
RF08	Deve possuir uma consulta de Locações Por Cliente, onde informado o CPF deste, deve mostrar as locações em ordem crescente de Data, da mais recente para a mais antiga.	B	M
RF09	Sistema deve ser integrado ao serviço legado de "Frota" para disponibilização de veículos disponíveis para locação	B	A
RF10	Sistema deve listar somente veículos que estão alocados para a agência selecionada no login.	B	A
RF11	Deve possuir rotina para listar todos os veículos da agência, podendo filtrá-los por status	B	M
RF12	Deve possuir rotina para listar próximas manutenções dos veículos, onde informado a placa, deve listar todas as manutenções da data atual até o próximo mês, obtidas do sistema de frota	M	B

RF13	Deve ser integrado ao sistema legado de “Reservas”	B	A
RF14	Deve possuir recurso para listar as reservas em aberto na agência, podendo ser filtradas pela data desejada	B	B
RF15	Deve possuir rotina que permita realizar uma nova locação a partir de uma reserva que deve ser obtida do sistema de “Reservas”, onde a partir da seleção para abrir uma nova locação, deverá listar as informações de Nome do Cliente, CPF do Cliente, Grupo do Veículo, Data da Locação, Data de Devolução, Quantidade de dias (diferença entre as datas), Valor da Reserva e Valor do depósito de segurança. A partir disto, deve permitir selecionar qual veículo desejado pertencente ao grupo da reserva para então confirmar a locação, que após finalizada.	A	A
RF16	Deve após realização de uma locação, enviar para o email do cliente o contrato de locação	B	A
RF17	Deve após realização de uma locação, registrar no serviço legado de “Frota” que o veículo está locado	M	M
RF18	Deve após realização de uma locação, registrar no serviço legado de “Locação” que a mesma ocorreu, informando o código do veículo e código do cliente, data retirada e valor	M	M
RF19	Deve após realização de uma locação, registrar no serviço legado de “Reservas” que a reserva se encontra encerrada	M	M
RF20	Deve permitir realizar o encerramento da locação. Informando o número da locação, data de devolução e confirmar o valor total e Km Atual, receber o valor através de terminal Pinpad, para então encerrá-la.	M	B
RF21	Deve quando encerrar a locação, alterar o status do veículo para “Disponível” no serviço de “Frota”	M	M
RF22	Deve quando encerrar a locação, registrar no serviço de “Locação” o encerramento, informando o código da locação, data de encerramento e valor total	M	M

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

Os Requisitos Não-Funcionais são aqueles associados às restrições de funcionalidades e devem dizer “o como” a aplicação deve fazer. Abaixo a lista de Requisitos Não-Funcionais identificados para o desenvolvimento inicial do Sistema de Balcão de Atendimento.

Quadro 4 – Requisitos Não Funcionais

ID	Descrição	Prioridade B/M/A
RNF 01	O sistema deve ser apresentar disponibilidade 24 X 7 X 365	A
RNF 02	O sistema Balcão de Atendimento, bem como todos as suas APIs que se fizerem necessárias, devem possuir mecanismo de autenticação e autorização integrado ao serviço legado de Identidade, plataforma Microsoft Identity, utilizando JWT com chave assimétrica	A
RNF 03	O histórico de alterações do banco de dados deve ser controlado através das “Migrations” do Entity Framework	A
RNF 04	As compatibilizações de movimentações com serviços já existentes devem ser realizadas através de Azure Functions	M
RNF 05	A aplicação web deve ser responsiva, dispor de mecanismos de acessibilidade e funcionar perfeitamente nas últimas versões de todos os navegadores web modernos	A
RNF 06	A Observabilidade da aplicação e rastreamento de falhas deve ser realizada através da ferramenta Application Insights	A

3.4 Mecanismos Arquiteturais

Esta seção apresenta uma visão geral dos mecanismos que compõem a arquitetura do sistema, baseando-se em três estados: (1) análise, (2) *design* e (3) implementação.

- **Análise:** devem ser listados os aspectos gerais que compõem a arquitetura do sistema.
- **Design:** deve-se identificar o padrão tecnológico a seguir para cada mecanismo identificado na análise.
- **Implementação:** deve-se identificar o produto a ser utilizado na solução.

Quadro 5 – Mecanismos Arquiteturais

Análise	Design	Implementação
Persistência	ORM	Entity Framework Core
Persistência	Micro ORM	Dapper
Persistência	Banco de Dados Relacional	MySQL
Front end	MVC	Asp.Net 6.0 MVC
Back end	API	Asp.Net 6.0 Web API
Back end	Serveless	Azure Functions
Integração	Mensageria	Azure Service Bus
Teste de Software	Testes Unitários	xUnit
Teste de Software	Testes de Integração	xUnit
Autenticação e Autorização	Authorization Token	JWT – Json Web Token
Distribuição	CI/CD - Integração e Entrega Continua	Azure DevOps
Observabilidade	Telemetria	Azure Application Insights

A Figura 1 mostra a especificação do diagrama de contexto geral da solução proposta, com todos seus principais módulos e suas integrações com os demais sistemas da empresa.

É importante observar que o fluxo se inicia com o cliente realizando a reserva no *site* de reservas da empresa. Esta ação resultará na criação de uma reserva que aparecerá no Balcão de Atendimento para realização de uma locação.

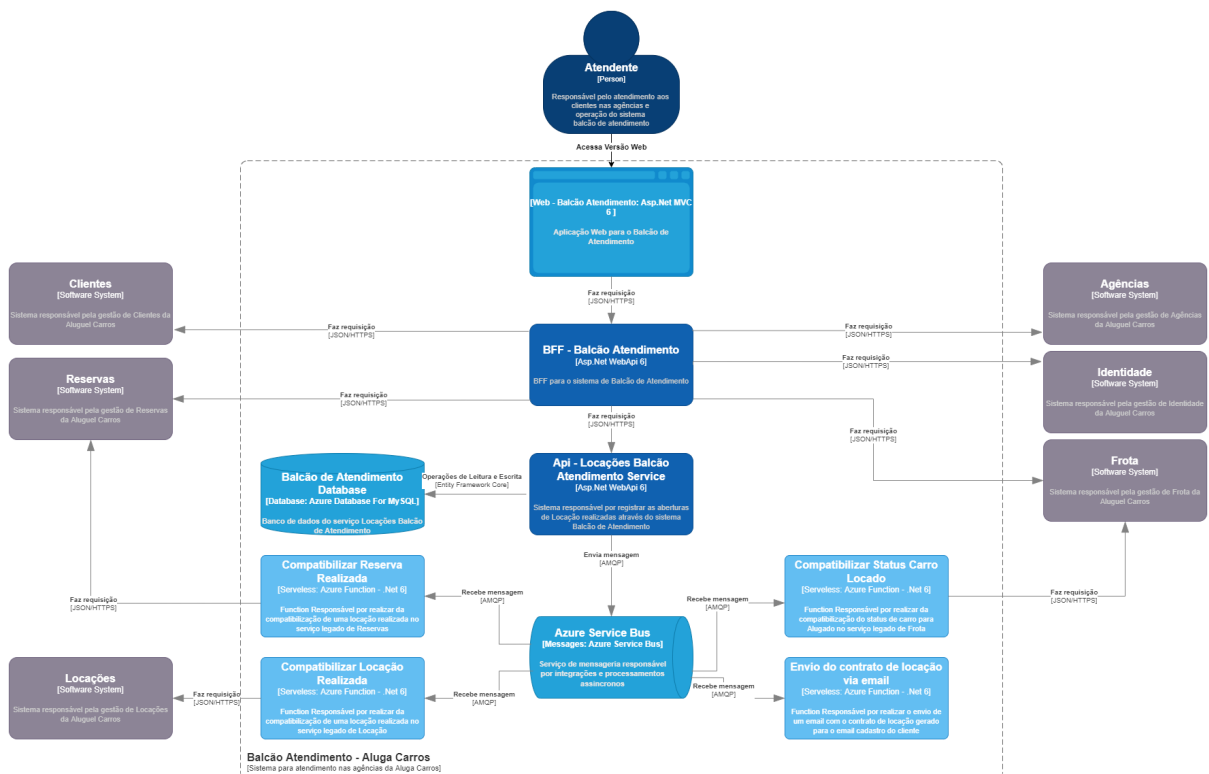
Também há integração com o sistema de Frota, onde serão obtidas informações de veículos bem como *status* e situação atual para locação e com o sistema de clientes para obtenção de informações necessárias destes. Ainda possui integração com o sistema de Identidade para gestão de acesso, e com o serviço de Locações para compatibilização das locações efetuadas no Balcão de Atendimento a partir das reservas.

Link Vídeo de Apresentação Etapa 1: <https://youtu.be/wWJmtfOHbhc>

4.2 Diagrama de Container

Na figura 2 está demonstrado o Diagrama de Container no modelo C4 da aplicação.

Figura 2 – Diagrama de Container



O Atendente é o usuário do sistema, que atua nas agências operando-o através do navegador, acessando o projeto Web Balcão de Atendimento que é desenvolvido em Asp.Net MVC 6.

O projeto Web se comunica com o projeto BFF através de chamadas HTTP, e este é desenvolvido em Asp.Net WebApi 6. O BFF é responsável por realizar a comunicação com os sistemas externos de Clientes, Reservas, Agências, Identidade e Frota, para atender as necessidades da aplicação Web. Ele também se comunica com o serviço de Locações do Balcão de Atendimento. Todas as comunicações do BFF são realizadas através de HTTP.

