INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Richard de Souza Vieira

Vitor Firmino da Silva

PROJETO INTEGRADOR ENTRE AS DISCIPLINAS:
Análise e Projeto de Sistemas, Ferramentas de Programação I e Banco de Dados

Sistema Auxiliar para Transporte Escolar

Presidente Epitácio – SP

2019

Sumário

Índice de Figuras

1 Introdução

1.1 Objetivo

Este documento tem como objetivo esclarecer os requisitos para a criação do produto, explicitar os benefícios para o cliente geral e apresentar as funções do produto para o usuário final, impondo limites a essas funções.
Como também, expor diagramas de funcionamento do produto para que possíveis atualizações ou manutenções possam ser feitas por terceiros.

1.2 Escopo

O SATE (sistema auxiliar para transporte escolar) auxilia no controle e gestão dos serviços de transporte realizados pela empresa, visando informatizar os dados para facilitar a obtenção e manipulação de informações relevantes.

O sistema é composto por apenas um nível de acesso, o administrativo, onde se é disponibilizada toda a manipulação de dados do sistema;

O sistema possibilita a inclusão de um administrador que pode vincular um monitor à um motorista; E o motorista à um veículo em um determinado período. Quando um novo funcionário é contratado, o administrador, se for necessário, pode cadastrá-lo. O mesmo ocorre em relação aos veículos, quando um novo é adquirido pode ser cadastrado pelo administrador se necessário.

Com o veículo já cadastrado, caso o motorista faça uma manutenção no veículo, o mesmo deverá comunicar o administrador para que ele possa atualizar a lista de manutenções, que está presente no histórico do veículo.

O motorista tem uma lista de clientes a que é responsável. Quando um cliente novo pede o serviço o administrador poderá cadastrá-lo. Se for necessário, deverá cadastrar também a origem e o destino do cliente. Um cliente só pode ser incluído na lista se os veículos da empresa suportarem a quantidade de contratantes do motorista, caso contrário, o cliente não poderá ser adicionado à lista do motorista.

A viagem de um motorista é formada por uma lista de contratos dos clientes, que é comtém os pontos de origem e destino dos passageiros do motorista, se a lista for alterada a viagem também sofrerá modificações.

O administrador pode gerar relatórios do sistema a qualquer momento. Sendo um deles o de movimentação geral onde estão contidas todas as entradas e saída financeiras da empresa, disponibilizadas em relatórios.

O sistema tem como objetivo informatizar o sistema antigo da empresa. E assim facilitar a manipulação de dados no processo de obtenção de informações, proporcionando agilidade e precisão nos processos da empresa.

1.3 Regras de negócio

O recebimento dos clientes, será aceito somente se o mesmo for em

dinheiro por motivos contratuais. O pagamento dos funcionários tem de ser feito de acordo com o Art. 457, § 1º,CLT.

O sistema não pode apagar nenhum dado que já foi registrado, sendo assim podendo somente alterar ou arquivar os dados registrados.

1.4 Perspectiva do produto

O sistema será construído para ser usado em ambiente desktop. Deve se comportar adequadamente para todos os tipos de acesso. O sistema deve ser escrito e entregue em linguagem de programação Java.
O mesmo se constitui em telas com formas e cores agradáveis, visando conforto em longas utilizações do sistema e ícones intuitivos, para que o usuário se habitue facilmente ao sistema após um treinamento simples.
O tempo máximo de resposta das funções requeridas pelo usuário deve ser de até vinte segundos.
Para o uso dos serviços, os usuários deveram ser autenticados no nível de acesso administrativo, que é onde podem ocorrer: cadastros, alterações, visualizações e remoções de funcionários, veículos, clientes e/ou funções que permitem o vínculo de um funcionário a um veículo, vincular um monitor a um motorista e gerar relatórios.
A autenticação possibilita que o sistema tenha controle de disponibilização de informações entre os níveis de acesso.
O sistema será integrado a uma função de backup automático fornecido por terceiros, e deverá responder as requisições de backup a cada três dias de uso.
O sistema ficará conectado integralmente a uma impressora/scanner para facilitar a obtenção de documentos externos.

1.5 Funções do produto

1.5.1 Funções Fundamentais

RF_F1. Definir motorista do veículo: essa função permite vincular um motorista a um veículo requisitando as seguintes informações: registro do motorista, número do chassi do veículo, data do vínculo e horário estimado de devolução. Após vincular o veículo ao funcionário o sistema permite que o funcionário acesse o histórico de manutenções do veículo.

RF_F2. Vincular monitor ao motorista: essa função permite vincular um monitor a um motorista requisitando as seguintes informações: registro do monitor, registro do motorista, data do vínculo e horário. Após o vínculo o sistema permite que o administrador acesse o histórico de vinculos.

RF_F3. Registrar manutenção do veículo: essa função permite que após uma manutenção possam ser registrados as seguintes informações: tipo da manutenção,

registro do motorista, valor, estado do veículo, data e uma breve descrição. Ao ser registrada a manutenção, o sistema permite verificar o histórico do veículo.

RF_F4. Atribuir clientes ao motorista: essa função permite que após o cadastro do cliente e uma criança o administrador vincule-o a um motorista por meio de um contrato com as seguintes informações: CPF do responsável, endereço origem, endereço destino, registro do motorista, data início, data fim, identificação da criança e valor. Após registrar, o sistema faz a adiciona o contrato a viagem do motorista.

RF_F5. Receber pagamento do cliente: essa função permite que o administrador registre a efetuação de pagamento do cliente com as seguintes informações: data de vencimento referente a parcela, valor referente a uma parcela do contrato, número do contrato, CPF do cliente. Ao registrar o sistema permite consultar um relatório de pagamentos dos clientes.

RF_F6. Efetuar pagamento de funcionários: essa função permite que o administrador obtenha uma lista dos funcionários com as seguintes informações: nome do funcionário, salário fixo e quantidade de horas trabalhadas. Ao obter a lista de funcionários o administrador pode liberar o salário dos funcionários.

1.5.2 Funções Básicas

RF_B1. Manter funcionário: essa função permite o gerenciamento dos dados do funcionário, ou seja, permite a inserção, consulta, alteração e exclusão dos dados do funcionário (operações CRUD - *Create, Retrieve, Update* e *Delete*). Para isso são necessários os seguintes itens de informação: Número de registro, nome do funcionário, sexo, CPF, RG, data de nascimento, naturalidade, nome do pai, nome da mãe, grau de escolaridade, estado civil, e-mail, CNH, endereço, tipo de funcionário (motorista (CNH), monitor (registro pedagogico) e administrador (login e senha)) e forma de pagamento.

RF_B2. Manter veículo: essa função permite o gerenciamento dos dados do veículo, ou seja, permite a inserção, consulta, alteração e exclusão dos dados do veículo (operações CRUD - *Create, Retrieve, Update* e *Delete*). Para isso são necessários os seguintes itens de informação: modelo, número do chassi, número da placa, quantidade de assentos, quilometragem do veículo e sua condição atual.

RF_B3. Manter cliente: essa função permite o gerenciamento dos dados do cliente, ou seja, permite a inserção, consulta, alteração e exclusão dos dados do cliente (operações CRUD - *Create*, *Retrieve*, *Update* e *Delete*). Para isso são necessários os seguintes itens de informação: nome do cliente, data de nascimento, CPF Responsável, nome do pai, nome da mãe, endereço, e-mail e telefone.

RF_B4. Manter criança: essa função permite o gerenciamento dos dados da criança, ou seja, permite a inserção, consulta, alteração e exclusão dos dados da criança (operações CRUD - *Create*, *Retrieve*, *Update* e *Delete*). Para isso são necessários os seguintes itens de informação: nome da crainça, data de nascimento.

RF_B5. Manter local: essa função permite o gerenciamento dos dados do destino, ou seja, permite a inserção, consulta, alteração e exclusão dos dados do destino (operações CRUD - *Create*, *Retrieve*, *Update* e *Delete*). Para isso são necessários os seguintes itens de informação: nome do local, responsável, rua, bairro, cidade, cep, e-mail e telefone.

RF_B6. Manter Contrato: essa função permite o gerenciamento dos dados do contrato, ou seja, permite a inserção, consulta, alteração e exclusão dos dados do contrato (operações CRUD - *Create*, *Retrieve*, *Update* e *Delete*). Para isso são necessários os seguintes itens de informação: Data de Início do contrato, data de fim do contrato, CPF cliente, número de registro do motorista, local de origem e local de destino e valor do contrato.

1.5.3 Funções de Saída

RF_S1. Gerar relatório de funcionários: esse relatório permite a geração da listagem dos pagamentos em um determinado período.

Filtros: período (data inicial e data final).

Saída: número de registro, nome do funcionário e salário.

RF_S2. Gerar relatório de manutenção de veículos: esse relatório permite a geração da listagem das manutenções de um veículo em determinado período.

Filtros: período (data inicial e data final) e número do chassi do veículo.

Saída: chassi do veículo, descrição, valor e no final valor total das manutenções pelo periodo.

RF_S3. Gerar relatório de pagamento dos clientes: esse relatório permite a geração da listagem dos clientes pagantes ou inadimplentes, ou seja, que deixaram de pagar sua mensalidade em um determinado período.

Filtros: período (data inicial e data final).

Saída: nome do cliente, CPF, data vencimento, situação, valor.

RF_S4. Gerar relatório de movimentação geral: esse relatório permite a listagem das movimentações em relação ao tipo e um determinado período.

Filtros: período (data inicial e data final).

Saída: tipo de movimento, data, valor, descrição. Ao final um valor total da movimentação no periodo.

1.6 Restrições, Suposições e Dependências

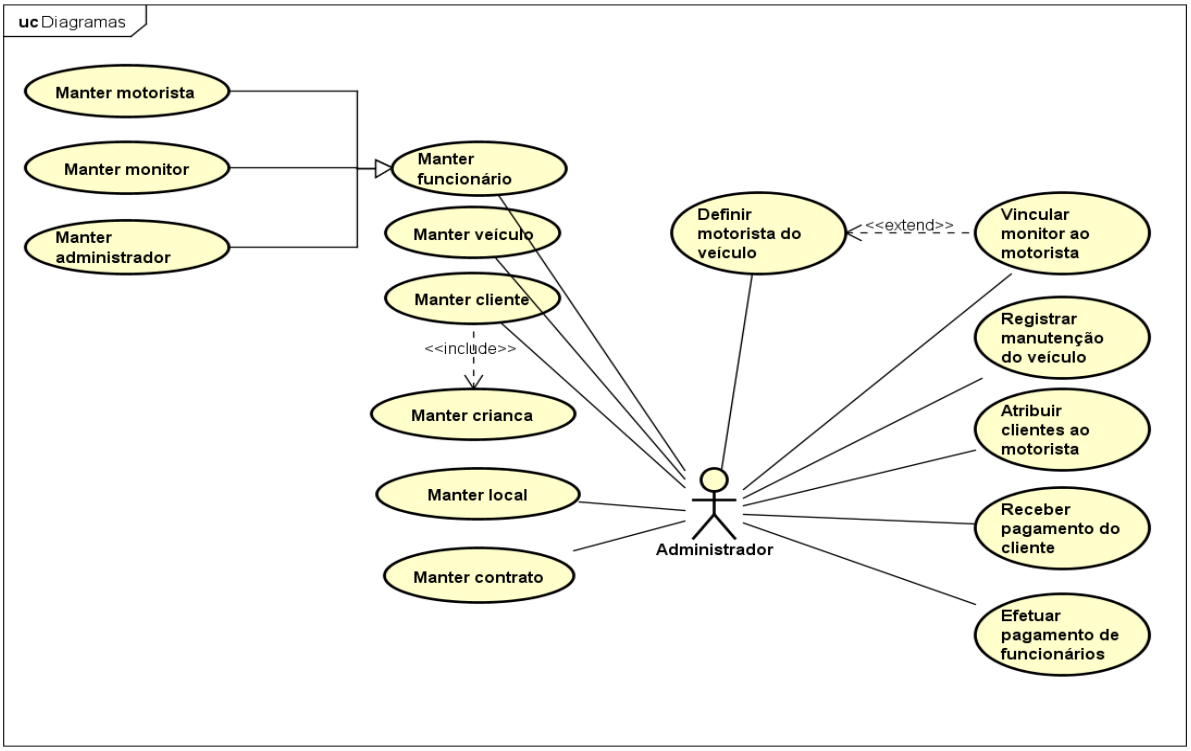
Não houve restrições na realização do projeto.

2 REQUISITOS ESPECÍFICOS

2.1 Diagrama de Caso de Uso

O diagrama a seguir apresenta o atuante principal de cada caso de uso do sistema. Vale ressaltar que a palavra “Manter” no diagrama representa as quatro operações CRUD (*Create, Retrieve, Update e Delete*).

Figura 1: Diagrama de caso de uso



2.2 Especificações de Caso de Uso e Diagramas de Atividade

A seguir será demonstrado o segmento de atividades tanto textualmente como também com o auxílio do diagrama de atividades.

Especificação do Caso de Uso: Definir motorista do veículo

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Atrelar os dados do motorista a um veículo.

Motorista: Receber a autorização do administrador para utilizar o veículo.

Pré-condições:

Administrador logado e autenticado no sistema;

Veículos cadastrados no sistema.

Garantia de sucesso (pós-condições):

Um motorista é atrelado a um veículo no sistema. O motorista tem a autorização para utilizar o veículo.

