Universidade Norte do Paraná

SISTEMA DE ENSINO PRESENCIAL CONECTADO
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

OfiSys V.0.5:

Um ensaio de sistema de gerenciamento serviços e processos de funcionamento de uma oficina automotiva

Caicó
2018

SAINT-CLAIR DA CUNHA LIMA

OfiSys V.0.5:

Um ensaio de sistema de gerenciamento serviços e processos de funcionamento de uma oficina automotiva

Trabalho em grupo apresentado à Universidade Norte do Paraná - UNOPAR, como requisito parcial para a obtenção de média semestral nas seguintes disciplinas: Engenharia e Projeto de Software, Projeto Orientado a Objetos, Programação Para Web II e Seminários V.

Orientadores: Profº. Marco Ikuro Hisatomi
Profº. Paulo Henrique Terra
Profº. Anderson E. M. Gonçalves
Profº. Roberto Yukio Nishimura

Caicó
2018

SUMÁRIO

ÍNDICE DE FIGURAS

1. INTRODUÇÃO

Trata-se, esse trabalho, da produção escrita referente ao portfólio do quarto semestre do curso de Análise e Desenvolvimento de Sistemas, contemplando as disciplinas de Engenharia e Projeto de Software, Projeto Orientado a Objetos, Programação para Web II e Seminários V. A proposta é desenvolver um sistema de gerenciamento de dados e processos para um centro automotivo chamado Oficina Chave de Rodas.

Nomeou-se o sistema de OfiSys e se traçou como objetivo a implementação do sistema de acordo com as diretrizes do trabalho conforme postas no documento de proposta do trabalho, publicado na página do dito curso. Doravante, serão as tais diretrizes tratadas como requisitos a serem alcançados quando do desenvolvimento do software em si, bem como o próprio documento de proposta tratado como cliente interessado no desenvolvimento do sistema.

A elaboração desse trabalho escrito ocorre de forma paralela ao desenvolvimento do sistema proposto, de forma que foi construído de forma incremental, assim como o software em si. É de natureza descritiva, assemelhando-se a um artefato de documentação de software, ou mesmo um documento de especificação de requisitos, dado que em sua maior parte trata da elaboração do projeto, em vez de lidar com sua feitura, efetivada na geração de código propriamente dita.

O trabalho está estruturado em 5 seções, além dessa introdutória, a saber: Projeto do Sistema, na qual se esboça um documento de projeto de sistema; Modelando o Negócio, que trata da análise dos requisitos propostos, que ocorre por meio da utilização de diagramas UML, Mapa Mental e Diagramas Entidade Relacionamento; Criando a Aplicação Web, que descreve a lógica da implementação dos requisitos em forma de código escrito; Business Process Reengineering, em que se esboça possíveis aplicações de boas práticas de reengenharia de processos de negócios, bem como técnicas de aprimoramentos serem utilizadas no desenvolvimento do software; e, por fim, a Conclusão, que arremata o trabalho como um todo, realizando certas considerações a respeito do processo como um todo.

2. PROJETO DO SISTEMA

2.1 ESCOPO DO PROJETO

A Oficina Chave de Rodas é um centro automotivo de pequeno porte que oferece um serviço voltado às necessidades e fidelização dos clientes. Sua popularidade tem crescido no sentido de que o seu número de clientes incrementou, bem como a positividade dos comentários entre os clientes, gerando um sistema de "marketing boca a boca" que tem trazido novos usuários para a oficina. Dentre suas políticas de serviço, um dos diferenciais é a forma como os serviços são agendados e sua conclusão é estimada, se forma que os clientes sempre têm uma ideia precisa de quando o seu carro estará disponível, após a conclusão do serviço.

A empresa mantém uma política de preços de serviços que reflete uma posição de valorização interna da especialização dos funcionários, de forma que os preços por hora de serviço são diferenciados com base na categoria da especialidade do serviço. Assim, os serviços são categorizados em termo de custo e pagamento dos funcionários nas categorias de (a) motor, (b) suspensão, (c) freio, (d) parte elétrica e (e) parte eletrônica.

Os clientes solicitantes dos serviços realizados na oficina, atualmente, são recebidos por um técnico que, por meio de ouvir o cliente e analisar o veículo, busca a chegar a um diagnóstico do problema, de forma a poder informar o cliente sobre quais serviços precisarão ser realizados, seus custos e tempo necessário para sua execução. Diante da concordância do cliente, é gerada uma Ordem de Serviço e o veículo é direcionado ao setor (ou setores) em que os serviços serão efetivamente executados.

Perceba-se que, uma vez que o veículo transitará entre diversos setores, torna-se necessário que haja uma forma bem planejada e específica de compartilhar informações a respeito do veículo, dos serviços já realizados e das tarefas a serem efetuadas. Adicionalmente, é necessário um registro preciso de todos os itens de peças trocadas/instaladas e produtos utilizados durante o serviço, de forma a poder incluir seus custos no preço final.

Ao final de toda a sequência de serviços, o veículo é enviado ao setor financeiro e é gerado um pagamento a ser realizado pelo cliente. Tal pagamento

abrange todo o custo pelo serviço e, calculando com base nesse custo, aplica um valor de desconto sobre o valor total a ser pago pelo cliente. O pagamento pode ser feito, também, por meio de parcelamento, quer junto à oficina, ou por meio da própria função de parcelamento oferecida por cartão de crédito.

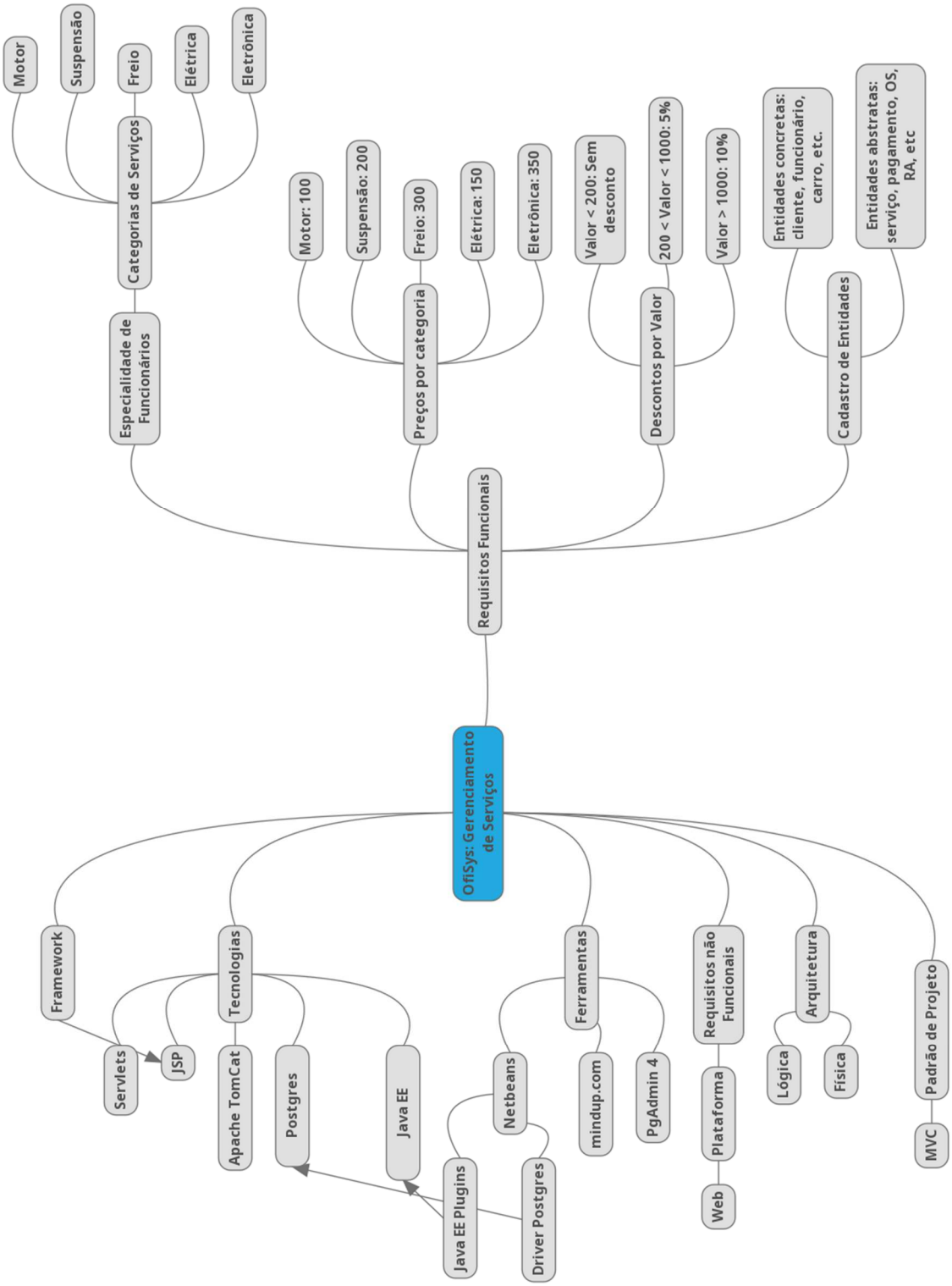
Trata-se, portanto, de um ambiente com atividades dinâmicas e variadas, focado em eficiência dos processos, o que é traduzido na exigência de satisfação do cliente, agilidade na feitura das tarefas e flexibilidade nas diversas possibilidades de cursos de ação.

2.2 JUSTIFICATIVA

Diante de todo o panorama em que se encontra a Oficina Chave de Rodas, bem como do tipo de serviço que se espera que seja realizado, os gestores da oficina chegaram à conclusão de que precisam de um sistema de gerenciamento de dados e processos, de forma que as atividades e informações necessárias ao funcionamento da oficina estejam melhor organizados e bem estruturados. Sente-se que essa melhor organização e estruturação certamente contribuirão para a formalização das boas práticas já utilizadas na oficina, cristalizando-as em um padrão estabelecido de serviço, o que resultará não somente em um aprimoramento em termos de gerenciamento de informação, mas também um incremento na qualidade do serviço e do atendimento como um todo.

2.3 MAPA MENTAL

Figura 1: Mapa Mental com Especificações do Projeto



Fonte: Autoria Própria

2.4 MAPEAMENTO DE REQUISITOS

Durante o processo de levantamento de dados, percebeu-se que o estabelecimento carece de uma estrutura mais organizada e padronizada. Os dados a respeito de como as tarefas na oficina são estruturadas não informam nenhuma sequência organizacional específica; e, apesar de fornecer algumas restrições na maneira como as informações são organizadas, assim como algumas políticas de cobrança e divisão de tarefas, não é possível abstrair uma maneira específica de funcionamento da oficina. A forma como o atendimento, encaminhamento e realização dos serviços não acontece de uma forma padronizada, adaptando-se à situação e necessidade do cliente.

De forma a respeitar essa filosofia e abordagem da empresa, bem como traçar um curso de ação que garanta consistência, é necessário que o projeto permita a natural fluidez de que a oficina lança mão, sem, contudo, perder a possibilidade de formalizar os processos e estruturar os dados e informações que deles derivam e com eles colaboram.

2.4.1 Requisitos Funcionais

- Os serviços realizados podem ter um desconto de acordo com o valor final dos serviços e o pagamento ser parcelado em até 3 vezes também.
    - A política de descontos segue a seguinte lógica
        - De R$200,00 a R$ 1.000,00: desconto de 5%
        - Superior a R$ 1.000,00: desconto de 10%
- Os colaboradores da oficina mecânica “Chave de Rodas”, são divididos em cinco especialidades: Motor, Freio, Suspensão, Parte Elétrica e Parte Eletrônica
- O Caso de Uso em destaque seria a Ordem de Serviço, onde o cliente chega na oficina, o técnico abre uma Ordem de Serviço e começa a apontar os serviços solicitados pelo cliente, em seguida já deixa o status dessa OS como “Em Execução”
- Há preços diferenciados para cada tipo de serviço, conforme suas especialidades:

  - Motor R$ 100,00
  - Suspensão R$ 200,00
  - Freio R$ 300,00
  - Parte Elétrica R$ 150,00
  - Parte Eletrônica (i) R$ 350,00
- O sistema deverá contemplar o cadastramento dos clientes, dos carros, das ordens de serviço, dos funcionários e suas especialidades, dos preços dos serviços, das formas de pagamento, processos de abertura de ordem de serviço, alteração e fechamento da ordem de serviço.

2.4.2 Requisitos Não Funcionais

O único requisito não funcional é a utilização de uma estrutura que permita a utilização do sistema como um serviço por meio da internet. O sistema deve ser desenvolvido de forma que se possa acessá-lo por meio de um navegador, que em um ambiente desktop bem como por meio de dispositivos portáteis e móveis.

2.5 CRONOGRAMA DE EXECUÇÃO

A execução das atividades a serem realizadas para o desenvolvimento do sistema ficaram programadas para ocorrer da seguinte forma:

Tabela 1: Cronograma de desenvolvimento

DATAS PREVISTAS	TIPO DE TAREFA A SER REALIZADA
15 a 21 de abril	Desenho dos casos de uso
22 a 28 de abril	Desenho dos DER e Diagramas de Classe
29 de abril a 05 de maio	Desenvolvimento das classes Java e JSPs
06 a 09 de maio	Conclusão da Documentação do projeto

Fonte: Autoria própria

2.6 CICLO DE VIDA E METODOLOGIA DE DESENVOLVIMENTO

O ciclo de vida de desenvolvimento do projeto será o Desenvolvimento Incremental e Iterativo, uma vez que permite a sequenciação do tradicional modelo em Cascata, partindo do Levantamento e Análise de Requisitos, de forma a

entender o problema; então seguindo rumo ao Desenho do Sistema e sua Implementação, as quais dentro do escopo deste trabalho acadêmico, serão feitas de forma combinada, bem como a Verificação e Manutenção posterior ao processo de desenvolvimento. Como forma de controle de versão, utilizar-se-á a plataforma GitHub para permitir acesso às diversas formas pelas quais o sistema há de passar antes de alcançar sua cristalização na forma mais estável de distribuição.

De maneira a se ajustar à pretendida arquitetura lógica a qual se pretende que o sistema assuma, optamos por utilizar a Orientação a Objeto em todas as fases do processo de desenvolvimento. Assim teremos uma Análise de Requisitos Orientada a Objetos, utilizando UML para modelagem de casos de uso e diagramas de classe, por exemplo, bem como a utilização de Programação Orientada a Objetos em Java para a execução do sistema *per se*.

2.7 DEFINIÇÃO DA ARQUITETURA

2.7.1 Arquitetura Lógica

A forma como a aplicação será organizada logicamente busca abstrair o processo da maneira proposta pelo padrão MVC.

Na camada inferior, ficará o banco de dados com as informações a serem gerenciadas pelo sistema. Logo acima dela, se apoiará dois conjuntos de classes Java, a saber (1) as classes responsáveis por garantir a estruturação dos dados de forma orientada a objeto (JavaBeans), e (2) um grupo de classes que faz o acesso direto aos dados no banco subjacente, chamadas de Direct Access Objects (DAO). Esse arranjo é análogo à camada de Modelo do padrão MVC.

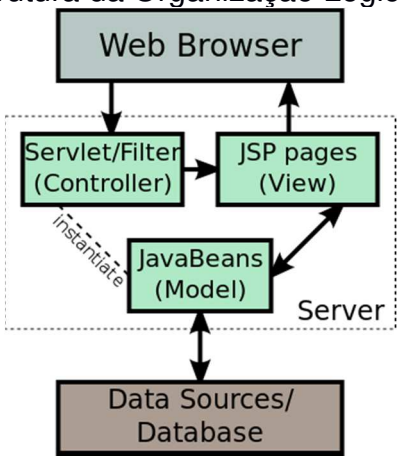
Gerenciando as regras de negócio e mediando as interações com o usuário, estarão um grupo de classes do tipo HttpServlet, as quais recebem as requisições, tratam-nas e devolvem as respostas com os recursos ou informações solicitadas. Dessa maneira, esse grupo de classes se assemelha à camada de Controle no padrão MVC.

Finalmente, um segundo grupo de HttpServlets (gerados automaticamente por documentos do tipo JSP) se encarregam de fazer a interface de utilização gráfica para utilização do usuário final. Faz isso pela geração *on demand* de conteúdo

HTML a ser renderizado do lado do cliente, por meio do browser.

A estruturação lógica do sistema pode ser verificada na figura seguinte:

Figura 2: Estrutura da Organização Lógica do Sistema



Fonte: https://en.wikipedia.org/wiki/JavaServer_Pages#/media/File:JSP_Model_2.svg

2.7.2 Arquitetura Física

A arquitetura física segue o modelo Cliente-Servidor, dado que todo o funcionamento será por meio de requisições feitas por meu de um navegador web ou ferramenta análoga, às quais responderá um servidor web (o web-container Glassfish, nesse caso), devolvendo as informações e/ou recursos solicitados.

2.8 PADRÕES DE PROJETO E FRAMEWORKS

Para o desenvolvimento do sistema como um todo, será utilizado o padrão MVC, de maneira a se obter uma boa modularidade, bem como menor acoplamento entre as diferentes partes do projeto. A proposta inicial do projeto abrange um ambiente de gerenciamento da oficina que se aloca no contexto de uma aplicação web. Todavia, há a possibilidade de, em algum momento futuro, ser necessária a expansão do sistema para algum tipo de plataforma móvel. Destarte, o padrão MVC permite que se tenha a possibilidade de utilizar as camadas de controle e modelo, havendo necessidade de implementar apenas a camada de visão dentro do contexto de um aplicativo móvel.

Devido a restrições técnicas, optou-se por não utilizar um dos frameworks comumente utilizados para o tipo de proposta do trabalho. Em vez disso, lançou-se

mão da já consolidada tecnologia de Servlets Java, bem como de JavaServer Pages (JSPs), com o intuito de gerar as páginas de gerenciamento de informação do software.

2.9 TECNOLOGIAS APLICADAS

Visto que objetiva-se, neste estudo, o planejamento, esboço/prototipação e desenvolvimento (1) das estruturas de armazenamento de dados, (2) processos de negócio envolvendo a manutenção do estoque de peças para reposição e materiais utilizados nos serviços e (3) sistema informacional de gerenciamento dos dados modelados e implementados na camada de armazenamento de dados, de forma a prover escalabilidade e facilidade de acesso do sistema (já se antecipando à possibilidade de posterior ampliação da empresa em diversas filiais), optar-se-á pelo uso de uma aplicação web, a qual será hospedada em um servidor local ou *cloud based*. Destarte, poder-se-á acessar a aplicação independentemente de restrições espaciais.

Para satisfazer a exigência de levantamento e análise de requisitos, será utilizada a Unified Modeling Language – UML como ferramenta para melhor compreensão das atividades a serem realizadas pelo sistema, descrevendo e especificando os requisitos funcionais do sistema. O uso da UML proporcionará, também, refinamento gradual das especificações do sistema, caminhando na direção da implementação definitiva da solução.

Para a modelagem dos dados subjacentes ao sistema, elaborar-se-á um Diagrama Entidade Relacionamento – DER, tanto em uma visão conceitual quanto uma visão lógica, mais perto da implantação em um Sistema de Gerenciamento de Banco de Dados – SGBD, propriamente dito. Optou-se por utilizar um banco de dados seguindo o modelo Relacional, utilizando a Structured Query Language – SQL como linguagem de gerenciamento e consulta de dados.

A aplicação será desenvolvida utilizando tecnologia Java, que se comunicará com o banco de dados subjacente por meio do Java Database Connectivity – JDBC, responsável por fazer a interface entre aplicação e dados. Isso facilitará o envio de comandos SQL para o SGBD. Essa combinação de SGBD e JDBC permite realizar a comunicação entre aplicação e banco de dados de maneira fluida, leve e robusta.

A interface entre as classes/objetos Java e a apresentação das informações ao usuário será feita por meio do uso de JavaServer Pages, encapsulando muitos dos procedimentos repetitivos e passíveis de erro que consiste da geração manual e declarativa de páginas por meio de outras classes Java. Objetiva-se seguir o Padrão Model-View-Controller – MVC no desenvolvimento. Com isso, poderá ser mantida a modularidade de partes do sistema, de maneira a facilitar a manutenção e a organização das classes/objetos e outros componentes da aplicação.

De maneira a garantir concisão na descrição desse trabalho, o código fonte dos principais arquivos descritos no texto pode ser encontrado nos apêndices. Foram omitidos os métodos setters e getters que não possuíam comportamento específico diferente do de um Plain Old Java Object, apresentando somente os métodos que possuem alguma lógica interna específica ou cuja apresentação seja de relevância para a compreensão da descrição.

Além disso, diante da impossibilidade de incluir todo o código fonte no corpo deste trabalho, os arquivos resultantes do processo de desenvolvimento realizado podem ser acessados na íntegra no repositório criado pelo autor, hospedado na plataforma GitHub. O acesso é público e pode ser realizado pelos hyperlinks: https://goo.gl/H252jv ou http://bit.ly/portfolio4_saint.

Além das classes Java e dos arquivos JavaServer Pages, pode-se encontrar o arquivo em formato .sql gerador do banco de dados. Além do trecho desse mencionado arquivo que incluído no Apêndice A, responsável pela definição da estrutura de dados do banco, pode-se encontrar um script que, ao ser executado, realizará a população do banco com dados de teste, os quais foram utilizados durante o processo de desenvolvimento. Pode ser encontrada, ainda, uma pasta compactada com todo o conteúdo do projeto para ser importado no Netbeans, bem como uma pasta compactada com os diagramas UML utilizados bem como as imagens deles exportados.

2.10 FERRAMENTAS

Para se elaborar os diagramas em notação UML utilizar-se-á a ferramenta Astah Community Edition, que possui algumas limitações, por ser versão não paga, sem, não obstante, interferir nas metas a serem alcançadas por este trabalho.

Como base para a modelagem e gerenciamento de dados, utilizar-se-á a ferramenta BrModelo para fazer a modelagem da estrutura de dados nos diagramas Entidade Relacionamento – DER lógico e conceitual.

O SGBD utilizado será o PostgreSQL, juntamente com o programa PgAdmin, para gerenciamento do banco em alto nível.

De forma a otimizar o processo de desenvolvimento, o Ambiente Integrado de Desenvolvimento (IDE) utilizado será o Netbeans, integrando-o com o WebContainer Glassfish, em sua versão 8.

E finalmente, como forma de controle de versão, optamos por utilizar a plataforma GitHub, uma vez que permite a sincronização por meio de git, bem como upload de arquivos diretamente por meio de um Browser, dadas as restrições quando da elaboração do projeto e desenvolvimento do sistema.

3. MODELANDO O NEGÓCIO

3.1 DIAGRAMAS ENTIDADE RELACIONAMENTO – DER

Outro passo importante para definição de todos os aspectos do sistema é a determinação de como os dados a serem utilizados pelo sistema serão estruturados. Isso impactará em todos os outros aspectos do desenvolvimento e implementação, visto que todo o desenvolvimento das outras partes do sistema precisa se conformar com o formato em que os dados, na camada mais baixa do sistema, estão organizados.

