1. Introdução

Esta especificação detalha o escopo do projeto de desenvolvimento de uma nova plataforma para revenda de veículos automotores. O nome desse projeto se chamará “Car Sales”.

Link do repositório: https://github.com/LeonardoComar/fase-1-car-sales/tree/develop

2. Regra de negócio

A plataforma Car Sales, processará o cadastro e vendas de veículos automotores. A plataforma garantirá a segurança dos dados dos usuários e dos veículos cadastrados, através de mecanismos de criptografia e controle de acesso.

Os veículos disponíveis são específicos da empresa contratante, para ter um maior controle sobre as vendas, também será feito um cadastro dos colaboradores internos. A empresa vende apenas veículos automotores do tipo carro e moto.

Para atender a demanda do projeto, será definido os seguintes escopos:

Escopo dos clientes:

- O cliente poderá acessar a plataforma e aplicar diferentes filtros para visualizar os veículos automotivos disponíveis pelo vendedor.
- O cliente poderá solicitar contato com algum vendedor da empresa, informando: nome, e-mail, telefone, mensagem.

Escopo dos funcionários:

- Mediante perfil de acesso, o funcionário pode fazer o cadastro e edição dos veículos
- Visualizar as mensagens recebidas pelos clientes
- Efetuar a venda do veículo
- Visualizar informações de vendas
- Cadastrar colaboradores internos (administrador)

3. Domain-Driven Design (DDD)

Abordagem para o desenvolvimento de software que coloca o foco principal no domínio do problema, ou seja, na área de negócio que o software está destinado a resolver.

3.1 Linguagem ubíqua (dicionário)

Car Sales: Nome da plataforma para revenda de veículos automotores.

Usuário: Colaboradores internos que utilizam o sistema.

Perfil: Perfil associado ao usuário, que define acessos no sistema. Administrador e Vendedor.

Login: Texto para identificação e autenticação do usuário, sendo único do sistema.

E-mail: E-mail válido para recuperação de senha e notificações do sistema, sendo único no sistema para os colaboradores internos. No contexto do cliente, será um texto salvo para contato.

Cliente: Pessoas que enviam seus dados para contato através da plataforma e que efetuam a compra de veículos.

Veículos automotores: Carro ou moto que são anunciados na plataforma.

Mensagem de potencial cliente: O cliente consegue mandar mensagens pela plataforma pedindo contato, informando: nome, e-mail, telefone e a mensagem.

3.1.1 Dicionário para desenvolvedores

Pela experiência da equipe de desenvolvimento em desenvolver o código em inglês, considerar o seguinte de/para dos termos:

Usuário: User (modelo).

Perfil: role (determina se o usuário é administrador ou vendedor).

E-mail: email.

Login: login.

Cliente: Client (modelo).

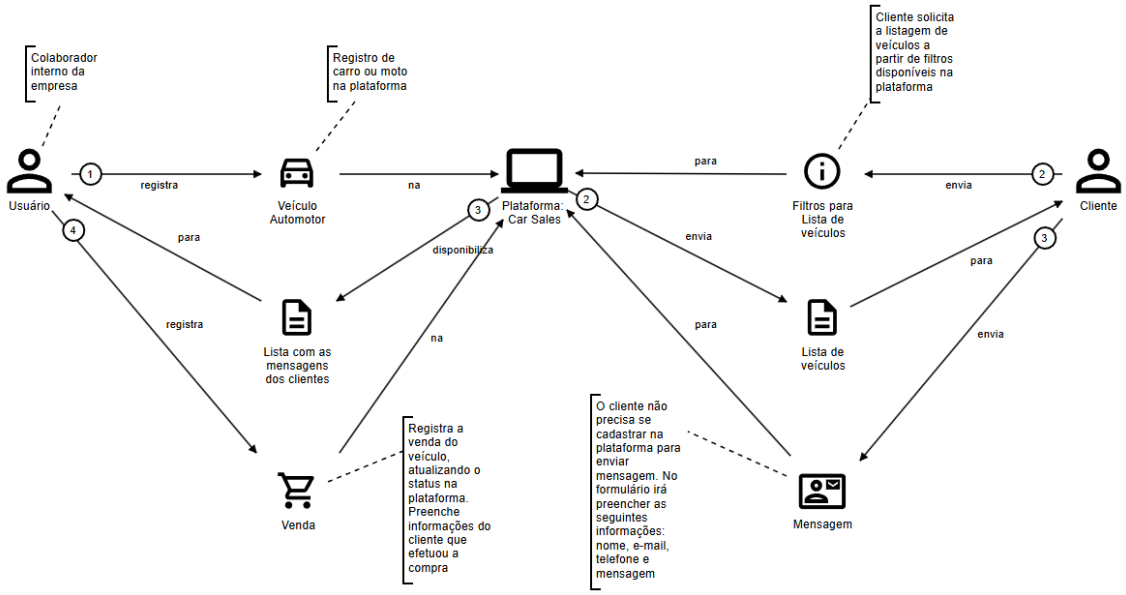
Veículos automotores: MotorVehicles (modelo). Herança para dividir os modelos em Car e Motorcycle.

