UNIVERSIDADE FEDERAL DE MATO GROSSO

CAMPUS ARAGUAIA

ACADÊMICO: ANTHONY MUNIZ PRADO DE OLIVEIRA

RELATÓRIO TRABALHO 1 DA MATÉRIA DE LABORATÓRIO DE BANCO DE DADOS - DESENVOLVIMENTO DO SISTEMA LOCACAR

Barra do Garças-MT, 2023.

RELATÓRIO DO TRABALHO 1 DA MATÉRIA DE LABORATÓRIO DE BANCO DE DADOS - DESENVOLVIMENTO DO SISTEMA LOCACAR

Nome do Sistema: Sistema Locacar.

Prof. Dr. Linder Cândido da Silva.

Instituição: Universidade Federal de Mato Grosso, *Campus Araguaia*.

Resumo: Este relatório descreve um projeto no qual um sistema para uma locadora de veículos é implementado. O projeto é dividido em três partes distintas que são elas: Modelagem Conceitual, Mapeamento EER para Relacional, Integração do Banco de Dados com Aplicação.

Sumário

INTRODUÇÃO

Este relatório que como foi falado visa descrever o processo de desenvolvimento do 1º trabalho de Laboratório de Banco de Dados, seguindo as seguintes etapas:

Parte 1 Modelagem Conceitual Nesta fase, é realizada a modelagem conceitual do banco de dados. Um diagrama Entidade-Relacionamento (EER) é projetado, considerando os requisitos fornecidos pelo professor. Os requisitos incluem informações sobre clientes, carros disponíveis, aluguéis realizados e outras informações relevantes. O diagrama EER é projetado para representar essas entidades e seus relacionamentos. Requisitos fornecidos:

> 1. O banco de dados mantém informações sobre os CLIENTES. Cada CLIENTE tem um ID único (assume um valor inteiro auto incrementável pelo SGBD), um Nome (uma string) e um Telefone (uma sequência de 12 caracteres como “66999436679”).

2. O banco de dados mantém informações sobre os CARROs disponíveis para locação, que são categorizados de acordo com seu tipo. Existem cinco tipos principais: COMPACTO, MÉDIO, GRANDE, SUV (Veículo utilitário esportivo) e CAMINHÃO. Cada tipo de carro tem um valor de diária e um valor semanal (suponha que todos os carros do mesmo tipo tenham as mesmas taxas de aluguel).

3. Cada CARRO tem um Id (um número único), Marca (Chevy, Toyota, Ford, etc), modelo e Ano (2015, 2014, etc).

4. O banco de dados deve armazenar todos ALUGUEIS de carros já realizados, bem como distinguir os alugueis ativos, os agendados e os finalizados. Para cada ALUGUEL, as informações mantidas incluirão o CARRO e o CLIENTE específicos, bem a DataInicial e o número de dias (a DataRetorno pode ser calculada a partir da DataInicial e NumeroDeDias). Cada aluguel também terá o ValorDevido, que é um valor derivado que pode ser calculado a partir das demais informações.

5. O banco de dados também deve possibilitar o acompanhamento de quais CARROs estão disponíveis para locação em quais períodos, incluindo a verificação de choque de reservas.

