Relatório de POO

Fase 1

Tomás Silva | a20451
Telmo Silva | a20456
14 de novembro de 2023

Introdução

O projeto ***AUTOPREM*** propõe-se a desenvolver soluções inovadoras para desafios práticos no contexto da gestão de uma oficina/stand automóvel. O objetivo principal é criar um sistema eficiente que possibilite a gestão integrada de veículos, componentes, funcionários e serviços, proporcionando uma experiência melhorada tanto para os profissionais do setor como para os clientes.

A gestão eficaz de uma oficina ou stand automóvel é crucial para otimizar processos, melhorar a eficiência operacional e oferecer serviços de alta qualidade aos clientes. O setor automóvel é dinâmico e complexo, procurando ferramentas especializadas para lidar com as diversas operações, desde a marcação de serviços de manutenção até a venda de veículos.

Este projeto surge da necessidade de fornecer uma solução abrangente que aborde áreas específicas da gestão automóvel. A escolha estratégica de concentrar esforços na implementação de um sistema baseado em C# visa consolidar os conceitos do Paradigma Orientado a Objetos e potencializar a experiência no desenvolvimento de software.

Ao abordar problemas reais do setor automóvel, como a marcação de serviços, a venda de veículos e a gestão de componentes, esperamos contribuir significativamente para a formação prática dos desenvolvedores envolvidos no projeto. A compreensão que se aprofundou do funcionamento interno de uma oficina/stand automóvel será fundamental para o sucesso deste empreendimento, capacitando os envolvidos a aplicar conhecimentos teóricos na resolução de desafios do mundo real.

Neste contexto, a fase inicial do projeto concentra-se na identificação e implementação essencial das classes fundamentais, estabelecendo as bases para um sistema robusto e funcional. O presente relatório visa documentar o progresso até a data estipulada, destacando as classes identificadas, a implementação realizada e as estruturas de dados utilizadas, cumprindo rigorosamente os prazos estabelecidos.

Com este projeto, desejamos não apenas cumprir os objetivos académicos, mas também proporcionar uma solução prática e valiosa para os desafios enfrentados na gestão de oficinas e stands automóveis.

Objetivos da Fase 1

A Fase 1 do projeto ***AUTOPREM*** foi delineada com metas específicas que visam estabelecer as bases essenciais para a implementação bem-sucedida do sistema de gestão de uma oficina/stand automóvel. Os objetivos estabelecidos para esta fase foram os seguintes:

Identificação de Classes:

Objetivo: Identificar as classes fundamentais necessárias para o funcionamento do sistema.

Abordagem: Analisamos as regras de negócio propostas, destacando entidades-chave, como Veículo, Componente, Funcionário e ServiçoManutencao. Cada classe foi concebida para representar uma entidade distinta e desempenhar um papel específico no contexto da gestão automóvel.

Implementação Essencial das Classes:

Objetivo: Desenvolver a estrutura básica das classes identificadas.

Abordagem: Cada classe foi implementada com as propriedades essenciais que a caracterizam. Métodos foram adicionados para permitir a adição e listagem de instâncias, estabelecendo as funcionalidades essenciais necessárias para a manipulação dos dados.

Estruturas de Dados:

Objetivo: Escolher estruturas de dados adequadas para armazenar informações relevantes.

Abordagem: Optamos por utilizar listas (devíamos armazenar em arrays) para armazenar instâncias de Veículo, Componente, Funcionário e ServiçoManutencao. A escolha dessas estruturas foi baseada na flexibilidade oferecida para manipulação dinâmica de dados.

Relatório do Trabalho Desenvolvido:

Objetivo: Documentar o progresso até à data estipulada.

Abordagem: Elaboramos uma documentação clara e concisa, destacando as principais decisões as classes implementadas, as estruturas de dados

utilizadas e o design. O relatório fornece uma visão geral do trabalho desenvolvido, cumprindo os requisitos estabelecidos.

Cumprimento dos Prazos:

Objetivo: Assegurar o cumprimento rigoroso dos prazos estabelecidos.

Abordagem: Controlamos e gerimos o progresso do projeto de maneira a garantir que cada objetivo fosse alcançado dentro do cronograma estipulado. Este compromisso com os prazos é crucial para o sucesso contínuo do projeto.

