# Plano de Testes para o Projeto Digital Booking – Digital Cars

## Introdução

### Objetivo

O objetivo deste documento é apresentar um plano estruturado de casos de teste para o Projeto Digital Booking – Digital Cars. Este documento apresenta os objetivos abaixo:

- Identificar as funcionalidades a serem testadas no projeto;
- Listar as ferramentas e/ou tecnologias a serem utilizadas nos testes;
- Recomendar uma estimativa de cobertura de testes para mitigar possíveis erros na aplicação;

### Escopo

O plano de testes detalhado neste documento refere-se aos testes de requisitos, testes de código e teste de funcionamento da API desenvolvidos no projeto Digital Booking – Digital Cars. A execução correta de todos os testes assegura a correta integração entre as áreas de frontend, backend, banco de dados e infraestrutura.

As telas da interface de usuário a serem testadas são as seguintes:

- Tela de registro do usuário;
- Tela de login do usuário;
- Tela de Home;
- Tela de filtro dos produtos;
- Tela de detalhes dos produtos;
- Tela de cadastro de reserva de um produto;
- Tela de minhas reservas;
- Tela de cadastro de um produto (apenas usuário ADMIN)

Os testes de código com JUnit devem ser das seguintes classes e métodos:

- Categoria Service:
    - Cadastrar categoria;
    - Listar todas as categorias;
    - Buscar categoria por id;
- Cidade Service:
    - Cadastrar cidade;
    - Listar todas as cidades;
    - Buscar cidade por nome;

- Produto Service;
    - Cadastrar produto;
    - Listar todos os produtos;
    - Buscar produtos por id;
    - Buscar produtos por id da categoria;
    - Buscar produtos por id da cidade;
    - Buscar produtos de acordo com as datas inicial e final;
    - Buscar produto por id da cidade e de acordo com as datas inicial e final;
    - Buscar produtos por id da categoria e de acordo com as datas inicial e final;
- Reserva Service;
    - Cadastrar reserva;
    - Buscar reserva por id;
- Usuário Service:
    - Cadastrar usuário;
    - Buscar usuário por id.

A APIs devem ser testadas utilizando o Postman devem ser as seguintes:

- API  de categoria:
    - Adicionar categoria;
    - Listar todas as categorias;
    - Editar categoria;
    - Eliminar categoria;
- API de Cidade:
    - Adicionar cidade;
    - Listar todas as cidades;
- API de Produto;
    - Adicionar produto;
    - Listar todos os produtos;
    - Buscar produtos por nome da categoria;
    - Buscar produtos por id da categoria;
    - Buscar produtos por nome da cidade;
    - Buscar produtos por id da cidade;
    - Buscar produtos por id da cidade e id da categoria;
    - Buscar produto por id da cidade e de acordo com as datas inicial e final;
- API de Reserva;
    - Cadastrar reserva;
    - Buscar reserva por id do produto;
- API de Usuário:
    - Cadastrar usuário;
    - Login de usuário.

## Tipos de testes

Para cada sprint devem ser executados os testes manuais de requisitos (Testes de fumaça).
A partir de segunda spint os testes das sprints anteriores de ser testados novamente para verificar se as implementações dos testes da sprint atual não quebraram as funcionalidades implementadas das sprints anteriores (Testes de regressão).
Nos testes de requisitos devem ser desenhados vários cenários para cada funcionalidade a serem testadas, devendo incluir tanto os cenários positivos, como os cenários negativos.

## Ferramentas e/ou tecnologias a serem utilizadas

JUnit
Mock MVC
Postman

## Cobertura mínima dos testes

Para mensurar a qualidade da aplicação recomenda-se uma cobertura de no mínimo 40% das linhas de códigos desenvolvidas para as APIs do projeto.