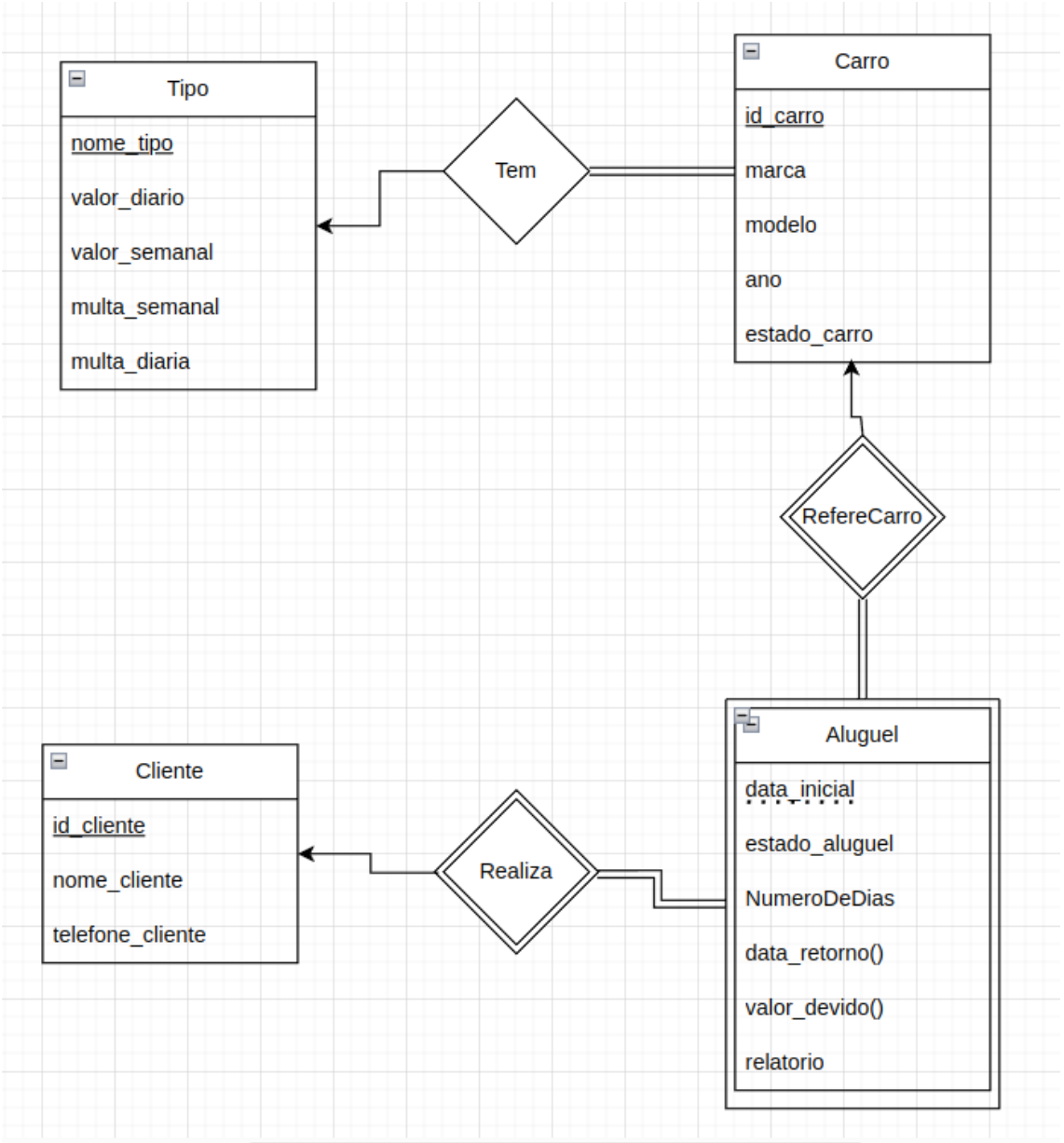
Parte 2 Mapeamento EER para Relacional, nesta etapa, o design EER é mapeado para um esquema de banco de dados relacional. São criadas tabelas correspondentes ao esquema relacional usando o MySQL. Restrições de chave e integridade referencial são especificadas no esquema. As instruções SQL CREATE TABLE são fornecidas para cada tabela, e as escolhas feitas durante o mapeamento são documentadas.

Parte 3: Integração do Banco de Dados com Aplicação, essa terceira parte do projeto envolve o desenvolvimento de um sistema para gerenciar cadastros de clientes, carros e aluguéis. O sistema deve garantir a consistência dos dados e incluir funcionalidades como cadastrar clientes e carros, agendar aluguéis e calcular valores a pagar com base nas taxas de aluguel. A interface de interação pode ser textual ou web. O sistema assegura que não ocorram conflitos de agendamento e registra as informações pertinentes, como valores em aberto.

Por fim, o trabalho visa atender aos requisitos coletados, proporcionando uma solução completa para a gestão de uma locadora de veículos. Cada parte do projeto é documentada, garantindo a compreensão das escolhas e implementações realizadas.

PARTE 1: MODELAGEM CONCEITUAL

Aqui foi projetado o seguinte diagrama EER:



Iniciando com a entidade Cliente, esta possui três atributos: 'id_cliente', que serve como chave primária, além dos atributos 'nome_cliente' e 'telefone_cliente'. A entidade Cliente mantém um relacionamento com a entidade Aluguel.

A entidade Aluguel, por sua vez, apresenta 'data_inicial' como sua única chave candidata à chave primária, modelada como uma entidade fraca, uma vez que não possui atributos distintos por si só e não pode existir independentemente das entidades Cliente e Carro. Além disso, conta com dois atributos derivados, 'data_retorno' e 'valor_devido', que são calculados com base em outros atributos do Banco de Dados (BD). O atributo 'estado_aluguel' é usado para armazenar o status atual do aluguel, que pode ser ativo, agendado, finalizado ou pendente. Adicionalmente, há o atributo 'NumeroDeDias', que é um atributo normal indicando o número de dias do aluguel. Por fim, temos o atributo 'relatorio', fornecido pelo funcionário no

momento do pagamento do aluguel para indicar se o aluguel foi pago corretamente ou se houve algum dano ao carro, entre outros detalhes.