O sistema registra a viagem de um motorista e monitor como a fazer.

Fluxo Básico:

1. Este caso de uso é iniciado quando o motorista chega ao local de trabalho e o administrador precisa definir o veículo a ser utilizado pelo motorista.
2. O motorista informa seus dados ao administrador. O administrador preenche os dados do motorista no sistema.
3. O sistema consulta o registro do motorista no sistema.
4. O sistema consulta se um monitor está vinculado ao motorista.
5. O administrador define capacidade necessária para a viagem
6. O sistema consulta os veículos vagos.
7. O administrador define um dos veículos ao motorista.

Fluxos Alternativos:

3. O motorista não está registrado no sistema. Estender para o caso de uso “Manter Motorista”.

3.1 Retornar ao passo 4.

4. O motorista não está vinculado a um monitor. Estender para o caso de uso “Vincular monitor ao motorista”.

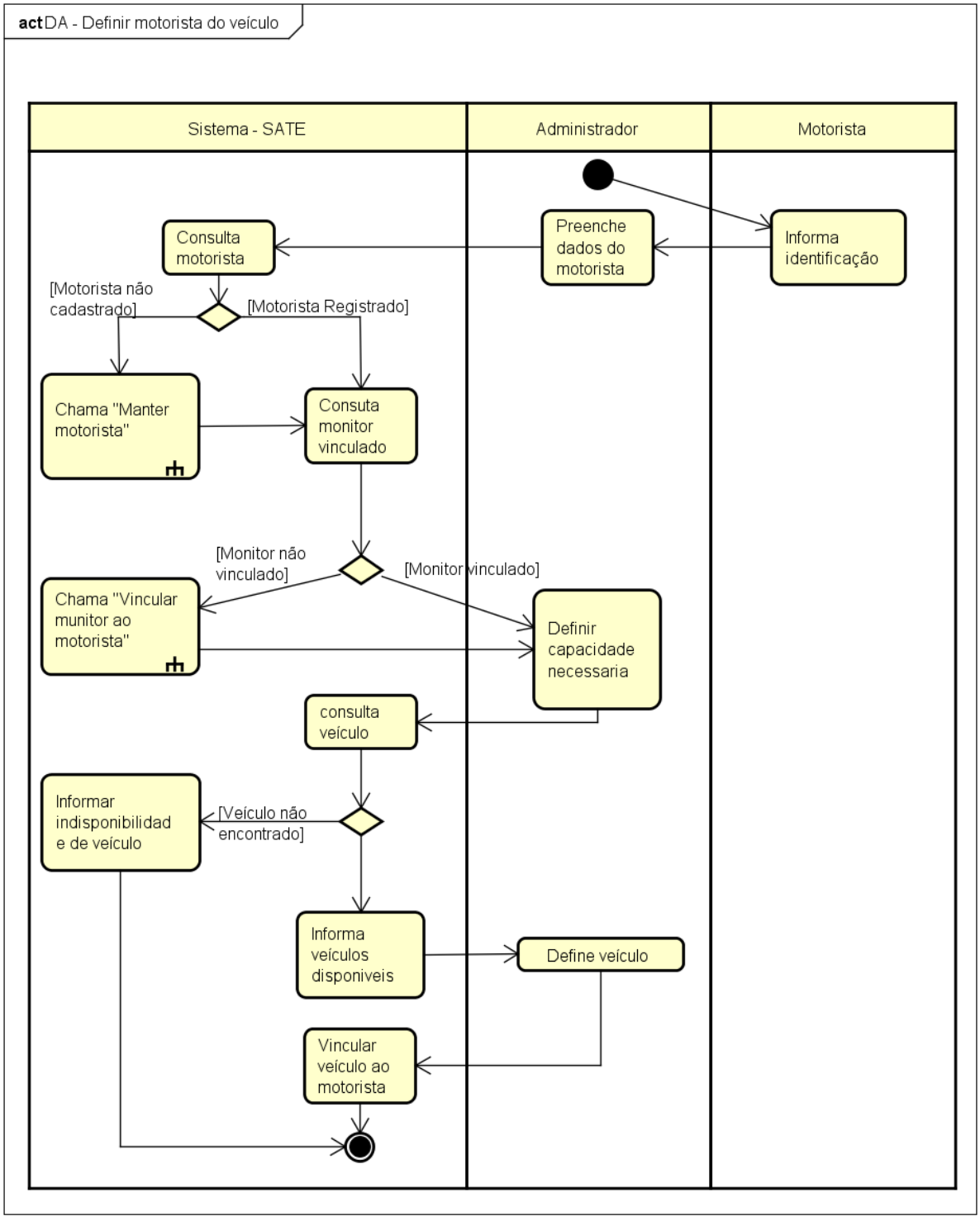
4.1 Retornar ao passo 5.

6. Não há veículos vagos.

6.1 O sistema informa indisponibilidade de veículo.

6.2 Encerra o caso de uso.

Figura 2: Definir motorista do veículo



Especificação do Caso de Uso: Vincular monitor ao motorista

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Atrelar os dados do monitor a um motorista.

Monitor: Receber a autorização do administrador para acompanhar o motorista.

Pré-condições:

Administrador logado e autenticado no sistema;

Garantia de sucesso (pós-condições):

Um monitor é atrelado a um motorista no sistema. O motorista tem a autorização para acompanhar o motorista.

O sistema registra a viagem de um motorista e monitor como a fazer.

Fluxo Básico:

1. Este caso de uso é iniciado quando o administrador precisa vincular um monitor a um motorista.
2. Monitor informa seus dados ao administrador. O administrador preenche os dados do monitor no sistema.
3. O sistema consulta o registro do monitor no sistema.
4. O sistema consulta se há motoristas sem vínculo.
5. O sistema define um motorista ao monitor.

Fluxos Alternativos:

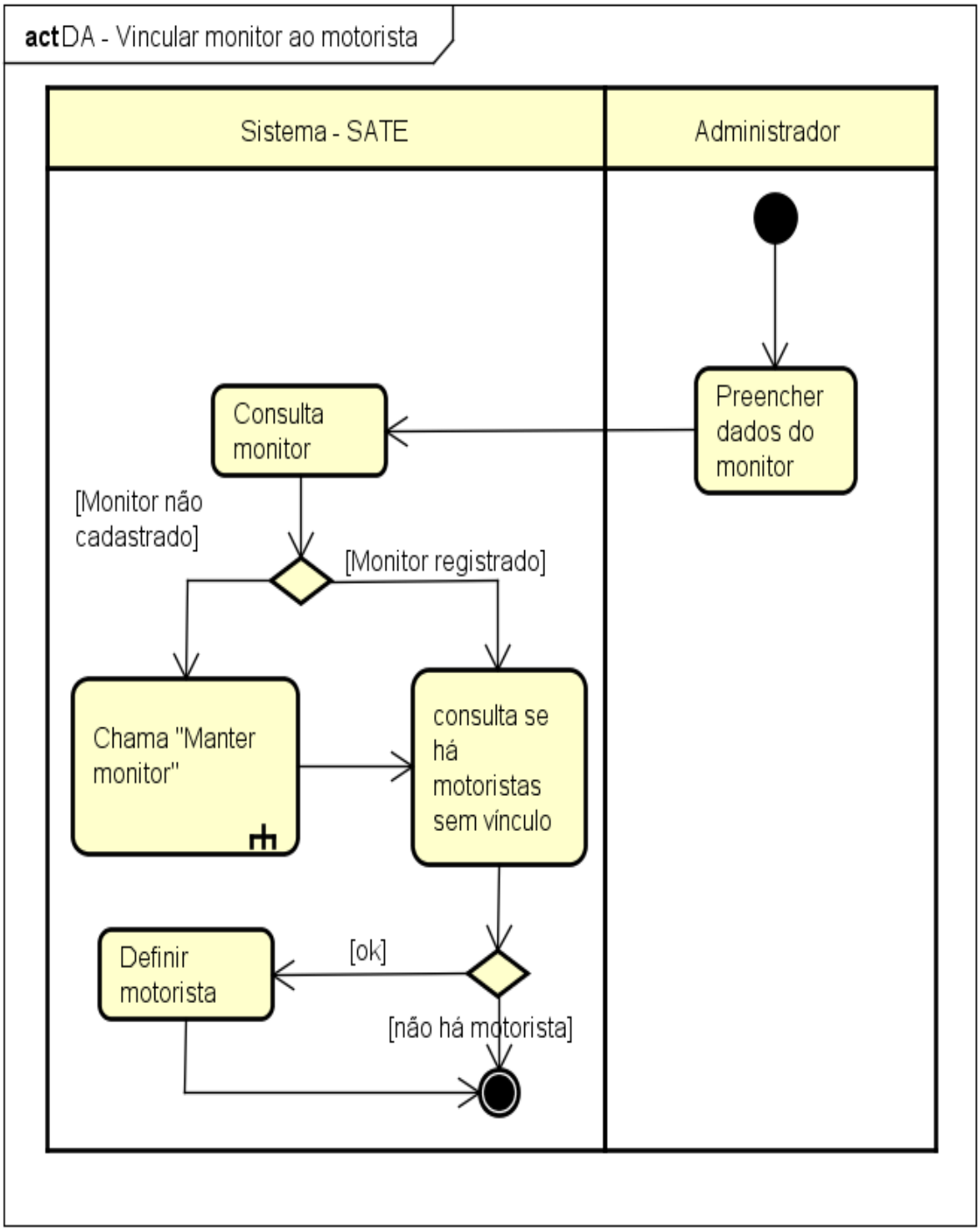
3. O monitor não está registrado no sistema. Estender para o caso de uso “Manter Monitor”.

3.1 Retornar ao passo 4.

4. Não há motorista sem vínculo.

4.1 Encerrar caso de uso.

Figura 3: Vincular monitor ao motorista



Especificação do Caso de Uso: Registrar manutenção do veículo

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Registrar dados da manutenção realizada no veículo.

Pré-condições:

Administrador logado e autenticado no sistema;

Veículo cadastrado no sistema;

Motorista cadastrado no sistema.

Garantia de sucesso (pós-condições):

É registrado e atualizado o histórico de manutenções do veículo.

Fluxo Básico:

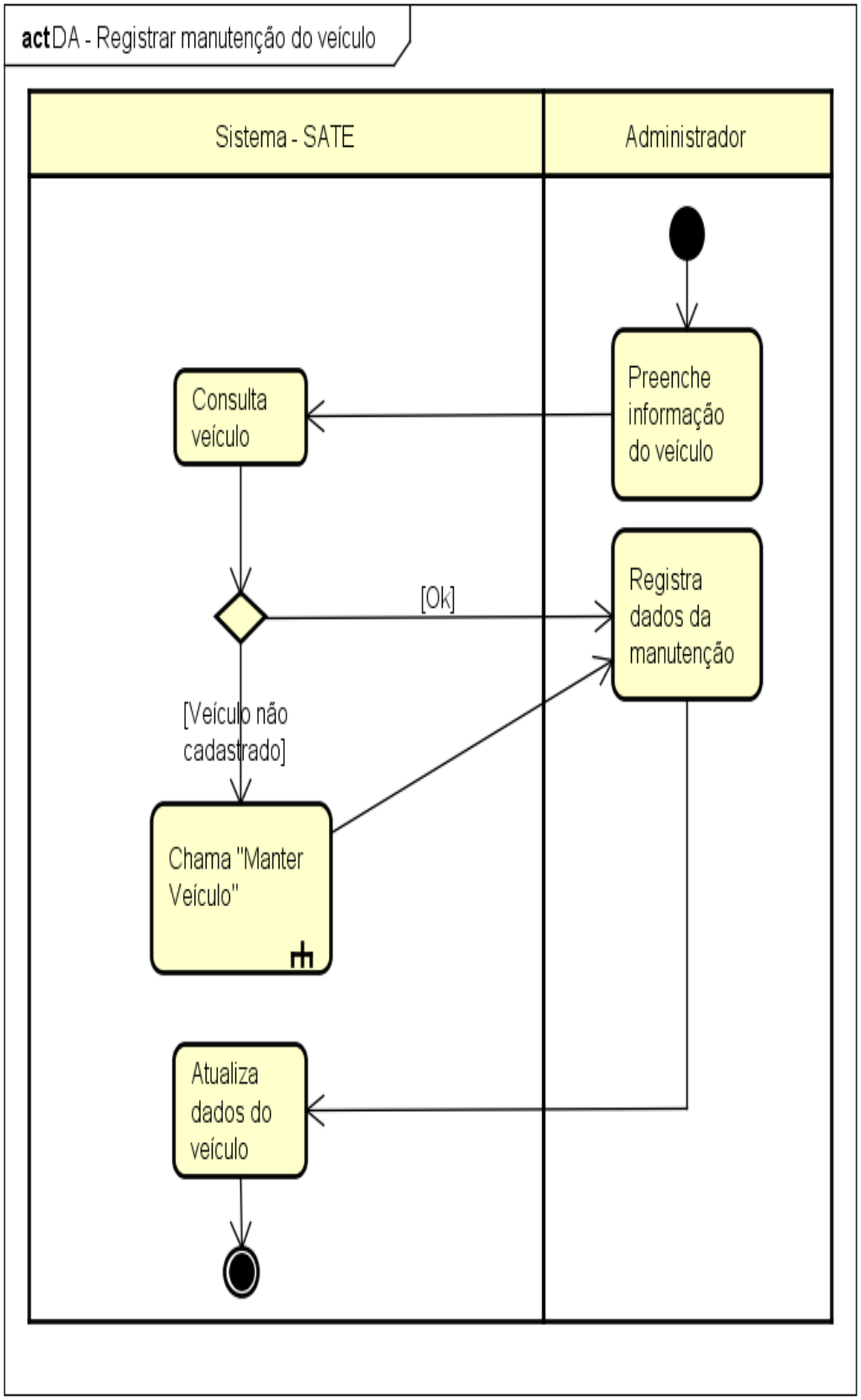
1. Este caso de uso inicia quando o motorista leva o veículo para realizar uma manutenção.
2. O administrador registra os dados da manutenção no sistemas informados pelo motorista.
3. O sistema consulta o veículo.
4. O sistema atualiza o histórico do veículo.