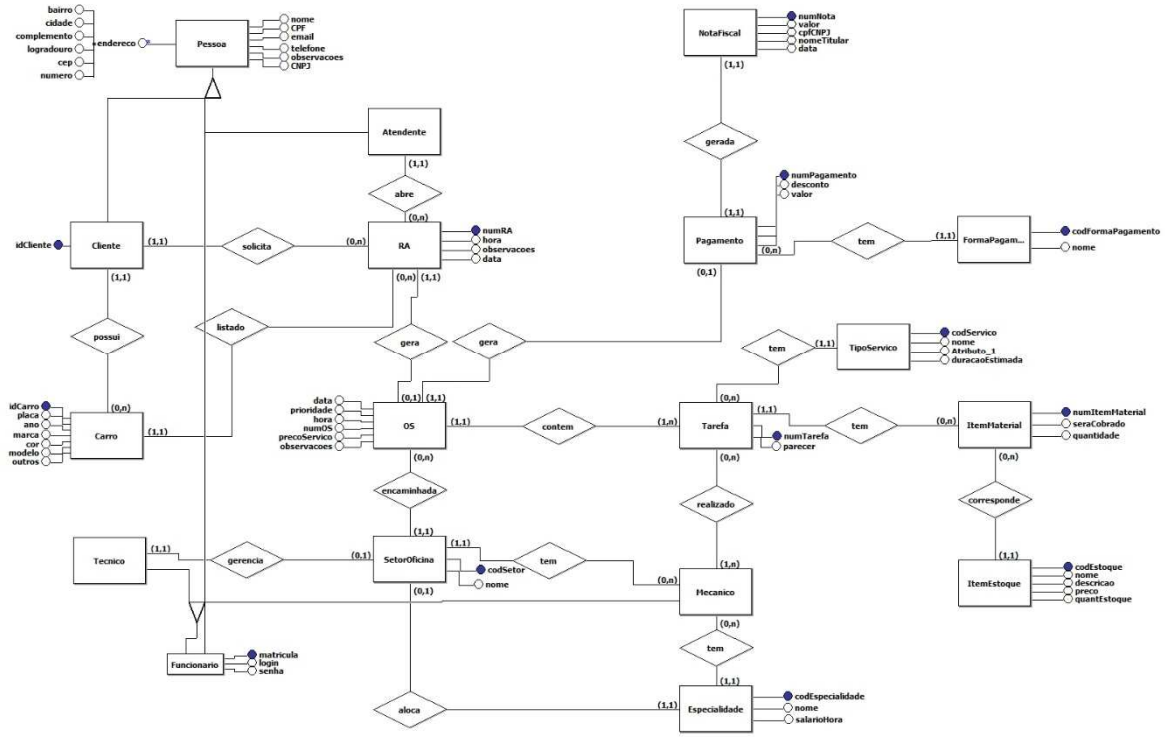
Tão importante é esse processo que, embora somente agora esteja sendo abordado diretamente, teve precedência sobre algumas das etapas já tratadas nesse trabalho. Por exemplo, a estrutura das classes, representadas no Diagrama de Classes, foi determinada pela maneira que os dados foram modelados e estruturados.

Por meio da ferramenta BrModelo, foi feita a representação da estrutura que os dados do sistema devem assumir. Elaborou-se um Diagrama Entidade Relacionamento (DER) para se ter uma representação de alto nível da estrutura assumida pelos dados, sendo um modelo conceitual do banco de dados relacional a ser implementado. Por meio desse modelo criado, foi possível elaborar um modelo lógico, representando a forma com que os dados serão representados na camada mais baixa do sistema, mais especificamente no SGBD.

3.1.1 Modelo Conceitual

Por meio das especificações fornecidas pelos requisitos chegou-se a uma possível representação dos dados, elencando os atributos que permitiriam dar suporte a ações que garantam a implementação das regras de negócio da empresa. Definiu-se quais os dados adicionais que precisariam ser incluídos na representação, além dos listados nos requisitos funcionais, e que tipo de relacionamento cada entidade participante no sistema teria com as demais. Chegou-se ao seguinte Diagrama:

Figura 3: DER - Modelo Conceitual



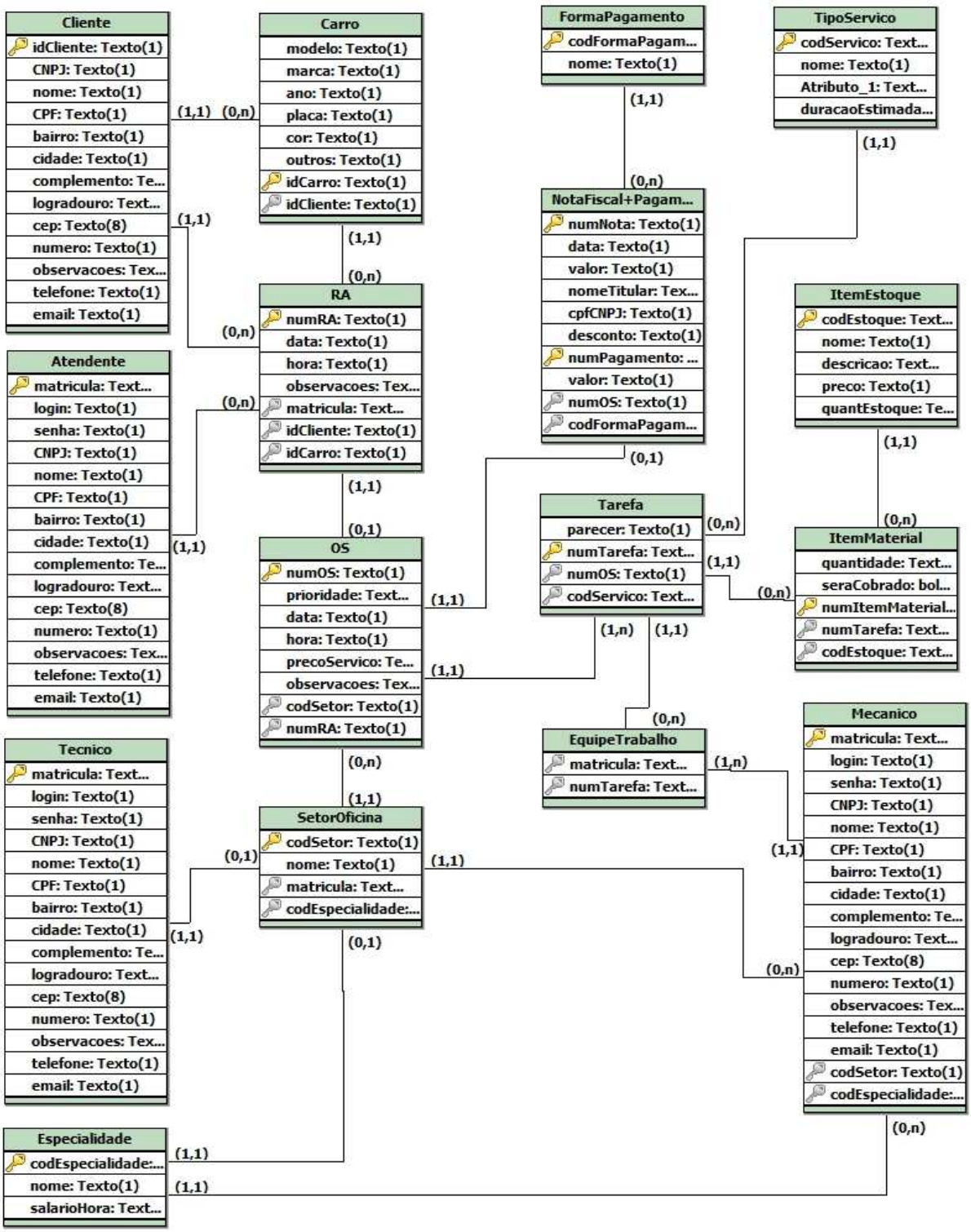
Fonte: Autoria Própria

3.1.2 Modelo Lógico

A partir do modelo conceitual elaborado, apresentado anteriormente, foi possível ter uma ideia de, em um banco de dados relacional, quais relações precisariam ser definidas para efetivar a implementação do que foi modelado. Além disso, com base nos atributos elencados no modelo conceitual, definiu-se os atributos de identificação de cada relação (Chaves Primárias) bem como os atributos identificadores de relacionamento entre as entidades (Chaves Estrangeiras).

Segue o modelo lógico elaborado seguindo as determinações do modelo conceitual:

Figura 4: DER Modelo Lógico



Fonte: Autoria Própria

3.2 GERANDO O BANCO DE DADOS

Havendo sido estabelecido o modelo lógico, prosseguiu-se para a criação efetivado banco de dados a ser utilizado pela aplicação web. Para tal, utilizou-se do PostgreSQL versão 9.6.1, sendo que, para facilitar a inspeção das estruturas criadas, utilizou-se também o PgAdmin III, o qual permite a visualização dos dados de forma tabular, bem como facilita algumas tarefas de ordem prática.

Foi criado um banco de dados e adicionadas as relações em conformidade com o que ficou definido nos modelos conceitual e lógico do DER. Com o intuito de facilitar a geração de dados e minimizar quaisquer erros de consistência, bem como violações de *constraints*, optou-se por utilizar sequências para geração de chaves primárias das tuplas em cada relação.

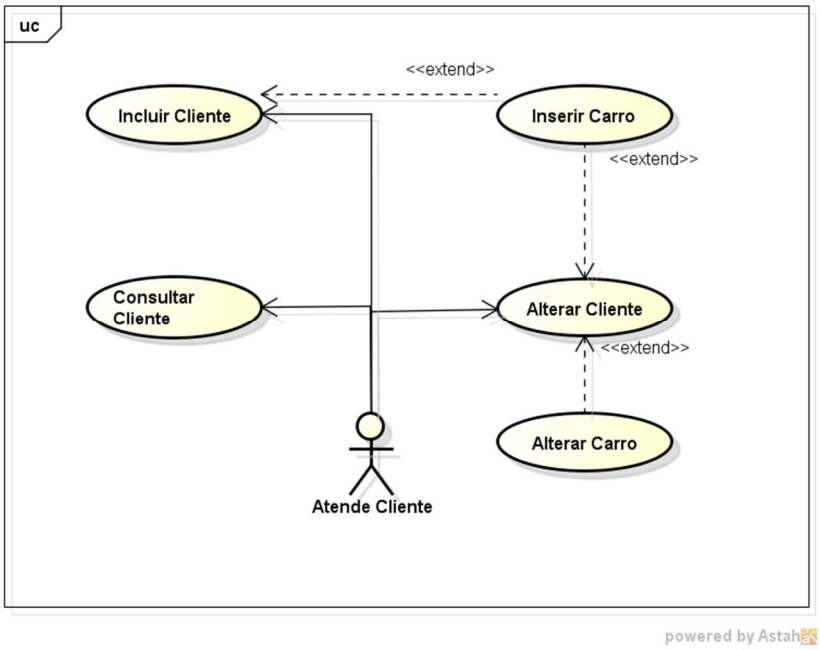
Confira-se no Apêndice A o código com os comandos SQL utilizados para gerar o banco de dados com os detalhes descritos.

3.3 DIAGRAMAS DE CASOS DE USO

De maneira a melhor compreender os detalhes de como o sistema deve funcionar, elaborou-se uma série de diagramas de Caso de Uso, seguindo o padrão UML, utilizando a ferramenta Astah Community Edition. O foco principal do sistema são as ações que atuam sobre as seguintes entidades (as quais foram também representadas em DER, mais adiante apresentado): (1) Atendente, (2) Carro, (3) Cliente, (4) Especialidade, (5) Forma de Pagamento, (6) Item de Estoque, (7) Itens de Material Utilizado, (8), Mecânico, (9) Ordem de Serviço, (10) Pagamento, (11) Registro de Atendimento, (12) Setor da Oficina, (13) Técnico, e (14) Tipo de Serviço.

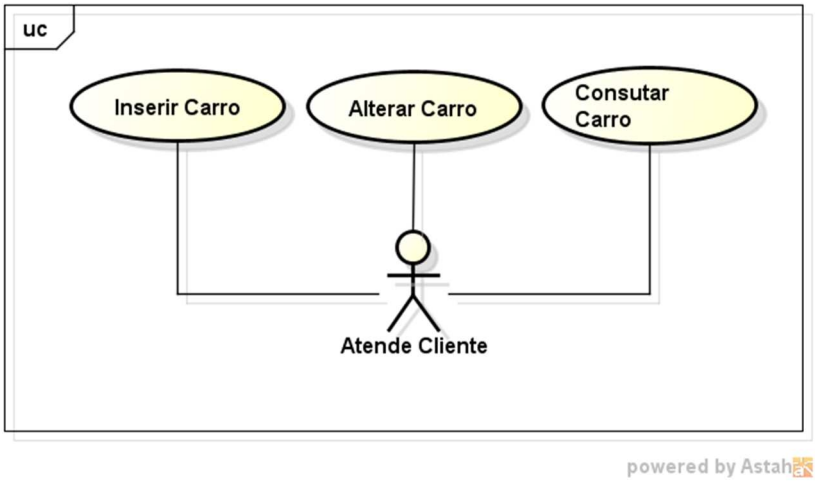
Cada uma das entidades, bem como outras que se constituem em itens componentes do todo, devem, nas especificações de um sistema completo, ter operações a serem executadas pelo sistema que permitam a inclusão, alteração, consulta e exclusão (*Create*, *Read*, *Update* e *Exclude* – CRUD). Todavia, dentro do escopo determinado para a execução desse trabalho, exigiu-se apenas operações sobre o cadastro de cliente, cadastro de produto e o controle de serviços do sistema. Destarte, apresenta-se os Casos de Uso dos controles de cada uma das entidades apontadas.