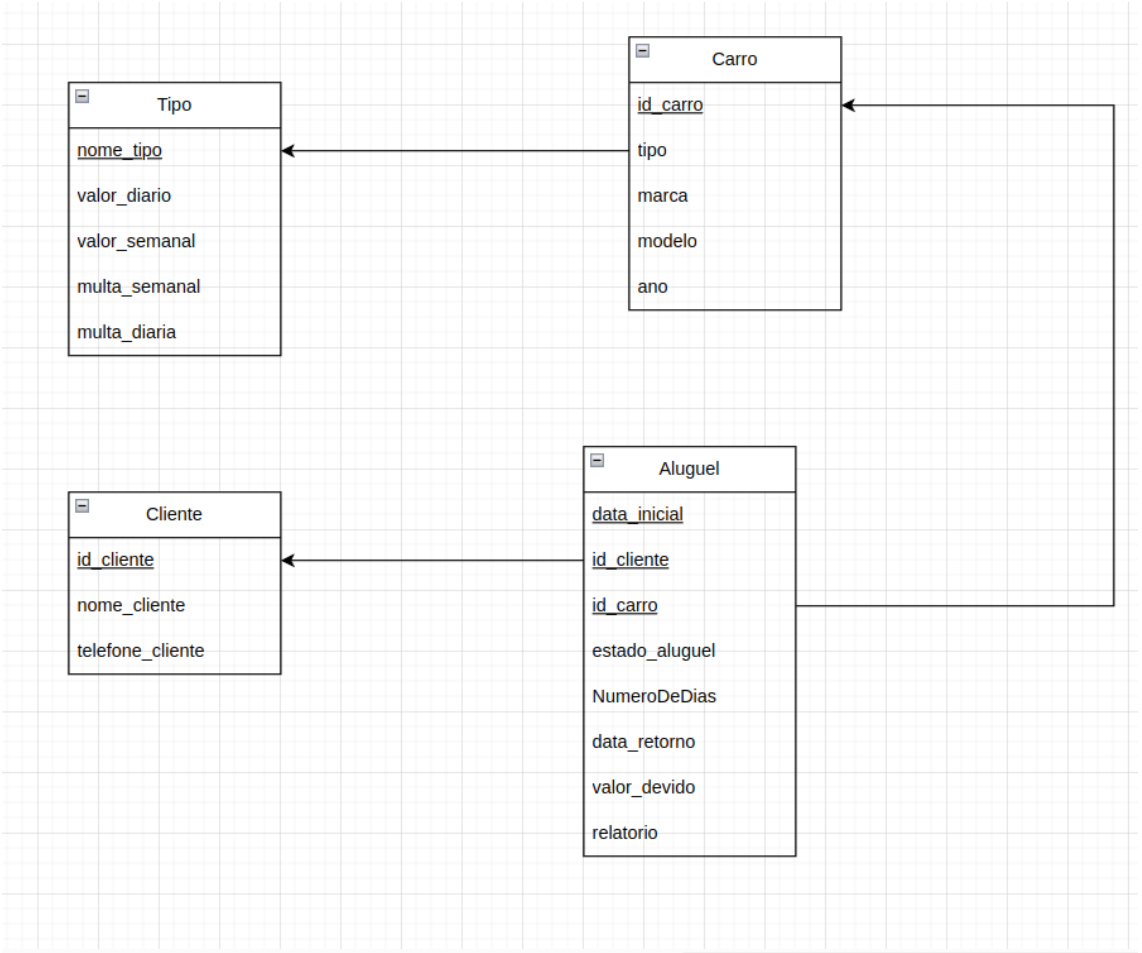
A relação entre Cliente e Aluguel foi projetada para permitir que um Cliente tenha múltiplos aluguéis em seu nome, enquanto cada aluguel está associado a apenas um cliente. Um cliente pode ter vários aluguéis ou nenhum, mas todos os aluguéis devem estar vinculados a pelo menos um cliente. Além disso, o Aluguel também se relaciona com a entidade Carro, onde um carro pode estar associado a vários aluguéis, mas cada aluguel está vinculado a apenas um carro.

No que diz respeito à entidade Carro, a relação com Aluguel foi modelada como 1:N, uma vez que os requisitos não especificaram que um aluguel está vinculado a apenas um carro. Esta modelagem foi realizada considerando a possibilidade de um cliente agendar ou alugar vários carros ao mesmo tempo em diferentes aluguéis, ou até mesmo nenhum carro, o que difere da relação Cliente e Aluguel. A entidade Carro possui 'id_carro' como chave primária, juntamente com outros atributos normais, como 'marca', 'modelo' e 'ano', conforme exigido pelos requisitos. Além disso, estabelece uma relação 1:N com a entidade Tipo, onde cada carro pode ter apenas um tipo, enquanto os tipos podem estar associados a vários carros.

Por fim, a entidade Tipo é identificada por 'nome_tipo' como chave primária e inclui atributos normais, como 'valor_diario', 'valor_semanal', 'multa_diaria' e 'multa_semanal', que foram adicionados posteriormente para permitir a cobrança de multas por aluguéis pendentes.

PARTE 2: MAPEAMENTO EER PARA RELACIONAL

Nesta fase do trabalho, eu realizei o mapeamento do modelo Entidade-Relacionamento Estendido (EER) para o modelo de dados relacional da seguinte maneira:



A tabela Cliente permaneceu inalterada, mantendo os mesmos atributos que foram herdados do modelo EER.

No que diz respeito à tabela Aluguel, houve uma modificação significativa. Ela agora inclui duas chaves estrangeiras: 'id_cliente' (referente à entidade Cliente) e 'id_carro' (referente à entidade Carro). Essas chaves estrangeiras fazem parte da superchave da tabela, que agora é composta por 'data_inicial', 'id_cliente', e 'id_carro'. Essa alteração reflete a relação entre os aluguéis, os clientes e os carros, garantindo a integridade referencial no banco de dados.

Quanto à tabela Carro, no processo de mapeamento da entidade homônima, adicionei o atributo de chave estrangeira 'nome_tipo', que faz referência ao tipo do carro a ser alugado. Essa relação estabelece uma conexão entre os carros e os tipos correspondentes, permitindo uma representação eficaz dos dados no banco de dados relacional.

Por último, a tabela Tipo, importada diretamente do modelo EER, não sofreu alterações. Ela mantém os mesmos atributos originados da entidade com o mesmo nome, preservando a estrutura de dados previamente definida.

