

# **Sistema Java Desktop de Gestão de Ordens de Serviço Automotivo, com uso de MVC e Hibernate**

**Bruno Luiz Reinicke**

Instituto Federal Catarinense - Campus Blumenau

reinickebruno@gmail.com

**Resumo.** *O artigo trata sobre o desenvolvimento de um sistema java desktop para monitorar ordens de serviço automotivo, e foi desenvolvido pelo primeiro autor, com o intuito de obter o grau de tecnólogo em análise e desenvolvimento de sistemas no IFC Campus Blumenau. O texto contextualiza o sistema a ser desenvolvido, apresenta os métodos utilizados para a construção do sistema, menciona trabalhos relacionados e detalha informações sobre o projeto e a implementação do sistema. A programação foi realizada utilizando o padrão MVC, utilizando a linguagem Java, com o uso de Hibernate. O código-fonte está disponível em repositório público.*

## **1. Introdução**

O atual sistema desktop dá oficina do meu pai controla apenas os gastos das compras das peças para carros, então eu resolvi desenvolver um CRUD para controlar o estoque de peças, e controlar também as ordens de serviço prestadas nos carros dos clientes, disponibilizando um acompanhamento online para o cliente acompanhar a tarefa realizada no conserto do seu carro.

Cada cliente poderá acessar e acompanhar o serviço de manutenção realizado apenas no seu carro, somente o usuário administrador poderá acessar os serviços realizados nos carros de todos os clientes. Os acessos para controlar as ordens de serviço também serão concedidos apenas ao administrador.

### **1.1. Tema/Problema**

Para os clientes não é disponibilizado nenhum acompanhamento online dos serviços de manutenção realizados nos carros dos mesmos, então o cliente nunca sabe como está o andamento do serviço de manutenção realizado no seu carro. No atual sistema da oficina, não há como fazer remotamente uma consulta, ele também não possui um controle de estoque de peças, controle este que facilitaria a gestão da oficina.

### **1.2. Objetivos Propostos/Solução dos Problemas**

Desenvolver um sistema java desktop que tenha a funcionalidade de controle de estoque, e que disponibilize um acompanhamento online das ordens de serviço de manutenção dos carros dos clientes, para os clientes acessarem o sistema remotamente e cada cliente acompanhar o serviço de manutenção realizado no seu carro.

### **1.3. Escopo**

O sistema fará uma interação com banco de dados MySQL utilizando Hibernate para interagir com o banco de dados, a implementação fará uso de interfaces gráficas e uso do padrão MVC.

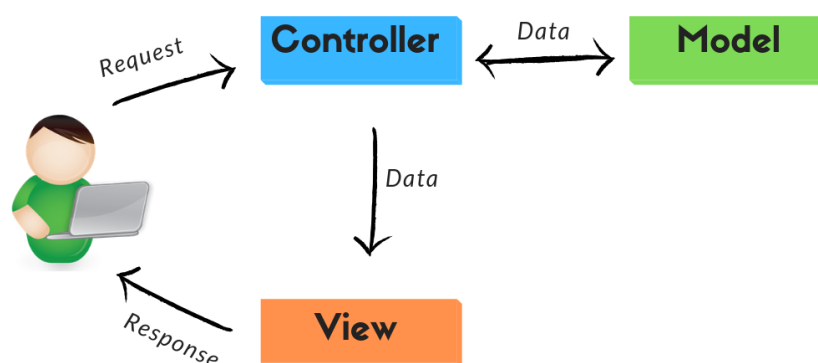
É possível o cliente consultar a ordem de serviço que remete ao conserto do carro que ele pediu para a oficina consertar. Quando o administrador do sistema alterar a ordem de serviço atualizando as informações, o cliente terá as informações atualizadas na consulta online.

#### 1.4. Viabilidade do Projeto

O projeto é viável, pois eu nunca tive muita dificuldade com as tecnologias que serão utilizadas no desenvolvimento desse projeto. A implementação fluiu atendendo o esperado, proporcionando para mim uma vivência mais aprofundada em Hibernate.

#### 1.5. Método de Trabalho (arquitetura, ferramentas, tecnologias aplicadas)

O sistema java desktop de gestão automotiva desenvolvido neste trabalho foi desenvolvido utilizando o padrão de modelo MVC (Figura 1). O padrão MVC serve para implementar as interfaces utilizadas pelo usuário(View), implementar a ponte(Controller) entre as interfaces e as informações que são trazidas do banco para os objetos que estão em memória e a partir desses objetos tratar no controller como as informações chegam até o usuário. O mesmo se aplica ao processo inverso em que o usuário informa em tela as informações e depois as mesmas são gravadas no banco.