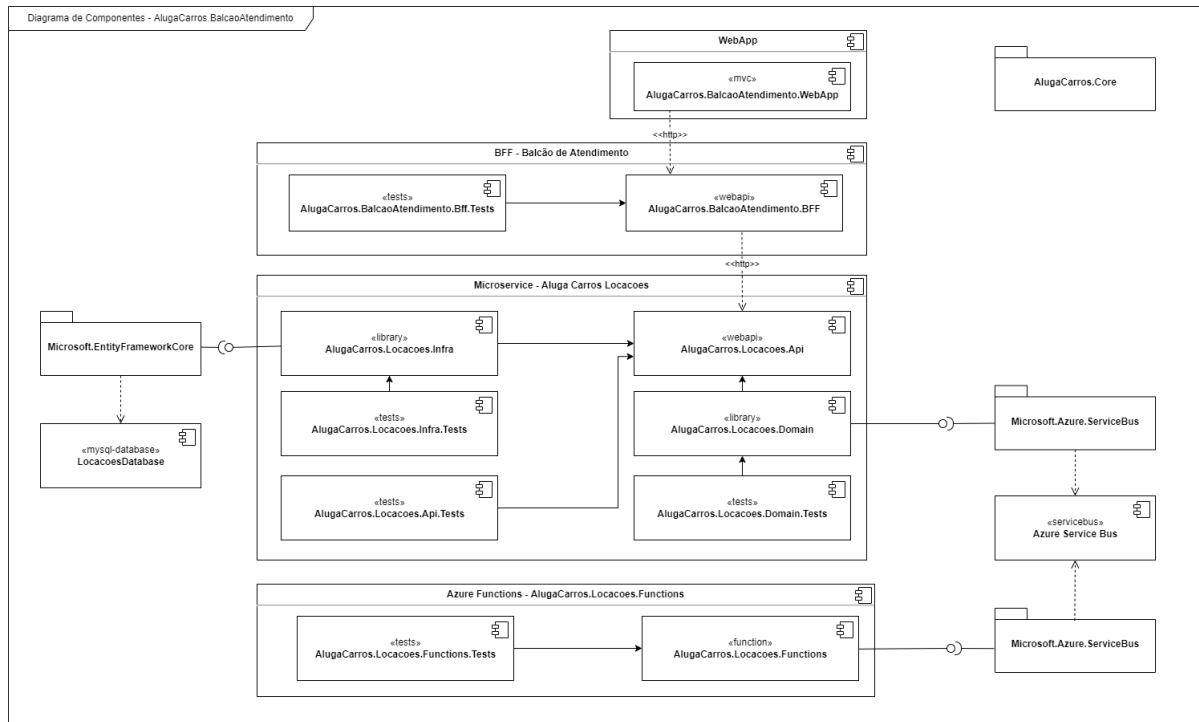
O serviço de Locações também foi desenvolvido em Asp.Net WebApi 6. Possui comunicação com o banco de dados MySql, para registrar informações específicas e necessárias de locações do balcão de atendimento, e com o Azure Service Bus através de AMQP para streaming de eventos relacionados a Locação.

A aplicação também possui 4 Azure Functions, desenvolvidas em .Net 6, que são acionadas através de Bus Triggers, com o evento de criação de reserva, para compatibilizar as informações nos serviços externos.

4.3 Diagrama de Componentes

Na figura 3 está demonstrado o Diagrama de Componentes da aplicação que tem por objetivo detalhar os componentes utilizados para construção do sistema Balcão de Atendimento.

Figura 3 – Diagrama de Componentes



A lista a seguir detalha cada um dos componentes da aplicação.

- **AlugaCarros.BalcaoAtendimento.WebApp:** Aplicação *Web*, desenvolvida em Asp.Net MVC. Onde o usuário acessará o sistema através de seu navegador, podendo este ser *desktop* ou *mobile*. Suas comunicações com demais serviços são realizadas através de chamadas http, no formato JSON.
- **BFF Balcão Atendimento:** É responsável por recuperar e organizar todas as informações necessárias à aplicação web, sendo um ponto focal de suas requisições.
 - **AlugaCarros.BalcaoAtendimento.Bff:** Desenvolvido em Asp.Net WebApi. Serviço é um *Backend for Frontend*, responsável por receber as requisições da aplicação web. Todas as comunicações do serviço são realizadas através de chamadas http, no formato JSON.
 - **AlugaCarros.BalcaoAtendimento.Bff.Tests:** Projeto de testes unitários para a aplicação AlugaCarros.BalcaoAtendimento.Bff.

- Aluga Carros Locações: Serviço central da aplicação. É responsável por registrar as locações realizadas no Balcão de Atendimento, bem como envio de eventos via mensagens ao Service Bus, e gravações de movimentações no banco de dados.
 - o AlugaCarros.Locacoes.Api: Desenvolvida em Asp.Net WebApi. Api responsável por expor os recursos do serviço Locação.
 - o AlugaCarros.Locacoes.Api.Tests: Projeto de testes unitários para a aplicação AlugaCarros.Locacoes.Api.
 - o AlugaCarros.Locacoes.Domain: Biblioteca de classes responsável pelas entidades, serviços e contratos (*interfaces*) do serviço de Locações, e também pelo envio de eventos para o Azure Service Bus.
 - o AlugaCarros.Locacoes.Domain.Tests: Projeto de testes unitários para a biblioteca de classes AlugaCarros.Locacoes.Domain.
 - o Microsoft.Azure.ServiceBus: Pacote responsável por abstrair a comunicação entre a aplicação de Locações e o Azure Service Bus.
 - o Azure Service Bus: Serviço de mensagens do Azure. Irá receber os eventos do serviço de Locações e direcioná-los as devidas filas para posterior consumo das mensagens.
 - o AlugaCarros.Locacoes.Infra: Biblioteca de classes responsável pela infraestrutura e acessos externos da aplicação de Locações. Possui integração com a base de dados MySql, através do pacote Microsoft.EntityFrameworkCore.
 - o AlugaCarros.Locacoes.Infra.Tests: Projeto de testes unitários para a biblioteca de classes AlugaCarros.Locacoes.Infra.
 - o Microsoft.EntityFrameworkCore: Pacote ORM da Microsoft que proporciona muitas facilidades de consultas e manipulações de dados, além do desenvolvimento *Code First* para o serviço AlugaCarros.Locacoes.Infra.
 - o LocacoesDatabase: Banco de dados MySql responsável por armazenar as informações do serviço de Locações.

- AlugaCarros.Locacoes.Functions: Projeto de Azure Functions, desenvolvidas em .Net 6, responsáveis por processamentos assíncronos da aplicação.
- AlugaCarros.Core: Responsável por centralizar artefatos comuns as aplicações, como configurações de autenticação e autorização via JWT, configurações de log, *Middlewares* para manipulação de erros e *logging*, dentre outras configurações.

5. Prova de Conceito (PoC)

5.1 Integrações entre Componentes

Para demonstrar o protótipo do sistema, os 3 requisitos funcionais prioritários selecionados foram:

- RF11 - Deve possuir rotina para listar todos os veículos da agência, podendo filtrá-los por status
- RF14 - Deve possuir recurso para listar as reservas em aberto na agência, podendo ser filtradas pela data desejada
- RF15 - Deve possuir rotina que permita realizar uma nova locação a partir de uma reserva que deve ser obtida do sistema de “Reservas”, onde a partir da seleção para abrir uma nova locação, deverá listar as informações de Nome do Cliente, CPF do Cliente, Grupo do Veículo, Data da Locação, Data de Devolução, Quantidade de dias (diferença entre as datas), Valor da Reserva e Valor do depósito de segurança. A partir disto, deve permitir selecionar qual veículo desejado pertencente ao grupo da reserva para então confirmar a locação, que após finalizada deverá gerar um evento de sua criação.

O protótipo de telas sistema e o protótipo navegável para apresentar os três requisitos selecionados que foram desenvolvidos podem ser encontrados respectivamente em:

- <https://www.figma.com/file/DEusRFurdHExw95LI2mqAU/Balc%C3%A3o-Atendimento---Aluga-Carros?node-id=0%3A1>

- <https://www.figma.com/proto/DEusRFurdHExw95LI2mqaU/Balc%C3%A3o-Atendimento---Aluga-Carros?page-id=0%3A1&node-id=3%3A2&viewport=1517%2C318%2C0.1&scaling=min-zoom&starting-point-node-id=3%3A2>

O fluxo do sistema se inicia na tela de login. Onde fornecido um login e senha válidos e selecionada a agência desejada, redirecionará o usuário para a tela *Home*.

O Sistema dispõe de um menu principal superior, ao qual possui os seguintes itens:

- *Home*: Página principal do sistema
- Reservas: Possui dois submenus para visualizar as Reservas: Por Data, Em Aberto.
 - Por Data: Irá demonstrar as reservas realizadas na data filtrada. Pode-se também iniciar a locação a partir da reserva por esta tela no botão “Locar”.
 - Em Aberto: Requisito Funcional 14, tela responsável por listas as próximas reservas ainda não realizadas para a agência. Pode-se filtrar a data desejada, bem como outros dados da tabela no campo “Procurar”. Pode-se também iniciar a locação a partir da reserva por esta tela no botão “Locar”.
- Locações: Possui três submenus: Realizadas, Nova e Fechar
 - Realizadas: Tela onde o usuário poderá visualizar as reservas realizadas na agência, podendo filtrá-las pelos dados da tabela desejados no campo “Procurar”.
 - Nova: Irá redirecionar o usuário para uma tela onde poderá informar o código da reserva, para então iniciar a locação na tela “Abrir Locação”
 - Abrir Locação: Requisito Funcional 15, tela onde é apresentado as informações da reserva e cliente, para que o atendente possa realizar a locação. O campo Veículo mostra apenas os veículos que estão com status “Disponível” e alocados na agência. Após selecionar o veículo desejado, clicando no botão “Abrir Locação” a locação será aberta e o usuário redirecionado para a tela de “Locações Realizadas”.