Fluxos Alternativos:

2. Não foi registrado nenhuma manutenção no histórico do veículo.

2.1 Encerra o caso de uso.

Figura 4: Registrar manutenção do veículo



Especificação do Caso de Uso: Atribuir clientes ao motorista

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Registrar dados do cliente e atrelá-lo à um motorista.

Motorista: Adicionar um novo cliente à sua lista.

Pré-condições:

Administrador logado e autenticado no sistema;

Motorista cadastrado no sistema.

Garantia de sucesso (pós-condições):

Registrar os dados do cliente e atrelá-lo a um motorista.

Fluxo Básico:

1. Este caso de uso inicia quando um cliente pede os serviços da empresa e se é acordado um contrato.
2. O cliente informa seus dados ao administrador. O administrador preenche os dados do cliente no sistema.
3. O sistema consulta o registro do cliente no sistema.
4. O sistema consulta a criança deste cliente que será inclusa no contrato.
5. O sistema consulta local de origem da criança no sistema.
6. O sistema consulta local de destino da criança no sistema.
7. O sistema consulta se há motoristas com vagas.
8. O administrador define um motorista para o cliente.
9. O sistema finaliza o contrato com todas as informações.

Fluxos Alternativos:

3. O cliente não está registrado no sistema. Estender para o caso de uso “Manter Cliente”.

3.1 Retornar ao passo 4.

4. A criança não está registrado no sistema. Estender para o caso de uso “Manter Criança”.

4.1 Retornar ao passo 5.

5. O local de origem não está registrado no sistema. Estender para o caso de uso “Manter local”.

5.1 Retorna ao passo 5.

6. O local de destino não está registrado no sistema. Estender para o caso de uso “Manter local”.

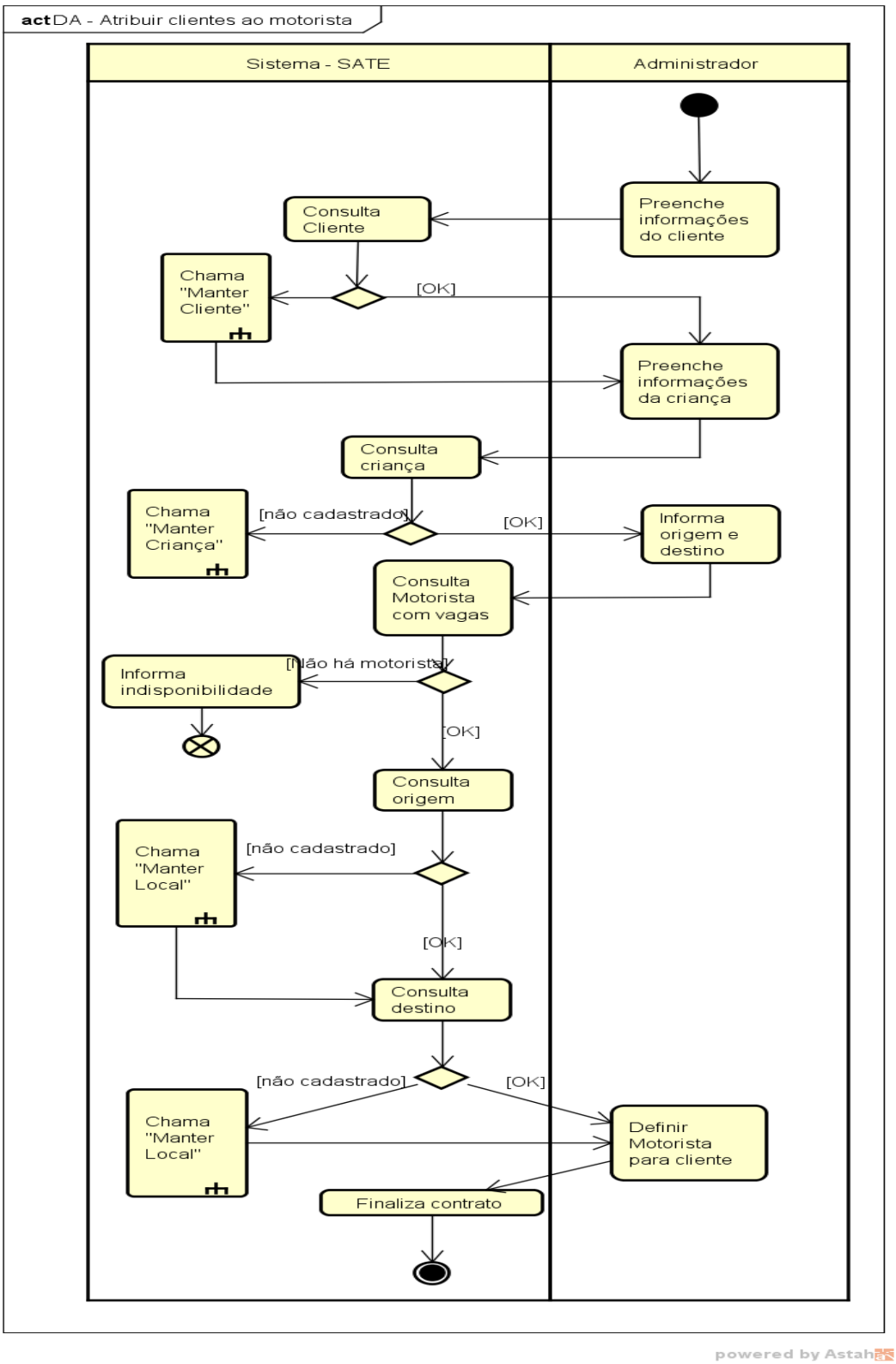
6.1 Retorna ao passo 6.

7. Não há motoristas com vagas.

7.1 O sistema informa indisponibilidade de motorista.

7.2 Encerra o caso de uso.

Figura 5: Atribuir clientes ao motorista



Especificação do Caso de Uso: Receber pagamento dos clientes

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Registrar o recebimento da mensalidade dos clientes ativos.

Pré-condições:

Administrador logado e autenticado no sistema;

Cliente cadastrado no sistema;

Recebimento somente em dinheiro.

Garantia de sucesso (pós-condições):

Ter os registros de pagamento de clientes atualizados no sistema.

Fluxo Básico:

1. Este caso de uso inicia quando o cliente chega ao caixa para realizar o pagamento da mensalidade.
2. O cliente informa o CPF.
3. O administrador registra o CPF
4. O sistema consulta o contrato do cliente.
5. O sistema traz fatura referente ao mês.
6. O cliente paga a fatura.
7. O administrador devolve a diferença ao cliente.
8. O sistema registra o pagamento.

Fluxos Alternativos:

5a. A fatura já foi paga

5a.1 O sistema informa o pagamento da fatura.

5a.2 Encerra o caso de uso.

5b. A fatura está atrasada.

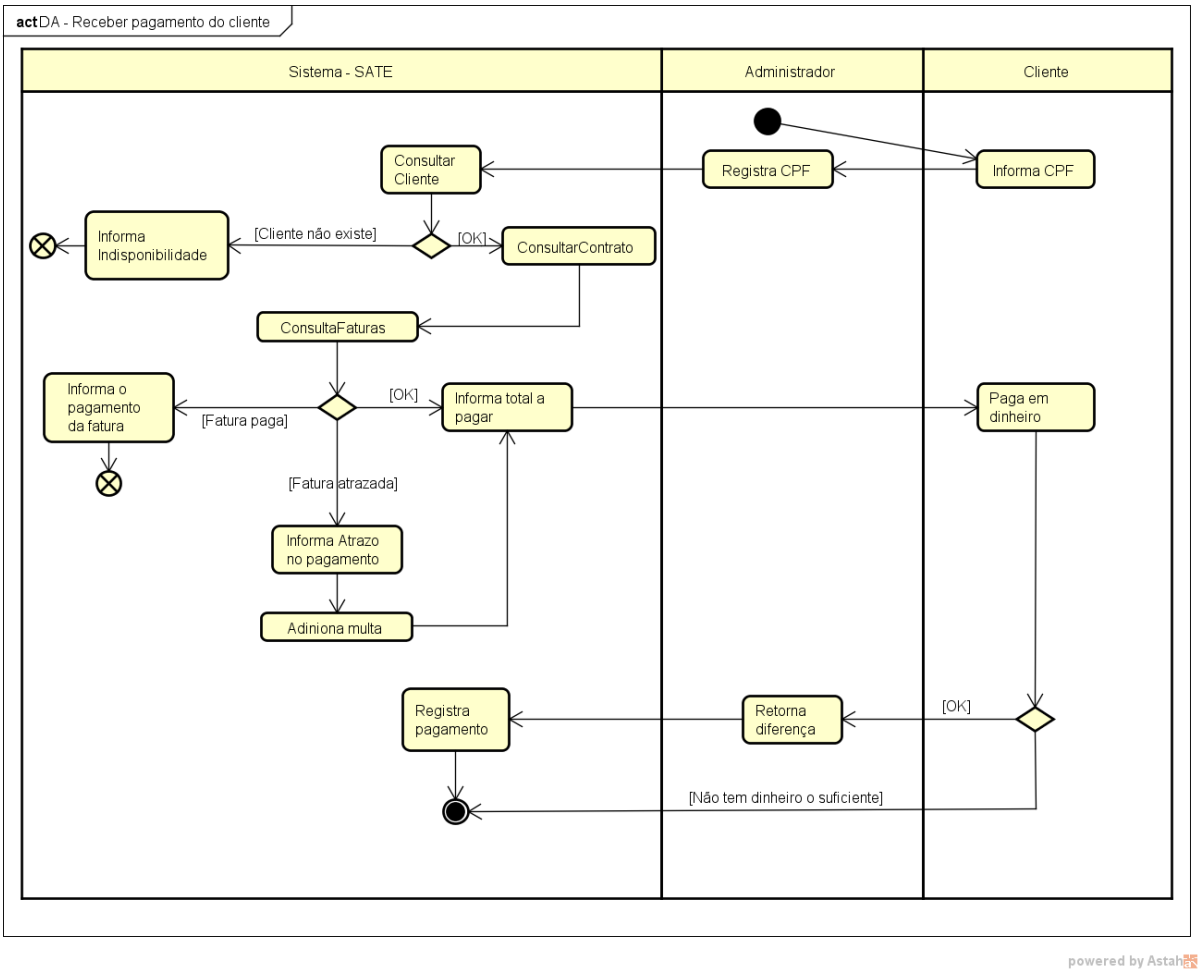
5b.1 O sistema aplica a multa a fatura.

5b.2 Retornar ao passo 5.

6. O cliente não tem dinheiro suficiente.

6.1 Encerra o caso de uso.

Figura 6: Receber pagamento do cliente



Especificação do Caso de Uso: Efetuar pagamento de funcionários

Ator Principal: Administrador

Interessados e Interesses:

Administrador: Registrar o pagamento aos funcionários ativos.

Funcionário: Receber o pagamento.

Pré-condições:

Administrador logado e autenticado no sistema;

Funcionário cadastrado no sistema.

Garantia de sucesso (pós-condições):

Ter o registro atualizado do pagamento aos funcionários.

Fluxo Básico:

1. Este caso de uso se inicia quando a data de pagamento dos funcionários marcada chega e o administrador libera o pagamento.
2. O sistema registro a data de pagamento.
3. O sistema busca os funcionários pelo registro.
4. O administrador libera o pagamento aos funcionários.