Mensagem do cliente: Message

3.2 Domínio

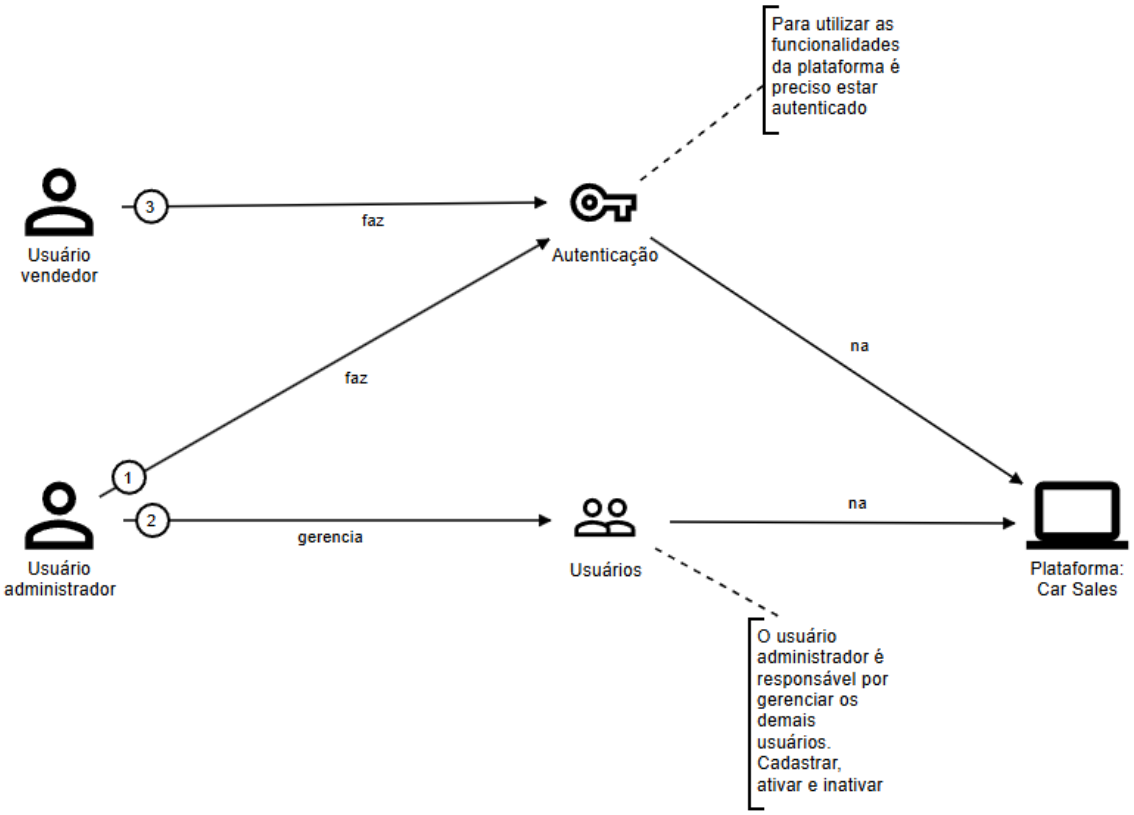
Sendo o gerenciamento dos veículos e sua visualização pelos clientes o coração da aplicação, entende-se como sendo a base principal do projeto.