**Figura 1. Padrão de modelo MVC. Adaptado de**  
(<https://www.treinaweb.com.br/blog/o-que-e-symfony>)

Neste trabalho a camada View é composta pelas telas de cadastro e consulta. A camada Controller é responsável pelas regras e restrições que o sistema deve seguir e respeitar, é nessa camada que é feita a ponte entre as informações presentes na tela e as classes modelo que são usadas para fazer a persistência dos dados no banco. Para fazer a persistência poucos comandos sql foram utilizados, pois com o uso de Hibernate no MVC, os tratamentos ficaram mais nos objetos usados na persistência. Foram feitos tratamentos utilizando um framework específico que abstrai para o programador a interação com as informações do banco de dados.

## **2. Trabalhos Correlatos Existentes**

Existem na internet e no Instituto Federal Catarinense Campus Blumenau, trabalhos que seguem o padrão exigido pelo curso, com isso eu pude consultar o TCC de Desenvolvimento de Aplicação Web para E-Commerce de Vestuário, feito pelo meu colega de sala Jonas Stasiak, e para o desenvolvimento deste trabalho eu também consultei o repositório Github do meu professor Hylson Vescovi Netto.

## **3. Requisitos do sistema**

A seguir encontram-se os requisitos funcionais e não-funcionais do sistema desenvolvido neste trabalho. No total, 10 requisitos funcionais e 8 requisitos não-funcionais fazem parte do projeto.

### **3.1. Requisitos funcionais**

RF001 - Cadastrar clientes.

RF002 - Cadastrar carros.

RF003 - Cadastrar peças.

RF004 - Cadastrar ordens de serviço.

RF005 - Consultar clientes.

RF006 - Consultar carros.

RF007 - Consultar peças.

RF008 - Consultar ordens de serviço.

RF009 - Cadastrar fornecedores de peças.

RF010 - Consultar fornecedores de peças.

### **3.1. Requisitos não-funcionais**

RN001 - Deve ter uma tela de login para determinar quais telas o usuário pode acessar de acordo com o seu privilégio.

RN002 - Apenas o usuário administrador pode fazer cadastros.

RN003 - As senhas devem ficar armazenadas no banco de dados e criptografadas com uma hash.

RN004 - Deve ter uma boa performance na hora de cadastrar e extrair informações.

RN005 - Apenas o usuário administrador pode abrir todas as telas.

RN006 - Os clientes podem consultar apenas as ordens de serviço.

RN007 - Cada cliente pode apenas consultar a ordem de serviço dos seus veículos, e para isso a ordem precisa estar com o status “Em execução”.

RN008 - Apenas o usuário administrador pode consultar todas as ordens serviço do sistema.

## **4. Diagramas UML**

Os diagramas UML basicamente servem para ilustrar o sistema e mostrá-lo de uma forma gráfica, serve para a visualização e documentação dos artefatos do sistema. A seguir são ilustrados os seguintes diagramas: diagrama de classes que serve para representar graficamente as entidades do sistema, diagrama de caso de uso que mostra o papel dos atores do sistema, diagrama de atividade que mostra o fluxo das atividades executadas pelos atores, e diagrama de componentes.

### **4.1 Diagrama de Classes**

O Diagrama de Classes mostra o que há nas classes que compõem o sistema, mostrando os atributos e os métodos das mesmas, mostra também como as entidades se relacionam (Figura 2). O diagrama da aplicação desktop mostra quais entidades o sistema utiliza para fazer a persistência dos dados no banco via Hibernate, representando os modelos de dados manipulados pela camada Controller que seguem a regra de negócio e as restrições do sistema.

A classe “Usuario” serve para inserir no banco e extrair do banco as informações dos usuários cadastrados no sistema, é a classe que serve para permitir ou negar o acesso ao sistema. O usuário administrador é apenas um, o usuário cliente pode ter vários usuários cadastrados. Para ter um carro cadastrado, o cliente necessita que seja primeiro feita a persistência dos seus dados no banco através da classe “Cliente”.

Feita a persistência através da classe “Cliente”, o cliente pode solicitar ao administrador o cadastro do seu carro. Quando o cliente solicita ao administrador a abertura de uma ordem de serviço, o administrador precisa ter cadastrado o fornecedor que fornece as suas peças, o fornecedor não utiliza o sistema mas o cadastro dele é fundamental para a organização, pois por ele sabe-se quem vendeu determinada peça para a oficina.

Com a persistência dos dados do fornecedor concluída, pode ser feita a persistência do cadastro das peças, com as peças necessárias cadastradas, com o carro e o cliente cadastrados, o cliente pode solicitar ao administrador a abertura da ordem de serviço, representada pela classe “OrdemServico”.

Se o cliente possuir mais de um carro cadastrado e ele decidir solicitar a abertura de uma ordem serviço, é feita a abertura de uma ordem de serviço para cada carro, a ordem de serviço, identificada na consulta pelo atributo numero, permite saber quais são os serviços realizados no carro, um outro carro terá um outro número gravado na persistência, mesmo esse carro sendo do mesmo cliente.