A abordagem meticulosa em relação a cada objetivo permitiu-nos estabelecer uma base sólida para o desenvolvimento subsequente do projeto ***AUTOPREM***. Cada classe e estrutura de dados implementada representa um passo significativo na direção de uma solução completa e funcional para os desafios apresentados pela gestão automotiva.

Classes Identificadas

1. Veiculo

Propósito: Representar um veículo na oficina/stand automóvel.

Atributos:

- ID: Identificador único do veículo.
- Marca: Marca do veículo.
- Modelo: Modelo do veículo.
- Preco: Preço do veículo.

2. Componente

Propósito: Representar um componente disponível na oficina/stand automóvel.

Atributos:

- ID: Identificador único do componente.
- Nome: Nome do componente.
- Preco: Preço do componente.

3. Funcionario

Propósito: Representar um funcionário da oficina/stand automóvel.

Atributos:

- ID: Identificador único do funcionário.
- Nome: Nome do funcionário.
- Cargo: Cargo ocupado pelo funcionário.

4. ServicoManutencao

Propósito: Representar um serviço de manutenção marcado na oficina.

Atributos:

- ID: Identificador único do serviço.
- DataAgendamento: Data em que o serviço está marcado.
- Cliente: Nome do cliente para o serviço.
- Veiculo: Veículo associado ao serviço.
- Custo: Custo do serviço de manutenção.

Cada classe foi cuidadosamente identificada para desempenhar um papel específico na gestão automotiva, abrangendo desde a representação de entidades básicas, como veículos e componentes, até a gestão de funcionários e serviços de manutenção. Essas classes formam a espinha dorsal do sistema, permitindo uma abordagem modular e eficaz para a gestão integral de uma oficina ou stand automóvel.

Implementação Essencial das Classes

1. Veiculo

Características Principais:

- Propriedades: ID, Marca, Modelo, Preco.
- Construtor: Aceita os atributos essenciais para criar uma instância.
- Método AdicionarVeiculo: Adiciona um novo veículo à lista, evitando duplicatas com base no ID.
- Método ListarVeiculos: Exibe informações de todos os veículos na lista.

```
public class Veiculo
{
    // ... (propriedades e construtor)

    public static void AdicionarVeiculo(List<Veiculo> veiculos, Veiculo novo
    {
        // Lógica para adicionar um novo veículo
    }

    public static void ListarVeiculos(List<Veiculo> veiculos)
    {
        // Lógica para listar todos os veículos
    }
}
```

Figura 1 - Características Principais - Veiculo

2. Componente

Características Principais:

- Propriedades: ID, Nome, Preco.
- Construtor: Aceita os atributos essenciais para criar uma instância.
- Método AdicionarComponente: Adiciona um novo componente à lista, evitando duplicatas com base no ID.
- Método ListarComponentes: Exibe informações de todos os componentes na lista.

```
public class Componente
{
    // ... (propriedades e construtor)

    public static void AdicionarComponente(List<Componente> componentes, Cor
    {
        // Lógica para adicionar um novo componente
    }

    public static void ListarComponentes(List<Componente> componentes)
    {
        // Lógica para listar todos os componentes
    }
}
```

Figura 2 - Características Principais - Componente

3. Funcionario

- Características Principais:
- Propriedades: ID, Nome, Cargo.
- Construtor: Aceita os atributos essenciais para criar uma instância.
- Método AdicionarFuncionario: Adiciona um novo funcionário à lista, evitando duplicatas com base no ID.
- Método ListarFuncionarios: Exibe informações de todos os funcionários na lista.

```
public class Funcionario
{
    // ... (propriedades e construtor)

    public static void AdicionarFuncionario(List<Funcionario> funcionarios,
    {
        // Lógica para adicionar um novo funcionário
    }

    public static void ListarFuncionarios(List<Funcionario> funcionarios)
    {
        // Lógica para listar todos os funcionários
    }
}
```