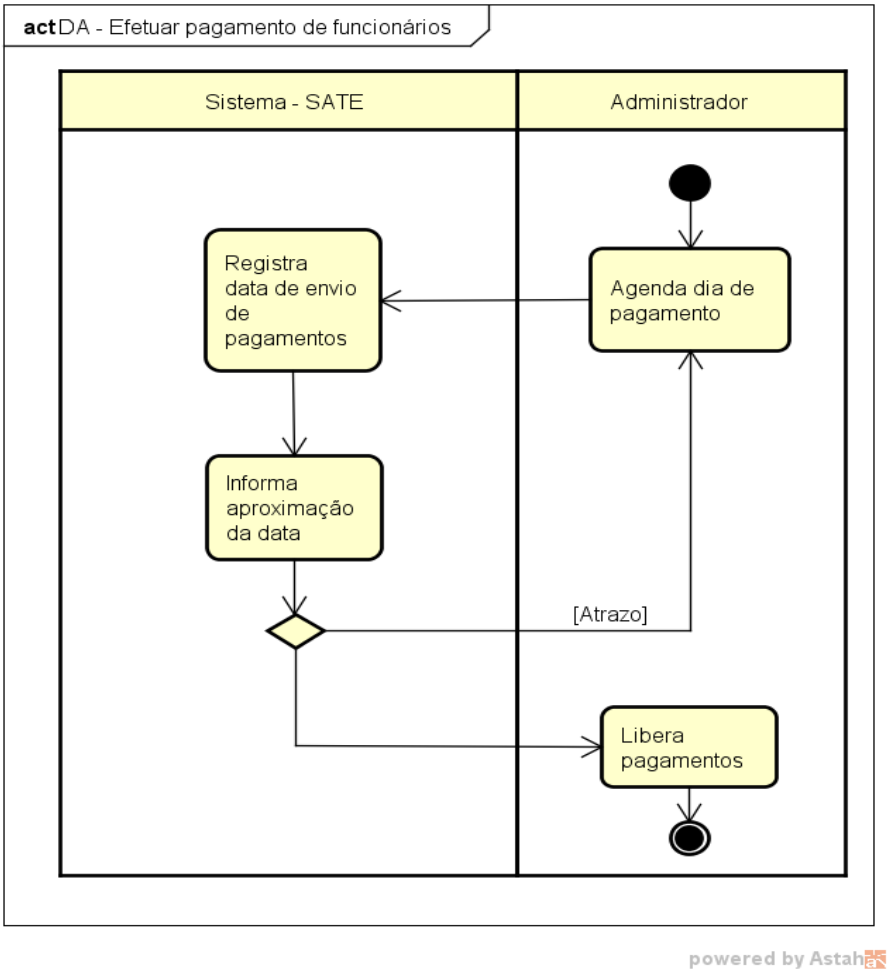
Fluxos Alternativos:

1. O administrador atrasa a liberação do pagamento.

1.1 A data de liberação do pagamento é reagendada.

1.2 Encerra o caso de uso.

Figura 7: Efetuar pagamento de funcionários



Especificação do Caso de Uso: Manter Motorista

Ator Principal: Administrador

Fluxo básico:

1. O caso de uso inicia quando o Administrador necessita fazer a manutenção (inclusão, alteração, exclusão ou consulta) de um motorista.

2. De acordo com o tipo de operação de manutenção desejado pelo Administrador, um dos subfluxos é executado:

a. Se o Administrador deseja incluir um novo motorista, o subfluxo “Incluir Motorista” é executado.

b. Se o Administrador deseja alterar informações de um motorista já cadastrado, o subfluxo “Alterar Motorista” é executado.

c. Se o Administrador deseja excluir um motorista já cadastrado, o subfluxo “Deletar Motorista” é executado.

d. Se o Administrador deseja consultar informações sobre um motorista cadastrado, o subfluxo “Consultar Motorista” é executado.

Subfluxo Inserir Motorista

Este subfluxo inicia quando o Administrador solicita incluir um motorista;

O sistema solicita ao Administrador o preenchimento dos seguintes atributos;

Nome do motorista *

Sexo * (campo de escolha fechada, valores possíveis: feminino e masculino)

CPF *

RG *

CNH *

Data de nascimento *

Naturalidade

Nome do pai

Nome da mãe

Grau de escolaridade *

Estado civil

E-mail *

Rua

Bairro

Cidade

Estado

Forma de pagamento *

3. O Administrador preenche os atributos e confirma a inclusão;

4. O sistema realiza a inclusão dos dados informados pelo Administrador;

5. O sistema exibe uma mensagem informando que a inclusão do motorista foi efetivada com sucesso;

(*) atributos obrigatórios.

Subfluxo Atualizar Motorista

Este subfluxo inicia quando o Administrador solicita alterar um motorista;

O Administrador seleciona um único motorista;

O sistema solicita a alteração dos atributos listados no passo 2 do subfluxo “Incluir Motorista”.

O Administrador altera os dados desejados e confirma a alteração;

O sistema realiza a alteração dos dados informados no passo 4;

O sistema exibe uma mensagem de confirmação informando que a alteração do motorista foi efetivada com sucesso;

Subfluxo Deletar Motorista

Este subfluxo inicia quando o Administrador solicita remover um motorista;

O Administrador seleciona qual motorista deseja remover e solicita a remoção;

O sistema solicita a confirmação para remoção;

O Administrador confirma a remoção;

O sistema remove o motorista confirmado;

O sistema exibe uma mensagem informando que a remoção do motorista foi efetivada com sucesso;

Subfluxo Consultar Motorista

Este subfluxo inicia quando o Administrador solicita consultar motorista;

O sistema solicita o preenchimento dos seguintes filtros:
- nome e/ou CPF;

O Administrador preenche os filtros e solicita a consulta;

O sistema apresenta as seguintes informações do motorista obtidos na consulta:

Nome do motorista , Sexo, CPF, RG, CNH,Data de nascimento.

Valiações e regras de negócio

Esta regra se aplica a todos os subfluxos. Atributos obrigatórios. Se algum atributo obrigatório não tiver sido preenchido, o sistema não completará a operação e notificará ao Administrador, informando quais campos obrigatórios não foram preenchidos e solicitando o preenchimento dos mesmos;

Esta regra se aplica a todos os subfluxos. Atributos com valores não permitidos. Se algum atributo for preenchido com valor não permitido, o sistema não completará a operação e notificará ao Administrador, informando quais campos foram preenchidos com valores inválidos e solicitando o preenchimento correto;

No subfluxo **Remover**, somente passará o motorista para o estado de inativo. O sistema valida o motorista selecionado de acordo com as seguintes regras:

Motorista que tiver algum salário pendente não poderá ser removido.

O Administrador não poderá ser removido por ele mesmo.

Motorista que tiver ativo não poderá ser removido.

Como só subfluxos foram feitos separados os diagramas também foram feitos da mesma forma. A primeira figura desta sequência a figura 8 demonstra as chamadas agrupadas de todas as funções.