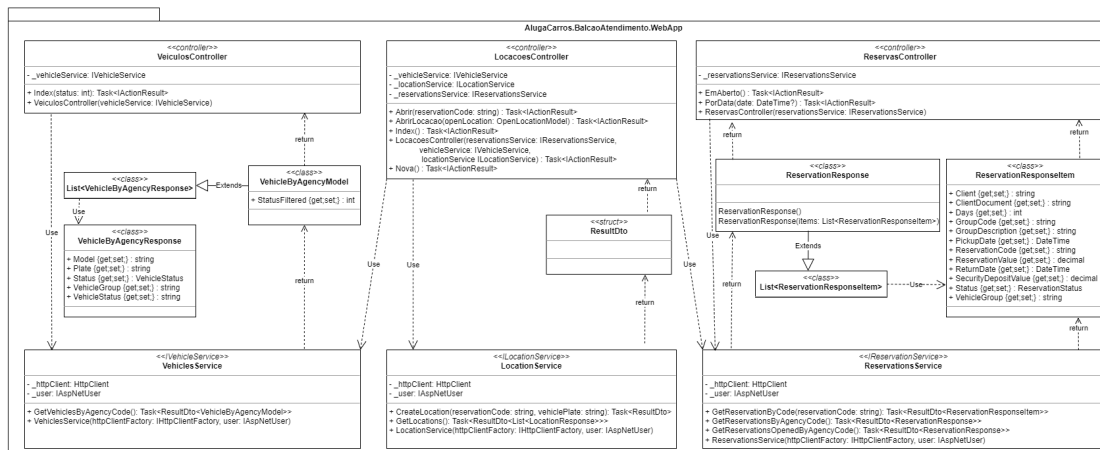
- o Fechar: Recurso não implementado por não se tratar de um dos três requisitos prioritários selecionados

- Veículos: Requisito Funcional 11, tela responsável por mostrar os veículos alocados na agência e seus status, podendo filtrá-los por status.

O vídeo apresentando o protótipo navegável pode ser acessado através do link: <https://youtu.be/mlZ9NJr-NrQ>

5.2 Código da Aplicação

Figura 4 – Diagrama de Código - AlugaCarros.BalcaoAtendimento.WebApp

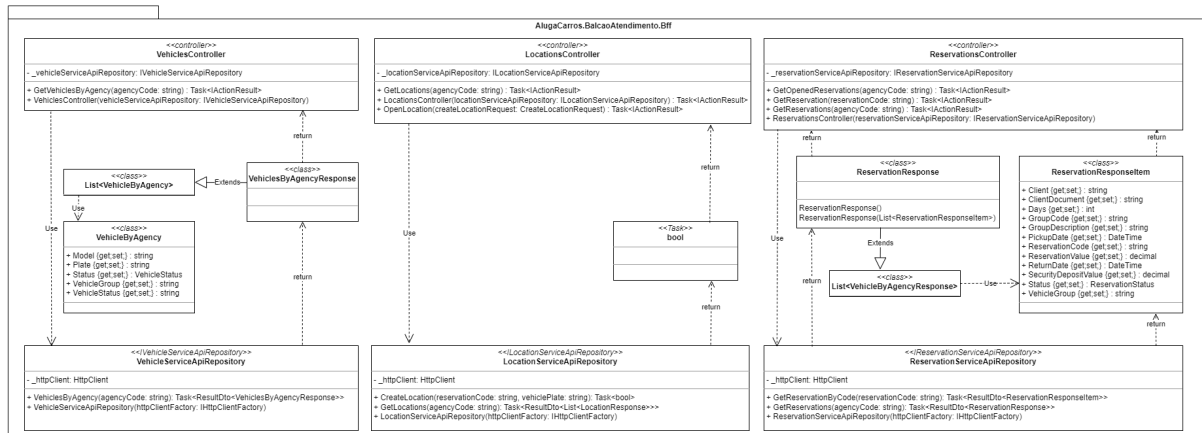


A estrutura da aplicação mostrada na Figura 4 apresenta os componentes de código e suas funções no componente AlugaCarros.BalcaoAtendimento.WebApp. Abaixo uma explanação dos itens agrupados por requisito funcional prioritário:

- RF 11:
 - o VeiculosController: Responsável por controlar as Views de veículos e suas ações.
 - o VehiclesService: Serviço responsável por controlar as regras de negócio, consultas e demais necessidades do contexto de veículo. Realiza chamada http a aplicação BFF Balcão Atendimento para obter os veículos alocados para a agência selecionada.
 - o VehicleByAgencyModel: Modelo para a View de consulta de Veículos por Agência. Classe herda de List<VehicleByAgencyResponse>.

- o VehicleByAgencyResponse: Item da coleção VehicleByAgencyModel.
- RF 14
 - o ReservasController: Responsável por controlar as *Views* de reservas e suas ações.
 - o ReservationsService: Serviço responsável por controlar as regras de negócio, consultas e demais necessidades do contexto de reservas. Realiza chamada http a aplicação BFF Balcão Atendimento para obter as reservas em aberto para a agência selecionada.
 - o ReservationResponse: Modelo para a *View* de Reservas em Aberto obtido de requisição ao BFF. Classe herda de List<ReservationResponseItem>.
 - o ReservationResponseItem: Item da coleção de ReservationResponse.
- RF 15
 - o LocacoesController: Responsável por controlar as *Views* de locações e suas ações.
 - o LocationService: Serviço responsável por controlar as regras de negócio, consultas e demais necessidades do contexto de locações. Realiza chamada http a aplicação BFF Balcão Atendimento para a criação de uma locação. O método “*CreateLocation*” retorna apenas uma confirmação se a criação foi realizada com sucesso ou não, através de um ResultDto.

Figura 5 – Diagrama de Código - AlugaCarros.BalcaoAtendimento.BFF



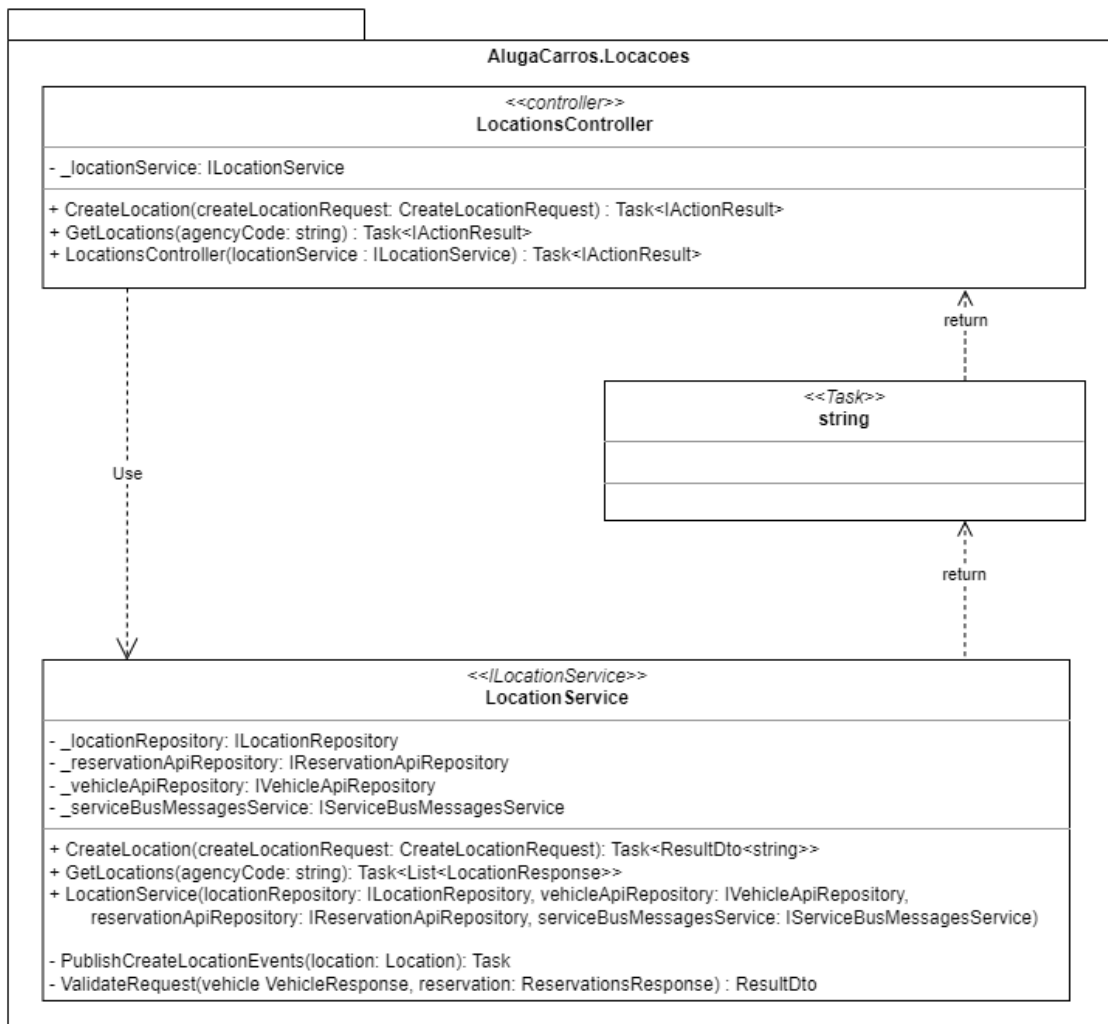
A estrutura da aplicação mostrada na Figura 5 apresenta os componentes de código e suas funções no componente AlugaCarros.BalcaoAtendimento.BFF. Abaixo uma explanação dos itens agrupados por requisito funcional prioritário:

- RF 11
 - o VehiclesController: Responsável por gerenciar os recursos de Veículo no componente.
 - o VehicleServiceApiRepository: Responsável por realizar a comunicação através de chamadas http com o serviço externo de Frota.
 - o VehiclesByAgencyResponse: Resultado da consulta de veículos ao serviço externo de Frota. Classe herda de List<VehicleByAgency>.
 - o VehicleByAgency: Objeto de retorno da consulta de veículos ao serviço externo de Frota. Item da coleção VehiclesByAgencyResponse.
- RF 14
 - o ReservationsController: Responsável por gerenciar os recursos de Reserva no componente.
 - o ReservationServiceApiRepository: Responsável por realizar a comunicação através de chamadas http com o serviço externo de Reservas.
 - o ReservationResponse: Resultado da consulta de reservas ao serviço externo de Reservas. Classe herda de List<ReservationResponseItem>.

Sistema Balcão de Atendimento - Aluga Carros

- o ReservationResponseItem: Objeto de retorno da consulta de reservas ao serviço externo de Reservas. Item da coleção ReservationResponse.
- RF 15
 - o LocationsController: Responsável por gerenciar os recursos de Locação no componente
 - o LocationServiceApiRepository: Responsável por realizar a comunicação através de chamadas http com o serviço de Locação.

