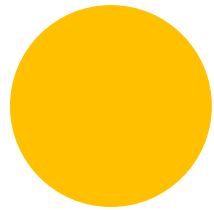
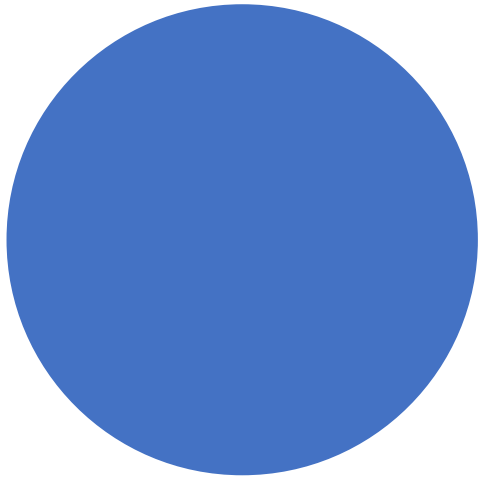


Ciclo de vida do desenvolvimento de software

Danielly Garcia Jardim
Bruna Cristina Lopes
Rone Felipe Bento



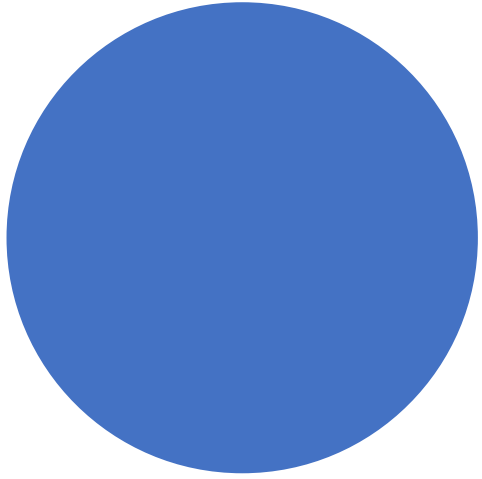
Engenharia de Software Requisitos

“Clotech”

O que são requisitos?

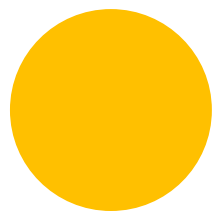
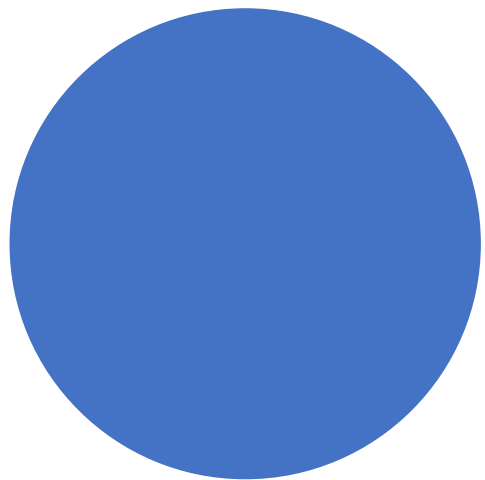
- Requisitos são as **necessidades** do cliente!
- Os requisitos de um software são divididos em 2 categorias:
 - *Requisitos Funcionais*: são as tarefas, as funcionalidades de um sistema – eles são descritos na forma de verbos no infinitivo, pois são ações que os usuários executam em um software (ex: sacar dinheiro, buscar um livro, comprar um produto e etc)
 - *Requisitos Não Funcionais*: são as qualidades que um sistema precisa possuir para atender adequadamente as necessidades do usuário – eles são descritos na forma de adjetivos (ex. Usabilidade, segurança, desempenho e etc)





Atividade de Requisitos

“Clotech”



Requisitos funcionais

“Clotech”

Diagrama de Casos de Uso

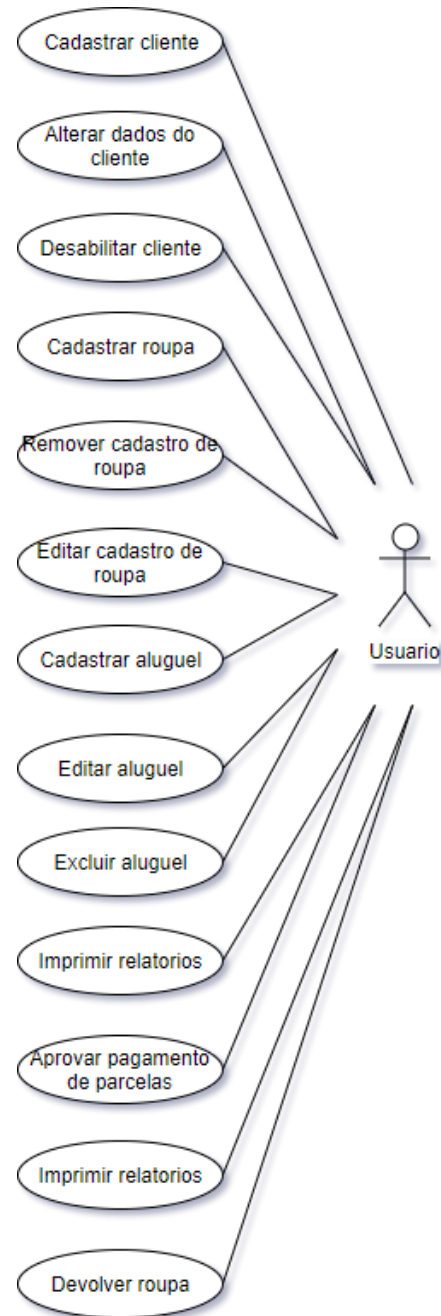


Tabela Cadastro de clientes

Caso de uso	Cadastrar cliente
Ator principal	Funcionário
Resumo	Salvar dados pessoais como: <ul style="list-style-type: none"> - Nome - RG - CPF - Sexo - Data de nascimento - Medidas - Endereço - E-mail - Celular - Telefone residencial - Telefone para recados - Nome dos pais
Pré-condição	Logado
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Clicar no botão de cadastro de cliente	
	2. Exibir formulário de cadastro
3. Preencher o formulário <ul style="list-style-type: none"> - Nome - RG - CPF - Sexo - Data de nascimento - Medidas - Endereço - E-mail - Celular - Telefone residencial - Telefone para recados - Nome dos pais 	
4. Clicar em salvar	
	5. Salvar dados do cliente
Restrições/Validações	3. Validar: <ul style="list-style-type: none"> - CPF (apenas números) - E-mail (texto@.com) - Celular ((xx) 9xxxx-xxxx (Apenas números)) - CEP (apenas números) - Medidas (apenas números) - Data de nascimento (dd/mm/aaaa) - Nome (Apenas letras)
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	3. Caso algum dado seja preenchido incorretamente “Dado invalido, preencha corretamente” em frente ao campo que está incorreto. <ul style="list-style-type: none"> - CPF (apenas números) - E-mail (texto@.com) - Celular ((xx) 9xxxx-xxxx (Apenas números)) - CEP (apenas números) - Medidas (apenas números) - Data de nascimento (dd/mm/aaaa) - Nome (Apenas letras)
3. Verificar os dados que estão incorretos e corrigi-los	
	3. Caso não seja preenchido, exibe a mensagem “Campo obrigatório” em frente ao campo que está em branco <ul style="list-style-type: none"> - CPF (apenas números) - E-mail (texto@.com) - Celular ((xx) 9xxxx-xxxx (Apenas números)) - CEP (apenas números) - Medidas (apenas números) - Data de nascimento (dd/mm/aaaa) - Nome (Apenas letras)
3. Verificar os campos que estão em branco e preenche-los	

Tabela
Cadastro de
roupas

Caso de uso	Cadastrar roupa
Ator principal	Funcionário
Resumo	Cadastrar vestimentas disponíveis
Pré-condição	Logado
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Clicar em cadastrar roupa	
	2. Abrir formulário de cadastro de roupa <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasão- Faixa etária- Tipo de tecido- Quantidade
3. Preencher o formulário <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasão- Faixa etária- Tipo de tecido- Quantidade	
4. Clicar em salvar	
	5. Salvar dados
	6. Gerar QR code específico
	7. Gerar código da roupa
Restrições/Validações	3. Validar: <ul style="list-style-type: none">- Sexo (F/M)- Tipo (Apenas letras)- Quantidade (Apenas numeros)- Cor (apenas letras)- Tamanho (Apenas letras)
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	3. Caso algum dado seja preenchido incorretamente “Dado invalido, preencha corretamente” em frente ao campo que está incorreto. <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasão- Faixa etária- Tipo de tecido- Quantidade
3. Verificar os dados que estão incorretos e corrigi-los	
	3. Caso não seja preenchido, exibe a mensagem “Campo obrigatório” em frente ao campo que está em branco. <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasão- Faixa etária- Tipo de tecido- Quantidade
3. Verificar os dados que estão em branco e corrigi-los	

Tabela Cadastro de aluguel

Caso de uso	Cadastrar aluguel
Ator principal	Funcionário
Resumo	Cadastrar pedido do cliente
Pré-condição	1. Logado 2. Possuir o cadastro da roupa que o cliente deseja alugar 3. Ter o cliente cadastrado
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Digitar na barra de pesquisa o código da roupa	
	2. Buscar no banco o código solicitado
	3. Mostrar roupa correspondente
4. Clicar em formulário de aluguel	
	5. Abrir formulário de aluguel - Nome - Documento - Preço do aluguel da roupa - Definir parcelas - Data de retirada - Data de entrega - Data de locação
6. Preencher formulário de aluguel - Nome - Documento - Preço do aluguel da roupa - Definir parcelas	
7. Clicar em salvar	
	8. Encontrar cliente no banco de dados
	9. Exibir dados do cliente
10. Confirmar dados do cliente	
11. Clicar em confirmar	
	12. Salvar alteração de status da roupa
	13. Gerar contrato de uso
14. Clicar em imprimir contrato	
	15. Gerar carnê
16. Clicar em imprimir carnê	
	17. Gerar recibo
18. Clicar em imprimir recibo	
Restrições/Validações	6. Validar: - Documento (alfanumérico) - Nome (somente letras)
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	2. Caso o código da roupa não seja encontrado exibe a mensagem "Código não encontrado".
	7. Caso o cliente não seja encontrado, exibe a mensagem "Cliente não cadastrado".

