Documento de Projeto de Sistema

Projeto: Locadora de Carros

Responsáveis: Ricardo Falbo

1. Introdução

Este documento apresenta o documento de projeto do sistema Locadora de Carros. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação do sistema; a Seção 3 apresenta a especificação dos requisitos não funcionais (atributos de qualidade), definindo as táticas e o tratamento a serem dados aos atributos de qualidade considerados condutores da arquitetura; a Seção 4 apresenta a arquitetura de software; a Seção 5 apresenta o projeto dos componentes da arquitetura; por fim, a Seção 6 apresenta o modelo relacional relativo ao sistema como um todo.

2. Plataforma de Implementação

O sistema em questão será implementado usando a linguagem de programação Java, o banco de dados relacional PostgreSQL e a API JPA com o *framework* de mapeamento objeto-relacional Hibernate.

3. Especificação de Requisitos Não Funcionais

A seguir, é a apresentada a especificação dos requisitos não funcionais identificados no Documento de Definição de Requisitos, os quais foram considerados condutores da arquitetura.

Tabela 3.1 – Especificação de Requisitos Não Funcionais

RNF02 – As funcionalidades de reserva e cadastro de clientes devem estar disponíveis pela Internet, a partir dos principais navegadores disponíveis no mercado, considerando, inclusive, dispositivos móveis (tablets, celulares).	
Categoria:	Portabilidade
Tática/ Tratamento:	Para manter a independência dos demais componentes, a interface com o usuário deverá ser separada do restante da aplicação. Além disso, serão usadas linguagens (Java) e bibliotecas capazes de rodar em diferentes plataformas. Os navegadores a serem considerados são: Google Chrome, Microsoft Edge, Mozilla Firefox e Safari. Deverão ser considerados os correlatos ao Chrome e Safari nas plataformas Android e iOS para tablets e smartphones, respectivamente.
Medida:	Tempo gasto para completar uma tarefa por um cliente quando usando um novo navegador.
Critério de Aceitação:	O sistema deverá funcionar essencialmente da mesma maneira nos navegadores Google Chrome, Microsoft Edge e Mozilla Firefox. Nesses três navegadores não poderá haver variações superiores a 5% no tempo gasto para completar uma tarefa. Entre o grupo acima e o navegador Safari, serão admitidas variações de até 20%. Em relação a dispositivos móveis, serão admitidas variações de até 50%.

4. Arquitetura de Software

A Figura 1 mostra a arquitetura do sistema. Essa arquitetura baseia-se em uma combinação dos estilos camadas e partições. Tomando por base a divisão em subsistemas realizada na fase de análise, inicialmente, foram definidas duas partições principais: *atendimentoCliente* e *controleFrota*. Uma vez que há a necessidade de ter parte do sistema rodando na Web, essa porção foi deslocada para uma nova partição, o subsistema *atendimentoClienteWeb.* Além disso, visando ao desenvolvimento para e com reúso, foram reutilizados os utilitários Pessoa (*utilitarioPessoa*) e Pagamento (*utilitarioPagamento*), bem como para tratar o requisito não funcional relativo a controle de acesso (RNF01) foi reusado o utilitário Segurança (*utilitarioSeguranca*), o qual encapsula a API Java de autenticação e autorização JAAS (Java Authentication and Authorization Service).

