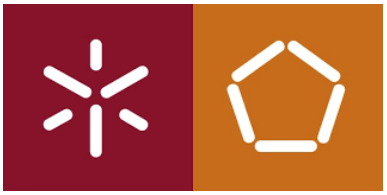


Universidade do Minho

Departamento de Informática
2020/2021

Desenvolvimento de Sistemas de Software

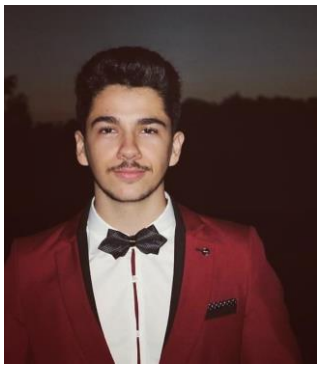
30 de Dezembro de 2021

Grupo 2

João Lourenço, A93233

Rita Gomes, A87960

Duarte Moreira, A93321



Lucas Carvalho, A93176

João Ferreira, A93263

Conteúdo

Introdução

No âmbito da unidade curricular de Desenvolvimento de Sistemas de Software foi-nos proposto a implementação de um Sistema de Gestão para Centros de Reparação de equipamentos eletrónicos. Este projeto evoluiu de forma iterativa e incremental, tendo o seu processo se baseado em duas fases, sendo que o presente relatório é o correspondente à explicação do trabalho desenvolvido na segunda fase. A primeira fase, tal como vimos no relatório anterior, consistiu na análise dos cenários apresentados, através dos quais definimos o Modelo de Domínio e os Modelos de Use Cases. Já na segunda fase, para garantir a coerência global do procedimento foi necessário fazer uma reavaliação de todo o trabalho realizado na primeira fase. Além disto, definiu-se os subsistemas e os correspondentes Diagrama de Componentes, o Diagrama de Packages, os Diagramas de Classes e os de Sequência. Numa etapa final, após termos toda a modelação necessária concretizada passamos para a implementação dos Use Cases em Java.

Ao longo do relatório vamos voltar a apresentar os modelos de domínio e de Use Cases juntamente com os novos diagramas assim como as decisões tomadas e problemas encontrados.

Descrição da abordagem utilizada

Primeiramente, fizemos uma análise detalhada do trabalho realizado na fase anterior, corrigindo situações que não tínhamos previsto ou não tínhamos dado tanta atenção.

Posteriormente, foi necessário, tendo em conta os Use Cases definidos previamente e o modelo de Domínio, o levantamento das responsabilidades do sistema e a definição dos respetivos métodos (API da Lógica de Negócio), sendo necessária a criação de vários subsistemas pelos quais estes métodos seriam distribuídos.

De seguida, passamos para a criação de um diagrama de classes, definindo atributos e classes bem como as relações entre essas mesmas classes. O facto deste diagrama ser navegável, facilitou o processo de desenvolvimento de diagramas de sequência, que por sua vez representam as trocas de mensagens e a "interação" entre os vários objetos. Foi definido um diagrama de sequência para cada método levantado para conseguirmos ter uma visão das várias funcionalidades do programa. Após isto, criámos o diagrama de componentes no qual foram representados os vários subsistemas que o grupo considerou apropriados. Por último, desenvolvemos o diagrama de Packages. Todos estes diagramas foram desenvolvidos usando a ferramenta *Visual Paradigm*.

Finalmente, após todas as análises e decisões efetuadas passamos para a implementação em Java dos mesmos Use Cases.

Modelo de Domínio

Com o desenvolver do trabalho, e dada a natureza incremental adotada, tivemos de atualizar o modelo de domínio inicialmente feito, devido ao facto de este apresentar algumas inconsistências.

Este novo modelo de domínio encontra-se mais simplificado, optamos por retirar todos os tipos de registos (exemplos: registo de pagamento, registo de pedido de orçamento) e decidimos colocar apenar as principais listas (“lista de pedidos orçamento”, “lista de ids dos equipamentos reparados”, “lista de ids dos equipamentos por reparar”).

Para uma implementação mais simples achamos pertinente criar uma “ficha registo” por cliente, de forma a guardar toda a informação relativa ao mesmo e ao seu equipamento. Um problema a apontar aqui é a impossibilidade de um cliente ter mais que um equipamento a ser reparado simultaneamente.

Por último mudamos também a estrutura de forma a ficar mais fácil a distinção entre um serviço expresso e um serviço programado.

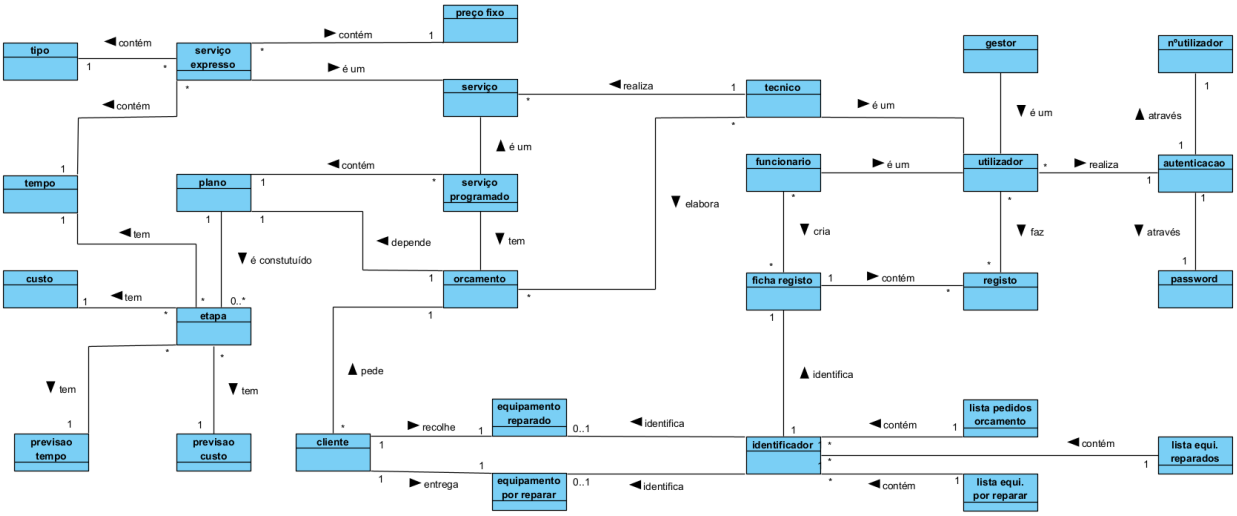


Figura1 Modelo de Domínio

API da Lógica de Negócio

Após a análise e correção dos Use Cases definidos na primeira fase do projeto (na fase anterior alguns Use Cases ficaram incorretos pois ao serem passados para tabelas foram trocados títulos), para a sua complementação foram elaboradas as seguintes tabelas, cujo papel é crucial para uma mais simples implementação uma vez que nelas já são pensados todos os métodos para implementar cada Use Case, assim como a sua ordem.

Use Cases: Utilizadores

1.Autenticar-se no sistema.

Este use case trata da autenticação dos 3 tipos de utilizadores do sistema (técnico, gestor e funcionário).

