Universidade do Minho

Departamento de Informática

Desenvolvimento de Sistemas de Software

A78037 João Costeira

A78203 Paulo Mendes

A79845 Paulo Ribeiro

A77310 Nuno Reis

Relatório – 2ª Fase do Projeto

Introdução

O presente relatório é efetuado no âmbito do projeto de Desenvolvimento de Sistemas de Software, Mestrado Integrado em Engenharia Informática da Universidade do Minho.

O projeto consiste no desenvolvimento de uma aplicação informática capaz de auxiliar um utilizador a configurar um automóvel na stand onde eventualmente será ser efetuada a sua compra.

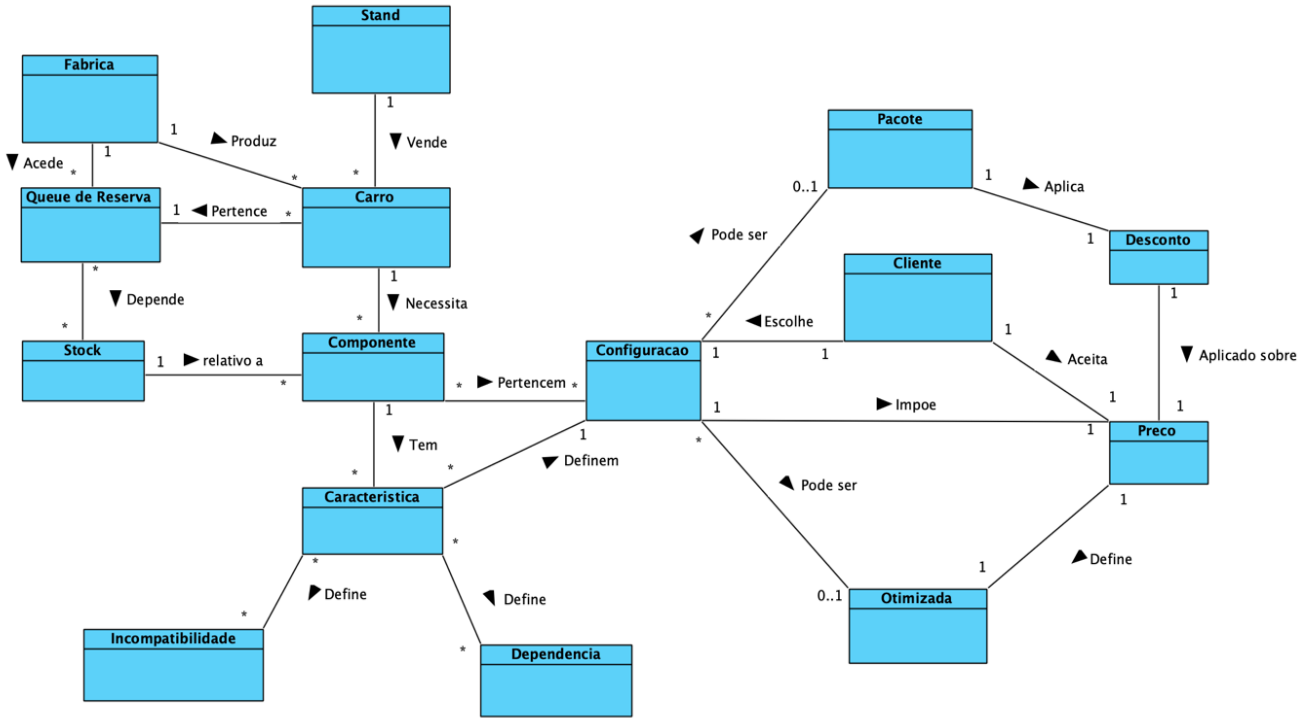
O sistema deverá suportar uma interface gráfica que permita o utilizador escolher cada uma das componentes configuráveis do veículo em questão. De modo a facilitar o processo também será possível escolher um pacote que agrupa e seleciona componentes de modo a auxiliar a configuração. Escolhendo os extras de acordo com o estilo que o cliente pretenda dar ao veículo. Por exemplo escolher um pacote desportivo.

A interface que o cliente interage também será capaz de sugerir uma configuração ótima. Após uma configuração básica como por exemplo a cor do veículo, o cliente fornece o seu orçamento e o sistema configura o veículo de modo a fornecer a melhor qualidade-preço.

Para além da interface gráfica para efetuar a configuração com o utilizador, o sistema informático também será utilizado na linha de produção. O sistema será capaz de gerir o stock dos componentes dos veículos como também será responsável de gerir a linha de produção dos veículos. A linha de produção destes veículos não funciona apenas de acordo com uma fila tradicional FIFO (First In First Out) mas existe um estatuto de prioridade, onde os veículos que têm as suas componentes em stock na fábrica são produzidos primeiro que encomendas efetuadas mais antigas, mas que não podem ser completas por causa de componentes esgotados.

Descrição dos Diagramas Utilizados

- ## Modelo de domínio



O sistema representado possui três classes fundamentais, que são o componente, cliente e configuração.
A classe componente é uma classe fundamental do projeto, pois é o elemento que o utilizador seleciona, permitindo a evolução da configuração ao longo do sistema.

Associado a cada componente temos um stock. No sistema implementado foi decidida a centralização da informação associada à disponibilidade de cada componente. Deste modo esta classe é utilizada tanto pelo o utilizador de modo a configurar o veículo, tanto pela fábrica com o intuito de produzir as encomendas efetuadas de acordo com o stock existente.
O cliente é o ator principal do sistema, este é que tem acesso as funcionalidades de configuração de cada veículo, sendo caracterizado no sistema pelo seu username, email e password.
A configuração pode ser simplificada em três linhas de execução. O cliente pode escolher cada um dos componentes do veículo, onde o sistema testa a compatibilidade com a restante configuração e a necessidade de a dependência entre as peças. Por outro lado, a configuração pode ser terminada com escolha de uma configuração ótima, que de acordo com o orçamento fornecido pelo cliente configura o veículo ou com a escolha de um pacote. Este possui um desconto no preço em comparação ao somatório do preço individual dos componentes que o constituem. Após a configuração ele pode aceitar o preço, confirmando a encomenda.

A classe configuração possui uma variável de estado que permite identificar em que estado do processo o sistema se encontra.
Temos um estado que representa uma configuração vazia, um estado que representa quando o cliente está a configurar o veículo, um estado que representa a confirmação e terminação da configuração. A partir deste ponto, uma configuração é concretamente uma encomenda, que a fabrica alterará o seu estado para completa após a produção do veículo configurado pelo utilizador.

• Arquitetura lógica e Física

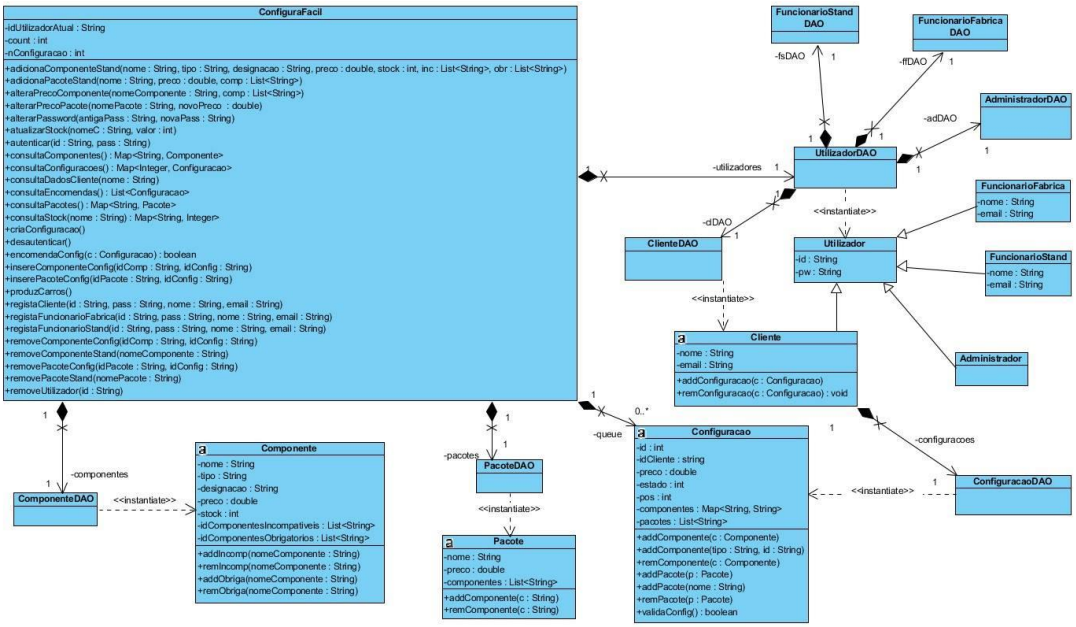
Considerando o modelo de domínio e os requisitos apresentados, podemos identificar as classes e os atributos do projeto. De modo a gerir grande parte das funcionalidades, foi definida a classe ConfiguraFacil.
Nesta classe são declaradas as estruturas que geram a grande parte dos dados deste sistema.
É declarada uma instância de UtilizadorDAO, que na realidade é um do Map username-utilizador (chave identificador do utilizador), que permite armazenar no sistema todos os seus utilizadores, desde cliente, FuncionarioFabrica, FuncionarioStand e Administrador. Todos estes são subclasses da classe utilizador.
Definida uma instância de ComponenteDAO, estrutura que associa cada identificador-componente. Deste modo é permitida o aceso a todas as peças por cada um dos diferentes tipos de utilizadores, de acordo com o papel que eles desempenham no sistema.
É instanciada um PacoteDAO (map identificador-pacote) que permite gerir/aceder a todas as configurações existentes no sistema.
Finalmente, nesta classe também é instanciada uma lista de configurações que representa a fila de produção dos veículos. Esta estrutura é acedida pelos funcionários da fábrica de modo a obter todas as configurações que se apresentam no estado de espera, à espera de serem produzidas. O grupo decidiu não definir uma fila tradicional devido ao facto de esta ser uma instância de fila com prioridade, simplesmente é permitida a produção de configurações que contenham todas as suas componentes em stock.

Outras relações de destacar entre as classes são a relação que cliente tem com configuração. O cliente possui uma instância de ConfiguracaoDAO (map identificador-configuração) de modo a guardar o histórico de configurações realizadas por cada um dos utilizadores.

Cada componente armazena em duas listas os identificadores das componentes obrigatórias e incompatíveis. Estas listas apresentam um papel essencial neste projeto, pois com estes identificadores que é garantida a preservação da validade de cada configuração efetuada no sistema.

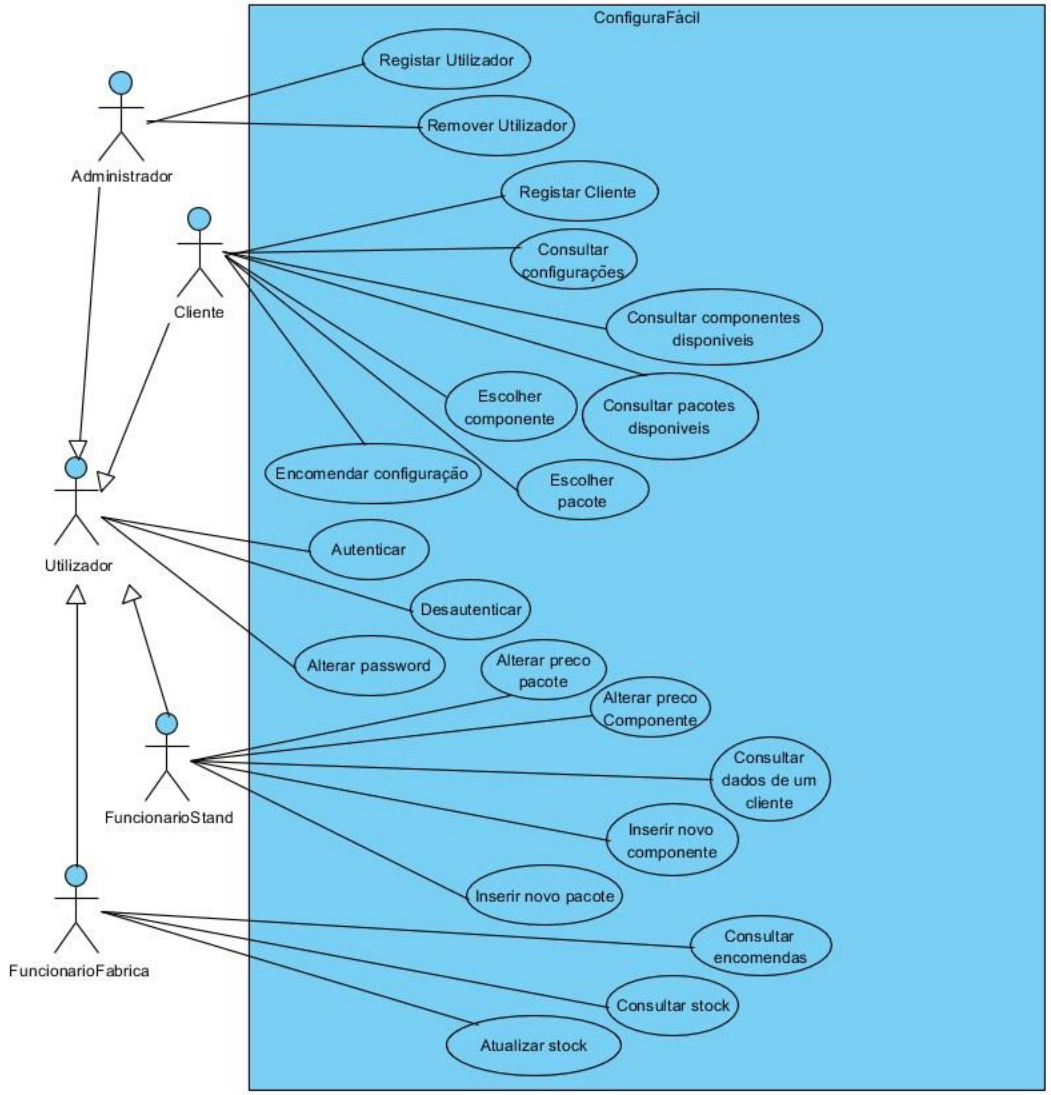
Associada a cada configuração é necessário armazenar as componentes que a constituem e a lista com os pacotes que possam existir nesta, para permitir armazenar as opções selecionadas pelo cliente e a qualquer momento a poder alter, desde que a configuração não tenha sido terminada.

A última relação existente entre as diferentes classes é a relação entre pacote e componente. Cada pacote armazena a lista de componentes que o constituem.

