Universidade Federal de Sergipe
Centro de Ciências Exatas e Tecnologia
DEPARTAMENTO DE COMPUTAÇÃO - DCOMP
CIÊNCIA DA COMPUTAÇÃO

Documentos Desenvolvimento de Software

Prof. Michel dos Santos Soares

São Cristóvão, Sergipe
Outubro – 2016

Componentes:

LUCAS RENATO ARAGÃO SILVA – 201220001325
NOME DO ALUNO B – MATRICULA
NOME DO ALUNO C – MATRICULA
NOME DO ALUNO D – MATRICULA
NOME DO ALUNO E – MATRICULA

Conteúdo

1 Levantamento de Requisitos

1.1 Propósito do Documento

Escrever Propósito do documento

1.2 Escopo do Produto

Escrever escopo do produto

1.3 Definições e Abreviações

1.3.1 Definições

Estudante: pessoa sem dinheiro.

1.3.2 Abreviações

RF: Requisito Funcional.

RNF: Requisito Não Funcional.

1.4 Referências

SOMMERVILLE, I. Engenharia de Software. Pearson/Prentice Hall.

1.5 Visão Geral do Restante do Documento

Escrever visão geral.

1.6 Descrição Geral

1.6.1 Perspectiva do Produto

...

1.6.2 Funções do Produto

...

1.6.3 Características do Usuário

...

User 1: Faz isso.

User 2: Faz aquilo.

1.6.4 Restrições Gerais

...

1.6.5 Suposições e Dependências

...

1.7 Requisitos específicos

1.7.1 Requisitos Funcionais

RF1 Inclusão de fornecedores. (Pr.: 3)

O sistema deve efetuar o cadastro dos fornecedores.

RF2 Alteração de fornecedores. (Pr.: 2)

O sistema deve efetuar a alteração dos dados cadastrais de fornecedores.

RF3 Exclusão de fornecedores. (Pr.: 1)

O sistema deve efetuar a exclusão de fornecedores.

1.7.2 Requisitos Não Funcionais

RNF1 (Pr.: 1): O sistema deve retornar as consultas em, no máximo, 6 segundos, em 90% dos casos.

RNF2 (Pr.: 1): O sistema deve retornar as consultas em, no máximo, 6 segundos, em 90% dos casos.

2 Plano de Projeto

Descrever Plano de Projeto

2.1 Motivação

Motivação para o projeto:

- Motivação 1
- Motivação 2
- Motivação 3

3 Casos de Uso

Nome: ESCREVER NOME.
Descrição: ESCREVER DESCRIÇÃO.
Identificador: ESCREVER IDENTIFICADOR.
Importância: ESCREVER IMPORTÂNCIA.
Ator Primário: ESCREVER ATOR PRIMÁRIO.
Fluxo Principal:

Sistema	Gerente	Funcionário
	1 - Ação	
2 - Ação		
	3 - Ação	
		4 - Ação
	5 - Ação	
6 - Ação		

Nome: ESCREVER NOME.
Descrição: ESCREVER DESCRIÇÃO.
Identificador: ESCREVER IDENTIFICADOR.
Importância: ESCREVER IMPORTÂNCIA.
Ator Primário: ESCREVER ATOR PRIMÁRIO.
Pré-condições: ESCREVER PRÉ-CONDIÇÕES.
Fluxo Principal:

Sistema	Gerente	Funcionário
	1 - Ação	
2 - Ação		
	3 - Ação	
		4 - Ação
	5 - Ação	
6 - Ação		

4 Codificação JavaFx MVC

4.1 Modelos

4.1.1 Cliente

```java
package opencarshop;

import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Cliente {

    private String CPF;

    private String Nome;

    private Date DataNascimento;

    private String Email;

    private String Telefone1;

    private String Telefone2;

        private static boolean validarCPF(String cpf){


            return false;
        }

        private static boolean validarEmail(String email){

            final String PADRAO_EMAIL =
        "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
        + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

            final Pattern PADRAO = Pattern.compile(PADRAO_EMAIL, Pattern.CASE_INSENSITIVE);

            Matcher casador;
            casador = PADRAO.matcher(email);

            return casador.matches();
        }
}
```

Código 1: Cliente.java

4.1.2 Endereço

```java
package opencarshop;

public class Endereco {

    private String CEP;

    private String Estado;

    private String Cidade;

    private String Bairro;

    private String Rua;

    private int Numero;

    private String Complemento;

    private Character Tipo;

}
```