Figura 5: Diagrama de Casos de Uso do Controle de Clientes



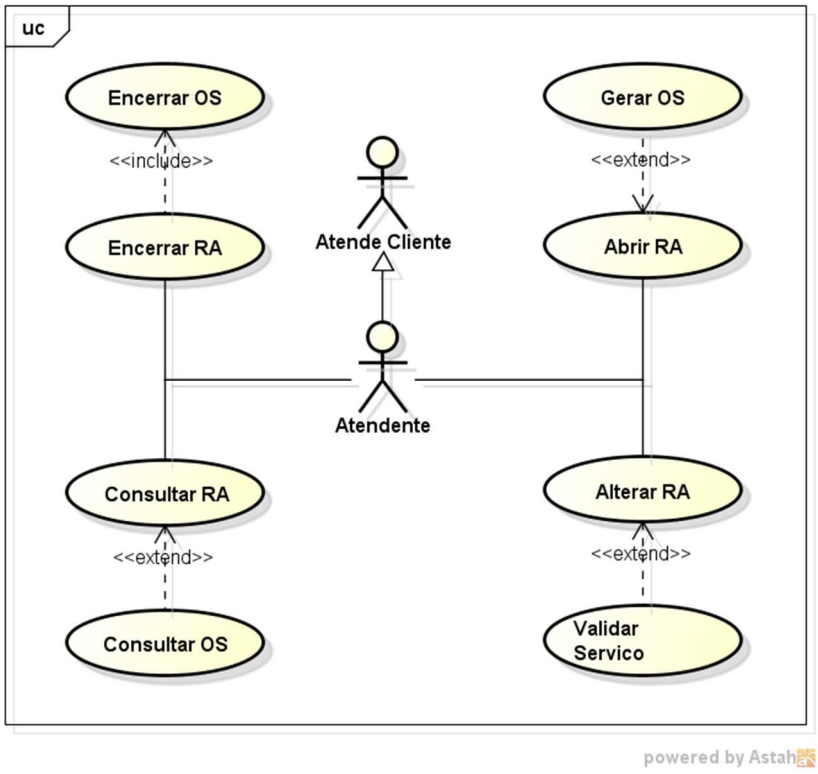
Fonte: Autoria Própria

Figura 6: Diagrama de Casos de Uso do Controle de Carros



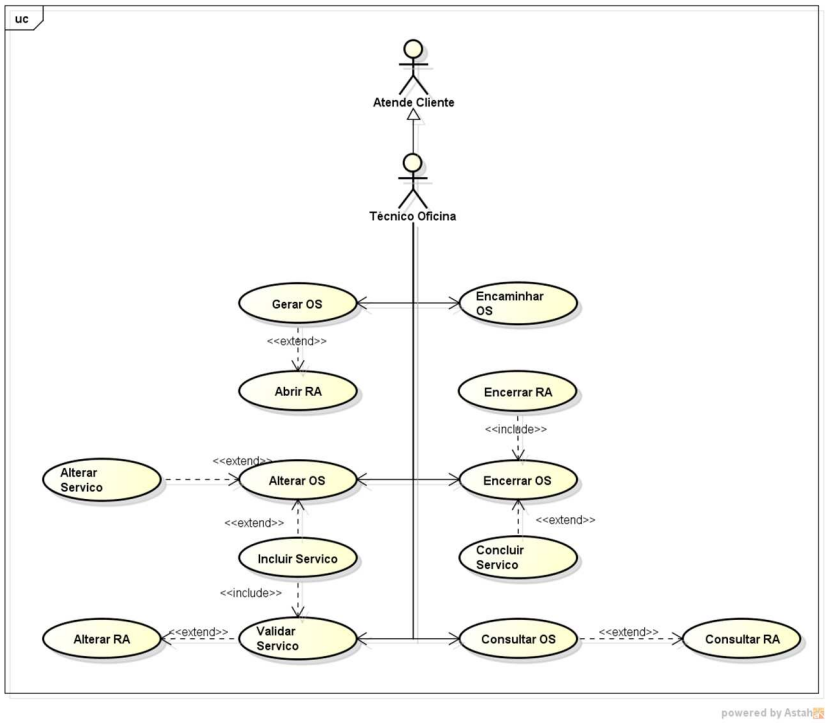
Fonte: Autoria Própria

Figura 7: Diagrama de Casos de Uso do Controle de Registros de Atendimento



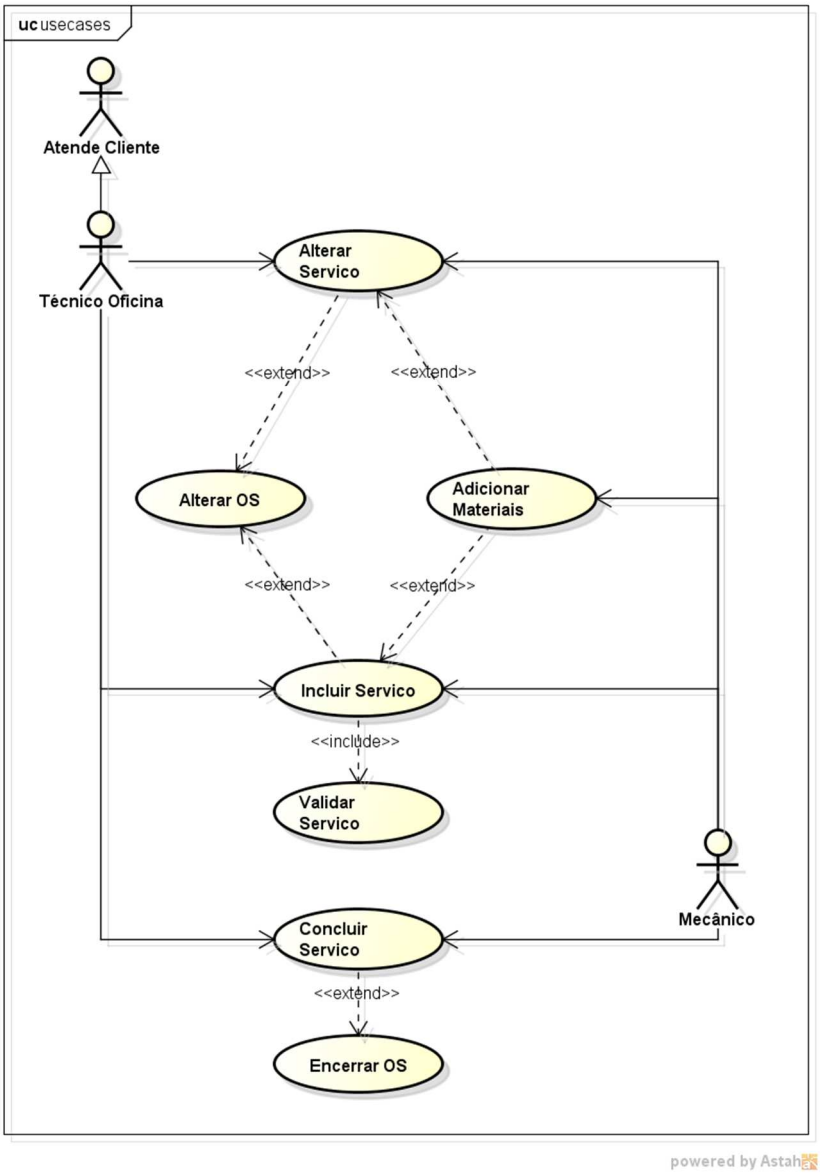
Fonte: Autoria Própria

Figura 8: Diagrama de Casos de Uso do Controle de Ordens de Serviço



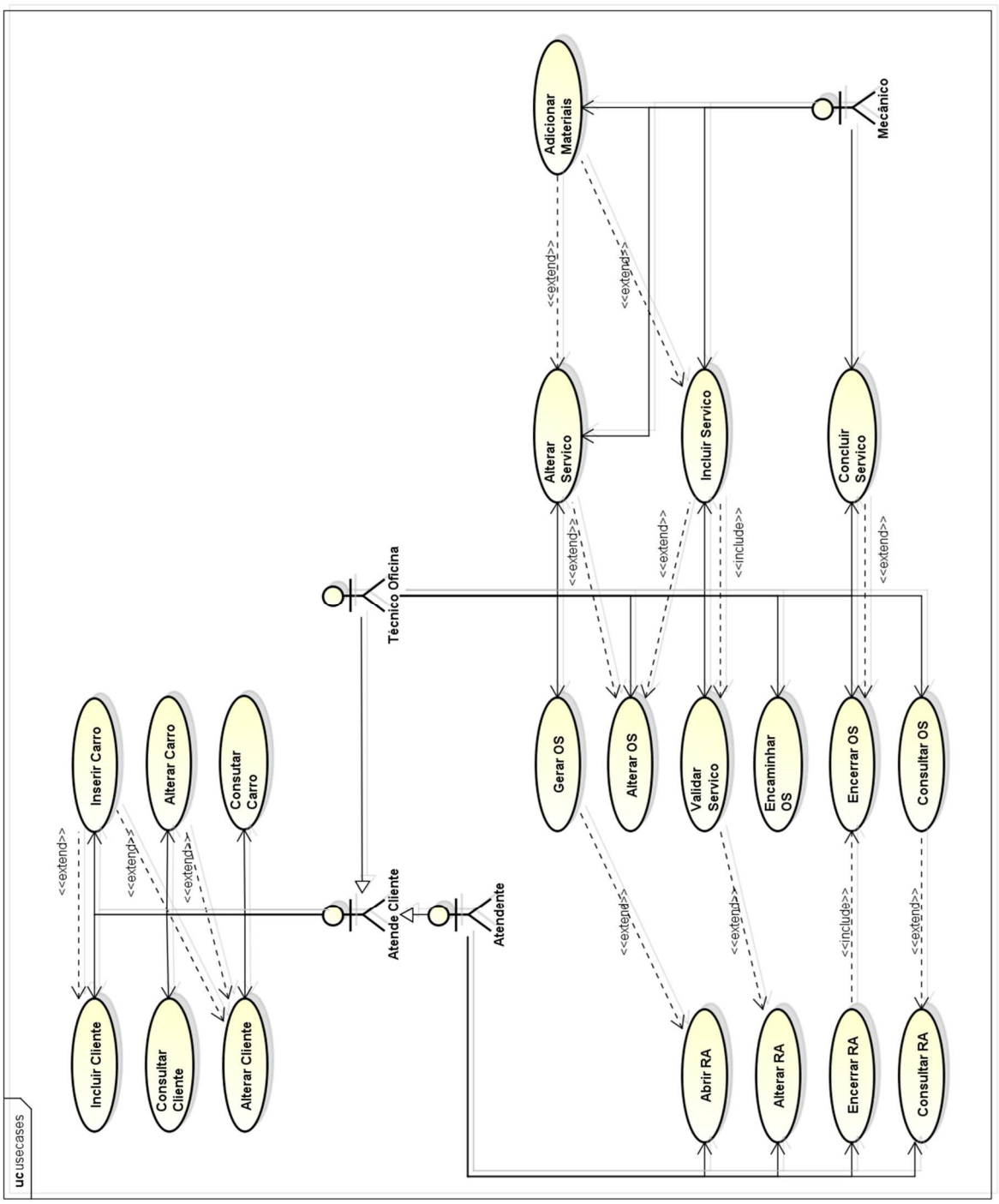
Fonte: Autoria Própria

Figura 9: Diagrama de Casos de Uso do Controle de Serviços a Realizar



Fonte: Autoria Própria

Figura 10: Diagrama de Casos de Uso do Sistema



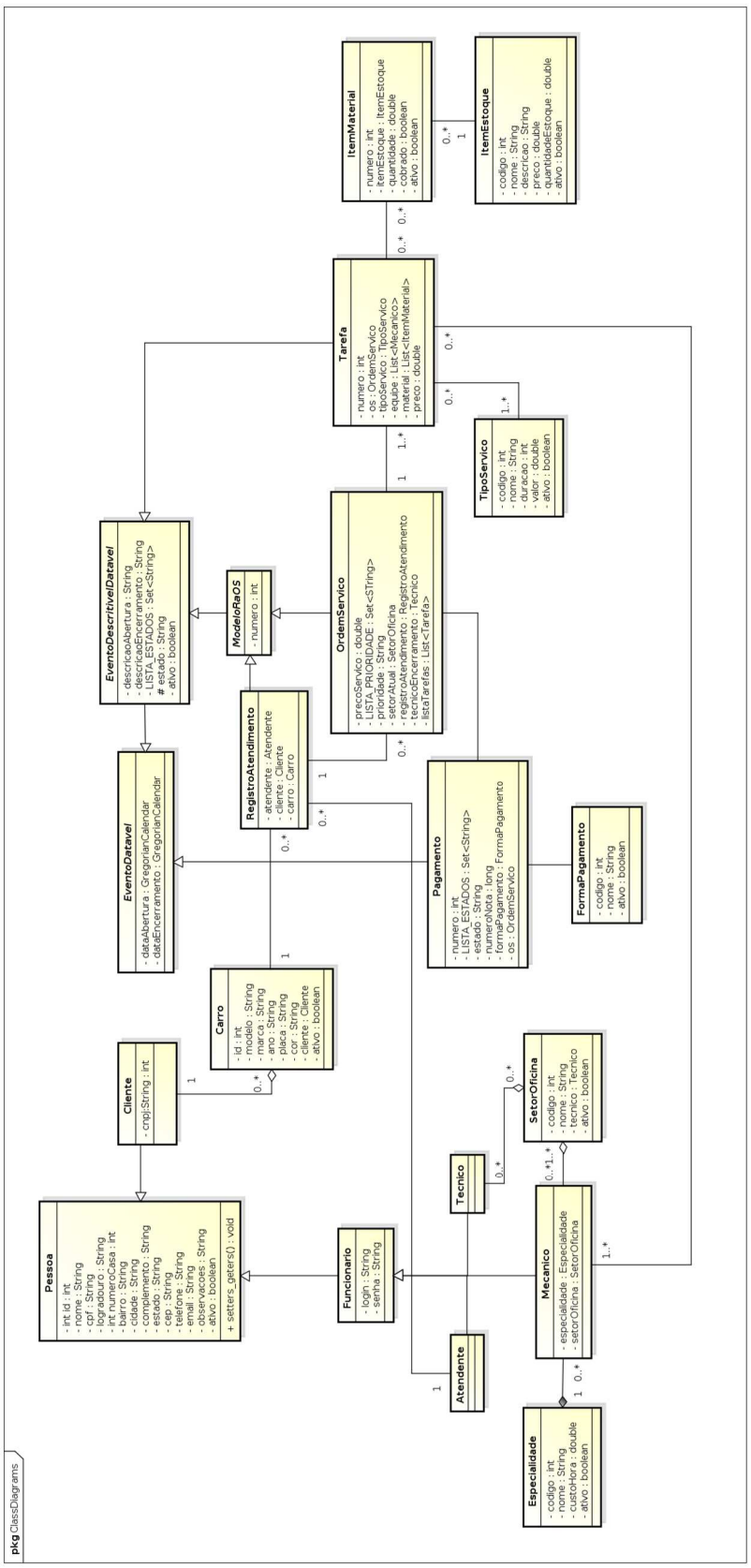
Fonte: Autoria Própria

3.4 DIAGRAMA DE CLASSES

Havendo sido definidos e especificados, por meio dos Diagramas de Casos de Uso, os processos a serem realizados pelo sistema prosseguiu-se com a elaboração

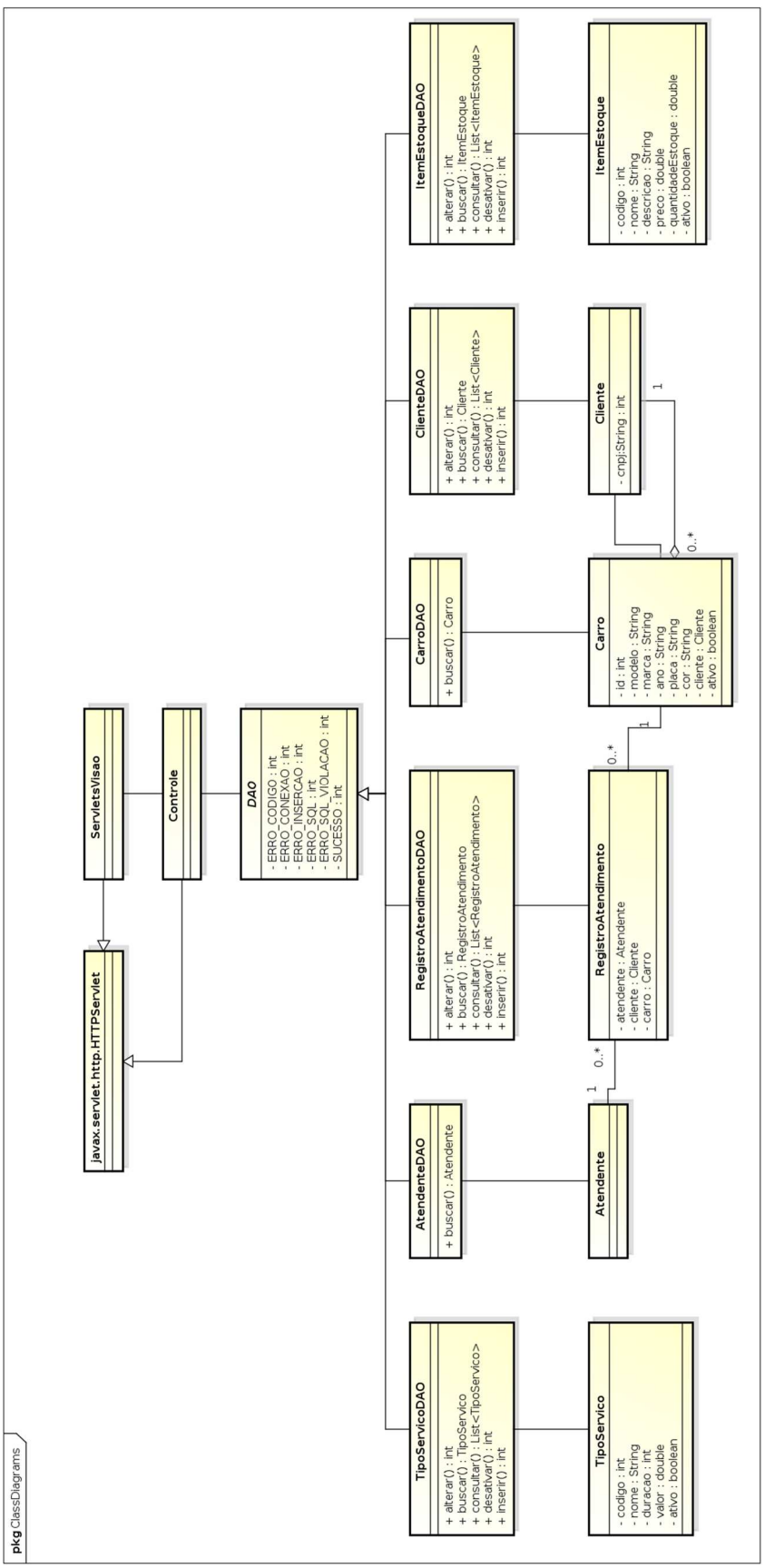
do conceitual de como se daria a implementação per se do sistema, idealizando-se quais as classes que se poderia utilizar para modelar os objetos, mensagens e interações componentes do sistema. Com esse intuito, elaborou-se o Diagrama de Classes, tendo em mente as funcionalidades elencadas no Levantamento de Requisitos, bem como as especificações resultantes dos Diagramas de Caso de Uso anteriormente elaborados. De igual importância para a definição de quais classes se utilizaria, bem como quais seriam as relações entre elas, foi a modelagem da estrutura de dados subjacente ao sistema.

Figura 11: Diagrama de Classes Usadas para Modelar os Dados



Fonte: Autoria Própria

Figura 12: Diagrama de Classes DAO e de Interação com Usuário



Fonte: Autoria Própria

Note-se que as classes representadas no diagrama contemplam as classes necessárias à modelagem dos dados no sistema, bem como as classes que servirão de interface para gravação e recuperação de informações junto ao banco de dados, responsável fará a persistência dos dados. Além dessas classes, o sistema conta com dois conjuntos de Servlets Java.

O primeiro dos conjuntos contém Servlets que atuam como controladores, do padrão MVC, de forma a mediar o acesso ao banco de dados e a apresentação. Há um *Servlet* de Controle para cada uma das Classes de modelagem, o qual recebe os dados do usuário, faz o seu tratamento e, por meio da interface de acesso aos dados, os envia para a persistência.

Quanto ao segundo grupo de Servlets, trata-se, na verdade de um conjunto de páginas do tipo JavaServer Pages que, em tempo de execução, são convertidas em classes Servlets Java as quais são compiladas. Esses Servlets atuam como geradores dinâmicos de páginas HTML, encarregando-se de enviá-las ao Browser para interação com o usuário.

O fluxo de informação é orientado pelos Servlets de Controle, recebendo inputs do usuário, via requisições HTTP e gerando conteúdo por interação com o banco de dados subjacente, direcionando a resposta HTTP para as JSPs gerarem o conteúdo final a ser exibido ao usuário no Browser.

4. CRIANDO A APLICAÇÃO WEB

Havendo sido preparada toda a camada inferior da aplicação, prosseguiu-se com o desenvolvimento da aplicação em si. Foram escritas os JavaBeans para modelagem dos dados a serem recuperados do banco de dados, bem como as classes de Acesso Direto aos Dados (DAO Classes) para cada classe de modelagem. Algumas outras classes auxiliares foram construídas para simplificar determinadas tarefas e deixar cada classe responsável apenas pelas tarefas específicas de sua natureza.

Concluída a camada de Modelo, foram escritos os Servlets de Controle de cada conjunto de operações requerido pelas especificações e requisitos. Optou-se, nesse ponto do desenvolvimento, para se conformar apenas com as exigências expressas no documento que explica as tarefas motivadoras desse trabalho. Mais especificamente, incluiu-se nas classes de controle atributos constantes com nome de usuário, senha e endereço do banco de dados, de maneira que não se precisasse construir uma interface para execução de login, liberando tempo e recursos para enfocar nas tarefas especificadas.

Foram executados testes utilizando um simples programa de linha de comando, de forma a garantir o bom funcionamento das partes elaboradas até então. Diante dos resultados positivos dos testes, e havendo sido encerrado o desenvolvimento da camada de controle, deu-se início ao desenvolvimento da camada de visão.

Para a geração da interface gráfica de usuário (GUI), optou-se por utilizar JavaServer Pages – JSP – que, embora sejam mais trabalhosas que os frameworks oferecidos hodiernamente, permitiram uma maior compreensão dos detalhes nos procedimentos ocorrentes em segundo plano. Uma solução utilizando frameworks, pela própria transparência oferecida por estes, não permitiria a percepção exata do fluxo de interações entre os diversos elementos. Assim, o desenvolvimento certamente ficou mais custoso, porém o processo foi mais eficiente no sentido de permitir maior compreensão do processo de comunicação da GUI com as classes de controle.

Não se apresentará, de maneira a garantir a brevidade do trabalho, toda a coleção código nos arquivos JSP. Em vez disso, apresentar-se-á em detalhes

apenas as interfaces implementadas como solução para dois Casos de Uso específicos, a saber: a inserção de Clientes, a inserção de Tipos de Serviço e a inserção de Registros de Atendimento.

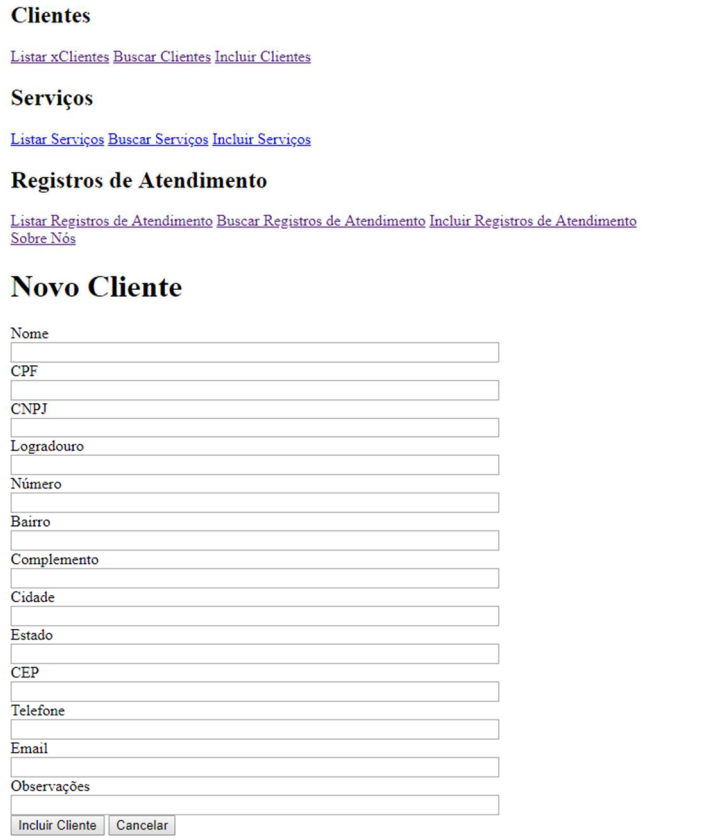
4.1 GERENCIAMENTO DE CLIENTES

De maneira a viabilizar o gerenciamento de informação sobre clientes da Oficina Chave de Rodas, foi criada a classe `Cliente` (ver Apêndice B), com o fim de facilitar o encapsulamento e manipulação dos valores que representam cada cliente. Trata-se de uma classe simples, com seus atributos e seus setters/getters, nada mais.

Em conjunto com ela, utilizamos uma classe para implementar o acesso ao banco de dados e dar conta das transações, chamando-a de `ClienteDAO` (ver Apêndice C) e uma classe `ControleCliente` (ver Apêndice D), que herda de `javax.servlet.http.HttpServlet`. Juntas, essas classes realizam o acesso ao banco de dados para recuperar informações e prepará-las para visualização na camada de visão, assim como recebem dados da GUI, tratam-nos e realizam a persistência no banco de dados.

Para as camadas de visão, utilizamos quatro JSPs principais (`inserirCliente.jsp`, `listarClientes.jsp`, `detalharCliente.jsp` e `alterarCliente.jsp`) e uma JSP alternativa chamada (`buscarCliente.jsp`). O código fonte de inserirCliente.jsp e listarClientes.jsp consta do Apêndice E.

Figura 13: Tela de Inserir Cliente



Fonte: Autoria Própria

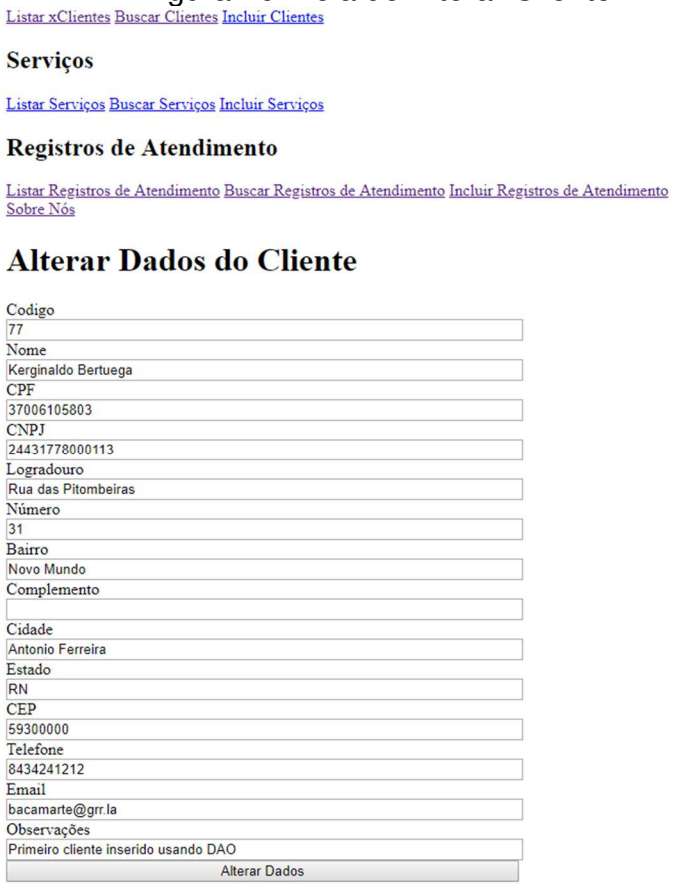
Figura 14: Tela de Listar Cliente

Fonte: Autoria Própria

Figura 15: Tela de Detalhar Cliente

Fonte: Autoria Própria

Figura 16: Tela de Alterar Cliente



Fonte: Autoria Própria

Figura 17: Tela de Buscar Cliente

Fonte: Autoria Própria

Por meio de inserirCliente.jsp, o usuário pode informar e pré-validar dados a

serem usados para geração de um novo cliente a ser inserido no banco de dados. Enquanto que listarClientes.jsp mostra uma lista dos clientes cadastrados com as opções de excluir (diretamente implementada na lista, sem qualquer outra interface intermediária), alterar (direcionando para `alterarCliente.jsp`) e detalhar, que abre uma tela contendo os detalhes do cliente (`detalharCliente.jsp`). A página `buscarCliente.jsp` pede um código de cliente e, encontrado o cliente no banco de dados, redireciona para a página `detalharCliente.jsp` com os dados do cliente buscado, tendo esta, também, as opções de alterar e excluir.

4.2 GERENCIAMENTO DE SERVIÇOS E REGISTROS DE ATENDIMENTO

A maneira de gerenciar os serviços oferecidos pela oficina segue o mesmo padrão utilizado para o gerenciamento de clientes: (1) uma classe `TipoServico` para encapsular os dados relativos aos serviços, apresentada no Apêndice F, (2) uma classe Java para promover o acesso direto aos dados – ver `TipoServicoDAO` no Apêndice G – e (3) uma segunda herdada de `javax.servlet.http.HttpServlet` para atuar no controle de requisições de usuários e suas respostas – ver `ControleTipoServiço` no Apêndice H. O mesmo se deu com os meios de acesso aos dados e controle de requisições e respostas relativos aos Registros de Atendimento, especificamente por meio das classes `RegistroAtendimentoDAO` e `ControleRegistroAtendimento` (Apêndices I, J e K).

Assim como com a interface de manipulação dos dados de clientes, os Registros de Atendimento e os Tipos de Serviços são gerenciados por páginas JSP responsáveis por listar todos os registros (`listarRegistroAtendimento.jsp` e `listarTipoServico.jsp`), buscar por um registro específico (`buscarRegistroAtendimento.jsp` e `buscarTipoServico.jsp`), detalhar um registro em particular (`detalharRegistroAtendimento.jsp` e `detalharTipoServico.jsp`), bem como por inserir (`inserirRegistroAtendimento.jsp` e `inserirTipoServico.jsp`) e alterar (`alterarRegistroAtendimento.jsp` e `alterarTipoServico.jsp`) um dado registro.

Figura 18: Tela de Gerar Registro de Atendimento

Fonte: Autoria Própria

Figura 19: Tela de Listar Registros de Atendimento

Fonte: Autoria Própria

Figura 20: Tela de Alterar Tipo de Serviço

Fonte: Autoria Própria

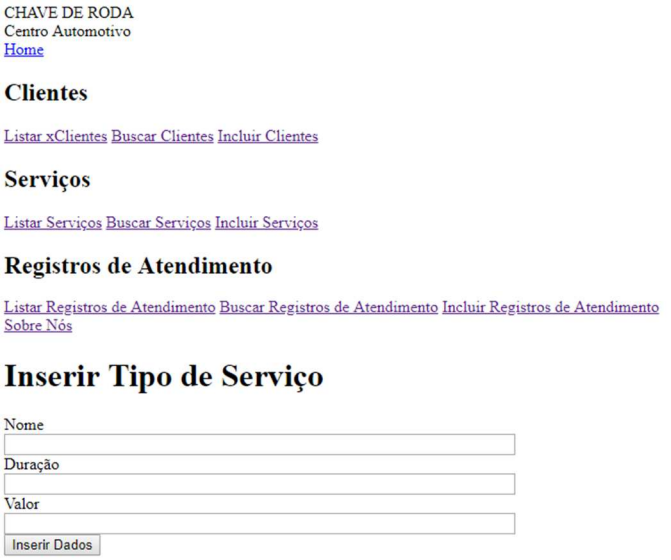
Figura 21: Tela de Buscar Tipo de Serviço

Fonte: Autoria Própria

Figura 22: Tela de Detalhar Tipo de Serviço

Fonte: Autoria Própria

Figura 23: Tela de Inserir Tipo de Serviço



Fonte: Autoria Própria

Figura 24: Tela de Listar Tipo de Serviço

Fonte: Autoria Própria

5. BOAS PRÁTICAS DE GERENCIAMENTO DE PROJETOS

De maneira a se poder gerenciar bem o transcorrer de um projeto de desenvolvimento de software, a organização e formalização dos procedimentos é essencial. Levando em consideração a complexidade de um processo de desenvolvimento de software, há algumas áreas específicas que se tornam prementes de um gerenciamento e controle preciso e eficaz. Dentre todo esse elenco de áreas críticas, são muito expressivos os campos de (1) definição do domínio do problema e detalhamento da proposta de solução a se desenvolver, garantindo uma descrição expressa dos passos, fases e subfases a comporem todo o processo de desenvolvimento; (2) planejamento de recursos relativos ao prazo que se tem para realizar todas as atividades do processo de desenvolvimento, de forma a permitir que seja efetivado de forma eficiente e não consumidora de tempo em demasia; e, finalmente, (3) determinação de abordagens de avaliação do produto em desenvolvimento, incluindo as características a serem aferidas, as formas pelas quais esse mensuramento será realizado, bem como dos recursos necessários a sua implementação.

Partindo disso, podemos perceber a importância de um plano de desenvolvimento de software com o fim de dar conta desses aspectos, especialmente com relação a esses campos de gerenciamento de escopo, gerenciamento de tempo e gerenciamento de qualidade. Destarte, esboçar-se-á uma abordagem em resposta a essa necessidade de se planejar o desenvolvimento do projeto em foco nesse trabalho, a saber, o sistema de gerenciamento de informações da Oficina Chave de Rodas – OfiSys.