Tabela Edição dos dados do cliente

Caso de uso	Alterar dados do cliente
Ator principal	Funcionário
Resumo	Alterar dados do cliente - Nome - Sexo - Medidas - Endereço - E-mail - Celular - Telefone residencial - Telefone para recados - Nome dos pais
Pré-condição	- Logado - Possuir cadastro ativo do cliente
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Clicar em buscar cliente	
	2. Abrir janela de busca de clientes
3. Digitar dados do cliente - Nome - Documento	
4. Clicar em pesquisar	
	5. Localizar cadastro do cliente no banco de dados
	6. Exibir dados do cliente
7. Alterar os dados desejados - Nome - Sexo - Medidas - Endereço - E-mail - Celular - Telefone residencial - Telefone para recados - Nome dos pais	
8. Clicar em salvar	
	9. Salvar dados alterados
Restrições/Validações	7. Validar dados - E-mail (texto@.com) - Celular (apenas números) - CEP (apenas números) - Medidas (apenas números) - Nome (Apenas letras)
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	5. Caso o cliente não seja encontrado, exibir a mensagem "Cliente não cadastrado"

Tabela
Edição dos
dados da
roupa

Caso de uso	Editar cadastro de roupa
Ator principal	Funcionário
Resumo	Editar cadastro das vestimentas disponíveis <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasião- Faixa etária- Tipo de tecido- Quantidade
Pré-condição	1. Logado 2. Possuir o cadastro da roupa
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Digitar na barra de pesquisa o código da roupa	
	2. Buscar no banco o código solicitado
	3. Exibir formulário de cadastro de roupa <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasião- Faixa etária- Tipo de tecido- Quantidade
3. preencher formulário de cadastro de roupa <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasião- Faixa etária- Tipo de tecido- Quantidade	
Restrições/Validações	4. Validar: <ul style="list-style-type: none">- Sexo- Tipo- Quantidade- Cor- Tamanho
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	5. Exibe uma mensagem de erro e não salva no banco de dados, caso não preencha ou preencha incorretamente alguns dos seguintes campos: <ul style="list-style-type: none">- Tipo de roupa- Marca- Tamanho- Sexo- Cor- Estação- Ocasião- Faixa etária- Tipo de tecido- Quantidade
5. Corrigir campos preenchidos incorretamente.	

Tabela Edição dos dados do aluguel

Caso de uso	Editar aluguel
Ator principal	Funcionário
Resumo	Caso o cliente queira adicionar, remover ou trocar alguma das peças anteriormente selecionadas - Preço do aluguel da roupa - Definir parcelas - Código da roupa
Pré-condição	- Logado - Possuir cliente ativo - Possuir aluguel ativo
Fluxo principal	
- Ações do ator	- Ações do sistema
1. Clicar em buscar cliente.	2. Abrir janela de busca de clientes.
3. Digitar dados do cliente - Nome - Documento	
4. Clicar em pesquisar.	
	5. Localizar cadastro do cliente no banco de dados.
	6. Exibir janela com os dados do cliente.
7. Clicar em visualizar aluguéis.	8. Abrir janela com os aluguéis.
9. Localizar na lista ou digitar no campo de pesquisa o número do recibo.	
10. Clicar no aluguel desejado	
	11. Exibir aluguel desejado.
12. Clicar em editar aluguel	
	13. Exibir formulário de edição de aluguel - Preço do aluguel da roupa - Definir parcelas - Código da roupa
14. Fazer alterações necessárias - Preço do aluguel da roupa - Definir parcelas - Código da roupa	
15. Clicar em salvar dados alterados	
	16. Buscar o código da roupa
	17. Armazenar dados - Preço do aluguel da roupa (apenas numeros) - Definir parcelas (apenas numeros) - Código da roupa (alfanumérico)
Restrições/Validações	
Fluxo de Exceção	
- Ações do ator	- Ações do sistema
	5. Caso não localize o cadastro do cliente, exibe a mensagem "Cliente não encontrado".
	8. Caso o cliente não possua aluguéis cadastrados, exibir a mensagem "O cliente não possui alugueis cadastrados".
	16. Caso o código da roupa não seja encontrado, exibe a mensagem "Código da roupa não cadastrado".