Código 2: Endereco.java

4.2 Controles

4.2.1 Cliente

```java
package opencarshop;

import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Cliente {

    private String CPF;

    private String Nome;

    private Date DataNascimento;

    private String Email;

    private String Telefone1;

    private String Telefone2;

        private static boolean validarCPF(String cpf){


            return false;
        }

        private static boolean validarEmail(String email){

            final String PADRAO_EMAIL =
        "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
        + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

            final Pattern PADRAO = Pattern.compile(PADRAO_EMAIL, Pattern.CASE_INSENSITIVE);

            Matcher casador;
            casador = PADRAO.matcher(email);

            return casador.matches();
        }
}
```

Código 3: Cliente.java

4.2.2 Endereço

```java
package opencarshop;

public class Endereco {

    private String CEP;

    private String Estado;

    private String Cidade;

    private String Bairro;

    private String Rua;

    private int Numero;

    private String Complemento;

    private Character Tipo;

}
```

Código 4: Endereco.java

4.3 Fronteiras

4.3.1 Cliente

```java
package opencarshop;

import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Cliente {
```

```
8
9       private String CPF;
10
11      private String Nome;
12
13      private Date DataNascimento;
14
15      private String Email;
16
17      private String Telefone1;
18
19      private String Telefone2;
20
21          private static boolean validarCPF(String cpf){
22
23
24              return false;
25          }
26
27          private static boolean validarEmail(String email){
28
29              final String PADRAO_EMAIL =
30          "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
31          + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";
32
33              final Pattern PADRAO = Pattern.compile(PADRAO_EMAIL, Pattern.CASE_INSENSITIVE);
34
35              Matcher casador;
36              casador = PADRAO.matcher(email);
37
38              return casador.matches();
39          }
40  }
```

Código 5: Cliente.java

4.3.2 Endereço

```
1   package opencarshop;
2
3   public class Endereco {
4
5       private String CEP;
6
7       private String Estado;
8
9       private String Cidade;
10
11      private String Bairro;
12
13      private String Rua;
14
15      private int Numero;
16
17      private String Complemento;
18
19      private Character Tipo;
20
21  }
```

