INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO** **CAMPUS – São Paulo**	
SPOWEB1		
Linguagem de Programação	**Projeto Sistema de Locação de Veículos**	**3º e 4º. Bimestre**
Profª. Claudete de Oliveira Alves	Observação: Equipe de 4 alunos	

Introdução ao Projeto

Imagine que sua equipe de desenvolvedores foi contratada pela "Mobilidade Flex", uma locadora de veículos que busca modernizar seus processos. O objetivo é criar um sistema de gestão completo que facilite as operações diárias, desde o cadastro de clientes e veículos até o controle de locações e devoluções, tornando o serviço mais ágil e eficiente.

História

A "Mobilidade Flex" nasceu como uma pequena empresa familiar com apenas dez carros. Com um serviço de qualidade, a empresa cresceu e hoje possui uma frota diversificada para atender a diferentes necessidades. No entanto, todo o controle ainda é feito em planilhas e papéis, o que gera erros e lentidão. Para continuar crescendo de forma organizada, a "Mobilidade Flex" precisa de um sistema digital que centralize todas as informações e automatize os processos de aluguel, garantindo uma melhor experiência para os clientes e uma gestão mais eficaz para os funcionários.

Requisitos Gerais do Sistema

- **Atores:** O sistema terá dois tipos principais de usuários:
  - **Cliente:** Pessoa que aluga os veículos da empresa.
  - **Funcionário:** Colaborador responsável por operar o sistema, cadastrar clientes, gerenciar a frota de veículos e processar as locações e devoluções.
- **Funcionamento**
  1. **Gestão de Frota:** O funcionário deve poder cadastrar novos veículos na frota, com informações como marca, modelo, ano, placa, cor, quilometragem, valor da diária, etc. Também deve ser possível remover veículos da frota (por venda ou acidente).
  2. **Gestão de Clientes:** O funcionário deve cadastrar os clientes que desejam alugar um carro ou atualizar as informações de clientes já existentes.
  3. **Processo de Locação:** Um cliente, após ser identificado no sistema, escolhe um veículo disponível. O funcionário registra a locação informando o tempo previsto de uso e outras informações que influenciam no valor. O sistema deve exigir um valor de caução antes de liberar o veículo.
  4. **Processo de Devolução:** Na devolução, o funcionário registra a data, hora e a quilometragem atual do veículo. O sistema deve verificar o estado do carro e calcular custos adicionais, como diárias extras ou danos. O valor final é calculado, e o restante da caução é devolvido, se for o caso.

Perguntas para Gerar Reflexões (Desenvolvimento do Projeto)

Estas perguntas são um guia para sua equipe pensar sobre as regras de negócio e funcionalidades do sistema de locação.

1. Sobre os Clientes e Funcionários:

- Quais informações são essenciais para o cadastro de um novo cliente (ex: nome, CNH, telefone)?
- O sistema precisa ter diferentes níveis de acesso para funcionários (ex: atendente, gerente)?
- Como o sistema deve lidar com clientes que possuem histórico de problemas (ex: atrasos, danos)?

2. Sobre a Frota de Veículos:

- Quais dados são importantes para cadastrar um veículo na frota?
- Como o sistema deve indicar se um veículo está disponível, alugado ou em manutenção?
- Como o valor da locação de um carro será calculado? (ex: por diária, por categoria, por km)

3. Sobre o Processo de Locação e Devolução:

- Como o funcionário poderá consultar os veículos disponíveis para uma data específica?
- Como o sistema irá calcular o valor total da locação, incluindo o tempo de uso e a caução?
- No momento da devolução, como o sistema calculará os custos extras por dias adicionais ou quilometragem excedida?
- Como serão registrados possíveis danos ao veículo e como isso impactará no custo final para o cliente?

4. Sobre a Interface e Experiência do Usuário (UX/UI):

- Como a tela de "painel de controle" do funcionário poderia exibir as informações mais importantes (carros alugados, devoluções previstas para hoje, etc.)?
- O processo de locação deve ser um passo a passo? Como seria essa sequência de telas?
- Como o sistema pode gerar um contrato de locação ou um recibo de devolução para ser impresso ou enviado ao cliente?

Requisitos por Disciplina

O projeto é interdisciplinar e cada área contribuirá com uma parte fundamental do sistema:

- **Artes Visuais e Design:** Responsável pela criação de toda a experiência visual do sistema. Isso inclui o desenvolvimento da interface gráfica, layout, escolha de tipografia e cores, e garantia de responsividade. A equipe deverá pensar na hierarquia das informações e dos elementos visuais.
- **JavaScript:** Será a base da lógica e interatividade do sistema.
    - **Parte 1 (3º Bimestre):** Foco nos aspectos fundamentais da linguagem, como criação de variáveis, condicionais, laços de repetição, arrays e funções para implementar as funcionalidades básicas.