Ao se pretender responder a uma demanda por uma solução, é necessário se compreender tanto a natureza do problema em si, como também fazer um planejamento de cada uma das etapas de desenvolvimento da dita solução. É necessário definir o escopo do projeto, bem como delinear todas as partes constituintes de cada uma de suas etapas.

5.1 GERENCIAMENTO DE ESCOPO

Ao tratar de escopo, trata-se de abranger um conjunto de artefatos de entrega

ou características desejadas em um projeto, os quais são derivados dos requisitos de um projeto específico. Refere-se ao trabalho a ser realizado de forma a se entregar um produto, serviço ou resultar em características e/ou funções que satisfaçam aos requisitos de um projeto.

Pode-se apontar três processos de Gerenciamento de Escopo em um projeto: (a) planejamento, que faz uma tentativa de compreender e definir o trabalho que necessita ser realizado; (b) controle e monitoramento, os quais enfocam em documentar, supervisionar e dar conta da aprovação ou não de alterações no projeto; e (c) encerramento, sendo o processo final, que inclui uma auditoria dos produtos do projeto, bem como uma verificação dos resultados em contraste com o plano original.

Definir as necessidades do projeto é o primeiro passo rumo ao estabelecimento de um cronograma do projeto, à alocação de recursos e estabelecimento das metas do projeto. Somente ao definir essas etapas será possível compreender as tarefas a serem realizadas. Nisso consiste a própria definição do escopo do projeto. Com isso, as equipes podem ser designadas a tarefas específicas e se possibilita o direcionamento de que se precisa para a entrega de um projeto dentro do prazo e orçamento estabelecidos.

Uma das ferramentas a se utilizar para se alcançar essa compreensão é o Estrutura Analítica de Projetos (EAP), do Inglês, Work Breakdown Structure (WBS). Em sua essência, trata-se de é um processo de subdivisão das entregas e do trabalho do projeto em componentes menores e mais facilmente gerenciáveis. Garante que se tenha uma visão geral e concisa de toda a proposta do projeto, bem como permite que se tenha uma granularidade de cada uma das partes que constituem as fases mais gerais.

Assim, dentro do contexto em que está a Oficina Chave de Rodas, uma possível forma para um Diagrama de EAP seria da seguinte forma:

- **Documentação Inicial**
  - Elicitação de Requisitos
  - Análise de Requisitos
  - Validação de Requisitos
- **Definição das funções**
  - Estipular as funções dos membros da equipe

- Definir tarefas para cada função.
- Averiguar satisfação da equipe.

- **Definição das ferramentas**
  - Escolher qual servidor de hospedagem será utilizado
  - Definir a linguagem de programação que será utilizada
  - Definir o banco de Dados
- **Desenvolvimento**
  - Desenvolvimento de diagramas
  - Implementação do código
  - Teste do software
  - Encerramento do projeto
  - Carregamento do software para a web
  - Treinamento dos funcionários que utilizarão o software

5.2 GERENCIAMENTO DE TEMPO

Controle de tempo no projeto é a disciplina de gerenciamento que visa o controle da quantidade de tempo que leva a efetuação das tarefas componentes do trabalho no projeto. O guia PMBOK (A Guide to the Project Management Body of Knowledge) define uma forma particular do processo de gerenciamento de Tempo no Projeto. O guia aponta alguns processos chave para o se alcançar sucesso no gerenciamento de tempo, os quais serão esboçados em seguida.

Determinadas atividades de gerenciamento de cronograma buscam estabelecer todas as políticas, procedimentos e documentação necessários para o gerenciamento dos prazos do plano em si, bem como do desenvolvimento corrente, da sua execução e controle de prazos. Esse grupo de atividades resulta em um plano de gerenciamento de prazos.

O processo de definição e atividades identifica e documenta o que precisa ser realizado de forma a produzir os artefatos de entrega, ou seja, determina as tarefas do projeto. Trata-se de um processo crítico, visto que traça a linha guia do que se espera que aconteça para que o projeto seja funcional e eficaz, resultando numa lista definida de tarefas atômicas.

Por meio dessa lista de tarefas individuais, pode-se logo, definir uma priorização do que é mais premente e urgente de ser realizado. Somente após se ter uma clara ideia de tudo o que se necessita fazer é que se pode determinar as prioridades do projeto, em termos de tarefas. Essa categorização das tarefas em termos de prioridade colabora profundamente para que se possa fazer uso eficiente de recursos do projeto, bem como viabiliza sua entrega sua entrega tão logo seja possível. Nesse contexto, uma lista de dependências entre as atividades, como potenciais datas de início e fim das tarefas são formas eficazes de otimização.

Ademais, para garantir a otimização das ações que colaboram para a realização do projeto, é necessário se chegar a uma estimativa de recursos e tempo que se demandarão. É essencial ter um conhecimento preciso de quais recursos humanos, de equipamento e suprimentos serão utilizados para que cada tarefa se concretize, bem como as quantidades e outras informações relacionadas a cada uma delas. Igualmente importante é a compreensão de quanto tempo cada tarefa levará para ser concluída, dadas as suas necessidades e o contexto em que será executada.

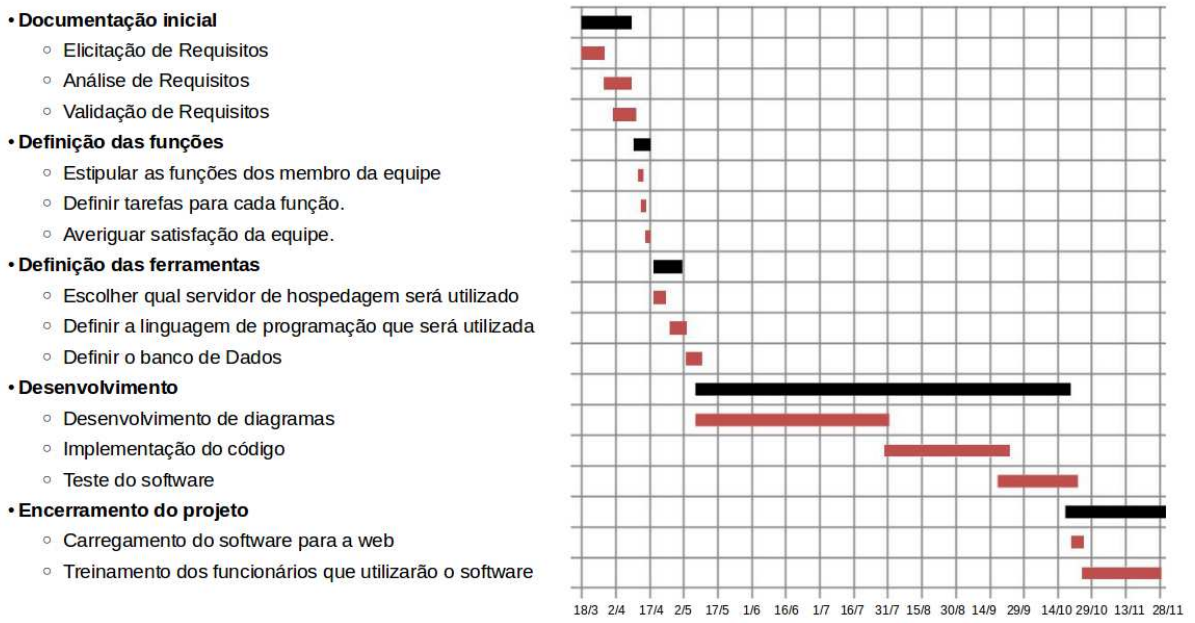
Por meio de todos esses processos mencionados se pode chegar a um cronograma de desenvolvimento, o que pode se demonstrar em um dos mais intrincados processos, conforme afirma o Guia PMBOK. O cronograma é um documento crítico para o gerenciamento do desempenho do projeto, de maneira que sua atualização e monitoramento são as ações que permitirão que se mantenha um controle to timing do projeto. Tanto a elaboração quanto a manutenção do cronograma são tarefas penosas e massivas, porém, garantem que se possa contar com uma ferramenta confiável de controle e avaliação da execução das atividades do projeto.

Uma das ferramentas para se gerar esse cronograma, bem como mantê-lo e adaptá-lo, é o diagrama de Grantt. Trata-se de um gráfico que representa o avanço de diversas atividades de um projeto em função do tempo. Os intervalos de tempo representando o início e fim de cada tarefa ou uma de suas subdivisões são codificados como barras sobre o eixo horizontal do gráfico. Sua representação permite que se visualize as tarefas, seus responsáveis e duração de tempo, bem como inferir o estado atual planejado de cada tarefa, o que permite uma análise de desempenho em termos de concretização da tarefa, bem como de engajamento

individual de cada responsável por uma tarefa específica.

No contexto do desenvolvimento do OfiSys, para a Oficina Chave de Rodas, levando em consideração o EAP apresentado anteriormente, uma possível representação do Diagrama de Grantt seria a seguinte:

Figura 25: Diagrama de Grantt



Fonte: Adaptado de http://www.joinville.udesc.br/portal/professores/claudinei/materiais/Exemplo_de_Plano_de_Software.docx

5.3 GERENCIAMENTO DE QUALIDADE

O gerenciamento de qualidade de um projeto contempla todos os processos necessários para determinar e se alcançar qualidade em um projeto. Sua função é permitir que o projeto tenha uma ideia precisa de quais são os parâmetros de qualidade a serem utilizados na sua execução, bem como garantir que essas exigências sejam satisfeitas.

Na execução do ciclo de vida do trabalho, o planejamento, a execução e o gerenciamento de projeto é uma parte integral do fluxo de trabalho. Dessa maneira, a área de conhecimento de gerenciamento de qualidade de projeto inclui os processos organizacionais que determinam políticas de qualidade, objetivos e

responsabilidades.

O guia PMBOK identifica três processos para o gerenciamento de qualidade, a saber: (1) o planejamento da qualidade, (2) a garantia de qualidade e (3) o controle de qualidade. Suas definições e aplicações podem ser postas da seguinte maneira:

5.3.1 Planejamento de qualidade

O planejamento de qualidade envolve a identificação dos requisitos de qualidade tanto para o projeto quanto para o produto, bem como a documentação de como se pode verificar a conformidade do projeto com os desses requisitos. Neste processo se determina uma definição clara das metas do projeto, de quais itens de entrega ele deve produzir, bem como se traçar estratégias para alcançar esse fim. Esses pontos incluem uma avaliação de riscos ao sucesso do projeto, estabelecimento de padrões e de métodos para se obter, controlar, prever e verificar o êxito no projeto

Esse processo resulta no Plano de Gerenciamento de Qualidade, métricas de qualidade, listas de verificação de qualidade e Plano de Aprimoramento do Processo.

5.3.2 Garantia de Qualidade

A garantia de qualidade é usada para assegurar que os processos do projeto estão ocorrendo de forma adequada, concorrendo rumo à efetiva entrega de um produto de boa qualidade. Possui métricas específicas para determinar se o projeto transcorre de maneira aceitável e adequada. A utilização tanto de métricas qualitativas quanto quantitativas é ajudarão a prever e assegurar a realização das metas, bem como identificar necessidades de ações corretivas, além de vincular métricas de qualidade a metas de qualidade, de forma que permita o relatório do estado corrente da qualidade do projeto. Listas de verificação e auditorias de projeto são dois métodos usados para aferir a garantia de qualidade em um projeto.

5.3.3 Controle de qualidade

A manutenção do controle de qualidade envolve técnicas operacionais

direcionadas à garantia de que os padrões de qualidade estejam sendo alcançados e à verificação da forma como o produto do projeto satisfaz os requisitos. Enquanto a garantia de qualidade tem caráter preventivo, atuando antes da emergência de problemas, o controle de qualidade é de natureza corretiva, se dando depois que um problema é identificado. Com isso monitora-se os itens produzidos por um projeto e analisa sua adequação quanto aos padrões aplicáveis, assim como se pode concretizar a identificação de fatores de risco e sua mitigação, além de buscar maneiras de prevenir e eliminar mau desempenho futuro. A identificação de quaisquer eventuais ações corretivas necessárias também é um ponto resultante da aplicação do controle de qualidade.

A realização de testes e revisão de trabalho por pares são dois dos métodos utilizados para a realização do controle de qualidade.

CONCLUSÃO

Após a realização do trabalho de planejamento e desenvolvimento do que foi solicitado, foi possível ganhar uma compreensão maior de todo o processo de desenvolvimento de software, em especial no que se refere a um ambiente Web, utilizando tecnologia Java. A exposição aos padrões de modelagem da UML, bem como aos conceitos da arquitetura MVC, garantiu uma visão *bottom up* holística do sistema, ao mesmo tempo em que tornou explícitos os detalhes de cada um dos módulos (modelo, visão e controle), suas características, seus papéis a serem desempenhados no sistema, assim como o escopo e extensão de sua atuação. Além disso, ganhou-se uma maior compreensão das interações entre as camadas da dita arquitetura, agindo em colaboração para fornecer uma solução para o problema proposto.

Como possibilidade de expansão, no lado do *frontend*, percebeu-se que, em vez de realizar as operações navegando entre diversas páginas dinamicamente geradas, há a opção de se utilizar de alguma tecnologia de alteração de página de forma assíncrona (como AJAX, por exemplo) ou algum framework que trate dessa questão de maneira transparente ao usuário. Isso reduziria o tráfego de dados na rede e garantiria uma experiência de usabilidade mais fluida, sem quebras por envio de informações e carregamento de novas páginas.

Em suma, a aplicação web até o momento desenvolvida, como todo e qualquer sistema computacional em sua primeira versão, funciona adequadamente, segundo as especificações e requisitos fornecidos, porém tem muito espaço para refinamento, ampliação, crescimento e aperfeiçoamento, os quais serão desenvolvidos, implementados e ajustados à medida que o feedback do usuário final, durante a fase de testes, chega à equipe de desenvolvedores. Não obstante, o processo de desenvolvimento foi de grande valia e satisfação, alcançando o propósito de aplicação dos conceitos aprendidos, bem como obtenção de experiência que apenas uma abordagem *hands on* poderia garantir.

REFERÊNCIAS

CARDOSO, Virgínia Mara. **Ferramentas para Sistemas WEB**. Londrina: Editora e Distribuidora Educacional S. A., 2017.

CEZAR, Douglas Fujita de Oliveira. **Banco de Dados II**. Londrina: Editora e Distribuidora Educacional S. A., 2017.

FABRIS, Polyanna Pacheco Gomes; CATARINO, Iolanda Cláudia Sanches. **Análise Orientada a Objetos II**. Londrina: Editora e Distribuidora Educacional S. A., 2017.

FABRIS, Polyanna Pacheco Gomes; PERINI, Luis Cláudio. **Processos de Software**. Londrina: Editora e Distribuidora Educacional S. A., 2014.

GONÇALVES, Anderson E. M.; SEPE, Adriano; TERRA, Paulo H. **Programação para Web II**. Londrina, Editora e Distribuidora Educacional S A., 2018.

MATOS, Maria Clotilde Pires. **Metodologia Científica**. Londrina: Editora e Distribuidora Educacional S. A., 2014.

MARKUP. **Markup Highlighter:** Online syntax highlighter like TextMate. Disponível em http://markup.su/highlighter/. Acessado em 11 de maio de 2018, às 17:03.

MONAPPA, Avantika. **Scope Management**: What It is and Why It's Important. Disponível em: https://www.simplilearn.com/project-scope-management-importance-rar89-article. Acessado em 15/05/2018, às 17:02.

MONTES, Eduardo. **Gerenciamento da qualidade do projeto**. Disponível em https://escritoriodeprojetos.com.br/gerenciamento-da-qualidade-do-projeto. Acessado em 16 de maio de 2018, às 04:47.

MOZER, Merris. **Sistemas WEB**. Londrina: Editora e Distribuidora Educacional S. A., 2014.

NISHIMURA, Roberto Yukio. **Banco de Dados I**. São Paulo, Pearson Prentice Hall, 2009.

NÓBREGA, Jarley P. **Plano de Desenvolvimento de Software**. Disponivel em http://www.joinville.udesc.br/portal/professores/claudinei/materiais/Exemplo_de_Plano_de_Software.docx. Acessado em 15/05/2018, às 11:21.

PROJECT MANAGEMENT SKILLS. Project Quality Management. Disponível em https://www.project-management-skills.com/project-quality-management.html. Acessado em16 de maio de 2018, às 04:59.

SOUCE CODE BEAUTIFIER. **Syntax highlighter**: convert snippets of code to html. Disponível em: http://hilite.me. Acessado em 10 de maio de 2018, às 04:43.

TANAKA, Simone Sawasaki. **Análise de sistemas I**. São Paulo, Pearson Prentice Hall, 2009.

TERRA, Paulo H.; GONÇALVES, Anderson E. M.; SEPE, Adriano. **Projeto Orientado a Objetos**. Londrina, Editora e Distribuidora Educacional S A., 2018.

UML DIAGRAMS. **UML 2.5 Diagrams Overview**. Disponível em: http://www.uml-diagrams.org/uml-25-diagrams.html. Acessado em 16 de abril de 2018, às 04:43.

VARNER, Marcus. The Basics of Project Quality Management. Disponível em https://resources.workfront.com/project-management-blog/the-basics-of-project-quality-management. Acessado em 16 de maio de 2018, às 04:57.

VENTURA, Plínio. **Caso de Uso – Include, Extend e Generalização.** Disponível em: http://www.ateomomento.com.br/caso-de-uso-include-extend-e-generalizacao. Acessado em 16 de abril de 2018, às 19:35.

APÊNDICES

APÊNDICE A

Código Fonte para Geração do Banco de Dados

```
drop database oficina;
create database oficina;

create table cliente (cliente_id int not null constraint cliente_pk primary key,
cliente_nome varchar(50) not null,
cliente_cpf varchar(11) unique not null,
cliente_cnpj varchar(15) unique,
cliente_logradouro varchar(40) not null,
cliente_numero int not null,
cliente_bairro varchar(20) not null,
cliente_cidade varchar(25) not null,
cliente_complemento varchar(25),
cliente_estado varchar(2) not null,
cliente_cep varchar(8) not null,
cliente_telefone varchar(14) not null,
cliente_email varchar(40),
cliente_observacoes text,
cliente_ativo boolean);

create table carro(carro_id int not null constraint carro_pk primary key,
carro_modelo varchar(20) not null unique,
carro_marca varchar(20) not null,
carro_ano varchar(4) not null,
carro_placa varchar(12) not null unique,
carro_cor varchar(15) not null,
carro_cliente_id int not null references cliente(cliente_id) on update cascade on delete set
null,
carro_ativo boolean);

create table atendente (atendente_matricula int not null constraint atendente_pk primary key,
atendente_nome varchar(50) not null,
atendente_login varchar(20) not null unique,
atendente_senha varchar(20) not null,
atendente_cpf varchar(11) not null unique,
atendente_logradouro varchar(40) not null,
atendente_numero int not null,
atendente_bairro varchar(20) not null,
atendente_cidade varchar(25) not null,
atendente_complemento varchar(25),
atendente_estado varchar(2) not null,
atendente_cep varchar(8) not null,
atendente_telefone varchar(14) not null,
atendente_email varchar(40),
atendente_observacoes text,
atendente_ativo boolean);

create table tecnico (tecnico_matricula int not null constraint tecnico_pk primary key,
tecnico_nome varchar(50) not null,
tecnico_login varchar(20) not null unique,
tecnico_senha varchar(20) not null,
tecnico_cpf varchar(11) not null unique,
tecnico_logradouro varchar(40) not null,
tecnico_numero int not null,
tecnico_bairro varchar(20) not null,
tecnico_cidade varchar(25) not null,
tecnico_complemento varchar(25),
tecnico_estado varchar(2) not null,
tecnico_cep varchar(8) not null,
tecnico_telefone varchar(14) not null,
tecnico_email varchar(40),
tecnico_observacoes text,
tecnico_ativo boolean);

create table especialidade (especialidade_codigo int not null constraint especialidade_pk
primary key,
especialidade_nome varchar(20) not null unique,
especialidade_salario_hora numeric not null,
especialidade_ativo boolean);
```

```
create table setor_oficina (setor_codigo int not null constraint setor_pk primary key,
setor_nome varchar(30) not null,
setor_tecnico_matricula int not null references tecnico(tecnico_matricula) on update cascade
on delete set null,
setor_ativo boolean);

create table mecanico (mecanico_matricula int not null constraint mecanico_pk primary key,
mecanico_nome varchar(50) not null,
mecanico_login varchar(20) not null unique,
mecanico_senha varchar(20) not null,
mecanico_cpf varchar(11) not null unique,
mecanico_logradouro varchar(40) not null,
mecanico_numero int not null,
mecanico_bairro varchar(20) not null,
mecanico_cidade varchar(25) not null,
mecanico_complemento varchar(25),
mecanico_estado varchar(2) not null,
mecanico_cep varchar(8) not null,
mecanico_telefone varchar(14) not null,
mecanico_email varchar(40),
mecanico_observacoes text,
mecanico_especialidade_codigo int references especialidade(especialidade_codigo) on update
cascade on delete set null,
mecanico_setor_codigo int references setor_oficina(setor_codigo) on update cascade on delete
set null,
mecanico_ativo boolean);

create table registro_atendimento(ra_numero int not null constraint ra_pk primary key,
ra_data_abertura date not null,
ra_data_encerramento date,
ra_descricao_abertura text not null,
ra_descricao_encerramento text,
ra_estado varchar(15),
ra_atendente_matricula int references atendente(atendente_matricula) on update cascade on
delete set null,
constraint ra_estado_chk check(ra_estado in ('Aberto', 'Em Andamento', 'Cancelado',
'Concluído')),
ra_cliente_id int references cliente(cliente_id) on update cascade on delete set null,
ra_carro_id int references carro(carro_id) on update cascade on delete set null,
ra_ativo boolean);

create table ordem_servico(os_numero int not null constraint os_pk primary key,
os_prioridade varchar(10),
os_data_abertura date not null,
os_data_encerramento date,
os_preco_servico numeric not null,
os_descricao_abertura text not null,
os_descricao_encerramento text,
os_estado varchar(15),
os_setor_atual int references setor_oficina(setor_codigo) on update cascade on delete set
null,
os_ra_numero int not null references registro_atendimento(ra_numero) on update cascade on
delete set null,
os_tecnico_encerramento int references tecnico(tecnico_matricula) on update cascade on delete
set null,
constraint os_prioridade_chk check(os_prioridade in ('Urgente', 'Alta', 'Mediana', 'Baixa',
'Prorrogavel')),
constraint os_estado_chk check(os_estado in ('Aberto', 'Em Andamento', 'Cancelado',
'Concluído')),
os_ativo boolean
);

create table tipo_servico(tipo_servico_codigo int not null constraint tipo_servico_pk primary
key,
tipo_servico_nome varchar(30) not null unique,
tipo_servico_duracao_estimada int not null,
tipo_servico_valor numeric not null,
tipo_servico_ativo boolean);

create table tarefa(tarefa_numero int not null constraint tarefa_pk primary key,
tarefa_os_numero int not null references ordem_servico(os_numero) on update cascade on delete
cascade,
tarefa_tipo_servico_codigo int references tipo_servico(tipo_servico_codigo) on update cascade
on delete set null,
tarefa_data_abertura date not null,
```

```
tarefa_data_encerramento date,
tarefa_descricao_abertura text not null,
tarefa_descricao_encerramento text,
tarefa_estado varchar(15),
constraint tarefa_estado_chk check(tarefa_estado in ('Aberto', 'Em Andamento', 'Cancelado',
'Concluído')),
tarefa_ativo boolean
);

create table item_estoque(item_estoque_codigo int not null constraint item_estoque_pk primary
key,
item_estoque_nome varchar(25) not null unique,
item_estoque_descricao text not null unique,
item_estoque_preco numeric not null,
item_estoque_quantidade_estoque int not null,
item_estoque_ativo boolean);

create table item_material(item_material_numero int not null,
item_material_tarefa_numero int not null references tarefa(tarefa_numero) on update cascade on
delete cascade,
item_material_quantidade numeric not null,
item_material_cobrado boolean,
constraint item_material_pk primary key(item_material_numero, item_material_tarefa_numero),
item_material_ativo boolean);

create table equipe_trabalho(equipe_trabalho_mecanico_matricula int not null references
mecanico(mecanico_matricula) on update cascade on delete cascade,
equipe_trabalho_tarefa_numero int not null references tarefa(tarefa_numero) on update cascade
on delete cascade,
constraint equipe_trabalho_pk primary key (equipe_trabalho_mecanico_matricula,
equipe_trabalho_tarefa_numero),
equipe_tranbalho_ativo boolean);

create table forma_pagamento(forma_pagamento_codigo int not null constraint forma_pagamento_pk
primary key,
forma_pagamento_nome varchar(20) unique,
forma_pagamento_ativo boolean);

create table pagamento(pagamento_numero int not null constraint pagamento_pk primary key,
pagamento_data_criacao date not null,
pagamento_data_conclusao date,
pagamento_estado varchar(20) not null,
pagamento_numero_nota int unique,
pagamento_forma_pagamento_codigo int references forma_pagamento(forma_pagamento_codigo) on
update cascade on delete set null,
pagamento_os_codigo int not null references ordem_servico(os_numero) on update cascade on
delete set null,
constraint pagamento_estado_chk check(pagamento_estado in ('Pendente', 'Pago', 'Cancelado')));

create sequence seq_cliente_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start 7
cache 2;
create sequence seq_carro_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start 7
cache 2;
create sequence seq_especialidade_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_atendente_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start
7 cache 2;
create sequence seq_tecnico_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start 7
cache 2;
create sequence seq_setor_oficina_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_mecanico_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start
7 cache 2;
create sequence seq_registro_atendimento_numero increment 7 minvalue 7 maxvalue
9223372036854775807 start 7 cache 2;
create sequence seq_ordem_servico_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_tipo_servico_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_tarefa_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start 7
cache 2;
create sequence seq_item_estoque_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_item_material_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
```

```
create sequence seq_forma_pagamento_codigo increment 7 minvalue 7 maxvalue 9223372036854775807
start 7 cache 2;
create sequence seq_pagamento_codigo increment 7 minvalue 7 maxvalue 9223372036854775807 start
7 cache 2;
```

