

# Relatório Trabalho Prático - Desenvolvimento de Sistemas de Software (2ª Fase)

Grupo 35:

António Luís de Macedo Fernandes (a93312)

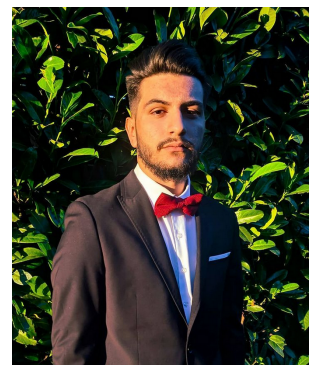
José Diogo Martins Vieira (a93251)

João Silva Torres (a93231)

João Paulo Sousa Mendes (a93256)

André Filipe Novais Vaz (a93221)

December 30, 2021



# Contents

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Alterações - 1ª Fase</b>	<b>4</b>
2.1	Modelo de Dominio . . . . .	4
2.2	Diagrama de Use Case e as suas Especificações . . . . .	4
<b>3</b>	<b>Modelação Conceptual e Implementação - 2ª Fase</b>	<b>6</b>
3.1	Diagrama de Componentes . . . . .	6
3.2	Diagramas de Sequência . . . . .	11
3.3	Diagrama de Classe . . . . .	13
3.4	Implementação . . . . .	13
3.4.1	emailHandler . . . . .	13
3.4.2	exceptions . . . . .	14
3.4.3	gui . . . . .	14
3.4.4	pedidos . . . . .	14
3.4.5	reparacoes . . . . .	17
3.4.6	trabalhadores . . . . .	19
3.5	sgrc . . . . .	20
3.5.1	SGRC . . . . .	20
3.5.2	utils . . . . .	20
<b>4</b>	<b>Main</b>	<b>20</b>
<b>5</b>	<b>Interface</b>	<b>21</b>
<b>6</b>	<b>Conclusão</b>	<b>28</b>

# 1 Introdução

No âmbito do desenvolvimento da segunda fase do projeto da unidade curricular Desenvolvimento de Sistemas de Software foi-nos proposto dar continuidade ao desenvolvimento do Sistema de Gestão para Centros de Reparação de equipamentos eletrónicos, disponibilizado na primeira fase.

Ao nível da projeção e implementação do projeto, primeiramente foi feita a API para todos os use cases e agrupamo-los nos subsistemas correspondentes. Existem três: reparações, pedidos e trabalhadores. Estes deram origem às interfaces. Em seguida, com base na descrição dos fluxos dos use cases, procedemos à realização do diagrama de componentes, diagrama de classes e dos diagramas de sequência, por esta ordem. Por fim e com base nos diagramas feitos, implementámos o nosso código, que irá ser abordado posteriormente.

Quanto à estruturação deste relatório, o mesmo é dividido em várias secções. Primeiro iremos abordar algumas pequenas alterações feitas nos diagramas já entregues na primeira fase. De seguida, vamos apresentar os restantes diagramas pedidos e abordar o código feito na implementação do nosso projeto.

Por fim, na última secção teremos uma conclusão acerca de todo o desenvolvimento, funcionalidade e opinião pessoal do grupo sobre o projeto.

## 2 Alterações - 1ª Fase

Em comparação com a primeira fase de entrega do trabalho, realizamos pequenas alterações nos diagramas que na altura decidimos resolver, tais como:

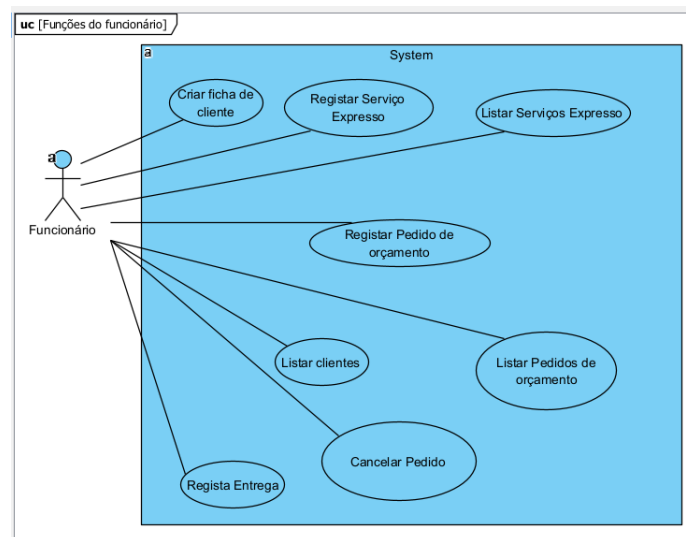
### 2.1 Modelo de Dominio

Este diagrama não sofreu nenhuma alteração, uma vez que a lógica entre as diversas classes do nosso trabalho se manteve igual conforme o que já foi anteriormente fornecido na 1ª Fase.

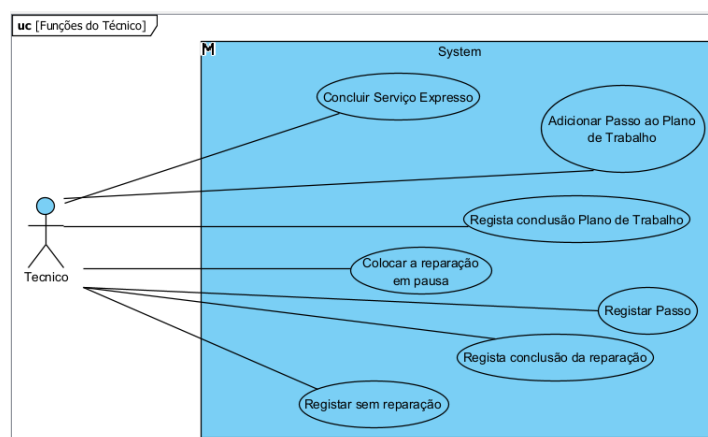
### 2.2 Diagrama de Use Case e as suas Especificações

Como dito anteriormente, completamos os fluxos dos diversos use cases já apresentados, acrescentando a definição dos métodos e subsistemas do nosso projeto.

