**FACULDADE UNIALFA**

ESPECIALIZAÇÃO EM ARQUITETURA E ENGENHARIA DE SOFTWARE

Trabalho final da disciplina: **Arquiteturas Orientadas a Serviços**

**APLICATIVO WEB PARA AGÊNCIA DE VIAGENS**

GOIÂNIA

11/2019**FACULDADE UNIALFA**

ESPECIALIZAÇÃO EM ARQUITETURA E ENGENHARIA DE SOFTWARE

Discentes:

Adão Jocelin Magalhães Júnior

Humberto Santana Melo e Silva

Rafael Borges de Aguiar

Rafael Gonzaga Gonçalves

Silvair Leite Soares

GOIÂNIA

11/2019

Sumário

[1. Introdução 1](#_Toc25390395)

[2. Solução 2](#_Toc25390396)

[2.1. Descrição do Projeto 2](#_Toc25390397)

[2.1.1 Funções do sistema 2](#_Toc25390398)

[2.1.2.1 Cadastro e alteração de clientes 2](#_Toc25390399)

[2.1.2.2 Reservas de viagens 3](#_Toc25390400)

[2.1.2.3 Pagamento das Reservas com cartões de crédito 3](#_Toc25390401)

[2.1.3 Padrões arquiteturais e de projetos (Design Patterns) 4](#_Toc25390402)

[2.1.3.1 Controller 4](#_Toc25390403)

[2.1.3.2 MVC 4](#_Toc25390404)

[2.1.3.3 Bridge 4](#_Toc25390405)

[2.1.4 Características da solução 4](#_Toc25390406)

[2.1.4.1 Comunicação 4](#_Toc25390407)

[2.1.4.2 Acoplamento 5](#_Toc25390408)

[3. Plataformas e Frameworks Utilizados 5](#_Toc25390409)

[3.1. Back-end 5](#_Toc25390410)

[3.1.1. Endereços do back-end 5](#_Toc25390411)

[3.2. Testes automatizados 6](#_Toc25390412)

[3.3. Front-End 6](#_Toc25390413)

# Introdução

O presente documento tem como objetivo documentar o desenvolvimento de uma aplicação web para uma empresa do tipo agência de viagens. De modo que a mesma seja capaz de fazer operações diversas, tais como cadastros e alterações de clientes, reservas de viagens incluindo: Voos, hotéis, carros e por fim efetuar o pagamento de uma ou mais reservas por meio de cartão de crédito. A arquitetura distribuída utilizada para dar solução a este projeto é a de microserviços.

# Solução

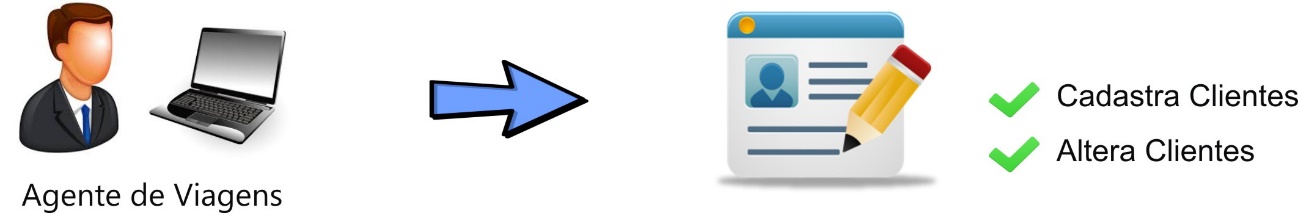
## Descrição do Projeto

Tendo em vista que o trabalho prático consta em desenvolver uma aplicação web para que um agente de viagem possa cadastrar e alterar clientes já cadastrados, realizar reservas de viagem e efetuar o pagamento das mesmas, o sistema foi composto pelos seguintes componentes definidos como micro serviços:

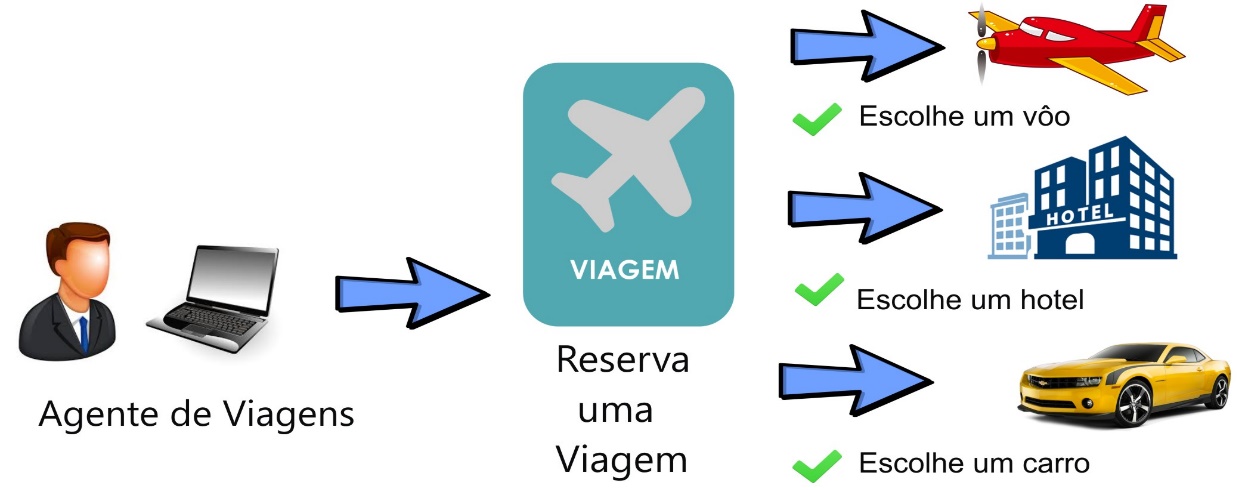
* API-Carro 🡪 realiza a consulta dos carros cadastrados;
* API-Cliente 🡪 Cadastro de Clientes: cadastra e altera clientes;
* API-Gateway 🡪 Redireciona as requisições para o serviço específico;
* API-Hotel 🡪 realiza a consulta dos hotéis cadastrados;
* API-Pagamento 🡪 faz o pagamento com cartão de crédito das reservas;
* API-Viagem 🡪 realiza uma reserva incluindo cliente, voo, hotel e carro;
* API-Voo 🡪 realiza a consulta dos voos cadastrados.

