**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ**

PROGRAMAÇÃO ORIENTADA A OBJETOS

Projeto RA3 - Autocare

[[1]](#footnote-1)Eduardo Lourenço da Silva

[[2]](#footnote-2)Fernando Padilha Jusviak

[[3]](#footnote-3)João Pedro Novak Rausis

[[4]](#footnote-4)Vinicius Veiga De Sant’ana

[[5]](#footnote-5)Valdinei José Saugo

**Classe 1: Usuario**

**Aluno responsável:** Eduardo Lourenço da Silva

**Descrição:** A classe Usuario representa os donos de veículos que se cadastram no sistema AutoCare. Ela estende da classe Model, herdando seus métodos e funcionalidades, além de implementar métodos adicionais específicos. Como Model define métodos abstratos, a classe Usuario é responsável por implementar esses comportamentos obrigatórios. A classe também implementa a interface Serializable, necessária para o funcionamento da leitura e gravação dos objetos nos arquivos.

**Diagrama UML:**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**Análise:** O desenvolvimento da classe Usuario, e das demais classes que fazem o sistema funcionar (Applicattion, LayoutController, NavbarController) ocorreu de forma tranquila, porém foi necessário buscar conhecimento do funcionamento dos layouts do javafx, principalmente para determinar a estrutura que seria seguida para a mudança das classes, o layout escolhido foi o BorderPane. A utilização de inteligência artificial (IA) foi apenas como apoio, não foi utilizado código extraído diretamente das IA (ChatGPT).

**Classe 2: Prestador**

**Aluno responsável:** Fernando Padilha Jusviak

**Descrição:** A classe Prestador representa as oficinas e estéticas automotivas que se cadastram no sistema AutoCare. Assim como a classe Usuário ela estende da classe Model, herdando seus métodos e funcionalidades, além de implementar métodos adicionais específicos. Como Model define métodos abstratos, a classe Prestador é responsável por implementar esses comportamentos obrigatórios. A classe também implementa a interface Serializable, necessária para o funcionamento da leitura e gravação dos objetos nos arquivos.

**Diagrama UML:**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**Análise**: A classe que escolhi para realizar foi a de Prestador, sendo ela a mesma que desenvolvida no projeto de experiencia criativa, a lógica da classe segue a mesma de todo o projeto.Possuindo um arquivo de Controller, Model, Área de aceeso pela NavBar, Tabela e Forms. Uma das dificuldades foi lembrar de utilizar o mesmo nome de alguns atributos, em algumas áreas onde parecia não ser necessário.

**Classe 3: FuncionarioMapa**

**Aluno responsável:** João Pedro Novak Rausis

**Descrição:** A classe FuncionarioMapa representa os funcionários e, também, a localização desses funcionários dentro do sistema AutoCare. Assim como as demais classes ela também estende da classe Model, herdando seus métodos e funcionalidades, além de implementar métodos adicionais específicos. Como Model define métodos abstratos, a classe FuncionarioMapa é responsável por implementar esses comportamentos obrigatórios. A classe também implementa a interface Serializable, necessária para o funcionamento da leitura e gravação dos objetos nos arquivos.

**Diagrama UML:**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**Análise:** Conforme os outros integrantes do grupo, segui o mesmo padrão para o CRUD básico, utilizando uma tabela. Adotei o mesmo raciocínio de usar um módulo e um formulário à esquerda, com a única diferença de que, no meu programa, adicionei um *Dropbox* (combobox) com todos os estados brasileiros. Utilizei o ChatGPT como apoio no projeto, com prompts pedindo ajuda para corrigir falhas no código. Um exemplo foi:  
now why the new form box not show up? (*agora por que o novo formulário não está aparecendo?*)

package br.com.autocare.FuncionarioMapa;

import br.com.autocare.components.FormControl;

import br.com.autocare.model.FuncionarioMapa; (...resto do código)

**Classe 4: Veiculo**

**Aluno responsável:** Vinicius Veiga De Sant’ana

**Descrição:** A classe Veiculo representa os veículos que são cadastrados pelos usuários dentro do sistema AutoCare. Assim como as demais classes ela também estende da classe Model, herdando seus métodos e funcionalidades, além de implementar métodos adicionais específicos. Como Model define métodos abstratos, a classe Veiculo é responsável por implementar esses comportamentos obrigatórios. A classe também implementa a interface Serializable, necessária para o funcionamento da leitura e gravação dos objetos nos arquivos.

**Diagrama UML:**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

**Análise:** A classe desenvolvida por mim foi a classe de Veiculo, a mesma que desenvolvi no projeto de experiencia criativa, a lógica da minha classe segue a mesma de todo o projeto...Tendo um controller, model, o ícone de menu, a tabela e o forms. Uma das dificuldades foi salvar os dados no arquivo desejado para posterior edição, mas após auxilio de meus colegas foi possível. Achei muito interessante o desenvolvimento para evidenciar as diferenças em um desenvolvimento em PHP e em Java.

**Diagrama UML Completo:**

**Diagrama

O conteúdo gerado por IA pode estar incorreto.**

1. Aluno Graduando em Engenharia de software na PUCPR [↑](#footnote-ref-1)
2. Aluno Graduando em Engenharia de software na PUCPR [↑](#footnote-ref-2)
3. Aluno Graduando em Engenharia de software na PUCPR [↑](#footnote-ref-3)
4. Aluno Graduando em Engenharia de software na PUCPR [↑](#footnote-ref-4)
5. Professor na disciplina de Performance em Sistema Ciberfisicos na PUCPR [↑](#footnote-ref-5)