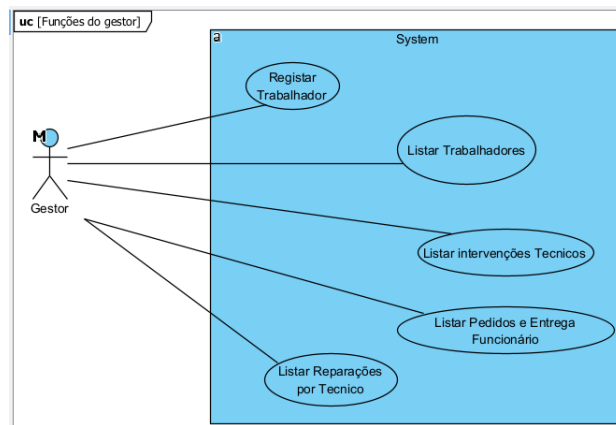
#### 1. Funcionário



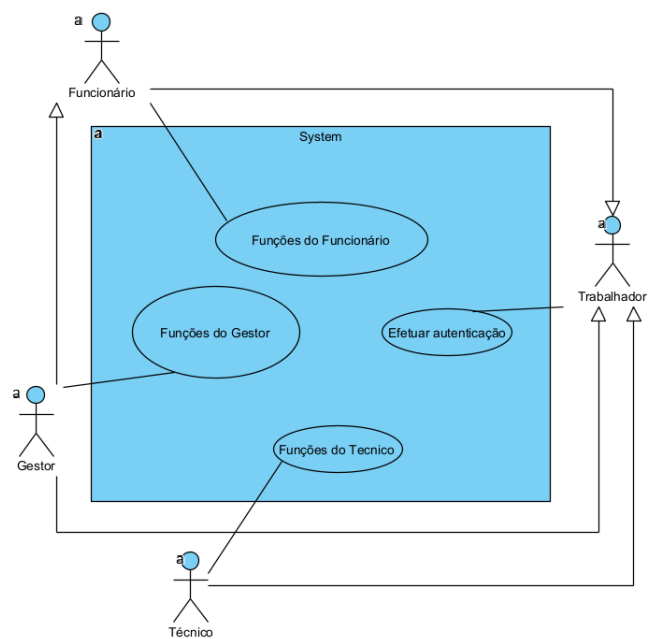
#### 2. Técnico



#### 3. Gestor



#### 4. Geral



<b>USE CASE:</b>		Criar Ficha de Cliente
<b>CENÁRIOS:</b>		/
<b>PRÉ-CONDIÇÃO:</b>		Sistema tem de estar incializado
<b>PÓS-CONDIÇÃO:</b>		Sistema fica com mais um cliente registado
<b>FLUXO NORMAL:</b>		
	1.	Funcionário fornece nome, número de utente, e-mail e número de telemóvel/telefone
	2.	Sistema cria ficha de cliente

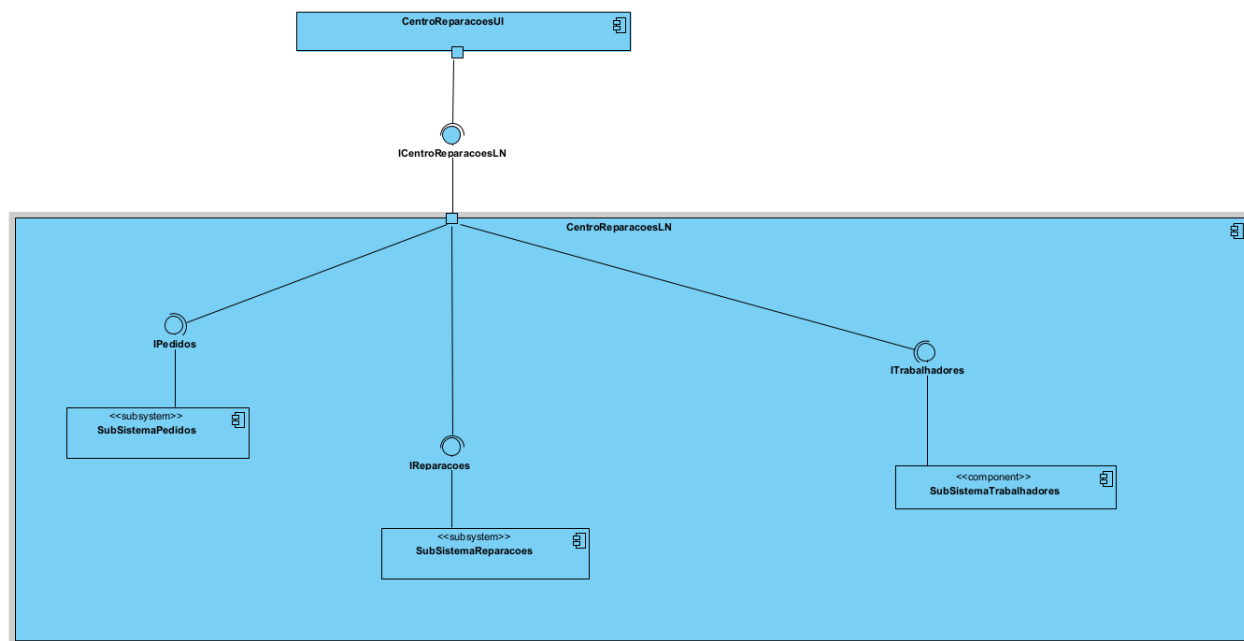
Figure 1: UC - Criar Ficha de Cliente

<b>USE CASE:</b>		Registar Serviço Expresso
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais um registo de um Serviço Expresso
<b>FLUXO NORMAL:</b>	1.	Funcionário fornece o tipo do Serviço Expresso
	2.	Funcionário fornece o identificador do Cliente
	3.	Funcionário fornece uma descrição sobre o pedido
	4.	Sistema regista Serviço Expresso

Figure 2: UC - Registar Serviço Expresso

### 3 Modelação Conceptual e Implementação - 2ªFase

#### 3.1 Diagrama de Componentes



<b>USE CASE:</b>		Listar Serviço Expresso
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema apresenta uma lista com os Serviço Expressos
<b>FLUXO NORMAL:</b>	1.	Sistema lista os Serviços Expresso registados