Diagrama de sequência Cadastro de cliente

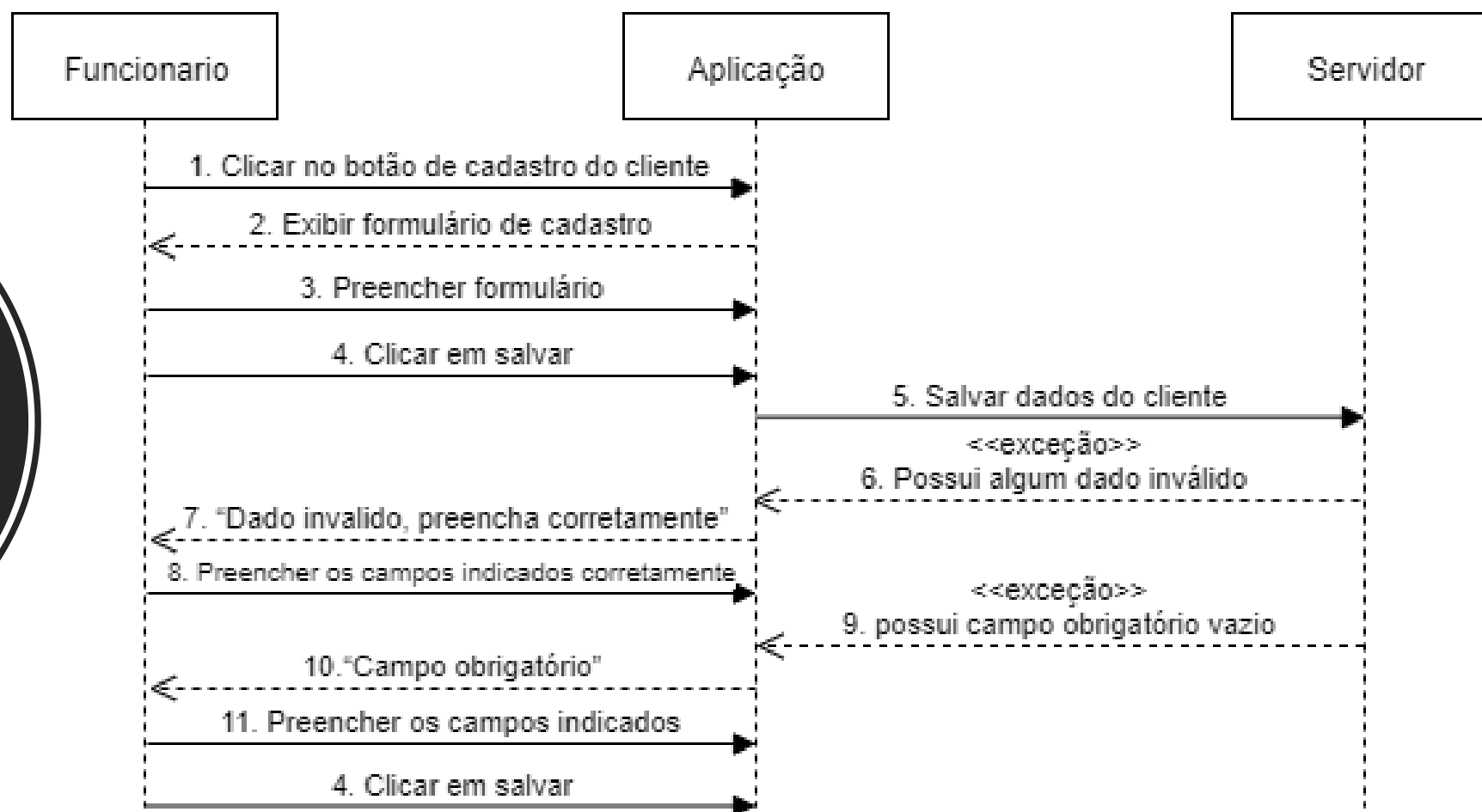


Diagrama de sequência Cadastro de roupas

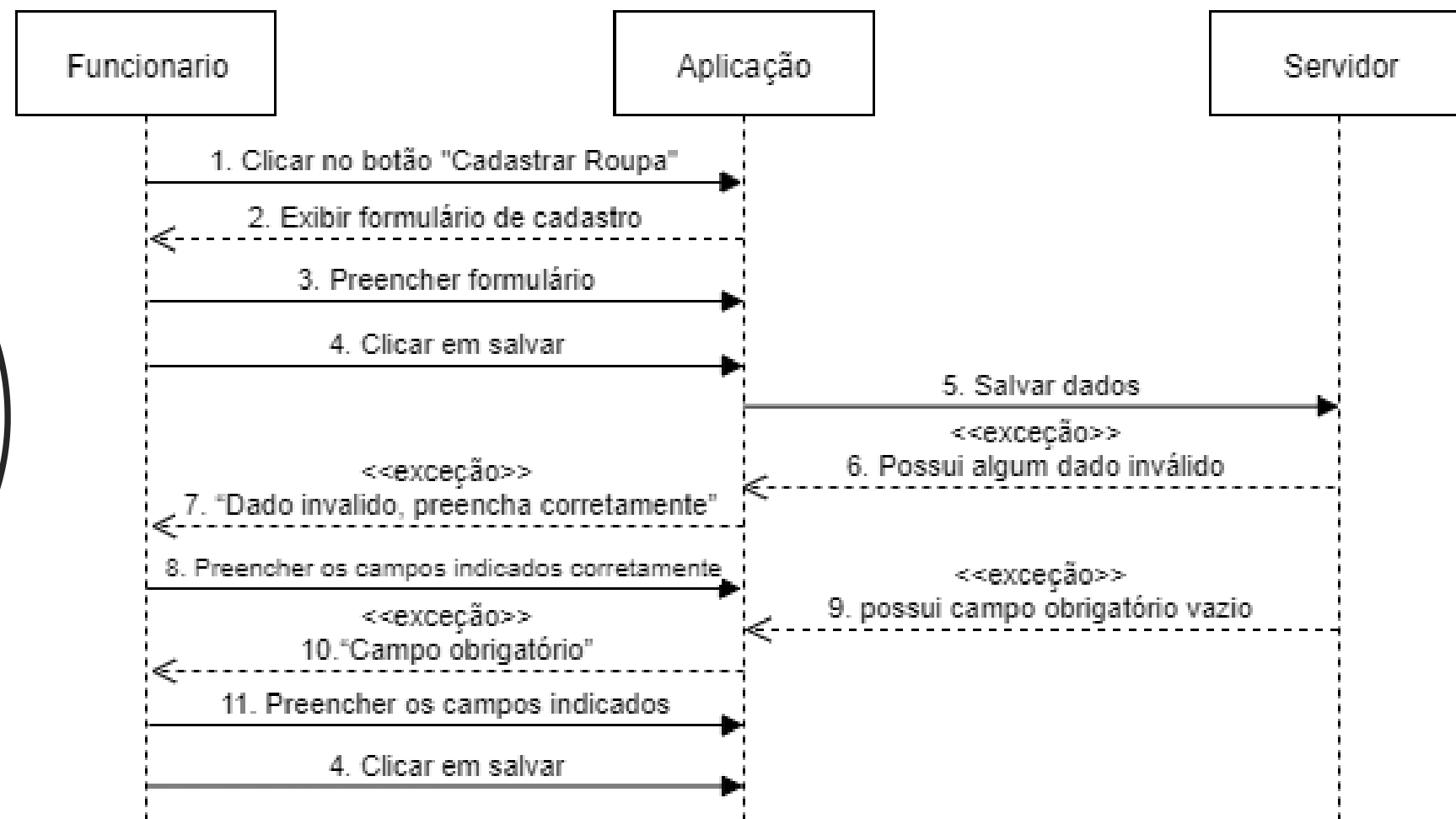


Diagrama de sequência Cadastro de aluguel

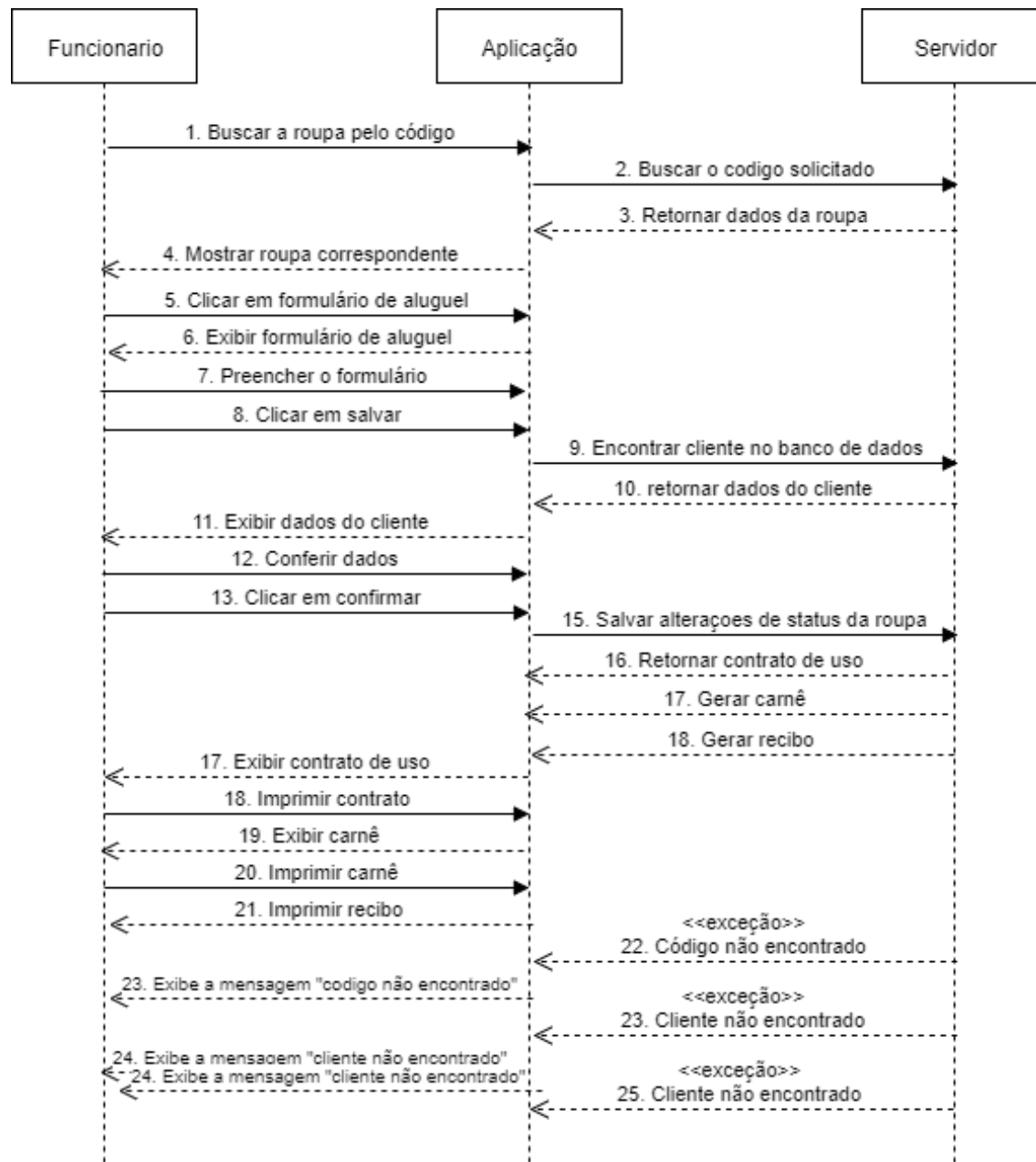


Diagrama de sequência alterar cadastro de roupas

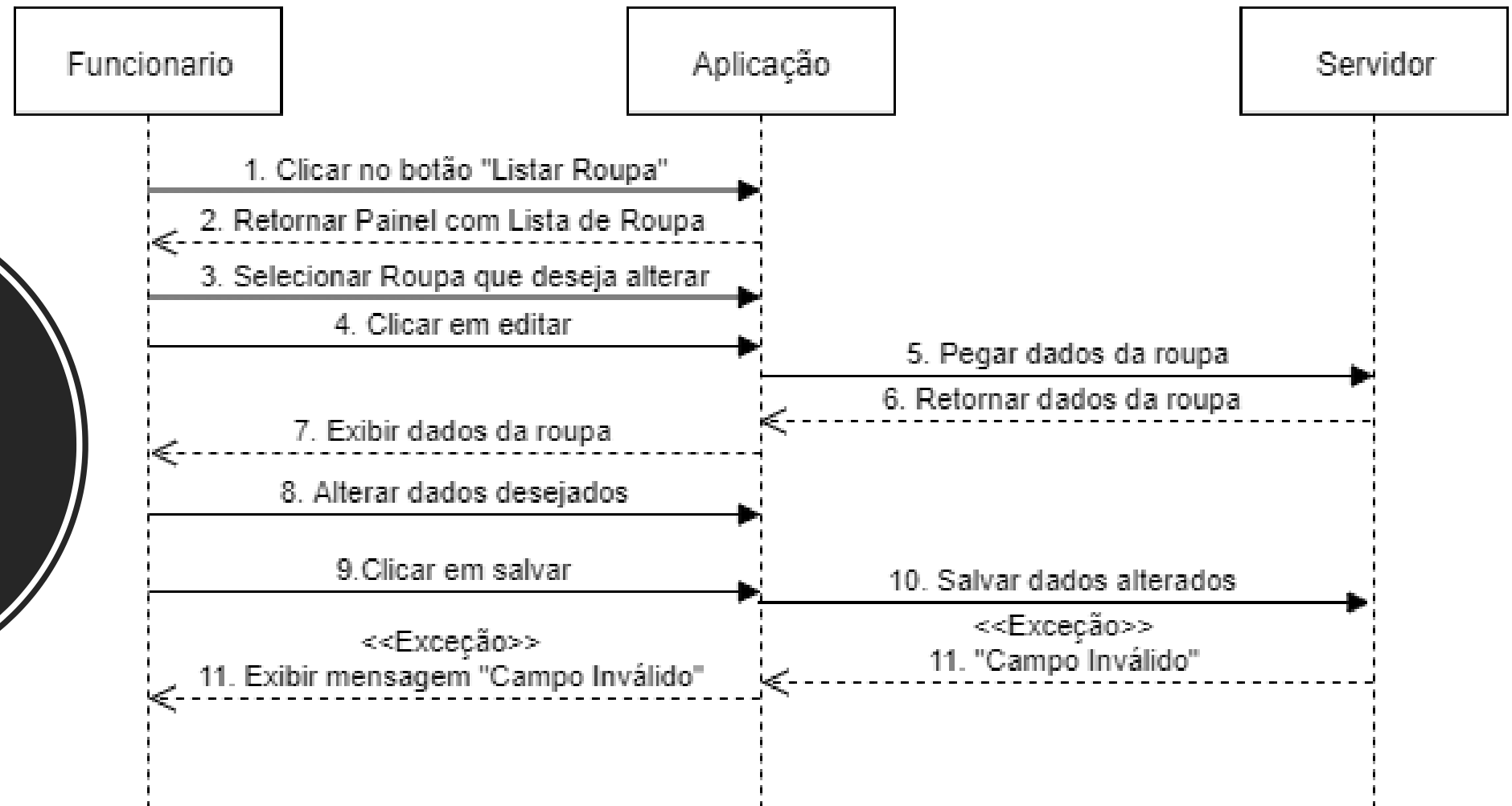


Diagrama de sequencia – alterar cadastro de clientes

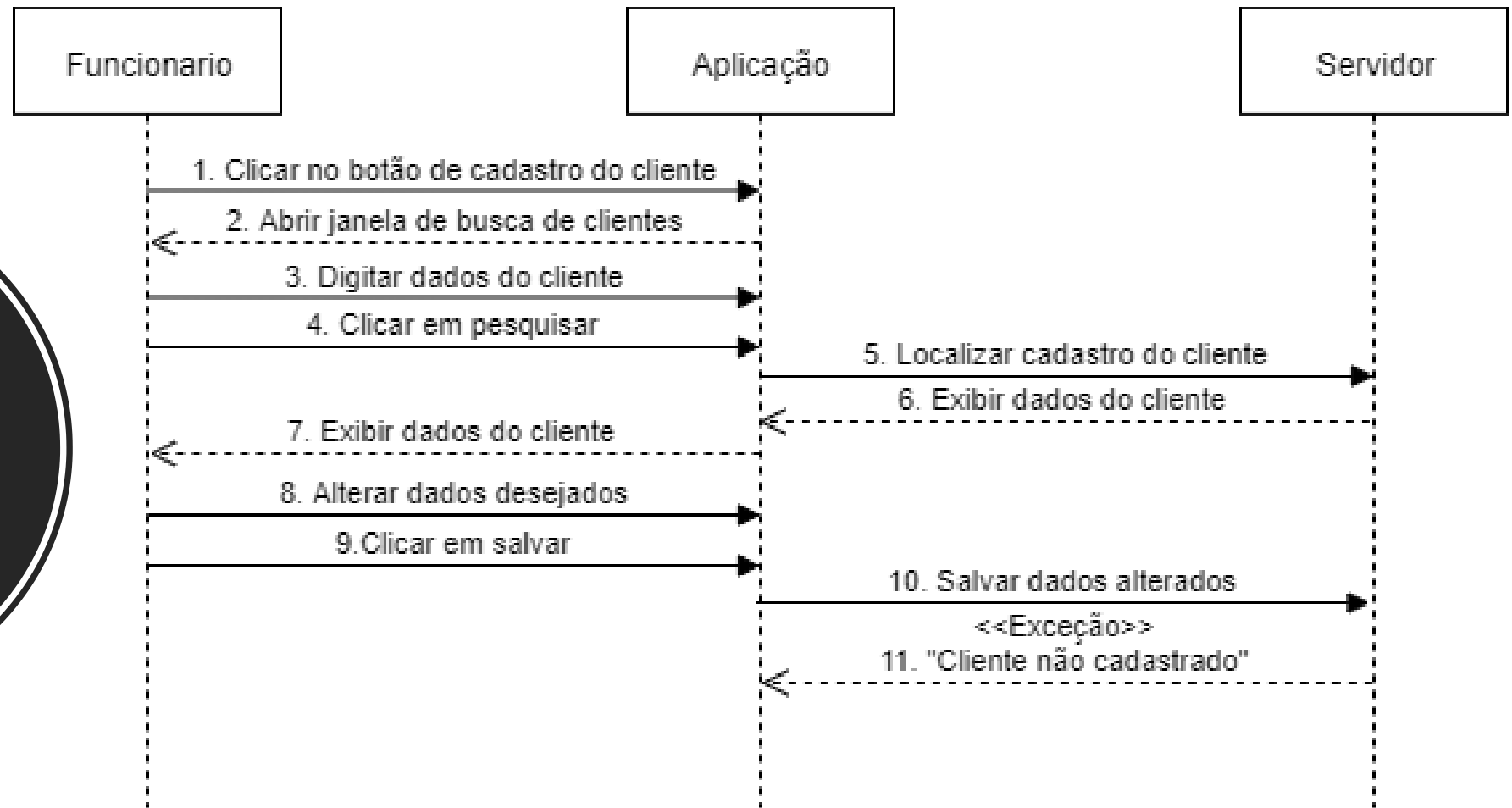
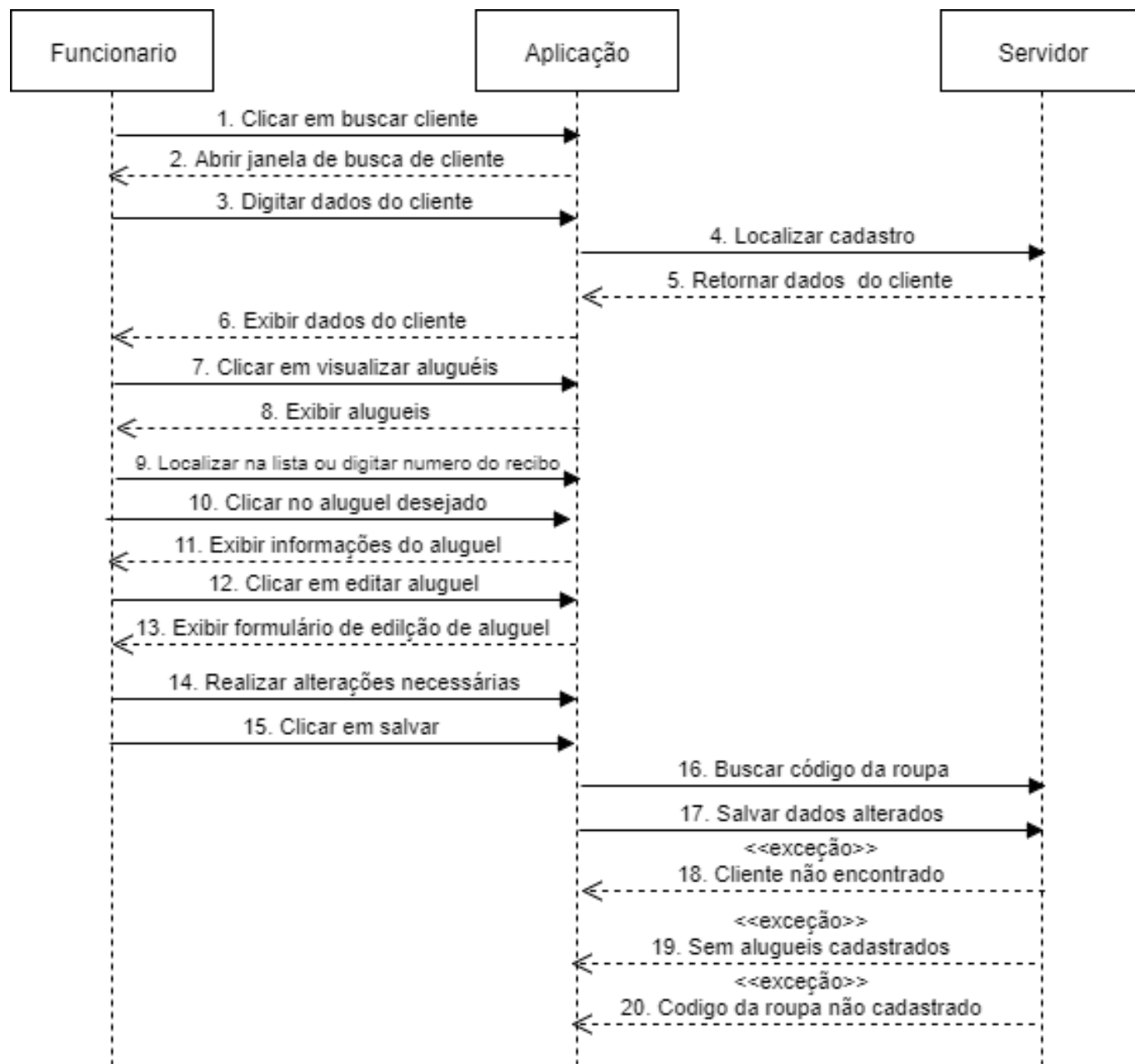
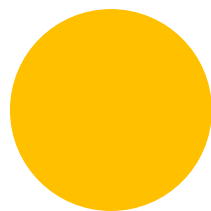
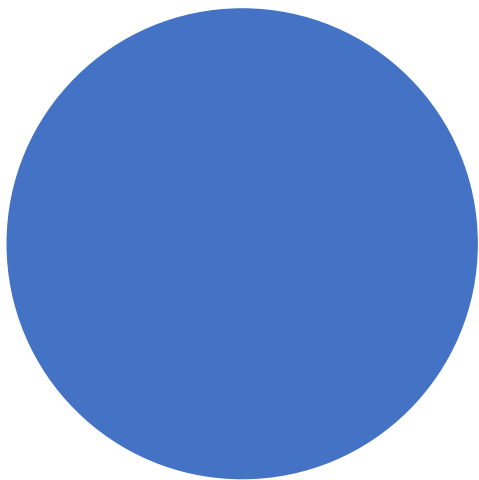


Diagrama de
sequência
Alterar
cadastro de
aluguéis





Requisitos não
funcionais

“Clotech”

Visibilidade do Status do Sistema(feedback)

RENTAL DRESS

INÍCIO

AGENDA

LOGÍSTICA

LOCAÇÃO

CLIENTES

ESTOQUE

CONFECÇÕES

FINANCEIRO

USUÁRIOS

CONFIGURAÇÕES

LOCAÇÃO

Inserir Locação

LISTA DE LOCAÇÃO ⓘ

BUSQUE POR NOME OU CPF

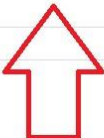
Filtrar

NUMERO DE LOCAÇÃO: 101

CODIGO	NOME CLIENTE	TELEFONE	EVENTO	RETIRADA	DEVOLUÇÃO	STATUS	AÇÕES	APAGAR
178	22/05/2016			● EM ABERTO	Opções ▾	🗑
177	09/07/2016	08/07/2016	11/07/2016	● EM ABERTO	Opções ▾	🗑
175	15/04/2016	17/06/2016	20/06/2016	● EM ABERTO	Opções ▾	🗑
173	15/04/2016	14/04/2016	17/04/2016	● EM ABERTO	Opções ▾	🗑
171	15/10/2016	14/10/2016	17/10/2016	● EM ABERTO	Opções ▾	🗑
169	02/01/2017	01/01/2017	04/01/2017	● EM ABERTO	Opções ▾	🗑