PARTE 3: INTEGRAÇÃO DO BANCO DE DADOS COM APLICAÇÃO

Agora eu apresentarei as telas da aplicação com suas respectivas funções e também a modelagem do banco de dados com suas funções:

PARTE 3.1: Banco de Dados:

```
CREATE DATABASE LOCACAR;
USE LOCACAR;

CREATE TABLE Cliente(
    id_cliente INT AUTO_INCREMENT,
    nome_cliente VARCHAR(80) UNIQUE NOT NULL,
    telefone_cliente VARCHAR(15) NOT NULL,
    PRIMARY KEY (id_cliente)
);

CREATE TABLE Tipo(
    nome_tipo VARCHAR(30) CHECK (nome_tipo IN ('COMPACTO', 'MEDIO','GRANDE', 'SUV', 'CAMINHAO')),
    valor_diario DECIMAL(10,2) NOT NULL,
    valor_semanal DECIMAL(10,2) NOT NULL,
    multa_diaria DECIMAL(10,2) NOT NULL,
    multa_semanal DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (nome_tipo)
);

CREATE TABLE Carro(
    id_carro INT AUTO_INCREMENT,
    tipo VARCHAR(30),
    marca VARCHAR(30) NOT NULL,
    modelo VARCHAR(30) NOT NULL,
    ano INT NOT NULL,
    PRIMARY KEY (id_carro),
    FOREIGN KEY (tipo) REFERENCES Tipo(nome_tipo)
    ON UPDATE CASCADE
    ON DELETE RESTRICT -- empedira a deleção do tipo em caso de a ver alguem carro cadastrado com este tipo
);
```

Esta primeira Imagem demonstra a modelagem do Cliente, Carro e Tipo feitos no mysql podemos notar que o Tipo faz uma verificação de segurança para confirmar que só serão cadastrados os tipos com apenas esses 5 respectivos nomes, e demais tabelas foram modeladas como descrito no Modelo Relacional.

```sql
CREATE TABLE Aluguel(
    data_inicial TIMESTAMP,
    id_cliente INT,
    id_carro INT,
    estado_aluguel VARCHAR(10)
    CHECK (estado_aluguel IN ('ATIVO', 'AGENDADO','FINALIZADO', 'PENDENTE')) NOT NULL,
    NumeroDeDias INT,
    valor_devido NUMERIC(10,2),
    data_retorno TIMESTAMP,
    relatorio VARCHAR(300) CHECK (LENGTH(relatorio) <= 300), -- Usei a cláusula CHECK para impor o limite de caracteres
    PRIMARY KEY (data_inicial, id_cliente, id_carro),
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)
    ON UPDATE CASCADE
    ON DELETE RESTRICT, -- empedira a deleção do cliente em caso de a ver algum aluguel cadastrado com o id dele
    FOREIGN KEY (id_carro) REFERENCES Carro(id_carro)
    ON UPDATE RESTRICT -- Não permite update de carros cadastrados em um aluguel
    ON DELETE RESTRICT -- Não permite a deleção do carro se ele estiver referenciado aqui
);
```

A tabela Aluguel foi modelada com cuidado, incluindo checks para garantir a integridade do estado do aluguel. Embora nossa aplicação cadastre e modifique automaticamente o estado do aluguel, mantive essas verificações por questões de segurança, a fim de evitar a inserção de estados defeituosos no banco de dados, o que poderia comprometer o funcionamento adequado do sistema.

Além disso, observe que eu defini restrições 'ON DELETE RESTRICT' nas chaves primárias da tabela Aluguel relacionadas às tabelas Cliente e Carro. Isso significa que não é permitida a exclusão de um cliente ou carro se estiverem vinculados a um aluguel existente. Essa restrição visa manter a integridade dos dados e garantir que informações críticas relacionadas a aluguéis não sejam perdidas.

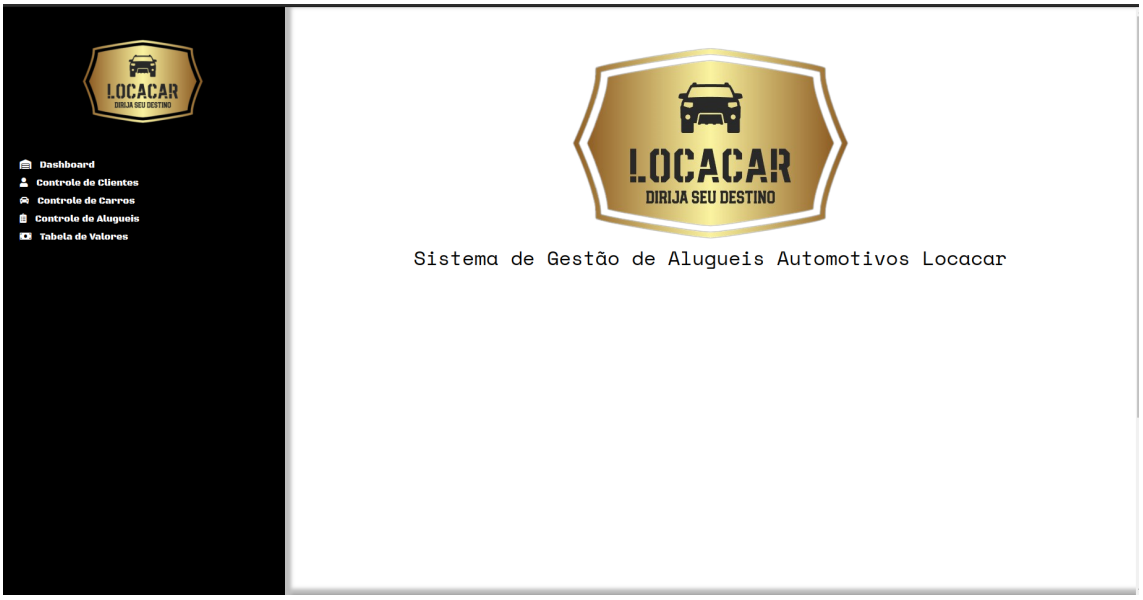
Da mesma forma, aplicamos a restrição 'ON UPDATE RESTRICT', que impede a modificação de registros na tabela Carro se esse carro estiver associado a um aluguel. Isso ocorre porque uma atualização no tipo de carro pode afetar o cálculo do valor do aluguel, o que, por sua vez, poderia impactar negativamente os registros existentes de aluguel. Essa precaução visa evitar atualizações que possam prejudicar a integridade dos dados relacionados aos aluguéis.