APÊNDICE B

Código fonte das Classes Pessoa e Cliente

```
package br.com.oficina.modelo;
import br.com.oficina.utils.CNP;
import br.com.oficina.utils.Utils;

public class Pessoa {
    private int id;
    private String nome;
    private String cpf;
    private String logradouro;
    private int numeroCasa;
    private String bairro;
    private String cidade;
    private String complemento;
    private String estado;
    private String cep;
    private String telefone;
    private String email;
    private String observacoes;
    private boolean ativo;

    public Pessoa(int id, String nome, String cpf, String logradouro, int numero,
            String bairro, String cidade, String complemento, String estado, String cep,
            String telefone, String email,  String observacoes, boolean ativo) throws
InsercaoException{

        this.setId(id);
        this.setNome(nome);
        this.setCpf(cpf);
        this.setLogradouro(logradouro);
        this.setNumeroCasa(numero);
        this.setBairro(bairro);
        this.setCidade(cidade);
        this.setComplemento(complemento);
        this.setEstado(estado);
        this.setCep(cep);
        this.setTelefone(telefone);
        this.setEmail(email);
        this.setObservacoes(observacoes);
        this.setAtivo(ativo);
    }


    public int getId() {
        return id;
    }

    public void setId(int id) throws InsercaoException {
        if (id < 0){
            throw new InsercaoException("Codigo invalido: valor menor que zero");
        } else {
            this.id = id;
        }
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) throws InsercaoException {
        if (nome.length() > 50){
            throw new InsercaoException("Nome inválido: número de caracteres maior que 50");
        } else {
            this.nome = nome;
        }
    }

    public String getCpf() {
        return cpf;
    }
```

```
    public void setCpf(String cpf) throws InsercaoException {
        if (! CNP.isValidCPF(cpf)){
            throw new InsercaoException("CPF invalido");
        } else {
            this.cpf = cpf;
        }
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) throws InsercaoException {
        if (logradouro.length() > 40){
            throw new InsercaoException("Logradouro inválido: número de caracteres maior que
40");
        } else {
            this.logradouro = logradouro;
        }
    }

    public int getNumeroCasa() {
        return numeroCasa;
    }

    public void setNumeroCasa(int numeroCasa) throws InsercaoException {
        if (numeroCasa < 1){
            throw new InsercaoException("Número inválido: número negativo");
        } else {
            this.numeroCasa = numeroCasa;
        }
    }

    public String getBairro() {
        return bairro;
    }

    public void setBairro(String bairro) throws InsercaoException {
        if (bairro.length() > 20){
            throw new InsercaoException("Bairro inválido: número de caracteres maior que 20");
        } else {
            this.bairro = bairro;
        }
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) throws InsercaoException {
        if (cidade.length() > 25){
            throw new InsercaoException("Cidade inválida: número de caracteres maior que 25");
        } else {
            this.cidade = cidade;
        }
    }

    public String getComplemento() {
        return complemento;
    }

    public String getComplemento(boolean SQL) {
        if (SQL && this.complemento.equals("")){
            return "NULL";
        } else {
            return complemento;
        }
    }

    public void setComplemento(String complemento) throws InsercaoException {
        complemento = Utils.checaNull(complemento);
        if (complemento.length() > 25){
            throw new InsercaoException("Complemento inválido: número de caracteres maior que
25");
```

```
        } else {
            this.complemento = complemento;
        }
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) throws InsercaoException {
        if (estado.length() > 2){
            throw new InsercaoException("Estado inválido: número de caracteres maior que 2");
        } else {
            this.estado = estado;
        }
    }

    public String getCep() {
        return cep;
    }

    public void setCep(String cep) throws InsercaoException {
        try{
            boolean numeroNegativo = Integer.parseInt(cep) < 0;
            if (numeroNegativo){
                throw new InsercaoException("CEP inválido: conversão em número com valor
negativo");
            }
        } catch (NumberFormatException e){
            throw new InsercaoException("CEP inválido: detectada presença de caracteres não
numéricos" + e.getMessage());
        }
        if (cep.length() > 8){
            throw new InsercaoException("CEP inválido: numero de caracteres maior que 8");
        } else {
            this.cep = cep;
        }
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) throws InsercaoException {
        try{
            boolean numeroNegativo = Long.parseLong(telefone) < 0;
            if (numeroNegativo){
                throw new InsercaoException("Telefone inválido: conversão em número com valor
negativo");
            }
        } catch (NumberFormatException e){
            throw new InsercaoException("Telefone inválido: detectada presença de caracteres
não numéricos" + e.getMessage());
        }

        if (telefone.length() > 14){
            throw new InsercaoException("Telefone inválido: mais que 14 dígitos inseridos");
        } else {
            this.telefone = telefone;
        }
    }

    public String getEmail() {
        return email;
    }

    public String getEmail(boolean SQL) {
        if (SQL && this.email.equals("")){
            return "NULL";
        } else {
            return email;
        }
    }

    public void setEmail(String email) throws InsercaoException {
```

```
        email = Utils.checaNull(email);
        if (email == "" || email == null){
            this.email = email;
        } else if (email.indexOf('@') < 1 || email.indexOf('.') < 1){
            throw new InsercaoException("Email inválido: não há presença de @ ou de . no
endereço de email");
        } else if (email.length() > 40){
            throw new InsercaoException("Email inválido: mais que 40 dígitos inseridos");
        } else {
            this.email = email;
        }
    }

    public String getObservacoes() {
        return observacoes;
    }

    public String getObservacoes(boolean SQL) {
        if (SQL && this.observacoes == ""){
            return "NULL";
        } else {
            return observacoes;
        }
    }

    public void setObservacoes(String observacoes) throws InsercaoException {
        observacoes = Utils.checaNull(observacoes);
        if (observacoes.length() > 500){
            throw new InsercaoException("Valor de observações inválido: mais que 500 dígitos
inseridos");
        } else {
            this.observacoes = observacoes;
        }
    }

    public boolean isAtivo() {
        return ativo;
    }

    public void setAtivo(boolean ativo) {
        this.ativo = ativo;
    }
}
```

```
package br.com.oficina.modelo;

import br.com.oficina.utils.CNP;


public class Cliente extends Pessoa{
    private String cnpj;
    public Cliente(int id, String nome, String cpf, String cnpj, String logradouro,
            int numero, String bairro, String cidade, String complemento, String estado,
            String cep, String telefone, String email, String observacoes, boolean ativo)
throws InsercaoException {

        super(id, nome, cpf, logradouro, numero, bairro, cidade, complemento, estado, cep,
telefone, email, observacoes, ativo);
        this.setCnpj(cnpj);
    }

    public String getCnpj() {
        return cnpj;
    }

    public String getCnpj(boolean SQL) {
        if (SQL && this.getCnpj().equals("")){
            return "NULL";
        } else {
            return cnpj;
        }
    }
```

```
    public void setCnpj(String cnpj) throws InsercaoException {
        if (cnpj == null || cnpj.equals("")){
            this.cnpj = "";
        } else if (! CNP.isValidCNPJ(cnpj)){
            throw new InsercaoException("CNPJ invalido");
        } else {
            this.cnpj = cnpj;
        }
    }

}
```

APÊNDICE C

Código fonte da Classe ClienteDAO

```
package br.com.oficina.dao;

import br.com.oficina.modelo.Cliente;
import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import br.com.oficina.utils.Utils;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ClienteDAO extends DAO{

    public int inserir(String usuario, String senha, String endereco, Cliente cliente) {
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente. ClienteDAO.inserir()")
== null){
            return ClienteDAO.ERRO_CONEXAO;
        }

        try {
            String comandoSQL = "INSERT INTO cliente (cliente_id, cliente_nome, cliente_cpf,
cliente_cnpj, cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade,
cliente_complemento, cliente_estado, cliente_cep, cliente_telefone, cliente_email,
cliente_observacoes, cliente_ativo)
VALUES(NEXTVAL('seq_cliente_codigo'),?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            Utils.checaNull(ps,"PreparedStatementpara inserir cliente. ClienteDAO.inserir()");


            ps.setString(1, cliente.getNome());
            ps.setString(2, cliente.getCpf());
            if (cliente.getCnpj().equals("") || cliente.getCnpj() == null){
                ps.setNull(3, java.sql.Types.VARCHAR);
            } else {
                ps.setString(3, cliente.getCnpj());
            }
            ps.setString(4, cliente.getLogradouro());
            ps.setInt(5, cliente.getNumeroCasa());
            ps.setString(6, cliente.getBairro());
            ps.setString(7, cliente.getCidade());
            if (cliente.getComplemento().equals("") || cliente.getComplemento() == null){
                ps.setNull(8, java.sql.Types.LONGVARCHAR);
            } else {
                ps.setString(8, cliente.getComplemento());
            }
            ps.setString(9, cliente.getEstado());
            ps.setString(10, cliente.getCep());
            ps.setString(11, cliente.getTelefone());
            if (cliente.getEmail().equals("") || cliente.getEmail() == null){
                ps.setNull(12, java.sql.Types.VARCHAR);
            } else {
                ps.setString(12, cliente.getEmail());
            }
            ps.setString(13, cliente.getObservacoes(true));
            ps.setBoolean(14, cliente.isAtivo());
            ps.execute();
            conexao.close();
            return ClienteDAO.SUCESSO;

        } catch(SQLException e) {
            System.err.print(e.getMessage());
        }
        return 0;
    }
```

```
    public int alterar(String usuario, String senha, String endereco, Cliente cliente) {
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente. ClienteDAO.alterar()")
== null){
            return ClienteDAO.ERRO_CONEXAO;
        }

        try {
            String comandoSQL = "UPDATE cliente SET cliente_nome = ?, cliente_cpf = ?,
cliente_cnpj = ?, cliente_logradouro = ?, cliente_numero = ?, cliente_bairro = ?,
cliente_cidade = ?, cliente_complemento = ?, cliente_estado = ?, cliente_cep = ?,
cliente_telefone = ?, cliente_email = ?, cliente_observacoes = ?, cliente_ativo = ? WHERE
cliente_id = ?";

            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setString(1, cliente.getNome());
            ps.setString(2, cliente.getCpf());
            if (cliente.getCnpj().equals("") || cliente.getCnpj() == null){
                ps.setNull(3, java.sql.Types.VARCHAR);
            } else {
                ps.setString(3, cliente.getCnpj());
            }
            ps.setString(4, cliente.getLogradouro());
            ps.setInt(5, cliente.getNumeroCasa());
            ps.setString(6, cliente.getBairro());
            ps.setString(7, cliente.getCidade());
            if (cliente.getComplemento().equals("") || cliente.getComplemento() == null){
                ps.setNull(8, java.sql.Types.LONGVARCHAR);
            } else {
                ps.setString(8, cliente.getComplemento());
            }
            ps.setString(9, cliente.getEstado());
            ps.setString(10, cliente.getCep());
            ps.setString(11, cliente.getTelefone());
            if (cliente.getEmail().equals("") || cliente.getEmail() == null){
                ps.setNull(12, java.sql.Types.VARCHAR);
            } else {
                ps.setString(12, cliente.getEmail());
            }
            ps.setString(13, cliente.getObservacoes(true));
            ps.setBoolean(14, cliente.isAtivo());
            ps.setInt(15, cliente.getId());
            ps.executeUpdate();
            conexao.close();

            return ClienteDAO.SUCESSO;
        } catch (org.postgresql.util.PSQLException e){
            System.err.println("ERRO: Falha ao alterar cliente de codigo " + cliente.getId() +
". Violação de valor único\r\n " + e.getMessage());
            return ClienteDAO.ERRO_SQL;
        } catch (SQLException e) {
            System.err.println("ERRO: Falha ao alterar cliente de codigo " + cliente.getId() +
"\r\n " + e.getMessage());
            return ClienteDAO.ERRO_SQL;
        }
    }

    public int desativar(String usuario, String senha, String endereco, int idBusca) {
        Cliente cliente;
        try {
            cliente = this.buscarId(usuario, senha, endereco, idBusca);
            cliente.setAtivo(false);
            int codigoResultado = this.alterar(usuario, senha, endereco, cliente);
            return codigoResultado;

        } catch (EntidadeNulaException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
            return ClienteDAO.ERRO_CODIGO;
        }
    }

    public List<Cliente> consultar(String usuario, String senha, String endereco) throws
EntidadeNulaException {
        List<Cliente> listaCliente = new ArrayList<Cliente>();
```

```
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente.
ClienteDAO.consultar()") == null){
            throw new EntidadeNulaException("Connection nula consultar lista de clientes.
ClienteDAO.consultar()");
        }
        Cliente cliente = null;

        try {
            String comandoSQL = "SELECT cliente_id, cliente_nome, cliente_cpf, cliente_cnpj,
cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade, cliente_complemento,
cliente_estado, cliente_cep, cliente_telefone, cliente_email, cliente_observacoes,
cliente_ativo FROM cliente";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    cliente = new Cliente(rs.getInt("cliente_id"),
                                          rs.getString("cliente_nome"),
                                          rs.getString("cliente_cpf"),
                                          rs.getString("cliente_cnpj"),
                                          rs.getString("cliente_logradouro"),
                                          rs.getInt("cliente_numero"),
                                          rs.getString("cliente_bairro"),
                                          rs.getString("cliente_cidade"),
                                          rs.getString("cliente_complemento"),
                                          rs.getString("cliente_estado"),
                                          rs.getString("cliente_cep"),
                                          rs.getString("cliente_telefone"),
                                          rs.getString("cliente_email"),
                                          rs.getString("cliente_observacoes"),
                                          rs.getBoolean("cliente_ativo"));
                    listaCliente.add(cliente);
                }catch(InsercaoException e){
                    System.err.println("InsercaoException: Falha na busca por cliente:" +
"\r\n" + e.getMessage());
                }
                Utils.checaNull(cliente, "Cliente nulo: ClienteDAO.consultar()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(cliente, "Cliente") == null){
            throw new EntidadeNulaException("Cliente não encontrado no banco de dados");
        }
        return listaCliente;
    }

    public List<Cliente> consultar(String usuario, String senha, String endereco, boolean
mostraInativos) throws EntidadeNulaException {
        List<Cliente> listaCliente = new ArrayList<Cliente>();
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente.
ClienteDAO.consultar()") == null){
            throw new EntidadeNulaException("Connection nula consultar lista de clientes.
ClienteDAO.consultar()");
        }
        Cliente cliente = null;

        try {
            String comandoSQL = "";
            if (mostraInativos){
                comandoSQL = "SELECT cliente_id, cliente_nome, cliente_cpf, cliente_cnpj,
cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade, cliente_complemento,
cliente_estado, cliente_cep, cliente_telefone, cliente_email, cliente_observacoes,
cliente_ativo FROM cliente";
            } else {
                comandoSQL = "SELECT cliente_id, cliente_nome, cliente_cpf, cliente_cnpj,
cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade, cliente_complemento,
cliente_estado, cliente_cep, cliente_telefone, cliente_email, cliente_observacoes,
cliente_ativo FROM cliente WHERE cliente_ativo IS TRUE";
            }
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
```

```
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    cliente = new Cliente(rs.getInt("cliente_id"),
                                          rs.getString("cliente_nome"),
                                          rs.getString("cliente_cpf"),
                                          rs.getString("cliente_cnpj"),
                                          rs.getString("cliente_logradouro"),
                                          rs.getInt("cliente_numero"),
                                          rs.getString("cliente_bairro"),
                                          rs.getString("cliente_cidade"),
                                          rs.getString("cliente_complemento"),
                                          rs.getString("cliente_estado"),
                                          rs.getString("cliente_cep"),
                                          rs.getString("cliente_telefone"),
                                          rs.getString("cliente_email"),
                                          rs.getString("cliente_observacoes"),
                                          rs.getBoolean("cliente_ativo"));
                    listaCliente.add(cliente);
                }catch(InsercaoException e){
                    System.err.println("InsercaoException: Falha na busca por cliente:" +
"\r\n" + e.getMessage());
                }
                Utils.checaNull(cliente, "Cliente nulo: ClienteDAO.consultar()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(cliente, "Cliente") == null){
            throw new EntidadeNulaException("Cliente não encontrado no banco de dados");
        }
        return listaCliente;
    }

    public Cliente buscarId(String usuario, String senha, String endereco, int idBusca) throws
EntidadeNulaException {
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente. ClienteDAO.buscarId()")
== null){
            throw new EntidadeNulaException("Connection nula para inserir cliente.
ClienteDAO.buscar()");
        }
        Cliente cliente = null;

        try {
            String comandoSQL = "SELECT cliente_id, cliente_nome, cliente_cpf, cliente_cnpj,
cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade, cliente_complemento,
cliente_estado, cliente_cep, cliente_telefone, cliente_email, cliente_observacoes,
cliente_ativo FROM cliente WHERE cliente_id = ?";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setInt(1, idBusca);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    cliente = new Cliente(rs.getInt("cliente_id"),
                                          rs.getString("cliente_nome"),
                                          rs.getString("cliente_cpf"),
                                          rs.getString("cliente_cnpj"),
                                          rs.getString("cliente_logradouro"),
                                          rs.getInt("cliente_numero"),
                                          rs.getString("cliente_bairro"),
                                          rs.getString("cliente_cidade"),
                                          rs.getString("cliente_complemento"),
                                          rs.getString("cliente_estado"),
                                          rs.getString("cliente_cep"),
                                          rs.getString("cliente_telefone"),
                                          rs.getString("cliente_email"),
                                          rs.getString("cliente_observacoes"),
                                          rs.getBoolean("cliente_ativo"));
                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por cliente:" + "\r\n" +
e.getMessage());
                }
```

```
                Utils.checaNull(cliente, "Cliente nulo: ClienteDAO.buscarId()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(cliente, "Cliente") == null){
            throw new EntidadeNulaException("Cliente não encontrado no banco de dados");
        }
        return cliente;
    }

    public Cliente buscarCPF(String usuario, String senha, String endereco, String cpf) throws
EntidadeNulaException {
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente.
ClienteDAO.buscarCPF()") == null){
            throw new EntidadeNulaException("Connection nula para inserir cliente.
ClienteDAO.buscar()");
        }
        Cliente cliente = null;

        try {
            String comandoSQL = "SELECT cliente_id, cliente_nome, cliente_cpf, cliente_cnpj,
cliente_logradouro, cliente_numero, cliente_bairro, cliente_cidade, cliente_complemento,
cliente_estado, cliente_cep, cliente_telefone, cliente_email, cliente_observacoes,
cliente_ativo FROM cliente WHERE cliente_cpf = ?";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setString(1, cpf);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    cliente = new Cliente(rs.getInt("cliente_id"),
                                          rs.getString("cliente_nome"),
                                          rs.getString("cliente_cpf"),
                                          rs.getString("cliente_cnpj"),
                                          rs.getString("cliente_logradouro"),
                                          rs.getInt("cliente_numero"),
                                          rs.getString("cliente_bairro"),
                                          rs.getString("cliente_cidade"),
                                          rs.getString("cliente_complemento"),
                                          rs.getString("cliente_estado"),
                                          rs.getString("cliente_cep"),
                                          rs.getString("cliente_telefone"),
                                          rs.getString("cliente_email"),
                                          rs.getString("cliente_observacoes"),
                                          rs.getBoolean("cliente_ativo"));
                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por cliente:" + "\r\n" +
e.getMessage());
                }
                Utils.checaNull(cliente, "Cliente nulo: ClienteDAO.buscarCPF()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(cliente, "Cliente") == null){
            throw new EntidadeNulaException("Cliente não encontrado no banco de dados");
        }
        return cliente;
    }

}
```