Figure 3: UC - Listar Serviço Expresso

<b>USE CASE:</b>		Registrar Pedido de Orcamento
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema mais um pedido de orcamento registado
<b>FLUXO NORMAL:</b>	1.	O funcionário fornece o identificador do cliente
	2.	O funcionário fornece uma descrição do pedido
	3.	O sistema regista o pedido de orçamento

Figure 4: UC - Registrar Pedido de Orçamento

<b>USE CASE:</b>		Listar Serviço Expresso
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema apresenta uma lista com os Serviço Expressos
<b>FLUXO NORMAL:</b>	1.	Sistema lista os Serviços Expresso registados

Figure 5: UC - Listar Serviços Expressos

<b>USE CASE:</b>		Registrar Pedido de Orcamento
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema mais um pedido de orcamento registado
<b>FLUXO NORMAL:</b>	1.	O funcionário fornece o identificador do cliente
	2.	O funcionário fornece uma descrição do pedido
	3.	O sistema regista o pedido de orçamento

Figure 6: UC -Registrar Pedido de Orçamento

<b>USE CASE:</b>		Listar Pedido de Orcamento
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema apresenta uma lista de pedidos de orçamento
<b>FLUXO NORMAL:</b>	1.	O sistema lista os pedidos de orçamento

Figure 7: UC - Listar Pedido de Orçamento

<b>USE CASE:</b>		Listar Clientes
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema apresenta uma lista dos clientes registados
<b>FLUXO NORMAL:</b>	1.	O sistema lista os clientes registados

Figure 8: UC - Listar Clientes

<b>USE CASE:</b>		Registrar Entrega
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais uma entrega registrada
<b>FLUXO NORMAL:</b>	1.	Funcionario fornece identificador do pedido
	2.	Sistema registra a entrega

Figure 9: UC - Registrar Entrega

<b>USE CASE:</b>		Registrar Entrega
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais uma entrega registrada
<b>FLUXO NORMAL:</b>	1.	Funcionario fornece identificador do pedido
	2.	Sistema registra a entrega

Figure 10: UC - Registrar Entrega

<b>USE CASE:</b>		Concluir Serviço Expresso
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais um serviço expresso concluído
<b>FLUXO NORMAL:</b>	1.	Sistema conclui serviço expresso

Figure 11: UC - Conclusão Serviço Expresso

<b>USE CASE:</b>		Registrar passo da Reparacao
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado, existe uma reparação a decorrer
<b>PÓS-CONDIÇÃO:</b>		A reparação fica com mais um passo registado
<b>FLUXO NORMAL:</b>	1.	Técnico fornece horas, custo de peças e descrição
	2	Sistema regista passo de plano de trabalho

Figure 12: UC - Registrar Passo de Reparação

<b>USE CASE:</b>		Registrar Conclusão do Plano de Trabalho
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado, existe um plano de trabalho a decorrer
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais um plano de trabalho concluído
<b>FLUXO NORMAL:</b>	1.	Sistema regista conclusão do plano de trabalho
	2	Sistema envia e-mail para aceitação ao cliente

Figure 13: UC - Registrar Conclusão Plano de Trabalho



<b>USE CASE:</b>		Registrar Conclusão da Reparação
PRÉ-CONDIÇÃO:		Sistema já inicializado, existe uma reparação a decorrer
PÓS-CONDIÇÃO:		O sistema fica com mais uma reparação concluído
FLUXO NORMAL:	1.	Sistema regista conclusão da reparação
	2	Sistema envia e-mail para o levantamento do equipamento

Figure 14: UC - Registrar Conclusão Plano de Trabalho

<b>USE CASE:</b>		Adicionar Passo ao plano de Trabalho
PRÉ-CONDIÇÃO:		Sistema já inicializado, existe um plano de trabalho a decorrer
PÓS-CONDIÇÃO:		O sistema fica com mais um passo no plano de trabalho
FLUXO NORMAL:	1.	Técnico fornece as horas expectáveis
	2	Técnico fornece o custo das peças expectável
	3	Técnico fornece descrição sobre o passo
FLUXO DE EXCEÇÃO	(1)	Equipamento sem reparação (passo 1)
	1.1	«include»Registrar Equipamento Sem Reparação

Figure 15: UC - Adicionar Passo ao Plano de Trabalho

<b>USE CASE:</b>		Registrar Equipamento Sem Reparação
PRÉ-CONDIÇÃO:		Sistema já inicializado, existe um plano de trabalho a decorrer
PÓS-CONDIÇÃO:		O sistema fica com mais um plano de trabalho e pedido cancelado
FLUXO NORMAL:	1.	Sistema regista equipamento sem reparação
	2	Sistema regista cancelamento do pedido e do plano de trabalho
	3	Sistema envia e-mail a informar o cliente

Figure 16: UC - Registrar Equipamentos Sem Reparação

<b>USE CASE:</b>		Colocar Reparação em Pausa
PRÉ-CONDIÇÃO:		Sistema já inicializado, existe uma reparação a decorrer
PÓS-CONDIÇÃO:		O sistema fica com mais uma reparação em pausa
FLUXO NORMAL:	1.	Sistema regista muda o estado da reparação para pausa

Figure 17: UC - Colocar Reparação em Pausa

<b>USE CASE:</b>		Cancelar Pedido
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais um pedido cancelado
<b>FLUXO NORMAL:</b>	1.	O sistema altera o estado do pedido para cancelado

Figure 18: UC - Cancelar Pedido

<b>USE CASE:</b>		Registrar Trabalhador
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema fica com mais um trabalhador registrado
<b>FLUXO NORMAL:</b>	1.	Gestor fornece o identificador do trabalhador
	2.	Gestor fornece a palavra-passe do trabalhador
	3.	Gestor fornece a confirmação da palavra-passe do trabalhador
	4.	Sistema regista trabalhador

Figure 19: UC - Registrar Trabalhador

<b>USE CASE:</b>		Listar Trabalhadores
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema lista os trabalhadores registados
<b>FLUXO NORMAL:</b>	1.	O sistema lista os trabalhadores registados