Para obter o número do atributo numero, é feita uma consulta que traz o maior valor do campo numero da entidade ordemservico e incrementado o valor 1 no valor

carregado pela consulta. A mesma regra se aplica ao cadastrar uma nova ordem de serviço para o carro de um cliente que já possuir as suas informações persistidas em uma ordem de serviço mais antiga, pois as ordens mais antigas, mesmo remetendo ao mesmo carro e ao mesmo cliente da ordem atual, já foram encerradas. Ao cadastrar uma ordem de serviço de um outro cliente o mesmo processo é realizado.

Se a ordem de serviço estiver em execução, e o cliente solicitar o cadastro de um serviço de troca de peças para o seu carro, ao solicitar mais um serviço de troca de peças na ordem de serviço que estiver atendendo o conserto do seu carro, é incluído mais um serviço atribuindo ao atributo numero o valor do campo numero da entidade ordemservico que estiver armazenando e disponibilizando as informações no momento, ou seja, o serviço é incluído na ordem de serviço que está aberta no momento com o status “Em execução”.

Todos os usuários terão acesso a tela de consulta de ordens de serviço, porém o cliente poderá consultar apenas a ordem de serviço que estiver em execução e que estiver atendendo o conserto do seu carro no momento. Somente o usuário administrador poderá visualizar as ordens de serviço de todos os clientes independente do status ser “Em execução” ou “Encerrada”.