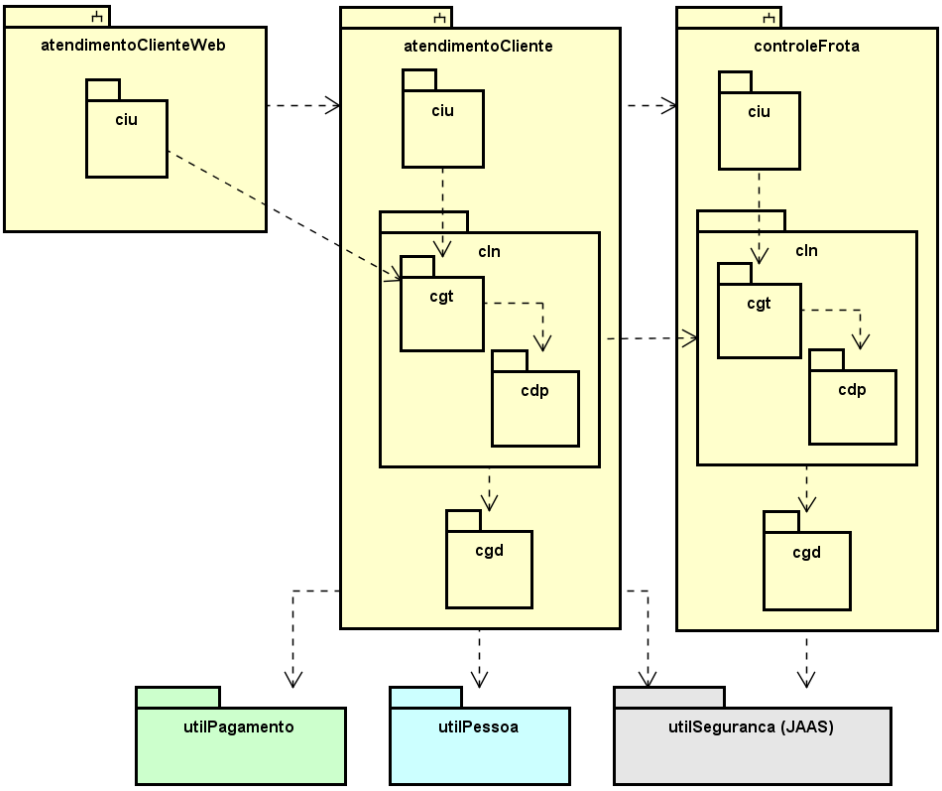

Figura 1 – Arquitetura de Software.

As partições mostradas na Figura 1, por sua vez, estão organizadas em três camadas, a saber: Camada de Interface com o Usuário (*ciu*), Camada de Lógica de Negócio (*cln*) e Camada de Gerência de Dados (*cgd*). A exceção fica por conta do pacote *atendimentoClienteWeb* que possui apenas a Camada de Interface com o Usuário (*ciu*), uma vez que utiliza os demais componentes (*cdp* e *cgd*) do subsistema *atendimentoCliente*.

No projeto da Camada de Lógica de Negócio, foi adotado o padrão Camada de Serviço. Assim, a camada de lógica de negócio é subdividida em duas camadas: Camada de Gerência de Tarefas (cgt) e Camada de Domínio do Problema (cdp).

Para o projeto da Camada de Interface com o Usuário e sua comunicação com a Camada de Lógica de Negócio, foi aplicado o padrão Modelo-Visão-Controlador (MVC). Quando combinados os padrões MVC e Camada de Serviço, a comunicação entre ciu e cln se dá entre classes controladoras de interação (da ciu) e classes gerenciadoras de tarefas (cgt).

5. Projeto dos Componentes da Arquitetura

Conforme mostrado na Figura 1, os dois principais subsistemas deste sistema estão organizados em três camadas: Camada de Interface com o Usuário (ciu), Camada de Lógica de Negócio (cln) e Camada de Gerência de Dados. Esta seção apresenta o projeto desses dois subsistemas.

5.1 - Subsistema *controleFrota*

5.1.1 – Camada de Domínio do Problema

Conforme citado anteriormente, no projeto da Camada de Lógica de Negócio, foi adotado o padrão Camada de Serviço. Assim, a camada de lógica de negócio é subdividida em duas camadas: Camada de Gerência de Tarefas (cgt) e Camada de Domínio do Problema (cdp). A Figura 2 mostra o diagrama de classes do cdp deste subsistema.