Figure 20: UC - Listar Trabalhadores

<b>USE CASE:</b>		Listar Intervenções Técnicos
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema lista as Intervenções dos Técnicos
<b>FLUXO NORMAL:</b>	1.	Gestor fornece o mês e o ano
	2.	Sistema lista as intervenções de cada técnico no mês dado

Figure 21: UC - Listar Intervenções Técnico

<b>USE CASE:</b>		Listar Pedidos e Entregas Funcionário
<b>PRÉ-CONDIÇÃO:</b>		Sistema já inicializado
<b>PÓS-CONDIÇÃO:</b>		O sistema listar pedidos e entregas dos Funcionários
<b>FLUXO NORMAL:</b>	1.	Gestor fornece o mês e o ano
	2.	Sistema lista os pedidos e entregas de cada funcionário no mês dado

Figure 22: UC - Listar Pedidos e Entregas Funcionário

<b>USE CASE:</b>		<b>Listar Reparações Técnico</b>
<b>PRÉ-CONDIÇÃO:</b>		<b>Sistema já inicializado</b>
<b>PÓS-CONDIÇÃO:</b>		<b>O sistema listar reparações dos Técnicos</b>
<b>FLUXO NORMAL:</b>	<b>1.</b>	<b>Gestor fornece o mês e o ano</b>
	<b>2.</b>	<b>Sistema lista as reparações serviço expresso e programadas de cada funcionário no mês dado</b>

Figure 23: UC - Listar Reparações Técnico

Este diagrama tem como finalidade ajudar a desenvolver o programa, assim a partir dele é possível visualizar a organização dos componentes do sistema e os relações de dependência entre eles.

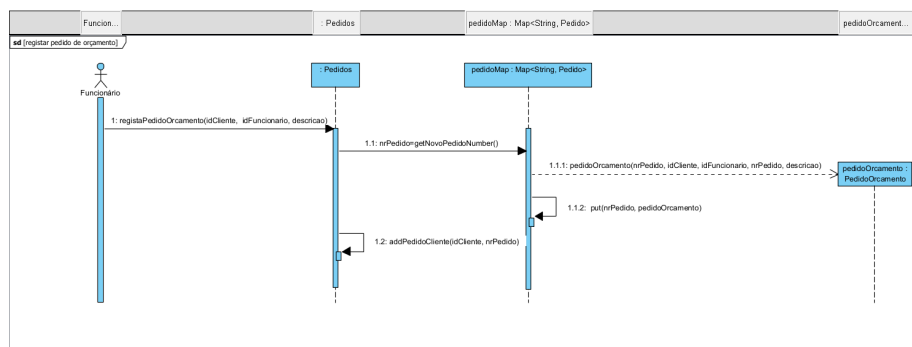
Portanto, ilustramos as nossas três interfaces, sendo estas IPedidos, IReparacoes e ITrabalhadores, que serão relativas aos pedidos, repações e trabalhadores, respetivamente. Nestas interfaces estão indicados os serviços fornecidos para cada subsistema.

### 3.2 Diagramas de Sequência

Aqui podemos visualizar alguns diagramas de certo métodos como:

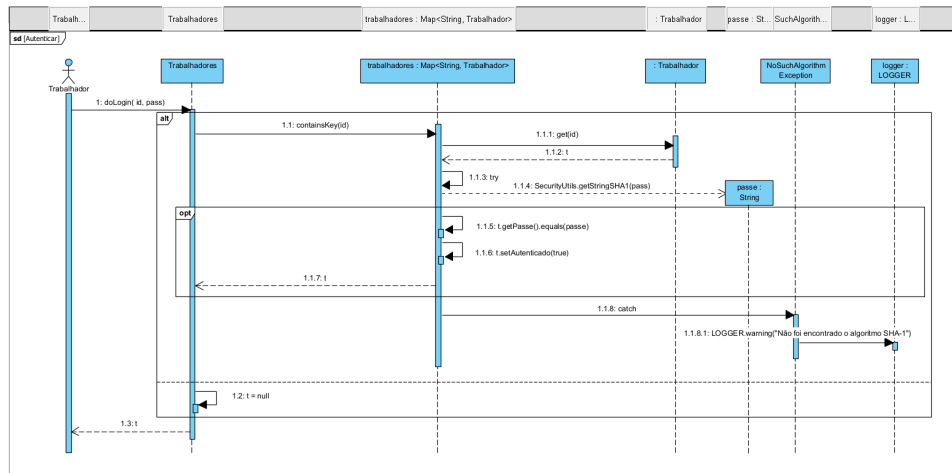
1. void registaPedidoOrçamento(String idCliente, String idFuncionario, String descricao);

Este método está relacionado com o caso de uso de um funcionário registar um pedido de orçamento.



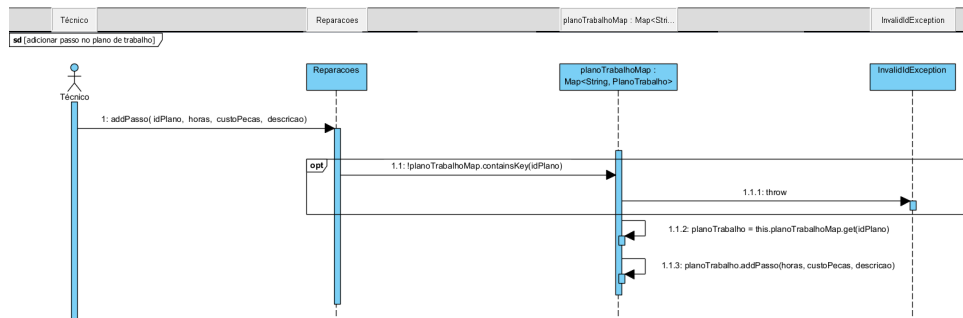
2. Trabalhador doLogin(String id, String pass);

Este método está relacionado com o caso de uso de um técnico efetuar um passo a ser realizado numa reparação.



