IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE ESTÁGIO

Licenciatura em Engenharia Informática

André Filipe Cunha Teixeira

janeiro | 2021

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE ESTÁGIO

André Filipe Cunha Teixeira

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Janeiro|2021

Agradecimentos

Tenho que começar por agradecer ao meu Pai e Mãe, sem eles nada disto era possível, enorme gratidão por todos os esforços que fizeram e motivação para que conseguisse tirar um curso que sonhava. Um objetivo está realizado graças a Vocês.

Toda a minha família, muito obrigado pela presença nesta fase da minha vida, pela ajuda e confiança que depositaram em mim.

Agradeço muito ao meu orientador, Prof. António Mário Ribeiro Martins, pela ajuda, por me ter acompanhado nesta fase final tão importante e pela disponibilidade para me orientar. Um enorme obrigado professor.

Sendo também uma parte importante neste meu desenvolvimento enquanto engenheiro, quero agradecer a todos os elementos da empresa MediaSis, pelas amizades criadas e pela toda ajuda prestada. Para o meu supervisor Eng. Vitor Moura por toda a simpatia e acompanhamento em toda a minha fase de integração, um sincero agradecimento, e também ao Eng. Marco Carneiro por toda a paciência e ajuda no desenvolvimento da aplicação, muito obrigado.

Um agradecimento muito especial para a minha namorada que se não fosse ela a dar-me força em fases difíceis ao longo do curso não teria sido possível concluir este objetivo. Aos seus pais também um agradecimento muito especial.

Aos meus amigos todos, um forte abraço de agradecimento, por todas as conversas e desabafos ao longo destes anos, bem como a todas as partilhas de notas em todos os aspetos da vida.

Ficha de Identificação

Aluno:

Nome: André Filipe Cunha Teixeira

Nº: 1011847

Licenciatura: Engenharia Informática

Estabelecimento de Ensino:

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Tel: 271 220 100 | Fax: 271 222 690 | E-mail: assessoria@ipg.pt

Local de Estágio:

MediaSis, Soluções de Informática Lda

Rua Manuel de Arriaga, 157 B, 4440-044 Campo Valongo

Tel: 224 223 766 | Site: www.mediasis.pt | E-mail: mediasis@mediasis.pt

Supervisor:

Nome: Vitor Aníbal de Moura

E-mail: vmoura@mediasis.pt

Orientador de Projeto:

Nome: Prof. António Mário Ribeiro Martins

E-mail: amrmartins@ipg.pt

Resumo

O presente relatório representa o projeto realizado em contexto de estágio no âmbito da unidade curricular Projeto de Informática, integrada na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

O projeto realizado consta do desenvolvimento de uma aplicação que faz a sincronização dos e-mails de suporte da empresa MediaSis com o software comercializado GPAC. O principal objetivo é a visualização das tarefas enviadas por e-mail no software GPAC em tabelas, facilitando assim a resolução, onde os e-mails são catalogados consoante vários parâmetros, fazendo a distinção de outros vindos externamente do software GPAC. Outro objetivo é a possibilidade de o cliente ter acesso ao estado do ticket sabendo assim em que ponto está a situação e não fazendo repetição de e-mails.

Para a realização do projeto foram feitas algumas recolhas de informação sobre aplicações idênticas, para possível integração ao software GPAC, mas devido a uniformização pretendida desenvolveu-se uma aplicação de raiz. Para o desenvolvimento da aplicação foram usadas tecnologias como ASP.NET e para o acesso há base de dados SQL Server.

O projeto foi desenvolvido com sucesso, superando o objetivo proposto pela empresa, que deu uma avaliação positiva. Foi muito gratificante realizar o projeto devido ao ambiente da empresa, tendo um feedback positivo e que pretende evoluir mais a aplicação, bem como trabalhar com tecnologias que tem bastante interesse.

Palavras-chave

MediaSis, GPAC, ASP.NET, SQL Server, Sincronização de e-mails

Abstract

This report represents the project carried out in the context of an internship in the Computer Science Project unit, integrated in the degree in Computer Science Engineering of the School of Technology and Management, Polytechnic Institute of Guarda.