APÊNDICE D

Código Fonte da Classe ControleCliente

```
package br.com.oficina.controle;

import br.com.oficina.dao.ClienteDAO;
import br.com.oficina.modelo.Cliente;
import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import java.io.IOException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ControleCliente extends Controle implements InterfaceControle{

    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String acao = request.getParameter("acao");
        int resultadoOperacao = 20;
        if (acao==null) acao = "";

        if (acao.equals("incluir")){
            resultadoOperacao = ControleCliente.INSERIR;


        } else if (acao.equals("inserir")){
            resultadoOperacao = inserir(request, response);


        } else if (acao.equals("alterar")){
            resultadoOperacao = alterar(request, response);


        }else if (acao.equals("atualizar")){
            try {
                Cliente cliente = buscar(request, response);
                request.setAttribute("cliente", cliente);
                resultadoOperacao = ControleCliente.ATUALIZAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = ClienteDAO.ERRO_CODIGO;
            }


        } else if (acao.equals("desativar")){
            resultadoOperacao = desativar(request, response);


        } else if (acao.equals("consultar")){
            try {
                List<Cliente> lista = consultar(request, response);
                request.setAttribute("listaClientes", lista);
                resultadoOperacao = this.LISTAR;
            } catch (EntidadeNulaException ex) {
                Logger.getLogger(ControleCliente.class.getName()).log(Level.SEVERE, null, ex);
                resultadoOperacao = ClienteDAO.ERRO_CONEXAO;
            }


        } else if (acao.equals("buscar")){
            resultadoOperacao = this.BUSCAR;


        } else if (acao.equals("detalhar")){
            try {
                Cliente cliente = buscar(request, response);
```

```
                request.setAttribute("cliente", cliente);
                resultadoOperacao = this.DETALHAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = ClienteDAO.ERRO_CODIGO;
            }
        }

        RequestDispatcher rd;
        request.setAttribute("title","Cliente - " + acao);
        switch (resultadoOperacao){
            case INSERIR:
                rd = request.getRequestDispatcher("inserirCliente.jsp");
                break;
            case ClienteDAO.SUCESSO:
                String processadas = " processadas ";
                if (acao.equals("inserir")){
                    processadas = " inseridas ";
                } else if (acao.equals("excluir")){
                    processadas = " excluÃdas ";
                } else {
                    processadas = " alteradas ";
                }
                request.setAttribute("conteudo","<h1>InformaÃ§Ãµes sobre o cliente" +
processadas + "com sucesso</h1>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case ClienteDAO.ERRO_SQL:
                request.setAttribute("conteudo","<h1>Falha ao Acessar o Banco</h1>"
                                    + "<p>Erro durante o acesso (leitura ou escrita) ao banco
de dados. Comando SQL malformado. Procure o Webmaster</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case ClienteDAO.ERRO_INSERCAO:
                request.setAttribute("conteudo","<h1>Falha na OperaÃ§Ã£o</h1>"
                                                + "<p>Os dados inseridos eram invÃ¡lidos. Por
favor verifcar e tentar outra vez</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case ClienteDAO.ERRO_CODIGO:
                request.setAttribute("conteudo","<h1>Falha na OperaÃ§Ã£o</h1>"
                                                + "<p>O CÃ³digo informado nao corresponde a um
cÃ³digo de cliente presente no banco de dados</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case LISTAR:
                rd = request.getRequestDispatcher("listarClientes.jsp");
                break;
            case BUSCAR:
                rd = request.getRequestDispatcher("buscarCliente.jsp");
                break;
            case DETALHAR:
                rd = request.getRequestDispatcher("detalharCliente.jsp");
                break;
            case ATUALIZAR:
                rd = request.getRequestDispatcher("atualizarCliente.jsp");
                break;
            default:
//                request.setAttribute("conteudo","<h1>Falha na OperaÃ§Ã£o</h1>"
//                                                + "<p>Por algum motivo nÃ£o identificado, a
operaÃ§Ã£o foi cancelada. Entre em contato com o webmaster</p>");
//                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");

                rd = request.getRequestDispatcher("buscarCliente.jsp");
        }

        rd.forward(request, response);
    }

    public int alterar(HttpServletRequest request, HttpServletResponse response){
        try {
            ClienteDAO clienteDAO = new ClienteDAO();
            Cliente cliente = new
Cliente(Integer.parseInt(request.getParameter("cliente_codigo")),
                                        request.getParameter("cliente_nome"),
                                        request.getParameter("cliente_cpf"),
```

```
                                        request.getParameter("cliente_cnpj"),
                                        request.getParameter("cliente_logradouro"),
Integer.parseInt(request.getParameter("cliente_numero")),
                                        request.getParameter("cliente_bairro"),
                                        request.getParameter("cliente_cidade"),
                                        request.getParameter("cliente_complemento"),
                                        request.getParameter("cliente_estado"),
                                        request.getParameter("cliente_cep"),
                                        request.getParameter("cliente_telefone"),
                                        request.getParameter("cliente_email"),
                                        request.getParameter("cliente_observacoes"),
                                        true);
            int resultadoOperacao = clienteDAO.alterar(this.usuario, this.senha,
this.endereco, cliente);
            return resultadoOperacao;
        } catch (InsercaoException ex) {
            //tratamento de falha na insercao
            return ClienteDAO.ERRO_INSERCAO;
        }
    }

    //VERIFICAR OS MÉTODOS de BUSCAR
    public Cliente buscar(HttpServletRequest request, HttpServletResponse response) throws
EntidadeNulaException{return null;}

    public Cliente buscarCPF(HttpServletRequest request, HttpServletResponse response){return
null;}

    public Cliente buscarId(HttpServletRequest request, HttpServletResponse response){return
null;}

    public List<Cliente> consultar(HttpServletRequest request, HttpServletResponse response)
throws EntidadeNulaException{
        ClienteDAO clienteDAO = new ClienteDAO();
        return clienteDAO.consultar(this.usuario, this.senha, this.endereco);
    }

    public int desativar(HttpServletRequest request, HttpServletResponse response){
        ClienteDAO clienteDAO = new ClienteDAO();
        int clienteCodigo = Integer.parseInt(request.getParameter("cliente_codigo"));
        int resultadoOperacao = clienteDAO.desativar(this.usuario, this.senha, this.endereco,
clienteCodigo);
        return resultadoOperacao;
    }

    public int inserir(HttpServletRequest request, HttpServletResponse response){
        int resultadoOperacao = 0;
        try {
            ClienteDAO clienteDAO = new ClienteDAO();
            Cliente cliente = new Cliente(0,
                                        request.getParameter("cliente_nome"),
                                        request.getParameter("cliente_cpf"),
                                        request.getParameter("cliente_cnpj"),
                                        request.getParameter("cliente_logradouro"),
Integer.parseInt(request.getParameter("cliente_numero")),
                                        request.getParameter("cliente_bairro"),
                                        request.getParameter("cliente_cidade"),
                                        request.getParameter("cliente_complemento"),
                                        request.getParameter("cliente_estado"),
                                        request.getParameter("cliente_cep"),
                                        request.getParameter("cliente_telefone"),
                                        request.getParameter("cliente_email"),
                                        request.getParameter("cliente_observacoes"),
                                        true);
            resultadoOperacao = clienteDAO.inserir(this.usuario, this.senha, this.endereco,
cliente);
            return resultadoOperacao;
        } catch (InsercaoException ex) {
            //tratamento de falha na insercao
            return ClienteDAO.ERRO_INSERCAO;
        }
    }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    //</editor-fold>
}
```

APÊNDICE E

Código Fonte das JavaServer Pages de Manipulação de Clientes

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente - Inserir</title>
        <style type="text/css">
            .container {
                width: 500px;
                clear: both;
            }
            .container input {
                width: 100%;
                clear: both;
            }
        </style>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Novo Cliente</h1>
        <form action="ControleCliente" method="post">
            <div class="container">
                <input type="hidden" name="acao" value="inserir"/>
                <label for="cliente_nome">Nome</label>
                <input type="text" name="cliente_nome" maxlength="35"/><br/>
                <label for="cliente_cpf">CPF</label>
                <input type="text" name="cliente_cpf" maxlength="11"/><br/>
                <label for="cliente_cnpj">CNPJ</label>
                <input type="text" name="cliente_cnpj" maxlength="15"/><br/>
                <label for="cliente_logradouro">Logradouro</label>
                <input type="text" name="cliente_logradouro" maxlength="50"/><br/>
                <label for="cliente_numero_endereco">Número</label>
                <input type="text" name="cliente_numero_endereco"/><br/>
                <label for="cliente_bairro">Bairro</label>
                <input type="text" name="cliente_bairro" maxlength="50"/><br/>
                <label for="cliente_complemento">Complemento</label>
                <input type="text" name="cliente_complemento" maxlength="250"/><br/>
                <label for="cliente_cidade">Cidade</label>
                <input type="text" name="cliente_cidade" maxlength="50"/><br/>
                <label for="cliente_estado">Estado</label>
                <input type="text" name="cliente_estado" maxlength="50"/><br/>
                <label for="cliente_cep">CEP</label>
                <input type="text" name="cliente_cep" maxlength="8"/><br/>

                <label for="cliente_fone">Telefone</label>
                <input type="text" name="cliente_fone" maxlength="11"/><br/>
                <label for="cliente_email">Email</label>
                <input type="text" name="cliente_email" maxlength="11"/><br/>
                <label for="cliente_observacoes">Observações</label>
                <input type="text" name="cliente_observacoes" maxlength="11"/><br/>
            </div>
            <input class="bot_envio" type="submit" value="Incluir Cliente"/>
            <input class="bot_cancela" type="button" value ="Cancelar"
onClick="window.location='<%=request.getContextPath() %>/home.jsp';"/><br/>
        </form>
    </body>
</html>

<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente - Atualizar</title>
        <style type="text/css">
            .container {
                width: 500px;
```

```
                clear: both;
            }
            .container input {
                width: 100%;
                clear: both;
            }
        </style>

        <jsp:include page="/cabecalho.jsp"/>

        <%Cliente cliente = (Cliente) request.getAttribute("cliente");%>
        <h1>Alterar Dados do Cliente</h1>
        <form action="ControleCliente" method="post">
            <div class="container">
                <input type="hidden" name="acao" value="alterar"/>
                <label for="cliente_codigo">Codigo</label>
                <input type="number" name="cliente_codigo" readonly="true" min="1" value="<%=
cliente.getId() %>"/><br/>
                <label for="cliente_nome">Nome</label>
                <input type="text" name="cliente_nome" maxlength="35" value="<%= cliente.getNome()
%>"/><br/>
                <label for="cliente_cpf">CPF</label>
                <input type="text" name="cliente_cpf" maxlength="11" value="<%= cliente.getCpf()
%>"/><br/>
                <label for="cliente_cnpj">CNPJ</label>
                <input type="text" name="cliente_cnpj" maxlength="15" value="<%= cliente.getCnpj()
%>"/><br/>
                <label for="cliente_logradouro">Logradouro</label>
                <input type="text" name="cliente_logradouro" maxlength="50" value="<%=
cliente.getLogradouro() %>"/><br/>
                <label for="cliente_numero_endereco">Número</label>
                <input type="number" name="cliente_numero_endereco" min="1" value="<%=
cliente.getNumeroCasa() %>"/><br/>
                <label for="cliente_bairro">Bairro</label>
                <input type="text" name="cliente_bairro" maxlength="50" value="<%= cliente.getBairro()
%>"/><br/>
                <label for="cliente_complemento">Complemento</label>
                <input type="text" name="cliente_complemento" maxlength="250" value="<%=
cliente.getComplemento() %>"/><br/>
                <label for="cliente_cidade">Cidade</label>
                <input type="text" name="cliente_cidade" maxlength="50" value="<%= cliente.getCidade()
%>"/><br/>
                <label for="cliente_estado">Estado</label>
                <input type="text" name="cliente_estado" maxlength="50" value="<%= cliente.getEstado()
%>"/><br/>
                <label for="cliente_cep">CEP</label>
                <input type="text" name="cliente_cep" maxlength="8" value="<%= cliente.getCep()
%>"/><br/>
                <label for="cliente_fone">Telefone</label>
                <input type="tel" name="cliente_fone" maxlength="11" value="<%= cliente.getTelefone()
%>"/><br/>
                <label for="cliente_email">Email</label>
                <input type="email" name="cliente_email" value="<%= cliente.getEmail() %>"/><br/>
                <label for="cliente_observacoes">Observações</label>
                <input type="text" name="cliente_observacoes" maxlength="11" value="<%=
cliente.getObservacoes() %>"/><br/>
                <input class="bot_envio" type="submit" value="Alterar Dados"/><br/><br/>
            </div>
        </form>
    </body>
</html>

<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente - Detalhar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Detalhes do Cliente</h1>
```

```
        <%Cliente cliente = (Cliente) request.getAttribute("cliente");%>

        <p><b>Código: </b><%= cliente.getId() %>"</p>
        <p><b>Nome: </b><%= cliente.getNome() %>"</p>
        <p><b>Cpf: </b><%= cliente.getCpf() %>"</p>
        <p><b>Cnpj: </b><%= cliente.getCnpj() %>"</p>
        <p><b>Logradouro: </b><%= cliente.getLogradouro() %>"</p>
        <p><b>Número: </b><%= cliente.getNumeroCasa() %>"</p>
        <p><b>Bairro: </b><%= cliente.getBairro() %>"</p>
        <p><b>Complemento: </b><%= cliente.getComplemento() %></p>
        <p><b>Cidade: </b><%= cliente.getCidade() %>"</p>
        <p><b>Estado: </b><%= cliente.getEstado() %>"</p>
        <p><b>CEP: </b><%= cliente.getCep() %>"</p>
        <p><b>Telefone: </b><%= cliente.getTelefone() %>"</p>
        <p><b>Email: </b><%= cliente.getEmail() %>"</p>
        <p><b>Observações: </b><%= cliente.getObservacoes() %>"</p>

        <a class="bot_excluir" href="ControleCliente?acao=desativar&cliente_codigo=<%= cliente.getId()
%>" onclick="return confirm('Deseja realmente realizar a exclusão do registro?')">Desativar Cliente</a>
        <a class="bot_detalhar" href="ControleCliente?acao=atualizar&cliente_codigo=<%= cliente.getId()
%>">Atualizar Cliente</a><br/><br/>

    </body>
</html>

<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ page import="java.util.List" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente - Listar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Lista de Clientes Registrados</h1>

        <table border="1">
            <tr>
                <th>Código</th>
                <th>Nome</th>
                <th>CPF</th>
                <th>CNPJ</th>
                <th colspan="2">Ações</th>
            </tr>
            <%
                List<Cliente> lista = (List<Cliente>) request.getAttribute("listaClientes");
                for (Cliente cliente : lista) {%>
                    <tr>
                        <td><%= cliente.getId() %></td>
                        <td>
                            <a href="ControleCliente?acao=detalhar&cliente_codigo=<%= cliente.getId()
%>">
                                <%= cliente.getNome() %>
                            </a>
                        </td>
                        <td><%= cliente.getCpf() %></td>
                        <td><%= cliente.getCnpj() %></td>
                        <td>
                            <a href="ControleCliente?acao=atualizar&cliente_codigo=<%= cliente.getId()
%>">
                                Atualizar
                            </a>
                        </td>
                        <td>
                            <a href="ControleCliente?acao=excluir&cliente_codigo=<%= cliente.getId()
%>" onclick="return confirm('Deseja realmente realizar a desativação do registro do registro?')">
                                Desativar
                            </a>
                        </td>
                    </tr>
                <%}
```

```
            %>
            </table>

    </body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente - Buscar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Buscar Cliente pelo Código</h1>
        <form action="ControleCliente" method="post">
            <input type="hidden" name="acao" value="detalhar"/>
            <label for="cliente_codigo"><b>Código do Cliente: </b></label>
            <input type="number" name="cliente_codigo" max="2147483648"/>
            <input class="bot_envio" type="submit" value="Buscar"/>
        </form>
    </body>
</html>
```

APÊNDICE F

Código Fonte da Classe TipoServico

```
package br.com.oficina.modelo;

public class TipoServico {

    private int codigo;
    private String nome;
    private int duracao; //em dias
    private double valor;
    private boolean ativo;

    public TipoServico(int codigo, String nome, int duracao, double valor) throws
InsercaoException{
        this.setCodigo(codigo);
        this.setNome(nome);
        this.setDuracao(duracao);
        this.setValor(valor);
        this.setAtivo(true);
    }

    public TipoServico(int codigo, String nome, int duracao, double valor, boolean ativo)
throws InsercaoException{
        this.setCodigo(codigo);
        this.setNome(nome);
        this.setDuracao(duracao);
        this.setValor(valor);
        this.setAtivo(ativo);
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) throws InsercaoException {
        if (codigo < 0){
            throw new InsercaoException();
        } else {
            this.codigo = codigo;
        }
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getDuracao() {
        return duracao;
    }

    public void setDuracao(int duracao) {
        this.duracao = duracao;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

    public boolean isAtivo() {
        return ativo;
    }

    public void setAtivo(boolean ativo) {
```

```
        this.ativo = ativo;
    }
}
```

APÊNDICE G

Código Fonte da Classe TipoServicoDAO

```
package br.com.oficina.dao;

import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import br.com.oficina.modelo.TipoServico;
import br.com.oficina.utils.Utils;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TipoServicoDAO extends DAO{

    public int inserir(String usuario, String senha, String endereco, TipoServico
tipoServico){
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir tipo de serviço.
TipoServicoDAO.inserir()") == null){
            return TipoServicoDAO.ERRO_CONEXAO;
        }

        try {
            String comandoSQL = "INSERT INTO tipo_servico (tipo_servico_codigo,
tipo_servico_nome, tipo_servico_duracao_estimada, tipo_servico_valor, tipo_servico_ativo)
VALUES(NEXTVAL('seq_tipo_servico_codigo'),?,?,?,?)";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            Utils.checaNull(ps,"PreparedStatement para inserir tipo de servico.
TipoServicoDAO.inserir()");


            ps.setString(1, tipoServico.getNome());
            ps.setInt(2, tipoServico.getDuracao());
            ps.setDouble(3, tipoServico.getValor());
            ps.setBoolean(4, tipoServico.isAtivo());
            ps.execute();
            conexao.close();
            return TipoServicoDAO.SUCESSO;

        } catch(SQLException e) {
            System.err.print(e.getMessage());
        }
        return TipoServicoDAO.SUCESSO;
    }

    public int alterar(String usuario, String senha, String endereco, TipoServico
tipoServico){
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para alterar tipo de servico.
TipoServicoDAO.alterar()") == null){
            return TipoServicoDAO.ERRO_CONEXAO;
            //tipo_servico_codigo = ?, tipo_servico_nome = ?, tipo_servico_duracao_estimada =
?, tipo_servico_valor = ?, tipo_servico_ativo = ?,
        }

        try {
            String comandoSQL = "UPDATE tipo_servico SET tipo_servico_nome = ?,
tipo_servico_duracao_estimada = ?, tipo_servico_valor = ?, tipo_servico_ativo = ? WHERE
tipo_servico_codigo = ?";

            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setString(1, tipoServico.getNome());
            ps.setInt(2, tipoServico.getDuracao());
            ps.setDouble(3, tipoServico.getValor());
            ps.setBoolean(4, tipoServico.isAtivo());
            ps.setInt(5, tipoServico.getCodigo());
```

```
            ps.executeUpdate();
            conexao.close();

            return TipoServicoDAO.SUCESSO;
        } catch (org.postgresql.util.PSQLException e){
            System.err.println("ERRO: Falha ao alterar Tipo de Serviço de codigo " +
tipoServico.getCodigo() + ". Violação de valor único\r\n " + e.getMessage());
            return TipoServicoDAO.ERRO_SQL;
        } catch (SQLException e) {
            System.err.println("ERRO: Falha ao alterar Tipo de Serviço de codigo " +
tipoServico.getCodigo() + "\r\n " + e.getMessage());
            return ClienteDAO.ERRO_SQL;
        }
    }

    public int desativar(String usuario, String senha, String endereco, int codBusca){
        TipoServico tipoServico;
        try {
            tipoServico = this.buscar(usuario, senha, endereco, codBusca);
            tipoServico.setAtivo(false);
            int codResultado = this.alterar(usuario, senha, endereco, tipoServico);
            return codResultado;
        } catch (EntidadeNulaException ex) {
            Logger.getLogger(TipoServicoDAO.class.getName()).log(Level.SEVERE, null, ex);
            return TipoServicoDAO.ERRO_CODIGO;
        }
    }

    public List<TipoServico> consultar(String usuario, String senha, String endereco) throws
EntidadeNulaException{
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para consultar Tipo de Serviço.
TipoServicoDAO.consultar()") == null){
            throw new EntidadeNulaException("Connection nula para inserir Tipo de Serviço.
TipoServicoDAO.consultar()");
        }

        List listaTipoServicos = new ArrayList();
        TipoServico tipoServico = null;

        try {
            String comandoSQL = "SELECT tipo_servico_codigo, tipo_servico_nome,
tipo_servico_duracao_estimada, tipo_servico_valor, tipo_servico_ativo FROM tipo_servico";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    tipoServico = new TipoServico(rs.getInt("tipo_servico_codigo"),
                                                  rs.getString("tipo_servico_nome"),
                                                  rs.getInt("tipo_servico_duracao_estimada"),
                                                  rs.getDouble("tipo_servico_valor"),
                                                  rs.getBoolean("tipo_servico_ativo"));
                    listaTipoServicos.add(tipoServico);
                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por tipo de Serviço:" + "\r\n" +
e.getMessage());
                }
                Utils.checaNull(tipoServico, "TipoServico nulo: TipoServicoDAO.buscarId()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(tipoServico, "Cliente") == null){
            throw new EntidadeNulaException("Cliente não encontrado no banco de dados");
        }
        return listaTipoServicos;
    }

    public List<TipoServico> consultar(String usuario, String senha, String endereco, boolean
mostraInativos) throws EntidadeNulaException{
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para consultar Tipo de Serviço.
TipoServicoDAO.consultar()") == null){
```

```
            throw new EntidadeNulaException("Connection nula para inserir Tipo de Serviço.
TipoServicoDAO.buscarId()");
        }

        List listaTipoServicos = new ArrayList();
        TipoServico tipoServico = null;

        try {
            String comandoSQL;
            if (mostraInativos){
                comandoSQL = "SELECT tipo_servico_codigo, tipo_servico_nome,
tipo_servico_duracao_estimada, tipo_servico_valor, tipo_servico_ativo FROM tipo_servico";
            } else {
                comandoSQL = "SELECT tipo_servico_codigo, tipo_servico_nome,
tipo_servico_duracao_estimada, tipo_servico_valor, tipo_servico_ativo FROM tipo_servico WHERE
tipo_servico_ativo IS TRUE";
            }
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    tipoServico = new TipoServico(rs.getInt("tipo_servico_codigo"),
                                                  rs.getString("tipo_servico_nome"),
                                                  rs.getInt("tipo_servico_duracao_estimada"),
                                                  rs.getDouble("tipo_servico_valor"),
                                                  rs.getBoolean("tipo_servico_ativo"));
                    listaTipoServicos.add(tipoServico);
                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por tipo de Serviço:" + "\r\n" +
e.getMessage());
                }
                Utils.checaNull(tipoServico, "TipoServico nulo: TipoServicoDAO.buscarId()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(tipoServico, "TipoServico") == null){
            throw new EntidadeNulaException("Tipo de Serviço não encontrado no banco de
dados");
        }
        return listaTipoServicos;
    }

    public TipoServico buscar(String usuario, String senha, String endereco, int codBusca)
throws EntidadeNulaException{
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir Tipo de Serviço.
TipoServicoDAO.buscarId()") == null){
            throw new EntidadeNulaException("Connection nula para inserir Tipo de Serviço.
TipoServicoDAO.buscarId()");
        }

        TipoServico tipoServico = null;

        try {
            String comandoSQL = "SELECT tipo_servico_codigo, tipo_servico_nome,
tipo_servico_duracao_estimada, tipo_servico_valor, tipo_servico_ativo FROM tipo_servico WHERE
tipo_servico_codigo = ?";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setInt(1, codBusca);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    tipoServico = new TipoServico(rs.getInt("tipo_servico_codigo"),
                                                  rs.getString("tipo_servico_nome"),
                                                  rs.getInt("tipo_servico_duracao_estimada"),
                                                  rs.getDouble("tipo_servico_valor"),
                                                  rs.getBoolean("tipo_servico_ativo"));
                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por tipo de Serviço:" + "\r\n" +
e.getMessage());
                }
                Utils.checaNull(tipoServico, "TipoServico nulo: TipoServicoDAO.buscarId()");
```

```
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(tipoServico, "TipoServico") == null){
            throw new EntidadeNulaException("Tipo de Serviço não encontrado no banco de
dados");
        }
        return tipoServico;
    }

}
```

APÊNDICE H

Código Fonte da Classe ControleTipoServico

```
package br.com.oficina.controle;

import br.com.oficina.dao.TipoServicoDAO;
import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import br.com.oficina.modelo.TipoServico;
import java.io.IOException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author saintclair
 */
public class ControleTipoServico extends Controle implements InterfaceControle{