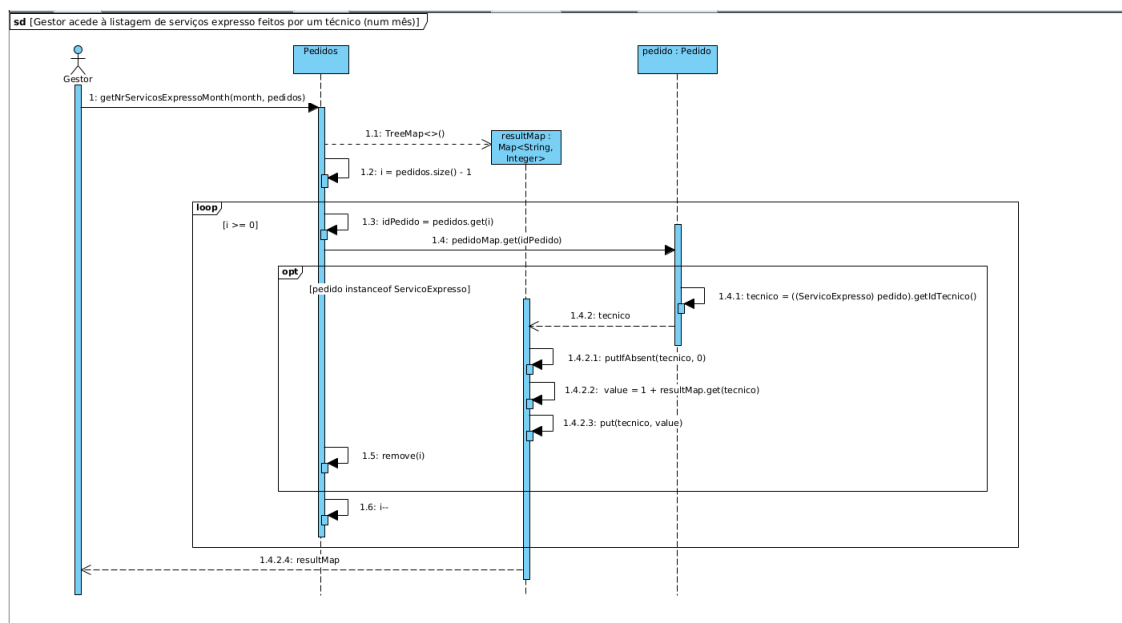
3. void addPasso(String idPlano, double horas, double custoPecas, String descricao) throws InvalidIdException;

Este método está relacionado com o caso de uso de um trabalhador efetuar a sua autenticação.

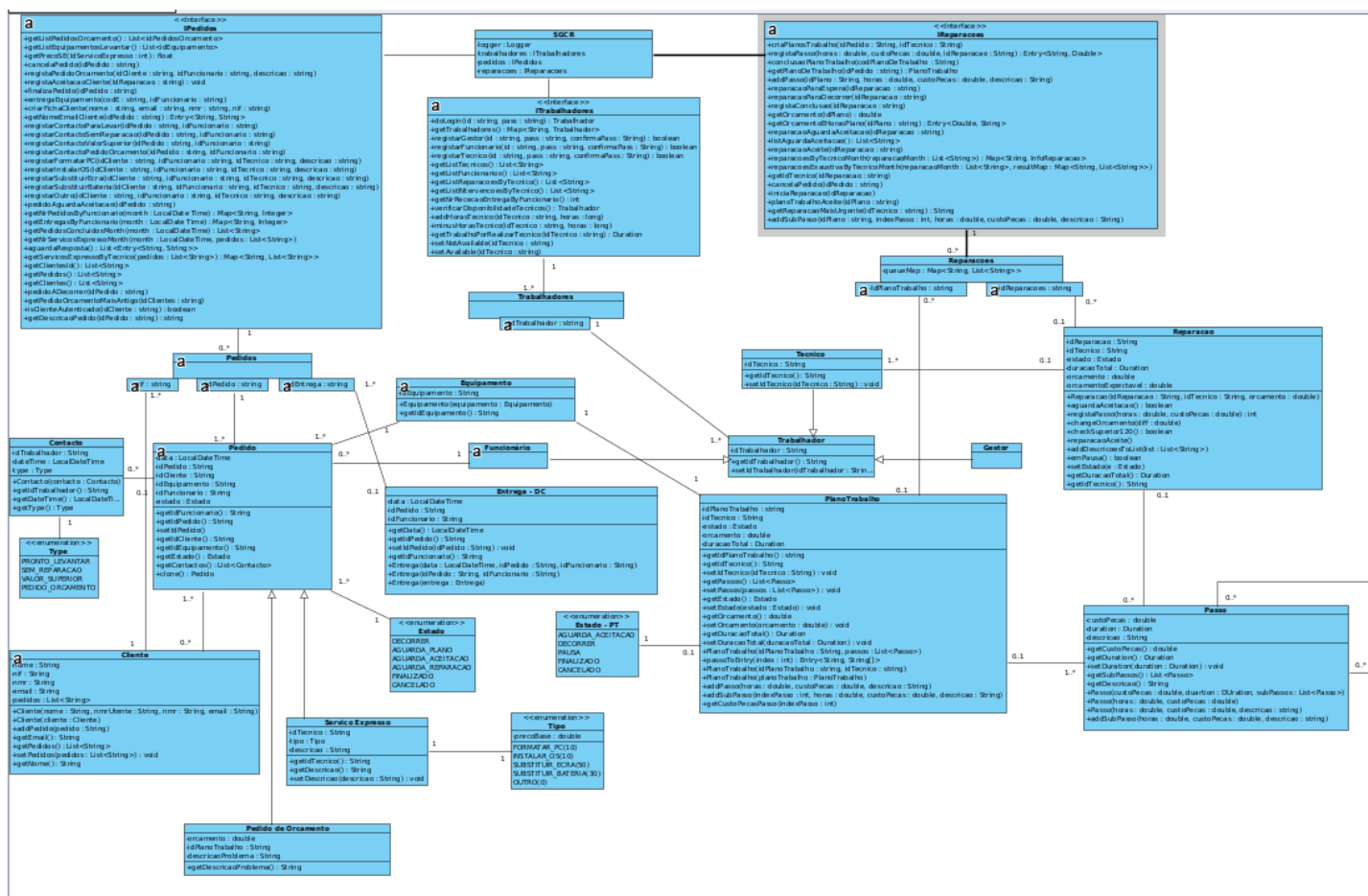


4. Map<String, Integer> getNrServicosExpressoMonth(LocalDateTime month, List<String> pedidos);

Este método está relacionado com o caso de uso de um gestor consultar a listagem de intervenções de um técnico, no final do mês, e analisar os serviços expresso.