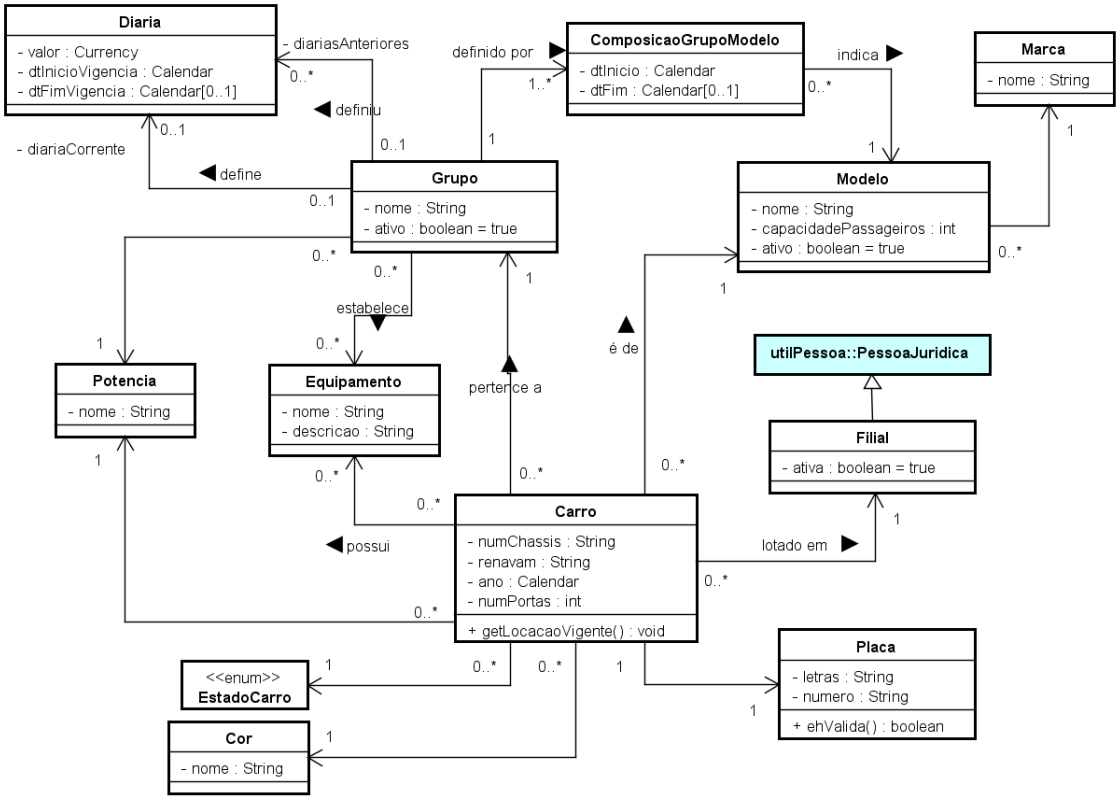

Figura 2 – Diagrama de classes do cdp do subsistema *controleFrota*

É importante destacar algumas decisões de projeto tomadas para se chegar a este modelo. Para tratar o tipo de dados de domínio Placa foi criada a classe de mesmo nome. Os atributos *cor* de **Carro** e *marca* de **Modelo** foram tratados como classes para permitir a aplicação da tática de usabilidade de permitir a seleção de dados a partir de um conjunto de valores pré-definidos de entrada. Por fim, a associação entre **Grupo** e **Diaria** foi desmembrada em duas para facilitar a recuperação da informação relativa à diária corrente de um grupo.

5.1.2 – Camada de Interface com o Usuário

Para o projeto da Camada de Interface com o Usuário e sua comunicação com a Camada de Lógica de Negócio, foi aplicado o padrão Modelo-Visão-Controlador (MVC). Quando combinados os padrões MVC e Camada de Serviço, a comunicação

entre ciu e cln se dá entre classes controladoras de interação (da ciu) e classes gerenciadoras de tarefas (cgt). A Figura 3 mostra o diagrama de classes da ciu deste subsistema e sua comunicação com as classes da cgt.