Vale ressaltar que, embora nossa aplicação tenha sido projetada para evitar atualizações em aluguéis já finalizados, ainda existe o potencial de riscos ao alterar o tipo de carro em aluguéis ativos ou em outros estados, como pendente.

Além das restrições mencionadas, o código SQL do projeto inclui triggers que evitam deleções e atualizações inadequadas. Também implementei algumas procedures úteis, como a 'atualizar_estado_alugueis()', que atualiza automaticamente o estado dos aluguéis quando é chamada pela aplicação. Essa procedure permite a transição automática de estados, como de 'AGENDADO' para 'ATIVO' ou de 'ATIVO' para 'PENDENTE'. É importante destacar que o estado 'FINALIZADO' só pode ser atribuído a um aluguel após o pagamento, o que significa que essa mudança só pode ser realizada pelo funcionário autorizado.

O segundo arquivo SQL contém inserções pré-cadastradas destinadas a fins de teste na aplicação, facilitando a avaliação e verificação do funcionamento do sistema.

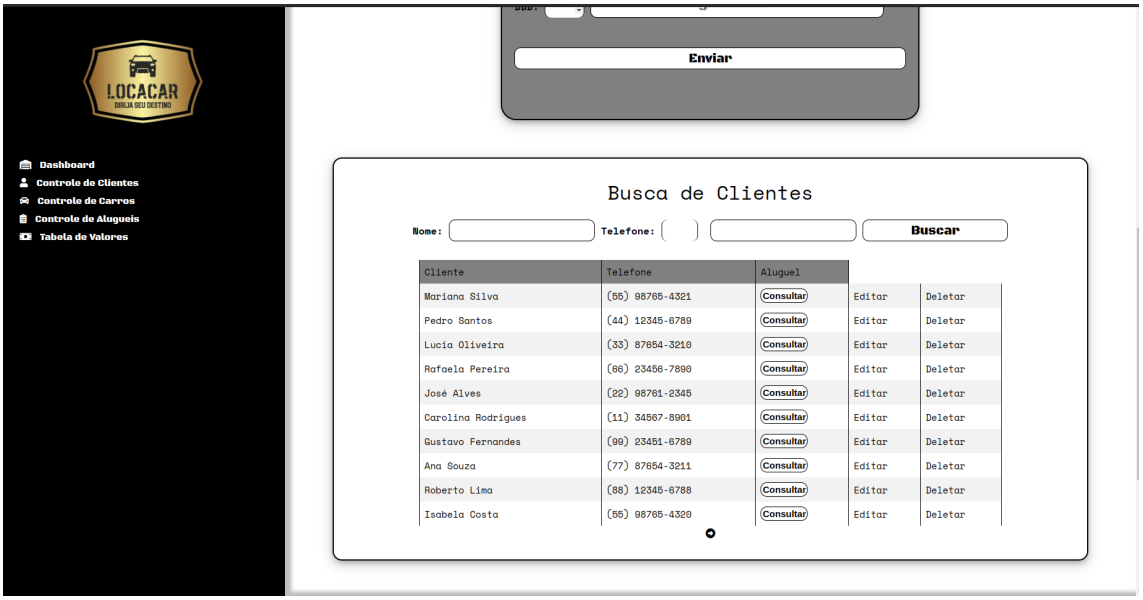
PARTE 3.2: APLICAÇÃO WEB:



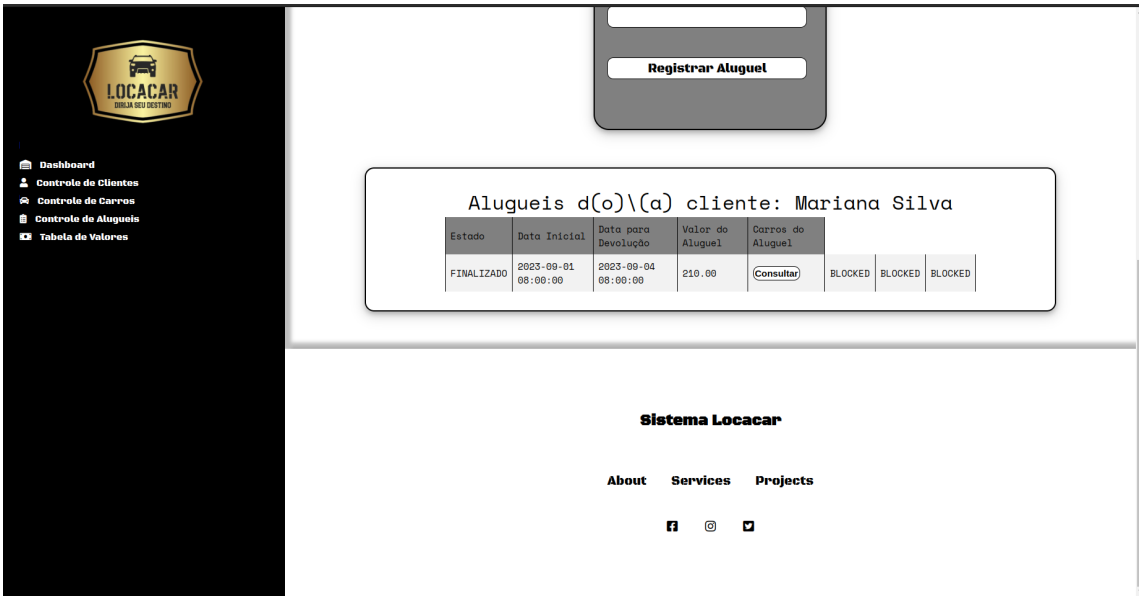
Esta é a página principal do site, servindo como ponto de entrada inicial ao abrir a aplicação. Sempre que acessada, ocorre a atualização dos estados dos aluguéis. Nesta página, você encontrará os seguintes elementos:

Mensagem Inicial: Uma mensagem de boas-vindas e a logo do site.

Menu Lateral Esquerdo: Um menu de navegação que oferece acesso rápido e fácil às diversas opções e funcionalidades disponibilizadas pelo sistema. Você pode utilizar esse menu para acessar as diferentes partes da aplicação.