### 3.3 Diagrama de Classe



Nota : A imagem está ilegível mas temos o diagrama em anexo.

### 3.4 Implementação

Para a implementação do nosso código achámos por bem dividir as várias classes em diversos grupos tal como podemos ver a seguir:

### 3.4.1 emailHandler

Esta secção corresponde ao envio e receção de emails utilizado no contacto entre o nosso centro e o cliente, que irá ser abordado posteriormente.

- Email

Esta classe irá apresentar os diferentes métodos para tornar possível o envio de emails para contactar o cliente e apresenta como variáveis de instância, constantes para se saber de que tipo de contacto se irá realizar.

### 3.4.2 exceptions

Aqui abordamos todas as exceções desenvolvidas para o bom funcionamento do programa, como :

- InvalidIdException -> corresponde às mensagens visualizadas ao fim de introduzir um id incorreto.
- ValorSuperior -> quando o valor da reparação é superior a 120%
- SemPedidosOrcamento -> quando nao ha pedidos de orçamento
- SemReparacoesException -> quando nao há reparação
- SemTecnicosDisponiveis -> quando não há técnicos disponiveis

### 3.4.3 gui

Esta secção foi realizada com o intuito de desenvolver a interface gráfica do programa. São visíveis várias funcionalidades desde a autenticação no sistema até aos diferentes comandos, dependendo se é o cliente ou um dos funcionários a aceder ao sistema. Para melhor organização, agrupamos as diferentes classes em quatro grupos:

- admin -> engloba as classes que representam as funcionalidades do administrador
- clientes -> engloba as classes que representam as funcionalidades dos clientes
- pedidos -> engloba classes que caracterizam as funcionalidades dos pedidos
- tecnico -> engloba as classes que representam as funcionalidades dos tecnicos

### 3.4.4 pedidos

Nesta secção desenvolvemos várias classes que constituem o funcionamento dos pedidos, nomeadamente as seguintes:

- Cliente

Esta classe é definida pelos seguintes atributos:

- nome : String -> nome do cliente
- nif : String -> NIF do cliente
- nmr : String -> n<sup>o</sup> de telemovel do cliente
- email : String -> email do cliente
- pedidos : List<String> -> pedidos do cliente

Elaboramos tambem os construtores necessários e os *getters* e os *setters*.

- Contacto

Esta classe define os diferentes tipos de contactos feitos entre o nosso centro e o cliente e é caracterizada pelos seguintes atributos:

- idTrabalhador : String -> Identificador do Trabalhador
- dateTime : LocalDateTime -> Hora do contacto
- type : Type -> este Type é um enum que definimos com os tipos de contactos que podem ser feitos, isto é,
  - \* PRONTO\_LEVANTAR, equipamento já reparado e a aguardar o levantamento do cliente
  - \* SEM\_REPARACAO, equipamento sem reparação
  - \* VALOR\_SUPERIOR, reparação do equipamento com custo superior a 120% o preço original
  - \* PEDIDO\_ORCAMENTO, contacto a informar o orçamento calculado para a reparação

Também definimos construtores e métodos de acesso a variáveis (*getters* e *setters*) que achámos pertinentes para o funcionamento da nossa aplicação.

- Entrega

A classe Entrega vai definir uma entrega de um equipamento e é caracterizada pelos seguintes atributos:

- data : LocalDateTime -> data de entrega do equipamento
- idPedido : String -> id que corresponde ao pedido
- idFuncionarios : String -> id que corresponde ao funcionário que trabalha no equipamento

Para além disso, criamos os construtores e os *getters* e os *setters* dos atributos.

- Equipamento

Nesta classe apresentamos apenas um atributo:

- idEquipamentos : String -> id do equipamento

Expomos também os construtores e os *getters* e os *setters*, que corresponde aos métodos de acesso a variáveis.

- IPedidos

Esta interface IPedidos corresponde ao subsistema Pedidos onde se encontram os protótipos dos métodos relativos à manipulação de pedidos. Estes métodos já foram devidamente explicados nos fluxos do use case e nos diagramas anteriormente ilustrados.

- Pedidos

Esta classe vai tratar de fazer a ligação entre os diferentes clientes, pedidos feitos e entregas. Apresenta os seguintes atributos:

- pedidoMap : Map<String , Pedido> -> Mapa que armazena como value os Pedidos usando como chave o id do mesmo.
- entregaMap : Map<String, Entrega> -> Mapa que armazena como value as Entregas usando como chave o id da mesma.
- clientesMap : Map<String, Cliente> -> Mapa que armazena como value os diferentes clientes e usando como chave o número de identificação fiscal do mesmo.

Para além disso, estão definidos os diferentes métodos definidos na interface IPedidos.

- PedidoOrcamento

A classe PedidoOrcamento apresenta os seguintes atributos:

- orcamento : double -> custo do arranjo
- idPlanoTrabalho : String -> plano de trabalho para o arranjo do equipamento
- descricaoProblema : String -> descrição do problema do equipamento

Estão também expostos os construtores e os métodos de acesso a variáveis (*getters* e *setters*).

- Pedido

A classe Pedido apresenta os seguintes atributos:

- data : LocalDateTime -> data do pedido.
- idPedido : String -> identificador do pedido.
- idCliente : String -> identificador do cliente que fez o pedido.
- idEquipamento : String-> identificador do equipamento
- idFuncionario : String -> identificador do funcionário que tratou do pedido.
- estado : Estado -> estado em que o pedido se encontra:
  - \* DECORRER
  - \* AGUARDA\_ACEITACAO
  - \* AGUARDA\_REPARACAO
  - \* FINALIZADO
  - \* CANCELADO
- contactos : List<Contacto> -> lista de contactos entre um trabalhador da empresa e o cliente. Cada contacto é composto pelo id do trabalhador que o enviou, a data do envio e o tipo, que pode ser:



- \* PRONTO\_LEVANTAR
- \* SEM\_REPARACAO
- \* VALOR\_SUPERIOR
- \* PEDIDO\_ORCAMENTO

Estão também expostos os construtores e os métodos de acesso a variáveis (*getters* e *setters*).