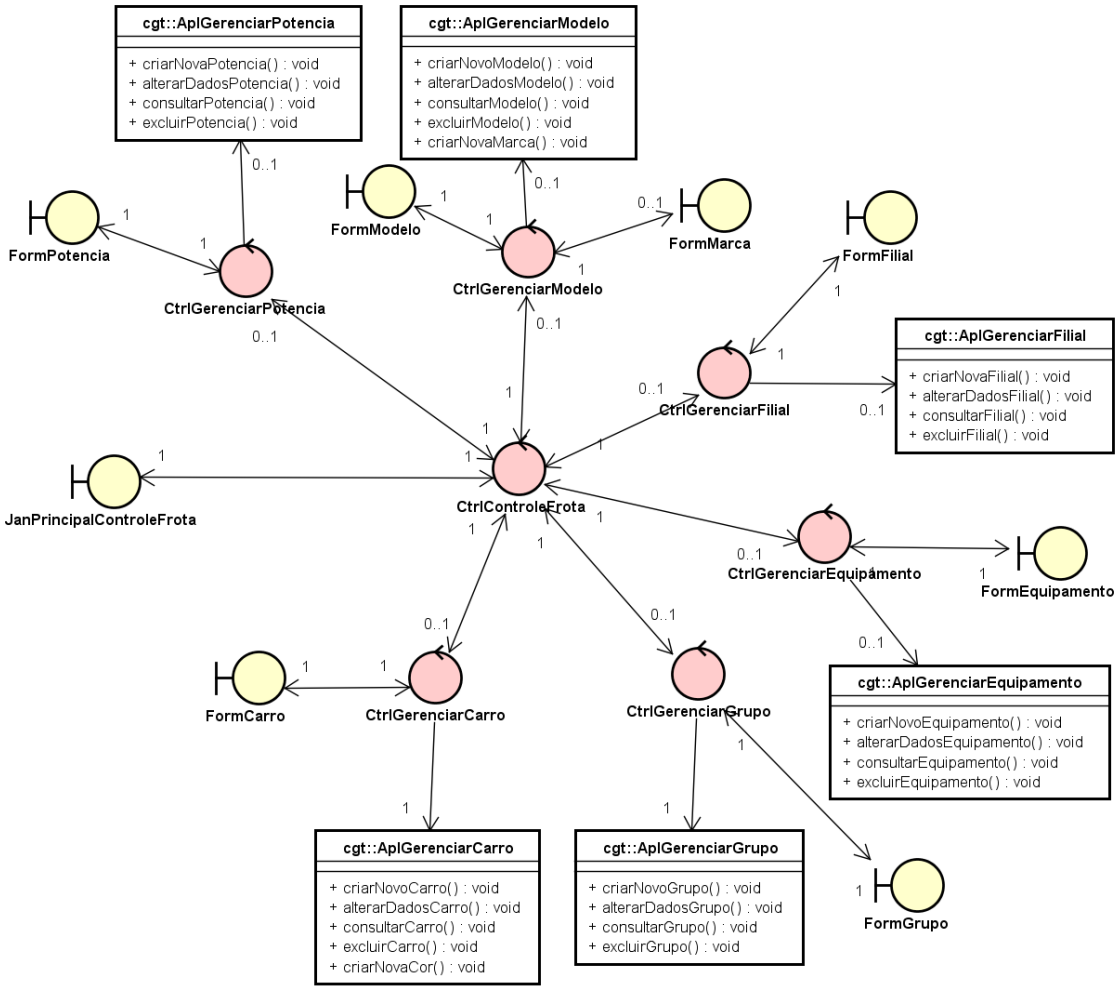

Figura 3 – Diagrama de classes da ciu + cgt do subsistema *controleFrota*

De maneira geral, foi adotada a estratégia de se definir uma classe de aplicação (classes da cgt prefixadas por Apl, mostradas em branco na Figura 3) e uma classe controladora de interação (classes da ciu prefixadas por Ctrl, mostradas em vermelho na Figura 3) para tratar cada caso de uso. Além dessas classes, por se tratar de uma aplicação desktop, foram criados um controlador principal para a aplicação (**CtrlControleFrota**) e uma janela principal (**JanPrincipalControleFrota**).

Uma vez que os atributos *cor* de **Carro** e *marca* de **Modelo** foram tratados como classes no cdp, foram criados métodos que permitem criar novos objetos dos tipos **Cor** e **Marca**, no momento em que se está incluindo/alterando um carro ou modelo. Assim, foram adicionados métodos nas respectivas classes (**AplGerenciarCarro** e **AplGerenciaMarca**).

5.1.3 – Camada de Gerência de Dados

Para o projeto da Camada de Gerência de Dados foram aplicados os padrões Campo de Identidade, para definição de chaves primárias, e Objeto de Acesso a Dados (*Data Access Object* – DAO) para a definição das classes responsáveis pela comunicação com o mecanismo de persistência. A Figura 4 mostra o diagrama de classes da cgd deste subsistema.