- **Parte 2 (4º Bimestre):** Aplicação de conceitos de Orientação a Objetos (classes, objetos, atributos, métodos, encapsulamento, herança) para organizar e aprimorar o código.

- **Python:** Poderá ser utilizado para o desenvolvimento do back-end, focando nos aspectos fundamentais da linguagem e na conexão com um banco de dados.
- **Banco de Dados** (escopo das entregas conforme apêndice A)**:**
  - **Etapa 1 (3º Bimestre):** Modelo Conceitual (brModelo) + Dicionário de Dados
  - **Etapa 2 (4º Bimestre):** Entrega em SQLs separados + DOCX explicativo

Desenvolvimento e Entrega

O projeto será desenvolvido em duas grandes etapas, alinhadas com o 3º e 4º bimestres.

Parte 1: Estrutura, Documentação e Protótipo Funcional

- **Foco:** Planejamento, design da interface, documentação e implementação das funcionalidades essenciais com JavaScript fundamental.
- **Data de Entrega (Documentação, Apresentação e Código no Moodle): 19/09/2025 até as 23h55.**
- **Data da Apresentação:** (ver na aba da sua turma dentro do Moodle, serão as duas últimas aulas do mês de Novembro).

Requisitos da Parte 1

1. **Estrutura do Sistema:**
   - Desenvolver a interface de usuário (HTML e CSS) com seções para: cadastro de clientes, cadastro de veículos e um painel para visualizar a frota.
   - Implementar um menu de navegação claro para facilitar o acesso às diferentes áreas do sistema.
2. **Formulários Funcionais:**
   - **Clientes:** Criar o formulário para registrar novos clientes.
   - **Veículos:** Criar o formulário para cadastrar novos veículos, usando arrays ou objetos em JavaScript para simular o armazenamento dos dados.
3. **Funcionalidades Essenciais:**
   - Implementar a listagem de veículos, mostrando se estão "disponíveis" ou "alugados".
   - Permitir que o funcionário simule o processo de locação (sem salvar em banco de dados).
4. **Documentação:**
   - Elaborar um documento (normas ABNT) detalhando a estrutura do sistema.
   - Incluir wireframes ou diagramas de fluxo (ex: Figma) para ilustrar a interface e o funcionamento.
   - Banco de dados conforme apêndice

Parte 2: Código Aprimorado, Orientação a Objetos e Banco de Dados

- **Foco:** Refatorar o código, aplicar conceitos de Orientação a Objetos, implementar as funcionalidades completas e conectar o sistema a um banco de dados real.
- **Data de Entrega (Documentação, Apresentação e Código no Moodle): 08/11/2025**
- **Data da Apresentação:** (ver na aba da sua turma dentro do Moodle, serão as duas últimas aulas do mês de Novembro).

Requisitos da Parte 2:

1. **Melhorias no Código (Refatoração):**
    - Reorganizar o código para melhorar a legibilidade e a manutenção.
    - Implementar validações de dados nos formulários (ex: verificar se a placa do carro está no formato correto).
2. **Orientação a Objetos:**
    - Utilizar classes e objetos para representar as entidades do sistema (

Cliente, Veiculo, Locacao, etc.).

    - Aplicar conceitos como encapsulamento, herança e polimorfismo onde for relevante.
3. **Funcionalidades Adicionais:**
    - Implementar a lógica completa de locação e devolução, com todos os cálculos de valores.
    - Permitir a edição e exclusão dos cadastros de clientes e veículos.
    - Implementar um histórico de locações por cliente e por veículo.
4. **Integração com Banco de Dados:**
    - Conectar a aplicação ao banco de dados modelado na primeira etapa. Todas as operações deverão ser persistidas no banco.
5. **Documentação Atualizada:**
    - Atualizar o documento do projeto para refletir todas as mudanças, incluindo a nova arquitetura com Orientação a Objetos e exemplos de código.
    - Banco de dados conforme apêndice

Artigos e Sites Interessantes

- **Refatoração de Código:**
    - Refactoring Guru:

https://refactoring.guru

    - Martin Fowler's Blog:

https://martinfowler.com

- **Orientação a Objetos:**
    - IBM Developer (Artigos sobre OO):

https://developer.ibm.com/technologies/java/articles/j-intro/

    - GeeksforGeeks (Conceitos de OO):

https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

- **Modelos de Wireframes e Design de Interfaces:**
    - Smashing Magazine:

https://www.smashingmagazine.com

    - Miro (O que é Wireframe):

https://miro.com/pt/wireframe/o-que-e-wireframe/

    - Nielsen Norman Group:

https://www.nngroup.com

Considerações Finais

1. A documentação escrita deve seguir as normas ABNT.
2. A entrega no Moodle será em um formulário com 3 questões: Questão 1 para a

apresentação (PDF), Questão 2 para a **documentação (PDF)**, e Questão 3 para o **projeto (.zip)**.