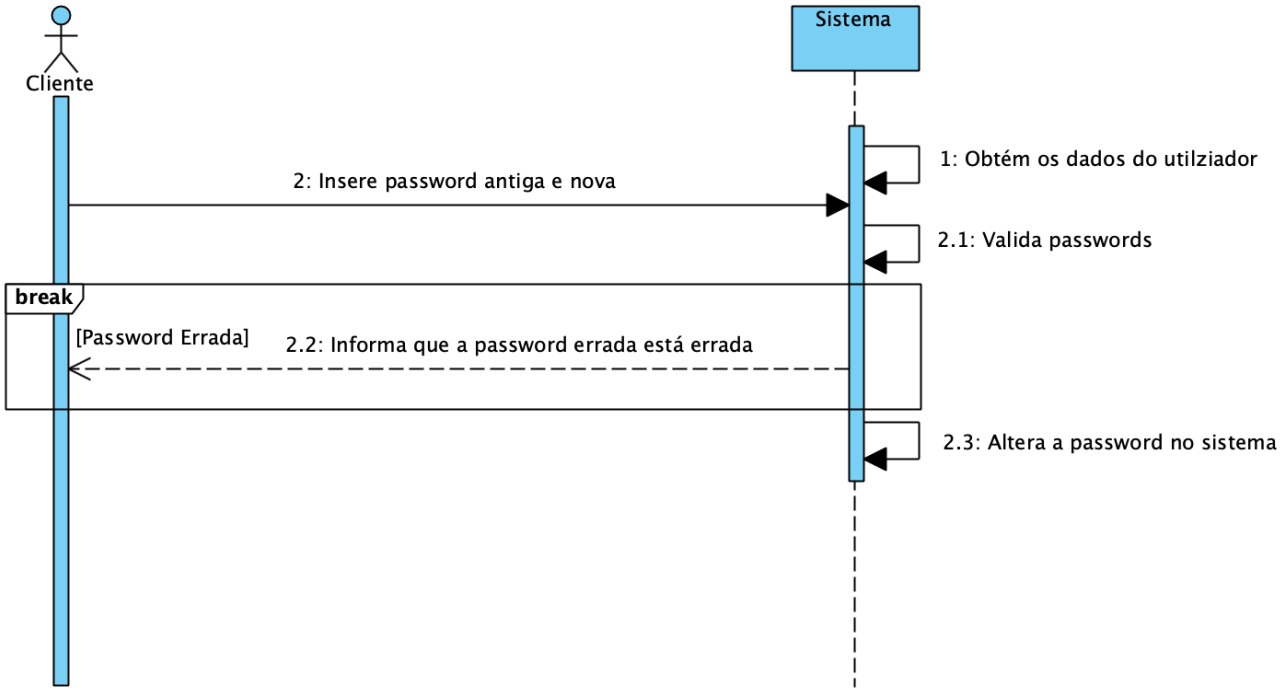

ConfiguraFacil
-idUtilizadorAtual : String
-count : int
-nConfiguracao : int
+adicionaComponenteStand(nome : String, tipo : String, designacao : String, preco : double, stock : int, inc : List<String>, obr : List<String>)
+adicionaPacoteStand(nome : String, preco : double, comp : List<String>)
+alteraPrecoComponente(nomeComponente : String, comp : List<String>)
+alterarPrecoPacote(nomePacote : String, novoPreco : double)
+alterarPassword(antigaPass : String, novaPass : String)
+atualizarStock(nomeC : String, valor : int)
+autenticar(id : String, pass : String)
+consultaComponentes() : Map<String, Componente>
+consultaConfiguracoes() : Map<Integer, Configuracao>
+consultaDadosCliente(nome : String)
+consultaEncomendas() : List<Configuracao>
+consultaPacotes() : Map<String, Pacote>
+consultaStock(nome : String) : Map<String, Integer>
+criaConfiguracao()
+desautenticar()
+encomendaConfig(c : Configuracao) : boolean
+insereComponenteConfig(idComp : String, idConfig : String)
+inserePacoteConfig(idPacote : String, idConfig : String)
+produzCarros()
+registaCliente(id : String, pass : String, nome : String, email : String)
+registaFuncionarioFabrica(id : String, pass : String, nome : String, email : String)
+registaFuncionarioStand(id : String, pass : String, nome : String, email : String)
+removeComponenteConfig(idComp : String, idConfig : String)
+removeComponenteStand(nomeComponente : String)
+removePacoteConfig(idPacote : String, idConfig : String)
+removePacoteStand(nomePacote : String)
+removeUtilizador(id : String)
FuncionarioStand DAO
FuncionarioFabrica DAO
-fsDAO
-ffDAO
AdministradorDAO
-adDAO
UtilizadorDAO
-utilizadores
FuncionarioFabrica
-nome : String
-email : String
-cDAO
<<instantiate>>
ClienteDAO
Utilizador
-id : String
-pw : String
FuncionarioStand
-nome : String
-email : String
<<instantiate>>
Cliente
-nome : String
-email : String
+addConfiguracao(c : Configuracao)
+remConfiguracao(c : Configuracao) : void
Administrador
-queue
0..*
Configuracao
-id : int
-idCliente : string
-preco : double
-estado : int
-pos : int
-componentes : Map<String, String>
-pacotes : List<String>
+addComponente(c : Componente)
+addComponente(tipo : String, id : String)
+remComponente(c : Componente)
+addPacote(p : Pacote)
+addPacote(nome : String)
+remPacote(p : Pacote)
+validaConfig() : boolean
-configuracoes
<<instantiate>>
ConfiguracaoDAO
-componentes
ComponenteDAO
<<instantiate>>
Componente
-nome : String
-tipo : String
-designacao : String
-preco : double
-stock : int
-idComponentesIncompativeis : List<String>
-idComponentesObrigatorios : List<String>
+addIncomp(nomeComponente : String)
+remIncomp(nomeComponente : String)
+addObriga(nomeComponente : String)
+remObriga(nomeComponente : String)
-pacotes
PacoteDAO
<<instantiate>>
Pacote
-nome : String
-preco : double
-componentes : List<String>
+addComponente(c : String)
+remComponente(c : String)

• Modelo de Use Case

A análise de requisitos do sistema levou à conclusão da existência de quatro atores que interagem com o sistema. Esses atores são o cliente, o funcionário da fábrica, o funcionário da Stand e o Administrador.
O cliente interage com o sistema ao efetuar a configuração do veículo. Deste modo foi definido use cases que descrevem cada forma possível de configurar o veículo e outro caso para confirmar a sua compra.
O funcionário da fábrica interage com o sistema no processo de conclusão de cada encomenda, logo este tem acesso à informação do stock, podendo atualizar este e aceder à fila das encomendas em espera de produção.
O funcionário da fábrica interage com o sistema no gerenciamento da stand. Este pode alterar informações sobre cada componente, como por exemplo o seu preço e atualizar o catálogo de configurações existentes no sistema.
O administrador é o utilizador que possui permissões para adicionar ou retirar cada um dos outros utilizadores do sistema, como um dado cliente ou funcionário.

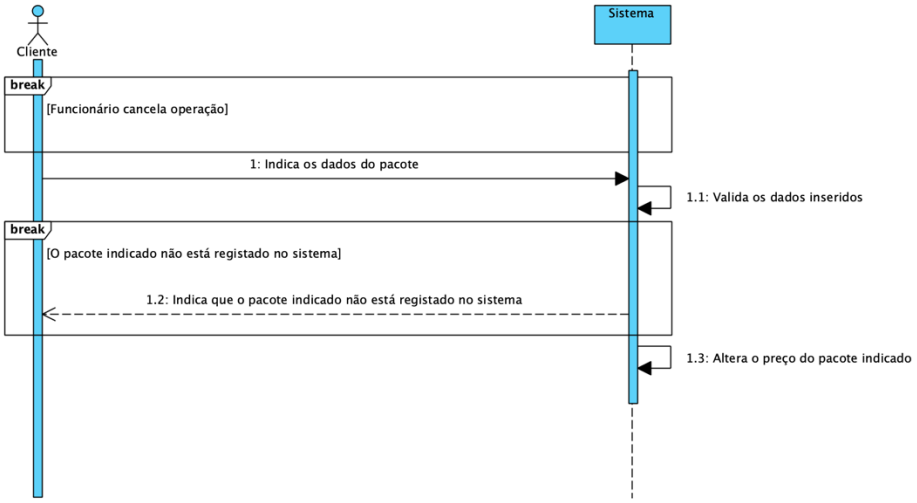


Alterar password:



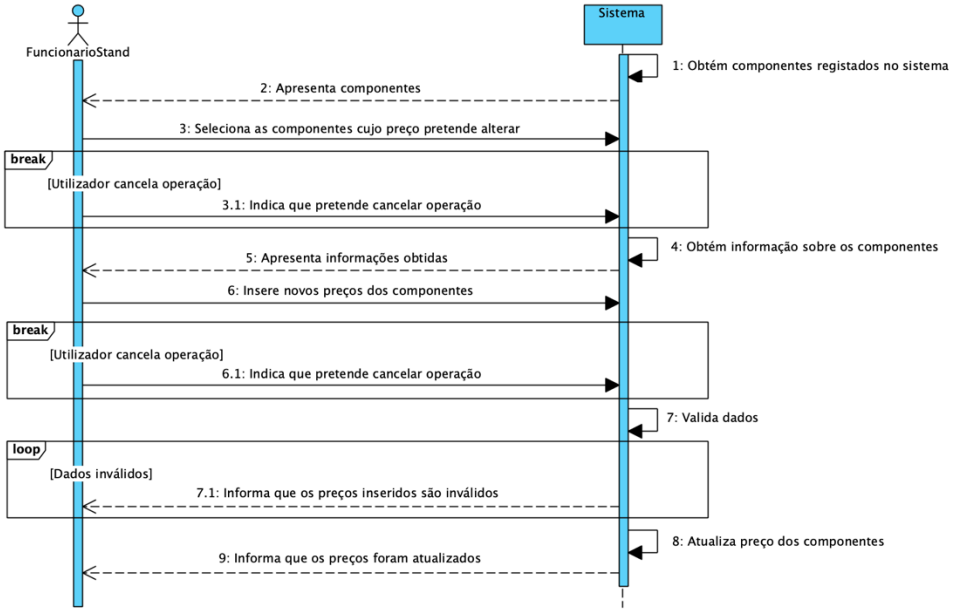
Use Case:	Alterar Password	
Descrição:	Utilizador altera a sua password no sistema	
Pré condição:	Utilizador autenticado no sistema	
Pós condição:	Altera a sua password no sistema	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém os dados do utilziador
	2. Insere password nova e a password antiga	
		3. Valida passwords
		4. Altera a password no sistema
Exceção 1 [Password Errada] (passo 3)		3.1 Informa que a password errada está errada
		3.2 Termina o processo

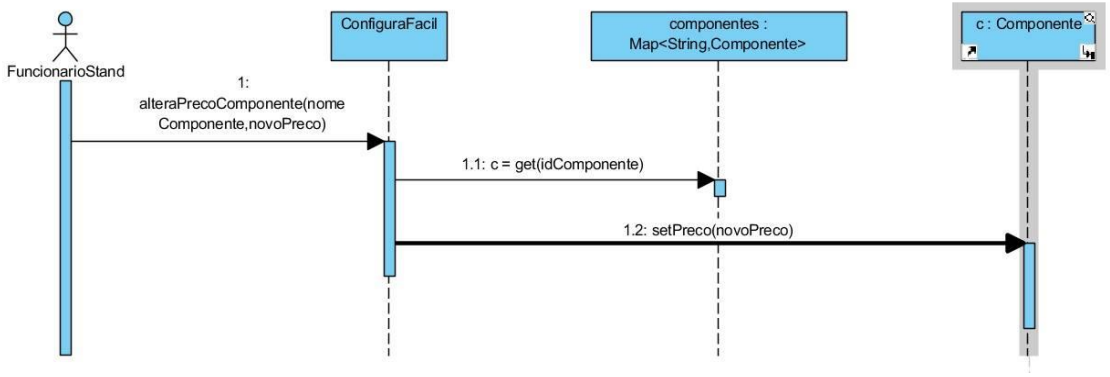
Alterar preço do pacote:



Use Case:	Alterar preço pacote	
Descrição:	FuncionárioStand altere preço associado a um pacote	
Pré condição:	FuncionárioStand autenticado no sistema	
Pós condição:	Altera o preço de um dado pacote	
Comportamento Normal	**Ator**	**Sistema**
	1.Indica o nome do pacote que pretende alterar preço e o novo preço	
		2.Valida dados inseridos
		3.Altera o preço do pacote indicado
Exceção 1 [Funcionário cancela operação] (passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Exceção 2 [O pacote indicado não está registado no sistema] (passo 2)		2.1 Indica que o pacote indicado não está registado no sistema
		2.2 Termina processo

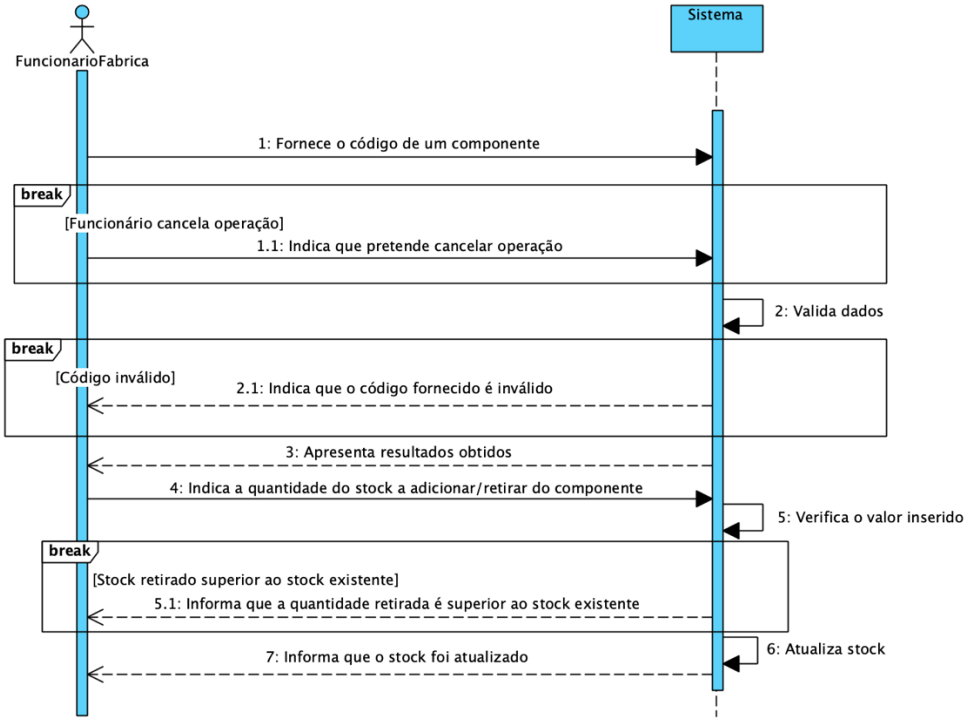
Alterar o preço componente:

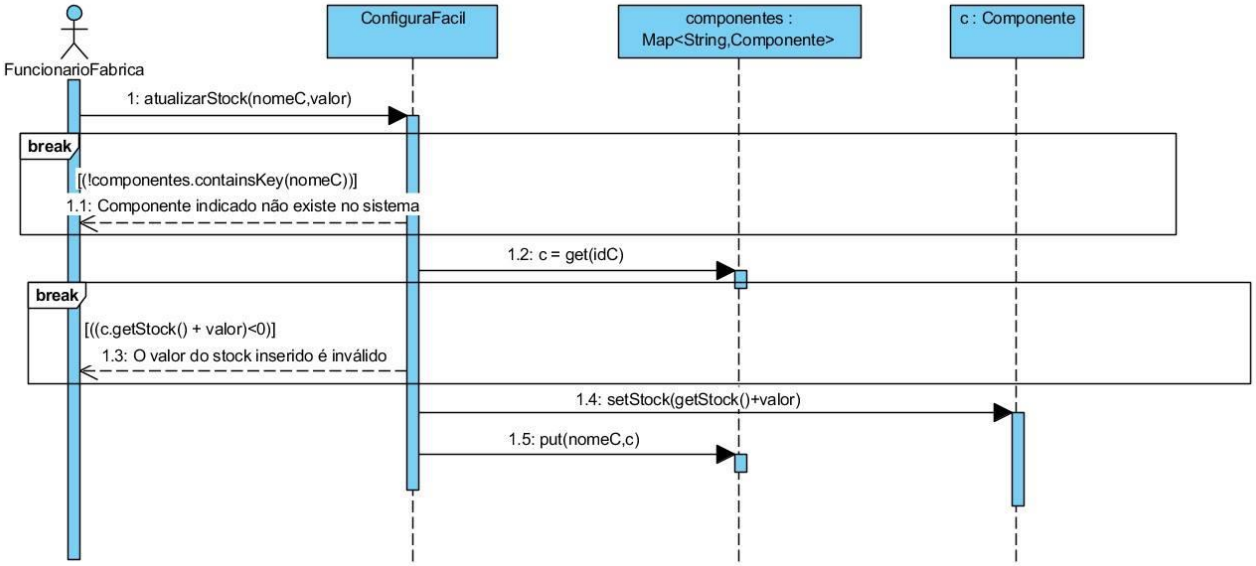




Use Case:	Alterar preço componente	
Descrição:	FuncionárioStand altere preço associado a um componente	
Pré condição:	FuncionárioStand autenticado no sistema	
Pós condição:	Altera o preço de um dado componente	
Comportamento Normal	**Ator**	**Sistema**
	1.Indica o nome do componente que pretende alterar preço e o novo preço	
		2.Valida dados inseridos
		3.Altera o preço do componente indicado
Exceção 1 [Funcionário cancela operação] (passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Exceção 2 [O componente indicado não está registado no sistema] (passo 2)		2.1 Indica que o componente indicado não está registado no sistema
		2.2 Termina processo

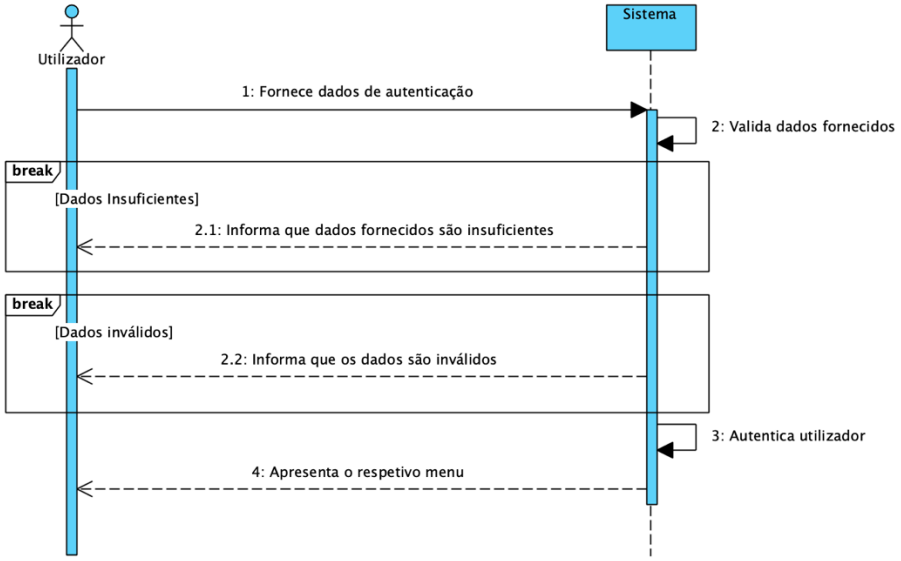
Atualizar Stock:


FuncionarioFabrica
Sistema
1: Fornece o código de um componente
break
[Funcionário cancela operação]
1.1: Indica que pretende cancelar operação
2: Valida dados
break
[Código inválido]
2.1: Indica que o código fornecido é inválido
3: Apresenta resultados obtidos
4: Indica a quantidade do stock a adicionar/retirar do componente
5: Verifica o valor inserido
break
[Stock retirado superior ao stock existente]
5.1: Informa que a quantidade retirada é superior ao stock existente
6: Atualiza stock
7: Informa que o stock foi atualizado


FuncionarioFabrica
ConfiguraFacil
componentes :
Map<String,Componente>
c : Componente
1: atualizarStock(nomeC,valor)
break
[(!componentes.containsKey(nomeC))]
1.1: Componente indicado não existe no sistema
1.2: c = get(idC)
break
[((c.getStock() + valor)<0)]
1.3: O valor do stock inserido é inválido
1.4: setStock(getStock()+valor)
1.5: put(nomeC,c)

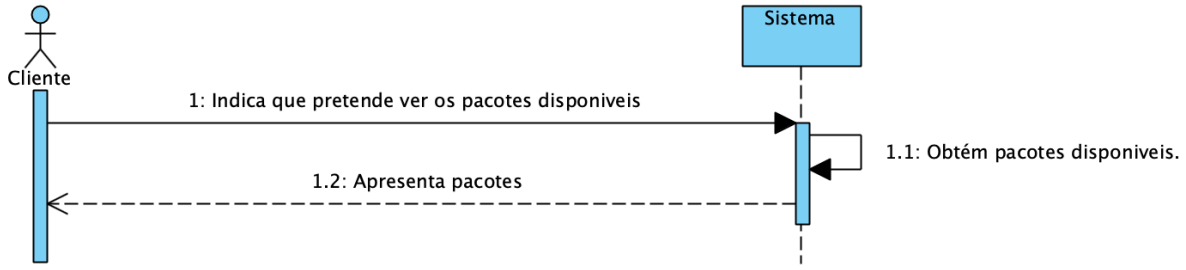
Use Case:	Atualizar stock	
Descrição:	Atualizar a quantidade em stock de um componente	
Pré condição:	Utilizador autenticado no sistema	
Pós condição:	Atualizada a quantidade em stock de um componente	
Comportamento Normal	**Ator**	**Sistema**
	1.Fornece o código de um componente	
		2.Fornece dados associados ao componente
	3.Indica a quantidade a adicionar/retirar	
		4.Verifica os dados inseridos
		5.Atualiza no sistema o stock do componente
		6.Informa que o stock foi atualizado
Exceção 1 [Utilizador pretende cancelar operação] (passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Exceção 2 [Código inválido] (passo 2)		2.1 Indica que o código fornecido é inválido
		2.2 Termina processo
Exceção 3 [Stock retirado superior ao stock existente] (passo 4)		4.1 Informa que a quantidade retirada é superior ao stock existente
		4.2 Termina processo

Autenticar no sistema:



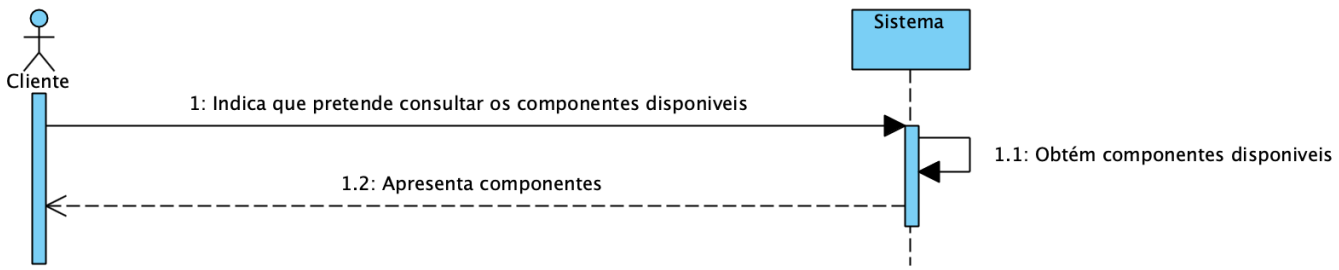
Use Case:	Autenticar	
Descrição:	Autenticar um utilizador no sistema	
Pré condição:		
Pós condição:	Efetuada a autenticação do utilizador	
Comportamento Normal	**Ator**	**Sistema**
	1.Fornece dados pessoais	
		2.Valida os dados fornecidos
		3.Autentica o utilizador
		4.Apresenta o respetivo menu
Exceção 1 [dados insuficientes] (passo 2)		2.1 Informa que dados fornecidos são insuficientes.
		2.2 Termina processo
Exceção 2 [dados invalidos] (passo 2)		2.3 Indica que os dados fornecidos são invalidos
		2.4 Termina processo

Consultar pacotes disponíveis:



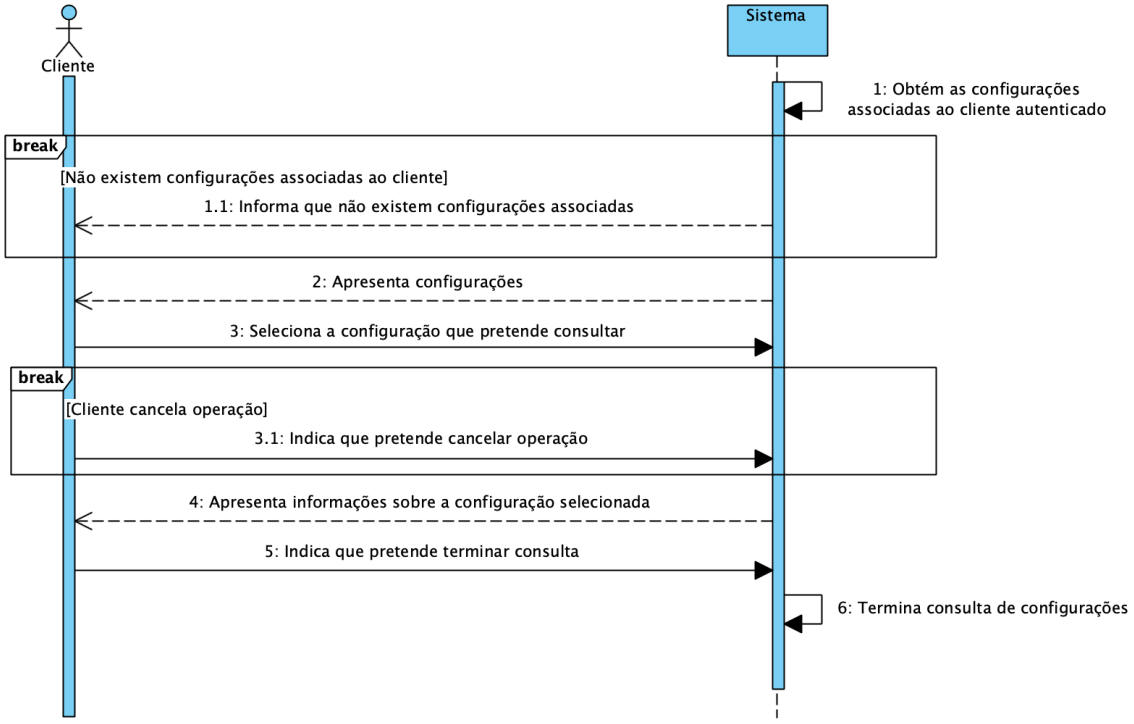
Use Case:	Consultar Pacotes Disponveis	
Descrição:	Cliente consulta pacotes disponiveis	
Pré condição:	Cliente autenticado no sistema	
Pós condição:	Aprensenta pacotes disponiveis	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém pacotes disponiveis.
		2.Apresenta pacotes.

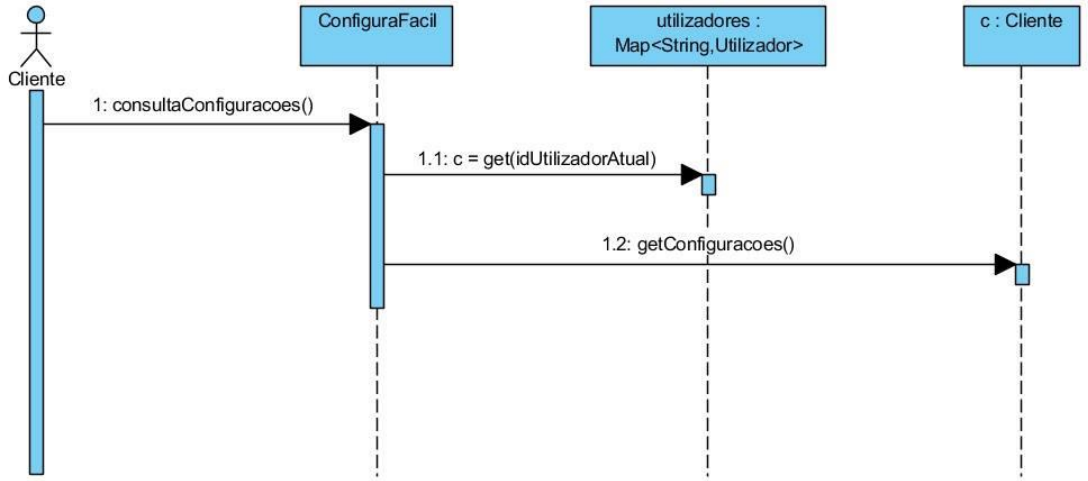
Consultar componentes disponíveis:



Use Case:	Consultar Componentes Disponiveis	
Descrição:	Cliente consulta os componentes disponiveis	
Pré condição:	Cliente autenticado no sistema	
Pós condição:	Apresenta componentes disponiveis	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém componentes disponiveis.
		2.Apresenta componentes.