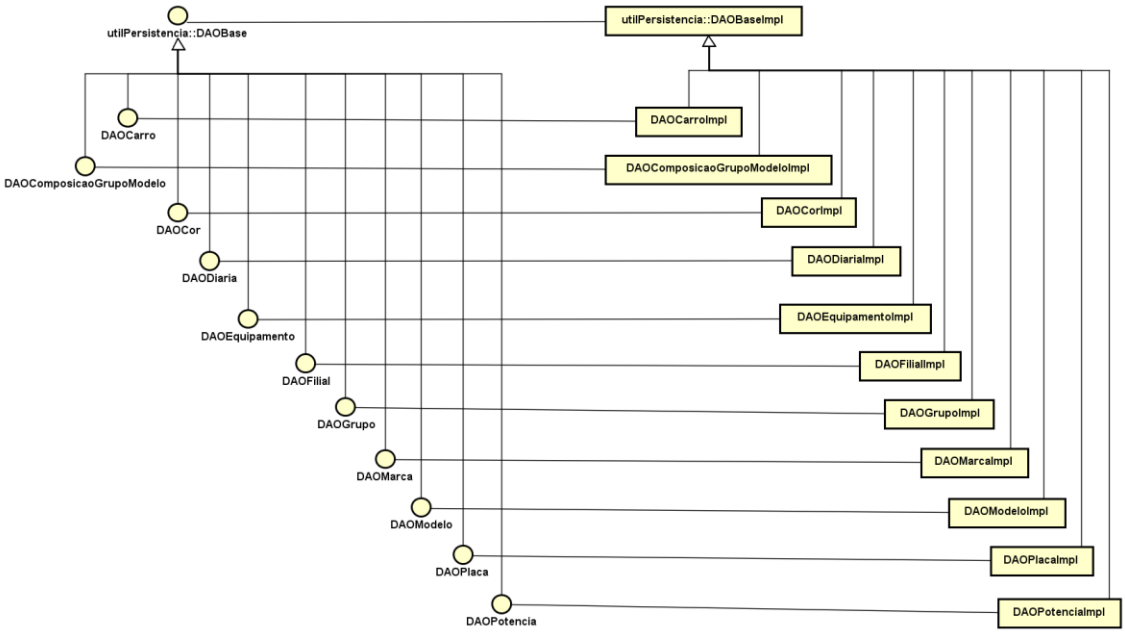

Figura 4 – Diagrama de classes da cgd do subsistema *controleFrota*

5.2 - Subsistema *atendimentoCliente*

Os padrões arquitetônicos usados foram os mesmos daqueles usados no *controleFrota*.

5.2.1 – Camada de Domínio do Problema

A Figura 5 mostra o diagrama de classes do cdp do *atendimentoCliente*.