Diagrama de Classes - OSTPA

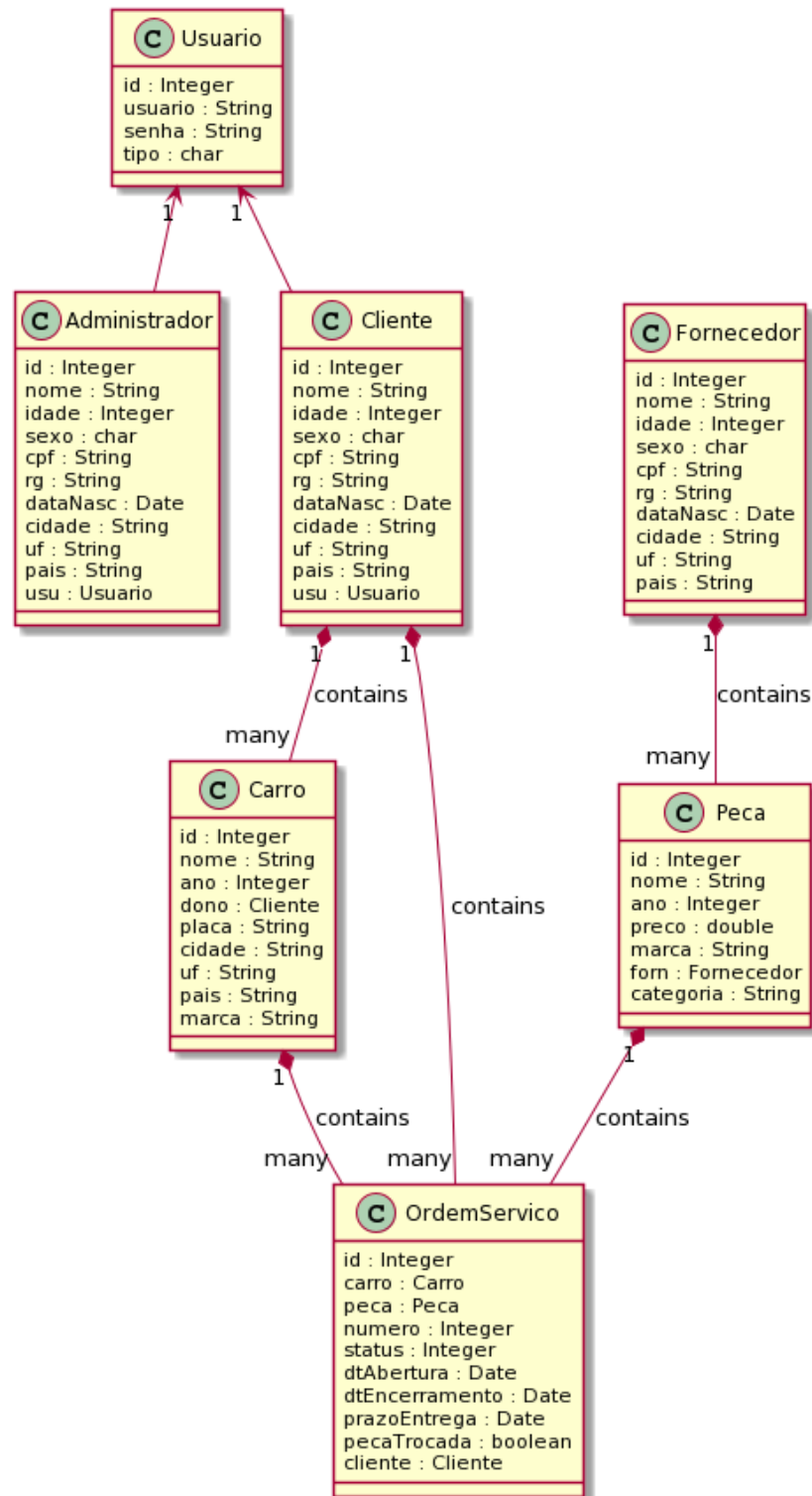
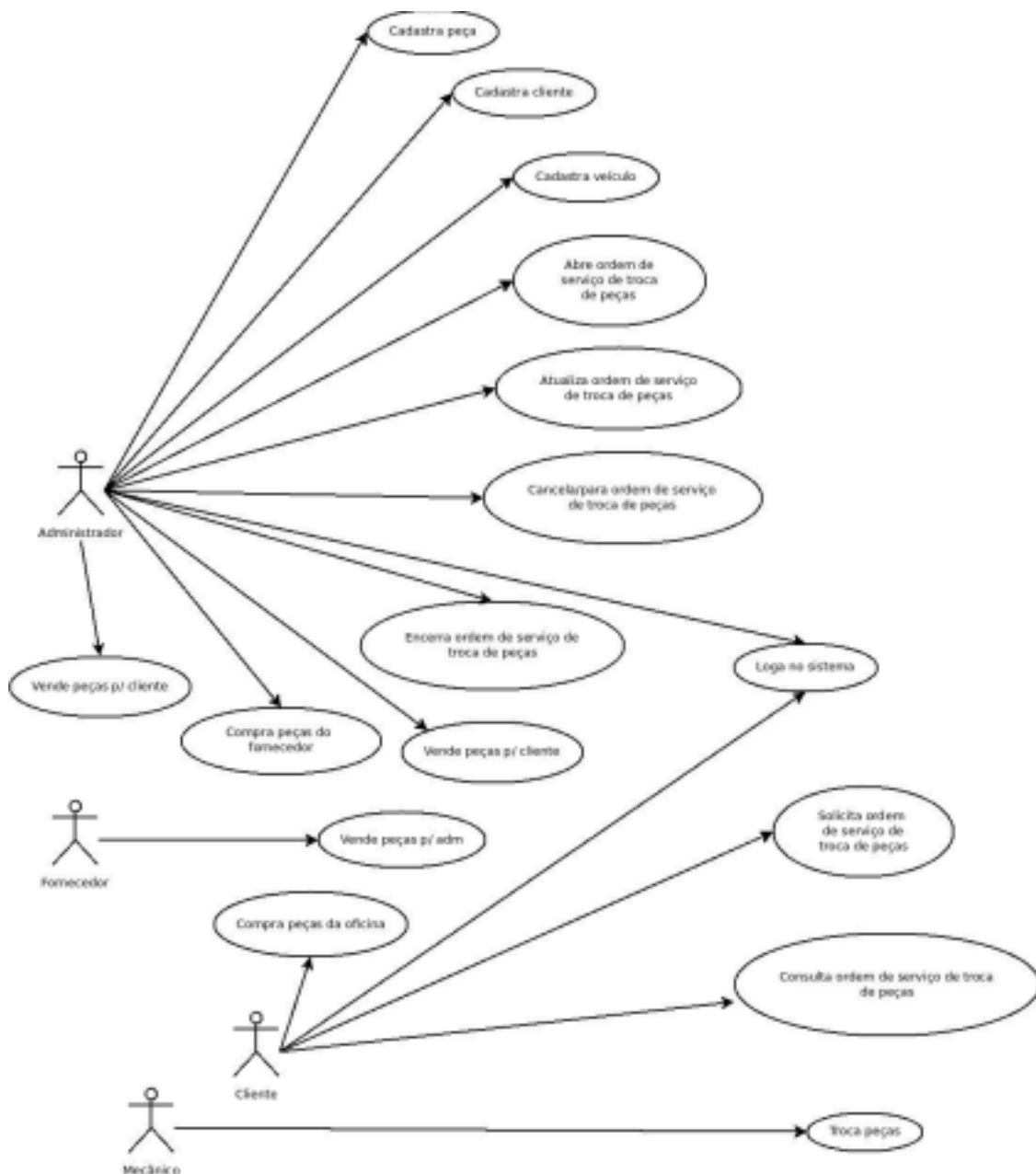


Figura 2. Diagrama de Classes representando as entidades. Fonte: o autor.