168	14/05/2016	13/05/2016	13/05/2016	● EM ABERTO	Opções ▾	🗑
167	30/04/2016	29/04/2016	02/05/2016	● EM ABERTO	Opções ▾	🗑
166	18/03/2016	18/03/2016	20/03/2016	● RETIRADO	Opções ▾	🗑
162	30/04/2016	29/04/2016	02/05/2016	● EM ABERTO	Opções ▾	🗑

« ANTERIOR 1 2 3 4 5 6 7 8 9 10 11 PRÓXIMA »

Apresenta os status dos alugueis presentes no sistema.



Compatibilidade do sistema com o mundo real

The screenshot displays the 'RENTAL DRESS' system interface. On the left is a dark sidebar with navigation links: INÍCIO, AGENDA, LOGÍSTICA, LOCAÇÃO, CLIENTES, ESTOQUE, CONFECÇÕES, FINANCEIRO, USUÁRIOS, and CONFIGURAÇÕES. The main header area includes the title 'LOCAÇÃO' and a subtitle 'ESCOLHA A DATA PARA RETIRADA E PARA DEVOLUÇÃO.' Below this is a horizontal flow of five icons: a person, a hanger, a calendar (highlighted in green), a dollar sign, and a document. The central section is titled 'LISTA DE ITENS CONTRATADOS' and contains a table with the following columns: FOTO, TÍTULO, CODIGO, HISTÓRICO, and DISPONIVEL PARA DATA. A single row is visible with a blurred photo, a blurred title, the code '01', a 'HISTÓRICO' button, and the status 'DISPONIVEL' in green text, which is circled in red. Below the table, there are input fields for 'DATA DA RETIRADA' (17/12/2015) and 'DATA DA DEVOLUÇÃO' (20/12/2015), followed by 'Voltar' and 'Continuar' buttons.