Figura 8: Manter motorista

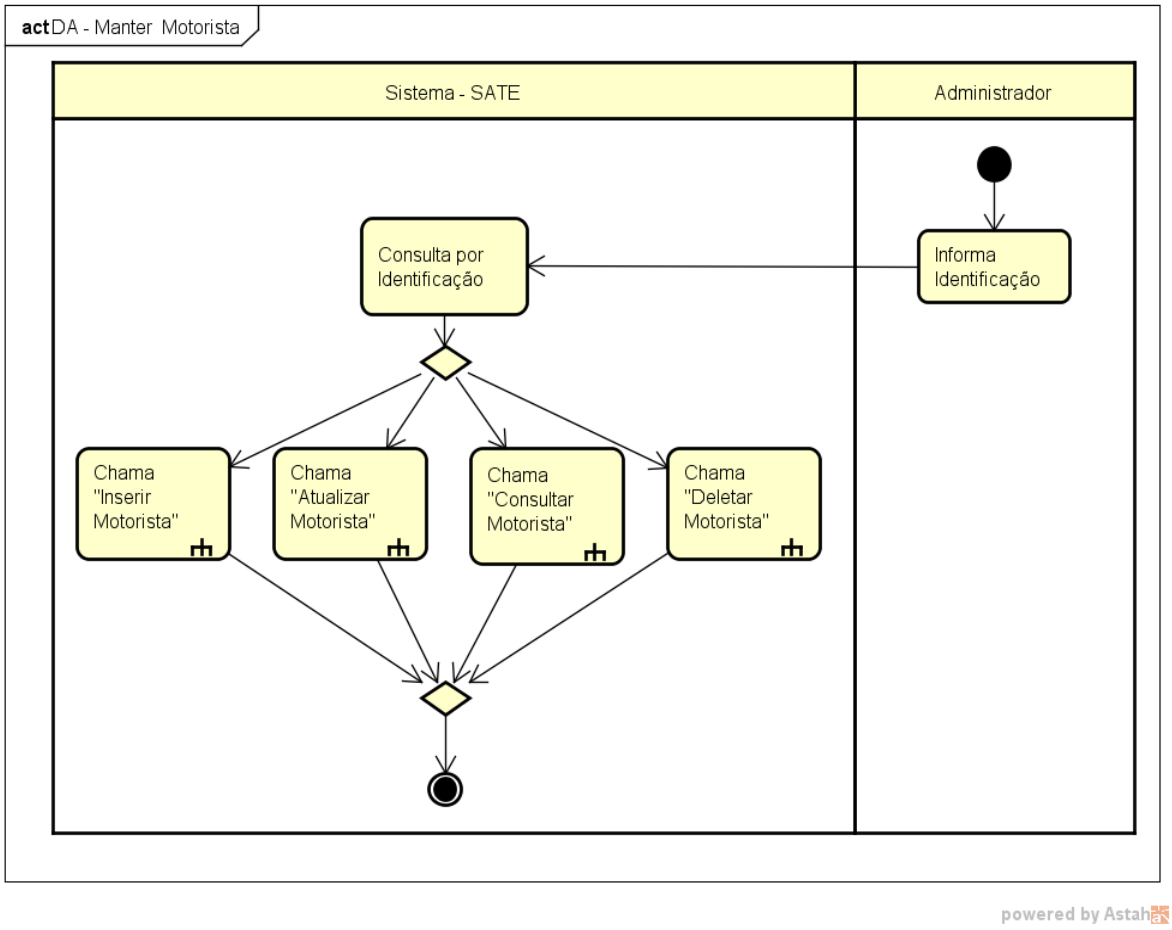


Figura 9: Inserir motorista

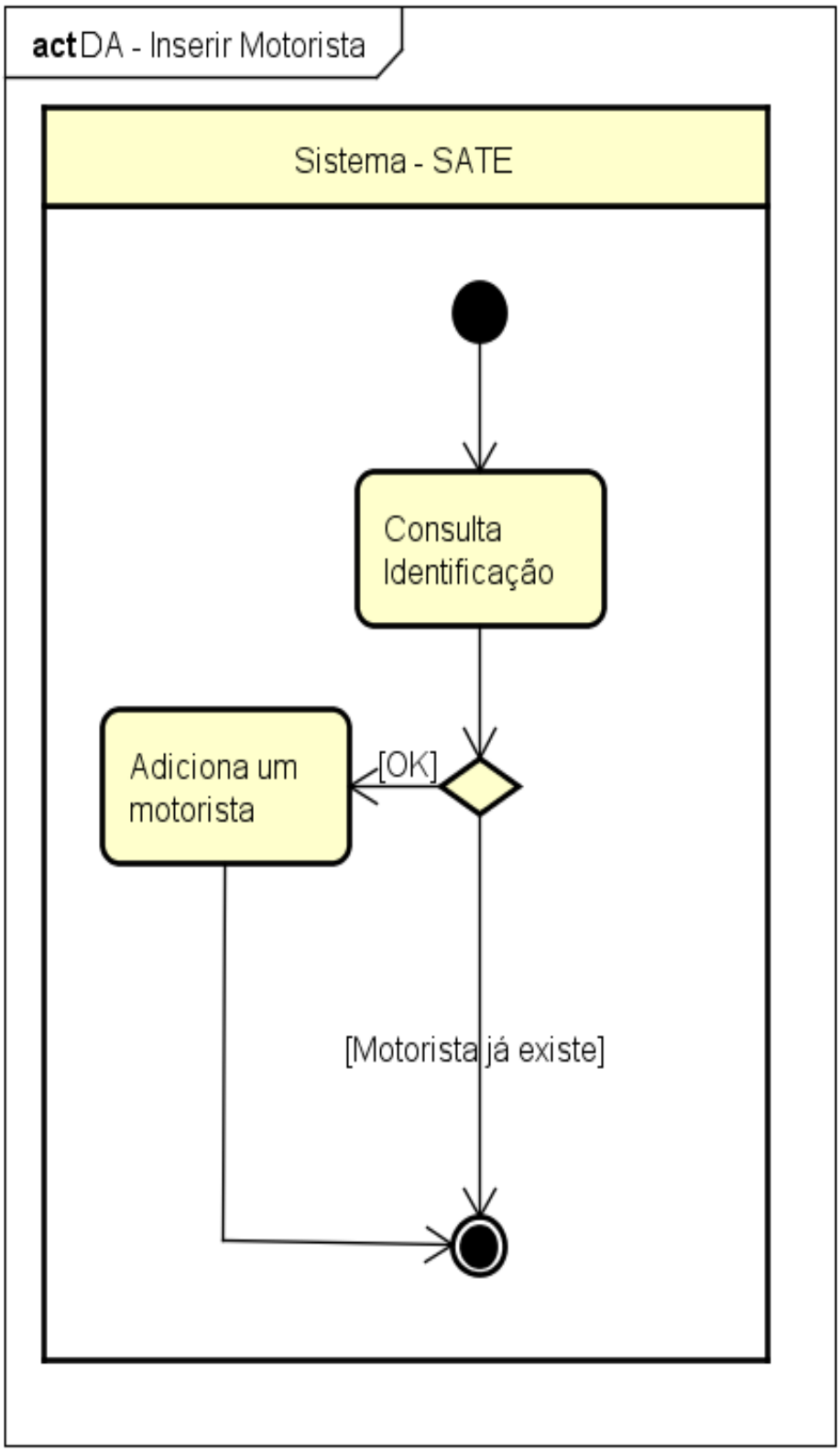


Figura 10: Atualizar motorista

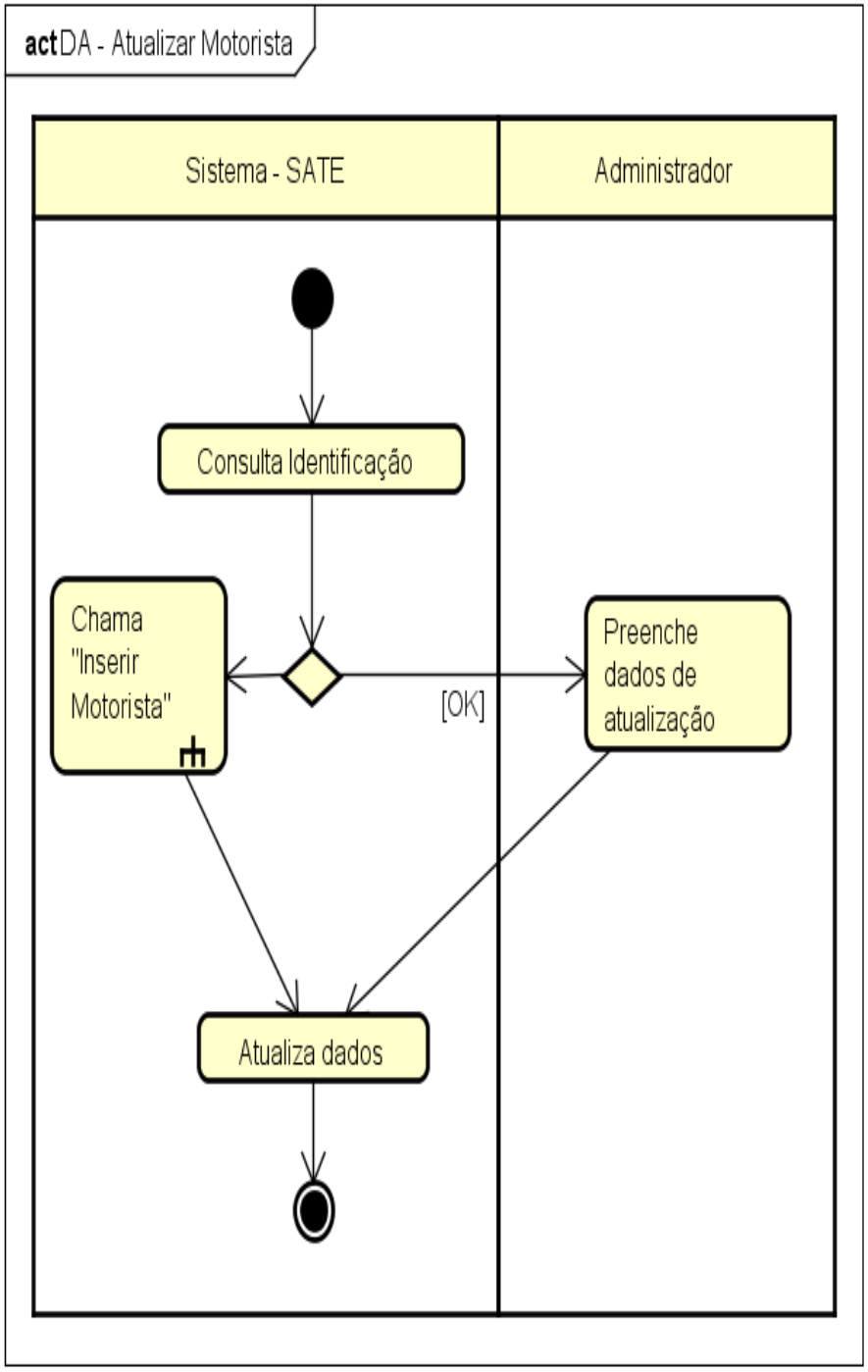


Figura 11: Deletar motorista

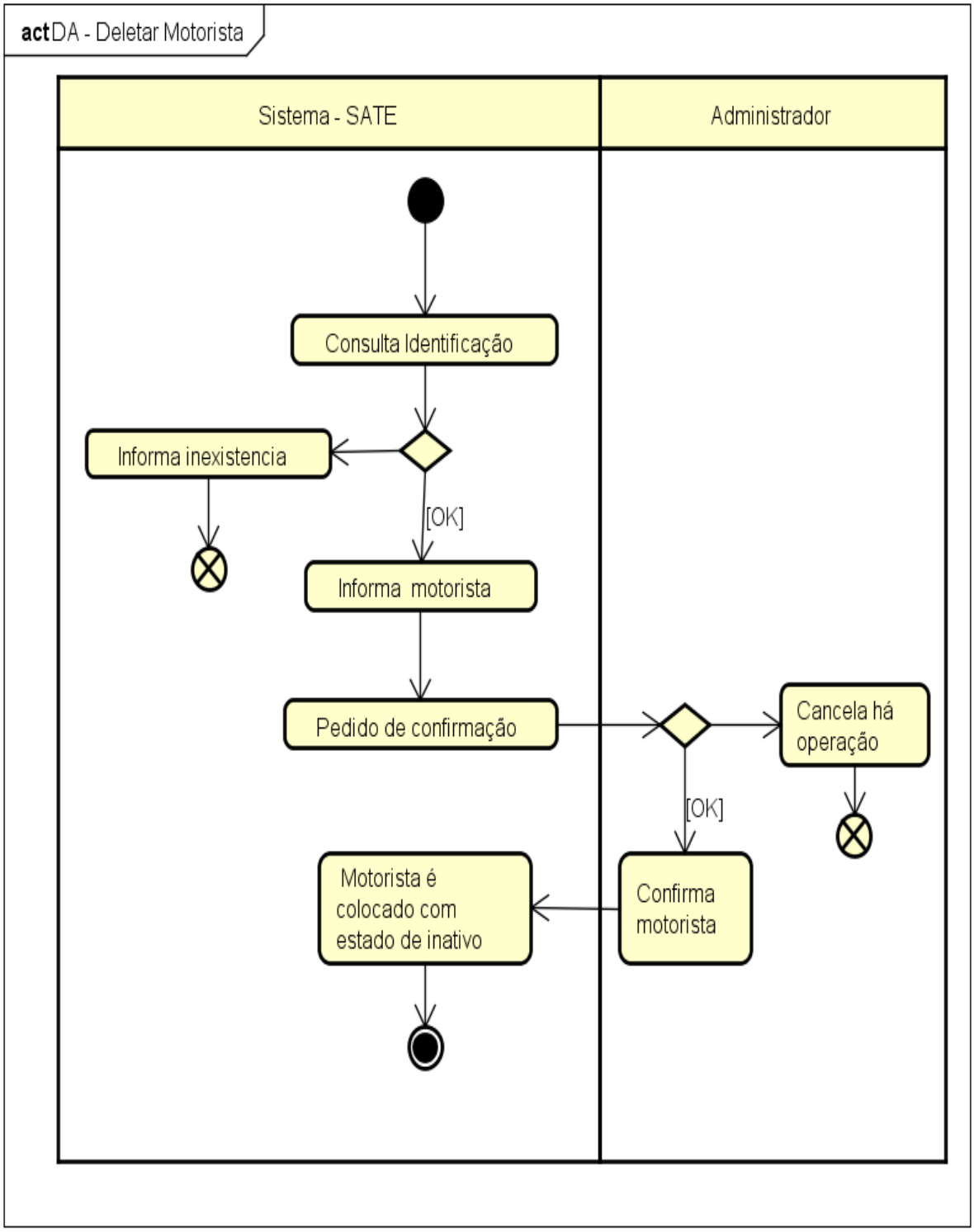
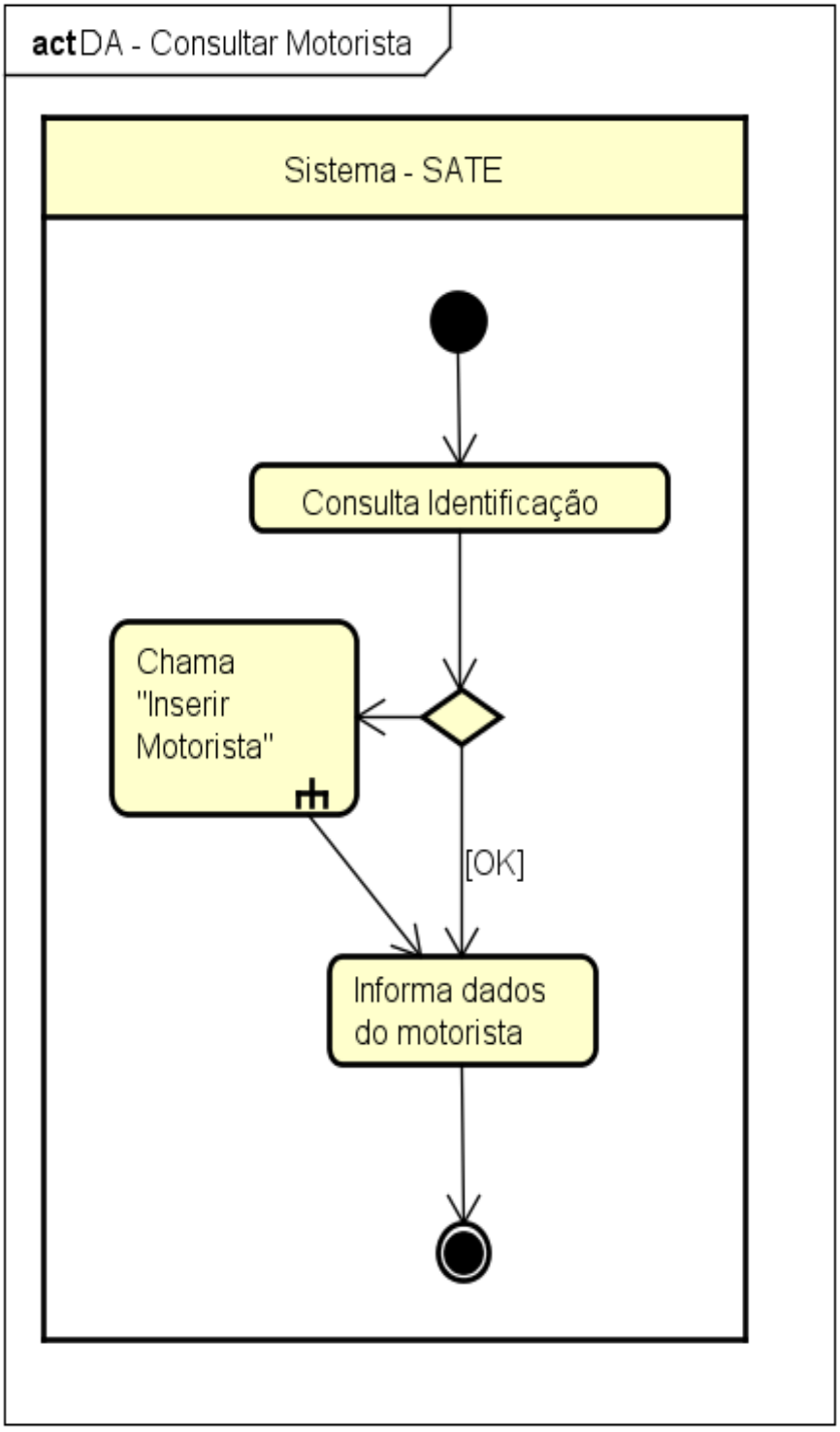


Figura 12: Consultar motorista

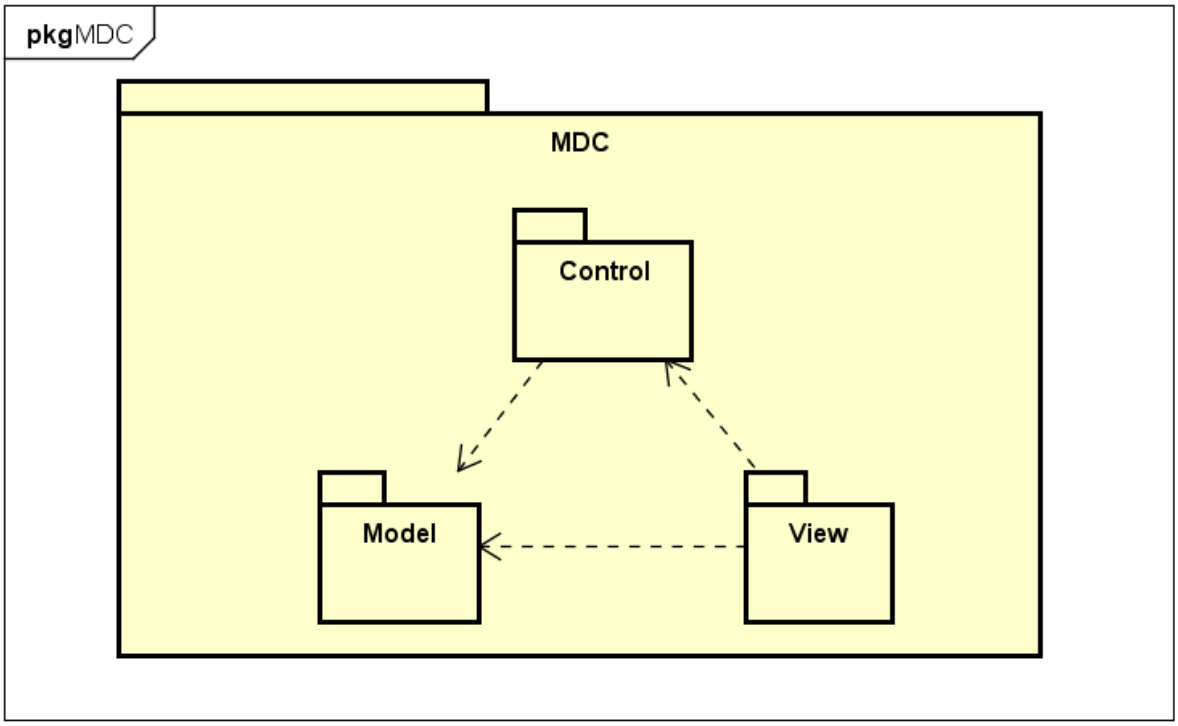


3 Projeto de Software

3.1 Arquitetura Lógica de Software

A arquitetura utilizada é a MVC, mas não a convencional e sim uma adaptada para este projeto, onde a *View* (classe de visual do projeto) pode instanciar tanto as classes *Model* (classes dos objetos de manipulação) quanto as de *Control* (classes de controle e de manipulação do negócio) e a classe *Control* só pode instanciar e manipular as classes *Model.* Uma representação do MVC utilizado é demonstrado na figura 13.