#### 4.2 Diagrama de Caso de Uso

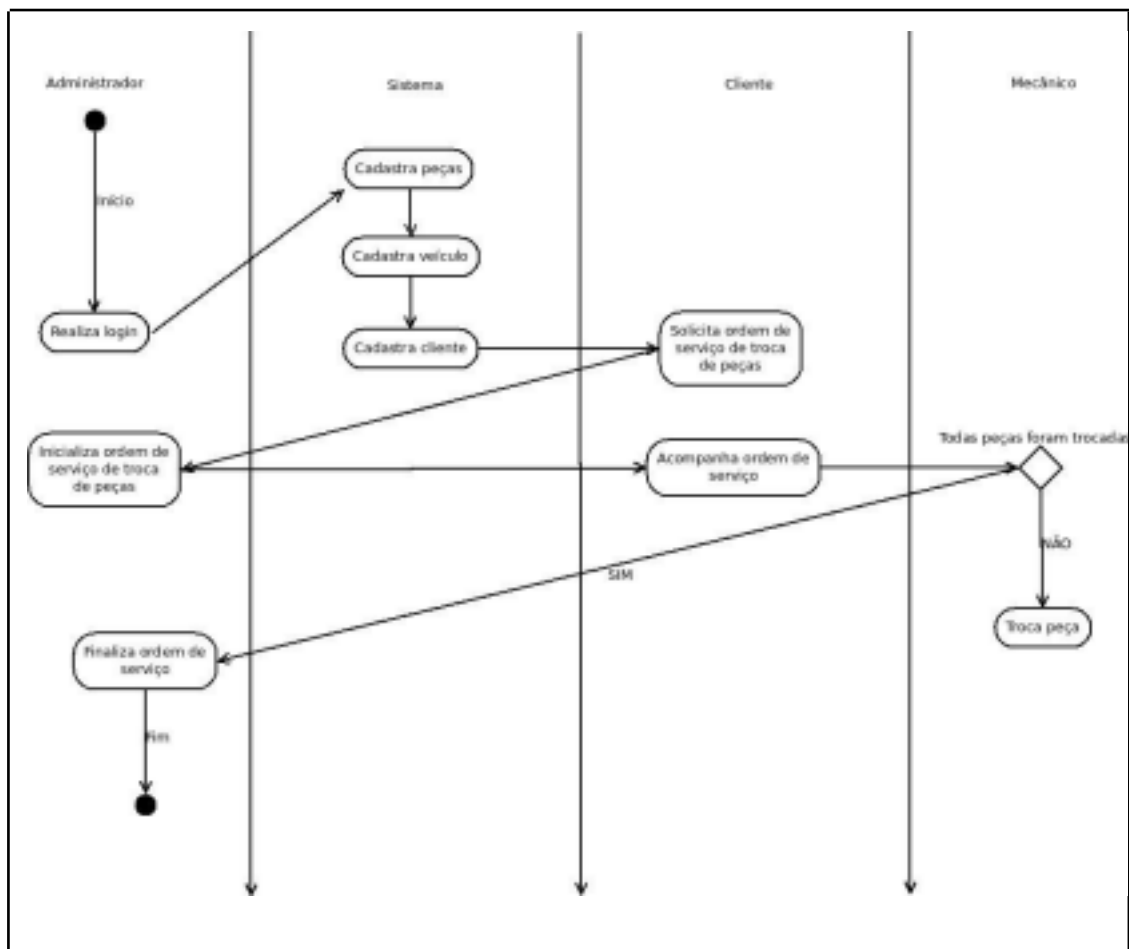
Mostra o papel que cada ator possui na utilização do sistema, os balões representam as atividades desempenhadas pelos atores, ou seja, mostra quais são as atividades exercidas pelos usuários do sistema (Figura 3).



**Figura 3. Diagrama de Caso de Uso representando os papéis desempenhados pelos atores do sistema. Fonte: o autor.**

#### 4.3 Diagrama de Atividade

Mostra o fluxo de atividades do sistema, exibindo quais procedimentos são seguidos pelos atores quando o sistema está em funcionamento. O diagrama mostra quais atividades devem ser exercidas para que possam ser exercidas as atividades seguintes, e assim sucessivamente (Figura 4).



**Figura 4. Diagrama de Atividade representando o fluxo das ações dos atores do sistema. Fonte: o autor.**

## 5 Modelagem de Dados

Para a modelagem de dados, primeiramente foram levantados os requisitos necessários para a implementação das classes java, depois de criadas as classes, as tabelas foram criadas automaticamente via Hibernate. Com as tabelas criadas no banco, eu usei o SGBD para gerar o modelo ER (Entidade Relacionamento). O modelo ER serve para mostrar os campos das entidades do banco e os seus relacionamentos.

### 5.1 Diagrama de Entidades e Relacionamentos

O diagrama de entidades e relacionamentos representa as entidades do sistema no banco de dados (Figura 5). A base de dados é composta por sete (7) entidades, que permitem o cadastro e a extração das informações necessárias para o usuário, mostrando para o cliente o status da ordem de serviço executada para realizar o conserto do seu carro.