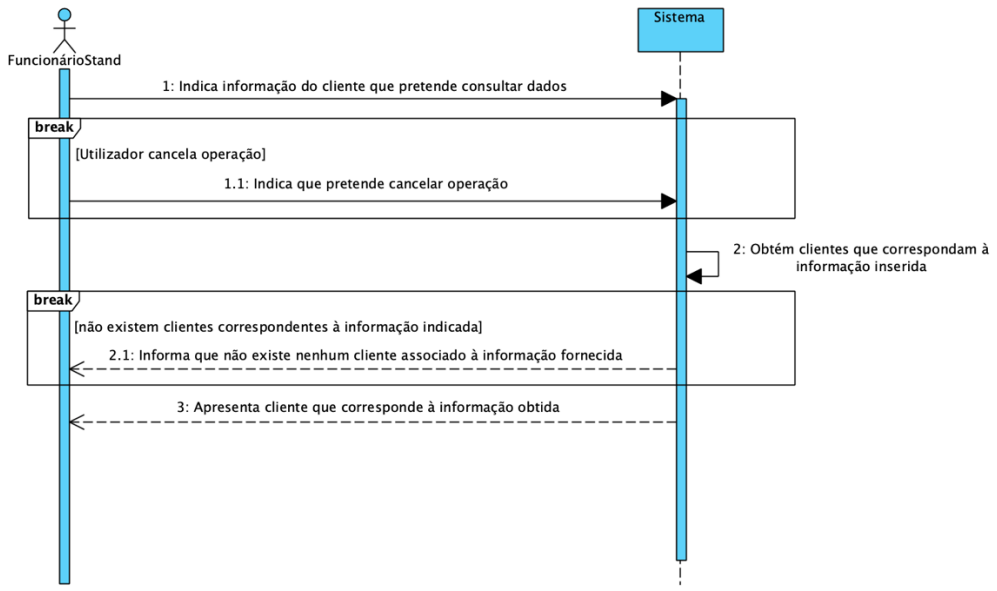
Consultar configurações:





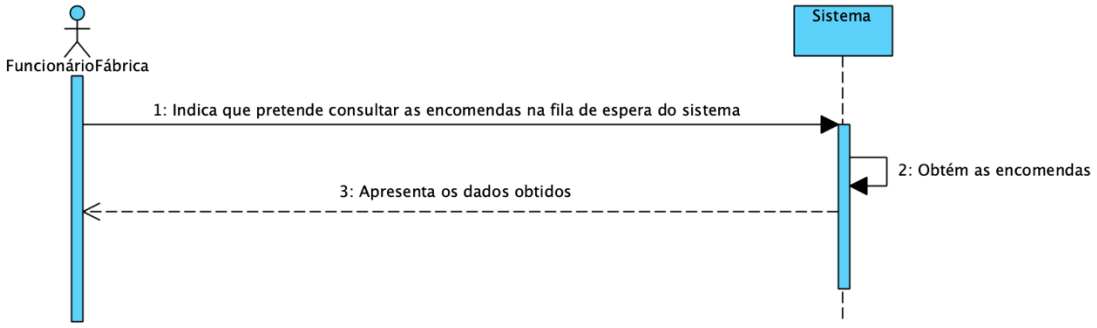
Use Case:	Consultar configurações efetuadas	
Descrição	Cliente consulta as suas configurações efetuadas	
Pré condição:	Cliente autenticado	
Pós condição:	São apresentadas as configurações efetuadas	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém configurações associadas ao cliente autenticado
		2.Apresenta configurações
	3.Seleciona a configuração que pretende consultar	
		4.Apresenta informações sobre a configuração selecionada
	5.Indica que pretende terminar consulta	
		6.Termina consulta de configurações efetuadas
Exceção 1 [não existem configurações associadas ao cliente] (passo 1)		1.1 Informa que não existem configurações associadas
		1.2 Termina processo
Exceção 2 [cliente cancela operação] (passo 3)	3.1 Indica que pretende cancelar operação	
		3.2 Termina processo

Consultar dados do cliente:



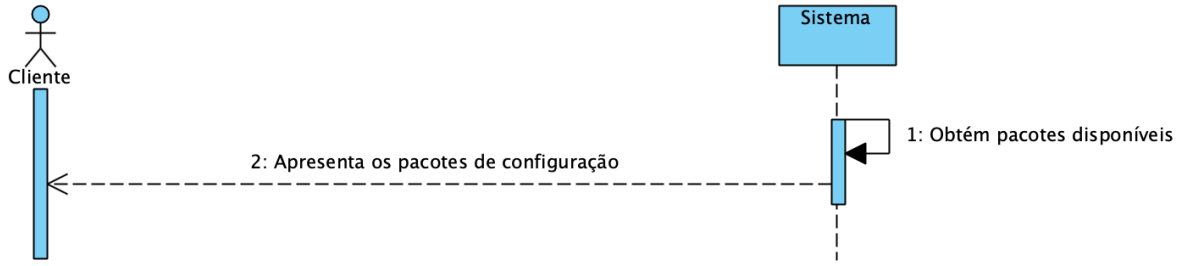
Use Case:	Consultar dados de um cliente	
Descrição	Funcionário do stand consulta os dados de um cliente	
Pré condição:	Funcionário do stand autenticado	
Pós condição:	São apresentados os dados referentes a um cliente	
Comportamento Normal	**Ator**	**Sistema**
	1. Indica nome do cliente que pretende consultar	
		2. Obtém cliente que corresponde à informação fornecida
		3. Apresenta dados do cliente correspondente
Exceção 1 [não existem clientes correspondentes à informação indicada] (passo 2)		2.1 Informa que não existe nenhum cliente associado à informação fornecida
		2.2 Termina processo
Exceção 2 [Funcionário cancela operação] (passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo

Consultar encomendas:



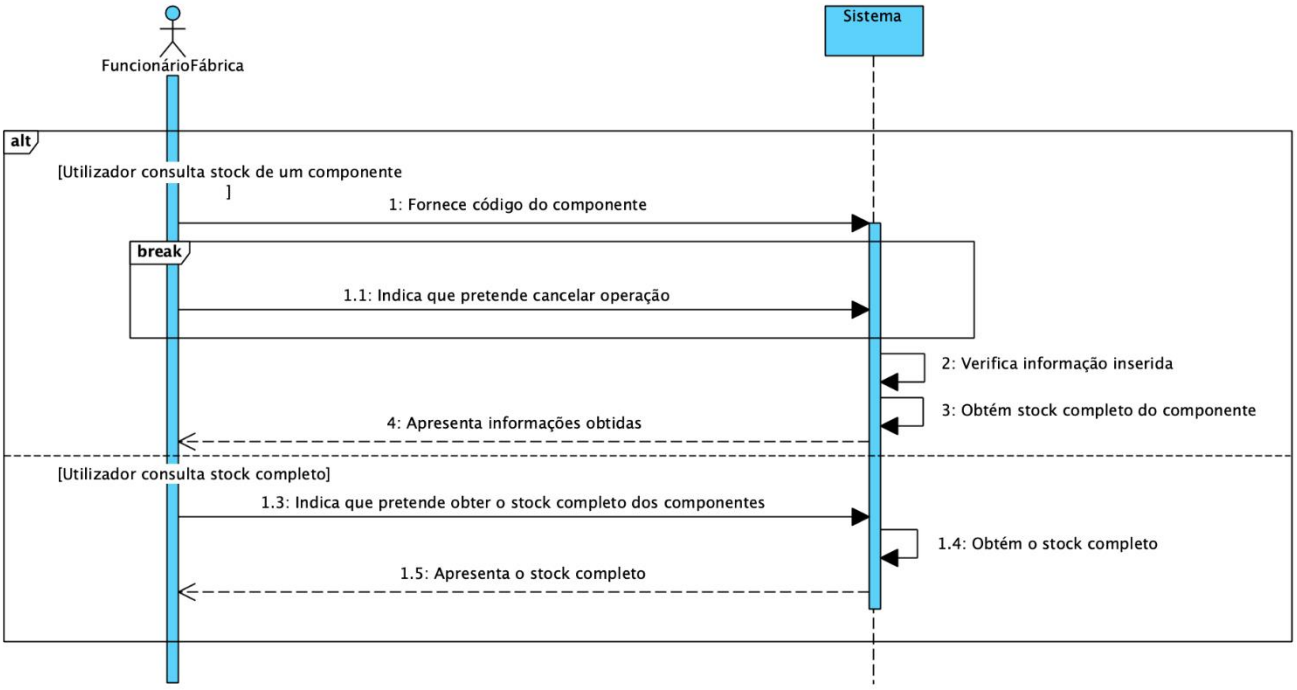
Use Case:	Consultar encomendas	
Descrição:	Funcionário da fábrica pretente consultar as encomendas existentes na fábrica	
Pré condição:	Funcionário da fábrica autenticado no sistema	
Pós condição:	É apresentada a lista das encomendas	
Comportamento Normal	**Ator**	**Sistema**
	1.Indica que pretende consultar as encomendas na fila de espera da fábrica	
		2.Obtém encomendas
		3.Apresenta os dados obtidos

Consultar pacotes disponíveis:



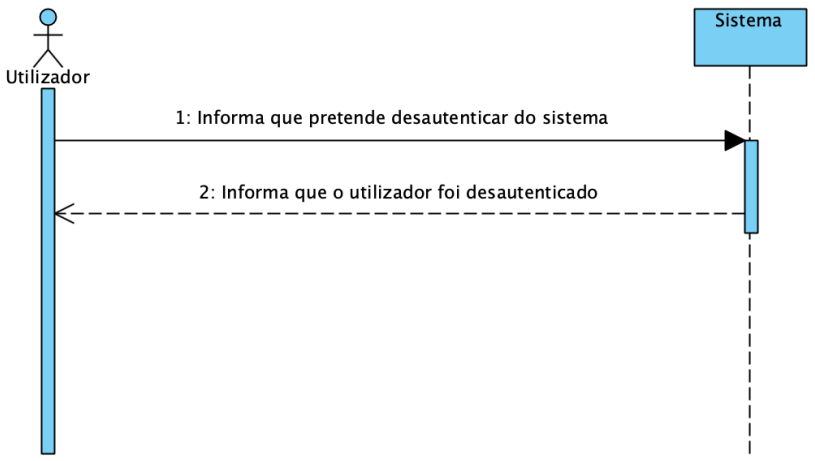
Use Case:	Consultar Pacotes Disponveis	
Descrição:	Cliente consulta pacotes disponiveis	
Pré condição:	Cliente autenticado no sistema	
Pós condição:	Aprensenta pacotes disponiveis	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém pacotes disponiveis.
		2.Apresenta pacotes.

Consultar Stock:



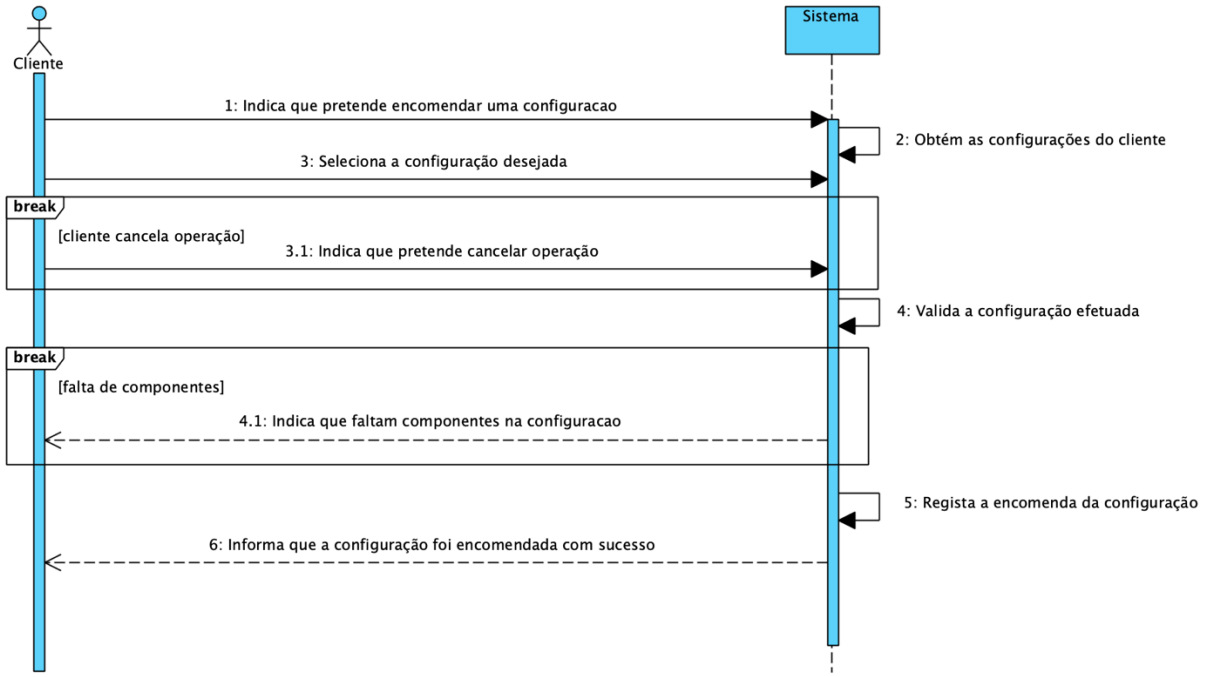
Use Case:	Consultar stock	
Descrição:	Obter o stock de componentes do sistema	
Pré condição:	Utilizador autenticado no sistema	
Pós condição:	É apresentado o stock de componentes do sistema	
Comportamento Normal	**Ator**	**Sistema**
	1.Fornece código do componente	
		2. Verifica a informação inserida
		3. Obtém o stock do componente selecionado
		3. Apresenta informações obtidas
Exceção 1 [Utilizador cancela operação] (passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Alternativa 1 [Funcionário pretende consultar stock completo] (passo 1)	1.3 Indica que pretende obter o stock de todos os componentes	
		1.4 Obtém o stock completo
		1.5 Apresenta o stock completo
Excepção 2 [código do componente invalido] (passo 2)		2.1 Informa que o codigo não corresponde a nenhum componente do sistema
		2.2 Termina o processo

Sair do sistema:



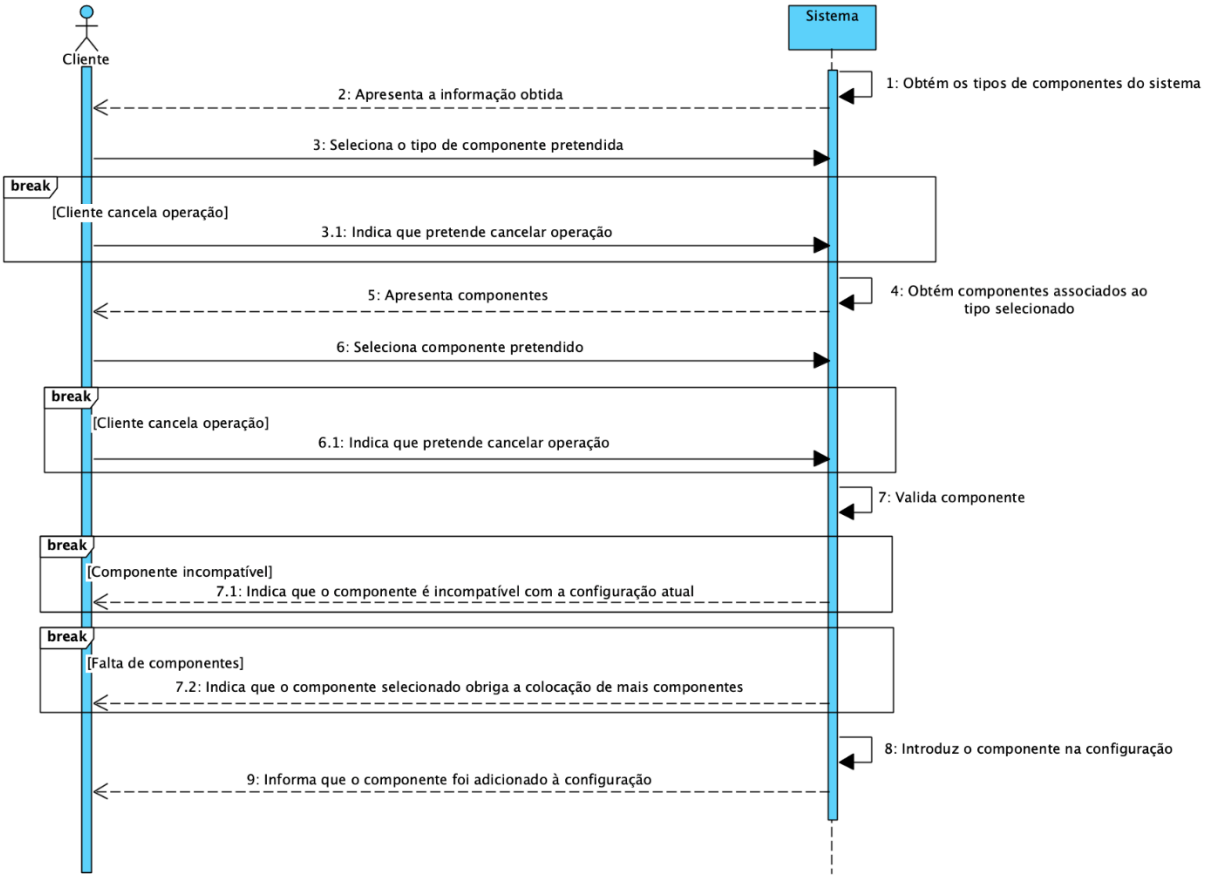
Use Case:	Desautenticar	
Descrição:	Utilizador efetua logout do sistema	
Pré condição:	Utilizador autenticado no sistema	
Pós condição:	Efetuada a desautenticação do utilizador	
Comportamento Normal	**Ator**	**Sistema**
	1.Informa que pretende desautenticar se do sistema	
		2.Informa que o utilizador foi desautenticado