Figura 6 – Diagrama de Código - AlugaCarros.Locacoes



A estrutura da aplicação mostrada na Figura 6 apresenta os componentes de código e suas funções no componente *AlugaCarros.Locacoes*. Este projeto é responsável pelo gerenciamento do contexto de Locações da aplicação. O requisito funcional prioritário representado na imagem refere-se ao RF 15, os demais requisitos não estão relacionados a este projeto.

A seguir uma explanação dos itens do requisito funcional prioritário RF 15:

- *LocationsController*: Responsável por gerenciar os recursos de Locação no componente.
- *LocationService*: Responsável executar as regras de negócio do contexto de Locações. O método *CreateLocation* é o responsável pelo RF 15. Este método também realiza a validação do *request* recebido na *LocationsController* e envia via mensagem ao Azure ServiceBus os eventos de criação de locação para que as *Functions* possam ser acionadas e realizar as devidas compatibilidades.

O repositório com o código das aplicações pode ser encontrado em: <https://github.com/alugacarros>.

O sistema web encontra-se publicado na nuvem Azure e pode ser acessado pelo link: <https://alugacarros-balcaoatendimento.azurewebsites.net>.

Para acessá-lo para testes, utilizar o *email* “**usuario@teste.com**” e senha “**TestePUC@123**”, selecionar a “**Agência**” desejada e clicar em “**Entrar**” para ser redirecionado a *Home* do sistema. Observação: o sistema pode apresentar lentidão devido a estar hospedado utilizando a camada *free* do Azure AppService.

Existem ainda outros 2 serviços que são WebApi's e estão publicados também no Azure, sendo eles: *AlugaCarros.BalcaoAtendimento.Bff* e *AlugaCarros.Locacoes*, que podem ser acessados respectivamente em:

- <https://alugacarros-balcaoatendimento-bff.azurewebsites.net/swagger>
- <https://alugacarros-locacoes.azurewebsites.net/swagger>

Para utilizá-los é necessário realizar a geração do token JWT, no *endpoint* `/api/v1/users/login` do serviço `AlugaCarros.BalcaoAtendimento.Bff`. No retorno da requisição, copiar o conteúdo do campo `accessToken` e clicar no botão `Authorize` no canto superior direito da Api, e inseri-lo no formato `Bearer accessToken`.

Os diagramas do projeto podem ser encontrados em:
<https://github.com/alugacarros/alugacarros-documentacao>.

6. Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

6.1. Análise das abordagens arquiteturais

Quadro 5 – Análise das abordagens arquiteturais

Atributos de Qualidade	Cenários	Importância	Complexidade
Disponibilidade	Cenário 1: O sistema deve apresentar disponibilidade 24 X 7 X 365	A	B
Segurança	Cenário 2: O sistema Balcão de Atendimento, bem como todos as suas APIs que se fizerem necessárias, devem possuir mecanismo de autenticação e autorização integrado ao serviço legado de Identidade, plataforma Microsoft Identity, utilizando JWT com chave assimétrica	A	M
Manutenibilidade	Cenário 3: O histórico de alterações do banco de dados deve ser controlado através das “Migrations” do Entity Framework.	M	B
Eficiência	Cenário 4: As compatibilizações de movimentações com serviços já existentes devem ser realizadas através de Azure Functions	B	M
Usabilidade	Cenário 5: A aplicação web deve ser responsiva, dispor de mecanismos de acessibilidade e funcionar perfeitamente nas últimas versões de todos os navegadores web modernos	A	M
Rastreabilidade	Cenário 6: A Observabilidade da aplicação e rastreamento de falhas deve ser realizada através da ferramenta Application Insights	A	B

6.2. Cenários

Cenário 1 – Disponibilidade: Os serviços do Azure garantem que nossas aplicações fiquem disponíveis no mínimo 99% do tempo.

Cenário 2 - Segurança: Utilização de autenticação com *tokens* JWT e chave assimétrica garante maior segurança a nossas aplicações. A aplicação de Identidade possui duas chaves, pública e privada, e as demais aplicações só precisam conhecer a chave pública para abrir o *token* JWT e validá-lo, bem como obter informações do usuário a partir deste.

Cenário 3 – Manutenibilidade: A utilização das *Migrations* do Entity Framework, garantem que sempre que seja necessária alguma atualização de banco de dados, a mesma será executada antes que a nova versão da aplicação seja executada pela primeira vez.

Cenário 4 – Eficiência: A utilização de Azure *Functions* garante que as aplicações não estejam acopladas e que o processamento seja efetuado de maneira assíncrona, garantindo maior experiência ao usuário e tornando o sistema mais eficiente durante o uso.

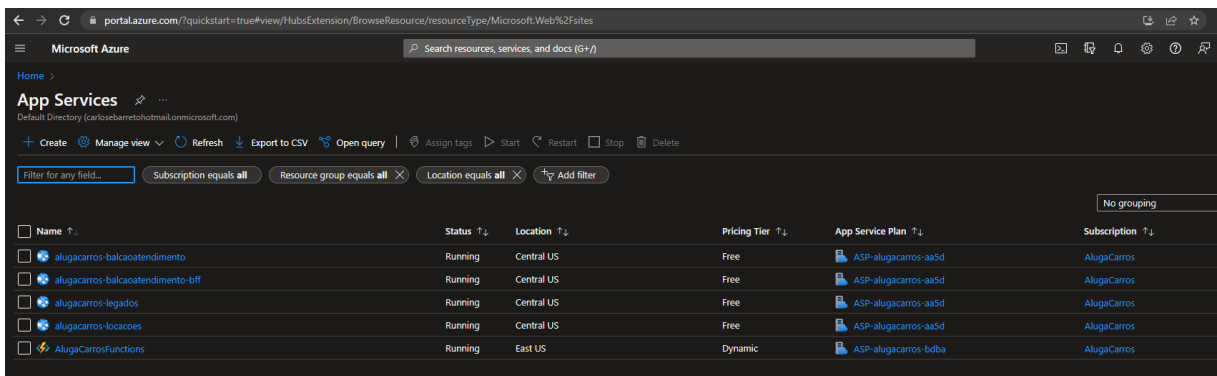
Cenário 5 – Usabilidade: A aplicação web deve ser de fácil navegação e funcionar perfeitamente em todos os principais navegadores web em suas versões mais recentes.

Cenário 6 – Rastreabilidade: O serviço do Azure *Application Insights* garante que a aplicação possua uma ótima qualidade em seu monitoramento e observabilidade, permitindo que possamos rastrear de uma maneira muito simples, quaisquer problemas que venham a acontecer ou indícios destes.

6.3. Evidências da Avaliação

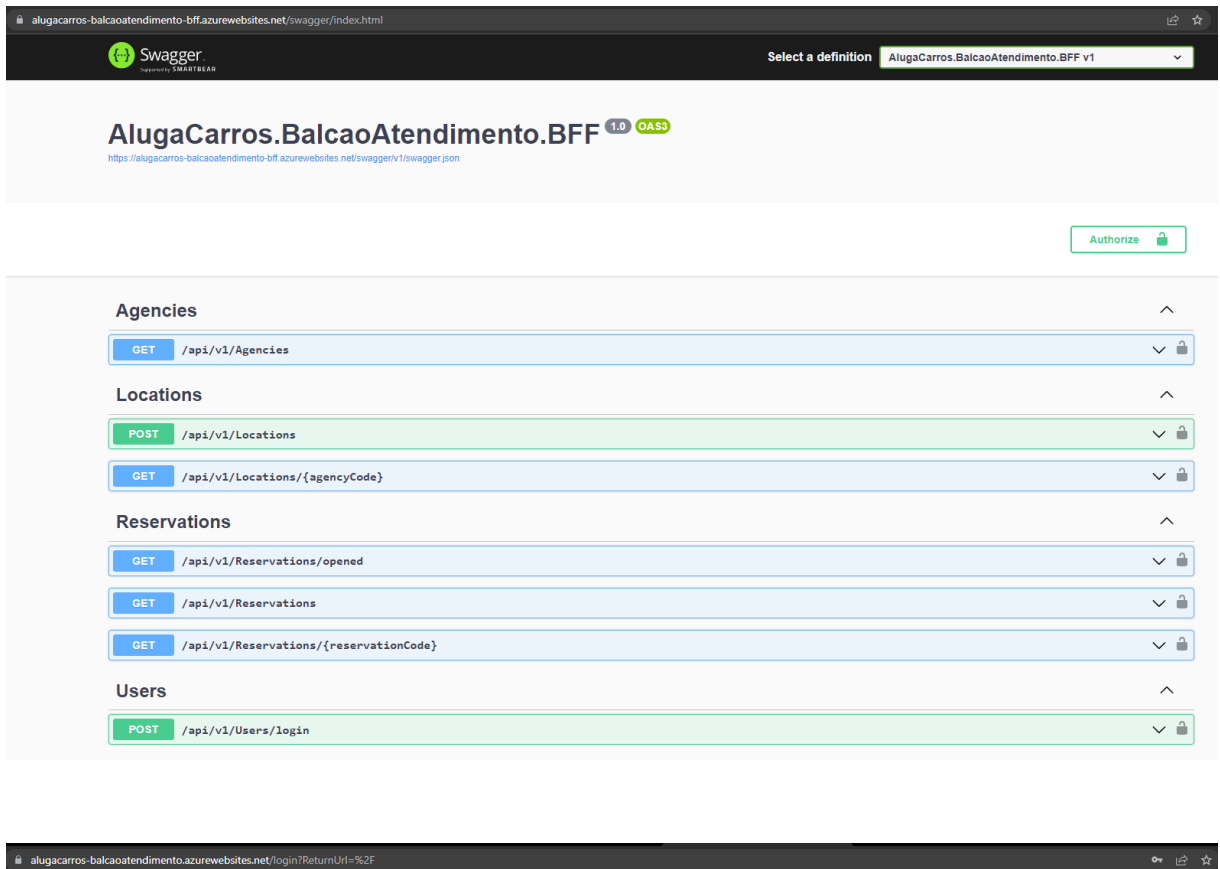
Apresente as medidas registradas na coleta de dados. Para o que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento ao requisito não-funcional.

Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema deve apresentar disponibilidade 24 X 7 X 365
Preocupação:	
O sistema deve apresentar a maior disponibilidade possível, visto que existem agências que operam no formato 24 x 7.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Usuários que utilizam o sistema	
Mecanismo:	
Acessar o sistema em qualquer dia e horário.	
Medida de resposta:	
Implantação do sistema na <i>cloud</i> Azure, em serviços com no mínimo 99,9% de disponibilidade	
Considerações sobre a arquitetura:	
Riscos:	Sistema pode ficar indisponível, causando problemas para a área de operação em sua utilização.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há



Name	Status	Location	Pricing Tier	App Service Plan	Subscription
alugacarros-balcaoatendimento	Running	Central US	Free	ASP-alugacarros-aa5d	AlugaCarros
alugacarros-balcaoatendimento-bff	Running	Central US	Free	ASP-alugacarros-aa5d	AlugaCarros
alugacarros-legados	Running	Central US	Free	ASP-alugacarros-aa5d	AlugaCarros
alugacarros-locacoes	Running	Central US	Free	ASP-alugacarros-aa5d	AlugaCarros
AlugaCarrosFunctions	Running	East US	Dynamic	ASP-alugacarros-bdba	AlugaCarros

Sistema Balcão de Atendimento - Aluga Carros



The image shows the Swagger UI for the 'AlugaCarros.BalcaoAtendimento.BFF v1' API. The interface is organized into sections: Agencies, Locations, Reservations, and Users. Each section lists the available endpoints with their HTTP methods and paths. The 'Agencies' section has one GET endpoint. The 'Locations' section has two endpoints: a POST endpoint and a GET endpoint with a path parameter. The 'Reservations' section has three GET endpoints. The 'Users' section has one POST endpoint. An 'Authorize' button is visible in the top right corner.

alugacarros-balcaoatendimento-bff.azurewebsites.net/swagger/index.html

Swagger
powered by SMARTBEAR

Select a definition AlugaCarros.BalcaoAtendimento.BFF v1

AlugaCarros.BalcaoAtendimento.BFF 1.0 OAS3
<https://alugacarros-balcaoatendimento-bff.azurewebsites.net/swagger/v1/swagger.json>

Authorize

Agencies

GET /api/v1/Agencies

Locations

POST /api/v1/Locations

GET /api/v1/Locations/{agencyCode}

Reservations

GET /api/v1/Reservations/opened

GET /api/v1/Reservations

GET /api/v1/Reservations/{reservationCode}

Users

POST /api/v1/Users/login

alugacarros-balcaoatendimento.azurewebsites.net/login?ReturnUrl=%2F

Sistema Balcão Atendimento

Email

Senha

Agência

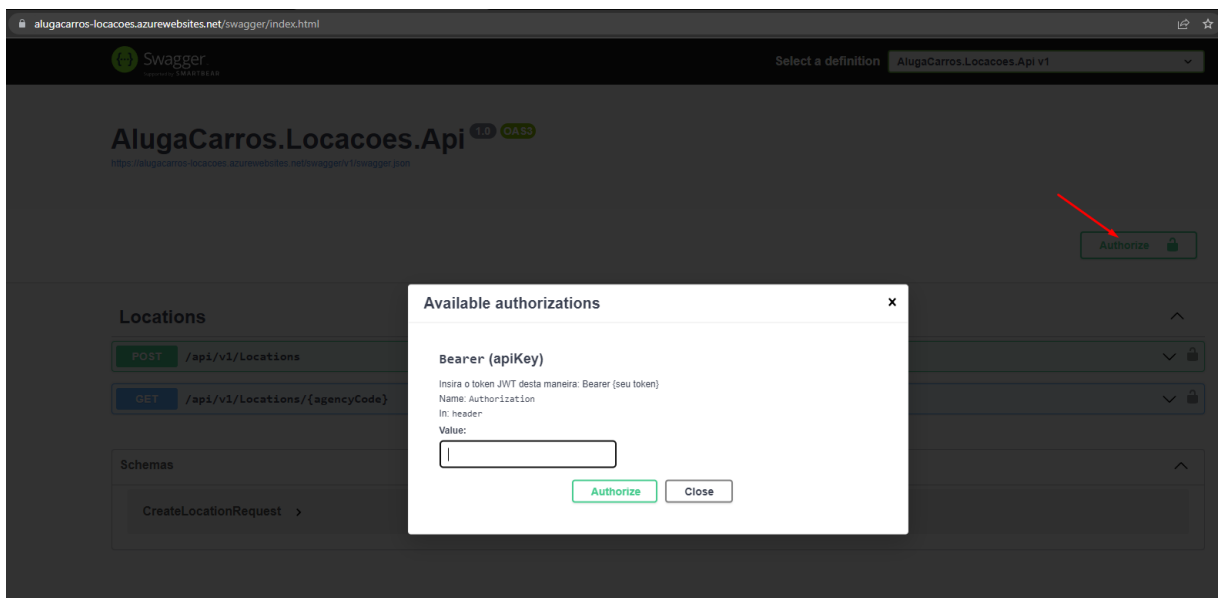
Entrar

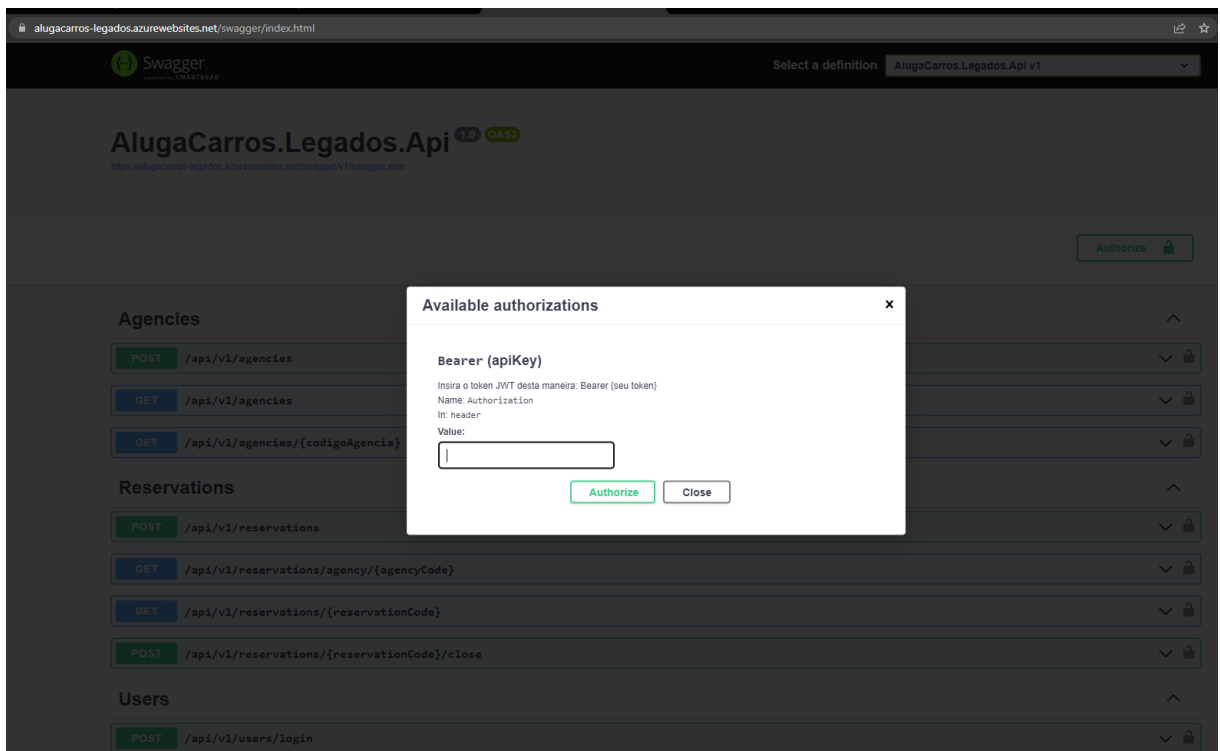
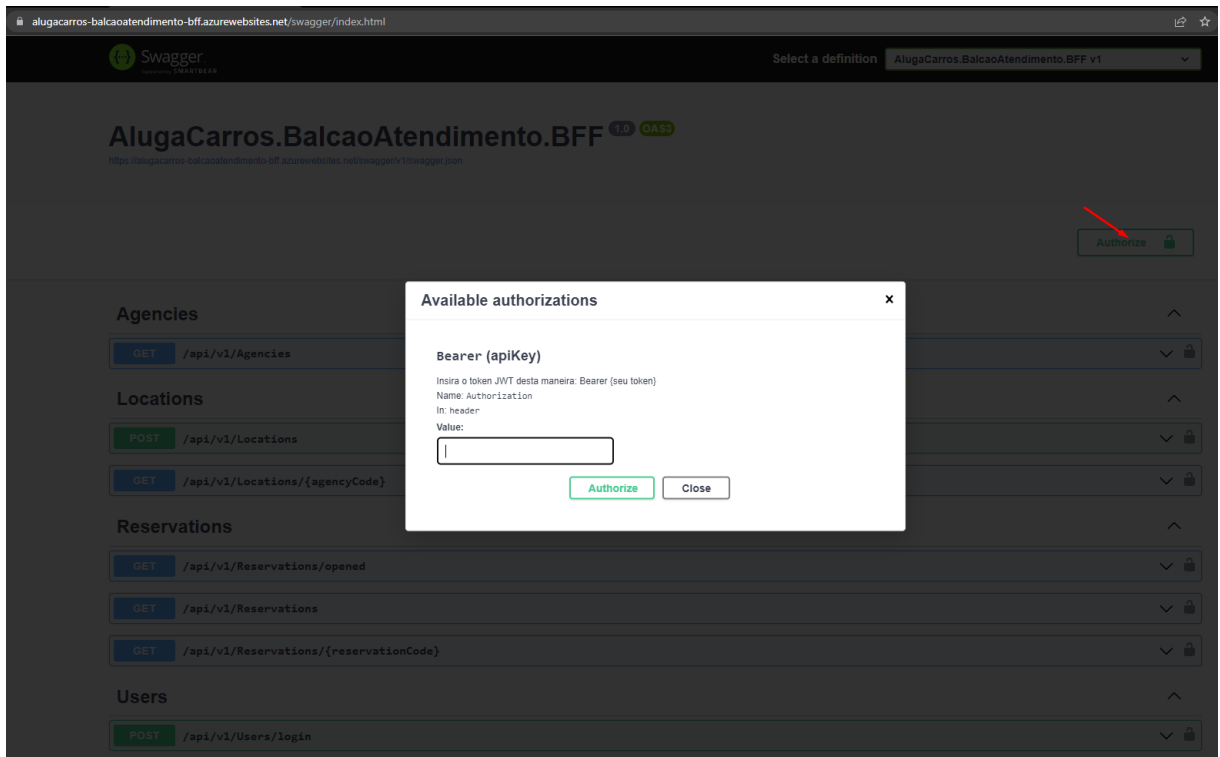
Como é possível observar nas imagens, as aplicações estão hospedadas no *Azure App Service*, e este garante alto nível de disponibilidade para SLA, de 99,95%.