Figura 3 - Características Principais - Funcionario

4. ServicoManutencao

- Características Principais:
- Propriedades: ID, DataAgendamento, Cliente, Veiculo, Custo.
- Construtor: Aceita os atributos essenciais para criar uma instância.
- Método AgendarServico: Agenda um novo serviço de manutenção.
- Método ListarServicos: Exibe informações de todos os serviços marcados.

```
public class ServicoManutencao
{
    // ... (propriedades e construtor)

    public static void AgendarServico(List<ServicoManutencao> servicos, Serv
    {
        // Lógica para agendar um novo serviço
    }

    public static void ListarServicos(List<ServicoManutencao> servicos)
    {
        // Lógica para listar todos os serviços
    }
}
```

Figura 4 - Características Principais - ServicoManutencao

Cada classe foi implementada com métodos que facilitam a adição e a listagem de instâncias, fornecendo assim uma interface clara para a manipulação dos dados associados a veículos, componentes, funcionários e serviços de manutenção. O design modular permite uma fácil expansão e manutenção do sistema.

Estruturas de Dados Utilizadas

1. Lista para Veículos
   - **Descrição**: Utilizamos uma lista (List<Veiculo>) para armazenar instâncias da classe Veiculo.
   - **Justificação**: Listas são estruturas de dados dinâmicas que oferecem facilidade na adição e remoção de elementos. A escolha de uma lista é apropriada para a gestão de veículos, pois permite uma manipulação eficiente dos dados, suportando operações como adição, remoção e listagem.

2. Lista para Componentes
   - **Descrição**: Optamos por utilizar uma lista (List<Componente>) para armazenar instâncias da classe Componente.
   - **Justificação**: Da mesma forma que para os veículos, uma lista é adequada para a gestão de componentes. A flexibilidade proporcionada por listas é crucial para lidar com um conjunto variável de componentes disponíveis.

3. Lista para Funcionários
   - **Descrição**: Escolhemos uma lista (List<Funcionario>) para armazenar instâncias da classe Funcionario.
   - **Justificação**: A utilização de uma lista para funcionários permite um gerenciamento eficiente da equipa, possibilitando adicionar, remover e listar funcionários de forma dinâmica.

4. Lista para Serviços de Manutenção
   - **Descrição**: Utilizamos uma lista (List<ServicoManutencao>) para armazenar instâncias da classe ServicoManutencao.
   - **Justificação**: Listas são apropriadas para gerenciar serviços agendados, oferecendo operações eficientes para a adição, remoção e listagem de serviços. Isso é essencial para uma gestão eficaz dos serviços de manutenção na oficina.

Exemplos de Utilização

A seguir, apresentamos exemplos de como as classes e métodos podem ser utilizados no contexto do projeto ***AUTOPREM***:

- **Criação e Adição de Veículos:**

```
// Criar instância de veículo
Veiculo veiculo1 = new Veiculo(1, "Toyota", "Corolla", 25000);

// Adicionar veículo à lista de veículos
Veiculo.AdicionarVeiculo(veiculos, veiculo1);
```

- **Criação e Adição de Componentes:**

```
// Criar instância de componente
Componente componente1 = new Componente(101, "Bateria", 100);

// Adicionar componente à lista de componentes
Componente.AdicionarComponente(componentes, componente1);
```

- **Criação e Adição de Funcionários:**

```
// Criar instância de funcionário
Funcionario funcionario1 = new Funcionario(1001, "João Silva", "Mecânico");

// Adicionar funcionário à lista de funcionários
Funcionario.AdicionarFuncionario(funcionarios, funcionario1);
```

- **Marcações e Listagem de Serviços de Manutenção:**

```
// Criar instância de serviço de manutenção
ServicoManutencao servico1 = new ServicoManutencao(10001, DateTime.Now.AddDays(1), "Carlos Oliveira", veiculo1, 150);

// Agendar serviço de manutenção
ServicoManutencao.AgendarServico(servicos, servico1);

// Listar todos os serviços agendados
ServicoManutencao.ListarServicos(servicos);
```

Estes exemplos proporcionam uma visão prática de como as classes e métodos podem ser utilizados para gerir veículos, componentes, funcionários e serviços de manutenção na oficina/stand automóvel. As operações de criação, adição e listagem mostram como é feita uma funcionalidade básica do sistema implementado.