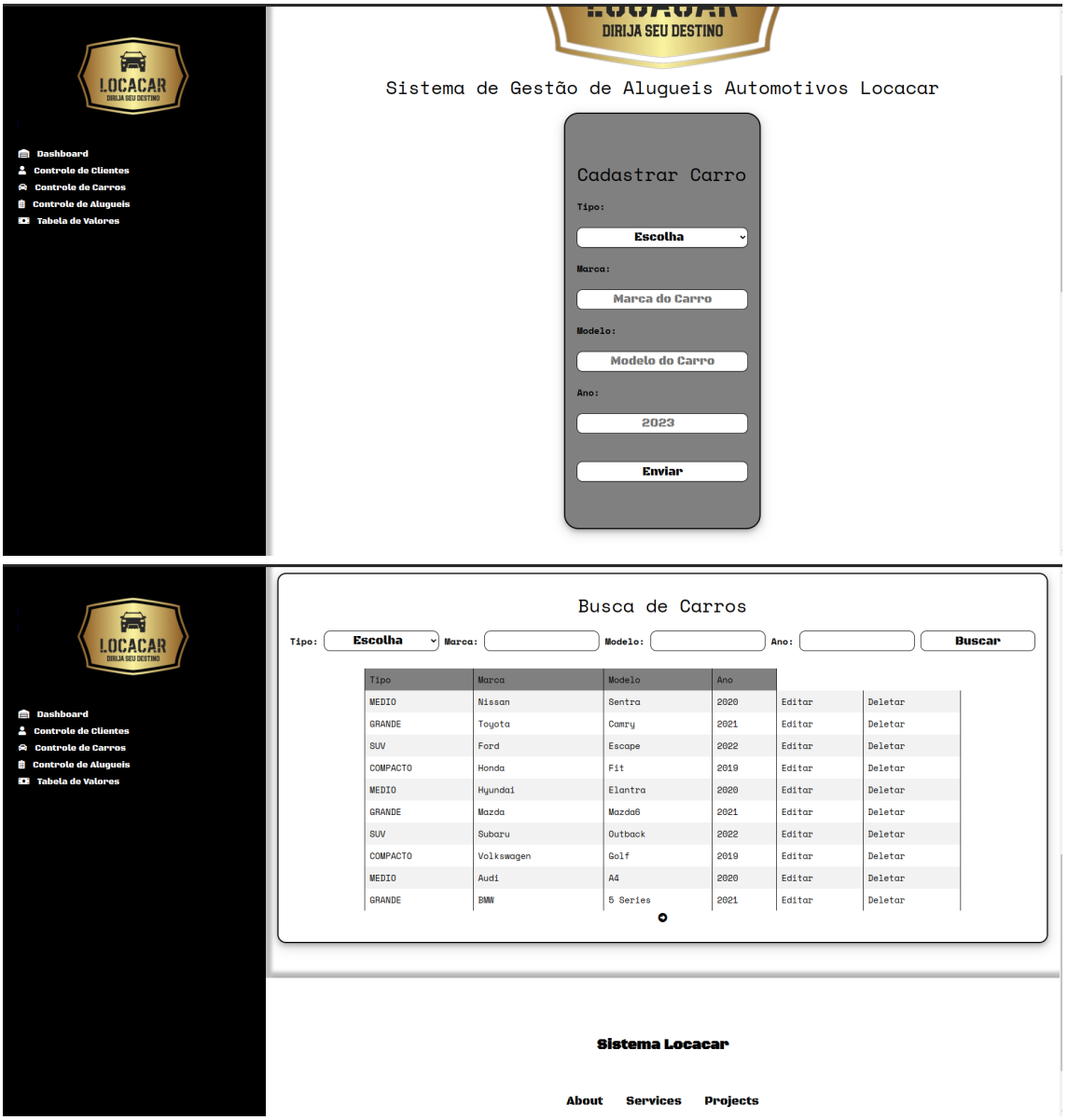


Essa é a segunda página do site que permite o cadastro de clientes e também a busca deles como é possível de visualizar nas imagens, note que é possível acessar o aluguel de cada cliente por essa página, que listará todo o histórico de aluguel daquele cliente, além de permitir também excluir o cliente(se o mesmo não for vinculado com outro aluguel e também permite editar o mesmo)

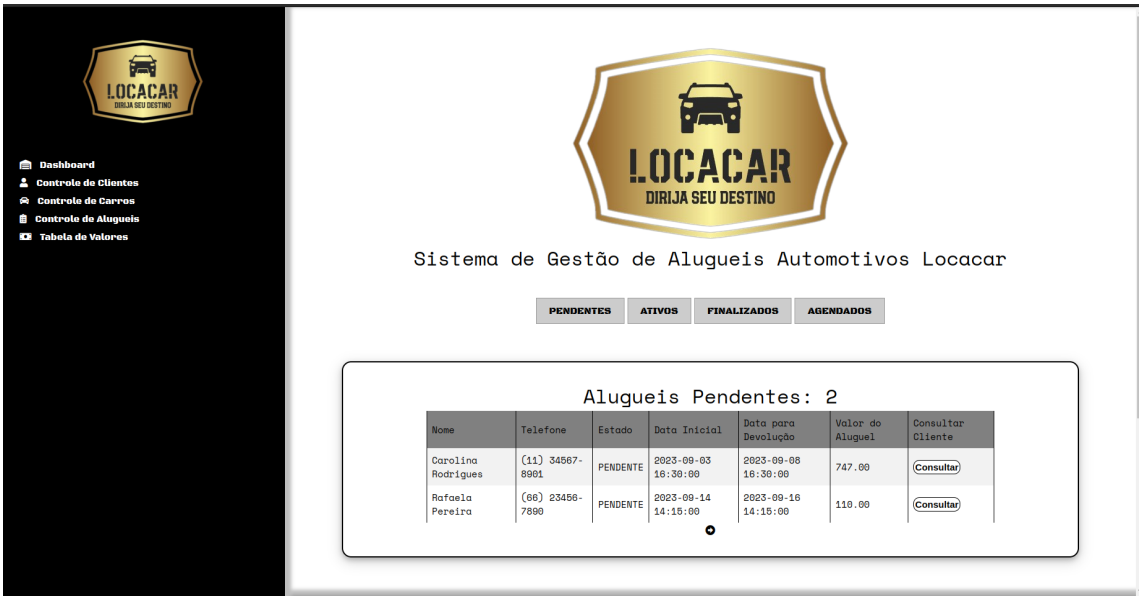


Ao clicar em consultar a seguinte página abrirá podendo ser realizado o cadastro de aluguéis para aquele cliente e até mesmo realizar a busca no histórico de aluguéis do mesmo. Também é possível ao clicar em consultar o carro vinculado ao aluguel e também o relatório referente a esse aluguel se tiver um a ser visto.

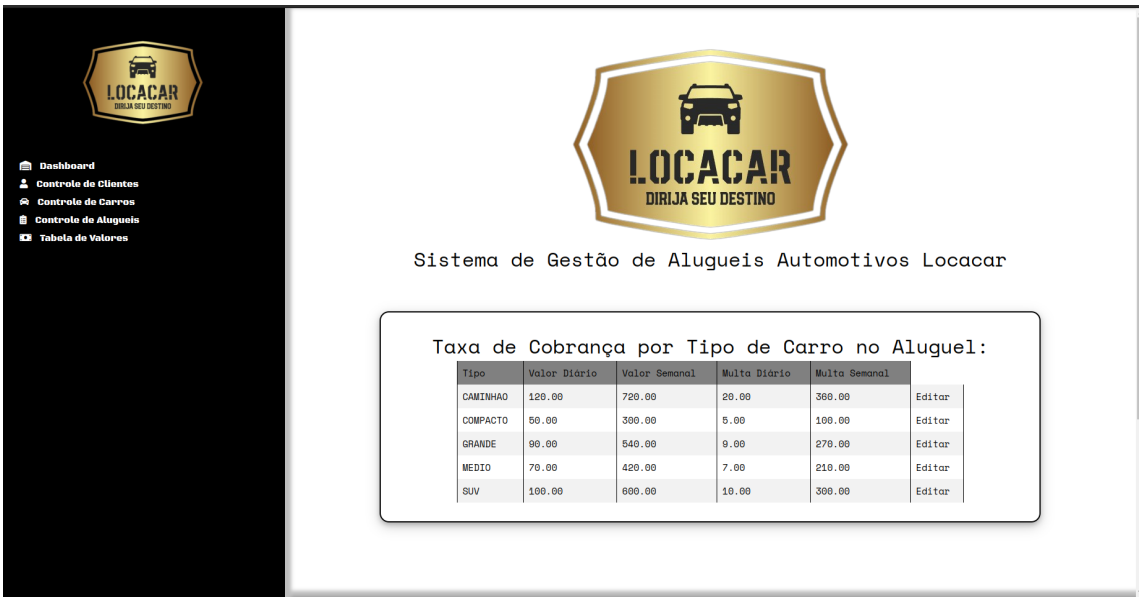
Este é um exemplo da página com seu respectivo relatório entregue e carro.