    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String acao = request.getParameter("acao");
        int resultadoOperacao = 20;
        if (acao==null) acao = "";

        if (acao.equals("incluir")){
            resultadoOperacao = ControleTipoServico.INSERIR;


        } else if (acao.equals("inserir")){
            resultadoOperacao = inserir(request, response);


        } else if (acao.equals("alterar")){
            resultadoOperacao = alterar(request, response);


        }else if (acao.equals("atualizar")){
            try {
                TipoServico tipoServico = buscar(request, response);
                request.setAttribute("tipoServico", tipoServico);
                resultadoOperacao = ControleTipoServico.ATUALIZAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = TipoServicoDAO.ERRO_CODIGO;
            }


        } else if (acao.equals("desativar")){
            resultadoOperacao = desativar(request, response);


        } else if (acao.equals("consultar")){
            try {
                List<TipoServico> lista = consultar(request, response);
                request.setAttribute("listaTipoServico", lista);
                resultadoOperacao = ControleTipoServico.LISTAR;
            } catch (EntidadeNulaException ex) {
                Logger.getLogger(ControleCliente.class.getName()).log(Level.SEVERE, null, ex);
                resultadoOperacao = TipoServicoDAO.ERRO_CONEXAO;
            }


        } else if (acao.equals("buscar")){
            resultadoOperacao = ControleTipoServico.BUSCAR;
```

```
        } else if (acao.equals("detalhar")){
            try {
                TipoServico tipoServico = buscar(request, response);
                request.setAttribute("tipoServico", tipoServico);
                resultadoOperacao = ControleTipoServico.DETALHAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = TipoServicoDAO.ERRO_CODIGO;
            }
        }

        RequestDispatcher rd;
        request.setAttribute("title","Tipo de Serviço - " + acao);
        switch (resultadoOperacao){
            case INSERIR:
                rd = request.getRequestDispatcher("inserirTipoServico.jsp");
                break;
            case TipoServicoDAO.SUCESSO:
                String processadas = " processadas ";
                if (acao.equals("inserir")){
                    processadas = " inseridas ";
                } else if (acao.equals("excluir")){
                    processadas = " excluídas ";
                } else {
                    processadas = " alteradas ";
                }
                request.setAttribute("conteudo","<h1>Informações sobre o tipo de serviço" +
processadas + "com sucesso</h1>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case TipoServicoDAO.ERRO_SQL:
                request.setAttribute("conteudo","<h1>Falha ao Acessar o Banco</h1>"
                                    + "<p>Erro durante o acesso (leitura ou escrita) ao banco
de dados. Comando SQL malformado. Procure o Webmaster</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case TipoServicoDAO.ERRO_INSERCAO:
                request.setAttribute("conteudo","<h1>Falha na Operação</h1>"
                                                + "<p>Os dados inseridos eram inválidos. Por
favor verifcar e tentar outra vez</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case TipoServicoDAO.ERRO_CODIGO:
                request.setAttribute("conteudo","<h1>Falha na Operação</h1>"
                                                + "<p>O Código informado nao corresponde a um
código de tipo de serviço presente no banco de dados</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case LISTAR:
                rd = request.getRequestDispatcher("listarTipoServico.jsp");
                break;
            case BUSCAR:
                rd = request.getRequestDispatcher("buscarTipoServico.jsp");
                break;
            case DETALHAR:
                rd = request.getRequestDispatcher("detalharTipoServico.jsp");
                break;
            case ATUALIZAR:
                rd = request.getRequestDispatcher("atualizarTipoServico.jsp");
                break;
            default:
//                request.setAttribute("conteudo","<h1>Falha na OperaÃ§Ã£o</h1>"
//                                                + "<p>Por algum motivo nÃ£o identificado, a
operaÃ§Ã£o foi cancelada. Entre em contato com o webmaster</p>");
//                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");

                rd = request.getRequestDispatcher("buscarTipoServico.jsp");
        }

        rd.forward(request, response);
    }

    public int alterar(HttpServletRequest request, HttpServletResponse response){
        try {
            TipoServicoDAO clienteDAO = new TipoServicoDAO();
```

```
            TipoServico cliente = new
TipoServico(Integer.parseInt(request.getParameter("tipo_servico_codigo")),
                                            request.getParameter("tipo_servico_nome"),

Integer.parseInt(request.getParameter("tipo_servico_duracao")),

Double.parseDouble(request.getParameter("tipo_servico_valor")),
                                            true);
            int resultadoOperacao = clienteDAO.alterar(this.usuario, this.senha,
this.endereco, cliente);
            return resultadoOperacao;
        } catch (InsercaoException ex) {
            //tratamento de falha na insercao
            return TipoServicoDAO.ERRO_INSERCAO;
        }
    }

    //VERIFICAR OS MÉTODOS de BUSCAR
    public TipoServico buscar(HttpServletRequest request, HttpServletResponse response) throws
EntidadeNulaException{
        return new TipoServicoDAO().buscar(usuario, senha, endereco,
Integer.parseInt(request.getParameter("tipo_servico_codigo")));
    }

    public List<TipoServico> consultar(HttpServletRequest request, HttpServletResponse
response) throws EntidadeNulaException{
        TipoServicoDAO clienteDAO = new TipoServicoDAO();
        return clienteDAO.consultar(this.usuario, this.senha, this.endereco);

    }

    public int desativar(HttpServletRequest request, HttpServletResponse response){
        TipoServicoDAO tipoServico = new TipoServicoDAO();
        int tipoServicoCodigo = Integer.parseInt(request.getParameter("tipo_servico_codigo"));
        int resultadoOperacao = tipoServico.desativar(this.usuario, this.senha, this.endereco,
tipoServicoCodigo);
        return resultadoOperacao;
    }

    public int inserir(HttpServletRequest request, HttpServletResponse response){
        int resultadoOperacao = 0;
        try {
            TipoServicoDAO tipoServicoDAO = new TipoServicoDAO();
            TipoServico tipoServico = new
TipoServico(Integer.parseInt(request.getParameter("tipo_servico_codigo")),
                                            request.getParameter("tipo_servico_nome"),

Integer.parseInt(request.getParameter("tipo_servico_duracao")),

Double.parseDouble(request.getParameter("tipo_servico_valor")),
                                            true);
            resultadoOperacao = tipoServicoDAO.inserir(this.usuario, this.senha,
this.endereco, tipoServico);
            return resultadoOperacao;
        } catch (InsercaoException ex) {
            //tratamento de falha na insercao
            return TipoServicoDAO.ERRO_INSERCAO;
        }
    }


    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    //</editor-fold>
```

```
}
```

APÊNDICE I

Código Fonte da Classe RegistroAtendimento

```
package br.com.oficina.modelo;

import java.util.GregorianCalendar;

public class RegistroAtendimento extends ModeloRaOs{
    private Atendente atendente;
    private Cliente cliente;
    private Carro carro;

    public RegistroAtendimento (int numero, GregorianCalendar dataAbertura,
            String descricaoAbertura, Atendente atendente, Cliente cliente,
            Carro carro, boolean ativo) throws InsercaoException, EntidadeNulaException{

        super(numero, dataAbertura, descricaoAbertura);
        this.setAtendente(atendente);
        this.setCliente(cliente);
        this.setCarro(carro);
    }

    public RegistroAtendimento (int numero, GregorianCalendar dataAbertura,
            String descricaoAbertura, GregorianCalendar dataEncerramento,
            String descricaoEncerramento, String estado, Atendente atendente, Cliente cliente,
            Carro carro, boolean ativo) throws InsercaoException, EntidadeNulaException{

        super(numero, dataAbertura, descricaoAbertura);
        this.setDataEncerramento(dataEncerramento);
        this.setDescricaoEncerramento(descricaoEncerramento);
        this.setEstado(estado);
        this.setAtendente(atendente);
        this.setCliente(cliente);
        this.setCarro(carro);
        this.setAtivo(ativo);

    }

    public Atendente getAtendente() {
        return atendente;
    }

    public void setAtendente(Atendente atendente) throws EntidadeNulaException {
        if (atendente == null){
            throw new EntidadeNulaException("Atendente inválido: objeto nulo passado como
referência");
        } else {
            this.atendente = atendente;
        }
    }

    public Cliente getCliente() {
        return cliente;
    }

    public void setCliente(Cliente cliente) throws EntidadeNulaException {
        if (cliente == null){
            throw new EntidadeNulaException("Cliente inválido: objeto nulo passado como
referência");
        } else {
            this.cliente = cliente;
        }
    }

    public Carro getCarro() {
        return carro;
    }

    public void setCarro(Carro carro) throws EntidadeNulaException {
        if (carro == null){
            throw new EntidadeNulaException("Carro inválido: objeto nulo passado como
referência");
```

```
        } else {
            this.carro = carro;
        }
    }

    @Override
    public String toString(){
        String descricao = super.toString() + "\nATENDENTE: " + this.getAtendente().getNome()
+ "\nCLIENTE: " + this.getCliente().getNome() + "\nPLACA CARRO: " + this.getCarro().getPlaca()
+ "\n============================\n";
        return descricao;
    }
}
```

APÊNDICE J

Código Fonte da Classe RegistroAtendimentoDAO

```
package br.com.oficina.dao;

import br.com.oficina.modelo.Atendente;
import br.com.oficina.modelo.Carro;
import br.com.oficina.modelo.Cliente;
import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import br.com.oficina.modelo.RegistroAtendimento;
import br.com.oficina.utils.Utils;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;

public class RegistroAtendimentoDAO extends DAO{
    public int inserir(String usuario, String senha, String endereco, RegistroAtendimento
registroAtendimento){
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir cliente.
TipoServicoDAO.inserir()") == null){
            return TipoServicoDAO.ERRO_CONEXAO;
        }

        try {
            String comandoSQL = "INSERT INTO registro_atendimento (ra_numero,
ra_data_abertura, ra_data_encerramento, ra_descricao_abertura, ra_descricao_encerramento,
ra_estado, ra_atendente_matricula, ra_cliente_id, ra_carro_id, ra_ativo)
VALUES(NEXTVAL('seq_registro_atendimento_numero'),?,?,?,?,?,?,?,?,?)";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            Utils.checaNull(ps,"PreparedStatement para inserir tipo de servico.
TipoServicoDAO.inserir()");

            ps.setDate(1, registroAtendimento.getDataAberturaAsSQLDate());
            ps.setDate(2, registroAtendimento.getDataEncerramentoAsSQLDate());
            ps.setString(3, registroAtendimento.getDescricaoAbertura());
            ps.setString(4, registroAtendimento.getDescricaoEncerramento());
            ps.setString(5, registroAtendimento.getEstado());
            ps.setInt(6, registroAtendimento.getAtendente().getId());
            ps.setInt(7, registroAtendimento.getCliente().getId());
            ps.setInt(8, registroAtendimento.getCarro().getId());
            ps.setBoolean(9, registroAtendimento.isAtivo());

            ps.execute();
            conexao.close();
            return RegistroAtendimentoDAO.SUCESSO;

        } catch(SQLException e) {
            System.err.print(e.getMessage());
        }
        return RegistroAtendimentoDAO.SUCESSO;
    }

    public int alterar(String usuario, String senha, String endereco, RegistroAtendimento
registroAtendimento){
        Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para alterar tipo de servico.
TipoServicoDAO.alterar()") == null){
            return TipoServicoDAO.ERRO_CONEXAO;
        }

        try {
            String comandoSQL = "UPDATE registro_atendimento SET ra_data_abertura = ?,
ra_data_encerramento = ?, ra_descricao_abertura = ?, ra_descricao_encerramento = ?, ra_estado
= ?, ra_atendente_matricula = ?, ra_cliente_id = ?, ra_carro_id = ?, ra_ativo = ?  WHERE
ra_numero = ?";
```

```
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setDate(1, registroAtendimento.getDataAberturaAsSQLDate());
            ps.setDate(2, registroAtendimento.getDataEncerramentoAsSQLDate());
            ps.setString(3, registroAtendimento.getDescricaoAbertura());
            ps.setString(4, registroAtendimento.getDescricaoEncerramento());
            ps.setString(5, registroAtendimento.getEstado());
            ps.setInt(6, registroAtendimento.getAtendente().getId());
            ps.setInt(7, registroAtendimento.getCliente().getId());
            ps.setInt(8, registroAtendimento.getCarro().getId());
            ps.setBoolean(9, registroAtendimento.isAtivo());
            ps.setInt(10, registroAtendimento.getNumero());

            ps.executeUpdate();
            conexao.close();

            return TipoServicoDAO.SUCESSO;
        } catch (org.postgresql.util.PSQLException e){
            System.err.println("ERRO: Falha ao alterar Registro de Atendimento de número " +
registroAtendimento.getNumero() + ". Violação de valor único\r\n " + e.getMessage());
            return TipoServicoDAO.ERRO_SQL;
        } catch (SQLException e) {
            System.err.println("ERRO: Falha ao alterar Registro de Atendimento de número " +
registroAtendimento.getNumero() + "\r\n " + e.getMessage());
            return ClienteDAO.ERRO_SQL;
        }
    }

    public int desativar(){
        return 0;
    }

    public List<RegistroAtendimento> consultar(){
        return new ArrayList();
    }

    public List<RegistroAtendimento> consultar(String usuario, String senha, String endereco,
boolean f){
        return new ArrayList();
    }

    public RegistroAtendimento buscar(String usuario, String senha, String endereco, int
codBusca) throws EntidadeNulaException{
            Connection conexao = ConectaBD.conectarBanco(usuario, senha, endereco);
        if (Utils.checaNull(conexao, "Connection para inserir Registro de Atendimento.
RegistroAtendimentoDAO.buscar()") == null){
            throw new EntidadeNulaException("Connection nula para inserir Registro de
Atendimento. RegistroAtendimentoDAO.buscar()");
        }

        RegistroAtendimento registroAtendimento = null;

        try {
            String comandoSQL = "SELECT ra_numero, ra_data_abertura, ra_data_encerramento,
ra_descricao_abertura, ra_descricao_encerramento, ra_estado, ra_atendente_matricula,
ra_cliente_id, ra_carro_id, ra_ativo FROM registro_atendimento WHERE ra_numero = ?";
            PreparedStatement ps = conexao.prepareStatement(comandoSQL);
            ps.setInt(1, codBusca);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                try{
                    /*TODO acessar banco para criar atendente*/
                    AtendenteDAO ad = new AtendenteDAO();
                    Atendente atendente = ad.buscarId(usuario, senha, endereco,
rs.getInt("ra_atendente_matricula"));
                    ClienteDAO cd = new ClienteDAO();
                    Cliente cliente = cd.buscarId(usuario, senha, endereco,
rs.getInt("ra_cliente_id"));
                    CarroDAO carDAO = new CarroDAO();
                    Carro carro = carDAO.buscar(usuario, senha, endereco,
rs.getInt("ra_carro_id"));

                    GregorianCalendar dataAbertura = new GregorianCalendar();
                    dataAbertura.setTime(rs.getDate("ra_data_abertura"));
                    GregorianCalendar dataEncerramento = new GregorianCalendar();
                    dataEncerramento.setTime(rs.getDate("ra_data_encerramento"));
```

```
                    registroAtendimento = new RegistroAtendimento(rs.getInt("ra_numero"),
                            dataAbertura, rs.getString("ra_descricao_abertura"),
dataEncerramento,
                            rs.getString("ra_descricao_encerramento"),
rs.getString("ra_estado"),
                            atendente, cliente, carro, rs.getBoolean("ra_estado"));

                }catch(InsercaoException e){
                    System.err.println("ERRO: Falha na busca por Registro de Atendimento:" +
"\r\n" + e.getMessage());
                }
                Utils.checaNull(registroAtendimento, "RegistroAtendimento nulo:
RegistroAtendimentoDAO.buscar()");
            }

            conexao.close();
        } catch (SQLException e) {
             System.err.println("Consulta inválida!" + "\nErro : " + e.getMessage());
        }
        if (Utils.checaNull(registroAtendimento, "RegistroAtendimento") == null){
            throw new EntidadeNulaException("Registr de Atendimento não encontrado no banco de
dados");
        }
        return registroAtendimento;
    }
}
```

APÊNDICE K

Código Fonte da Classe ControleRegistroAtendimento

```
package br.com.oficina.controle;

import br.com.oficina.dao.AtendenteDAO;
import br.com.oficina.dao.CarroDAO;
import br.com.oficina.dao.ClienteDAO;
import br.com.oficina.dao.RegistroAtendimentoDAO;
import br.com.oficina.dao.TipoServicoDAO;
import br.com.oficina.modelo.Atendente;
import br.com.oficina.modelo.Carro;
import br.com.oficina.modelo.Cliente;
import br.com.oficina.modelo.EntidadeNulaException;
import br.com.oficina.modelo.InsercaoException;
import br.com.oficina.modelo.RegistroAtendimento;
import br.com.oficina.modelo.TipoServico;
import br.com.oficina.utils.Utils;
import java.io.IOException;
import java.text.ParseException;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ControleRegistroAtendimento extends Controle implements InterfaceControle{

    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String acao = request.getParameter("acao");
        int resultadoOperacao = 20;
        if (acao==null) acao = "";

        if (acao.equals("incluir")){
            try {
                resultadoOperacao = ControleRegistroAtendimento.INSERIR;
                List<Cliente> listaClientes = new ClienteDAO().consultar(usuario, senha,
endereco);
                List<Carro> listaCarros = new CarroDAO().consultar(usuario, senha, endereco);
                List<Atendente> listaAtendentes = new AtendenteDAO().consultar(usuario, senha,
endereco);
                List<TipoServico> listaTiposServico = new TipoServicoDAO().consultar(usuario,
senha, endereco);

                request.setAttribute("listaClientes", listaClientes);
                request.setAttribute("listaCarros", listaCarros);
                request.setAttribute("listaAtendentes", listaAtendentes);
                request.setAttribute("listaTiposServico", listaTiposServico);
```

```
            } catch (EntidadeNulaException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            }


        } else if (acao.equals("inserir")){
            try {
                resultadoOperacao = inserir(request, response);
            } catch (ParseException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            } catch (EntidadeNulaException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            }


        } else if (acao.equals("alterar")){
            try {
                resultadoOperacao = alterar(request, response);
            } catch (ParseException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            } catch (EntidadeNulaException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InsercaoException ex) {

Logger.getLogger(ControleRegistroAtendimento.class.getName()).log(Level.SEVERE, null, ex);
            }


        }else if (acao.equals("atualizar")){
            try {
                RegistroAtendimento ra = buscar(request, response);
                request.setAttribute("registroAtendimento", ra);
                List<Cliente> listaClientes = new ClienteDAO().consultar(usuario, senha,
endereco);
                List<Carro> listaCarros = new CarroDAO().consultar(usuario, senha, endereco);
                List<Atendente> listaAtendentes = new AtendenteDAO().consultar(usuario, senha,
endereco);
                List<TipoServico> listaTiposServico = new TipoServicoDAO().consultar(usuario,
senha, endereco);

                request.setAttribute("listaClientes", listaClientes);
                request.setAttribute("listaCarros", listaCarros);
                request.setAttribute("listaAtendentes", listaAtendentes);
                request.setAttribute("listaTiposServico", listaTiposServico);
                resultadoOperacao = ControleTipoServico.ATUALIZAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = RegistroAtendimentoDAO.ERRO_CODIGO;
            }


        } else if (acao.equals("desativar")){
```

```
            resultadoOperacao = desativar(request, response);


        } else if (acao.equals("consultar")){
            try {
                List<RegistroAtendimento> lista = consultar(request, response);
                request.setAttribute("listaRegistroAtendimento", lista);
                resultadoOperacao = ControleTipoServico.LISTAR;
            } catch (EntidadeNulaException ex) {
                Logger.getLogger(ControleCliente.class.getName()).log(Level.SEVERE, null, ex);
                resultadoOperacao = RegistroAtendimentoDAO.ERRO_CONEXAO;
            }


        } else if (acao.equals("buscar")){
            resultadoOperacao = ControleRegistroAtendimento.BUSCAR;


        } else if (acao.equals("detalhar")){
            try {
                RegistroAtendimento ra = buscar(request, response);
                request.setAttribute("registroAtendimento", ra);
                resultadoOperacao = ControleTipoServico.DETALHAR;
            } catch (EntidadeNulaException ex) {
                resultadoOperacao = RegistroAtendimentoDAO.ERRO_CODIGO;
            }
        }

        RequestDispatcher rd;
        request.setAttribute("title","Registro de Atendimento - " + acao);
        switch (resultadoOperacao){
            case INSERIR:
                rd = request.getRequestDispatcher("inserirRegistroAtendimento.jsp");
                break;
            case RegistroAtendimentoDAO.SUCESSO:
                String processadas = " processadas ";
                if (acao.equals("inserir")){
                    processadas = " inseridas ";
                } else if (acao.equals("excluir")){
                    processadas = " excluídas ";
                } else {
                    processadas = " alteradas ";
                }
                request.setAttribute("conteudo","<h1>Informações sobre o tipo de serviço" +
processadas + "com sucesso</h1>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case RegistroAtendimentoDAO.ERRO_SQL:
                request.setAttribute("conteudo","<h1>Falha ao Acessar o Banco</h1>"
                                + "<p>Erro durante o acesso (leitura ou escrita) ao banco
de dados. Comando SQL malformado. Procure o Webmaster</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case RegistroAtendimentoDAO.ERRO_INSERCAO:
                request.setAttribute("conteudo","<h1>Falha na Operação</h1>"
```

```
                                                + "<p>Os dados inseridos eram inválidos. Por
favor verifcar e tentar outra vez</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case RegistroAtendimentoDAO.ERRO_CODIGO:
                request.setAttribute("conteudo","<h1>Falha na Operação</h1>"
                                                + "<p>O Código informado nao corresponde a um
código de tipo de serviço presente no banco de dados</p>");
                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");
                break;
            case LISTAR:
                rd = request.getRequestDispatcher("listarRegistroAtendimento.jsp");
                break;
            case BUSCAR:
                rd = request.getRequestDispatcher("buscarRegistroAtendimento.jsp");
                break;
            case DETALHAR:
                rd = request.getRequestDispatcher("detalharRegistroAtendimento.jsp");
                break;
            case ATUALIZAR:
                rd = request.getRequestDispatcher("atualizarRegistroAtendimento.jsp");
                break;
            default:
//                request.setAttribute("conteudo","<h1>Falha na OperaÃ§Ã£o</h1>"
//                                                + "<p>Por algum motivo nÃ£o identificado, a
operaÃ§Ã£o foi cancelada. Entre em contato com o webmaster</p>");
//                rd = request.getRequestDispatcher("/resultadoOperacao.jsp");

                rd = request.getRequestDispatcher("buscarRegistroAtendimento.jsp");
        }

        rd.forward(request, response);
    }

    public int alterar(HttpServletRequest request, HttpServletResponse response) throws
ParseException, EntidadeNulaException, InsercaoException{
        //tratamento de falha na insercao
            AtendenteDAO atendenteDAO = new AtendenteDAO();
            ClienteDAO clienteDAO = new ClienteDAO();
            CarroDAO carroDAO = new CarroDAO();
            RegistroAtendimentoDAO raDAO = new RegistroAtendimentoDAO();



            GregorianCalendar dataAbertura =
Utils.getGregCalendarDeString(request.getParameter("data_abertura"));
            String descricaoAbertura = request.getParameter("descricao_abertura");
            Atendente atendente = atendenteDAO.buscarId(usuario, senha, endereco,
Integer.parseInt(request.getParameter("id_atendente")));
            Cliente cliente = clienteDAO.buscarId(usuario, senha, endereco,
Integer.parseInt(request.getParameter("id_cliente")));
            Carro carro = carroDAO.buscar(usuario, senha, endereco,
Integer.parseInt(request.getParameter("cod_cliente")));