Desafios Encontrados

Durante a implementação do projeto ***AUTOPREM***, alguns desafios técnicos e de design foram identificados e abordados de maneira a garantir uma solução robusta e eficiente:

Validação de Dados:

- **Desafio:** Garantir que os dados fornecidos para a criação de instâncias estejam corretos e atendam aos requisitos do sistema.
- **Solução:** Implementação de validações nos construtores das classes, utilizando estruturas condicionais para verificar a integridade dos dados antes de criar uma instância. Mensagens de erro são exibidas em caso de dados inválidos.

Lógica de Adição de Elementos nas Listas:

- **Desafio:** Assegurar que a adição de veículos, componentes, funcionários e serviços seja feita de maneira eficiente e evitando a inserção de dados duplicados.
- **Solução:** Utilização de métodos específicos para adicionar elementos às listas, incorporando lógicas que verificam a existência prévia de um item antes de adicionar. Isso evita a inserção de dados duplicados.

Formato de Exibição de Preços:

- **Desafio:** Apresentar os preços de forma clara e consistente em diferentes partes do sistema.
- **Solução:** A aplicação da sobrescrita do método **ToString** foi essencial para proporcionar uma representação mais clara e coerente dos objetos nas classes relevantes, assegurando uma formatação adequada dos valores monetários e contribuindo para a consistência na exibição de informações em diferentes partes do sistema.

Esses desafios foram enfrentados com o objetivo de melhorar a qualidade e a usabilidade do sistema, resultando em soluções que contribuem para uma implementação mais robusta e eficaz.

Lições Aprendidas

Durante o desenvolvimento do projeto ***AUTOPREM***, algumas lições valiosas foram aprendidas, contribuindo para uma compreensão mais profunda do paradigma orientado a objetos e do uso prático da linguagem C#. Algumas considerações importantes incluem:

Abstração e Encapsulamento:

- A importância de criar abstrações eficazes para representar entidades do mundo real, utilizando o encapsulamento para proteger a integridade dos dados e esconder detalhes de implementação.

Sobrescrita de Métodos:

- A sobrescrita do método ToString mostrou-se uma ferramenta poderosa para personalizar a representação textual de objetos, melhorando a legibilidade e a consistência ao exibir informações em diferentes contextos.

Validação de Dados:

- A necessidade de implementar verificações de validação de dados nos construtores das classes, garantindo que as instâncias sejam criadas apenas com informações válidas, contribuindo para a robustez do sistema.

Eficiência na Manipulação de Listas:

- A eficiência na manipulação de listas foi alcançada ao incorporar métodos específicos para adição de elementos, evitando a inserção de dados duplicados e promovendo uma gestão eficaz dos registos.

Considerações para Melhorias Futuras:

- Melhoramento da Interface do Utilizador:
  - Considerar a implementação de uma interface de utilizador mais interativa, seja por meio de uma aplicação de console mais

elaborada ou a transição para tecnologias de interface gráfica, proporcionando uma experiência mais amigável.

Implementação de Persistência de Dados:

- Explorar técnicas de persistência de dados, como o armazenamento em ficheiros ou a integração com um base de dados, para garantir a retenção de informações entre execuções do programa.

Ampliação das Funcionalidades:

- Avaliar a inclusão de funcionalidades adicionais, como a capacidade de atualizar ou excluir registos, proporcionando um conjunto mais completo de operações para os utilizadores.

Essas lições e considerações fornecem uma base sólida para o melhoramento contínuo e expansão do projeto, refletindo um comprometimento com a excelência no desenvolvimento de software orientado a objetos em C#.

Próximos Passos

Considerando os avanços na fase atual do desenvolvimento, os próximos passos para a evolução do projeto ***AUTOPREM*** incluem:

Implementação de Atualizações e Exclusões:

- Adicionar funcionalidades que permitam a atualização e exclusão de registos existentes, proporcionando uma gestão mais completa e flexível dos dados.

Integração de Persistência de Dados:

- Explorar e implementar métodos de persistência de dados, como o armazenamento em ficheiros ou a conexão com a base de dados, para garantir a preservação das informações entre diferentes sessões do programa.