Código 6: Endereco.java

5 Diagramas

5.1 Diagrama de Classes - Analise

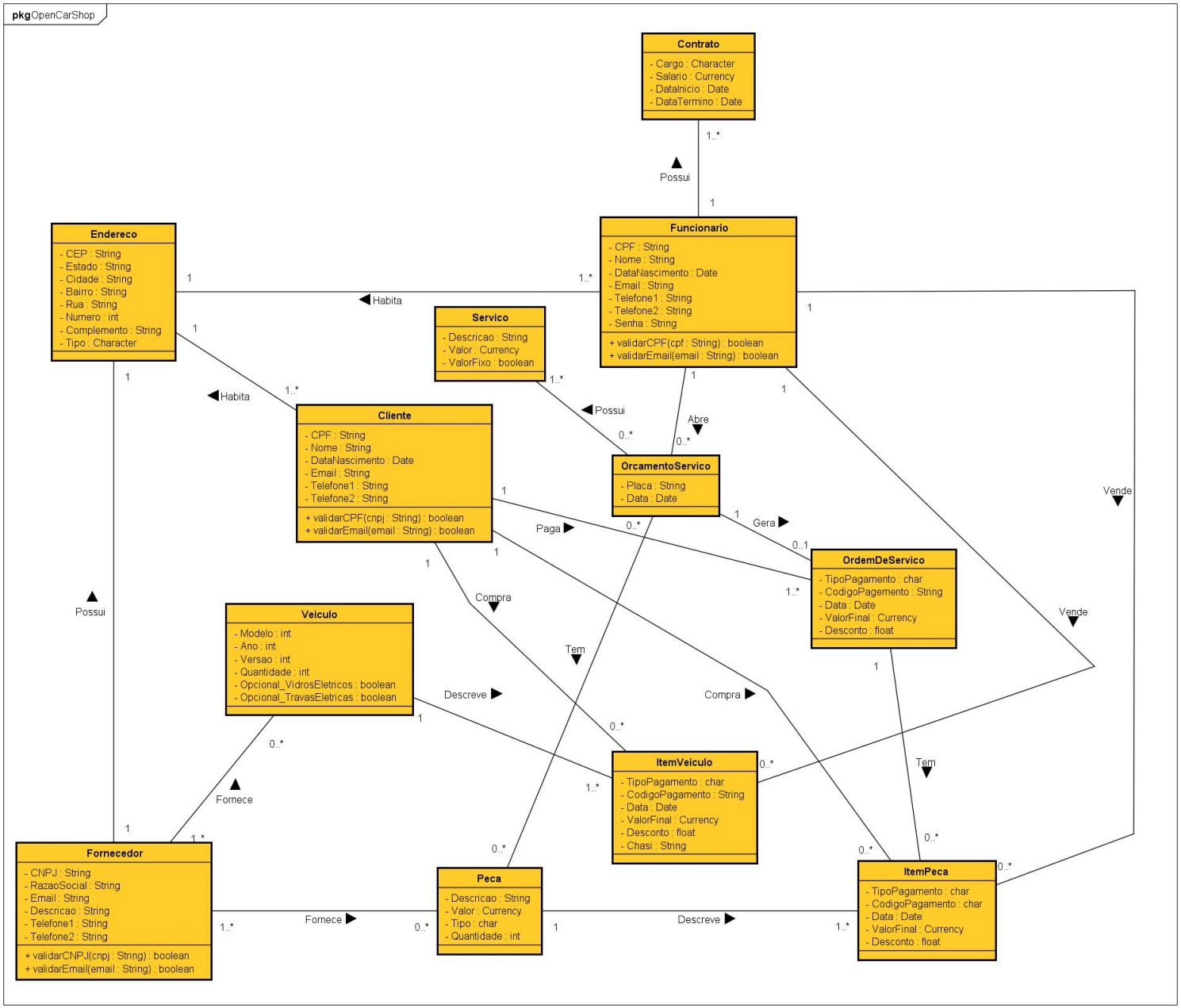

Figura 1: Diagrama de Classes - Analise

5.2 Diagrama de Classes - Projeto