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O sistema Balcão de Atendimento e todas as suas APIs devem possuir mecanismo de autenticação e autorização utilizando JWT com chave assimétrica
Preocupação:	
O sistema precisa de um modo de autenticação para garantir que somente usuários autorizados possam efetuar operações.	
A chave assimétrica se faz necessária para garantir maior segurança das aplicações, dado que a chave privada não será conhecida pelas aplicações que utilizavam o <i>token</i> .	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Segurança das informações	
Mecanismo:	

Sistema Balcão de Atendimento - Aluga Carros

Autenticação e autorização via <i>JWT Token</i>	
Medida de resposta:	
Implementação de mecanismos de autenticação e autorização em todas as aplicações via <i>JWT Token</i>	
Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	A utilização de chave assimétrica garante maior segurança no processo de autenticação, porém a performance do processo é degradada, se comparado a utilização de chaves simétricas.





Sistema Balcão de Atendimento - Aluga Carros

```

1 // 20220914073939
2 // https://alugacarros-legados.azurewebsites.net/jwks
3
4 {
5   "keys": [
6     {
7       "kty": "RSA",
8       "use": "sig",
9       "kid": "hRAUis1w-Eyrjaqy4FXGA",
10      "n": "n99tqqnmfC7Pdd-Zj1ZFlvtQe2CvrM_r-aMDvV4vH8eIQIC6lVRgg1UXqfS2JH5GHP2eKLTBSHFEDGyLLEKBAuT7VxqGwT8x7npoZHiDPp_04UMZV-5DG41vF0Tfrn8p2Fmzec1nRhuxX2Hly6-7Y_kANTUvbkZK3qT1uWqtca68e3smk1AoEDL2yTX5aqDMK8bv8sN8bYL1j1dek-1baZnVJHpOyha1LmsI8vm8BvSNPhLLSdzZeRozyLD08ZAdt8IMTvmLAYoY0EtB4s6Bk3LASBksaLEDO48CPE3eM5a8vZ_hxIEI_exKc9thV8xS-yM0k3Cn3IPL2D79ytcV7bosTJ53yKVCz8qPWRa8fj9BihYDVjX8jZ82ubxQ6-qHicLWShdsGG59kODE6Q-nXokm9GyNRgHeBK1iQy4g5z8UhtuKmrYM_0xUOHZEQoTyfshXLMohRKcz7JospQPDpGYd6TII2HQGCrB0PSV1530Jh-516wm167105_",
11      "e": "AQAB"
12    },
13    {
14      "kty": "RSA",
15      "use": "sig",
16      "kid": "hHGKUN9an8XyZsos7BIKXg",
17      "n": "zQ1s5n6ux8j3hb7eMedQqRjRD0r0oU5YR8c397oFH0A2X2zfzWdNyzn-L91gA4KIBS1vfaBubeQqSaB0TzC1vXPwv1NAU1wPck-5udy19G03ukrHT11aA4XxcgdnMUC9wsHapIxVF08FebJHISWJ0gbZVB2xNNE8ma77IS4Y-LA7k3pDuMCHZ8AjeRaTKVdr7ShVP6ja3ULuFgOesKD4bqFU11iekZ8AtHx0P0s40AfTHdoQqG5g5kl1vILUK6b_s1r6fN17Pxdkoy1ULnahpY3ff4py3pf-EboEza8H-D_XEzpkdyd_dGexEDzo1YEIn-i4jh_D5Yf6dVeYR2rI307hP-mmKqAhxm7Lm0f70AqahBf5mKjRaaU7hjV6xRTmb0ZUXvZq07n593PnktcrwyXqxt97-s0X0qz8k0YVR1qoUMAw5c6sQvRAG6LMZuH81j1UbYgIZqFesUj0knjuvGeVLqsKnsEwjq_B7LrtvHtd7qGfI7Wcd5gECK9",
18      "e": "AQAB"
19    }
20   ]
21 }

```

Na imagem acima é demonstrado o *endpoint* onde a chave pública para validação do *token* JWT é exposta para as demais aplicações utilizarem. Nas demais imagens é demonstrado que todas as API's utilizam processo de autorização via JWT *Bearer Token*.

Atributo de Qualidade:	Manutenabilidade
Requisito de Qualidade:	O histórico de alterações do banco de dados deve ser controlado através das <i>Migrations</i> do Entity Framework.
Preocupação:	
Controlar se o banco de dados está equivalente com a versão atual da aplicação para o correto funcionamento pode se tornar um processo de difícil acompanhamento. As <i>Migrations</i> do Entity Framework permitem que o framework gere todas as atualizações necessárias para o banco de dados sempre estar com a versão atualizada referente ao código atual da aplicação, <i>Code First</i> . O <i>rollback</i> da versão também é facilitado caso este seja necessário	
Cenário(s):	
Cenário 3	
Ambiente:	
Bancos de dados da aplicação	
Estímulo:	
Integridade dos esquemas de dados entre aplicação e base de dados.	

Mecanismo:	
ORM Entity Framework	
Medida de resposta:	
Criação de nova versão de <i>Migration</i> a cada alteração no código fonte que seja de alguma entidade da aplicação	
Considerações sobre a arquitetura:	
Riscos:	Não atualizar o modelo de migração quando realizada uma alteração nas entidades da aplicação ocorrerá erro por não sincronia de esquema de dados.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

```

4 #nullable disable
5
6 namespace AlugaCarros.Locacoes.Infra.Migrations
7 {
8     1 reference
9     public partial class Initial : Migration
10    {
11        0 references
12        protected override void Up(MigrationBuilder migrationBuilder)
13        {
14            migrationBuilder.AlterDatabase()
15                .Annotation("MySQL:Charset", "utf8mb4");
16
17            migrationBuilder.CreateTable(
18                name: "Locations",
19                columns: column => new
20                {
21                    Id = table.Column<Guid>(type: "char(36)", nullable: false, collation: "ascii_general_ci"),
22                    Code = table.Column<string>(type: "longtext", nullable: false)
23                        .Annotation("MySQL:Charset", "utf8mb4"),
24                    ReservationCode = table.Column<string>(type: "longtext", nullable: false)
25                        .Annotation("MySQL:Charset", "utf8mb4"),
26                    AgencyCode = table.Column<string>(type: "longtext", nullable: false)
27                        .Annotation("MySQL:Charset", "utf8mb4"),
28                    ClientDocument = table.Column<string>(type: "longtext", nullable: false)
29                        .Annotation("MySQL:Charset", "utf8mb4"),
30                    VehicleGroupCode = table.Column<string>(type: "longtext", nullable: false)
31                        .Annotation("MySQL:Charset", "utf8mb4"),
32                    VehiclePlate = table.Column<string>(type: "longtext", nullable: false)
33                        .Annotation("MySQL:Charset", "utf8mb4"),
34                    Date = table.Column<DateTime>(type: "datetime(6)", nullable: false),
35                    ReturnDate = table.Column<DateTime>(type: "datetime(6)", nullable: false),
36                    Value = table.Column<decimal>(type: "decimal(65,30)", nullable: false),
37                    SecurityDepositValue = table.Column<decimal>(type: "decimal(65,30)", nullable: false)
38                },
39                constraints: <#>
40                {
41                    table.PrimaryKey("PK_Locations", new { Id }, Microsoft.EntityFrameworkCore.Migrations.Operations.AddColumnOperation)
42                })
43                .Annotation("MySQL:Charset", "utf8mb4");
44
45        0 references
46        protected override void Down(MigrationBuilder migrationBuilder)
47        {
48            migrationBuilder.DropTable(
49                name: "Locations");
50        }
51    }

```

SQL Server Enterprise Manager - AlugaCarros-database.mysql.database.azure.com

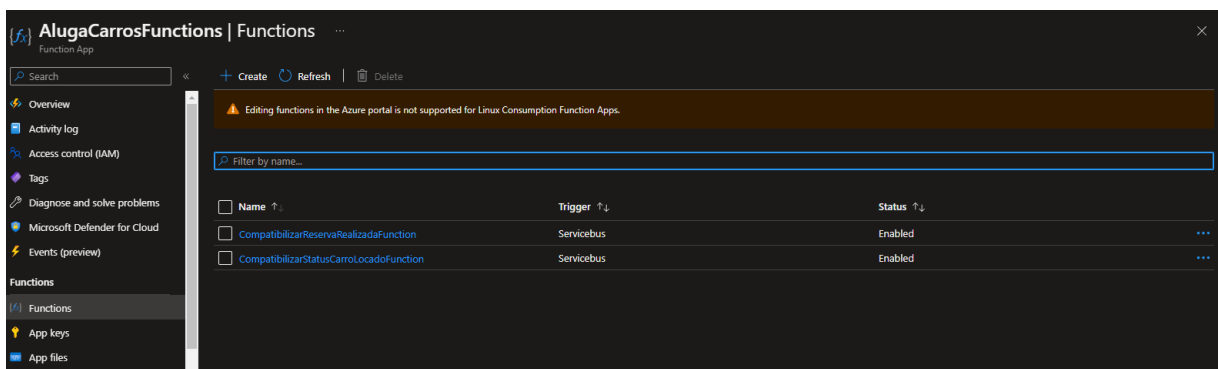
Script: select * from __efmigrationshistory

MigrationId	ProductVersion
20220606111354_Initial	6.0.5

Na primeira imagem é demonstrado como o Entity Framework realiza o controle das versões da base de dados que devem ser aplicadas. Quando uma nova versão é gerada, ele efetua a comparação com a versão anterior e gera os comandos SQL necessários para equivaler o esquema da base de dados ao código fonte da aplicação. Na segunda imagem é demonstrado que no banco de dados, é criada uma tabela com o nome “__efmigrationshistory” para guardar o histórico das versões de *Migrations* aplicadas para este banco. O nome salvo é o mesmo do arquivo gerado no código fonte.