This project appears in the development of an application that synchronizes the support e-mails of the company MediaSis with the software marketed GPAC. The main objective is the visualization of the tasks sent by e-mail in the GPAC software in tables, thus facilitating the resolution, where the e-mails are catalogued according to several parameters, making the distinction from others coming from the GPAC software. Another objective is the possibility of the customer to have access to the ticket status thus knowing at what point the situation is and not making repetition of e-mails.

For the realization of the project, some information was collected about identical applications, for possible integration to the GPAC software, but due to uniformization required, an application was developed from scratch. For the development of the application technologies like ASP.NET were used and for the access there is SQL Server database.

The project was successfully developed, surpassing the objective proposed by the company, which gave a positive evaluation. It was very rewarding to realize the project due to the company's environment, having a positive feedback and that belong more to evolve the application, as well as working with technologies that have a lot of interest.

Keywords

MediaSis, GPAC, ASP.NET, SQL Server, Synchronizes the e-mails

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

BD - Base de dados

DYP - Define Your Process

GUI - Graphical User Interface

HTML - HyperText Markup Language

IDE - Integrated Development Environment

SQL - Structured Query Language

SSMS - SQL Server Management Studio

UML - Unified Modeling Language

VB -Visual Basic

XML - Extensible Markup Language

XP - Extreme Programming

Id - Identidade

GPAC - Gestão da Produção Assistida por Computador

1. Introdução

O presente relatório foi elaborado para descrever o projeto realizado em contexto de estágio, no âmbito da unidade curricular de Projeto de informática do terceiro ano do curso de Engenharia Informática.

Neste primeiro capítulo é dada a conhecer a entidade empregadora, bem como o enquadramento do problema e o objetivo da aplicação desenvolvida. Também são apresentadas as etapas que foram feitas na entidade.

1.1. Caracterização da Empresa

A instituição da realização do estágio foi a MediaSis. Criada em 1992, com seis colaboradores, tem como foco principal modelar processos produtivos e operacionais, através do software desenvolvido pela empresa, para aplicar a qualquer setor de atividade, além de outras atividades. Situada na Rua Manuel de Arriaga em Valongo, é uma empresa que trabalha com clientes do setor mobiliário, distribuição, exploração de pedra, metalúrgico e restauração.

Figura 1 – Logotipo da empresa

O software tem como nome GPAC, com várias funcionalidades a nível da faturação, contagem de *stocks*, guias de compra e transporte, bem como fazer análises de vários processos da empresa. Além destas funcionalidades o software tem uma tecnologia desenvolvida pela empresa, DYP, que é responsável por toda a modelação e parametrização do processo produtivos, sendo uma tecnologia desenvolvida a pensar no processo antes de pensar no produto.

1.2. Enquadramento

Na empresa, o método mais usual para a resolução de problemas no software do cliente é por chamada telefónica, no entanto, e por ser um pequeno grupo, nem sempre há colaboradores disponíveis para atender todos os pedidos, e nessa situação ou em dificuldades mais específicas e demoradas o cliente acaba por mandar um e-mail, com todos os tópicos para resolução, ao colaborador da MediaSis.

Este envio de e-mails por parte do cliente acaba por significar a demora da sua resolução e/ou por vezes o e-mail surgia repetido por causa do tempo de execução ou até mesmo pelo cliente achar que ainda não estava resolvido.

Neste contexto, a pedido do responsável da empresa, surgiu a necessidade da criação de uma aplicação que fizesse a sincronização dos e-mails que os clientes enviam para o e-mail suporte da empresa para o software GPAC.

1.3. Objetivos

O objetivo da aplicação é facilitar tanto para o cliente como para o colaborador a visualização dos erros pendentes e resolvidos e facilitar a resolução de problemas com várias prioridades e não resolver problemas repetidos por causa de um e-mail duplicado.

1.4. Etapas de estágio

O estágio constou nas seguintes etapas:

- Familiarização do ambiente da empresa;
- Introdução ao software GPAC;
- Análise do problema;
- Implementação da aplicação com o software GPAC;
- Testes e divulgação.

2. Estado da Arte

Neste capítulo está descrito o estado da arte, que visa o estudo de aplicações semelhantes no mercado e a comparação destas em relação à aplicação a desenvolver.

2.1. Introdução

Não havendo aplicações semelhantes à pretendida, devido à aplicação personalizada ao software da empresa, foi feita uma comparação com o trabalho realizado pelo meu colega Leonardo Lourenço[1].