Figura 13: Modelo MVC (*Model, View, Control*) do projeto



3.2 Diagrama de Classes

O projeto tem dois tipos de diagrama de classe, pois utiliza o estilo de arquitetura de software MVC, em conjunto com o *framework JPA com Hibernate*. Os diagramas de classe feitos foram: Modelo (*Model*) e *Controle (Control)* respectivamente representados dos diagramas a seguir como figura 14 e 15.

O diagrama de classe da Visão (*View*) não será implementado pois não houve necessidade.

Figura 14: Diagrama de classe modelo

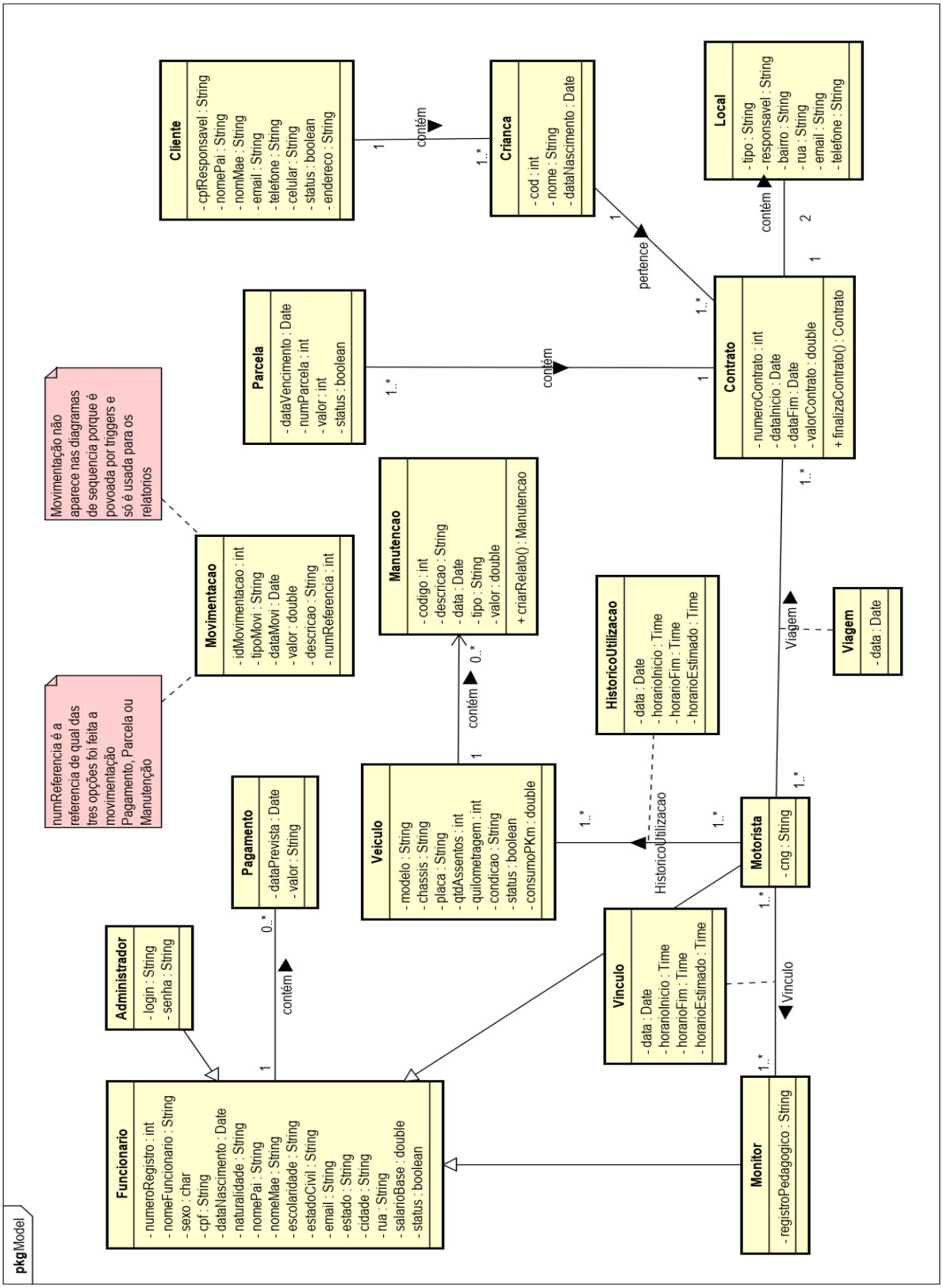
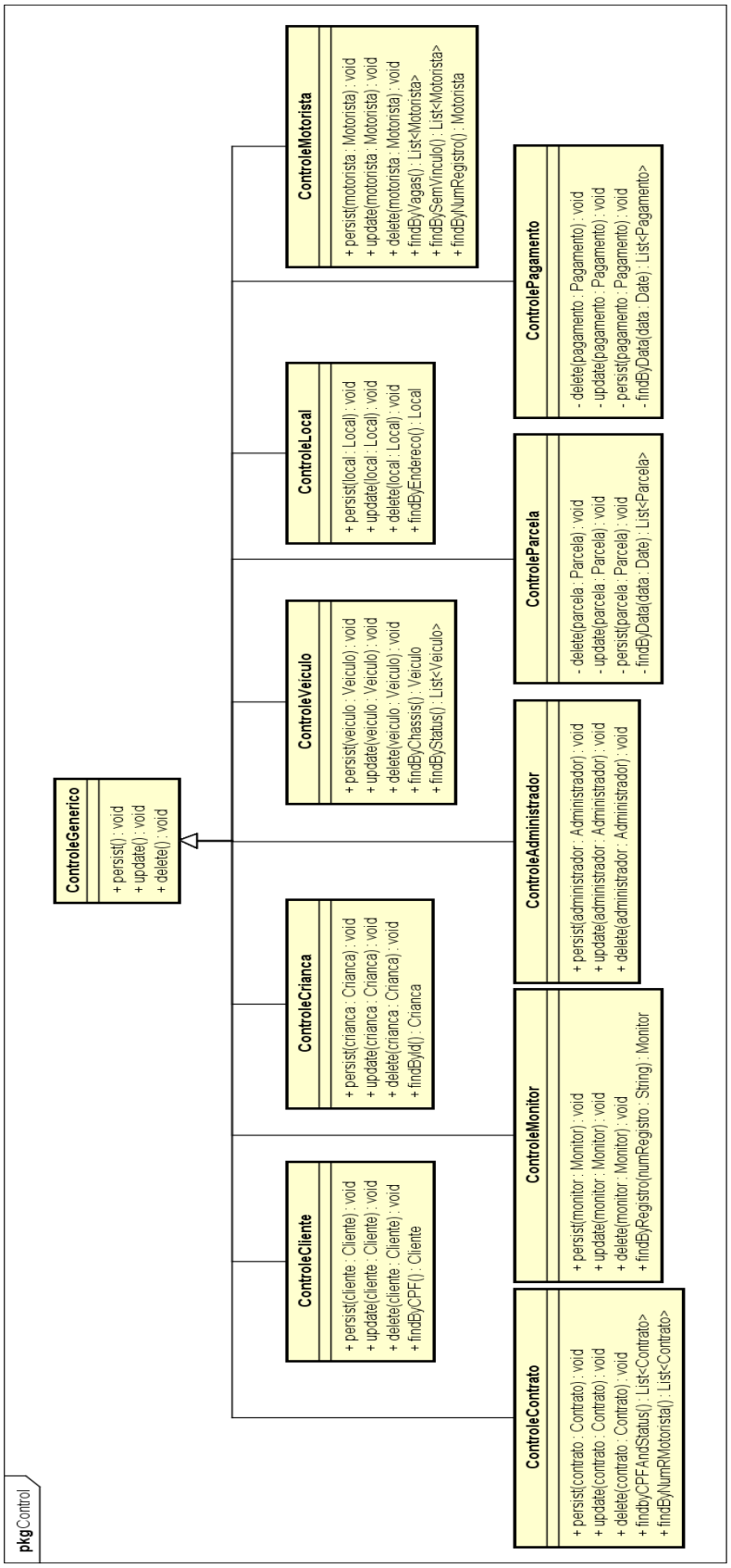


Figura 15: Diagrama de classe controle

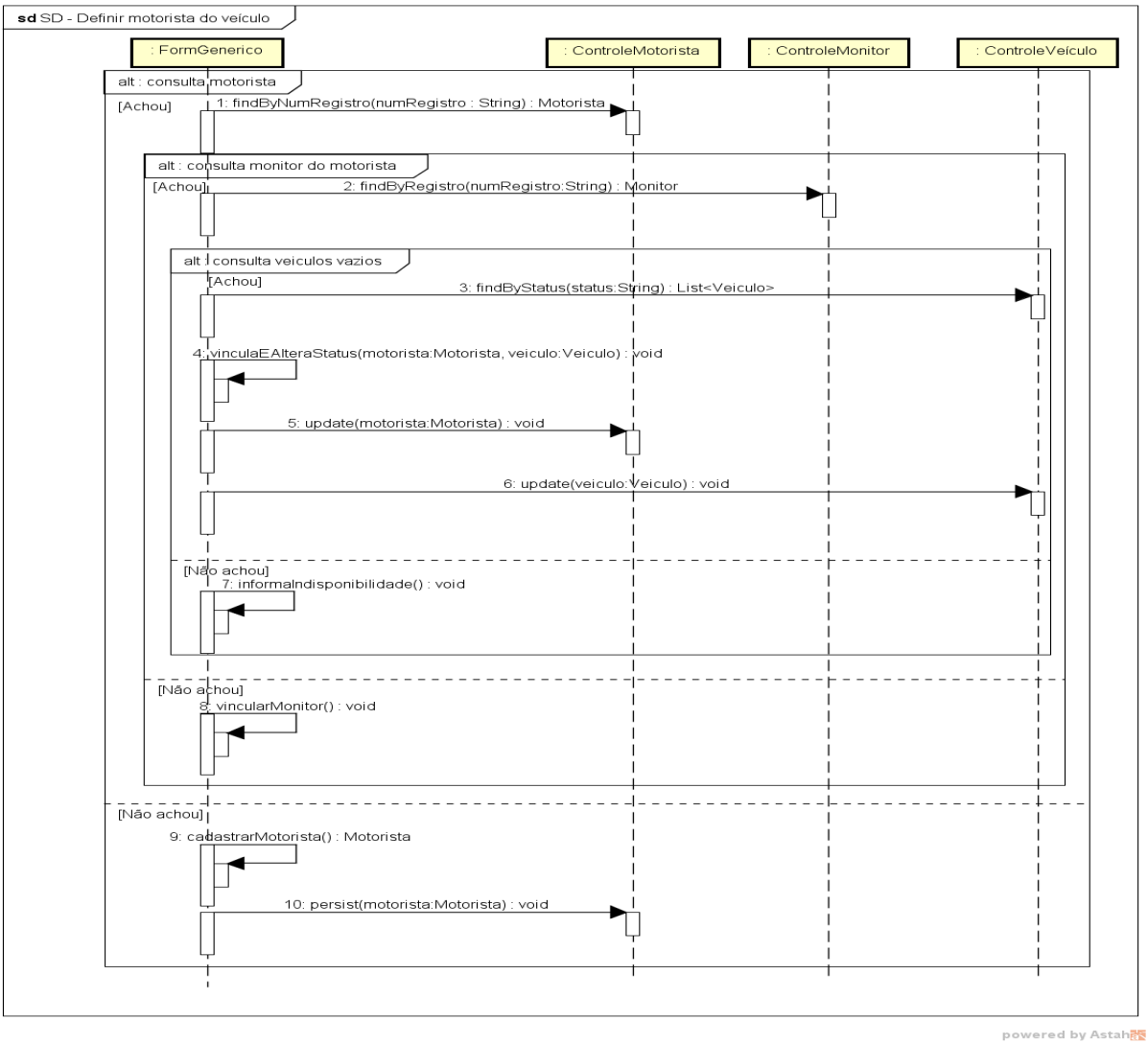


3.3 Diagramas de Sequência

Diagrama de sequência é utilizado para demonstrar por meio de diagramas uma abstração mais elevada o comportamento (envio e recebimento de mensagens) de uma função fundamental do sistema. O diagrama é de certa forma um apoio ao programador, ao qual, pode visualizar o comportamento de uma função em abstração mais alta.