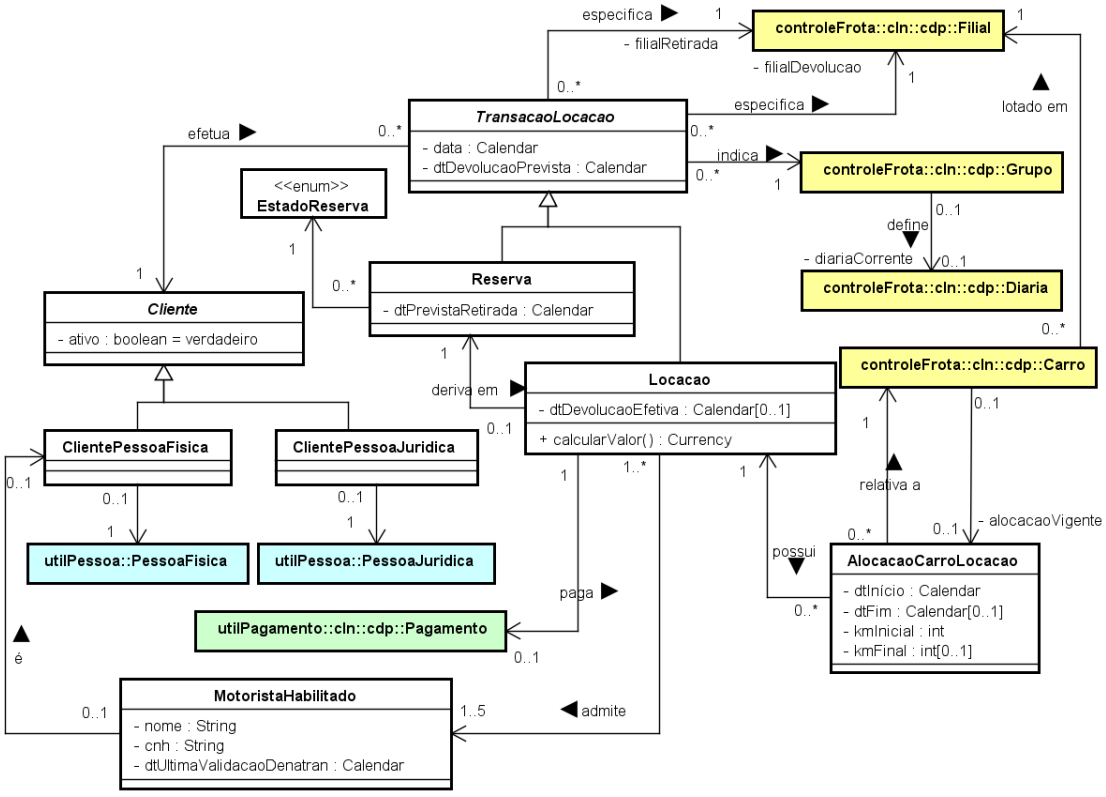

Figura 5 – Diagrama de classes do cdp do subsistema *atendimentoCliente*

É importante destacar algumas decisões de projeto tomadas para se chegar a este

modelo. Como a linguagem de programação a ser usada na implementação deste sistema (Java) não possui herança múltipla, as classes **ClientePessoaFisica** e **ClientePessoaJuridica** deixaram de herdar, respectivamente, das classes **PessoaFisica** e **PessoaJuridica** e passaram a ter associações com essas classes. Além disso, a associação entre **Carro** e **AlocacaoCarroLocacao** foi desmembrada em duas para facilitar a recuperação da informação relativa à alocação corrente de um carro.

5.2.2 – Camada de Interface com o Usuário

A Figura 6 mostra o diagrama de classes da ciu junto com a cgt do subsistema *atendimentoCliente*.