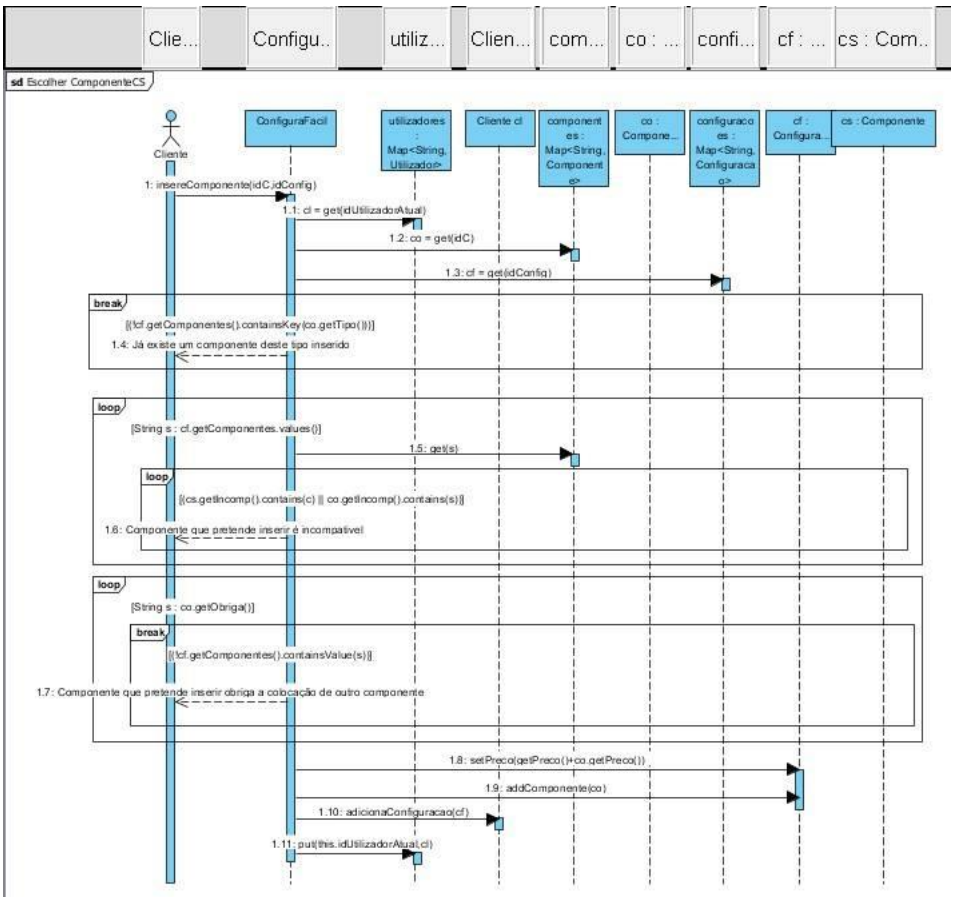
Encomendar configuração:



Use Case:	Encomendar configuração	
Descrição	Cliente encomenda a configuração de um carro	
Pré condição:	Cliente autenticado	
Pós condição:	Configuração selecionada pelo cliente é encomendada	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém as configurações do cliente
		2.Apresenta as configurações
	3.Seleciona a configuração desejada	
		4.Valida a configuração efetuada
		5.Regista a encomenda da configuração
		6.Informa que a configuração foi encomendada com sucesso
Exceção 1 [cliente cancela operação] (passo 3)	3.1 Indica que pretende cancelar operação	
		3.2 Termina processo
Exceção 2 [falta de componentes] (passo 4)		4.3 Indica que faltam componentes na configuração
		4.4 Termina processo

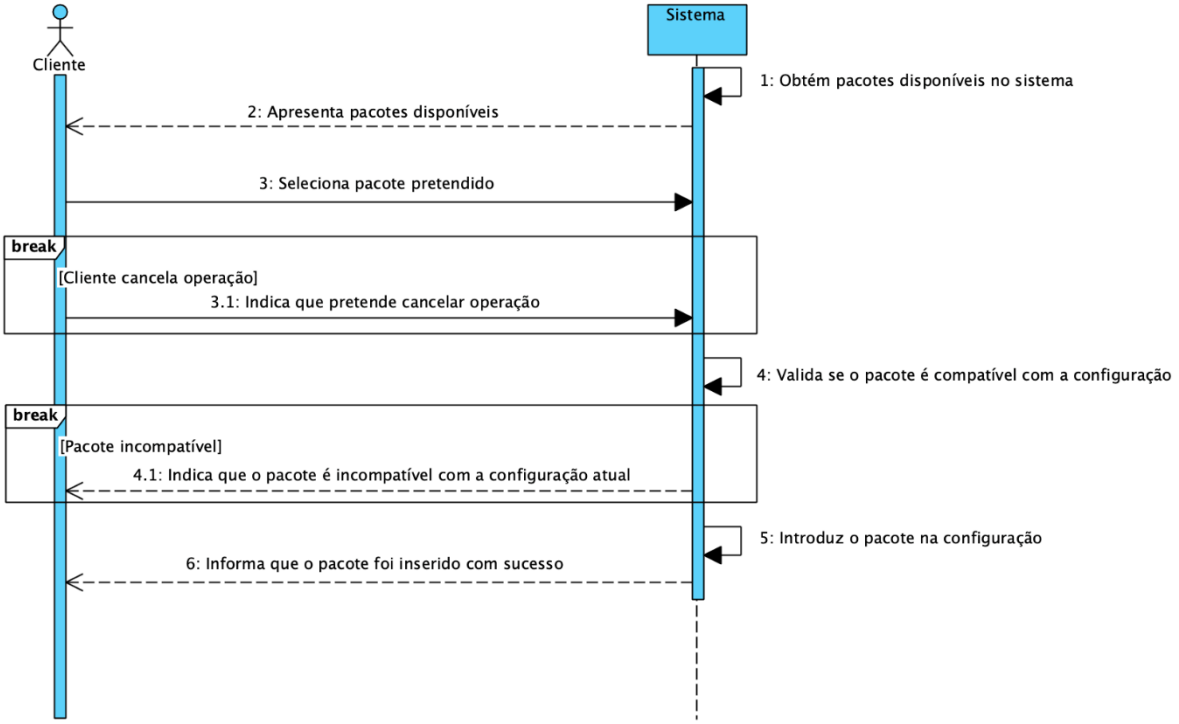
Escolher componente:

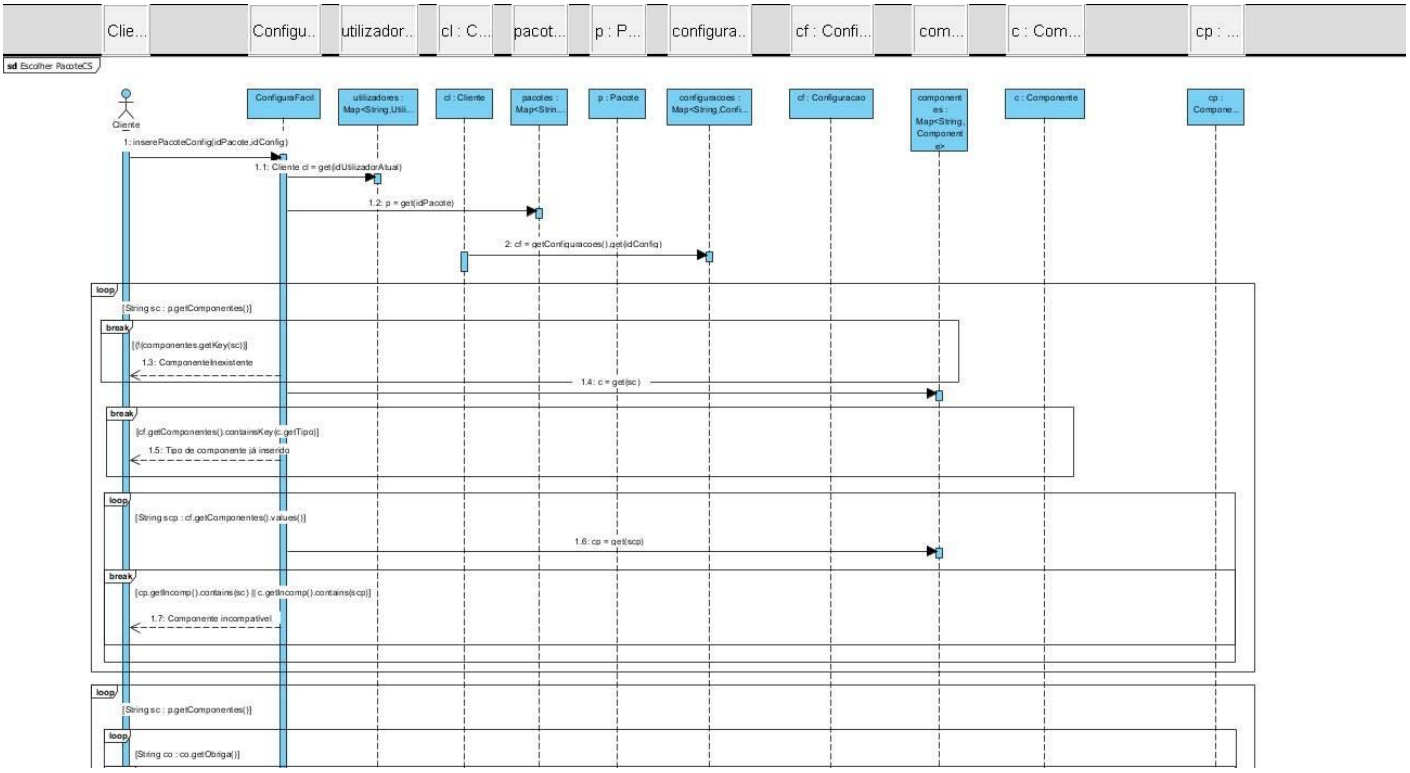


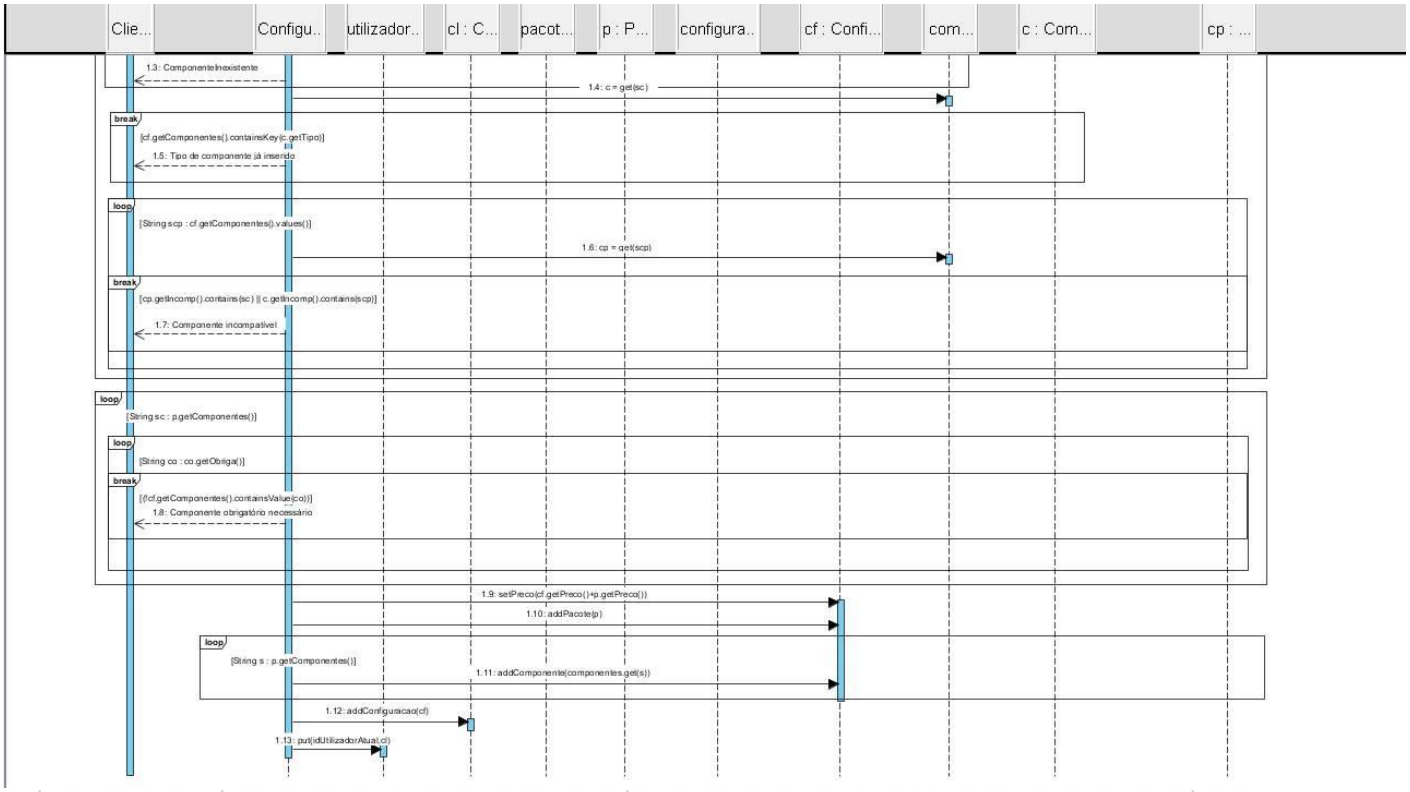

Clie...
Configu..
utiliz...
Clien...
com...
co : ...
confi...
cf : ...
cs : Com..
sd Escolher ComponenteCS
Cliente
ConfiguraFacil
utilizadores : Map<String, Utilizador>
Cliente cl
componentes : Map<String, Componente>
co : Compone...
configuracoes : Map<String, Configuracao>
cf : Configura...
cs : Componente
1: insereComponente(idC,idConfig)
1.1: cl = get(idUtilizadorAtual)
1.2: co = get(idC)
1.3: cf = get(idConfig)
break
[!(cf.getComponentes().containsKey(co.getTipo()))]
1.4: Já existe um componente deste tipo inserido
loop
[String s : cf.getComponentes.values()]
1.5: get(s)
loop
[(cs.getIncomp().contains(c) || co.getIncomp().contains(s))]
1.6: Componente que pretende inserir é incompativel
loop
[String s : co.getObriga()]
break
[!(cf.getComponentes().containsValue(s))]
1.7: Componente que pretende inserir obriga a colocação de outro componente
1.8: setPreco(getPreco()+co.getPreco())
1.9: addComponente(co)
1.10: adicionaConfiguracao(cf)
1.11: put(this.idUtilizadorAtual,cl)