- ServicoExpresso

Por fim, elaboramos esta classe para completar este grupo. Assim, os atributos que a especificam são:

- idServicoE : String -> id do Serviço Expresso
- idTecnico : String -> id do tecnico que trata do Serviço Expresso
- tipo : Tipo -> tipo de Serviço Expresso com custos diferenciados, estas podem ser:
  - \* FORMATAR\_PC com preço base de 10
  - \* INSTALAR\_OS com preço base de 10
  - \* SUBSTITUIR\_ECRA com preço base de 50
  - \* SUBSTITUIR\_BATERIA com preço base de 30
  - \* OUTRO

Para além disso, elaboramos um preço base que esta associado ao tipo de Serviço Expresso escolhido.
- descricao : String -> descrever o Serviço Expresso que pretende

Ademais desenvolvemos os construtores os métodos de acesso a variáveis (*getters* e *setters*) que achamos adequados para o funcionamento da aplicação.

### 3.4.5 reparacoes

Seguimos o mesmo método da secção anterior por isso desenvolvemos várias classes para definir os diferentes constituintes da realização das reparações possíveis de fazer, tais como:

- InfoReparacao

Nesta classe desenvolvemos os seguintes atributos para caracterizar o funcionamento de uma reparação:

- numeroTotalReparacoes : int -> número reparações realizadas
- duracaoTotal : Duration -> duração da reparação
- desvioDuracaoHoras : long -> diferença entre as horas programadas e as horas realizadas

Elaboramos os métodos que estão implementados na interface que foram abordados no diagrama de classe e melhor explicados nos fluxos de use cases que realizamos.

- Passo

Nesta classe desenvolvemos os seguintes atributos para caracterizar um dos passos de uma reparação:

- custoPecas : double -> preço das peças a utilizar para a reparação
- duration : Duration -> duração do passo de trabalhador
- subPassos : List<Passos> -> lista com os subpassos da reparação
- descricao : String -> descrição do passo

Estão também expostos os construtores e os métodos de acesso a variáveis (getters e setters).

- PlanoTrabalho

- idPlanoTrabalho : String -> id do plano de trabalho acerca da reparação
- idTecnico : String -> id do tecnico que trata da reparação
- passos : List<Passo> -> lista de passos que constituem o plano de trabalho
- estado : Estado -> estado da reparação, esta pode ser:
  - \* AGUARDA\_ACEITACAO
  - \* DECORRER
  - \* PAUSA
  - \* FINALIZADO
  - \* CANCELADO
- orcamento : double -> custo total a pagar
- duracaoTotal : Duration -> duração do plano de trabalho

Abordamos os métodos que são possíveis ver na interface no diagrama de classe criado.

- Reparacao

- idReparacao : String -> identificador da reparação
- idTecnico : String -> identificador do técnico que realizou a reparação
- estado : Estado -> estado da reparação
- passos : List<Passo> -> lista de passos que constituem o plano de trabalho
- duracaoTotal : Duration -> duração de uma reparação
- passos : List<Passo> -> lista de passos que foram realizados numa reparação

- orçamento : double -> custo final da reparação
- orçamentoExpectavel : double -> custo previsto para a reparação antes de a efetuar

Estão também expostos os construtores e os métodos de acesso a variáveis (getters e setters), para além disso estão implementados alguns métodos da interface que podemos ver pelo diagrama de classe.

- Reparacoes

Para tratar das reparações desenvolvemos estes atributos:

- reparacaoMap : Map<String, Reparacao> -> Mapa que armazena como *value* as reparações usando como chave o seu próprio id
- planoTrabalhoMap : Map<String, PlanoTrabalho> -> Mapa que armazena como *value* os planos de trabalho usando como chave o id do mesmo
- queueMap : Map<String, Queue<String> -> Mapa que armazena como *value* a lista de com os ids das reparações associadas a um tecnico (chave - id do tecnico).

Ademais estruturamos os métodos que se encontram na interface do diagrama de classe

- IReparacoes

Corresponde a interface onde estão apresentados todos os métodos que desenvolvemos noutras classes

### 3.4.6 trabalhadores

Nesta secção desenvolvemos várias classes que correspondem aos diferentes trabalhadores existentes, como podemos verificar seguidamente:

- Trabalhador

Esta classe (pai) irá definir os vários trabalhadores do nosso sistema, apresentando os seguintes atributos:

- idTrabalhador : String -> Id do trabalhador.
- passe : String -> Palavra passe para aceder ao sistema do centro.
- autenticado : boolean -> Variável que indica se o trabalhador já se encontra ou não autenticado

Também definimos construtores e métodos de acesso a variáveis (*getters* e *setters*) que achámos pertinentes para o funcionamento da nossa aplicação.

As seguintes classes são subclasses da classe Trabalhador e que definem os vários tipos que podem existir, isto é, funcionário balcão, técnico de reparações e o gestor do centro.

- Funcionario

- Tecnico
- Gestor
- ITrabalhadores

Esta interface ITrabalhadores corresponde ao subsistema Trabalhadores onde se encontram os protótipos dos métodos relativos à manipulação de trbalhadores. Estes métodos já foram devidamente explicados nos fluxos do use case e nos diagramas anteriormente ilustrados.

- Trabalhadores

- trabalhadores : Map<String, Trabalhador> -> Mapa que armazena como value os Trabalhadores usando como chave o id do mesmo.

Implementamos os métodos que se encontram na interface que podemos ver no diagrama de classe

## 3.5 sgrc

Esta secção irá conter a classe sgrc que realizará a ligação entre os nossos subsistemas e interface do programa.

### 3.5.1 SGRC

Esta classe apresenta então, os seguintes atributos:

- trabalhadores : Itrabalhadores -> interface dos trabalhadores
- pedidos : IPedidos -> interface dos pedidos
- reparacoes : IReparacoes -> interface das reparações

Estes atributos correspondem às nossas interfaces já anteriormente abordadas e permitirá assim fazer a ligação entre os 3 subsistemas do nosso projeto. Os métodos desta classe vão ser então os vários métodos .