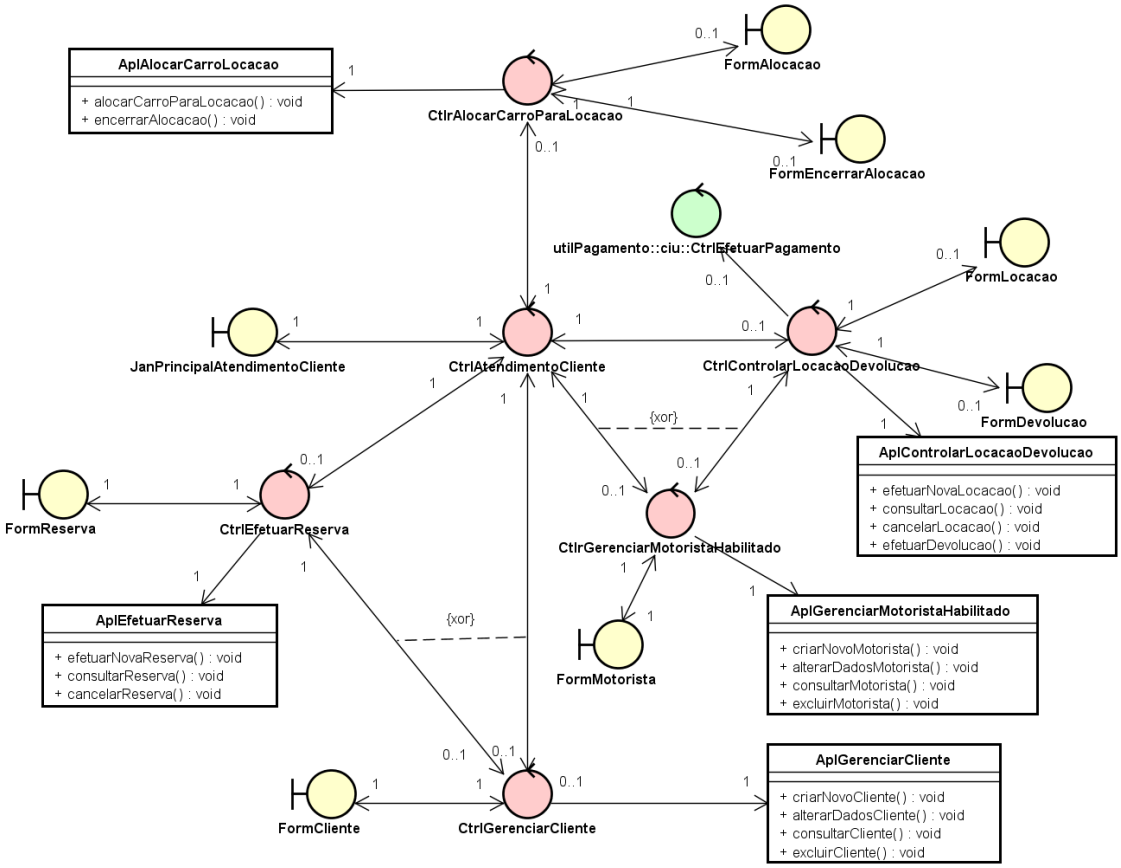

Figura 6 – Diagrama de classes da cih + cgt do subsistema *atendimentoCliente*

Da mesma forma que no subsistema *controleFrota*, neste subsistema, de maneira geral, foi adotada a estratégia de se definir uma classe de aplicação (classes da cgt prefixadas por Apl, mostradas em branco na Figura 6) e uma classe controladora de interação (classes da ciu prefixadas por Ctrl, mostradas em vermelho na Figura 6) para tratar cada caso de uso. A exceção ficou por conta dos casos de uso *Efetuar Locação* e *Efetuar Devolução* que, por serem muito correlatos, foram mapeados em uma mesma classe de gerência de tarefas (**AplControlarLocacaoDevolucao**) e tratados pelo mesmo controladore de interação (**CtrlControlarLocacaoDevolucao**).

Vale, ainda, destacar três aspectos nesse modelo. Primeiro, por se tratar de uma aplicação desktop, foram criados um controlador e uma janela principais para a aplicação (**CtrlAtendimentoCliente** e **JanPrincipalAtendimentoCliente**). Segundo, explorando a reutilização do pacote *utilPagamento*, para se ter acesso às funcionalidades de pagamento, foi reutilizado o controlador de interação **CtrlEfetuarPagamento** (mostrado em verde na Figura 6), o qual é acessado a partir do **CtrlControlarLocacaoDevolucao**. Finalmente, é importante observar que o controlador **CtrlGerenciarCliente** deve estar necessariamente associado ou ao

controlador principal (**CtrlAtendimentoCliente**) ou ao **CtrlEfetuarReserva**, uma vez que o caso de uso correspondente (*Gerenciar Informações de Cliente*) pode ser acessado diretamente pelo usuário ou por meio do caso de uso *Efetuar Reserva* (via relação de inclusão). Decisão análoga aplica-se ao controlador **CtrlGerenciarMotoristaHabilitado**.

6. Modelo Relacional

Conforme discutido na Seção 2, a persistência dos dados deste sistema é feita em um banco de dados relacional (PostgreSQL). Assim, em complemento ao projeto da Camada de Gerência de Dados, nesta seção é apresentado o modelo relacional do sistema. A Figura 7 apresenta o diagrama relacional, usando a UML como notação.

É importante destacar algumas decisões. Como foi aplicado o padrão Campo de Identidade para a definição de chaves primárias, as chaves são sempre identificadores (prefixados por id) gerados para este propósito e sem nenhuma relação com dados de domínio.

Para o mapeamento das relações de herança, as seguintes decisões foram tomadas:

- Hierarquia de **Cliente**: foi adotada a opção de mapeamento de uma tabela para mapear toda a hierarquia, uma vez que há poucas informações específicas nas subclasses e o processamento é feito majoritariamente na superclasse **Cliente**. Vale destacar que a organização já possui bases de dados de Pessoas Físicas e Jurídicas e estas serão integradas apenas.
- Hierarquia de **TransacaoLocacao**: foi adotada uma tabela para cada classe na hierarquia, uma vez que, de um lado, o processamento se dá principalmente nas subclasses (**Reserva** e **Locacao**), mas também há processamento importante na superclasse, uma vez que ela mantém relações muito importantes, tais como os relacionamentos com Cliente, Filial e Grupo.
- Hierarquia de **Pagamento**: foi adotada a opção de mapeamento de uma tabela para mapear toda a hierarquia, uma vez que há poucas informações específicas nas subclasses e o processamento é feito majoritariamente na superclasse **Pagamento**. Vale destacar, ainda, que foi necessário mapear todas as classes de utilitário, pois o sistema precisa registrar as informações de pagamento na base de dados.