Está acima agora é a página de controle de carros, é permitido nesta página realizar o cadastro e edição dos carros já cadastrados se assim for permitido né(se o carro não for cadastrado em nenhum aluguel é permitido sua deleção e edição). É possível buscar o carro por seus atributos assim como na página de clientes.



Esta é a penúltima página da chamada de 'pagina de controle de aluguéis' que permite ver todos os aluguéis ativos, pendentes, finalizados com seus respectivos históricos.



Esta página é a última opção no menu e permite a visualização das taxas diárias e semanais, juntamente com suas respectivas multas, categorizadas por tipo de carro. Além disso, oferece a funcionalidade de edição dessas taxas.

É importante destacar algumas peculiaridades da aplicação:

1 - Pagamento da Dívida: A aplicação não permite que um cliente pague uma dívida inferior a 70% do valor atual. Isso garante que os pagamentos sejam substanciais e evita que o cliente tente quitar apenas uma parte insignificante da dívida.

2 - Multa Contínua: Mesmo após o pagamento da multa, o valor equivalente a 1 dia de multa continuará sendo cobrado diariamente enquanto o cliente não pagar o valor total da dívida. Esta abordagem incentiva o cliente a quitar o valor total em vez de prolongar a dívida.

3 - Estado PENDENTE: A aplicação mantém o estado do carro como 'PENDENTE' até que o valor total seja pago. Nesse momento, a data de retorno do carro é redefinida, e o carro passa a ter a data de devolução definida como o dia atual do pagamento. Portanto, a partir do dia seguinte ao pagamento, a multa voltará a ser cobrada com base nos dias e semanas de atraso que se acumularem.

4 - Transição de Estados: Se um pagamento for feito e o aluguel estiver no estado 'ATIVO', ele será alterado para 'FINALIZADO' se o pagamento for igual ao valor total devido. No entanto, se o cliente pagar, mas o valor pago não corresponder ao valor total em dívida, o aluguel passará para o estado 'PENDENTE'. Isso garante que apenas os pagamentos completos conduzam ao estado 'FINALIZADO', enquanto pagamentos parciais resultam em 'PENDENTE'.

5 - Nível de Isolamento: O nível de isolamento SERIALIZABLE foi escolhido para garantir a integridade e consistência dos dados em uma aplicação pequena, onde a simplicidade de desenvolvimento, a segurança dos dados e a prevenção de erros de concorrência são prioridades. Mesmo em um contexto com baixo volume de requisições, o SERIALIZABLE oferece a mais alta consistência e elimina problemas de concorrência, simplificando a lógica da aplicação e fornecendo uma camada de segurança adicional para dados críticos. Isso ajuda a prevenir erros inesperados e prepara a aplicação para possíveis futuros crescimentos.

Essas peculiaridades foram projetadas para garantir a integridade dos dados e incentivar o pagamento integral dos débitos. Elas são parte integrante do funcionamento eficiente da aplicação.

PARTE 3.3: TECNOLOGIAS UTILIZADAS:

No desenvolvimento desta aplicação web, foi utilizado uma série de tecnologias para criar uma experiência eficaz e interativa para os usuários. Abaixo, descrevemos brevemente cada uma delas:

Python: Python é uma linguagem de programação de alto nível conhecida por sua simplicidade e legibilidade. Utilizamos Python para a lógica do servidor, manipulação de dados e processamento de solicitações dos clientes.

Flask: Flask é um framework web em Python que permitiu criar a estrutura dessa aplicação de forma rápida e eficiente. Usei o Flask para gerenciar rotas, tratar solicitações HTTP e fornecer uma base sólida para o aplicativo.

JavaScript: JavaScript é uma linguagem essencial para tornar nossa aplicação web interativa. Ele é responsável por criar funcionalidades dinâmicas, validar formulários que foi onde eu mais o utilizei, animar elementos da página e melhorar a experiência do usuário.

HTML (Hypertext Markup Language): Utilizei HTML para definir a estrutura e o conteúdo das páginas web. Com HTML, podemos criar elementos como cabeçalhos, parágrafos, links e formulários, proporcionando a base para a exibição do conteúdo.

CSS (Cascading Style Sheets): CSS é crucial para dar estilo e design às páginas web. Com CSS, controlamos a aparência, o layout e a formatação dos elementos HTML, tornando nossa aplicação atraente visualmente e responsiva em dispositivos diferentes.

Font Awesome: Font Awesome é uma biblioteca de ícones vetoriais que enriquece a experiência do usuário, adicionando ícones a botões, links e elementos de navegação. Isso ajuda a tornar nossa interface mais intuitiva e visualmente atraente.