O sistema valida as credenciais inseridas pelo utilizador, se estiverem corretas o utilizador tem acesso ao sistema, caso contrário o sistema volta a pedir as credenciais.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Autenticar-se no sistema				
CENÁRIOS:		Cenário 1, 2, 3, 4 e 5				
PRÉ-CONDIÇÃO:		Sistema iniciado				
PÓS-CONDIÇÃO:		Utilizador autenticado				
FLUXO NORMAL:						
	1.	Sistema pede credenciais (numero e password)	UI			
	2.	Utilizador insere as credenciais.	UI			
	3.	Sistema valida as credenciais		valida credenciais de utilizador	validaCredenciais(numero: String, password: String): boolean	SubUtilizadores
	4.	Utilizador tem acesso ao sistema				
FLUXO ALTERNATIVO	(1)	[credenciais erradas] (passo 3)				
	3.1	Sistema informa o utilizador que as credenciais não são válidas	UI			
	3.4	regressa ao passo 1				

Figura2 Use Case: Autenticação

Use Cases: Funcionários

1.Registar entrega do equipamento pelo cliente.

Este use case trata de registar a entrega do equipamento na loja, pelo cliente. Numa primeira fase, o técnico fornece o NIF do cliente ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Registar entrega do equipamento pelo cliente				
CENÁRIOS:		Cenário 1				
PRÉ-CONDIÇÃO:		Funcionário autenticado				
PÓS-CONDIÇÃO:		Ficha de registos de equipamento criada				
FLUXO NORMAL:						
	1.	Funcionário fornece dados de registo do cliente	UI			
	2.	Sistema cria ficha de resgisto		criar ficha do cliente	criaFicha(id:Identificador, tlm:String, email:String, nome:String,descr	SubRegistos
	3.	Sistema contabiliza receção do equipamento pelo funcionário		incrementar as recceções do funcionário	incrementaRececao(funcionario: Funcionario)	SubUtilizadores
FLUXO DE EXCEÇÃO	(2)	[serviço Expresso não disponivel] (passo 2)				
	2.1	Sistema informa o utilizador que não é possivel realizar o serviço	UI			
	2.2	Sistema cancela registo da entrega		cancelar registo da entrega		

Figura3 Use Case: Registar receção do equipamento

2.Registar pedido de orçamento.

Este use case trata de registar o pedido de orçamento de uma reparação.

Numa primeira fase, o técnico fornece o NIF do cliente ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

De seguida, o sistema anota o pedido de orçamento e adiciona o id á lista de pedidos de orçamento.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agr métod
USE CASE:		Registar pedido orçamento				
CENÁRIOS:		Cenário 1				
PRÉ-CONDIÇÃO:		Funcionário autenticado				
PÓS-CONDIÇÃO:		Registo do pedido de orçamento anotado				
FLUXO NORMAL:						
	1.	Funcionário fornece nif do cliente ao sistema	UI			
	2.	Sistema valida nif		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema anota pedido de orçamento		anotar pedido de orçamento como "true"	anotaPedidoOrcamento(id:Identificador)	SubRegistos
	4.	Sistema adiciona id à lista de pedidos de orçamento		adicionar id à lista de pedidos de orçamento	addPedidoOrcamento(id:Identificador)	SubRegistos
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	sistema nega a existência de uma ficha de registo	UI			
	2.2	sistema cancela registo do pedido		cancelar registo do pedido		

Figura4 Use Case: Registar pedido orçamento

3.Confirmação pedido de orçamento

Este use case trata de verificar a resposta do cliente em relação ao orçamento realizado. Numa primeira fase, o técnico fornece o NIF do cliente ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

Caso contrário, o sistema anota a resposta do cliente e remove o id da lista de pedidos de orçamento. Caso seja negativa, o sistema anota a reparação como não efetuada, adiciona o id do equipamento à lista de equipamentos reparadas e altera o estado do pagamento para que este possa ser levantado (assume-se que caso o aparelho não tenha reparação, não cobrado nenhum custo). Caso seja positiva, o id é adicionado á lista de equipamentos por reparar e removido da lista de pedidos de orçamento.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agru métod
USE CASE:		Confirmação pedido orçamento				
CENÁRIOS:		Cenário 1				
PRÉ-CONDIÇÃO:		Funcionário autenticado				
PÓS-CONDIÇÃO:		Confirmação anotada no registo e eliminação do NIF da lista de pedidos de orçamento				
FLUXO NORMAL:						
	1.	Funcionário fornece resposta e nif do cliente ao sistema	UI			
	2.	Sistema valida nif		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema anota confirmaçao orçamento		anotar confirmação do orcamento como "true"	anotaConfirmacaoOrcamento(id:Identificador, resposta: boo	SubRegistos
	4.	Sistema adiciona id à lista de equipamentos por reparar		adicionar id à lista de equipamentos por reparar	addEquipamentoPorReparar(id:Identificador)	SubRegistos
	5.	Sistema elimina id da lista de pedidos de orçamento		eliminar id da lista de pedidos orcamento	eliminaPedidoOrcamento(id:Identificador)	SubRegistos
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	Sistema nega a existência de uma ficha de registo	UI			
	2.2	Sistema cancela registo do pedido		cancelar registo do pedido		
FLUXO ALTERNATIVO	(2)	[resposta negativa] (passo 3)				
	3.1	Sistema anota recusa do orçamento		anotar confirmação do orcamento como "false"	anotaConfirmacaoOrcamento(id:Identificador, resposta: boo	SubRegistos
	3.2	Sistema anota recusa da reparação		anotar reparação como "não efetuada"	anotaReparacao(id:Identificador, reparacao: String)	SubRegistos
	3.3	Sistema elimina id da lista de pedidos de orçamento		eliminar id da lista de pedios orcamento	eliminaPedidoOrcamento(id:Identificador)	SubRegistos
	3.4	Sistema anota pagamento		anotar o pagamento como "true" uma vez que o equipamento não tem reparação	anotaPagamento(id:Identificador)	SubRegistos
	3.5	Sistema adicona id à lista de equipamentos reparados		adicionar id à lista de equipamentos reparados	addEquipamentoReparado(id:Identificador)	SubRegistos

Figura5 Use Case: Confirmação pedido orçamento

4.Registar pagamento.

Este use case trata de registar o pagamento da reparação de um equipamento.

Primeiramente, o técnico fornece o NIF do cliente ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

Caso contrário o pagamento é anotado como feito

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agr método
USE CASE:		Registar pagamento				
CENÁRIOS:		Cenário 1				
PRÉ-CONDIÇÃO:		Funcionário autenticado				
PÓS-CONDIÇÃO:		Pagamento anotado no registo				
FLUXO NORMAL:						
	1.	Funcionário fornece nif do cliente ao sistema	UI			
	2.	Sistema valida nif		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema anota pagamento		anotar pagamento como "true"	anotaPagamento(id:Identificador)	SubRegistos
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	Sistema nega a existência de uma ficha de registo	UI			
	2.2	Sistema cancela registo de pagamento		cancelar registo pagamento		

Figura6 Use Case: Registar pagamento

5.Registar entrega do equipamento pelo funcionário.

Este use case trata de registar a entrega de um equipamento ao cliente pelo funcionário. Primeiramente, o técnico fornece o NIF do cliente ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

Caso contrário, o sistema verifica se o pagamento já foi efetuado, se o cliente não tiver efetuado o pagamento o processo é cancelado. Se o pagamento já foi concluído, é anotada a entrega do equipamento e o id do mesmo é removido da lista de equipamentos reparados.