            RegistroAtendimento ra = new RegistroAtendimento(0, dataAbertura,
descricaoAbertura, atendente, cliente, carro, true);
```

```
        int resultadoOperacao = raDAO.alterar(this.usuario, this.senha, this.endereco, ra);
        return resultadoOperacao;
    }

    //VERIFICAR OS MÉTODOS de BUSCAR
    public RegistroAtendimento buscar(HttpServletRequest request, HttpServletResponse
response) throws EntidadeNulaException{
        return new RegistroAtendimentoDAO().buscar(usuario, senha, endereco,
Integer.parseInt(request.getParameter("ra_numero")));
    }

    public List<RegistroAtendimento> consultar(HttpServletRequest request, HttpServletResponse
response) throws EntidadeNulaException{
        RegistroAtendimentoDAO raDAO = new RegistroAtendimentoDAO();
        return raDAO.consultar(this.usuario, this.senha, this.endereco);

    }

    public int desativar(HttpServletRequest request, HttpServletResponse response){
        RegistroAtendimentoDAO raDAO = new RegistroAtendimentoDAO();
        int raNumero = Integer.parseInt(request.getParameter("ra_numero"));
        int resultadoOperacao = raDAO.desativar(this.usuario, this.senha, this.endereco,
raNumero);
        return resultadoOperacao;
    }

    public int inserir(HttpServletRequest request, HttpServletResponse response) throws
ParseException, EntidadeNulaException{
        int resultadoOperacao = 0;
        try {
            AtendenteDAO atendenteDAO = new AtendenteDAO();
            ClienteDAO clienteDAO = new ClienteDAO();
            CarroDAO carroDAO = new CarroDAO();
            RegistroAtendimentoDAO raDAO = new RegistroAtendimentoDAO();



            GregorianCalendar dataAbertura = new GregorianCalendar();
            String descricaoAbertura = request.getParameter("descricao_abertura");
            Atendente atendente = atendenteDAO.buscarId(usuario, senha, endereco,
Integer.parseInt(request.getParameter("atendente_id")));
            Cliente cliente = clienteDAO.buscarId(usuario, senha, endereco,
Integer.parseInt(request.getParameter("cliente_id")));
            Carro carro = carroDAO.buscar(usuario, senha, endereco,
Integer.parseInt(request.getParameter("carro_codigo")));

            RegistroAtendimento ra = new RegistroAtendimento(0, dataAbertura,
descricaoAbertura, atendente, cliente, carro, true);
            resultadoOperacao = raDAO.inserir(this.usuario, this.senha, this.endereco, ra);
            return resultadoOperacao;
        } catch (InsercaoException ex) {
            //tratamento de falha na insercao
            System.err.println(ex.getMessage() + "\n" + ex.getStackTrace());
            return RegistroAtendimentoDAO.ERRO_INSERCAO;
        }
    }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    //</editor-fold>
}
```

APÊNDICE L

Código Fonte das JavaServer Pages de Manipulação de TipoServico

```
<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Tipo de Serviço - Incluir</title>
        <style type="text/css">
            .container {
                width: 500px;
                clear: both;
            }
            .container input {
                width: 100%;
                clear: both;
            }
        </style>

        <jsp:include page="/cabecalho.jsp"/>

        <%TipoServico servico = (TipoServico) request.getAttribute("tipoServico");%>
        <h1>Inserir Tipo de Serviço</h1>
        <form action="<%=request.getContextPath() %>/tipoServico/" method="post">
            <div class="container">
                <input type="hidden" name="acao" value="inserir"/>
                <label for="servico_nome">Nome</label>
                <input type="text" name="servico_nome" maxlength="35"/><br/>
                <label for="servico_duracao">Duração</label>
                <input type="number" name="servico_duracao" min="0.00" step="1"/><br/>
                <label for="servico_valor">Valor</label>
                <input type="number" name="servico_valor"min="0.00" step="0.01" /><br/>
            </div>
            <input class="bot_envio" type="submit" value="Inserir Dados"/><br/><br/>
        </form>
    </body>
</html>


<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Tipo de Serviço - Atualizar</title>
        <style type="text/css">
            .container {
                width: 500px;
                clear: both;
            }
            .container input {
                width: 100%;
                clear: both;
            }
        </style>

        <jsp:include page="/cabecalho.jsp"/>

        <%TipoServico servico = (TipoServico) request.getAttribute("tipoServico");%>
        <h1>Alterar Dados do Tipo de Servico</h1>
        <form action="<%=request.getContextPath() %>/tipoServico/" method="post">
            <div class="container">
                <input type="hidden" name="acao" value="alterar"/>
                <label for="servico_codigo">Codigo</label>
                <input type="number" name="servico_codigo" readonly="true" min="1" value="<%=
servico.getCodigo() %>"/><br/>
                <label for="servico_nome">Nome</label>
                <input type="text" name="servico_nome" maxlength="25" value="<%=
servico.getNome() %>"/><br/>
```

```
                <label for="servico_duracao">Duração</label>
                <input type="number" name="servico_duracao" min="1" value="<%=
servico.getDuracao()%>"/><br/>
                <label for="servico_valor">Valor</label>
                <input type="number" name="servico_valor"min="0.00" step="0.01" value="<%=
servico.getValor()%>"/><br/>
            </div>
            <input class="bot_envio" type="submit" value="Alterar Dados"/><br/><br/>
        </form>
    </body>
</html>


<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Tipo de Servico - Detalhar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Detalhes do Tipo de Servico</h1>
        <%TipoServico servico = (TipoServico) request.getAttribute("tipoServico");%>

        <p><b>Código: </b><%= servico.getCodigo() %></p>
        <p><b>Nome: </b><%= servico.getNome() %></p>
        <p><b>Duração: </b><%= servico.getDuracao() %></p>
        <p><b>Valor: </b><%= servico.getValor() %></p>

        <a class="bot_excluir" href="<%=request.getContextPath()
%>/tipoServico/?acao=desativar&servicocodigo=<%= servico.getCodigo()%>" onclick="return
confirm('Deseja realmente realizar a exclusão do registro?')">Desativar Tipo de Serviço</a>
        <a class="bot_detalhar" href="<%=request.getContextPath()
%>/tipoServico/?acao=atualizar&servico_codigo=<%= servico.getCodigo()%>">Atualizar Tipo de
Serviço</a><br/><br/>

    </body>
</html>


<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<%@ page import="java.util.List" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Tipos de Serviços - Listar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Lista de Tipos de Serviços Registrados</h1>

        <table border="1">
            <tr>
                <th>Código</th>
                <th>Nome</th>
                <th colspan="2">Ações</th>
            </tr>
            <%
                List<TipoServico> lista = (List<TipoServico>)
request.getAttribute("listaTipoServico");
                for (TipoServico servico : lista) {%>
                    <tr>
                        <td><%= servico.getCodigo()%></td>
                        <td>
                            <a href="<%=request.getContextPath()
%>/tipoServico/?acao=detalhar&servico_codigo=<%= servico.getCodigo() %>">
                                <%= servico.getNome() %>
                            </a>
                        </td>
                        <td>
```

```
                            <a href="<%=request.getContextPath()
%>/tipoServico/?acao=atualizar&servico_codigo=<%= servico.getCodigo() %>">
                                Atualizar
                            </a>
                        </td>
                        <td>
                            <a href="<%=request.getContextPath()
%>/tipoServico/?acao=desativar&servico_codigo=<%= servico.getCodigo() %>" onclick="return
confirm('Deseja realmente realizar a desativação do registro?')">
                                Desativar
                            </a>
                        </td>
                    </tr>
                <%}
            %>
            </table>

    </body>
</html>


<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Tipo de Serviço - Buscar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Buscar Tipode Serviço pelo Código</h1>
        <form action="<%=request.getContextPath() %>/tipoServico/" method="post">
            <input type="hidden" name="acao" value="detalhar"/>
            <label for="servico_codigo"><b>Código do Tipo de Serviço: </b></label>
            <input type="number" name="servico_codigo" max="2147483648"/>
            <input class="bot_envio" type="submit" value="Buscar"/>
        </form>
    </body>
</html>
```

APÊNDICE M

Código Fonte das JavaServer Pages de Manipulação de RegistroAtendimento

```
<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page import="br.com.oficina.modelo.RegistroAtendimento"%>
<%@page import="br.com.oficina.modelo.Carro"%>
<%@page import="br.com.oficina.modelo.Atendente"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Registro de Atendimento - Inserir</title>

        <jsp:include page="/cabecalho.jsp"/>

        <script type="text/javascript">
            var todosCarros ={
                <%
                    List<Carro> listaCarros = (List<Carro>)
request.getAttribute("listaCarros");
                    for (Carro carro:listaCarros){
                %>
                        <%=carro.getId()%> :{descricao:"<%=carro.getModelo() + "|"
+carro.getPlaca()%>",
                                             clienteCodigo:<%=carro.getCliente().getId()%>},
                <%
                    }
                %>
            };




            function setCliente(){
                var cliente = document.getElementById("cliente");
                var codigoCliente = cliente.options[cliente.selectedIndex].value;

                document.getElementById("cliente_id").value=codigoCliente;

                carroSelect = document.getElementById('carro');
                carroSelect.disabled = false;

                var tamanho = carroSelect.options.length;
                for (i=tamanho-1; i>0; i--){
                    carroSelect.options.remove(i)
                }

                for (var carroId in todosCarros){
                    if (todosCarros[carroId].clienteCodigo == codigoCliente){
                        carroSelect.options[carroSelect.options.length] = new
Option(todosCarros[carroId].descricao, carroId);
                    }
                }
            }



            function setCarro(){
                var carro = document.getElementById("carro");
                var codigoCarro = carro.options[carro.selectedIndex].value;

                document.getElementById("carro_codigo").value=codigoCarro;
                console.log("foi exec");
            }
```

```
            function setAtendente(){
                var atendente = document.getElementById("atendente");
                var atendenteId = atendente.options[atendente.selectedIndex].value;
                document.getElementById("atendente_id").value=atendenteId;
                console.log("foi exec")
            }
        </script>

        <h1>Gerar Registro de Atendimento</h1>

        <form action="<%=request.getContextPath() %>/registroAtendimento/" method="post">
            <input type="hidden" name="acao" value="inserir"/>

            <label for="atendente"><b>Atendente: </b></label>
            <select name="atendente" id="atendente" onblur="setAtendente()">
                <option value="" disabled selected>Selecione o Atendente</option>
                <%
                    List<Atendente> listaAtendentes = (List<Atendente>)
request.getAttribute("listaAtendentes");
                    for (Atendente atendente:listaAtendentes){
                %>
                        <option value ="<%=atendente.getId()%>">
                            <%=atendente.getNome()%>
                        </option>
                <%
                    }
                %>
            </select>
            <input type="hidden" value="0" name="atendente_id" id="atendente_id"/>
            <br/>

            <label for="cliente"><b>Cliente: </b></label>
            <select name="cliente" id="cliente" onblur="setCliente()" onchange="setCliente()">
                <option value="" disabled selected>Selecione o Cliente</option>
                <%
                    List<Cliente> listaClientes = (List<Cliente>)
request.getAttribute("listaClientes");
                    for (Cliente cliente:listaClientes){
                %>
                        <option value ="<%=cliente.getId()%>">
                            <%=cliente.getNome() + " | CPF: " + cliente.getCpf()%>
                        </option>
                <%
                    }
                %>
            </select>
            <input type="hidden" value="0" name="cliente_id" id="cliente_id"/>
            <br/>

            <label for="carro"><b>Carro: </b></label>
            <select name="carro" id="carro" disabled onblur="setCarro()">
                <option value="" disabled selected>Selecione o Carro</option>

            </select>
            <input type="hidden" value="10" name="carro_codigo" id="carro_codigo"/>
            <br/>
            <br/>

            <label for="descricao_Abertura"><b>Descrição do Atendimento: </b></label><br/>
            <textarea name="descricao_abertura" id="descricao_abertura"></textarea>

            <br/>
            <br/>
            <input class="bot_envio" type="submit" value="Inserir Registro de Atendimento">
            <input class="bot_cancela" type="button" value ="Cancelar"
onClick="window.location='<%=request.getContextPath() %>/home.jsp';"/><br/>
        </form>

    </body>
</html>


<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page import="br.com.oficina.modelo.RegistroAtendimento"%>
```

```
<%@page import="br.com.oficina.modelo.Carro"%>
<%@page import="br.com.oficina.modelo.Atendente"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Registro de Atendimento - Inserir</title>

        <jsp:include page="/cabecalho.jsp"/>

        <script type="text/javascript">
            var todosCarros ={
                <%
                    List<Carro> listaCarros = (List<Carro>)
request.getAttribute("listaCarros");
                    for (Carro carro:listaCarros){
                %>
                        <%=carro.getId()%> :{descricao:"<%=carro.getModelo() + "|"
+carro.getPlaca()%>",
                                             clienteCodigo:<%=carro.getCliente().getId()%>},
                <%
                    }
                %>
            };




            function setCliente(){
                var cliente = document.getElementById("cliente");
                var codigoCliente = cliente.options[cliente.selectedIndex].value;

                document.getElementById("cliente_id").value=codigoCliente;

                carroSelect = document.getElementById('carro');
                carroSelect.disabled = false;

                var tamanho = carroSelect.options.length;
                for (i=tamanho-1; i>0; i--){
                    carroSelect.options.remove(i)
                }

                for (var carroId in todosCarros){
                    if (todosCarros[carroId].clienteCodigo == codigoCliente){
                        carroSelect.options[carroSelect.options.length] = new
Option(todosCarros[carroId].descricao, carroId);
                    }
                }
            }



            function setCarro(){
                var carro = document.getElementById("carro");
                var codigoCarro = carro.options[carro.selectedIndex].value;

                document.getElementById("carro_codigo").value=codigoCarro;
                console.log("foi exec");
            }

            function setAtendente(){
                var atendente = document.getElementById("atendente");
                var atendenteId = atendente.options[atendente.selectedIndex].value;
                document.getElementById("atendente_id").value=atendenteId;
                console.log("foi exec")
            }

            function toggleEncerrado(){
                if (document.getElementById("ra_esta_encerrado").checked==true){
                    document.getElementById("campo_encerramento").style.display = "block";
                    document.getElementById("descricao_encerramento").value=null;
                } else {
```

```
                    document.getElementById("campo_encerramento").style.display = "none";
                }
            }
        </script>
        <%RegistroAtendimento ra = (RegistroAtendimento) request.getAttribute("ra");
        String atendenteNome = ra.getAtendente().getNome();
        int atendenteId = ra.getAtendente().getId();
        String clienteNome = ra.getCliente().getNome();
        int clienteId = ra.getCliente().getId();
        String carroDescricao = ra.getCarro().getModelo() + "|" +ra.getCarro().getPlaca();
        int carroCodigo = ra.getCarro().getId();

        %>
        <h1>Gerar Registro de Atendimento</h1>

        <form action="<%=request.getContextPath() %>/registroAtendimento/" method="post">
            <input type="hidden" name="acao" value="alterar"/>

            <label for="atendente"><b>Atendente: </b></label>
            <select name="atendente" id="atendente" onblur="setAtendente()">
                <option value="<%=ra.getAtendente().getId()%>"
selected><%=ra.getAtendente().getNome()%></option>
                <%
                    List<Atendente> listaAtendentes = (List<Atendente>)
request.getAttribute("listaAtendentes");
                    for (Atendente atendente:listaAtendentes){
                %>
                        <option value ="<%=atendente.getId()%>">
                            <%=atendente.getNome()%>
                        </option>
                <%
                    }
                %>
            </select>
            <input type="hidden" value="0" name="atendente_id" id="atendente_id"/>
            <br/>

            <label for="cliente"><b>Cliente: </b></label>
            <select name="cliente" id="cliente" onblur="setCliente()" onchange="setCliente()">
                <option value="<%=ra.getCliente().getId()%>"
selected><%=ra.getCliente().getNome()%></option>
                <%
                    List<Cliente> listaClientes = (List<Cliente>)
request.getAttribute("listaClientes");
                    for (Cliente cliente:listaClientes){
                %>
                        <option value ="<%=cliente.getId()%>">
                            <%=cliente.getNome() + " | CPF: " + cliente.getCpf()%>
                        </option>
                <%
                    }
                %>
            </select>
            <input type="hidden" value="<%=ra.getCliente().getId()%>" name="cliente_id"
id="cliente_id"/>
            <br/>

            <label for="carro"><b>Carro: </b></label>
            <select name="carro" id="carro" disabled onblur="setCarro()">
                <option value="<%=ra.getCarro().getId()%>"
selected><%=ra.getCarro().getModelo() + "|" +ra.getCarro().getPlaca()%></option>

            </select>
            <input type="hidden" value="<%=ra.getCarro().getId()%>" name="carro_codigo"
id="carro_codigo"/>
            <br/>
            <br/>

            <label for="descricao_Abertura"><b>Descrição do Atendimento: </b></label><br/>
            <textarea name="descricao_abertura" id="descricao_abertura"></textarea>

            <br/>
            <input type="checkbox" name="ra_esta_encerrado" id="ra_esta_encerrado"
value="false">
            <label for="ra_esta_encerrado">Registro de Atendimento encerrado?</label>
```

```
            <div id="campo_encerramento" style="visibility:collapse">
                <textarea name="descricao_encerramento"
id="descricao_encerramento"></textarea>
            </div>
            <br/>
            <input class="bot_envio" type="submit" value="Inserir Registro de Atendimento">
            <input class="bot_cancela" type="button" value ="Cancelar"
onClick="window.location='<%=request.getContextPath() %>/home.jsp';"/><br/>
        </form>

    </body>
</html>


<%@page import="br.com.oficina.modelo.Cliente"%>
<%@page import="br.com.oficina.modelo.TipoServico"%>
<%@page

import="br.com.oficina.modelo.RegistroAtendimento"%>
<%@page import="br.com.oficina.modelo.Carro"%>
<%@page

import="br.com.oficina.modelo.Atendente"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE

html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Registro de Atendimento -

Atualizar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <script type="text/javascript">
            var todosCarros

={
                <%
                    List<Carro> listaCarros = (List<Carro>)
request.getAttribute("listaCarros");
                    for

(Carro carro:listaCarros){
                %>
                        <%=carro.getId()%> :{descricao:"<%=carro.getModelo() + "|"
+carro.getPlaca

()%>",
                                                clienteCodigo:<%=carro.getCliente().getId()%>},
                <%
                    }


        %>
            };




            function setCliente(){
                var cliente =

document.getElementById("cliente");
                var codigoCliente = cliente.options[cliente.selectedIndex].value;



   document.getElementById("cliente_id").value=codigoCliente;

                carroSelect = document.getElementById('carro');
```

```
            carroSelect.disabled = false;

                var tamanho = carroSelect.options.length;
                for

(i=tamanho-1; i>0; i--){
                    carroSelect.options.remove(i)
                }

                for (var carroId in

todosCarros){
                    if (todosCarros[carroId].clienteCodigo == codigoCliente){
                        carroSelect.options

[carroSelect.options.length] = new Option(todosCarros[carroId].descricao, carroId);
                    }
                }
            }




            function setCarro(){
                var carro = document.getElementById("carro");
                var

codigoCarro = carro.options[carro.selectedIndex].value;

                document.getElementById

("carro_codigo").value=codigoCarro;
                console.log("foi exec");
            }

            function setAtendente(){


       var atendente = document.getElementById("atendente");
                var atendenteId = atendente.options[atendente.selectedIndex].value;


              document.getElementById("atendente_id").value=atendenteId;
                console.log("foi exec")
            }



   function toggleEncerrado(){
                if (document.getElementById("ra_esta_encerrado").checked==true){


document.getElementById("campo_encerramento").style.display = "block";
                    document.getElementById

("descricao_encerramento").value=null;
                } else {
                    document.getElementById("campo_encerramento").style.display =

"none";
                }
            }
        </script>
        <%RegistroAtendimento ra = (RegistroAtendimento) request.getAttribute("ra");


 String atendenteNome = ra.getAtendente().getNome();
        int atendenteId = ra.getAtendente().getId();
        String clienteNome =

ra.getCliente().getNome();
        int clienteId = ra.getCliente().getId();
```

```
        String carroDescricao = ra.getCarro().getModelo() + "|"

+ra.getCarro().getPlaca();
        int carroCodigo = ra.getCarro().getId();

        %>
        <h1>Gerar Registro de Atendimento</h1>



        <form action="<%=request.getContextPath() %>/registroAtendimento/" method="post">
            <input type="hidden" name="acao"

value="alterar"/>

            <label for="atendente"><b>Atendente: </b></label>
            <select name="atendente" id="atendente"

onblur="setAtendente()">
                <option value="<%=ra.getAtendente().getId()%>"
selected><%=ra.getAtendente().getNome()%></option>


        <%
                    List<Atendente> listaAtendentes = (List<Atendente>)
request.getAttribute("listaAtendentes");


for (Atendente atendente:listaAtendentes){
                %>
                        <option value ="<%=atendente.getId()%>">


       <%=atendente.getNome()%>
                        </option>
                <%
                    }
                %>
            </select>


            <input type="hidden" value="0" name="atendente_id" id="atendente_id"/>
            <br/>

            <label

for="cliente"><b>Cliente: </b></label>
            <select name="cliente" id="cliente" onblur="setCliente()" onchange="setCliente()">


   <option value="<%=ra.getCliente().getId()%>"
selected><%=ra.getCliente().getNome()%></option>
                <%


List<Cliente> listaClientes = (List<Cliente>) request.getAttribute("listaClientes");
                    for (Cliente cliente:listaClientes){


           %>
                        <option value ="<%=cliente.getId()%>">
                            <%=cliente.getNome() + " | CPF: " +

cliente.getCpf()%>
                        </option>
                <%
                    }
                %>
            </select>


        <input type="hidden" value="<%=ra.getCliente().getId()%>" name="cliente_id"
id="cliente_id"/>
            <br/>
```

```
<label for="carro"><b>Carro: </b></label>
            <select name="carro" id="carro" disabled onblur="setCarro()">
                <option

value="<%=ra.getCarro().getId()%>" selected><%=ra.getCarro().getModelo() + "|"
+ra.getCarro().getPlaca()%></option>



</select>
            <input type="hidden" value="<%=ra.getCarro().getId()%>" name="carro_codigo"
id="carro_codigo"/>


<br/>
            <br/>

            <label for="descricao_Abertura"><b>Descrição do Atendimento: </b></label><br/>
            <textarea

name="descricao_abertura" id="descricao_abertura"></textarea>

            <br/>
            <input type="checkbox" name="ra_esta_encerrado"

id="ra_esta_encerrado" value="false">
            <label for="ra_esta_encerrado">Registro de Atendimento encerrado?</label>
            <div

id="campo_encerramento" style="visibility:collapse">
                <textarea name="descricao_encerramento"

id="descricao_encerramento"></textarea>
            </div>
            <br/>
            <input class="bot_envio" type="submit" value="Inserir

Registro de Atendimento">
            <input class="bot_cancela" type="button" value ="Cancelar"
onClick="window.location='<

%=request.getContextPath() %>/home.jsp';"/><br/>
        </form>

    </body>
</html>



<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
        <title>Registro de Atendimento - Buscar</title>

        <jsp:include page="/cabecalho.jsp"/>

        <h1>Buscar Cliente pelo Código</h1>
        <form action="<%=request.getContextPath() %>/registroAtendimento/" method="post">
            <input type="hidden" name="acao" value="detalhar"/>
            <label for="cliente_codigo"><b>Código do Registro de Atendimento </b></label>
            <input type="number" name="registro_atendimento_numero" max="2147483648"/>
            <input class="bot_envio" type="submit" value="Buscar"/>
        </form>
    </body>
</html>
```