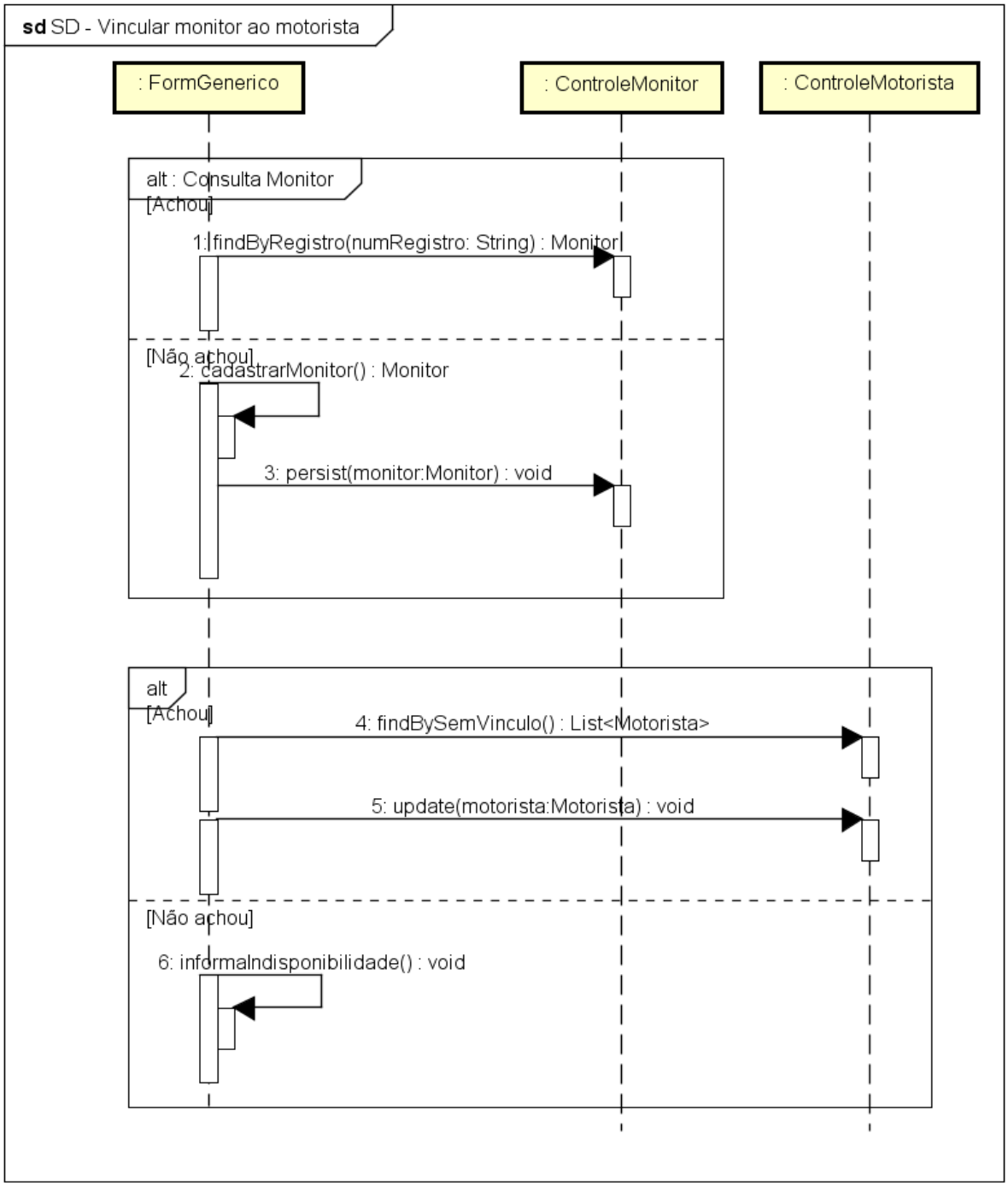
Para cada uma das figuras de diagrama será ancorada uma função em que a mesma demonstra a sequência que à compõem, as funções estão presentes no capítulo “1.5.1.Funções Fundamentais” deste documento A figura 16 mostra a sequência de ativações da função “Definir motorista do veículo” que tem o objetivo de atribuir a um motorista um veículo por um determinado tempo.

Figura 16: Diagrama de sequência – Definir motorista do veículo



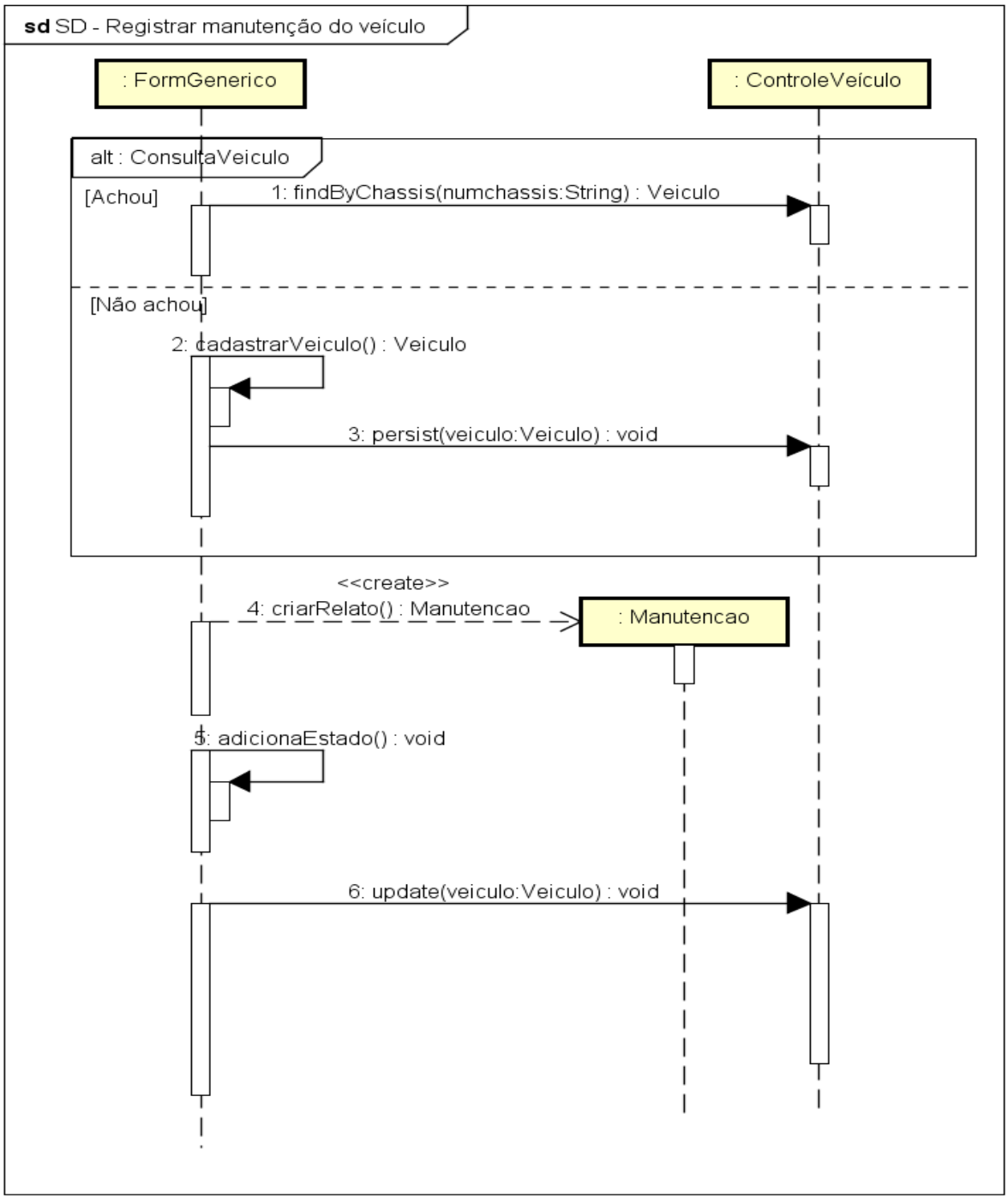
A figura 17 mostra a sequência de ativações da função “Vincular monitor ao motorista”. Essa função tem como objetivo vincular um motorista e um monitor em uma data específica.

Figura 17: Diagrama de sequência – Vincular monitor ao motorista



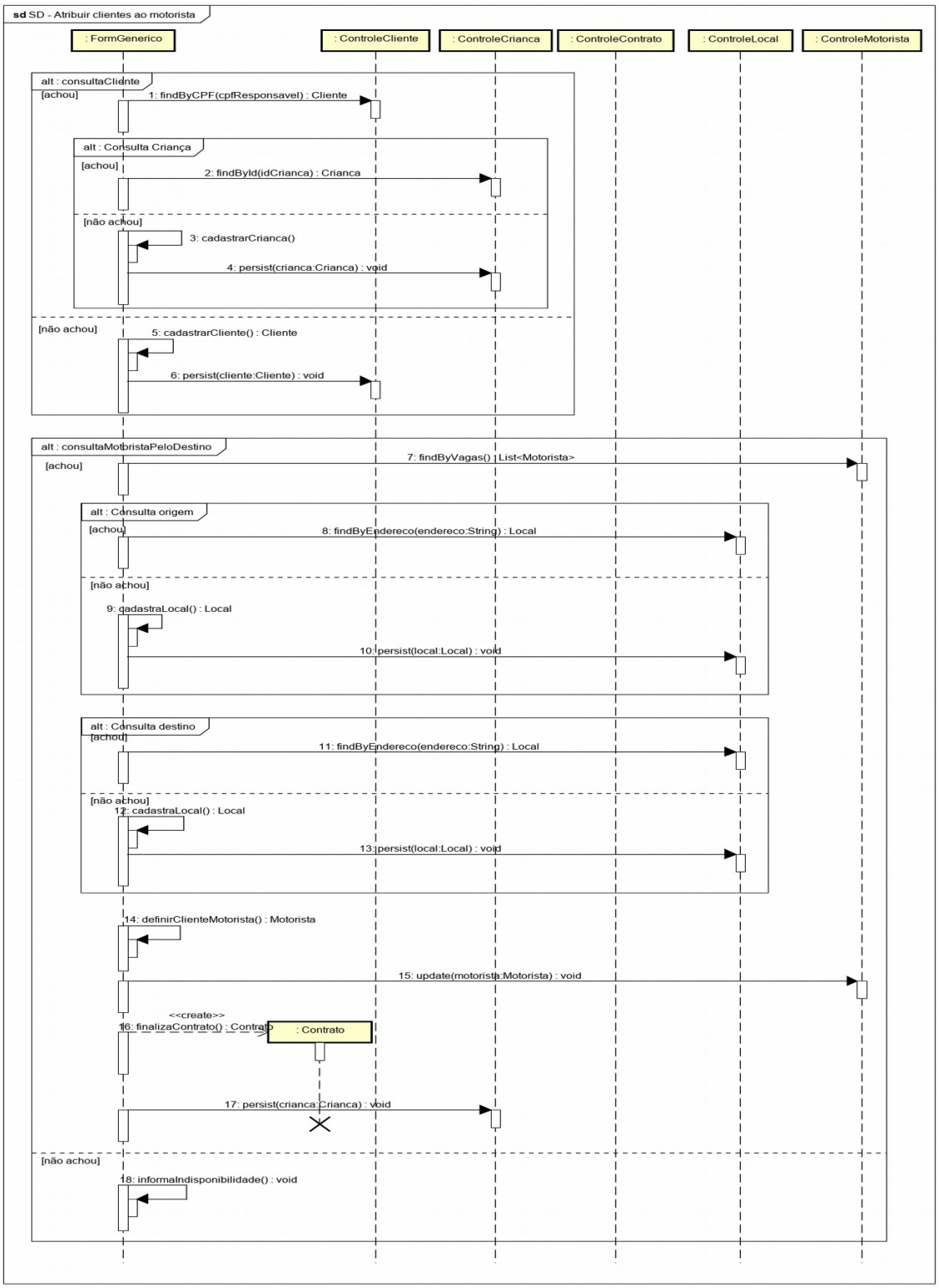
A figura 18 demonstra a sequência de ativações da função “Registrar manutenção do veículo” que tem como objetivo registrar a saída de dinheiro como o histórico de manutenção de um veículo.