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsis (agrupar méto
USE CASE:		Registar entrega do equipamento pelo funcionário				
CENÁRIOS:		Cenário 1				
PRÉ-CONDIÇÃO:		Funcionário autenticado				
PÓS-CONDIÇÃO:		Registo removido da lista de equipamentos reparados				
FLUXO NORMAL:						
	1.	Funcionário fornece nif do cliente	UI			
	2.	Sistema valida nif		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema confirma pagamento		verificar se o pagamento foi realizado	verificaPagamento(id:Identificador):boolean	SubRegistos
	4.	Sistema anota entrega		anotar entrega como "true" e funcionario responsavel	anotaEntrega(id:Identificador,funcionario:Funcionario)	SubRegistos
	5.	Sistema contabiliza entrega do equipamento pelo funcionário		incrementar as entregas do funcionário	incrementaEntrega(funcionario:Funcionario)	SubUtilizadores
	6.	Sistema elimina id da lista de equipamentos reparados		eliminar id da lista de equipamentos reparados	eliminaEquipamentoReparado(id:Identificador)	SubRegistos
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	Sistema nega a existência de uma ficha de registo	UI			
	2.2	Sistema cancela registo de entrega do equipamento		cancelar registo de entrega		
FLUXO DE EXCEÇÃO	(2)	[pagamento não realizado] (passo 3)				
	3.1	Sistema informa o utilizador que o pagamento não foi efetuado	UI			
	3.2	Sistema cancela registo de entrega do equipamento		cancelar registo de entrega		

Figura7 Use Case: Registar entrega do equipamento

Use Cases: Técnicos

1.Aceder a lista de pedidos de orçamento.

Este use case trata de apresentar ao técnico a lista de pedidos de orçamento.

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas / métodos
USE CASE:		Aceder à lista de pedidos de orçamento				
CENÁRIOS:		Cenário 3				
PRÉ-CONDIÇÃO:		Técnico autenticado				
PÓS-CONDIÇÃO:		Acesso à lista de pedidos de orçamento				
FLUXO NORMAL:						
	1.	Técnico pede ao sistema a lista dos pedidos de orçamento	UI			
	2.	Sistema lista os IDs dos equipamentos	UI			

Figura8 Use Case: Aceder à lista de pedidos de orçamento

2.Registar plano de trabalhos

Este use case trata de registar o plano de trabalhos para a reparação de um equipamento. Numa primeira fase, o técnico fornece o NIF do cliente e o plano de trabalhos ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

Caso contrário, é anotado o plano de trabalhos e com base no mesmo é calculado e anotado o orçamento que posteriormente é enviado ao cliente.

Se o equipamento não tiver reparação é anotado o plano de trabalhos e anotado que não é possível reparar de seguida é notificado o cliente.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas / métodos
USE CASE:		Registar plano de trabalhos para a reparação				
CENÁRIOS:		Cenário 3				
PRÉ-CONDIÇÃO:		Técnico autenticado				
PÓS-CONDIÇÃO:		Plano de trabalhos registado				
FLUXO NORMAL:						
	1.	Técnico fornece nif e plano de trabalhos ao sistema	UI			
	2.	Sistema valida nif		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema anota plano de trabalhos		anotar plano recebido	anotaPlano(id:Identificador, plano:List<Etapa>)	SubRegistos
	4.	Sistema calcula orçamento		calcular o orçamento com base no plano	calculaOrcamento(plano:List<Etapa>):Orcamento	SubRegistos
	5.	Sistema anota orçamento		anotar orçamento calculado na ficha	anotaOrcamento(id:Identificador, orcamento:Orcamento)	SubRegistos
	6.	Sistema notifica cliente				
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	Sistema nega a existência de uma ficha de registo	UI			
	2.2	Sistema cancela registo do serviço Expresso		cancelar registo do serviço		
FLUXO DE EXCEÇÃO	(3)	[não tem reparação] (passo 3)				
	3.1	Sistema anota plano de trabalhos		anotar plano recebido	anotaPlano(id:Identificador, plano:List<Etapa>)	SubRegistos
	3.2	Sistema anota reparação		anotar reparação como "não efetuada"	anotaReparacao(id:Identificador, reparacao: String)	SubRegistos
	3.3	Sistema elimina id da lista de equipamentos por reparar		eliminar id da lista de equipamentos por reparar	eliminaEquipamentoPorReparar(id:Identificador)	SubRegistos
	3.4	Sistema adiciona id à lista de equipamentos reparados		adicionar id à lista de equipamentos reparados	addEquipamentoReparado(id:Identificador)	SubRegistos
	3.5	Sistema notifica cliente				

Figura9 Use Case: Registar plano de trabalhos

3.Aceder lista de equipamentos por reparar.

Este use case trata de apresentar ao técnico a lista equipamentos por reparar.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas / métodos
USE CASE:		Aceder à lista de equipamentos por reparar				
CENÁRIOS:		Cenário 4				
PRÉ-CONDIÇÃO:		Técnico autenticado				
PÓS-CONDIÇÃO:		Acesso à lista de equipamentos por reparar				
FLUXO NORMAL:						
	1.	Técnico pede ao sistema a lista dos equipamentos por reparar	UI			
	2.	Sistema lista os IDs dos equipamentos	UI			

Figura10 Use Case: Aceder à lista de equipamentos por reparar

4.Registar conclusão da reparação.

Este use case trata de registar a conclusão da reparação de um equipamento. Numa primeira fase, o técnico fornece o NIF do cliente e o plano de trabalhos ao sistema, que por sua vez verifica se existe uma ficha de registo associada ao mesmo. Caso não exista, entramos num fluxo de exceção e o sistema encerra o pedido.

Caso contrário, o sistema contabiliza os serviços realizados pelo técnico, altera o estado da reparação para “efetuada”, elimina o id da lista de equipamentos por reparar, adiciona-o á lista de equipamentos reparados e informa o cliente da reparação concluída.

			1. Dividir os fluxos	2. identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Registar conclusão de reparação				
CENÁRIOS:		Cenário 2				
PRÉ-CONDIÇÃO:		Técnico autenticado				
PÓS-CONDIÇÃO:		Conslusão de reparação registada e cliente informado				
FLUXO NORMAL:						
	1.	Técnico fornece ao sistema um nif	UI			
	2.	Sistema verifica se o nif fornecido tem correspondência a uma ficha de registos existente		verificar se existe ficha de registo associada	validaNIF(id:Identificador):boolean	SubRegistos
	3.	Sistema contabiliza conclusão de reparação do equipamento pelo técnico		incrementar serviços realiazados pelo técnico	incrementaServicoProgramado(ou Expresso)(tecnico:Tecnico)	SubUtilizadores
	4.	Sistema altera o estado da reparação		alterar o estado de reparação para "efetuada"	anotaReparacao(id:Identificador, reparacao: String)	SubRegistos
	5.	Sistema elimina id da lista de equipamentos por reparar		eliminar id da lista de equipamentos reparados	eliminaEquipamentoReparado(id:Identificador)	SubRegistos
	6.	Sistema adiciona id à lista de equipamentos reparados		adicionar o id à lista de equipamentos reparados	addEquipamentoReparado(id:Identificador)	SubRegistos
	7.	Sistema informa cliente (reparação do equipamento concluída)		informar cliente da reparação concluida		
FLUXO DE EXCEÇÃO	(1)	[ficha de registo inexistente] (passo 2)				
	2.1	Sistema nega a existência de uma ficha de registo	UI			
	2.2	Sistema cancela registo de reparação		cancelar registo pagamento		

Figura11 Use Case: Registar conclusão de reparação

Use Cases: Gestor

1.Consultar lista dos técnicos de reparações

Este use case trata de apresentar ao gestor a lista dos técnicos de reparações.