Segue o Domain Storytelling do Domínio da aplicação:



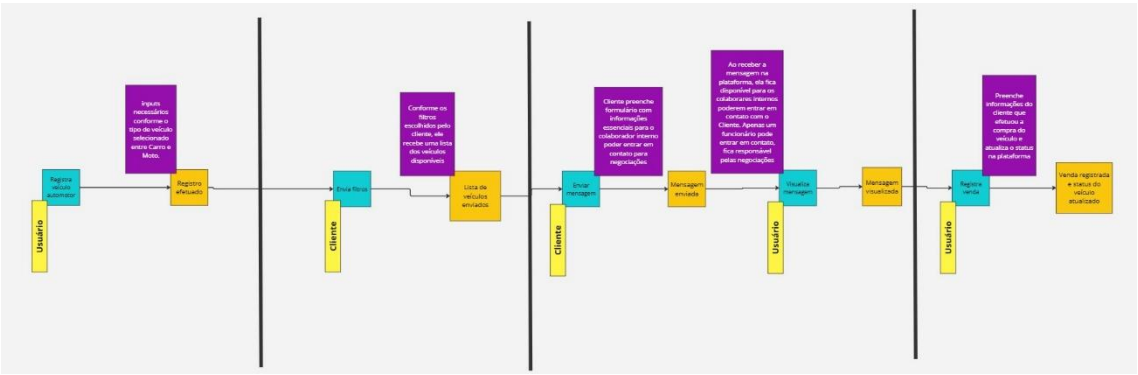
3.3 Subdomínio de suporte: Autenticação e gestão de usuários

O gerenciamento de usuários e autenticação será realizada pela própria plataforma. Para facilitar a visualização e criar uma diferenciação entre o usuário administrador e o usuário vendedor, foi criado o Domain Storytelling da autenticação e gestão de usuários:



3.4 Event Storming

Link para acessar:
https://miro.com/app/board/uXjVJbqGGkY=/?share_link_id=995764203385



4. Requisitos

4.1. Requisitos funcionais

Cadastro de veículos: Realizar o gerenciamento dos veículos pela plataforma.

Envio de mensagem de potenciais clientes pela plataforma: Os clientes que acessam a plataforma podem enviar mensagens para contato.

Efetuar venda de veículos: Os funcionários registram na plataforma que determinado veículo foi vendido. A partir disso o automóvel deixa de ser listado na plataforma para os clientes.

Gerenciamento de usuários: A plataforma deve incluir funcionalidades de cadastro, login e gerenciamento de usuários, com diferentes níveis de acesso.

4.2. Requisitos não funcionais

Desempenho: O sistema deve ser capaz de processar de forma rápida e eficiente todas as requisições.

Segurança: A plataforma deve garantir a segurança dos dados dos usuários, salvando as informações sensíveis com criptografia.

Disponibilidade: A plataforma deve ter disponibilidade de 99%.

Manutenibilidade: Código modular e bem documentado para facilitar futuras atualizações; Cobertura de testes; CI/CD.

5. Especificações técnicas para execução do projeto

A seguir segue o detalhamento das tecnologias escolhidas para solução do sistema Car Sales.

5.1 Linguagem de programação e framework

Uso da linguagem Python na versão 3.13.5, por ser versátil e amplamente utilizada no desenvolvimento web, com uma vasta comunidade e diversas bibliotecas. Principal framework utilizado: FastAPI. Framework moderno e de alta performance para desenvolvimento de APIs RESTful em Python. Baseado em padrões ASGI (Asynchronous Server Gateway Interface), proporcionando alta concorrência e escalabilidade.

5.2 Banco de dados

Tecnologia escolhida: MySQL

É um sistema de gerenciamento de banco de dados relacional (SGBDR) de código aberto, amplamente utilizado para armazenar e gerenciar dados de forma organizada.

5.4 Cobertura de testes

Tecnologia escolhida: pytest

É um framework poderoso e amplamente utilizado para testes em Python. Ele serve para simplificar a criação e execução de testes, além de oferecer diversos recursos que facilitam o processo de desenvolvimento e garantem a qualidade do código.

6. Arquitetura escolhida

Arquitetura monolítica: A escolha da arquitetura monolítica para aplicações de escopo limitado é uma decisão estratégica que visa otimizar o desenvolvimento, implantação e manutenção do sistema. Ao optar por essa abordagem, a equipe de desenvolvimento prioriza a simplicidade, eficiência e adequação ao escopo do projeto, garantindo um resultado satisfatório com menor investimento e complexidade.

7. Padrões de projeto

Arquitetura hexagonal: A Arquitetura Hexagonal, também conhecida como Arquitetura de Portas e Adaptadores, é um padrão de design que visa criar sistemas de software mais flexíveis, testáveis e fáceis de manter. Ela se baseia na separação de responsabilidades, isolando o núcleo da aplicação (a lógica de negócios) de suas dependências externas (bancos de dados, interfaces de usuário, etc.).