Figura 18: Diagrama de sequência – Registrar manutenção do veículo



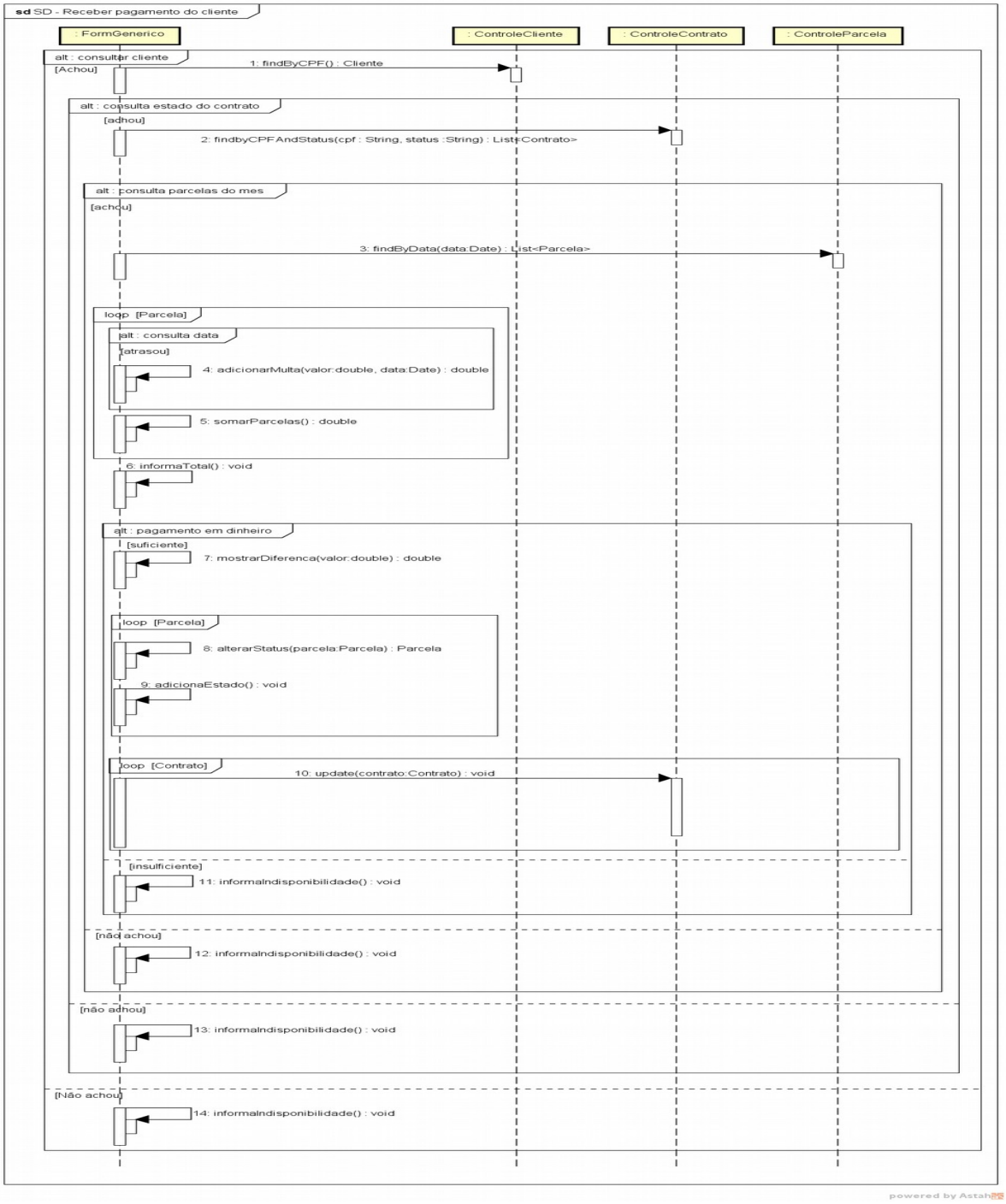
A figura 19 apresenta a sequência de ativações da função “Atribuir clientes ao motorista” que tem como objetivo atribuir um cliente na lista de clientes de um motorista.

Figura 19: Diagrama de sequência – Atribuir clientes ao motorista



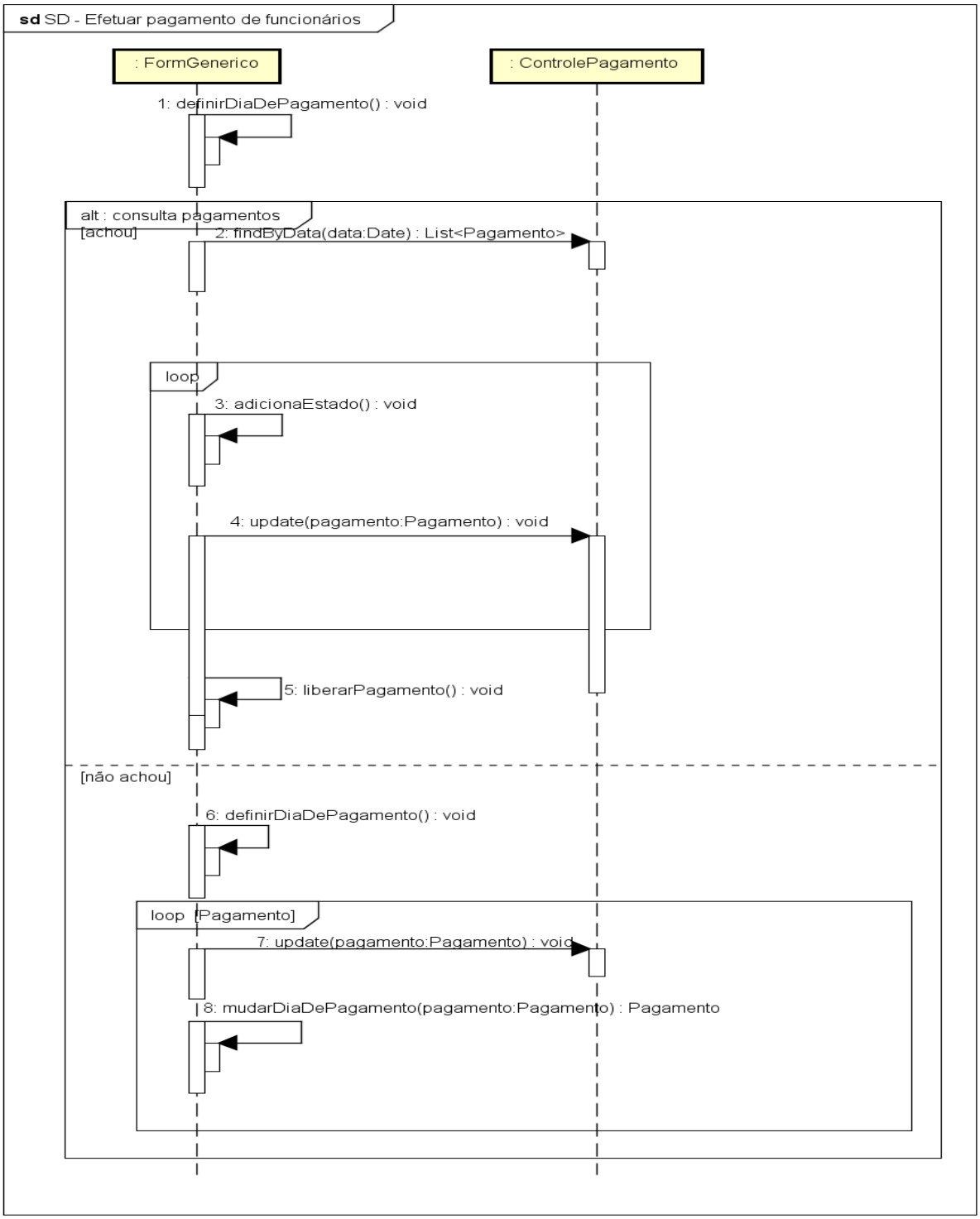
A figura 20 mostra a sequência de ativações da função “Receber pagamento do cliente” que tem como objetivo registrar o pagamento de uma fatura do cliente permitindo que depois de regsitrado possa-se gerar relatorios.

Figura 20: Diagrama de sequência – Receber pagamento do cliente



A figura 21 exibe a sequência de ativações da função "Efetuar pagamento de funcionários" que tem como objetivo registrar o pagamento do salario dos funcionários permitindo que depois de regsitrado possa-se gerar relatorios.

Figura 21: Diagrama de sequência – Efetuar pagamento de funcionários



3.4 Modelo Físico de Dados com as TRIGGERS e Funções

O modelo físico de dados (Diagrama do Banco de dados) foi gerado a partir das ferramentas do software MySQL Workbench. O modelo apresenta as relações das tabelas que são representativamente classes do sistema, assim fazendo um paralelo com o diagrama de classes modelo já apresentado, pode-se observar a conformidade entre os dois diagramas.

As Triggers são uma maneira de economizar processamento do sistema e passá-lo ao banco de dados, com isso em mente foram feitas quatro triggers para gerar dados importantes. Na figura 22 e 23 é apresentado três triggers que geram movimentações de pontos diferentes, sendo parcelas, pagamentos e movimentações.

Figura 22: Triggers para gerar movimentações P1

```
-- Gera movimentação ao pagar uma parcela --
DROP TRIGGER IF EXISTS `db_projeto`.`parcela_AFTER_UPDATE_WRONG_SCHEMA`;

DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `parcela_AFTER_UPDATE`
AFTER UPDATE ON `parcela` FOR EACH ROW

BEGIN
    IF(NEW.status = 'PG') THEN
        INSERT INTO movimentacao (dataMovi, descricao, tipoMovi, valor, numReferencia)
            VALUES (current_date(), 'PAGAMENTO DE UMA PARCELA DO CONTRATO', 'PARCELA', NEW.valor, NEW.numParcela);
    END IF;
END;$$
DELIMITER;

-- Gera movimentação ao liberar pagamento dos funcionários --
DROP TRIGGER IF EXISTS `db_projeto`.`pagamento_AFTER_INSERT`;

DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `db_projeto`.`pagamento_AFTER_INSERT`
AFTER INSERT ON `pagamento` FOR EACH ROW
BEGIN
    IF(NEW.status = 'PG') THEN
        INSERT INTO movimentacao (dataMovi, descricao, tipoMovi, valor, numReferencia)
            VALUES (current_date(), 'PAGAMENTO DO SALARIO DE UM FUNCIONARIO', 'PAGAMENTO', NEW.valor, NEW.idPagamento);
    END IF;
END$$
DELIMITER ;
```

Figura 23: Triggers para gerar movimentações P2

```
-- Gera movimentação ao pagar uma parcela --
DROP TRIGGER IF EXISTS `db_projeto`.`parcela_AFTER_UPDATE_WRONG_SCHEMA`;

DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `parcela_AFTER_UPDATE`
AFTER UPDATE ON `parcela` FOR EACH ROW

BEGIN
    IF(NEW.status = 'PG') THEN
        INSERT INTO movimentacao (dataMovi, descricao, tipoMovi, valor, numReferencia)
            VALUES (current_date(), 'PAGAMENTO DE UMA PARCELA DO CONTRATO', 'PARCELA', NEW.valor, NEW.numParcela);
    END IF;
END;$$
DELIMITER;

-- Gera movimentação ao liberar pagamento dos funcionários --
DROP TRIGGER IF EXISTS `db_projeto`.`pagamento_AFTER_INSERT`;

DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `db_projeto`.`pagamento_AFTER_INSERT`
AFTER INSERT ON `pagamento` FOR EACH ROW
BEGIN
    IF(NEW.status = 'PG') THEN
        INSERT INTO movimentacao (dataMovi, descricao, tipoMovi, valor, numReferencia)
            VALUES (current_date(), 'PAGAMENTO DO SALARIO DE UM FUNCIONARIO', 'PAGAMENTO', NEW.valor, NEW.idPagamento);
    END IF;
END$$
DELIMITER ;
```

Na figura 24 é apresentado a triggers que gera parcelas de um contrato baseando-se em usa data de início e fim como parâmetro.

Figura 24: Triggers para gerar parcelas de um contrato

```
-- Gera parcelas ao inserir um contrato com data de inicio e fim --
DROP TRIGGER IF EXISTS `db_projeto`.`contrato_AFTER_INSERT`;
DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `contrato_AFTER_INSERT`
AFTER INSERT ON `contrato` FOR EACH ROW
BEGIN
    DECLARE qtde, contador, veri, i INTEGER;
    DECLARE valorparcela FLOAT;


    SELECT TIMESTAMPDIFF(MONTH,NEW.dataInicio,NEW.dataFim) INTO qtde;
    SET contador = 30;
    SET valorparcela = NEW.valorContrato/qtde;
    SELECT numContrato INTO veri FROM parcela
        WHERE numContrato = new.numContrato AND status= 'PG';
    IF veri IS null THEN
        DELETE FROM parcela WHERE numContrato = new.numContrato;
        SET i = 1;

        WHILE i <= qtde DO
            INSERT INTO parcela
                VALUES (i, DATE_ADD(CURRENT_DATE, INTERVAL contador DAY), "PD", valorParcela, NEW.numContrato);
            SET contador = contador + 30;
            SET i = i + 1;
        END WHILE;
    END IF;
END$$
DELIMITER ;

DELIMITER $$
USE `db_projeto`$$
CREATE DEFINER = CURRENT_USER TRIGGER `db_projeto`.`contrato_AFTER_UPDATE`
AFTER UPDATE ON `contrato` FOR EACH ROW
BEGIN
 DECLARE qtde, contador, veri, i INTEGER;
    DECLARE valorparcela FLOAT;


    SELECT TIMESTAMPDIFF(MONTH,NEW.dataInicio,NEW.dataFim) INTO qtde;
    SET contador = 30;
    SET valorparcela = NEW.valorContrato/qtde;
    SELECT numContrato INTO veri FROM parcela
        WHERE numContrato = new.numContrato AND status= 'PG';
    IF veri IS null THEN
        DELETE FROM parcela WHERE numContrato = new.numContrato;
        SET i = 1;

        WHILE i <= qtde DO
            INSERT INTO parcela
                VALUES (i, DATE_ADD(CURRENT_DATE, INTERVAL contador DAY), "PD", valorParcela, NEW.numContrato);
            SET contador = contador + 30;
            SET i = i + 1;
        END WHILE;
    END IF;
END$$
DELIMITER ;
```

Na figura 25 pode ser encontrado o modelo físico de dados que foi construído utilizando mapeamento objeto relacional do *framework JPA com Hibernate e depois gerado usando engenharia reversa ferramenta do MySQL Workbench.*

Figura 25: Modelo físico de dados