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Consultar lista dos tecnicos de reparações				
CENÁRIOS:		Cenário 5				
PRÉ-CONDIÇÃO:		Gestor autenticado				
PÓS-CONDIÇÃO:		Listagem do pedido apresentada				
FLUXO NORMAL:						
	1.	Gestor pede ao sistema a listagem relativa aos técnicos de reparações	UI			
	2.	Sistema apresenta a lista		Calcular a lista relativa aos técnicos	calcularListaTecnicos()	SubUtilizadores

Figura12 Use Case: Consultar lista dos técnicos

2.Consultar lista dos funcionários

Este use case trata de apresentar ao gestor a lista dos funcionários.

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Consultar lista dos funcionarios				
CENÁRIOS:		Cenário 5				
PRÉ-CONDIÇÃO:		Gestor autenticado				
PÓS-CONDIÇÃO:		Listagem do pedido apresentada				
FLUXO NORMAL:						
	1.	Gestor pede ao sistema a listagem relativa aos funcionários	UI			
	2.	Sistema apresenta a lista		Calcular a lista relativa aos funcionários	calcularListaFuncionarios()	SubUtilizadores

Figura13 Use Case: Consultar lista dos funcionários

3.Consultar lista exaustiva dos técnicos

Este use case trata de apresentar ao gestor a lista exaustiva dos técnicos.

			1. Dividir os fluxos	2. Identificar responsabilidades da LN	3. definir API (identificar métodos)	4. identificar subsistemas (agrupar métodos)
USE CASE:		Consultar lista exaustiva dos tecnicos				
CENÁRIOS:		Cenário 5				
PRÉ-CONDIÇÃO:		Gestor autenticado				
PÓS-CONDIÇÃO:		Listagem do pedido apresentada				
FLUXO NORMAL:						
	1.	Gestor pede ao sistema a listagem exaustiva relativa aos técnicos de reparações	UI			
	2.	Sistema apresenta a lista		Calcular a lista exaustiva dos tecnicos	calcularListaExaustivaTecnicos()	SubUtilizadores

Figura14 Use Case: Consultar lista exaustiva dos técnicos

Diagrama de Componentes

A criação de subsistemas permite uma maior organização, uma vez que haverá o agrupamento dos métodos pelos subsistemas onde melhor se adequam. Além disto, esta prática é essencial na questão do encapsulamento uma vez que cada subsistema implementa uma interface com os métodos que lhes foram atribuídos.

Sendo assim, para os vários métodos de use cases considerados na lógica de negócio, conseguimos identificar os seguintes subsistemas: **SubUtilizadores** e **SubRegistos**.

Com base nisto, construímos o seguinte diagrama de componentes:

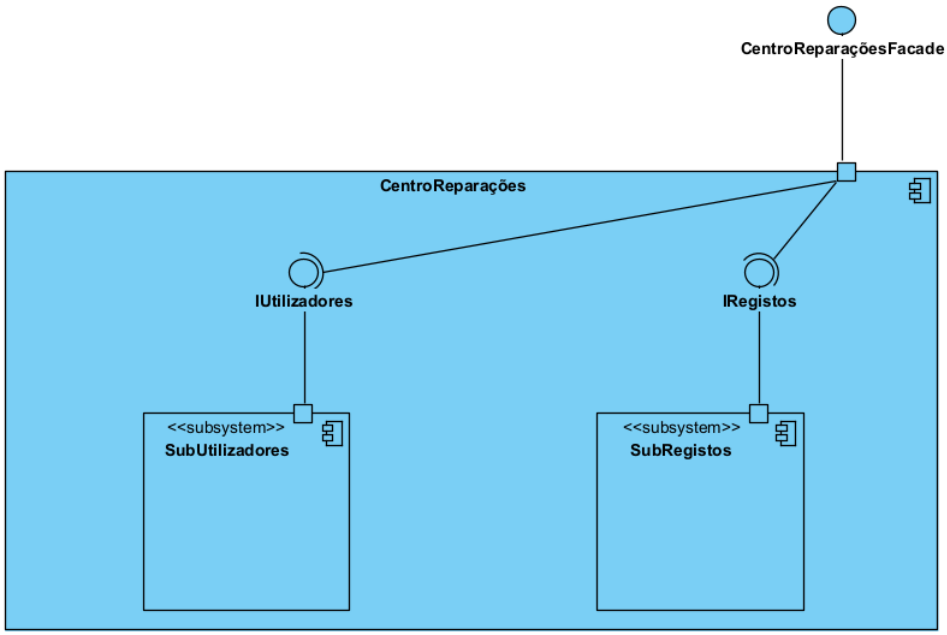


Figura15 Diagrama de componentes

Diagramas de Sequência

Os diagramas de sequência servem para representar em qual ordem, um grupo de objetos trabalha em conjunto. Estes diagramas são usados por desenvolvedores de software e profissionais de negócios para entender as necessidades de um novo sistema ou para documentar um processo existente.

Depois de definirmos os métodos necessários para todos e a sua ordem optamos por realizar um diagrama de sequência para cada método necessário na implementação dos Use Cases mais cruciais na realização deste projeto:

Use case: Confirmação do pedido de orçamento

Método responsável por verificar se existe alguma ficha de registo com o Identificador fornecido.

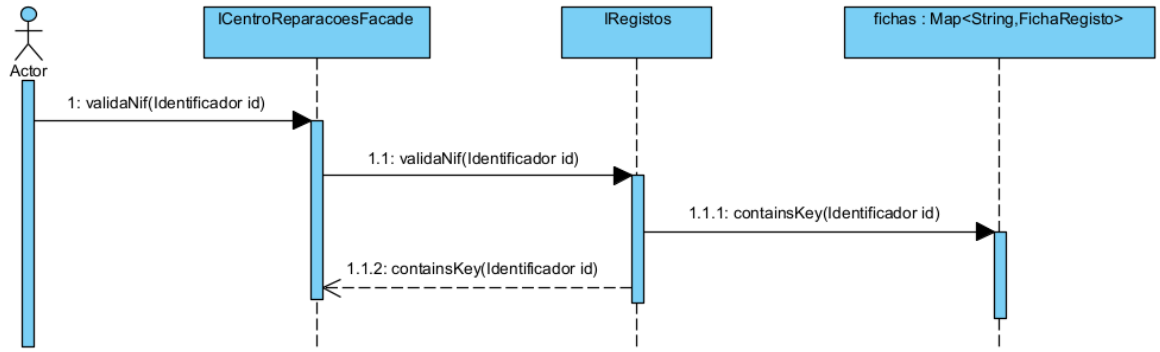


Figura16 Diagrama de sequência: validaNif

Método responsável por anotar a resposta do cliente em relação ao pedido de orçamento.