Use Case:	Escolher componente	
Descrição	Cliente escolhe uma componente para a sua configuração	
Pré condição:	Cliente autenticado	
Pós condição:	Componente introduzida na configuração	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém os tipos de componentes do sistema
		2.Apresenta a informação obtida
	3.Seleciona o tipo de componente pretendido	
		4.Obtém componentes associados ao tipo selecionado
		5.Apresenta componentes
	6.Seleciona o componente pretendido	
		7.Valida a compatibilidade do componente selecionado
		8.Verifica se o componente obriga a colocação de mais componentes
		9.Introduz o componente na configuração
		10.Informa que o componente foi introduzido com sucesso
Exceção 1 [cliente cancela operação] (passo 3)	3.1 Indica que pretende cancelar operação	
		3.2 Termina processo
Exceção 2 [cliente cancela operação] (passo 6)	6.1 Indica que pretende cancelar operação	
		6.2 Termina processo
Excecão 3 [componente incompatível] (passo 7)		7.1 Indica que o componente é incompatível com a configuração atual
		7.2 Termina processo
Execeção 4 [O componente inserido obriga a colocação de componentes] (passo 8)		8.1 Indica que o componente escolhido obriga a colocação de outros componentes
		8.2 Termina processo

Escolher pacote:

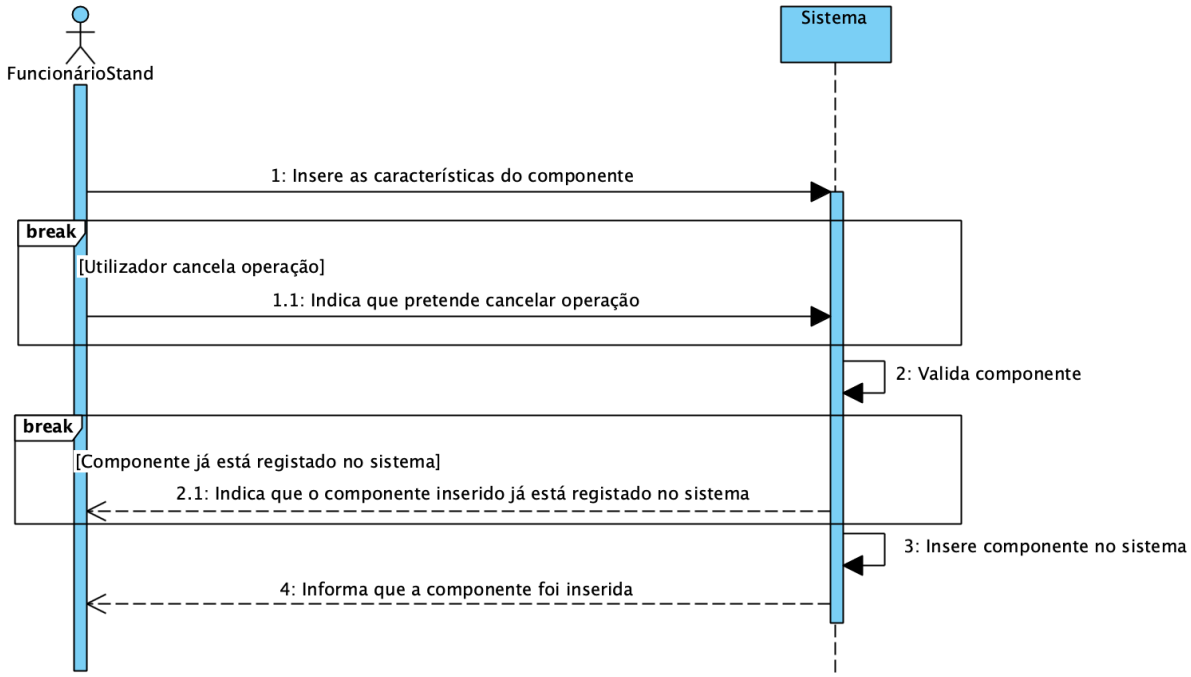

Cliente
Sistema
1: Obtém pacotes disponíveis no sistema
2: Apresenta pacotes disponíveis
3: Seleciona pacote pretendido
break
[Cliente cancela operação]
3.1: Indica que pretende cancelar operação
4: Valida se o pacote é compatível com a configuração
break
[Pacote incompatível]
4.1: Indica que o pacote é incompatível com a configuração atual
5: Introduz o pacote na configuração
6: Informa que o pacote foi inserido com sucesso


Clie...
Configu..
utilizador..
cl : C...
pacot...
p : P...
configura..
cf : Confi...
com...
c : Com...
cp : ...
sd Escolher PacoteCS
Cliente
ConfiguraFacil
utilizadores : Map<String,Utili...
cl : Cliente
pacotes : Map<Strin...
p : Pacote
configuracoes : Map<String,Confi...
cf : Configuracao
componentes : Map<String,Componente>
c : Componente
cp : Compone...
1: inserePacoteConfig(idPacote,idConfig)
1.1: Cliente cl = get(idUtilizadorAtual)
1.2: p = get(idPacote)
2: cf = getConfiguracoes().get(idConfig)
loop
[String sc : p.getComponentes()]
break
[!(componentes.getKey(sc))]
1.3: ComponenteInexistente
1.4: c = get(sc)
break
[cf.getComponentes().containsKey(c.getTipo)]
1.5: Tipo de componente já inserido
loop
[String scp : cf.getComponentes().values()]
1.6: cp = get(scp)
break
[cp.getIncomp().contains(sc) || c.getIncomp().contains(scp)]
1.7: Componente incompatível
loop
[String co : co.getObriga()]
loop
[String co : co.getObriga()]



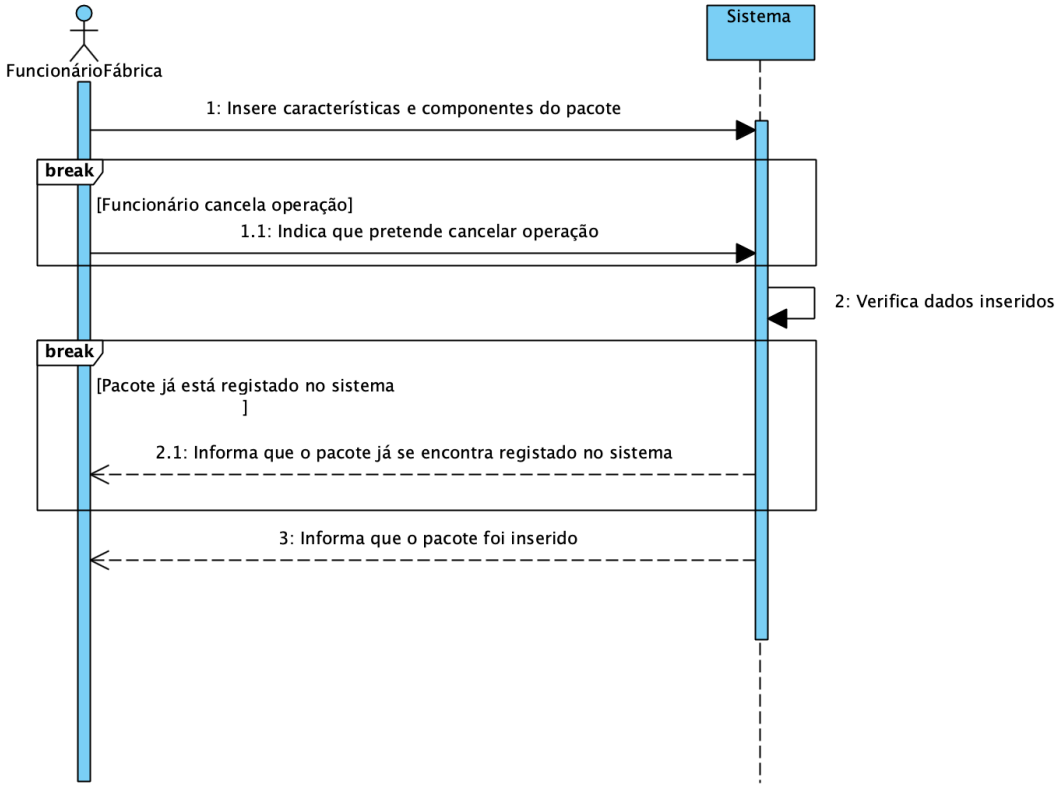
Use Case:	Escolher pacote	
Descrição	Cliente escolhe um pacote para a sua configuração	
Pré condição:	Cliente autenticado	
Pós condição:	Pacote introduzido na configuração	
Comportamento Normal	**Ator**	**Sistema**
		1.Obtém os pacotes disponíveis no sistema
		2.Apresenta os pacotes disponíveis
	3. Seleciona o pacote pretendido	
		4.Valida a compatibilidade do pacote selecionado com a configuração
		5.Verifica se o pacote obriga a colocação de mais componentes
		6.Introduz o pacote na configuração
		7.Informa que o pacote foi inserido com sucesso
Exceção 1 [cliente cancela operação] (passo 3)	3.1 Indica que pretende cancelar operação	
		3.2 Termina processo
Exceção 2 [pacote incompatível] (passo 4)		4.1 Indica que o pacote é imcompatível com a configuração atual
		4.2 Termina processo
Exceção 3 [falta de componentes] (passo 5)		5.1 Indica que é necessário adicionar mais componentes
		5.2 Termina processo

Inserir novo componente:



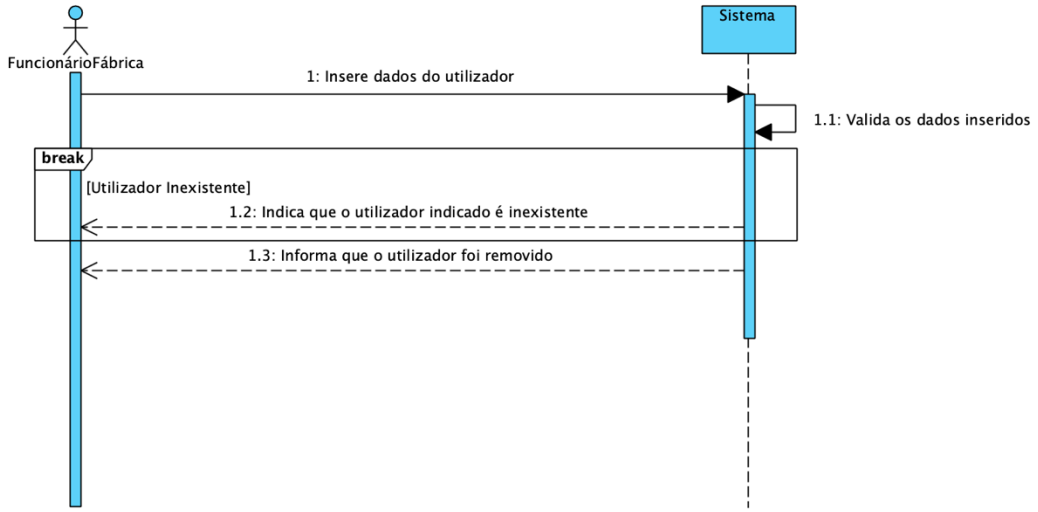
Use Case:	Inserir novo componente	
Descrição	Funcionário do stand insere um novo componente no sistema	
Pré condição:	Funcionário do stand autenticado	
Pós condição:	Componente inserido no sistema	
Comportamento Normal	**Ator**	**Sistema**
	1.Insere características do componente	
		2.Valida características do componente
		3.Insere componente no sistema
		3.Informa que a componente foi inserida
Exceção 1 [Funcionário cancela operação] (Passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Exceção 2 [O componente indicado já está registado no sistema] (Passo 2)		2.1 Informa que o componente inserido já existe no sistema
		2.2 Termina processo

Inserir novo pacote:



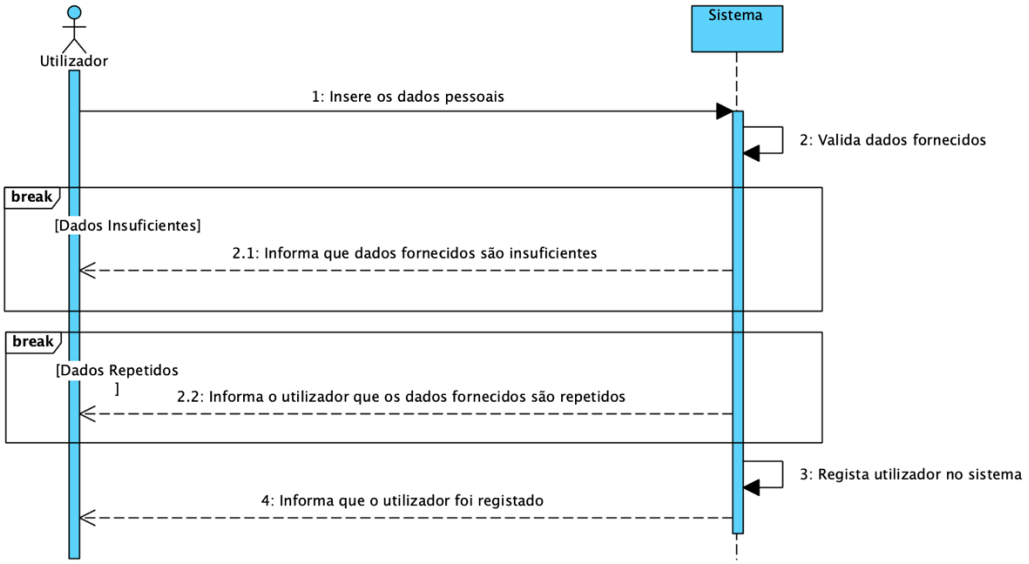
Use Case:	Inserir novo pacote	
Descrição	Funcionário do stand insere um novo pacote no sistema	
Pré condição:	Funcionário do stand autenticado	
Pós condição:	Pacote inserido no sistema	
Comportamento Normal	**Ator**	**Sistema**
	1.Insere características e componentes do pacote	
		2. Verifica dados inseridos
		3. Insere pacote no sistema
		4 .Informa que o pacote foi inserido
Exceção 1 [Funcionário cancela operação] (Passo 1)	1.1 Indica que pretende cancelar operação	
		1.2 Termina processo
Exceção 2 [Pacote já se encontra registado no sistema] (Passo 2)		2.1 Informa que o pacote já se encontra registado no sistema
		2.2 Termina processo

Remover utilizador:



Use Case:	Remover Utilizador	
Descrição:	Administrador remove utilizador do sistema	
Pré condição:	Administrador autenticado	
Pós condição:	Administrador remove utilizador	
Comportamento Normal	**Ator**	**Sistema**
	1. Insere dados do utilizador	
		2.Valida os dados inseridos
		3. Informa que o utilizador foi removido
Exceção 1 [Utilizador Inexistente] (passo 2)		2.1 Indica que o utilizador indicado é inexistente
		2.2 Termina processo

Registrar utilizador:



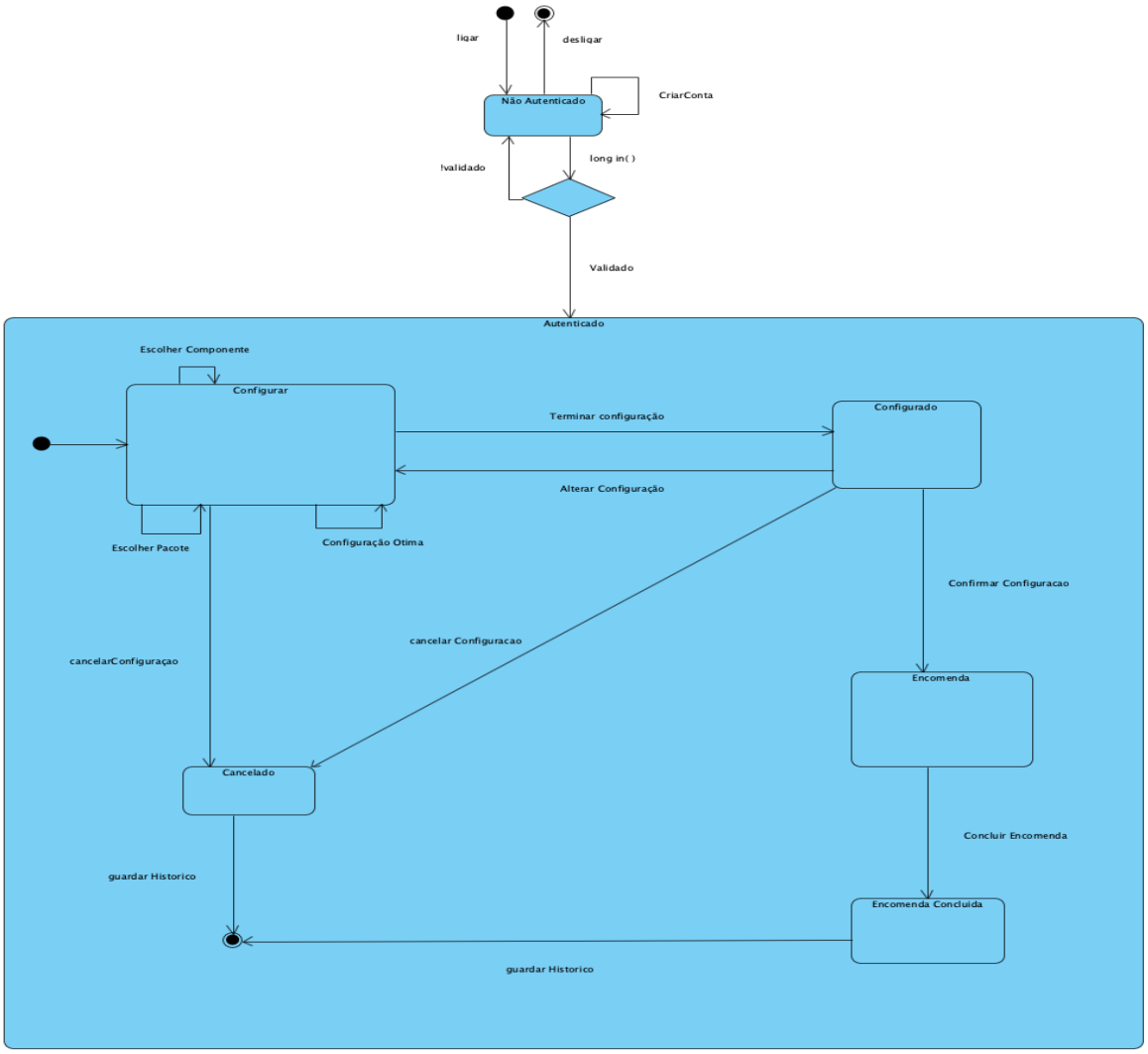
Use Case:	Registrar	
Descrição:	Registrar um utilizador no sistema	
Pré condição:		
Pós condição:	Utilizador registado no sistema	
Comportamento Normal	**Ator**	**Sistema**
	1.Insere os dados pessoais	
		2.Valida os dados inseridos
		3.Regista utilizador no sistema
		4.Informa que o utilizador foi registado
Exceção 1 [dados insuficientes] (passo 2)		2.1 Informa que os dados fornecidos são insuficientes
		2.2 Termina processo
Exceção 2 [dados repetidos] (passo 2)		2.3 Indica que os dados fornecidos são repetidos
		2.4 Termina processo

Diagrama de Estados

De modo a representar os diferentes estados do sistema foram representados dois diagramas de estado. Um diagrama que engloba todo o processo desde o utilizador criar a sua conta no sistema até ao veículo ser produzido. Um segundo diagrama representa todos os estados que a interface gráfica que o utilizador interage durante o processo de configuração tem.

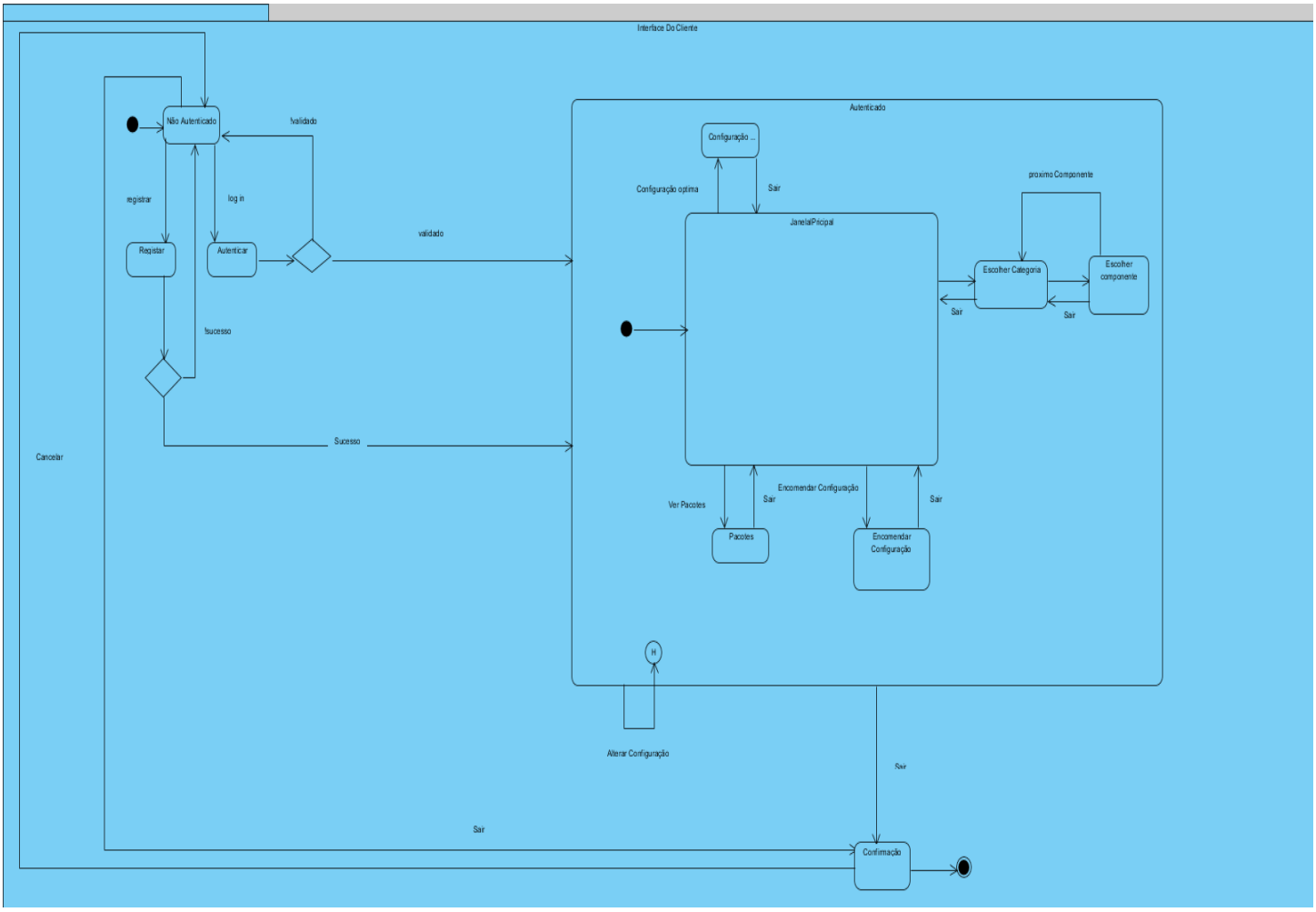
Diagrama que representa o Sistema

O processo começa com a autenticação no sistema. Deste modo são definidos dois estados, estar e não estar autenticado no sistema. Ao não estar autenticado, um utilizador pode criar uma conta ou fazer um log in, permitindo a mudança de estado.
Após a autenticação no sistema começa o processo de configuração, ou seja, estamos nesse estado. Este estado pode ser terminado com o cancelamento da configuração ou com a conclusão deste.
Após a conclusão da configuração, o sistema entra num estado de configurado. Neste estado o cliente poderá voltar atras a alterar a configuração, ou seja, volta para o estado de configuração. O cliente por outro lado pode cancelar a configuração feita ou pode confirmar, que efetivamente representa a formação de uma encomenda na fábrica.
O carro enquanto se encontra em espera/produção encontra-se no estado de encomenda que passará para o estado final encomenda concluída após a sua produção.
Deste modo é descrito todos os estados que da conceção de um veículo neste sistema.



Protótipo da interface gráfica.

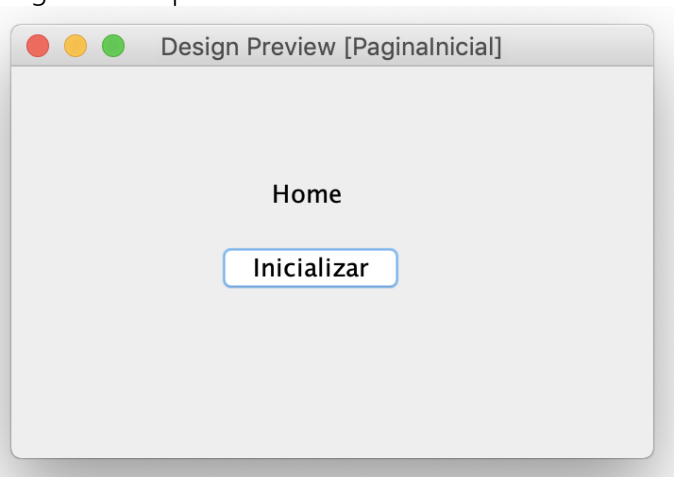
De modo a representar o protótipo da interface gráfica, inicialmente foi implemento um diagrama de estados para descrever a evolução do processo de configuração.
O utilizador numa fase inicial necessita de se autenticar no sistema, logo ele encontra-se num estado de não autenticado.
Ele pode criar uma conta, fazer log in (caso já possua uma conta) ou cancelar todo o processo e sair do sistema.
Com o utilizador autenticado ele encontrará uma janela principal. Aqui ele poderá configurar o seu veículo componente a componente após escolher uma categoria, escolher um pacote ou pedir a configuração ótima. Sobre qualquer configuração ele pode voltar atras e recomeçar a ver a sua configuração desde o início com a invocação do alterar configuração.
O utilizador poderá após a configuração enviar a confirmá-la e assim uma encomenda é gerada ou poderá cancelar e sair do sistema a qualquer momento.
Diagrama de estado com a interface gráfica:



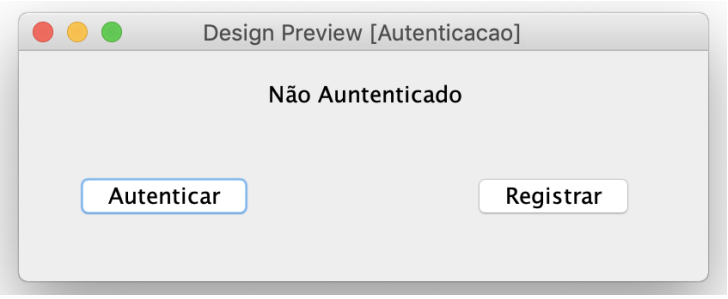
Numa segunda fase do projeto foi desenvolvida uma interface gráfica que permite o cliente efetuar a configuração do veículo e no lado da fábrica um funcionário da fábrica poder obter as encomendas que podem ser realizadas ou alterar o stock dos componentes.

Descrição da evolução da interface gráfica:

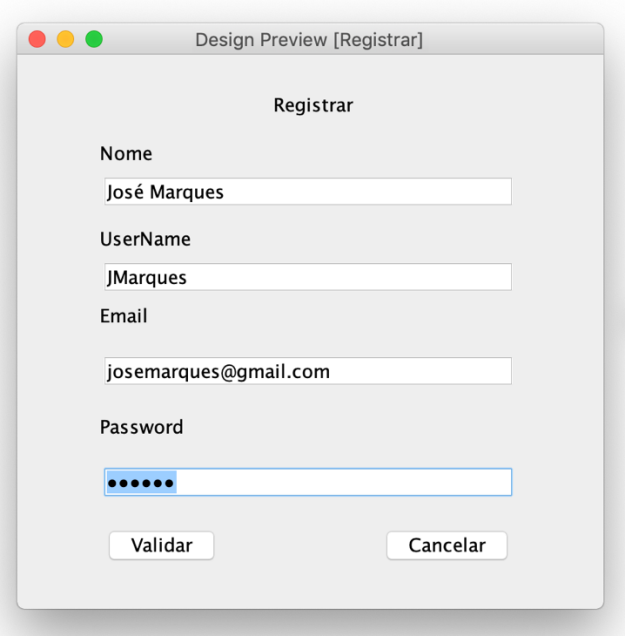
Página inicial para o cliente:



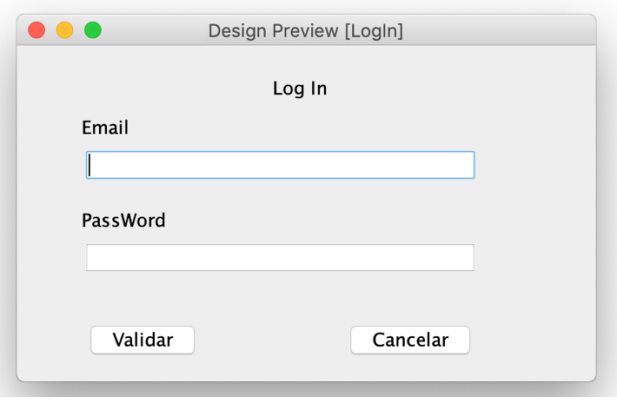
Este pode autenticar-se ou registrar-se no sistema:



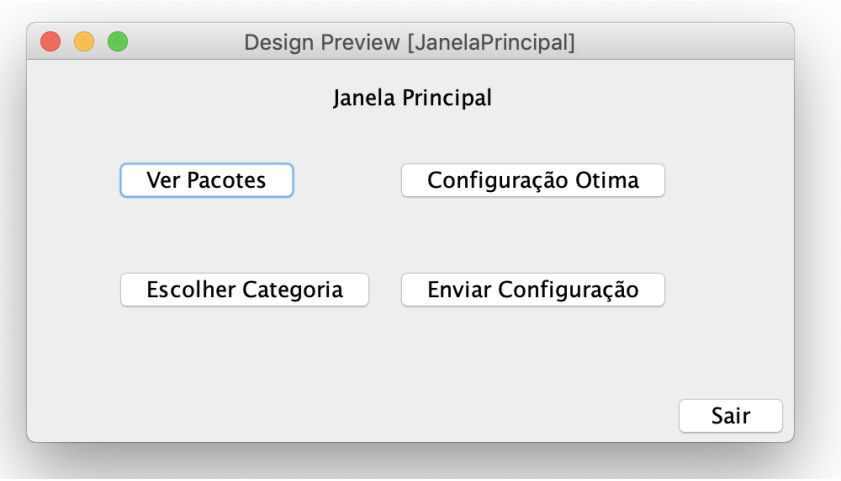
Registrar-se no sistema:



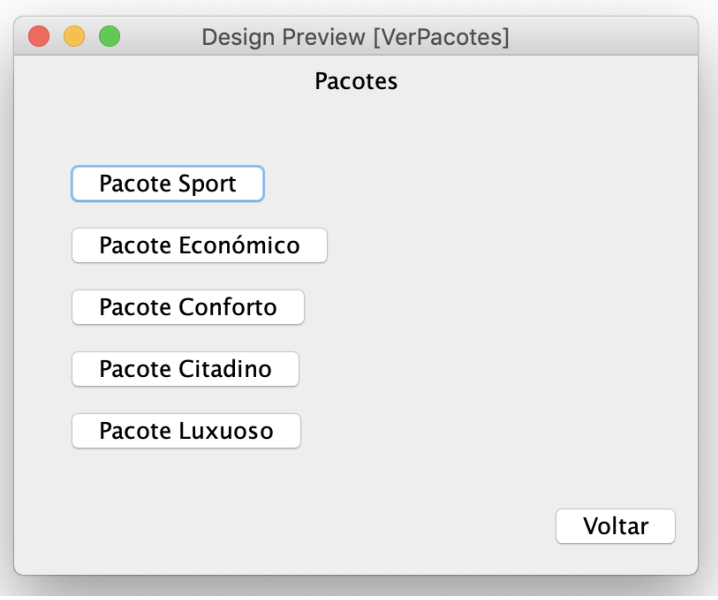
Autenticar:



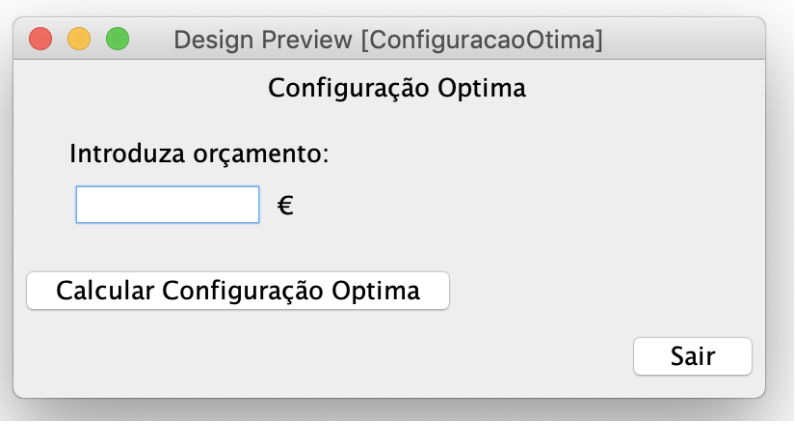
Após a entrada no sistema, o cliente encontra o menu principal:



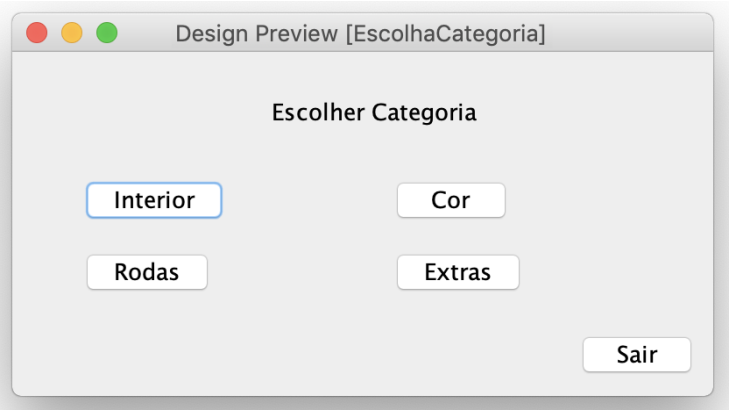
Caso selecione pacote, ele pode visualizar os pacotes disponíveis e selecioná-los:



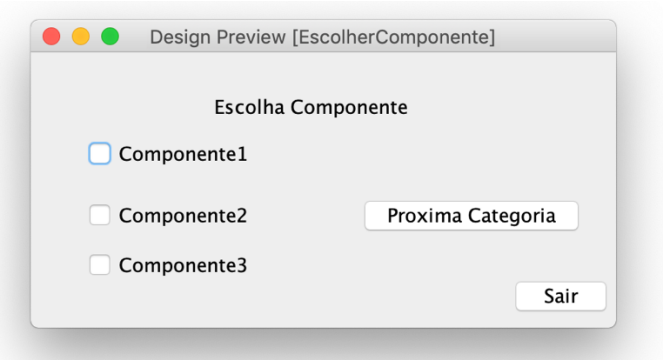
Caso selecione configuração ótima, ele diz o seu orçamento e o sistema responde o pacote ótimo:



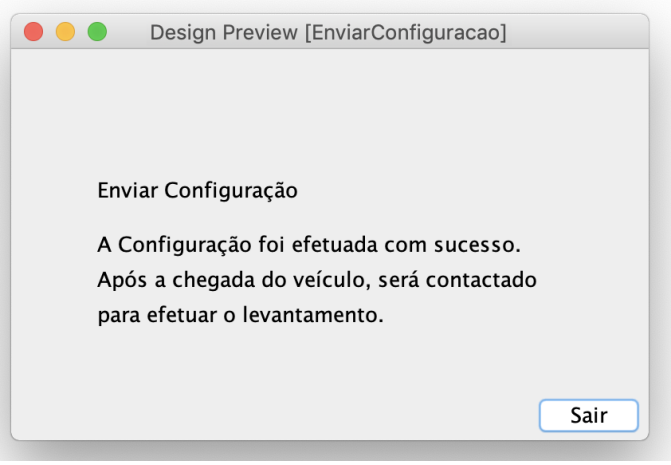
Caso escolha uma categoria, um cliente escolhe que opção pretende configurar:



Após a escolha da categoria, o cliente escolhe uma componente que será adicionada à sua configuração:



Caso escolha enviar configuração e esta está completa, é registrada a encomenda e enviada para a fabrica:



No lado da fábrica, existe um menu principal que permite um funcionário obter cada uma das informações associadas à produção de um veículo:

Menu que permite a qualquer momento um utilizador sair do sistema:

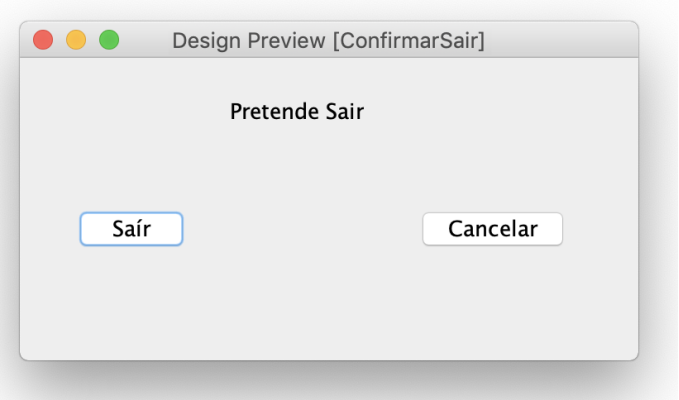
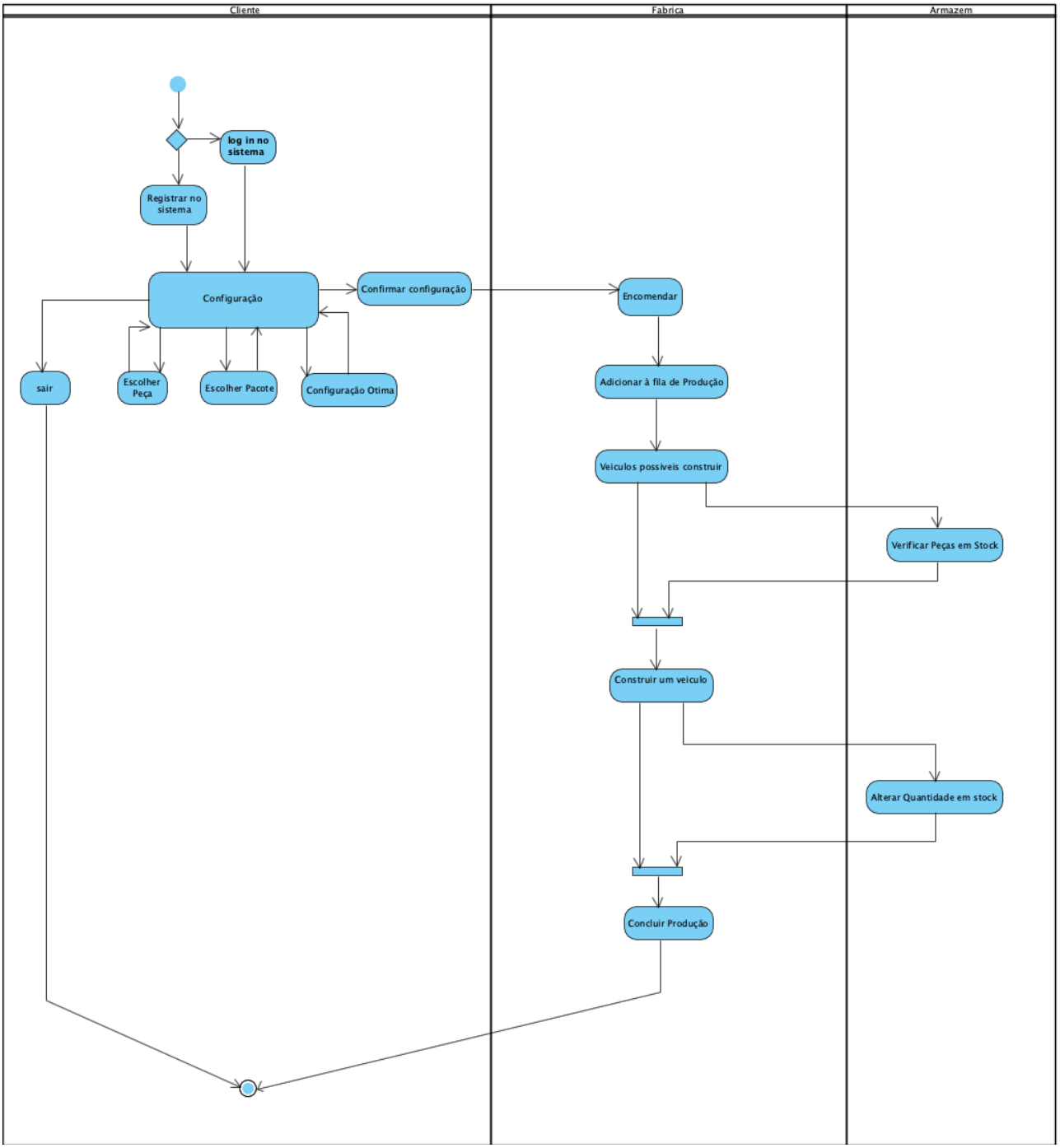


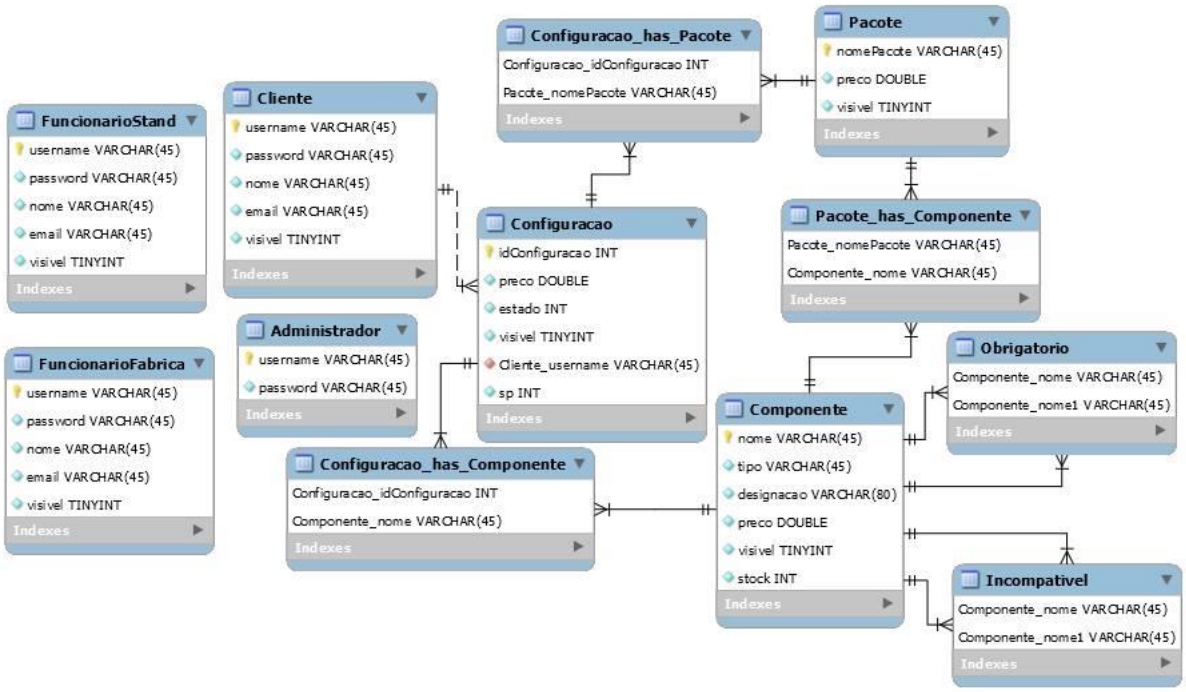
Diagrama de atividades

Diagrama de atividades que descreve o processo desde a configuração feita por o utilizador até à produção do veículo.



Base de Dados

Enquanto que em java tínhamos os maps de “Componentes Obrigatórios” e “Componentes Incompatíveis” (ambos do tipo Map<String,String>), no momento de elaboração do modelo lógico, decidimos criar duas novas tabelas (Obrigatorio e Incompativel) que estabelecem a ligação entre os nomes de dois componentes , recreando assim esses mapeamentos .
Nas tabelas Configuracao, Pacote, Componente, Cliente, FuncionarioStand e FuncionarioFabrica, para que não sejam eliminados registos, introduzimos o atributo “visível”. Assim garantimos que, quando for necessário que uma entrada de uma destas tabelas deixe de ser válida (ex: se a fabrica deixar de usar um determinado componente), basta colocar o atributo “visível” dessa entrada a falso.



Conclusão

Em relação a este projeto tivemos grande dificuldade em fazer a ligação entre a interface, a base de dados e a nossa camada de dados, de modo a que não fomos bem sucedidos nesse aspeto.
Como a configuração é um processo sequencial em que uma componente é escolhida após outra isso iria criar use cases extremamente longos. Isso iria contra os princípios lecionados nas aulas teóricas, em que um use case não deve ter mais que dez passos. De modo a não implementar use cases com todas as possibilidades de configuração e alternativas (como por exemplo escolher um pacote ou requerer a configuração ótima da configuração) foi decidido separar os use cases. Deste modo o processo foi dividido nas suas opções básicas: Escolher componente, escolher pacote ou escolher opção ótima.
O sistema em si é capaz de detetar se uma configuração é valida e assim após a configuração o utilizador é capaz de invocar um use case que termina o processo e regista a encomenda, em vez de concentrar todo o processo de configuração num só único use case.