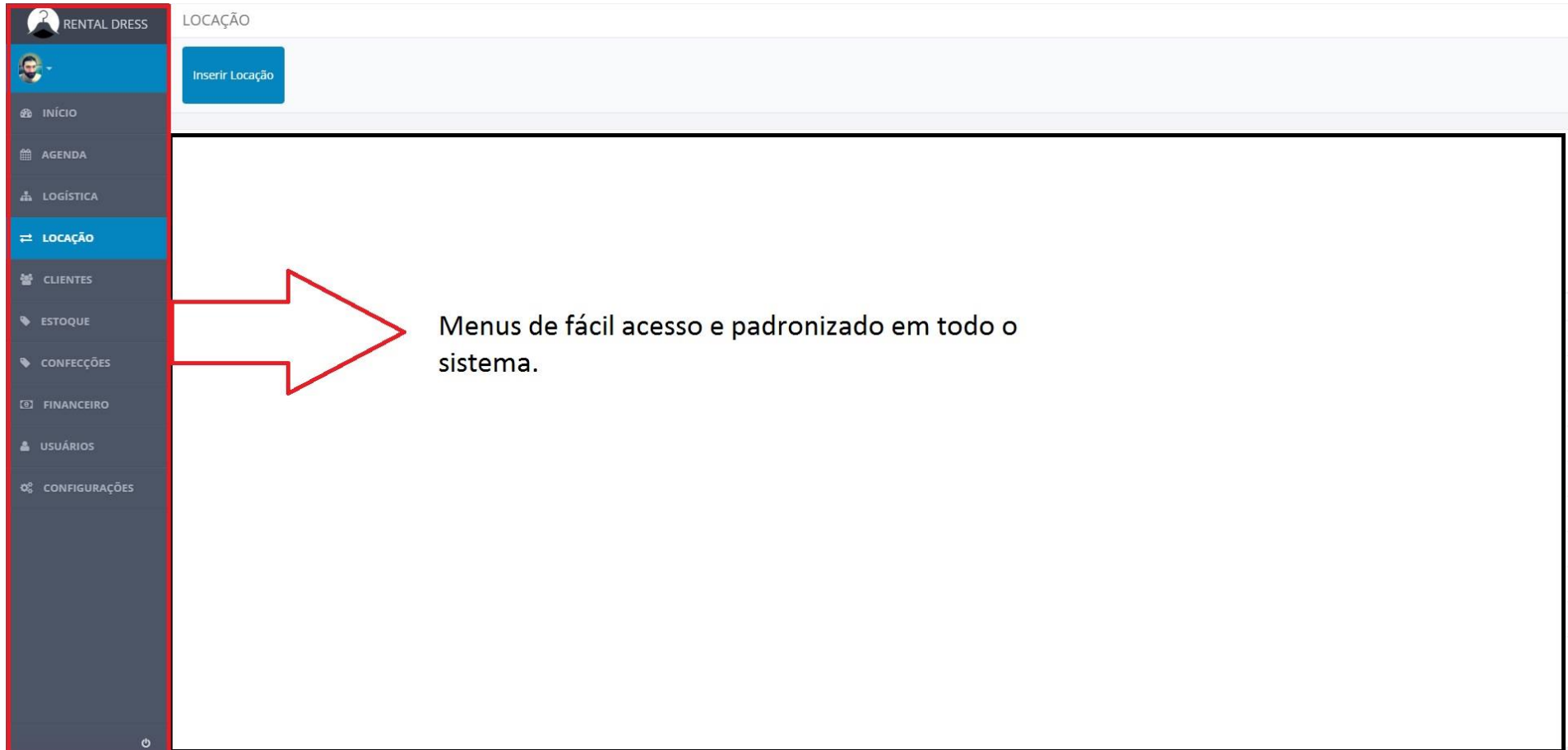
FOTO	TÍTULO	CODIGO	HISTÓRICO	DISPONIVEL PARA DATA
		01	<button>HISTÓRICO</button>	DISPONIVEL

Apresenta a disponibilidade da roupa no dia

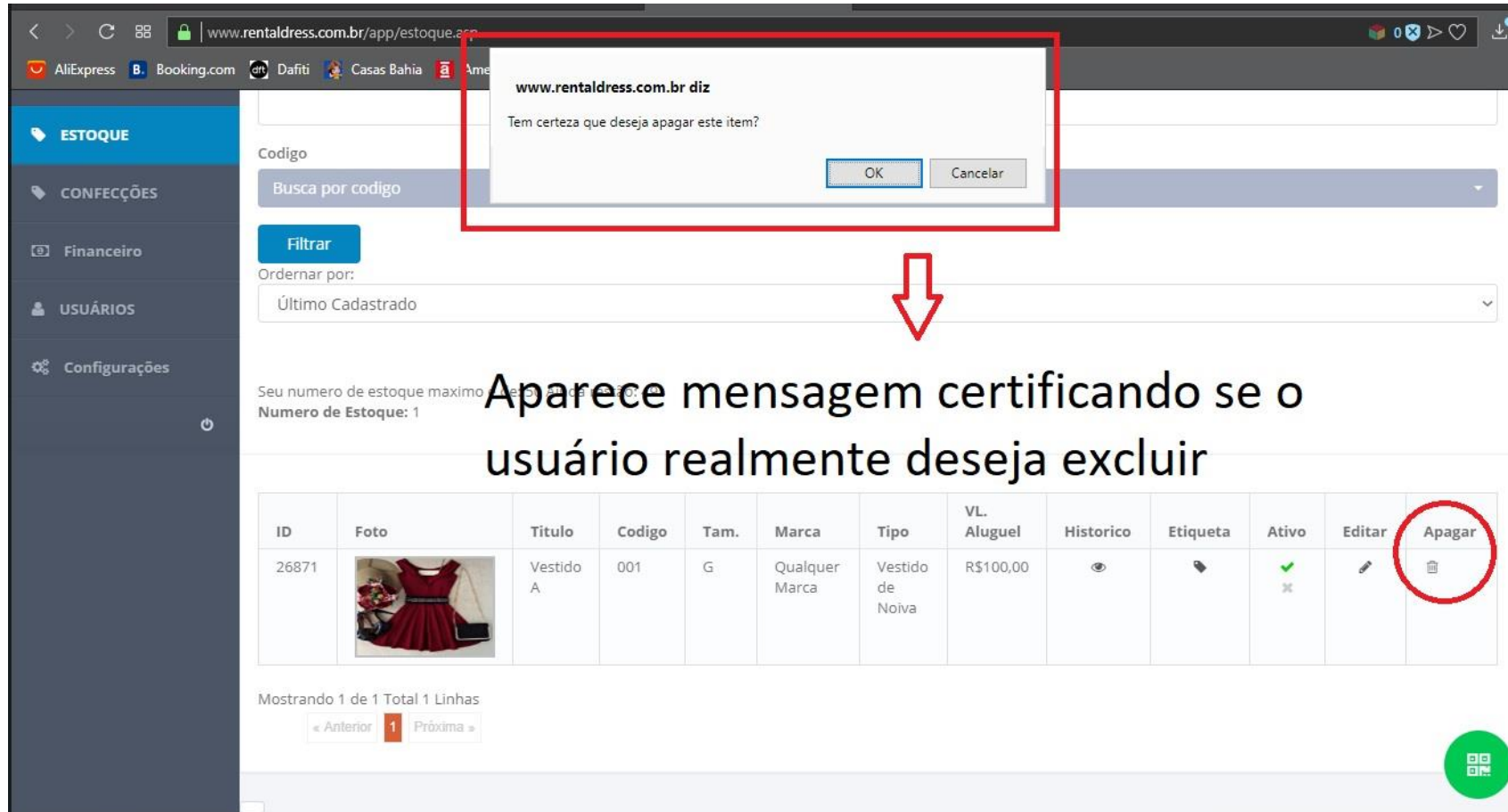
Controle do Usuário e Liberdade



Consistência e Padrões









Prevenção de Erros



The screenshot shows a web application interface for managing inventory. A confirmation dialog is displayed, asking the user to confirm the deletion of an item. The dialog is titled "www.rentaldress.com.br diz" and contains the text "Tem certeza que deseja apagar este item?". The dialog has "OK" and "Cancelar" buttons. A red box highlights the dialog, and a red arrow points from it to the "Apagar" button in the table below.

Aparece mensagem certificando se o usuário realmente deseja excluir

ID	Foto	Título	Codigo	Tam.	Marca	Tipo	VL. Aluguel	Historico	Etiqueta	Ativo	Editar	Apagar
26871		Vestido A	001	G	Qualquer Marca	Vestido de Noiva	R\$100,00					

Mostrando 1 de 1 Total 1 Linhas

« Anterior 1 Próxima »

Reconhecer em vez de relembrar

RENTAL DRESS

LOCAÇÃO

PRIMEIRAMENTE ESCOLHA O CLIENTE PARA QUE POSSA CONTINUAR COM A LOCAÇÃO.

Escolher Cliente

Reconhece o aluguel de acordo com as ultimas inseridas.

LISTA DE CLIENTES ⓘ

BUSQUE POR NOME OU CPF

Filtrar

NUMERO DE CLIENTES: 100

NOME	CPF	DATA DO EVENTO	ESCOLHER
ADRIANA ALMEIDA FERREIRA	00000000000	16/04/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	12/03/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	03/03/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	19/03/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	19/03/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	28/07/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	04/03/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	28/05/2016	Escolher este cliente
ADRIANA ALMEIDA FERREIRA	00000000000	01/04/2016	Escolher este cliente

Flexibilidade e Eficiência de Uso

The screenshot displays the 'MARCAÇÃO DE PROVAS' (Test Marking) section of the RENTAL DRESS application. A sidebar on the left contains navigation links: INÍCIO, AGENDA, LOGÍSTICA, LOCAÇÃO, CLIENTES, ESTOQUE, CONFECÇÕES, FINANCEIRO, USUÁRIOS, and CONFIGURAÇÕES. The main content area is titled 'DETALHES DA LOCAÇÃO' and includes fields for client information (NOME DO CLIENTE, CPF, RG, ESTADO, CIDADE, TELEFONE, EMAIL) and event details (DATA DO EVENTO: 05/03/2016). Below this, it shows 'DATA MARCADA PARA ENTREGA' (04/03/2016) and 'DATA MARCADA DE DEVOLUÇÃO' (07/03/2016). A table titled 'LISTA DE ITENS PARA MARCAR A PROVA' has columns for TITULO, CODIGO, and DATA. A date picker is open, showing a calendar for March 2016 with the 17th selected. The date picker is highlighted with a red box and a red arrow pointing to it from the text 'Usuário pode escolher a data de forma dinâmica'. To the right of the date picker, there are dropdowns for 'HORÁRIO' (12 Hora) and '30 minu', and a button 'Agendar prova'. Below the date picker, there is a table titled 'PROVAS MARCADAS PARA ESTA LOCAÇÃO' with columns for TITULO, CODIGO, and STATUS. The table is currently empty.