2.2. Comparação

Tal como o trabalho realizado pelo meu colega Leonardo, muitas tecnologias não permitem a integração com bases de dados externas, que é necessária para a realização deste projeto.

Resolveu-se, então, criar a própria aplicação que fizesse a integração dos e-mail de suporte da empresa ao software da empresa, GPAC.

2.3. Análise

A aplicação pretendida pela empresa tinha que ser integrada ao software GPAC, bem como fazer ligação entre base de dados da empresa e do cliente. Como foi dito anteriormente muitas tecnologias no mercado não estão aptas para fazer este tipo de ligação, não cumprindo todos os requisitos pretendidos pela empresa para a gestão de *tickets* dos clientes.

O software GPAC já continha uma opção de envio de e-mails, sendo então aproveitada para a solução do problema, usando o e-mail de suporte da empresa para ser sincronizado e podendo assim fazer a total uniformização da aplicação, gerindo os e-mails a serem tratados e lidos.

3. Metodologia

Nesta parte do relatório é descrita a metodologia a seguir na elaboração do projeto.

Entende-se por metodologia, a forma de se utilizar um conjunto coerente e coordenado de métodos para atingir um objetivo. [2]

Também neste capítulo são descritas as tarefas a serem realizadas ao longo do desenvolvimento do projeto bem como o tempo da realização das mesmas.

3.1. Extreme Programming (XP)

Para o desenvolvimento deste projeto foi utilizada a metodologia de Desenvolvimento Ágil, Extreme Programming (XP). É um método de desenvolvimento para pequenos e médios projetos onde a grande preocupação é equilibrar as variáveis custo, tempo e qualidade do produto. [3]

As principais vantagens deste método são: [4]

- **Simplicidade** – reduz a complexidade do problema do sistema;
- **Comunicação** – facilita a comunicação entre membros da equipa. Todos os problemas, dúvidas e preocupações são do conhecimento de todos;
- **Retro Alimentação** – quando se implementa uma funcionalidade nova, deve ser validada, assim garante-se que o projeto caminha no melhor sentido;
- **Coragem** – para que o método resulte é preciso um compromisso para aplicá-lo em diversos cenários que possam surgir, garante a confiança do programador.

Figura 2 - Resumo da metodologia XP

3.2. Descrição das Tarefas

De acordo com os objetivos descritos no capítulo 1 foram identificadas as seguintes tarefas:

1. Análise de requisitos;
2. Discussão com a entidade sobre o trabalho a ser realizado;
3. Criação da base de dados;
4. Discussão dos campos a serem utilizados no programa GPAC;
5. Criação de métodos de sincronização entre a aplicação e o programa GPAC;
6. Testes da aplicação;
7. Elaboração do relatório.

3.3. Mapa de Gantt

Tarefa	Início	Conclusão	Duração	Jul	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar
1	01/07/2019	19/07/2019	15d									
2	22/07/2019	26/07/2019	5d									
3	29/07/2019	30/07/2019	2d									
4	31/07/2019	02/08/2019	3d									
5	05/08/2019	27/09/2019	40d									
6	30/09/2019	04/10/2019	5d									
7	07/10/2019	27/03/2020	115d									

Figura 3 - Mapa de Gantt das tarefas realizadas

4. Análise de Requisitos

Os requisitos de um sistema descrevem os procedimentos dos serviços que o sistema deve fazer, oferecer e restringir. O processo de descobrir, analisar, documentar e verificar os serviços e restrições é chamado de engenharia de requisitos. [5]

Neste capítulo são descritos os elementos da linguagem UML (Unified Modeling Language) que, com base numa linguagem padrão, são modeladas, com o intuito de facilitar a compreensão do sistema para o programador. Esta linguagem não é uma metodologia de desenvolvimento, mas sim um auxílio na visualização do seu desenho e comunicação entre objetos. [6]

4.1. Diagrama de Contexto

Um ticket é um pedido de assistência por parte do cliente, mas para a aplicação é um simples e-mail que o cliente envia através do GPAC para o e-mail de suporte da empresa com o erro ocorrido ou uma alteração desejada. A aplicação desenvolvida vai tratar de todos os e-mails que estejam na caixa de entrada do e-mail de suporte da empresa usando o cabeçalho para distinguir entre e-mails tickets de outros e-mails porque os e-mails enviados pelo cliente através do GPAC têm um formato próprio, sendo que o cliente só preenche o assunto e o corpo da mensagem, enquanto que o GPAC retira informações para a elaboração do cabeçalho com o código de cliente, o número do documento e o assunto que o cliente colocou.