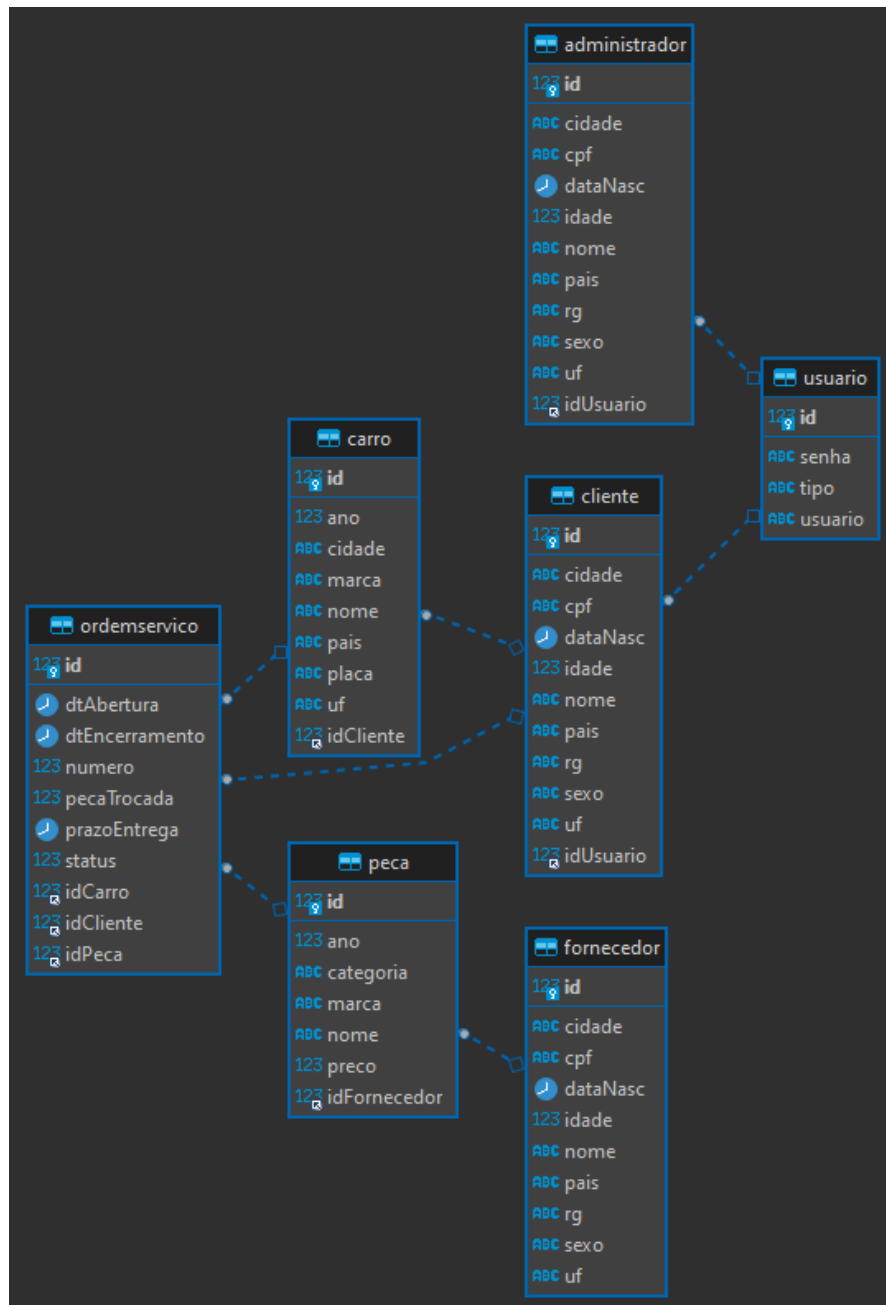
A entidade usuario possui as informações que dão início aos relacionamentos, ela permite a gravação de informações nas entidades cliente e administrador, a entidade cliente é necessária para que fluam os relacionamentos com as outras entidades.

A entidade cliente se relaciona com a entidade carro, podendo um cliente ter vários carros e nunca um carro pertencer a mais de um cliente, essa entidade também se relaciona com a entidade ordemservico, podendo o cliente ter várias ordens de serviço



abertas para ele. Para mais de um carro, o tratamento é feito em ordens de serviço separadas, cujo conceito é uma ordem para cada carro (controlado pelo campo número e não pelo campo id), apesar de o relacionamento carro para ordem de serviço no banco ser 1:N.

A entidade ordemservico se relaciona com a entidade peca, até o momento parte-se da ideia de que a entidade peca serve apenas para armazenar as informações dos exemplares sem levar em consideração a quantidade em estoque, portanto um exemplar gravado na entidade peca pode ser utilizado em mais de uma ordem de serviço, e a entidade peca se relaciona com a entidade fornecedor onde cada peça pode pertencer apenas a um fornecedor.