### 3.5.2 utils

As classes desenvolvidas,Constantes, FileUtils e SecurityUtils,bem como as HintTextArea e HintTextField, tem como objetivo auxiliar com informação, que completa o bom funcionamento do programa

## 4 Main

Para além destes grupos criados anteriormente, temos tambem definida a classe Main que tem como intuito correr o programa.

## 5 Interface

Nesta secção iremos colocar imagens legendadas da interface gráfica do nosso projeto.



Menu de Autenticação.



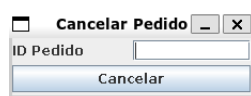
Menu Inicial. (Pedidos).



Menu Inicial. (Clientes)

ID	ID_CLIENT	ID_FUNCIONARIO	DESCRIÇÃO	ORÇAMENTO	ID_PLANO_TRABALHO	ESTADO	ENTREGUE
0	123	mendes	ok	Nao Definido	0	DECORRER	false
1	123	di	po	Nao Definido	1	AGUARDA_REPARACAO	false

Opção Listar Pedidos Orçamento.



Opção Cancelar Pedido.



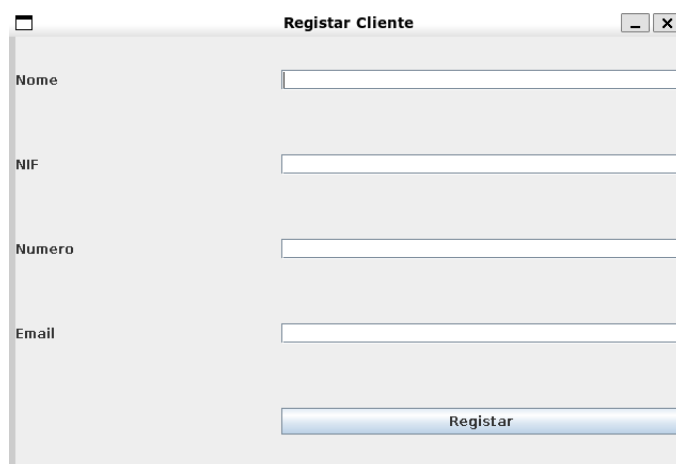
A screenshot of a software window titled "Registrar Pedido". The window has a standard Mac OS X-style title bar with a close button (X) and a minimize button (-). The main area is light gray. On the left, there are two labels: "Cliente" and "Descrição". To the right of "Cliente" is a single-line text input field. To the right of "Descrição" is a multi-line text area with the placeholder text "Explique o pedido do cliente". At the bottom right of the window is a blue button labeled "Registrar".

Opção Registrar Pedido.



A screenshot of a software window titled "Regista Entrega". The window has a standard Mac OS X-style title bar with a close button (X) and a minimize button (-). The main area is light gray. On the left, there is a label "ID Pedido". To the right of "ID Pedido" is a single-line text input field. At the bottom of the window is a blue button labeled "Registrar Entrega".

Opção Registrar Entrega.



A screenshot of a software window titled "Registrar Cliente". The window has a standard Mac OS X-style title bar with a close button (X) and a minimize button (-). The main area is light gray. On the left, there are four labels: "Nome", "NIF", "Numero", and "Email". To the right of each label is a single-line text input field. At the bottom right of the window is a blue button labeled "Registrar".


Opção Registrar Cliente.



The 'Listar Clientes' window displays a table with the following data:

Nome	NIF	Numero	Email	Pedidos
to	123		andre.vaz1411@gm...	[0, 1]

Opção Listar Clientes.



The 'Concluir Serviço Expresso' window includes a text input field for 'ID Pedido' and a 'Concluir' button.

Opção Concluir Serviço Expresso.



The 'Registrar Trabalhadores' window contains the following fields and controls:

- Cargo:** A dropdown menu currently showing 'Funcionario'.
- Username:** A text input field.
- Password:** A text input field.
- Confirma Password:** A text input field.
- Registrar:** A button at the bottom of the form.

Opção Registrar Trabalhadores.





Menu Inicial (Administrador).

Listar Trabalhadores

Username	Cargo
di	Tecnico
fn14	Gestor
mendes	Funcionario

Opção Listar Trabalhadores.

Listar Reparações por Tec...

? mes/ano

Yes No

Lista de Reparações por Técnico			
Id Técnico	Nr Reparacoes	Duracao Média	Desvio em relação à duração prev...

Opção Listar Reparações por Técnico.

Lista de Intervenções dos Técnicos	
Id Técnico	Intervencao

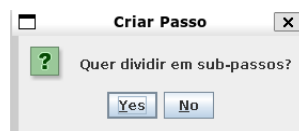
Opção Listar Intervenções por Técnico.



Menu Inicial (Técnico).



Opção Criar Plano de Trabalho.



Opção Criar Passo.

## SGCR - Pedido de Orçamento Inbox x



**sgcrgrupo35@gmail.com**

to me ▾

Caro to,

Informamos que o orçamento relativo ao seu pedido será 1000.0 euros.  
E que serão necessárias aproximadamente 2 dias de trabalho.  
Responda a este email caso aceite.  
Ignore caso contrário. E faça o levantamento do equipamento na loja

Cumprimentos, SGCR.

Exemplo de um email.

## 6 Conclusão

Nesta fase final do projeto constatamos que conseguimos cumprir com tudo o que nos foi pedido.

Na nossa opinião, sentimos alguma dificuldade em pensar e estruturar tudo de forma correta para que quando fossemos implementar código estivesse tudo correto. A interface gráfica também foi algo bastante desafiante.

Apesar disso, consideramos que cumprimos tudo o que nos foi pedido.

Enquanto grupo, conseguimos distribuir bem o trabalho entre todos. Ajudamo-nos mutuamente e, de forma geral, o grupo teve um aproveitamento positivo.

Concluindo, este trabalho ajudou-nos a desenvolver novas aptidões e a consolidar toda a matéria lecionada em aula.