RENTAL DRESS

MARCAÇÃO DE PROVAS

Voltar

DETALHES DA LOCAÇÃO

NOME DO CLIENTE: [REDACTED]

DATA DO EVENTO: 05/03/2016 CPF: [REDACTED] RG: [REDACTED] ESTADO: MG

CIDADE: UBERLÂNDIA

TELEFONE: [REDACTED]

EMAIL: [REDACTED]

DATA MARCADA PARA ENTREGA: 04/03/2016 DATA MARCADA DE DEVOLUÇÃO: 07/03/2016

LISTA DE ITENS PARA MARCAR A PROVA

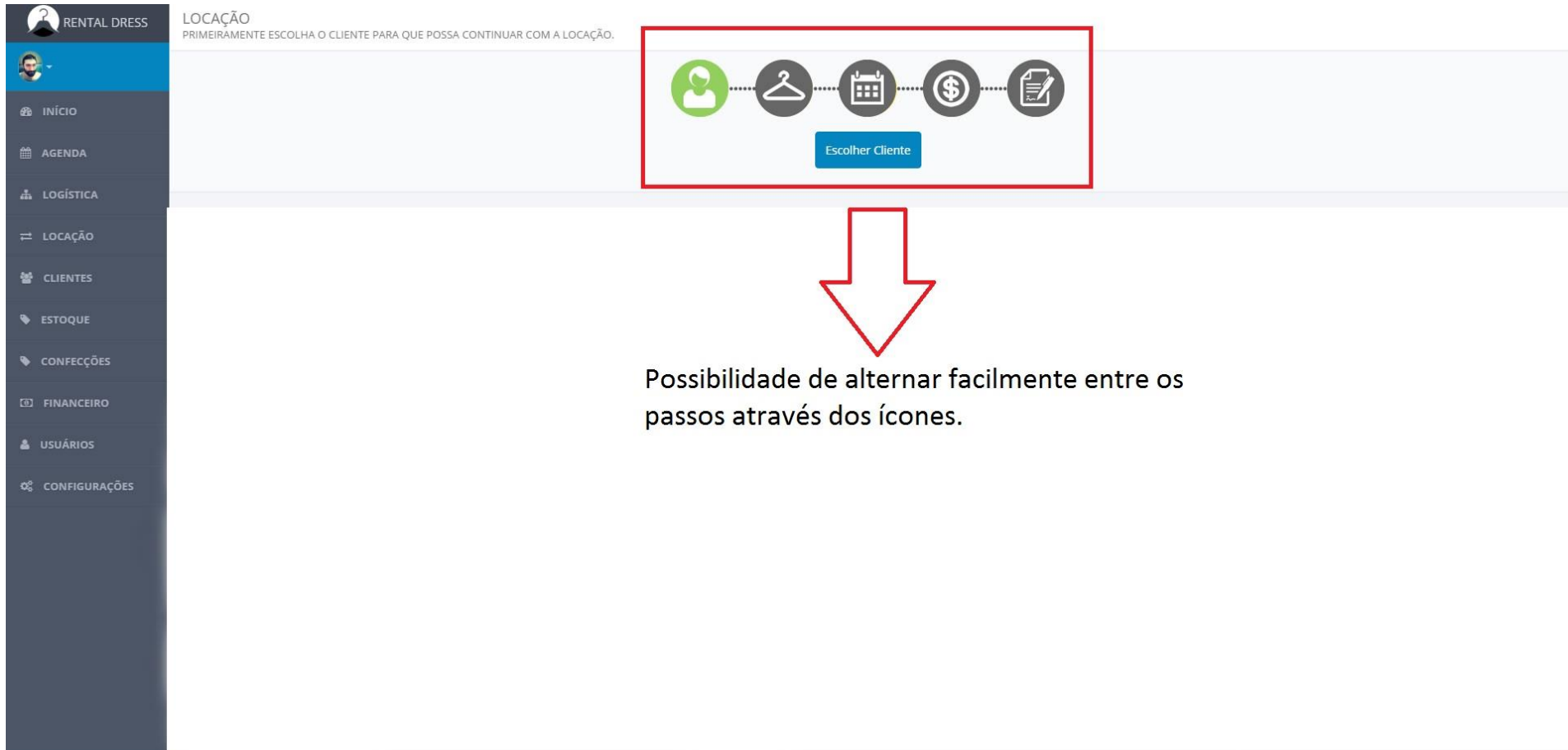
TITULO	CODIGO	DATA	HORÁRIO	AGENDAR PROVA
[REDACTED]	198	17-03-2016	12 Hora 30 minu	Agendar prova

PROVAS MARCADAS PARA ESTA LOCAÇÃO

TITULO	CODIGO	STATUS
--------	--------	--------

Usuário pode escolher a data de forma dinâmica

Estética e Design Minimalista



Ajudar os usuário a reconhece, diagnosticar e corrigir erros

The image shows a web form with a dark blue sidebar on the left. The main content area contains several input fields. A date picker is open, showing a calendar for the month of June. The date 15 is selected. A validation error message is displayed above the date picker, indicating that the field is required. Below the date picker, there are five more input fields, each with a label above it: 'Estilista', 'Makeup', 'Facebook', 'Instagram', and 'Como conheceu a empresa?'. The 'Estilista' field has a dropdown menu with 'NADA' selected. The 'Makeup' field has a small icon in the bottom right corner. The 'Facebook' and 'Instagram' fields have small icons in the bottom right corner. The 'Como conheceu a empresa?' field has a small icon in the bottom right corner.

Preencha este campo.

Do	Se	Te	Qu	Qt	Sx	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Estilista

Makeup

Facebook

Instagram

Como conheceu a empresa?

Ajuda e Documentação

- Cadastros
 - » Clientes
 - » Estoque
- Estoque
 - » Histórico
 - » Reforma
 - » Ajustes
- Vendas
 - » Vendas
- Financeiro
 - » Contas a Pagar
 - » Fluxo de Caixa
- Locação
 - » Cadastro de locação
 - » Disponibilidade
 - » Retirada
 - » Devolução
 - » Provas
- Confecção
 - » Acompanhamento de Confecção

Como funciona o Rental Dress?

Software de Gestão de aluguel de trajes que atende pequenas e médias e grandes empresas.

Com o nosso sistema online você pode controlar, em um só lugar, o financeiro, o setor de locação, o setor de vendas, estoque e todos os demais departamentos da sua empresa em um único software de forma rápida, simples e intuitiva.

Veja algumas vantagens e **experimente grátis!**

Atendimento Personalizado

Pensado em cada um dos clientes, preocupando-se, acima de tudo, com suas necessidades específicas. Seja por telefone, chat, e-mail, Skype ou WhatsApp. Vamos dar um jeito de responder as suas perguntas e achar uma solução para as suas dúvidas.

Segurança

Não se preocupe, seus dados estão totalmente seguros! Nossos servidores são monitorados 24h por dia, garantindo disponibilidade o tempo todo.

Acesse de qualquer lugar

Nosso sistema é 100% online e totalmente otimizado para acesso de celular, tablet, notebook e computador.

Benefícios de um sistema em nuvem e sua facilidade de acesso

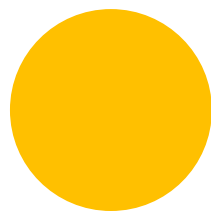
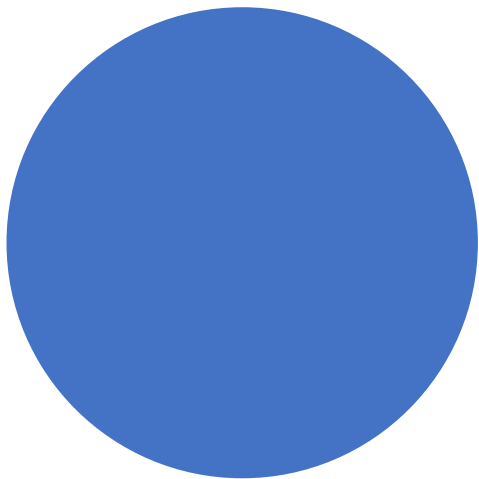
1- Redução dos custos: Os serviços pagos para softwares desenvolvidos na nuvem são bem mais em conta tanto na contratação quanto na manutenção do sistema.

2- Mais facilidade e menos preocupação: Por ser um sistema de acesso pela internet, não são necessárias instalações físicas, além disso, as atualizações são online, garantindo que a empresa não perca tempo de vendas com o software.

3- Flexibilidade: Acessibilidade total em qualquer lugar, garantindo mais conforto

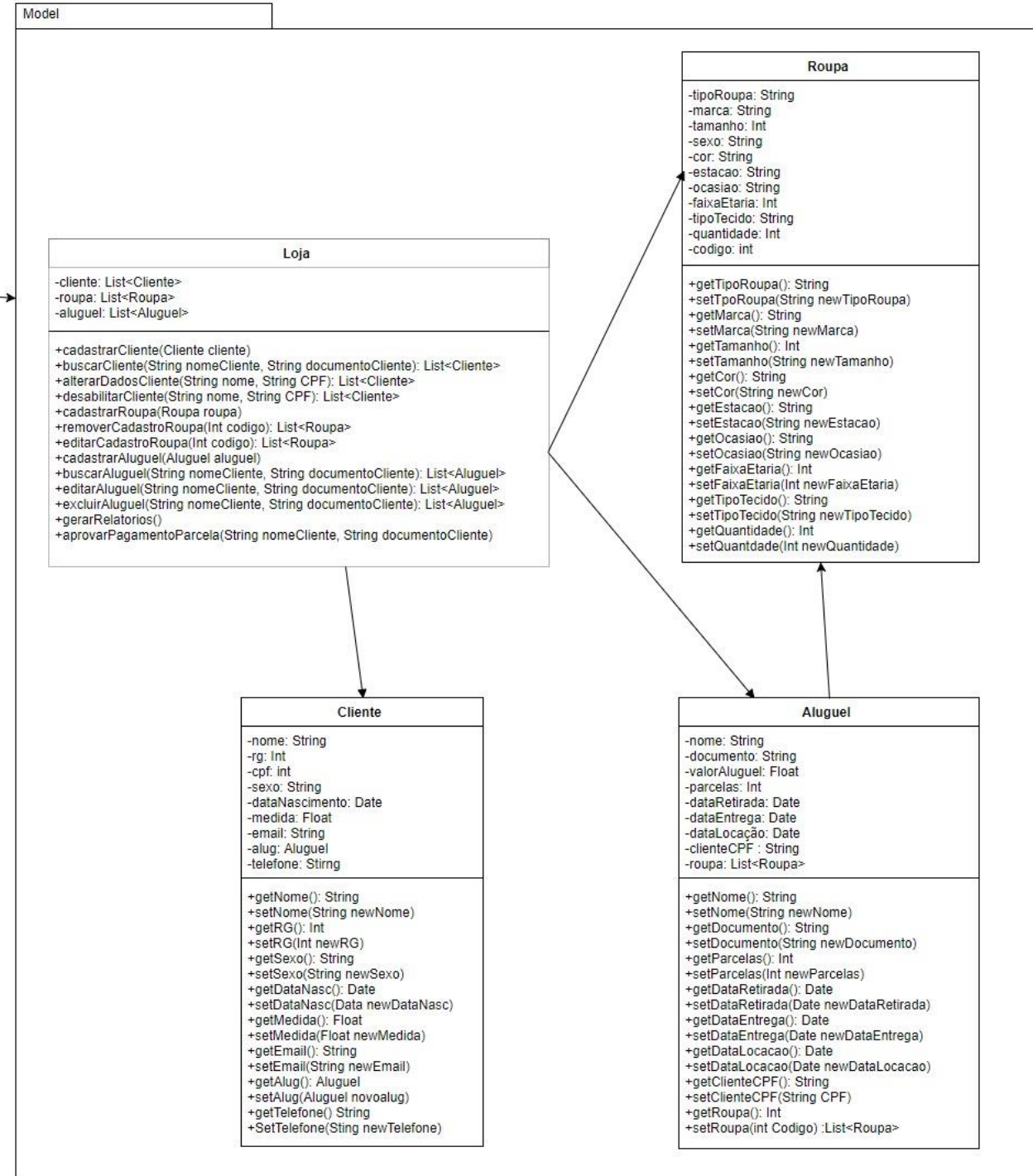
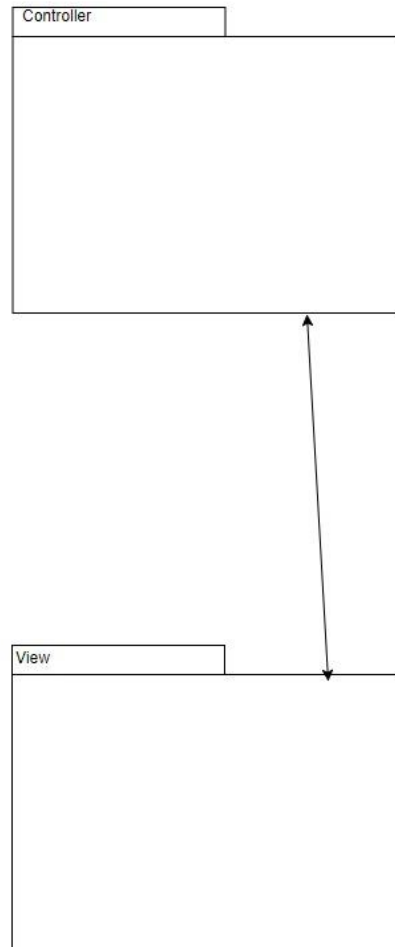
Envie-nos uma mensagem