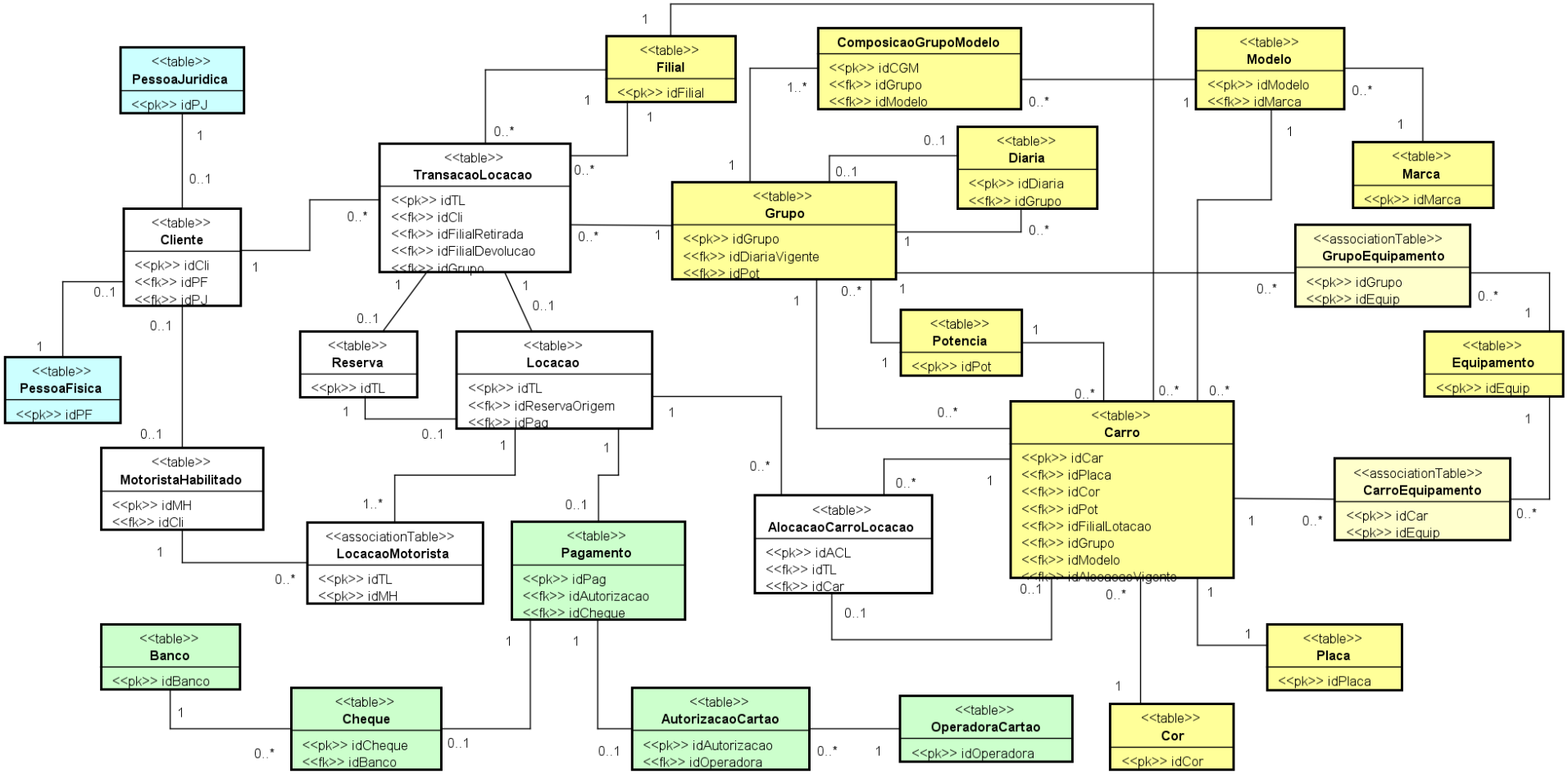

Figura 7 – Modelo Relacional