O diagrama de contexto é composto por fluxos de dados, mostrando as relações entre o sistema e as entidades externas, apresentando o sistema como um único processo, facilitando a visualização do sistema e a sua compreensão.

No sistema de sincronização do MediaSisMail a única relação é com outros sistemas, como o Outlook e a Base Dados (BD), tal como podemos ver na figura 4.

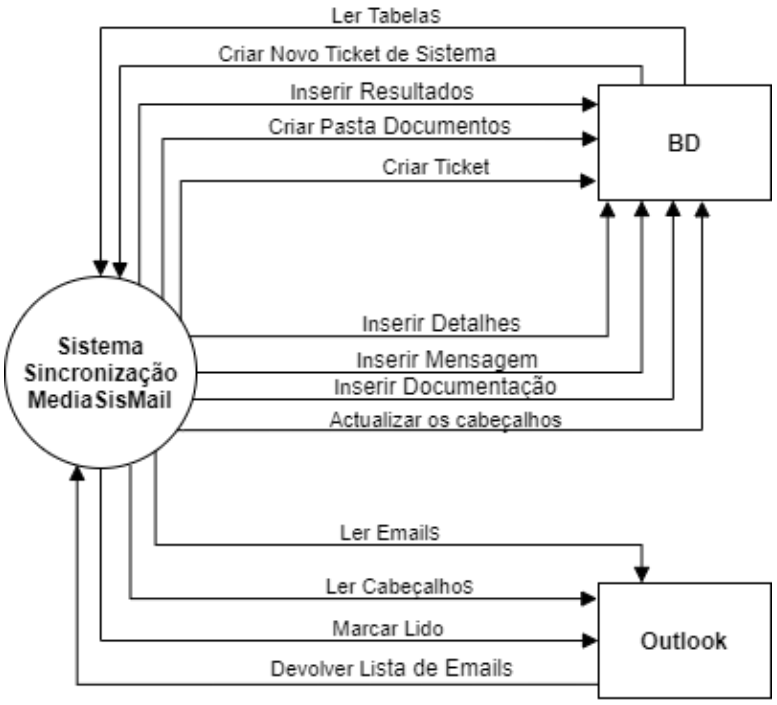

Figura 4 - Diagrama de Contexto

4.2. Casos de Uso

Atores e Respetivos Casos de Uso

A tabela seguinte (Tabela 1) tem como objetivo indicar o ator/atores, com os seus casos de uso que interagem com o sistema. Os casos de uso são narrativas em texto, representando os requisitos no sistema. [7]

Atores	Casos de Uso	Objetivo
Outlook	Ler e-mails	O objetivo é o sistema ler os e-mails de suporte
	Ler cabeçalhos	O objetivo é o sistema selecionar os e-mails com o cabeçalho de ticket
	Devolver lista de e-mails por ler	O objetivo é o sistema obter os e-mails por ler
	Marcar como lido	O objetivo é o sistema marcar os e-mails como lidos

Base Dados (BD)	Ler tabelas	O objetivo é o sistema consultar as tabelas e o seu *id*
	Inserir resultados	O objetivo é o sistema inserir os tickets
	Criar ticket	O objetivo é o sistema criar o ticket na base dados
	Inserir detalhes	O objetivo é o sistema inserir os detalhes dos e-mails
	Inserir Mensagem	O objetivo é o sistema inserir as mensagens dos e-mails
	Inserir Documentação	O objetivo é o sistema inserir a documentação anexa dos e-mails
	Criar pasta documentos	O objetivo é o sistema criar uma pasta documentos para os anexos
	Atualizar os cabeçalhos	O objetivo é o sistema fazer uma atualização dos cabeçalhos dos tickets criados com informação obtida na base dados
	Criar novo ticket no sistema	O objetivo é o sistema ter o ticket novo no formato a ser lido no software GPAC

Tabela 1 - Atores e respetivos casos de uso

Diagrama de Casos de Uso

O diagrama de casos de uso, como se representa na figura 5, descreve um simples cenário para a compreensão de todos os intervenientes nos requisitos do sistema, mostrando o que vai efetuar e não como o vai fazer. [8]