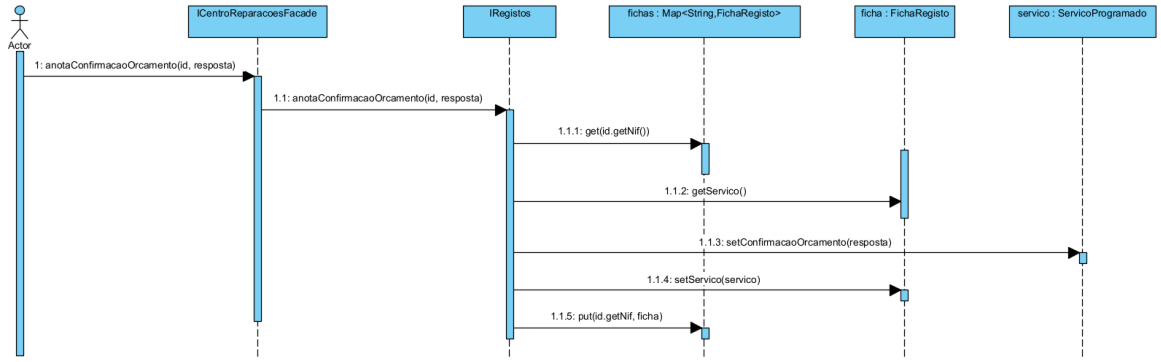


Figura17 Diagrama de sequência: anotaConfirmacao

Método responsável por adicionar um Identificador à lista de equipamentos por reparar.

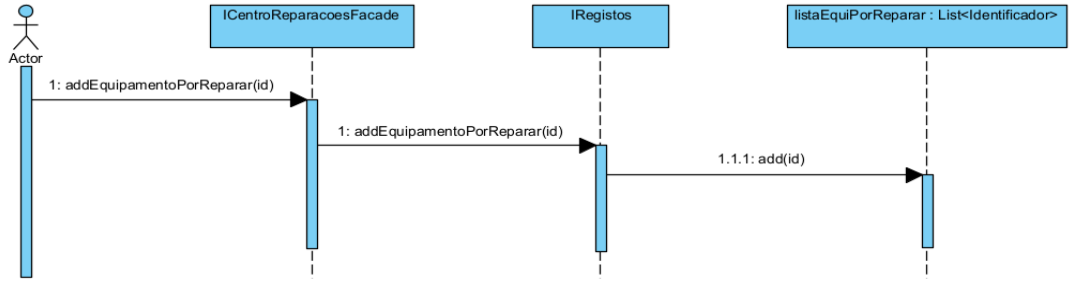


Figura18 Diagrama de sequência: addEquipamentoPorReparar

Método responsável por eliminar um Identificador da lista de pedidos de orçamento.

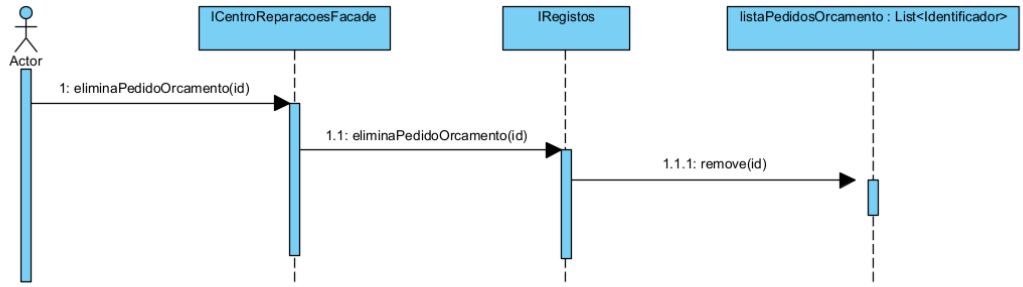


Figura19 Diagrama de Sequência: eliminaPedidoOrcamento

Método responsável por anotar o estado da reparação.

efetuada: reparação concluída.

não efetuada: reparação não concluída (cliente rejeita orçamento ou não tem reparação).

em espera: reparação em espera (falta de peças ou necessita de nova confirmação do cliente).

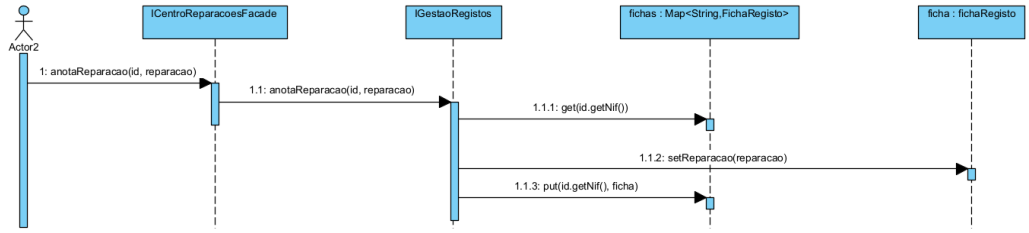


Figura20 Diagrama de Sequência: anotaReparacao

Método responsável por anotar o pagamento do serviço realizado.

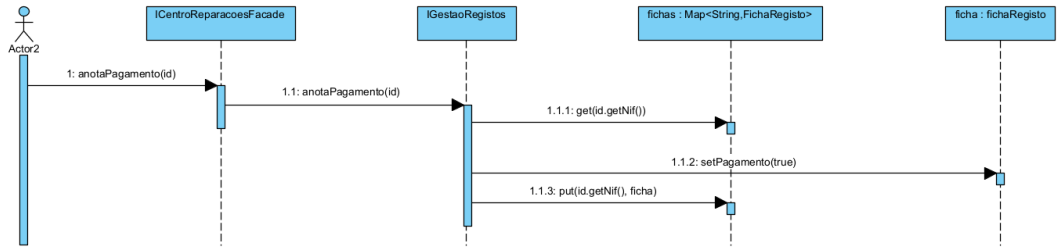


Figura21 Diagrama Sequência: anotaPagamento

Método responsável por adicionar um Identificador à lista de equipamentos reparados.

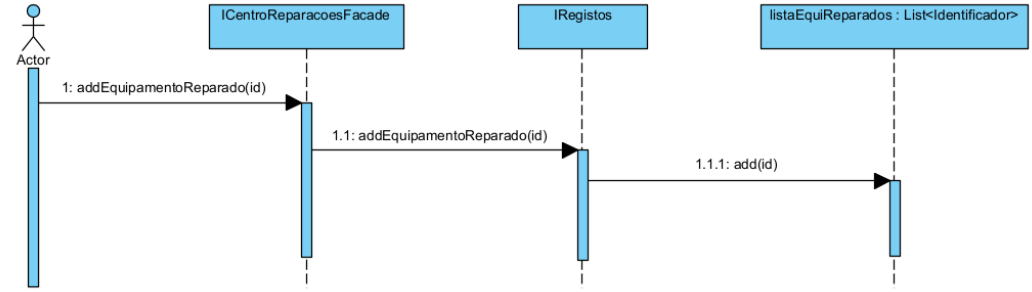


Figura22 Diagrama Sequência: addEquipamentoReparado

Use Case: Realizar Reparação

Método responsável por fazer o registo do contacto realizado com o cliente.

