**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-2-car-sales>

**2. Regra de negócio**

A plataforma Car Sales, processará o cadastro e vendas de veículos automotores.

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.
* O cliente não precisa estar autenticado para fazer as ações acima

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 funcionários (administrador)

Sobre as informações referente aos veículos de carro e moto, teremos a seguinte divisão:

Informações em comum: Modelo, ano, quilometragem, combustível, cor, cidade, descrição complementar, preço, status e imagens

Informações específicas de carros: Carroceria, câmbio

Informações específicas de motos: partida, alimentação, cilindradas, refrigeração, estilo, tipo de motor, marchas e freio dianteiro/traseiro

**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:** funcionário da empresa que utilizam o sistema.

**Funcionário:** Recurso Humano da empresa.

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

**E-mail:** E-mail válido para recuperação de senha e notificações do sistema, sendo único no sistema para os funcionários. 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.

**Mensagens:** 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.

**Informações em comum dos veículos automotores:** model, year, mileage, fuel\_type, color, city, additional\_description, price, status e images

**Informações específicas de carros:** bodywork, transmission

**Informações específicas de motos:** starter, fuel\_system, engine\_displacement, cooling, style, engine\_type, gears, front\_rear\_brake

**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:

Mapa com linhas pretas em fundo branco

O conteúdo gerado por IA pode estar incorreto.

**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:

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**3.4 Event Storming**

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

**Gráfico, Gráfico de cascata

O conteúdo gerado por IA pode estar incorreto.**

**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.

**Disponibilidade**: A plataforma deve ter disponibilidade de 99%.

**Manutenibilidade**: Código modular e bem documentado para facilitar futuras atualizações

**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.

**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**

**Clean Architecture (Arquitetura Limpa):** é um padrão de design que visa criar sistemas de software robustos, escaláveis e de fácil manutenção, com foco na separação de preocupações e na independência de fatores externos.

Ela organiza o código em camadas concêntricas, onde a regra principal é a Regra de Dependência: as dependências devem sempre apontar para dentro, ou seja, as camadas externas (como Frameworks, UI e Banco de Dados) podem depender das internas, mas as camadas internas (como as Entidades e Regras de Negócio) não podem ter conhecimento sobre as externas.

Dessa forma, o núcleo da aplicação (a lógica de negócios, ou Entidades e Casos de Uso) permanece completamente isolado de suas dependências tecnológicas, garantindo que mudanças na interface de usuário, no banco de dados ou no *framework* não afetem as regras de negócio essenciais.