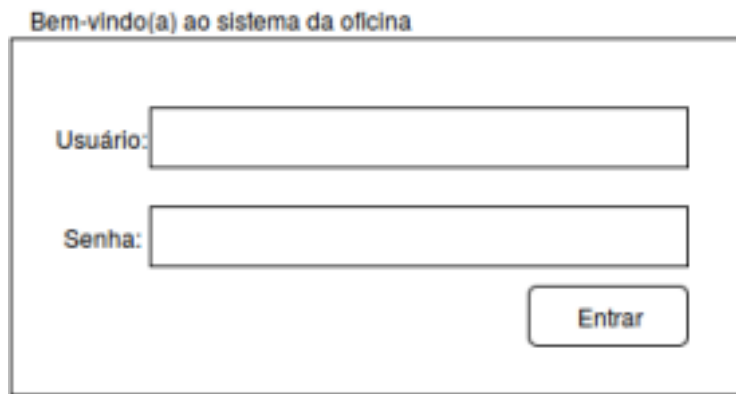


**Figura 5. Diagrama de Entidades e Relacionamentos. Fonte: o autor**

## 6. Projeto da Interface

As telas de cadastro e de consulta foram desenvolvidas com base nas telas dos sistemas desktop que fazem parte da minha vivência, apesar de eu ter trabalhado a maior parte do tempo com Delphi, a parte gráfica que o NetBeans proporciona eu achei semelhante ao que é proporcionado pelo Delphi, então eu usei isso como referência. A prototipação de telas foi feita utilizando uma ferramenta de prototipação encontrada na internet, e as telas do sistema foram criadas utilizando Java Swing.

É necessário clicar no botão Logar para abrir a tela de login, abrindo a tela de login basta informar os dados e clicar para logar (Figura 6).



Bem-vindo(a) ao sistema da oficina

Usuário:

Senha:

**Figura 6. Tela que concede ou nega o acesso ao sistema. Fonte: o autor.**

Ao logar no sistema, os clientes podem consultar a tela de consulta de ordens de serviço, porém o acesso às telas de cadastro e às outras telas de consulta só é concedido se o usuário que estiver logado é administrador, ou seja, os botões para abrir as outras telas só serão habilitados para o usuário administrador (Figura 7).



Peças ▼	Serviço de troca de peças ▼
Comprar...	Consultar...
Solicitar troca...	
Consultar...	

**Figura 7. Botões habilitados na tela principal depois que o administrador acessou. Fonte: o autor.**

Quando o cliente loga no sistema, ele pode consultar as ordens de serviço, porém com menos privilégios que o administrador. Quando eu montei o protótipo desta tela eu ainda considerava que o cliente podia visualizar o cadastro de peças, portanto o menu das peças não deve ser considerado nesta prototipação (Figura 8).



**Figura 8. Mostra o que aparece na tela principal depois que o cliente acessou.**  
**Fonte: o autor.**

Para o administrador, todos os botões devem ficar habilitados depois que ele realizou o acesso, com isso ele pode acessar todas as telas do sistema (Figura 9).

Cadastro ▼	Consulta ▼	Emissão ▼
Cientes...	Cientes...	Pedido de compra de peça...
Veículos...	Veículos...	
Peças...	Peças...	Nota de venda da peça...
Vendas...	Vendas...	
Trocas...	Trocas...	
Compras...	Compras...	
Fornecedores...	Fornecedores...	

**Figura 9. Mostra todos os privilégios do usuário administrador. Fonte: o autor.**

Protótipo da tela de cadastro de clientes, uma das telas de cadastro que o usuário administrador possui acesso (Figura 10).

Código:

Nome:

Idade:

CPF:

Endereço:

Cidade:

UF:

Usuário:

Senha:

**Figura 10. Tela para cadastrar clientes. Fonte: o autor.**

O cadastro dos carros também pode ser feito apenas pelo administrador (Figura 11).

Código:

Marca:

Nome:

Ano:

Cód. cliente:

Cor:

Preço:

Placa:

**Figura 11. Tela para cadastrar carros. Fonte: o autor.**

O cadastro das peças também pode ser feito apenas pelo administrador (Figura 12).

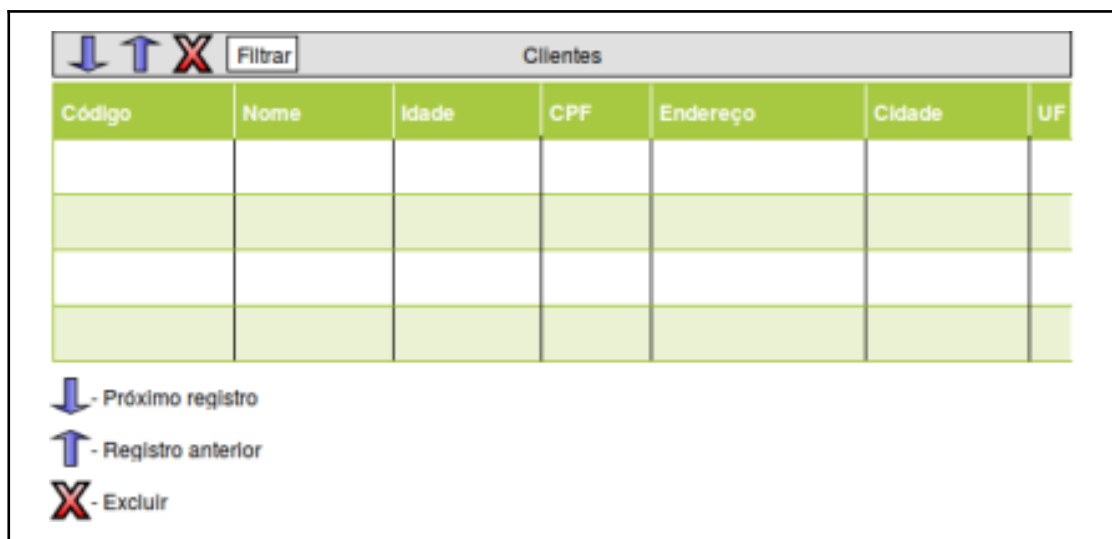
Código:

Código da peça:

Código do fornecedor:

**Figura 12. Cadastro de peças. Fonte: o autor.**

A tela de consulta de clientes também poderá ser acessada apenas pelo administrador (Figura 13).