Atributo de Qualidade:	Eficiência
Requisito de Qualidade:	As compatibilizações de movimentações com serviços já existentes devem ser realizadas através de Azure Functions
Preocupação:	
Quando uma operação que envolve algum outro serviço é efetuada no sistema, se dá a necessidade de atualizar as informações no outro serviço. Este serviço pode estar apresentando erros, estar fora por qualquer motivo, ou mesmo ser impossível de realizar a operação naquele momento. A utilização de Azure Functions garante que este processo seja assíncrono, o que não vai impedir nosso sistema de operar normalmente porque esta ação poderá ser executada naquele momento, ou em um momento posterior. Outra preocupação é em relação aos custos. Manter um serviço no ar para realizar estas operações pode vir a ser muito custoso, e as Azure Functions possuem um valor muito baixo para sua utilização	
Cenário(s):	
Cenário 4	
Ambiente:	
Operação do sistema	
Estímulo:	
Processamento assíncrono, economia de custos	

Mecanismo:	
Azure Functions	
Medida de resposta:	
Utilizar Azure Functions para as compatibilizações com demais sistemas serem assíncronas ao menor custo possível	
Considerações sobre a arquitetura:	
Riscos:	Por ser um processamento assíncrono, podem vir a ocorrer erros que não irão impedir o funcionamento correto do sistema à primeira vista, é necessário a implantação de monitoramento, <i>logging</i> e alertas para identificação de possíveis problemas em suas execuções
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há



No painel acima estão demonstradas as Azure Functions criadas para este projeto. Elas trabalham com *ServiceBus Triggers*, e quando uma operação é realizada no sistema, é gerado um evento para uma fila do *ServiceBus*, e com esta mensagem as Functions são acionadas e efetuam seus devidos processamentos.

Sistema Balcão de Atendimento - Aluga Carros

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A aplicação web deve ser responsiva e funcionar perfeitamente nas últimas versões de todos os navegadores web modernos
Preocupação:	
O sistema deve estar apto a atender a necessidade dos usuários e funcionar perfeitamente nas versões mais recentes dos navegadores web modernos	
Cenário(s):	
Cenário 5	
Ambiente:	
Operação do sistema	
Estímulo:	
Funcionamento em qualquer navegador web de maneira responsiva	
Mecanismo:	
Bootstrap	
Medida de resposta:	
Foi escolhido o framework bootstrap para garantir o estilo e responsividade da aplicação, pois este já dispõe de muitas funcionalidades para atender a esta necessidade.	
Considerações sobre a arquitetura:	
Riscos:	O sistema não funcionar corretamente em versões mais antigas dos navegadores
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Google Chrome

Abrir Locação

Reserva	Cliente	Documento
OBPWMBDA	Júlio Franco	01234267899
Grupo	Data Locação	Data Devolução
B3 - Sedan Premium	14/09/2022	12/09/2022
Dias	Valor Reserva	Depósito Segurança
9	R\$ 2.109,24	R\$ 2.005,92
Veículo		
Selecione um Veículo		

Abrir Locação

2022 - Aluga Carros

Edge

Abrir Locação

Reserva	Cliente	Documento
OBPWMBDA	Júlio Franco	01234267899
Grupo	Data Locação	Data Devolução
B3 - Sedan Premium	14/09/2022	12/09/2022
Dias	Valor Reserva	Depósito Segurança
9	R\$ 2.109,24	R\$ 2.005,92
Veículo		
Selecione um Veículo		

Abrir Locação

2022 - Aluga Carros


Sistema Balcão de Atendimento - Aluga Carros

Safari no iPad

20:21 Quarta-feira 14 de setembro

alugacarros-balcaoatendimento.azurewebsites.net

100%

Balcão de Atendimento Home Reservas ▾ Locações ▾ Veículos 

Abrir Locação

Reserva

OBPWMBDA

Grupo

B3 - Sedan Premium

Dias

9

Veículo

Selecione um Veículo ▾

Abrir Locação

Cliente

Júlio Franco

Data Locação

14/09/2022

Valor Reserva

R\$ 2.109,24

Documento

01234267899

Data Devolução

12/09/2022

Depósito Segurança

R\$ 2.005,92

2022 - Aluga Carros

20:21 Quarta-feira 14 de setembro

alugacarros-balcaoatendimento.azurewebsites.net

100%


Balcão de Atendimento

Home

Reservas ▾

Locações ▾

Veículos



Abrir Locação

Reserva	Cliente	Documento
OBPWMBDA	Júlio Franco	01234267899
Grupo	Data Locação	Data Devolução
B3 - Sedan Premium	14/09/2022	12/09/2022
Dias	Valor Reserva	Depósito Segurança
9	R\$ 2.109,24	R\$ 2.005,92
Veículo		
Selecione um Veículo ▾		

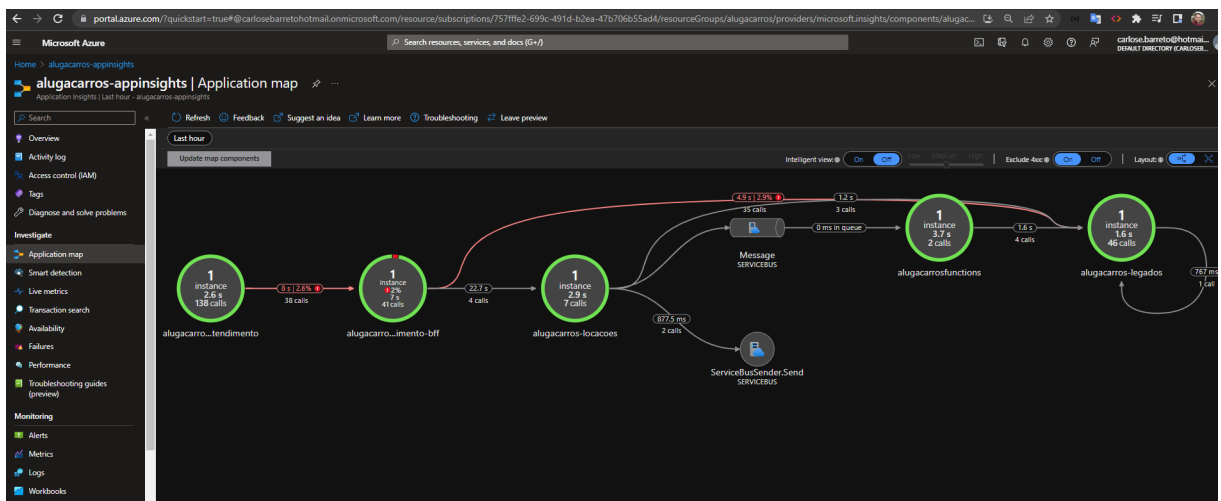
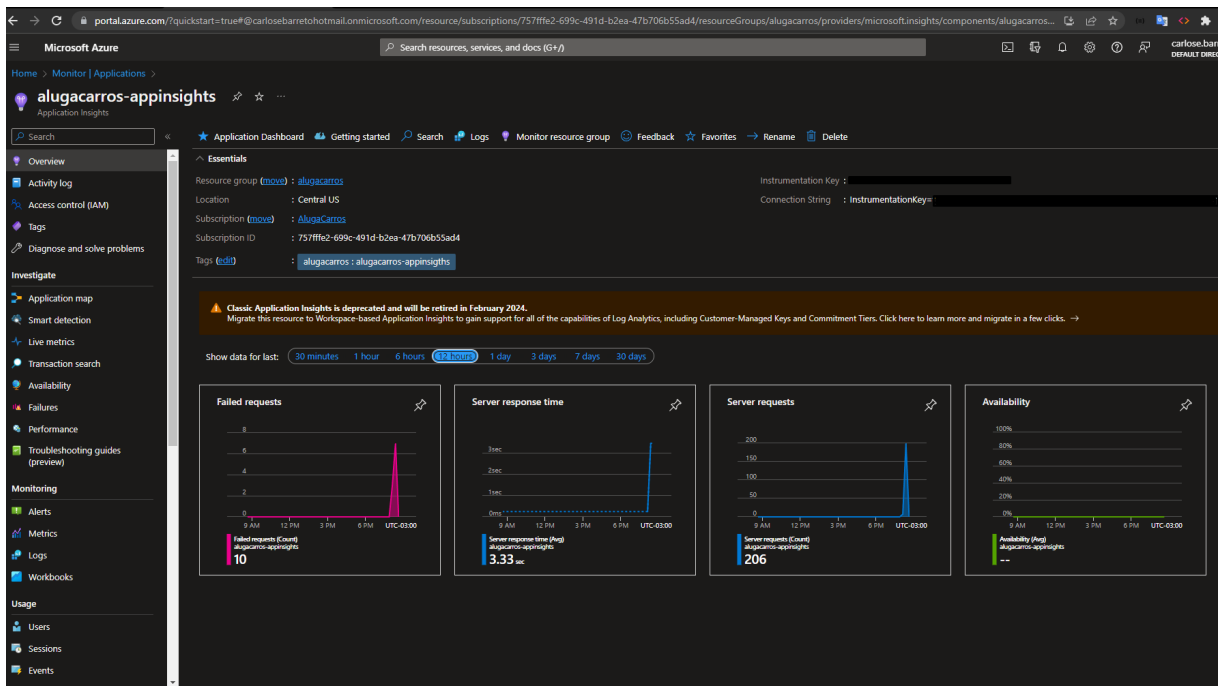
Abrir Locação