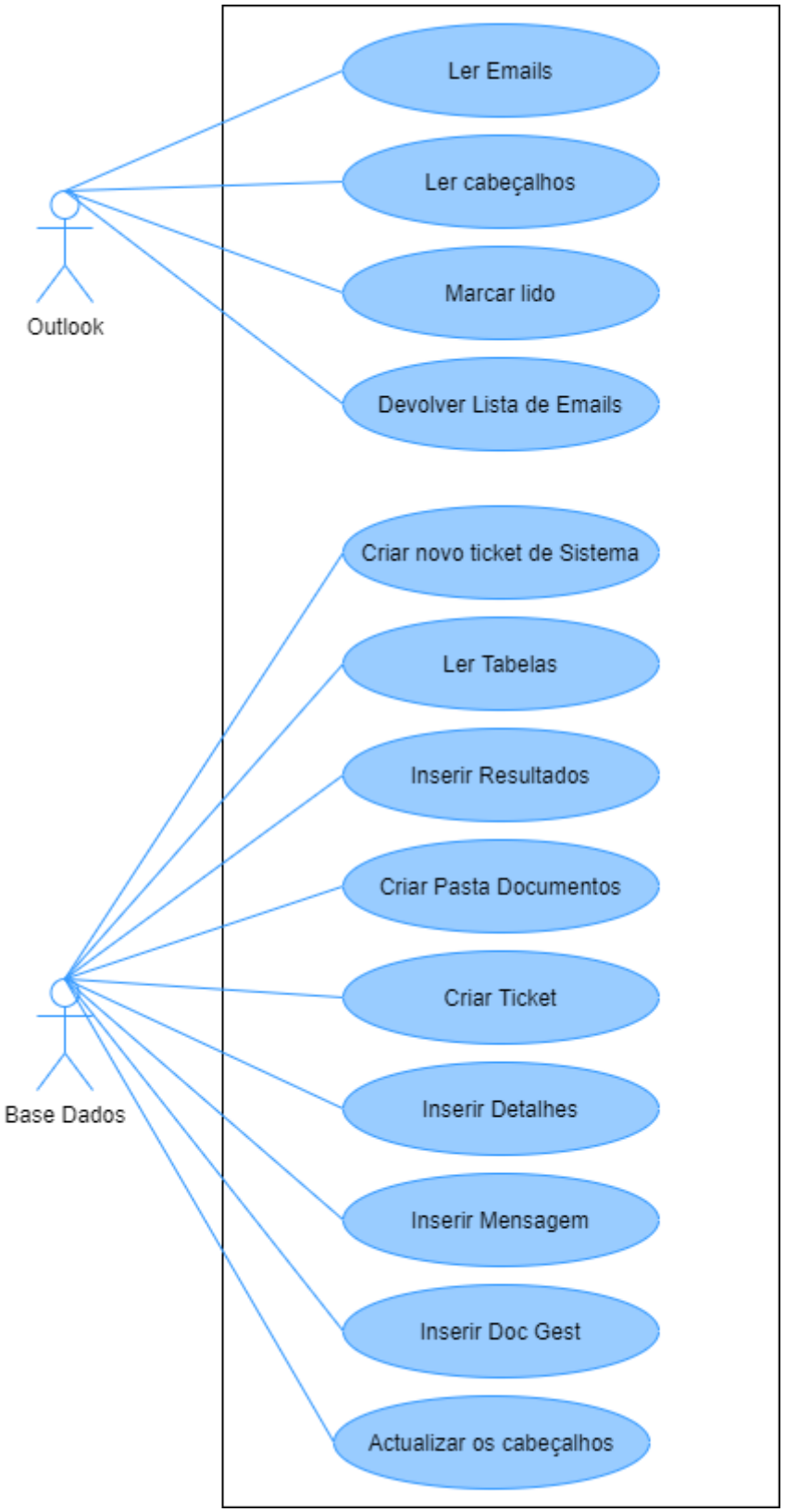

Figura 5 - Diagrama Casos de Uso

Descrição de Casos de Uso

Esta parte do relatório descreve alguns casos de uso que foram utilizados (Tabelas 2, 3, 4, 5 e 6). As tabelas seguem a seguinte estrutura:

Nome: Indica o nome do caso de uso.

Descrição: Descreve o objetivo do caso de uso.