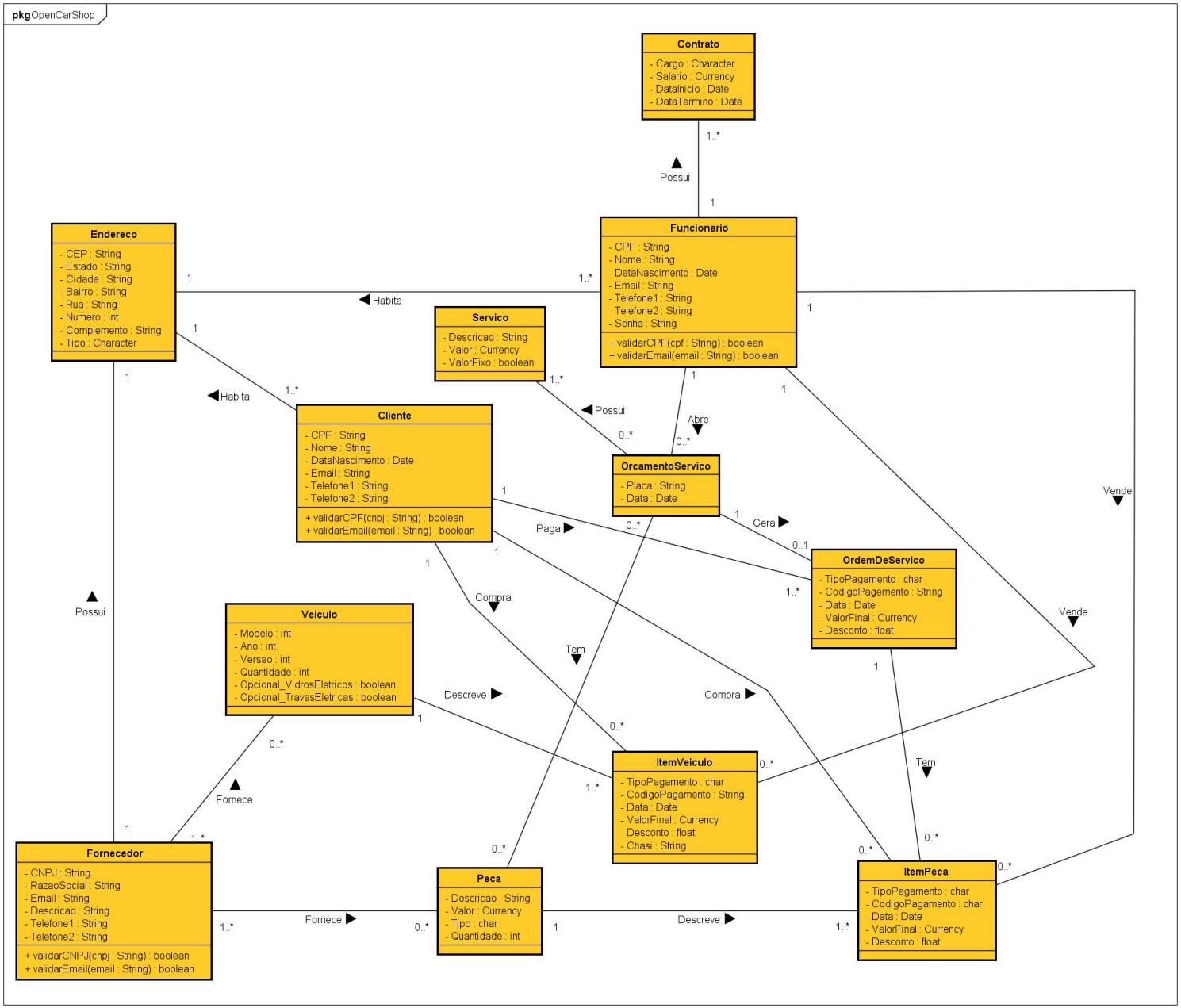

Figura 2: Diagrama de Classes - Projeto

5.3 Diagrama de Casos de Uso

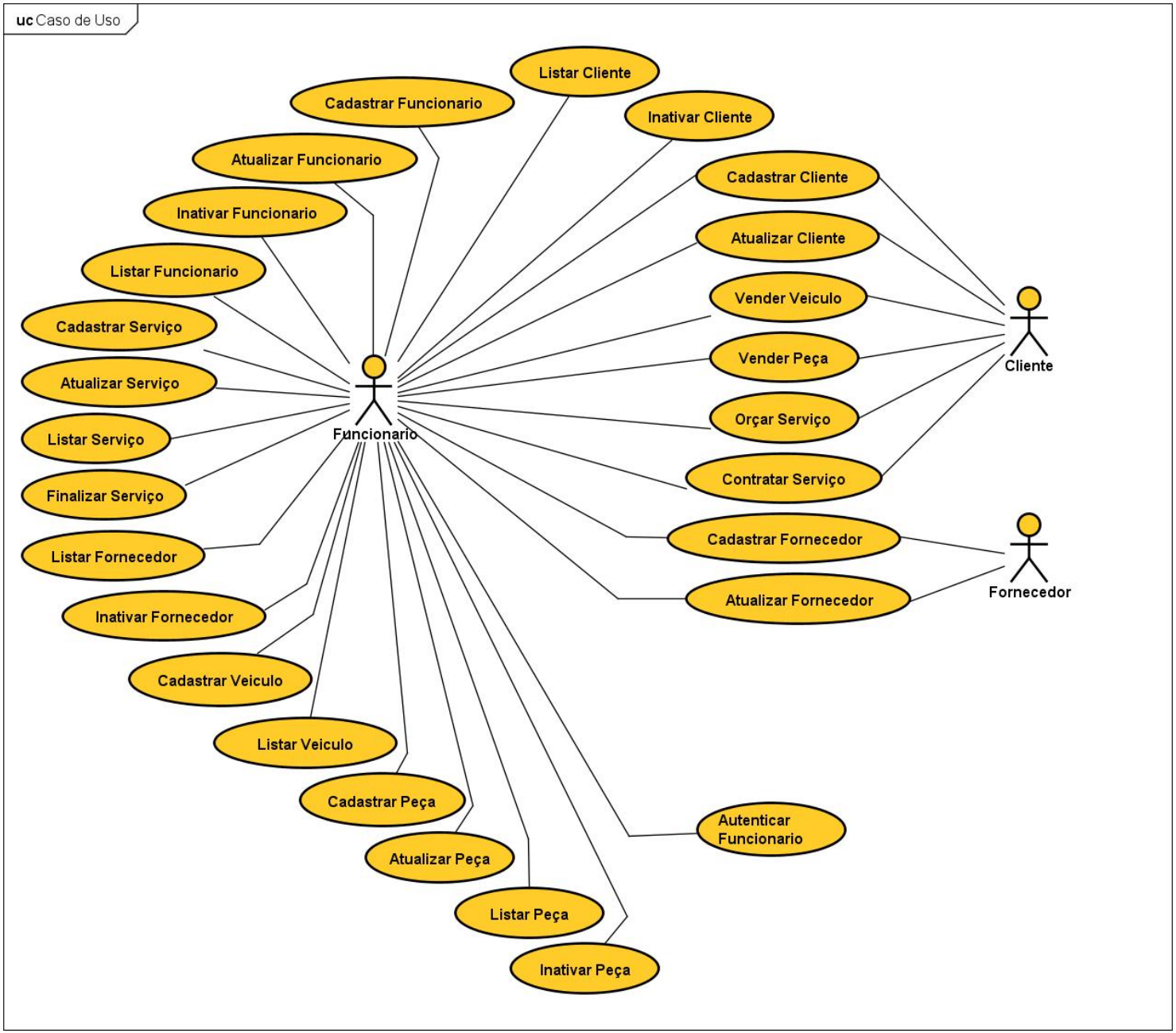

Figura 3: Diagrama de Casos de Uso