Código	Nome	Idade	CPF	Endereço	Cidade	UF

↓ - Próximo registro  
↑ - Registro anterior  
X - Excluir

**Figura 13. Consulta de clientes. Fonte: o autor.**

## 7. Implementação

O Código fonte da aplicação pode ser encontrado em um repositório público do Github disponibilizado na internet (<https://github.com/BrunoReinicke/GestaoOficina>), mostrando os commits feitos durante o projeto. O sistema é um sistema desktop escrito na linguagem Java, utilizando o NetBeans para a criação de interfaces gráficas e implementação do padrão MVC, e fazendo uso do framework Hibernate para interagir com o banco de dados MySQL.

## 8. Resultados Obtidos

Implementei a persistência de dados utilizando Hibernate para testar o cadastro e a consulta de usuários, através de telas criadas em java utilizando o padrão MVC no NetBeans. Fiz alguns testes e estão funcionando as telas de cadastro e consulta, que utilizando Hibernate, interagem com o banco de dados MySQL para gravar e extrair informações.

## 9. Considerações Finais e Trabalhos Futuros

Acredito que o uso de Java e Hibernate neste trabalho fará eu praticar programação Java de uma forma que este estudo seja produtivo, pois eu vou rever tecnologias que eu não tive muita dificuldade para me adaptar durante o curso, e irei praticar ainda mais uma linguagem de programação que eu gosto e que também é diferente da linguagem que eu utilizo para programar no meu emprego.

## 10. Referências Bibliográficas

GOMES, Janyenne L. S. **Definição e classificação dos requisitos: Levantamento e análise de requisitos.** Slideshare. Governador Valadares, 2015. 33 p. Disponível em: <https://www.slideshare.net/devnetgomez/definio-e-classificacao-dos-requisitos>. Acesso em: 12 jan. 2021.

VENTURA, Plínio. **Caso de Uso – Include, Extend e Generalização: Exemplificando.** Até o Momento. 2014. 1 p. Disponível em: <https://www.ateomomento.com.br/caso-de-uso-include-extend-e-generalizacao/>. Acesso em: 12 jan. 2021.

VENTURA, Plínio. **Entendendo o Diagrama de Atividades da UML: Exemplo de Uso.** Até o Momento. 2016. 1 p. Disponível em: <https://www.ateomomento.com.br/uml-diagrama-de-atividades/>. Acesso em: 12 jan. 2021.

GEMO, dr. Edgar; SAUGENE (MSC.), Dr. Zeferino. **Diagramas de componentes.** Slideshare. 2012. 12 p. Disponível em: [https://pt.slideshare.net/Portal\\_do\\_estudante\\_ADS/diagramas-de-componentes](https://pt.slideshare.net/Portal_do_estudante_ADS/diagramas-de-componentes). Acesso em: 12 jan. 2021.

Rodrigo. **Introdução à Engenharia de Requisitos: Engenharia de Software e Requisitos.** DevMedia. 2008. 1 p. Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-engenharia-de-requisitos/8034>. Acesso em: 12 jan. 2021.

Paulo. **Artigo SQL Magazine 68 - Utilizando UML: Diagramas de Implantação, Comunicação e Tempo** Artigo SQL Magazine 68 - Utilizando UML: Diagramas de Implantação, Comunicação e Tempo: O Diagrama de Implantação. DevMedia. 2010. 1 p. Disponível em: <https://www.devmedia.com.br/artigo-sql-magazine-68-utilizando-uml-diagramas-de-implantacao-comunicacao-e-tempoartigo-sql-magazine-68-utilizando-uml-diagramas-de-implantacao-comunicacao-e-tempo/16353>. Acesso em: 12 jan. 2021.

Leandro. **O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML: Diagrama de Casos de Uso.** DevMedia. 2012. 1 p. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 12 jan. 2021.

Projeto MPOO - Sistema para hospital: **Diagrama de casos de uso.** Google Sites. 1 p. Disponível em: <https://sites.google.com/site/hospitalprojetompoo/especificacoes-tecnicas/casos-de-uso>. Acesso em: 12 jan. 2021.

Douglas. **Orientações básicas na elaboração de um diagrama de classes.** DevMedia. 2016. 1 p. Disponível em: <https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224>. Acesso em: 12 jan. 2021.