Desenvolvimento da Interface do Utilizador:

- Avançar na criação de uma interface de utilizador mais robusta, seja por meio de melhoramentos na aplicação de console ou a transição para tecnologias gráficas (Windows Forms), visando proporcionar uma experiência mais intuitiva aos utilizadores.

Inclusão de Relatórios Detalhados:

- Implementar recursos de geração de relatórios mais detalhados, oferecendo aos utilizadores informações mais abrangentes sobre veículos, componentes, funcionários e serviços agendados.

Testes e Ajustes:

- Realizar testes abrangentes para identificar e corrigir possíveis problemas ou melhorias de desempenho, assegurando a estabilidade e a eficiência do sistema.

Esses próximos passos visam não apenas consolidar as funcionalidades existentes, mas também expandir a aplicação para atender às procuras mais

avançadas e complexas de uma gestão eficaz de oficinas e stands automóveis. O compromisso contínuo com a qualidade e a inovação será a chave para o sucesso futuro do projeto ***AUTOPREM***.

Conclusão

A conclusão da Fase 1 do projeto ***AUTOPREM*** marca um progresso significativo na implementação de uma solução de gestão eficiente para oficinas e stands automóveis. Ao longo deste período, identificamos, desenvolvemos e implementamos as classes essenciais necessárias para a estruturação do sistema. A criação de Veículos, Componentes, Funcionários e Serviços de Manutenção já está funcional, proporcionando uma base sólida para as fases subsequentes.

O destaque desta fase recai sobre a capacidade de criar instâncias, adicionar elementos e listar informações através de métodos específicos. A utilização da sobrescrita do método ToString contribuiu para uma exibição mais clara e legível das informações, facilitando a compreensão e utilização do sistema.

Enfrentamos alguns desafios durante a implementação, especialmente ao lidar com a gestão de listas e a validação de entradas. No entanto, cada desafio representou uma oportunidade valiosa para aprendizado e crescimento, resultando em soluções mais robustas e eficazes.

O próximo passo envolverá a expansão das funcionalidades, incluindo a implementação de atualizações e exclusões, a integração de persistência de dados, o desenvolvimento de interfaces mais avançadas e a inclusão de relatórios detalhados. Essas melhorias visam criar um sistema mais completo e prontamente utilizável.

Em suma, a Fase 1 foi marcada por avanços significativos, consolidando conhecimentos fundamentais de programação orientada a objetos e proporcionando uma base sólida para o desenvolvimento futuro. Estamos ansiosos para os desafios e conquistas que as próximas fases do projeto ***AUTOPREM*** trarão.

Cumprimento dos Prazos

A gestão do tempo durante a Fase 1 do projeto ***AUTOPREM*** foi fundamental para o sucesso das entregas. Todos os prazos estabelecidos foram cumpridos de acordo com o planeamento inicial. Isso reflete a dedicação e comprometimento da equipa em atender às expectativas e garantir o avanço consistente do projeto.

O uso eficiente do tempo permitiu a identificação e implementação essencial das classes, a escolha e utilização adequada de estruturas de dados, além da criação de métodos fundamentais para a funcionalidade do sistema. O resultado foi um conjunto de funcionalidades funcionais que atendem aos requisitos da Fase 1.

As lições aprendidas durante esta etapa forneceram *insights* valiosos sobre a gestão do tempo. A necessidade de equilibrar a eficiência na implementação com a atenção aos detalhes foi evidente. A comunicação eficaz entre os membros da equipa desempenhou um papel crucial no cumprimento dos prazos, permitindo uma distribuição equitativamente das tarefas e a resolução rápida de possíveis obstáculos.

Olhando para o futuro, reconhecemos a importância contínua da gestão de tempo para o sucesso contínuo do projeto. Identificar áreas de melhoria, como a otimização de processos e a implementação de técnicas de gestão de tempo mais avançadas, será essencial para garantir que os prazos sejam não apenas cumpridos, mas superados nas fases subsequentes.

O cumprimento dos prazos na Fase 1 revela a eficácia do planeamento e a execução habilidosa das tarefas. Estamos confiantes de que esse mesmo comprometimento será mantido nas próximas etapas do desenvolvimento do ***AUTOPREM***.