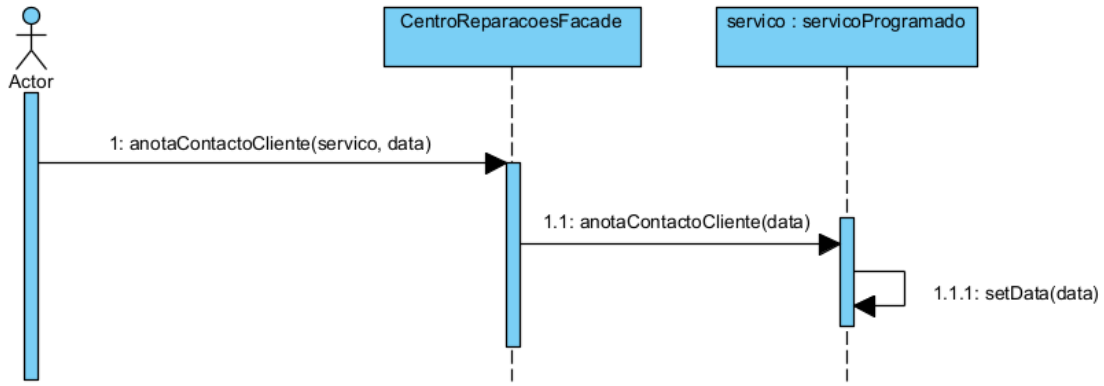


Figura23 Diagrama Sequência: anotaContactoCliente

Método responsável por receber e registar o custo e tempo real de cada Etapa de um plano de trabalhos de um determinado Serviço. O técnico é quem fornece estes dados.

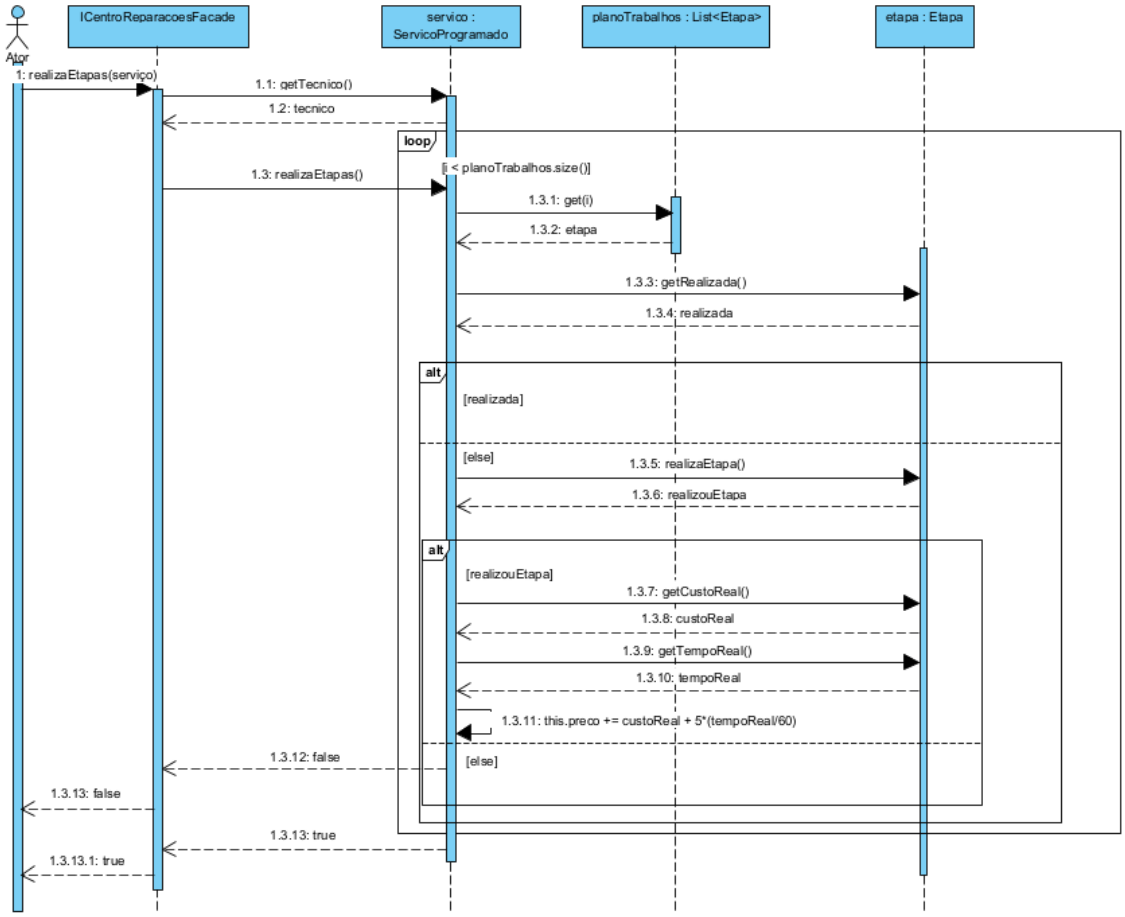


Figura24 Diagrama Sequência: realizaEtapas

Use Case - Registo do plano de trabalhos

Método responsável por anotar o plano recebido.

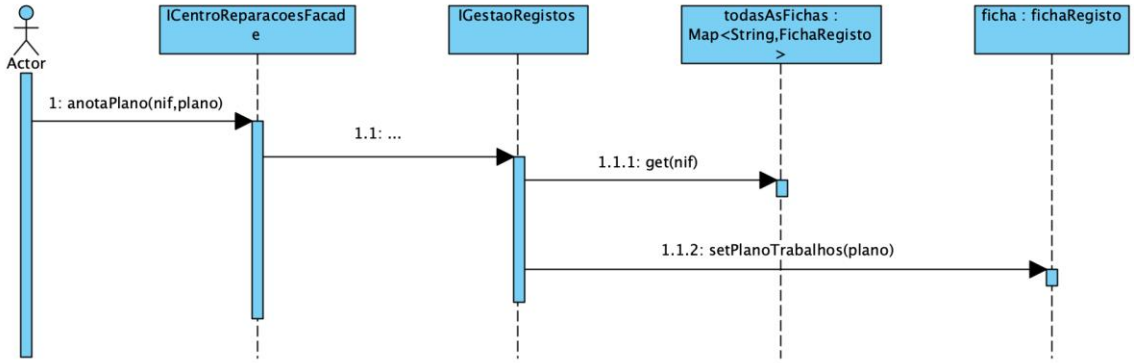


Figura25 Diagrama Sequência: anotaPlano

Método responsável por calcular o orçamento com base no plano:

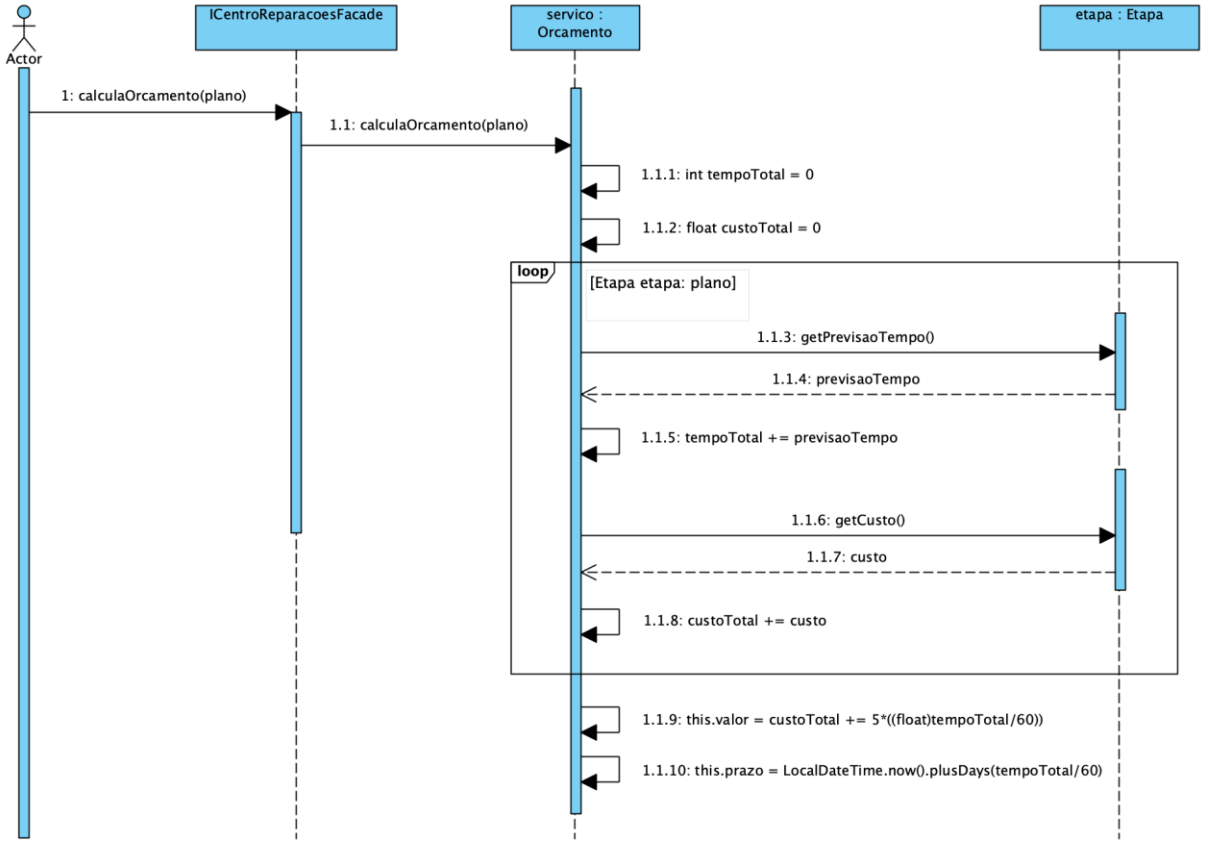


Figura26 Diagrama Sequência: calculaOrcamento

Método responsável por anotar o orçamento calculado na ficha.

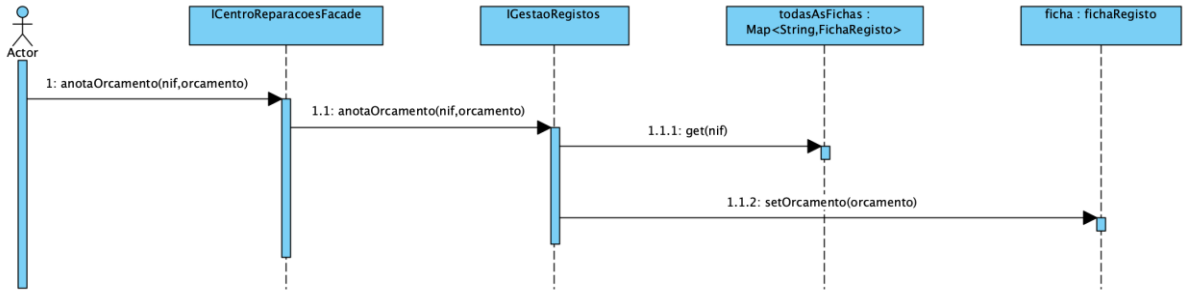


Figura27 Diagrama Sequência: anotaOrcamento

Método responsável por eliminar um identificador da lista de equipamentos por reparar.

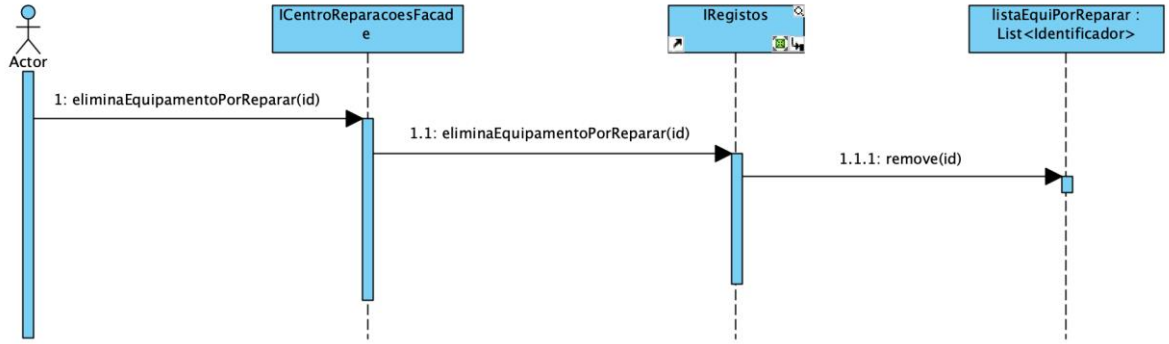


Figura28 Diagrama Sequência: eliminaEquipamentoPorReparar

Diagrama de Classes

Com base no modelo de domínio foram selecionadas as componentes que dariam origem a classes, como por exemplo, FichaRegisto, Servico, ServicoExpresso, ServicoProgramado, Utilizador, Funcionario, Tecnico e Gestor.

Após a análise de todos os diagramas de sequência desenvolvidos, bem como do diagrama de componentes onde estão definidos os subsistemas principais da modelação do nosso projeto, prosseguimos à realização do diagrama de classes, onde descrevemos as classes existentes, modeladas com os respetivos atributos e operações.