jivochat



Exemplo de Arquitetura do
Sistema

“Clotech”



Desenvolvimento


```
package Model;
```

```
import java.time.LocalDate;
import java.util.LinkedList;
import java.util.List;
```

```
public class Loja {
    private List <Roupa> roupas = new LinkedList <Roupa> ();
    private List <Aluguel> alugueis = new LinkedList <Aluguel> ();
    private List <Cliente> clientes = new LinkedList <Cliente> ();
    private List <Roupa> selectedRoupa = new LinkedList <Roupa> ();

    public Cliente buscaCliente(String nome, String cpf) {
        for(Cliente cliente:clientes) {
            if(cliente.getNome().equals(nome) || cliente.getCPF().equals(cpf))return cliente;
        }
        return null;
    }
    //Adicionar na documentação
    public List<Roupa> selecionaRoupa(int codigo) {
        for(Roupa roupa:roupas) {
            if(roupa.getCodigo() == codigo) {
                selectedRoupa.add(roupa);
            }
        }return selectedRoupa;
    }

    public Aluguel buscaAluguel(String nomeCliente, String cpfCliente) {
        for(Aluguel aluguel:alugueis) {
            if(aluguel.getNome().equals(nomeCliente) || aluguel.getDocumento().equals(cpfCliente))return aluguel;
        }
        return null;
    }

    public Roupa buscaRoupa(int codigoRoupa) {
        for(Roupa roupa:roupas) {
            if(roupa.getCodigo() == codigoRoupa) return roupa;
        }
        return null;
    }

    public void alterarCliente(String cpfClientedeBusca, String novoNome, String sexo,String medida ,String email, String telefone) {
        for(Cliente cliente:clientes) {
            if(cliente.getCPF().equals(cpfClientedeBusca)) {
                if(!novoNome.isEmpty()) {
                    cliente.setNome(novoNome);
                }
                if(!sexo.isEmpty()) {
                    cliente.setSexo(sexo);
                }
                if(medida.isEmpty()) {
                    cliente.setMedida(medida);
                }
                if(!email.isEmpty()) {
                    cliente.setEmail(email);
                }
                if(!telefone.isEmpty()) {
                    cliente.setTelefone(telefone);
                }
            }
        }
    }
}
```

Loja
-cliente: List<Cliente> -roupa: List<Roupa> -aluguel: List<Aluguel>
+cadastrarCliente(Cliente cliente) +buscarCliente(String nomeCliente, String documentoCliente): List<Cliente> +alterarDadosCliente(String nome, String CPF): List<Cliente> +desabilitarCliente(String nome, String CPF): List<Cliente> +cadastrarRoupa(Roupa roupa) +removerCadastroRoupa(Int codigo): List<Roupa> +editarCadastroRoupa(Int codigo): List<Roupa> +cadastrarAluguel(Aluguel aluguel) +buscarAluguel(String nomeCliente, String documentoCliente): List<Aluguel> +editarAluguel(String nomeCliente, String documentoCliente): List<Aluguel> +excluirAluguel(String nomeCliente, String documentoCliente): List<Aluguel> +gerarRelatorios() +aprovarPagamentoParcela(String nomeCliente, String documentoCliente)

```
public void alterarRoupa(int codigo, String novoTipoRoupa, String novoMarca, String novoTamanho, String novoCor, String novoEstacao, String novoOcasiao, int novoFaixaEtaria, String novoTipoTecido, int novoQuantidade ) {
```

```
    for(Roupa roupa:roupas) {
        if(roupa.getCodigo() == codigo) {

            if(!novoTipoRoupa.isEmpty()) {
                roupa.setTipoRoupa(novoTipoRoupa);
            }

            if(!novoMarca.isEmpty()) {
                roupa.setMarca(novoMarca);
            }

            if(!novoTamanho.isEmpty()) {
                roupa.setTamanho(novoTamanho);
            }

            if(!novoCor.isEmpty()) {
                roupa.setCor(novoCor);
            }

            if(!novoEstacao.isEmpty()) {
                roupa.setEstacao(novoEstacao);
            }

            if(!novoOcasiao.isEmpty()) {
                roupa.setOcasiao(novoOcasiao);
            }

            if(novoFaixaEtaria != 0) {
                roupa.setFaixaEtaria(novoFaixaEtaria);
            }

            if(!novoTipoTecido.isEmpty()) {
                roupa.setTipoTecido(novoTipoTecido);
            }

            if(novoQuantidade != 0) {
                roupa.setQuantidade(novoQuantidade);
            }
        }
    }
}
```

```
public void alterarAluguel(int codigoAluguel, float valor_aluguel, LocalDate data_retirada, LocalDate data_entrega, List<Roupa> roupa) {
```

```
    for(Aluguel aluguel:alugueis) {
        if(aluguel.getCodigoAluguel() == codigoAluguel) {
            if(valor_aluguel != 0) {
                aluguel.setValor_aluguel(valor_aluguel);
            }
            if(!aluguel.getData_retirada().equals(data_retirada)) {
                aluguel.setData_retirada(data_retirada);
            }
            if(!aluguel.getData_entrega().equals(data_entrega)) {
                aluguel.setData_entrega(data_entrega);
            }
            if(!roupa.isEmpty()) {
                aluguel.setRoupa(roupa);
            }
        }
    }
}
```

```
public void cadastrarCliente(Cliente cliente) {
    clientes.add(cliente);
}
```

```
public void cadastrarRoupa(Roupa roupa) {
    roupas.add(roupa);
}
```

```

public void cadastrarAluguel(Aluguel aluguel) {
    alugueis.add(aluguel);
}

public void removeRoupa(int codigoRoupa) {
    for(Roupa roupa:roupas) {
        if(roupa.getCodigo() == codigoRoupa) {
            roupas.remove(roupa);
        }
    }
}

public void desabilitarCliente(String cpf) {
    for(Cliente cliente:clientes) {
        if(cliente.getCPF() == cpf) {
            cliente.setDesabilitar(true);
        }
    }
}

public void desabilitarAluguel(int codigoAlu) {
    for(Aluguel aluguel:alugueis) {
        if(aluguel.getCodigoAluguel() == codigoAlu) {
            aluguel.setDesabilitar(true);
        }
    }
}

public void aprovarPagamentoParcela(int codigoAluguel) {
    for(Aluguel aluguel:alugueis) {
        if(aluguel.getCodigoAluguel() == codigoAluguel) {
            aluguel.setN_parcelas(aluguel.getN_parcelas() - 1);
            if(aluguel.getN_parcelas() <= 0) {
                aluguel.setDesabilitar(true);
            }
        }
    }
}

public List<Cliente> getClientes(){
    return this.clientes;
}

public List<Roupa> getRoupa() {
    return roupas;
}

```

}

```

public class Roupa {
    private String tipoRoupa, marca, sexo, cor, estacao, ocasiao, tipoTecido, tamanho;
    int codigo;
    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getTamanho() {
        return tamanho;
    }

    public void setTamanho(String tamanho) {
        this.tamanho = tamanho;
    }

    private int faixaEtaria, id, quantidade;

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade (int quantidade) {
        this.quantidade = quantidade;
    }

    public String getTipoRoupa() {
        return tipoRoupa;
    }

    public void setTipoRoupa(String tipoRoupa) {
        this.tipoRoupa = tipoRoupa;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public String getSexo() {
        return sexo;
    }

    public void setSexo(String sexo) {
        this.sexo = sexo;
    }

    public String getCor() {
        return cor;
    }

    public void setCor(String cor) {
        this.cor = cor;
    }
}

```

Roupa

-tipoRoupa: String
 -marca: String
 -tamanho: Int
 -sexo: String
 -cor: String
 -estacao: String
 -ocasio: String
 -faixaEtaria: Int
 -tipoTecido: String
 -quantidade: Int
 -codigo: int