3. A entrega deve ser realizada por apenas um integrante da equipe, mas os nomes de todos devem constar nos documentos.
4. **Prazos:** As datas de entrega são fixas. Haverá uma semana adicional para entregas atrasadas, com um decréscimo de 20% na nota. Após esse período, os projetos não serão aceitos.
5. **Qualidade de Código:** Serão avaliados: nomes de variáveis e funções seguindo boas práticas, indentação, organização de arquivos (HTML, CSS, JS separados) e outros padrões de código limpo.
6. Lembre-se de incluir as referências bibliográficas utilizadas no final do seu documento.

Bons Estudos!!!

APÊNDICE A

Etapa 1 (3º Bimestre): Modelo Conceitual (brModelo) + Dicionário de Dados

1) Modelo Conceitual (DER no brModelo)

- Ferramenta: **brModelo** (versão livre).
- Entregáveis:
  - **Arquivo editável do brModelo** (extensão nativa da ferramenta) contendo o **DER** completo.
  - **Imagem/Export do DER** (PNG/JPG) legível.
- O DER deve contemplar:
  - **Entidades** com nomes significativos e **atributos** claramente nomeados.
  - **Identificadores** (chaves primárias conceituais) definidos para cada entidade.
  - **Relacionamentos** com nomes verbais ou substantivos claros.
  - **Cardinalidades e participações** corretamente definidas.
  - **Generalizações/Especializações** (opcional), quando agregarem valor ao domínio.
- Convenções sugeridas (adaptar ao padrão da turma):
  - Nomes de entidades no **singular** (ex.: Pessoa, Curso).
  - Atributos em **camelCase** ou **snake_case** de forma consistente (ex.: dataNascimento ou data_nascimento).
  -

2) Conversão Conceitual → Lógico (implícita, sem entrega)

- **Converter** o Modelo Conceitual em Modelo Lógico, definindo as tabelas, atributos, chaves primárias e estrangeiras de forma já compatível com um SGBD **relacional**.
- Ajustar nomenclaturas e padronizações, garantindo consistência entre DER e o lógico.
- Preparar o ambiente de banco de dados, com escolha do SGBD e estudo dos comandos básicos de DDL (CREATE, ALTER, DROP).

3) Dicionário de Dados (padrão da disciplina)

- Entregáveis:
  - **Documento** (em docx) do Dicionário de Dados seguindo **exatamente o padrão fornecido em sala** (estrutura de colunas, terminologia, nível de detalhamento e ordem das tabelas).

Modelo:

NOMDE DA ENTIDADE					
Atributo	**Tipo**	**Tamanho**	**Nulo**	**Chave**	**Descrição**

Critérios de avaliação

- **Completude conceitual**: o modelo cobre os principais objetos e regras do domínio?
- **Correção semântica**: cardinalidades, participações e identificadores fazem sentido?
- **Consistência formal**: nomes padronizados, notação do brModelo adequada, ausência de ambiguidade.
- **Qualidade do Dicionário**: aderência ao padrão da disciplina; definições claras, domínios bem especificados; coerência com o DER.
- **Clareza de entrega**: arquivos nomeados e organizados conforme instruções.

Formato e organização da entrega

- **Padrão de nomes** (exemplo; ajustar ao padrão definido abaixo):
  3B_ModeloConceitual_NomedoTrabalho.brM3
  3B_DicionarioDados_ NomedoTrabalho.docx

<u>Etapa 2 (4º Bimestre): Entrega em SQLs separados + DOCX explicativo</u>

1) Estrutura de pastas e nomes de arquivos

/4B_entrega/

/sql/

01_modelo_fisico.sql

02_insercoes_basicas.sql

03_insercoes_casos_teste.sql

10_queries_basicas.sql

11_queries_relatorios.sql

20_triggers.sql

21_procedures.sql

90_verificacoes_pos_execucao.sql

99_limpeza_reset.sql **(opcional)**

/doc/

4B_Guia_Execucao_SobrenomeNome.docx

Convenções de nomes

- Prefixo **numérico** indica **ordem de execução** (01, 02, 03, 10, 11, 20…).
- Identificadores em **snake_case** e no **singular** quando fizer sentido.
- Todos os SQLs começam com **cabeçalho padrão**.
-