2022 - Aluga Carros

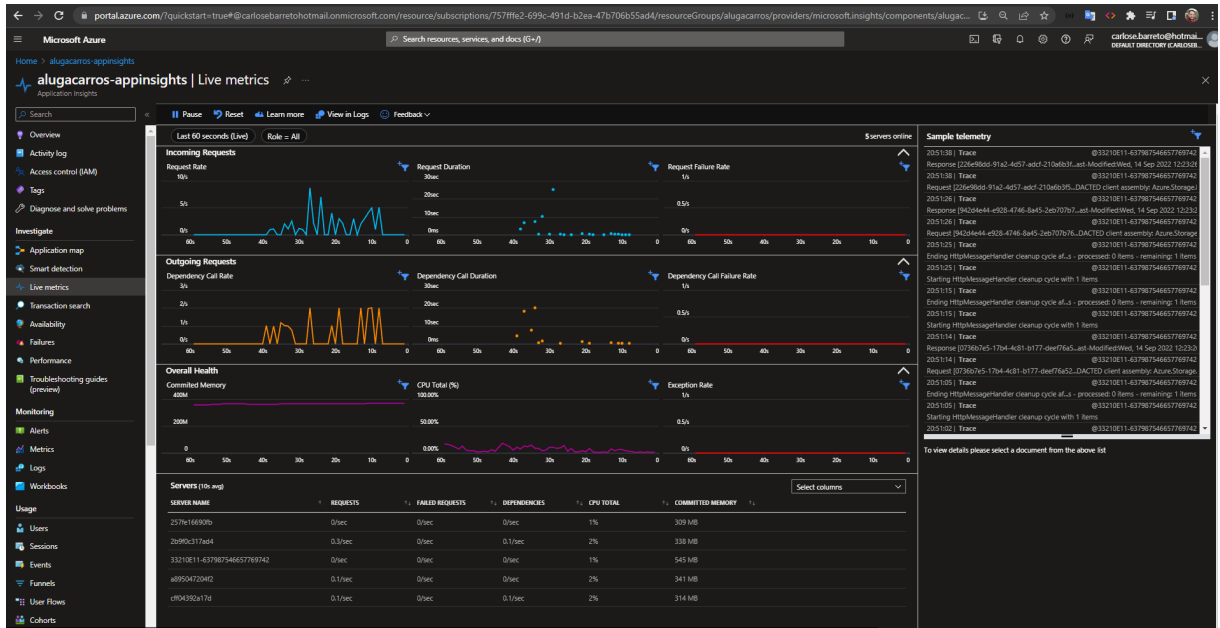
Como é possível observar nas imagens acima, a aplicação se comporta de maneira adequada em diferentes navegadores e também em tablets, nestes tanto em modo retrato quanto paisagem.

Atributo de Qualidade:	Rastreabilidade
Requisito de Qualidade:	A Observabilidade da aplicação e rastreamento de falhas deve ser realizada através da ferramenta Application Insights
Preocupação:	
Observabilidade deve ser uma das primeiras a serem tratadas quando pensamos em aplicações <i>cloud</i> e micro serviços. A verificação de possíveis erros pode se tornar muito difícil devido a quantidade de serviços se comunicando de maneira síncrona e até assíncrona, daí surge a necessidade de utilizarmos boas ferramentas para monitoramento e observabilidade e tornar este processo o menos moroso possível.	
Cenário(s):	
Cenário 6	
Ambiente:	
Monitoramento do sistema	
Estímulo:	
Verificar se a aplicação está se comportando de maneira adequada	
Mecanismo:	
Application Insights	
Medida de resposta:	
O Application Insights foi a ferramenta escolhida para monitoramento, por possuir um preço atrativo, muitas funcionalidades e supre todas as necessidades que a aplicação terá para telemetria, verificação de desempenho e disponibilidade e identificação proativa de problemas.	

Considerações sobre a arquitetura:	
Riscos:	Utilizando uma ferramenta proprietária do Azure, em caso de troca de provedor <i>cloud</i> , será necessário alterações nas aplicações
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há



Sistema Balcão de Atendimento - Aluga Carros



Nas imagens está demonstrado que o Application Insights permite acompanhar erros nas aplicações, visualizar o mapa de aplicações para identificar quais serviços se comunicam e também um acompanhamento em tempo real de utilização dos recursos das aplicações, dentre muitas outras funcionalidades.

6.4. Resultados Obtidos

Quadro 6 - Resultados Obtidos

Requisitos Não Funcionais	Teste	Homologação
RNF 01: O sistema deve ser apresentar disponibilidade 24 X 7 X 365	OK	OK
RNF 02: O sistema Balcão de Atendimento, bem como todos as suas APIs que se fizerem necessárias, devem possuir mecanismo de autenticação e autorização integrado ao serviço legado de Identidade, plataforma Microsoft Identity, utilizando JWT com chave assimétrica	OK	OK
RNF 03: O histórico de alterações do banco de dados deve ser controlado através das “Migrations” do Entity Framework	OK	OK

RNF 04: As compatibilizações de movimentações com serviços já existentes devem ser realizadas através de Azure Functions	OK	OK
RNF 05: A aplicação web deve ser responsiva, dispor de mecanismos de acessibilidade e funcionar perfeitamente nas últimas versões de todos os navegadores web modernos	OK	OK
RNF 06: A Observabilidade da aplicação e rastreamento de falhas deve ser realizada através da ferramenta Application Insights	OK	OK

7. Avaliação Crítica dos Resultados

Quadro 7 - Avaliação Crítica dos Resultados

Ponto avaliado	Descrição
Arquitetura das aplicações	A arquitetura se mostrou muito eficaz e capaz de ser evoluída sem grandes problemas devido ao alto nível de desacoplamento e utilização de aceleradores, bibliotecas com funcionalidades já desenvolvidas, que facilitam ainda mais este processo.
Aplicação Web responsivo	Facilita a operação do sistema, permitindo que qualquer navegador possa ser utilizado para operacionalizá-lo
Monitoramento	A ferramenta Application Insights mostrou-se muito eficaz para monitoramento e acompanhamento dos recursos da aplicação rodando no ambiente <i>Cloud</i>

Utilização de serviços do Microsoft Azure	A utilização de serviços como o Azure Service Bus, Azure Application Insights e as Azure Functions mostrou-se muito eficaz no desenvolvimento da aplicação, porém, são tecnologias que estão atreladas ao ambiente <i>cloud</i> em que a aplicação está rodando. Portanto, caso no futuro decida-se trocar de provedor <i>cloud</i> estes recursos terão que ser migrados para serviços do provedor escolhido ou para serviços agnósticos como RabbitMq e OpenTelemetry.
Ambiente <i>Cloud</i> Azure	O ambiente <i>cloud</i> da Microsoft foi muito simples para ser operado, necessitando de alguns cliques para montarmos nossa infraestrutura da aplicação. Os recursos SaaS utilizados seguem na mesma linha, agregando muita facilidade e agilidade no desenvolvimento da aplicação
Aplicações rodando em containers Docker	Graças às aplicações terem sido desenvolvidas para funcionarem em containers Docker, estão rodando no serviço Azure App Service, através de "Container Apps". Isto traz benefícios como economia de recursos computacionais em <i>cloud</i> , afinal um container para uma aplicação .Net 6 necessita de pouco recursos de processador e memória para funcionar a contento, e também proporciona uma mudança de provedor <i>cloud</i> no futuro ou mesmo serem implantadas em um <i>cluster</i> do Kubernetes.

Aplicações <i>Event Driven</i>	As aplicações foram projetadas para trabalharem com <i>event driven</i> , isto é, produzindo eventos em suas operações para que outras aplicações possam consumi-las, possibilitando um baixo acoplamento, possibilidade de assincronismo e maior velocidade nas interações dos usuários.
--------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

8. Conclusão

Este trabalho teve como objetivo apresentar a arquitetura para criação de um novo sistema fictício para uma locadora de veículos, que já possuía outros serviços legados que de certa forma, realizavam a mesma tarefa. A arquitetura proposta se mostrou muito aderente à proposta de que, além de migrar os serviços legados para aplicações *cloud native*, também era buscado maior performance, resiliência, tolerância a falhas, capacidade de evolução e manutenção da aplicação, e um maior desacoplamento.

O desenvolvimento deste relatório mostrou a importância de uma boa documentação de uma aplicação, e o quanto isto facilita aos arquitetos e desenvolvedores para darem início a novos projetos. Os diagramas apresentados são excelentes ferramentas que agregam muito no desenvolvimento de software. São fáceis de serem produzidos e deixam de uma maneira muito simples e clara a demonstração da informação ao qual são propostos.

A definição e criação de uma arquitetura limpa, orientada a serviços e *cloud native* proporcionou adquirir muitos novos conhecimentos e colocá-los em prática, que até então eram somente teoria.

Lições aprendidas:

- Um projeto bem definido proporciona uma fácil visualização e entendimento do objetivo final da arquitetura de nossas aplicações.
- Utilização de uma arquitetura de micro-serviços nos proporciona agilidade nas entregas, mas ao custo de agregar uma complexidade muito maior no gerenciamento e observabilidade destes.

- Azure Functions funcionam muito bem com uma arquitetura orientada a eventos para realização de processamentos assíncronos.
- Não é necessário utilização de SPAs para se produzir bons projetos de frontend. O Asp.Net Core 6 MVC se mostrou muito capaz de cumprir seu objetivo neste trabalho, resultando em uma aplicação rápida, responsiva, amigável e funcional.

Algumas melhorias que poderão ser realizadas no projeto são a adição de um serviço de *cache* distribuído como o Redis e a utilização de tópicos do Service Bus, assim as Azure Functions, ou qualquer outro serviço, que se interessarem nos eventos poderão se inscrever nos tópicos e não será necessário realizar o envio de vários eventos na criação de uma locação, por exemplo, proporcionando a aplicação uma maior facilidade de evolução e sem manutenções no caso da necessidade de novos assinantes dos eventos.

Vídeo da apresentação final: https://www.youtube.com/watch?v=NI1_B9XTq0Y