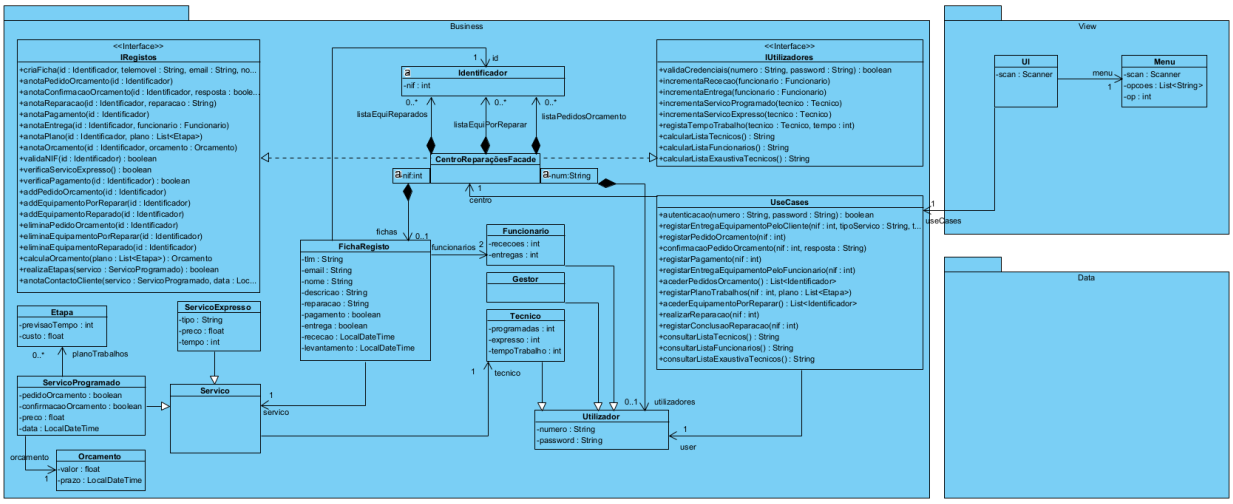


Figura29 Diagrama de Classes

Diagrama de Packages

Tendo em conta a complexidade de todo este projeto, acabou por ser necessário desenvolver um diagrama de Packages que acabasse por organizar todas as classes que já se encontram planeadas, definindo o seu papel no sistema em si de acordo com as dependências entre cada uma delas. Desta forma, para termos uma interpretação mais intuitiva de todo o programa decidimos proceder à criação dos seguintes packages: 'View', 'Exceptions' e 'Business', sendo este último constituído pelos packages 'Registo', 'Serviço' e 'Utilizador'.

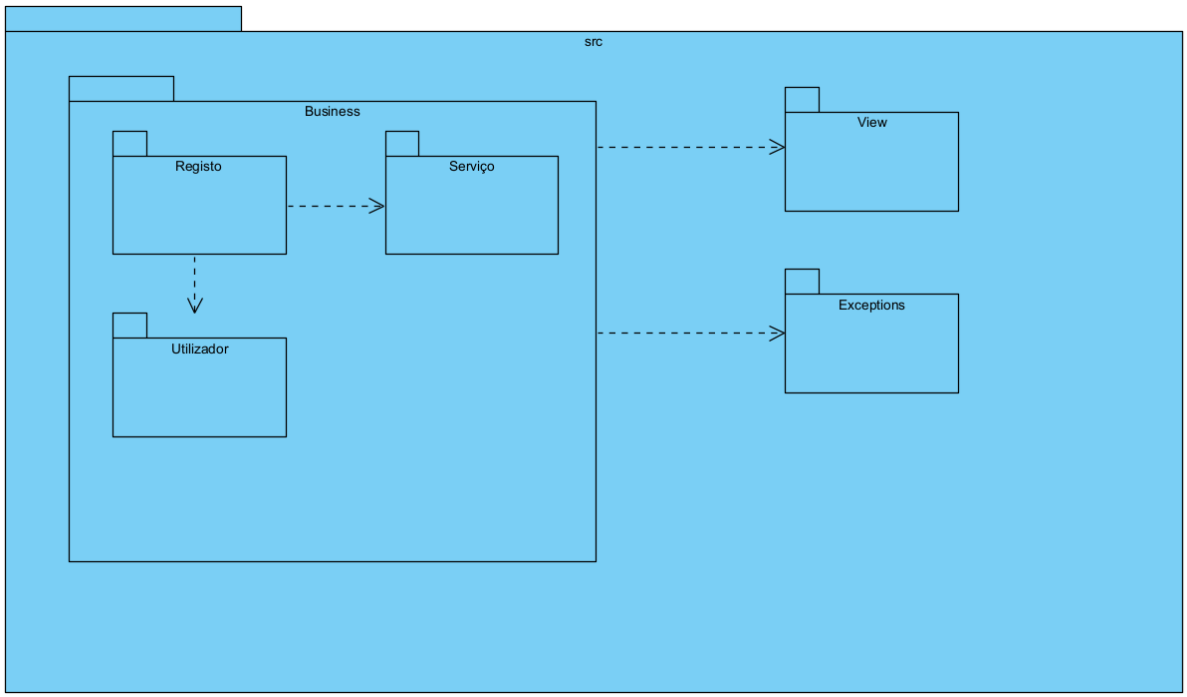


Figura30 Diagrama de Packages

Implementação

Depois de acabado todo o processo de modelação do projeto passou-se à implementação deste na linguagem de programação Java, utilizando o IDE IntelliJ. Para controlo de versões foi usado o git.

Baseamo-nos nos modelos desenvolvidos nas etapas anteriores para o desenvolvimento do código.

1. Business
    a. Registo
        i. Ficha Registo: Nesta classe é definido a ficha de registo para cada serviço. Conta com a identidade dos funcionários, todos os dados referentes ao equipamento, bem como a informação atual do serviço.
        ii. Identificador: Nesta classe define-se o identificador de cada cliente.
    b. Servico
        i. Etapa: Contém a caraterização de cada etapa do plano de trabalhos.
        ii. Orcamento: Define o valor e o tempo estimado para o serviço.
        iii. Servico: Classe abstrata, é a superclasse dos serviços.
        iv. ServicoExpresso: Carateriza um serviço expresso – define o tipo, o tempo e o valor.
        v. ServicoProgramado: Mais complexo que o ServicoExpresso, define o plano de trabalhos (uma lista de etapas), o Orçamento, a data e o preço final.
    c. Utilizador
        i. Utilizador: Classe abstrata, é a superclasse de todos os utilizadores da aplicação. Contém apenas duas variáveis, as credenciais de início de sessão.
        ii. Funcionario: Esta classe serve para registar o número de receções e o número de entregas feitas por cada funcionário.
        iii. Gestor: Classe do gestor.
        iv. Tecnico: Esta classe serve para registar o número de reparações expresso e agendadas realizadas, bem como o tempo total das reparações.
2. Exceptions: Package que contém todas as exceptions do programa.
3. View: Package que contém as interações com o utilizador.

Considerações finais

A última fase deste projeto consistiu essencialmente na aplicação das metodologias lecionadas nas aulas teóricas de Desenvolvimento de Sistemas de Software. Isto permitiu, não só consolidar todo o conhecimento adquirido, bem como permitiu ao grupo ter uma nova perspetiva de programação, não começando logo a criar código puro, mas sim a planear o

trabalho faseadamente, permitindo-nos ter uma visão mais geral e concreta. O grupo considera que o desenvolvimento faseado do projeto se mostrou bastante útil, na medida em que a definição dos use cases, dos subsistemas, dos diagramas de classes e outros, permite desenvolver progressivamente uma solução, sendo esta modificada sempre que necessário.

Durante a elaboração desta última fase do projeto deparamo-nos com algumas adversidades. Inicialmente, notámos algumas falhas no nosso Modelo de Domínio que tiveram de ser corrigidas. À parte disto, tudo o que foi mudado da primeira fase serviu apenas aumentar o grau de detalhe do trabalho e adicionar funcionalidades que achamos que seriam úteis.

A maior dificuldade residiu na construção do sistema de gestão de dados, sendo que o grupo não tem muitos conhecimentos relacionados com Base de Dados. Contudo, tentamos implementar um sistema de gestão de dados que guarda os dados em memória.

Deste modo, o grupo considera que fez um bom trabalho, uma vez que teve o cuidado de constantemente questionar se o projeto era coerente e realista. O grupo conclui ainda que o planeamento e modelação de um sistema de software é tão ou mais importante que a implementação concreta com a escrita do código, uma vez que dá uma maior possibilidade de organização e planeamento antes do desenvolvimento concreto da aplicação.