## Funções do sistema

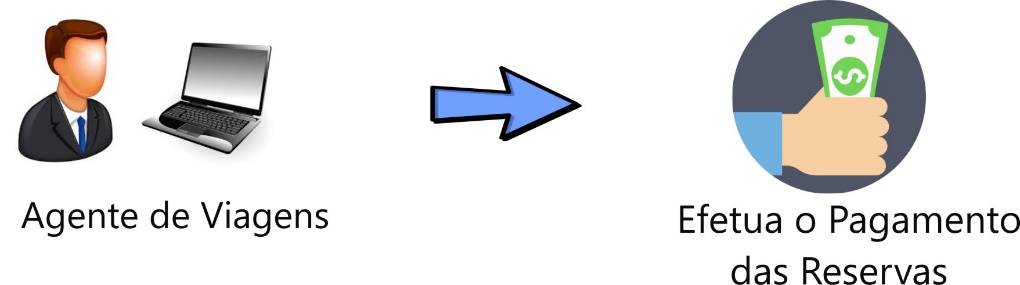
## **Cadastro e alteração de clientes**

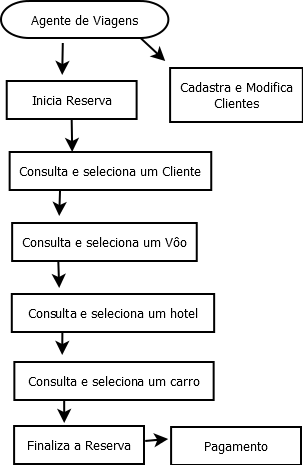


## **Reservas de viagens**



## **Pagamento das Reservas com cartões de crédito**



d.

## Padrões arquiteturais e de projetos (Design Patterns)

Foram utilizados os seguintes padrões no projeto:

## **Controller**

Este padrão foi aplicado em todas as APIS. Especificamente nos arquivos “routes/index.js” que são responsáveis por atender as requisições dos usuários, acionar os métodos internos da API que executam a ação solicitada e devolver a resposta para o usuário.

## **MVC**

Este padrão também foi aplicado em todas as APIS. Analisando o código podemos observar que temos os seguintes diretórios em todos os projetos: DAOS, MODELS e ROUTES, sendo que cada um destes diretórios desempenha suas respectivas funções dentro do padrão MVC.

## **Bridge**

Este padrão foi usado na API do gateway onde foi feito o envio das requisições para os métodos de testes de cada API (endereço\_api/health).

## Características da solução

## **Comunicação**

Como utilizamos APIS, a comunicação entre os serviços por padrão é via http.

## **Acoplamento**

O acoplamento dos serviços como geralmente ocorre em soluções que utilizam microserviços é baixo, pois um microserviço não depende diretamente da implementação do outro, mas sim apenas de um contrato, ou seja, podemos remover uma das APIs e substituir por outra que respeite o mesmo contrato que a solução continua funcionando.

* + - 1. **Modelos de Domínios**

Ficam encapsulados em cada microserviço que tem escopos bem definidos e responsabilidades únicas.

# Plataformas e Frameworks Utilizados

## Back-end

O projeto back-end é composto das seguintes APIs desenvolvidas em Node.JS:

## Endereços do back-end

**API-Gateway**<http://localhost:3000/>

**API-Carro**<http://localhost:3001/carros>

**API-Cliente**<http://localhost:3002/clientes>

**API-Hotel**<http://localhost:3003/hoteis>

**API-Pagamento**<http://localhost:3004/pagamentos>

**API-Viagem**<http://localhost:3005/viagens>

**API-Voo**<http://localhost:3006/voos>

Todos os microserviços devem ser acessados através da API-Gateway

**API-Carro**<http://localhost:3000/carros>

**API-Cliente**<http://localhost:3000/clientes>

**API-Hotel**<http://localhost:3000/hoteis>

**API-Pagamento** <http://localhost:3000/pagamentos>

**API-Viagem** <http://localhost:3000/viagens>

**API-Voo** <http://localhost:3000/voos>

## Testes automatizados

A criação de teste unitários automatizados foi feita utilizando os seguintes frameworks:

**Mocha** -> <https://www.npmjs.com/package/mocha>

**Chai** -> <https://www.npmjs.com/package/chai>

## Front-End

O projeto front-end foi construído utilizando-se se os seguintes frameworks:

**Angular** 🡪 <https://angular.io/>

**Angular Material** 🡪 <https://material.angular.io/>