+getTipoRoupa(): String
 +setTpoRoupa(String newTipoRoupa)
 +getMarca(): String
 +setMarca(String newMarca)
 +getTamanho(): Int
 +setTamanho(String newTamanho)
 +getCor(): String
 +setCor(String newCor)
 +getEstacao(): String
 +setEstacao(String newEstacao)
 +getOcasiao(): String
 +setOcasiao(String newOcasiao)
 +getFaixaEtaria(): Int
 +setFaixaEtaria(Int newFaixaEtaria)
 +getTipoTecido(): String
 +setTipoTecido(String newTipoTecido)
 +getQuantidade(): Int
 +setQuantdade(Int newQuantidade)

```
public String getEstacao() {  
    return estacao;  
}  
  
public void setEstacao(String estacao) {  
    this.estacao = estacao;  
}  
  
public String getOcasiao() {  
    return ocasio;  
}  
  
public void setOcasiao(String ocasio) {  
    this.ocasio = ocasio;  
}  
  
public String getTipoTecido() {  
    return tipoTecido;  
}  
  
public void setTipoTecido(String tipoTecido) {  
    this.tipoTecido = tipoTecido;  
}  
  
public int getFaixaEtaria() {  
    return faixaEtaria;  
}  
  
public void setFaixaEtaria(int faixaEtaria) {  
    this.faixaEtaria = faixaEtaria;  
}  
  
public Roupas(String tipoRoupa, String marca, String sexo, String cor, String estacao, String ocasio,  
    String tipoTecido, String tamanho, int faixaEtaria, int quantidade, int codigoRoupa) {  
    super();  
    this.tipoRoupa = tipoRoupa;  
    this.marca = marca;  
    this.sexo = sexo;  
    this.cor = cor;  
    this.estacao = estacao;  
    this.ocasio = ocasio;  
    this.tipoTecido = tipoTecido;  
    this.tamanho = tamanho;  
    this.faixaEtaria = faixaEtaria;  
    this.quantidade = quantidade;  
    this.codigo = codigoRoupa;  
}  
}
```

package Model;

```
import java.time.LocalDate;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;
```

```
public class Aluguel {
    private String nome, documento, clienteCPF;
    private int n_parcelas, codigoAluguel;

    Boolean desabilitar;

    public Boolean getDesabilitar() {
        return desabilitar;
    }
    public void setDesabilitar(Boolean desabilitar) {
        this.desabilitar = desabilitar;
    }

    public int getCodigoAluguel() {
        return codigoAluguel;
    }
    public void setCodigoAluguel(int codigoAluguel) {
        this.codigoAluguel = codigoAluguel;
    }
    private float valor_aluguel;
    private LocalDate data_retirada, data_entrega, data_locucao;
    private List <Roupa> roupa = new LinkedList <Roupa> ();

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getDocumento() {
        return documento;
    }
    public void setDocumento(String documento) {
        this.documento = documento;
    }
    public String getClienteCPF() {
        return clienteCPF;
    }
    public void setClienteCPF(String clienteCPF) {
        this.clienteCPF = clienteCPF;
    }
    public int getN_parcelas() {
        return n_parcelas;
    }
    public void setN_parcelas(int n_parcelas) {
        this.n_parcelas = n_parcelas;
    }
    public float getValor_aluguel() {
        return valor_aluguel;
    }
    public void setValor_aluguel(float valor_aluguel) {
        this.valor_aluguel = valor_aluguel;
    }
    public LocalDate getData_retirada() {
        return data_retirada;
    }
    public void setData_retirada(LocalDate data_retirada) {
        this.data_retirada = data_retirada;
    }
}
```

Aluguel

-nome: String
-documento: String
-valorAluguel: Float
-parcelas: Int
-dataRetirada: Date
-dataEntrega: Date
-dataLocação: Date
-clienteCPF : String
-roupa: List<Roupa>

+getNome(): String
+setNome(String newNome)
+getDocumento(): String
+setDocumento(String newDocumento)
+getParcelas(): Int
+setParcelas(Int newParcelas)
+getDataRetirada(): Date
+setDataRetirada(Date newDataRetirada)
+getDataEntrega(): Date
+setDataEntrega(Date newDataEntrega)
+getDataLocacao(): Date
+setDataLocacao(Date newDataLocacao)
+getClienteCPF(): String
+setClienteCPF(String CPF)
+getRoupa(): Int
+setRoupa(int Codigo) :List<Roupa>

```

public LocalDate getData_entrega() {
    return data_entrega;
}

public void setData_entrega(LocalDate data_entrega) {
    this.data_entrega = data_entrega;
}

public LocalDate getData_locucao() {
    return data_locucao;
}

public void setData_locucao(LocalDate data_locucao) {
    this.data_locucao = data_locucao;
}

public List<Roupa> getRoupa() {
    return roupa;
}

public void setRoupa(List<Roupa> roupa) {
    this.roupa = roupa;
}

}

public Aluguel(String nome, String documento, String clienteCPF, int n_parcelas, int codigoAluguel,
    Boolean desabilitar, float valor_aluguel, LocalDate data_retirada, LocalDate data_entrega,
    LocalDate data_locucao, List<Roupa> roupa) {
    super();
    this.nome = nome;
    this.documento = documento;
    this.clienteCPF = clienteCPF;
    this.n_parcelas = n_parcelas;
    this.codigoAluguel = codigoAluguel;
    this.desabilitar = desabilitar;
    this.valor_aluguel = valor_aluguel;
    this.data_retirada = data_retirada;
    this.data_entrega = data_entrega;
    this.data_locucao = data_locucao;
    this.roupa = roupa;
}

}

```


package Model;

import java.time.LocalDate;
import java.util.Date;

```
public class Cliente {  
    private String nome, sexo, email, telefone, RG, CPF, medida;  
  
    private LocalDate data_nascimento;  
    Boolean desabilitar;  
  
    public Boolean getDesabilitar() {  
        return desabilitar;  
    }  
    public void setDesabilitar(Boolean desabilitar) {  
        this.desabilitar = desabilitar;  
    }  
    public Cliente(String nome, String sexo, String email, String telefone, String RG, String CPF, String medida,  
        LocalDate data_nascimento) {  
        super();  
        this.nome = nome;  
        this.sexo = sexo;  
        this.email = email;  
        this.telefone = telefone;  
        this.RG = RG;  
        this.CPF = CPF;  
        this.medida = medida;  
        Data_nascimento = data_nascimento;  
    }  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
    public String getSexo() {  
        return sexo;  
    }  
    public void setSexo(String sexo) {  
        this.sexo = sexo;  
    }  
    public String getEmail() {  
        return email;  
    }  
    public void setEmail(String email) {  
        this.email = email;  
    }  
    public String getTelefone() {  
        return telefone;  
    }  
    public void setTelefone(String telefone) {  
        this.telefone = telefone;  
    }  
    public String getRG() {  
        return RG;  
    }  
    public void setRG(String rG) {  
        RG = rG;  
    }  
    public String getCPF() {  
        return CPF;  
    }  
    public void setCPF(String cPF) {  
        CPF = cPF;  
    }  
}
```

Cliente
-nome: String -rg: Int -cpf: int -sexo: String -dataNascimento: Date -medida: Float -email: String -alug: Aluguel -telefone: Stiring
+getNome(): String +setNome(String newName) +getRG(): Int +setRG(Int newRG) +getSexo(): String +setSexo(String newSexo) +getDataNasc(): Date +setDataNasc(Data newDataNasc) +getMedida(): Float +setMedida(Float newMedida) +getEmail(): String +setEmail(String newEmail) +getAlug(): Aluguel +setAlug(Aluguel novoalug) +getTelefone() String +SetTelefone(Sting newTelefone)


```
public String getMedida() {  
    return medida;  
}  
public void setMedida(String medida) {  
    this.medida = medida;  
}  
public LocalDate getData_nascimento() {  
    return Data_nascimento;  
}  
public void setData_nascimento(LocalDate data_nascimento) {  
    Data_nascimento = data_nascimento;  
}  
}
```

```
}
```

Atividade de Teste

- O que é testar? É (tentar) garantir que o software faz o que deveria fazer.
- É possível testar tudo? Não! Por que não?

```
funcao(int a){  
  
    b = a + 1 //deveria ser a - 1  
    c = b / 30000  
    return c  
  
}
```

Vou considerar apenas inteiros no range -32768 até +32767

Dentre estes 65536 números, apenas 4 revelam o defeito do software (-29999, 29999, -30000, 30000)

Atividade de Teste

```
package Model;

import static org.junit.jupiter.api.Assertions.*;

import java.time.LocalDate;
import java.time.Month;

import org.junit.jupiter.api.Test;

class teste {

    @Test
    void test() {
        /*Declarações*/
        Loja loja = new Loja();
        LocalDate data = LocalDate.of(2018, Month.NOVEMBER, 13);

        /*-----Teste Loja Inserts-Sets-Cadastros-----*/
        assertEquals(loja.getRoupa().size(), 0);

        loja.cadastrarCliente(new Cliente("Rone Felipe Bento", "Masculino", "ronefelipe@live.com",
            "98156592", "577040807", "13834529621", "berra tirar 2 cm", data));
        assertEquals(loja.getClientes().size(), 1); //Verifica se possui 1 cadastro

        //tipoRoupa, marca, sexo, cor, estacao, ocasiao, tipoTecido, tamanho, faixaEtaria, quantidade, codigoRoupa
        loja.cadastrarRoupa(new Roupa("Calça", "RC", "Masculino", "Preto", "Inverno",
            "Casamento", "Social", "44", 22, 5, 001));
        assertEquals(loja.getRoupa().size(), 1);

        loja.cadastrarRoupa(new Roupa("Camisa", "RC", "Masculino", "Azul", "Todas as Estações",
            "Casamento", "Social", "G", 18, 5, 002));

        assertEquals(loja.getRoupa().size(), 2);

        //nome, documento, clienteCPF, n_parcelas, codigoAluguel, valor_aluguel, data_retirada, data_entrega, data_locucao, roupa
        loja.cadastrarAluguel(new Aluguel("Rone Felipe Bento", "577040807", "13834529621", 2, 001, false, 100, LocalDate.of(2018, Month.NOVEMBER, 20),
            LocalDate.of(2018, Month.NOVEMBER, 25), LocalDate.of(2018, Month.NOVEMBER, 13),
            loja.selecionaRoupa(1)));

        /*-----Teste Loja Buscas de Dados (Gets)-----*/
        assertEquals(loja.buscaAluguel("Rone Felipe Bento", "13834529621").getCodigoAluguel(), 001);
        assertEquals(loja.buscaCliente("Rone Felipe Bento", "13834529621").getRG(), "577040807");
        assertEquals(loja.buscaRoupa(001).getTipoRoupa(), "Calça");
        /*-----Teste Roupa-----*/

        System.out.printf("Email Antigo: %s", loja.buscaCliente("Rone Felipe", "13834529621").getEmail());
        System.out.printf("\nCalça Tipo Antigo: %s", loja.buscaRoupa(001).getTipoTecido());
        /*-----Teste Loja Alterar dados (Updates)-----*/

        loja.alterarCliente("13834529621", "Rone Felipe", "", "", "ronefelipe97@gmail.com", "");
        loja.alterarRoupa(001, "", "", "", "", 19, "Jeans", 2);
        loja.alterarAluguel(001, 0, LocalDate.of(2018, Month.NOVEMBER, 28), LocalDate.of(2018, Month.DECEMBER, 2), loja.selecionaRoupa(1));
        loja.aprovarPagamentoParcela(001);

        /*-----Teste Loja Excluir dados (Remove, Desativar)-----*/
        loja.removeRoupa(001);
        //assertEquals(loja.getRoupa().size(), 2); Esse assert irá gerar um erro pois um item foi excluido, portanto exitem apenas 1 registro

        loja.desabilitarCliente("13834529621");
        assertEquals(loja.buscaCliente("Rone Felipe Bento", "13834529621").getDesabilitar(), true);

        loja.desabilitarAluguel(001);
        assertEquals(loja.buscaAluguel("Rone Felipe Bento", "13834529621").getDesabilitar(), true);
    }
}
```