2) Cabeçalho padrão para todos os SQLs (copiar/colar)

/* ==

Arquivo: 01_modelo_fisico.sql

Autor: Sobrenome, Nome

Curso/Turma: _____

SGBD: (PostgreSQL/MySQL) Versão: ____

Objetivo: Criação do modelo físico (DDL)

Execução esperada: rodar primeiro, em BD vazio

== */

3) Descrição dos arquivos SQL

01_modelo_fisico.sql

- **Conteúdo**: CREATE TABLE, PK, FK, NOT NULL, UNIQUE, índices essenciais.
- **Requisitos**:
  - Incluir IF EXISTS **ou** orientar no DOCX como criar o BD.

02_insercoes_basicas.sql

- **Conteúdo**: dados mínimos coerentes para popular **todas** as tabelas principais.
- **Requisitos**:
  - Pelo menos **5 registros** nas tabelas principais.
  - Comentário em cada bloco INSERT explicando **para que** servem esses dados.

03_insercoes_casos_teste.sql

- **Conteúdo**: dados específicos para **exercitar regras**.
- **Requisitos**:
  - Separar por **cenário** (ex.: -- Cenário A, -- Cenário B).
  - Explicar **o que será testado** depois com esses dados (em comentários e no DOCX).

10_queries_basicas.sql

- **Conteúdo mínimo**:
  1. **Listagem simples**;
  2. **JOIN** entre 2+ tabelas;
  3. **WHERE** com filtros;
  4. **Agregações** com GROUP BY;
  5. **Consulta ordenada** com ORDER BY.
- **Requisitos**: cada query inicia com cabeçalho **definido anteriormente**

11_queries_relatorios.sql

- **Conteúdo**: consultas "de relatório"
- **Requisitos**: para cada relatório, explicar **qual problema responde** e **porque** a abordagem foi escolhida (no cabeçalho).

20_triggers.sql

- **Conteúdo**: ao menos 2 **triggers** que valide/automatize regra(s) de negócio.
- **Requisitos**:
  - Comentar **quando dispara** (evento), **tabela** e **o que faz**.
  - Ao final, incluir **DML de teste** (INSERT/UPDATE) e uma SELECT que mostre o efeito.

21_procedures.sql (opcional)

- **Conteúdo**: ao men os 2 **procedures/functions** úteis.
- **Requisitos**: comentar **motivação** e **parâmetros**.

90_verificacoes_pos_execucao.sql

- **Conteúdo**: checagens rápidas para validar a base pós-execução.
- **Exemplos**:
  - SELECT COUNT(*) por tabela;
  - maiores/menores valores em colunas críticas;
  - *joins* para verificar integridade lógica.
- **Requisitos**: cada verificação com comentário **"O que comprova"**.

99_limpeza_reset.sql (opcional)

- **Conteúdo**: TRUNCATE/DELETE/DROP para reiniciar testes.
- **Requisitos**: avisar claramente que **apaga dados e quais dados**.

4) Documento — 4B_Guia_Execucao_NomedoTrabalho.docx

Conteúdo:

1. **Resumo do projeto** (2–3 parágrafos).
2. **Pré-requisitos** (SGBD/cliente SQL/usuário/BD).
3. **Ordem de execução (passo a passo)**
    - 01 → 02 → 03 → 20 → 10 → 11 → 90 (→ 99 opcional).
    - Para cada passo: **o que fazer** e **por que** essa ordem.
4. **Dados de teste**
    - Tabela ligando **cenários** (03) às **queries** (10/11) e **triggers** (20) que os exercitam.
5. **Como validar o funcionamento**
    - Resultados esperados das queries (descritos; prints opcionais).
    - Observação do efeito das triggers (antes/depois).
6. **Erros comuns e soluções**
    - Ex.: falha de FK → rodar 01 antes de 02/03; tipos inválidos; permissões etc.

Critérios de avaliação

- **Organização e padrão** (pastas/arquivos conforme instruções).
- **Modelo físico** correto e coerente com o lógico.
- **Cobertura de testes** (inserções básicas + cenários bem justificados).
- **Consultas** claras, úteis e comentadas.
- **Triggers** funcionais com evidências.
- **DOCX** com passo a passo e **porquês** bem explicados.