Atores Envolvidos: Indica os atores que interferem no caso de uso.

Pré Condições: Indica se tem alguma pré condição necessária para o início do caso de uso.

Caminho Principal: Mostra as etapas de interação entre os atores e o sistema.

Caminho Secundário: Mostra caminhos alternativos ao principal caso algo corra mal ou haja validações.

Suplementos: Indica os casos de testes concretos ao caso de uso.

Ler E-mails

A tabela seguinte (Tabela 2) descreve o processo de ler os e-mails da conta suporte.

Nome	Ler E-mails
Descrição	O objetivo é o sistema ler os e-mails de suporte
Atores Envolvidos	Outlook
Pré Condições	E-mail suporte aberto
Caminho Principal	1. O colaborador executa a aplicação 2. A aplicação lê todos os e-mails por ler
Caminho Secundário	Não tem
Suplementos	Verificar se todos os e-mails estão lidos

Tabela 2 - Ler E-mails

Ler cabeçalhos

A tabela seguinte (Tabela 3) descreve o processo de ler os cabeçalhos dos e-mails na conta suporte.

Nome	Ler cabeçalhos
Descrição	O objetivo é o sistema selecionar os e-mails com o cabeçalho de ticket
Atores Envolvidos	Outlook
Pré Condições	E-mail suporte aberto
Caminho Principal	1. O colaborador executa a aplicação 2. A aplicação filtra os e-mails por cabeçalho de ticket 3. A aplicação guarda todos os e-mails com o cabeçalho de ticket 4. A aplicação trata da mensagem do e-mail 5. A aplicação guarda os dados do e-mail
Caminho Secundário	3. a) A aplicação descarta e-mails sem o cabeçalho pretendido
Suplementos	Verificar se todos os e-mails estão lidos

Tabela 3 - Ler Cabeçalhos

Marcar como lido

A tabela seguinte (Tabela 4) descreve o processo de como o sistema marca como lidos os e-mails da conta suporte.

Nome	Marcar como lido
Descrição	O objetivo é o sistema marcar os e-mails como lidos
Atores Envolvidos	Outlook
Pré Condições	E-mail suporte aberto
Caminho Principal	1. O colaborador executa a aplicação 2. A aplicação lê todos os e-mails por ler 3. A aplicação depois de fazer uma rotina no e-mail suporte todo, marca os e-mails como lidos
Caminho Secundário	Não tem
Suplementos	Verificar se todos os e-mails estão lidos

Tabela 4 – Marcar como lido

Criar pasta documentos

A tabela seguinte (Tabela 5) descreve o processo de criação da pasta documentos para guardar os anexos dos e-mails.

Nome	Criar pasta documentos
Descrição	O objetivo é o sistema criar uma pasta documentos para os anexos
Atores Envolvidos	Base dados
Pré Condições	Não tem
Caminho Principal	1. A aplicação cria na base dados do cliente a pasta documentos 2. Caso existam anexos guarda na pasta documentos
Caminho Secundário	1. a) Caso exista a pasta não cria novamente 2. a) Caso não existam anexos não guarda nenhuma informação
Suplementos	Verificar se a pasta documentos está criada Verificar caso o e-mail tenha anexos se eles foram guardados na pasta

Tabela 5 - Criar pasta documentos

Criar novo ticket no sistema

A tabela seguinte (Tabela 6) descreve o processo de como a aplicação cria um novo ticket para interagir com o software GPAC.

Nome	Criar novo ticket no sistema
Descrição	O objetivo é o sistema ter o ticket novo no formato a ser lido no software GPAC
Atores Envolvidos	Base dados
Pré Condições	Não tem
Caminho Principal	1. A aplicação verifica o ticket 2. A aplicação analisa o cabeçalho 3. A aplicação trata os dados decompondo em XML 4. A aplicação envia os dados para o software GPAC
Caminho Secundário	Não tem
Suplementos	Verificar os dados no software GPAC

Tabela 6 - Criar novo ticket no sistema

4.3. Diagrama de Sequência

Os Diagramas de Sequência representam interações entre objetos numa sequência temporal de mensagens. São utilizados para representar casos de uso para mostrar o fluxo de mensagens e ações entre objetos e componentes.[8]

Neste projeto será mostrado um diagrama que representa o percurso de um ticket após a execução da aplicação desenvolvida, como mostra na figura 6.

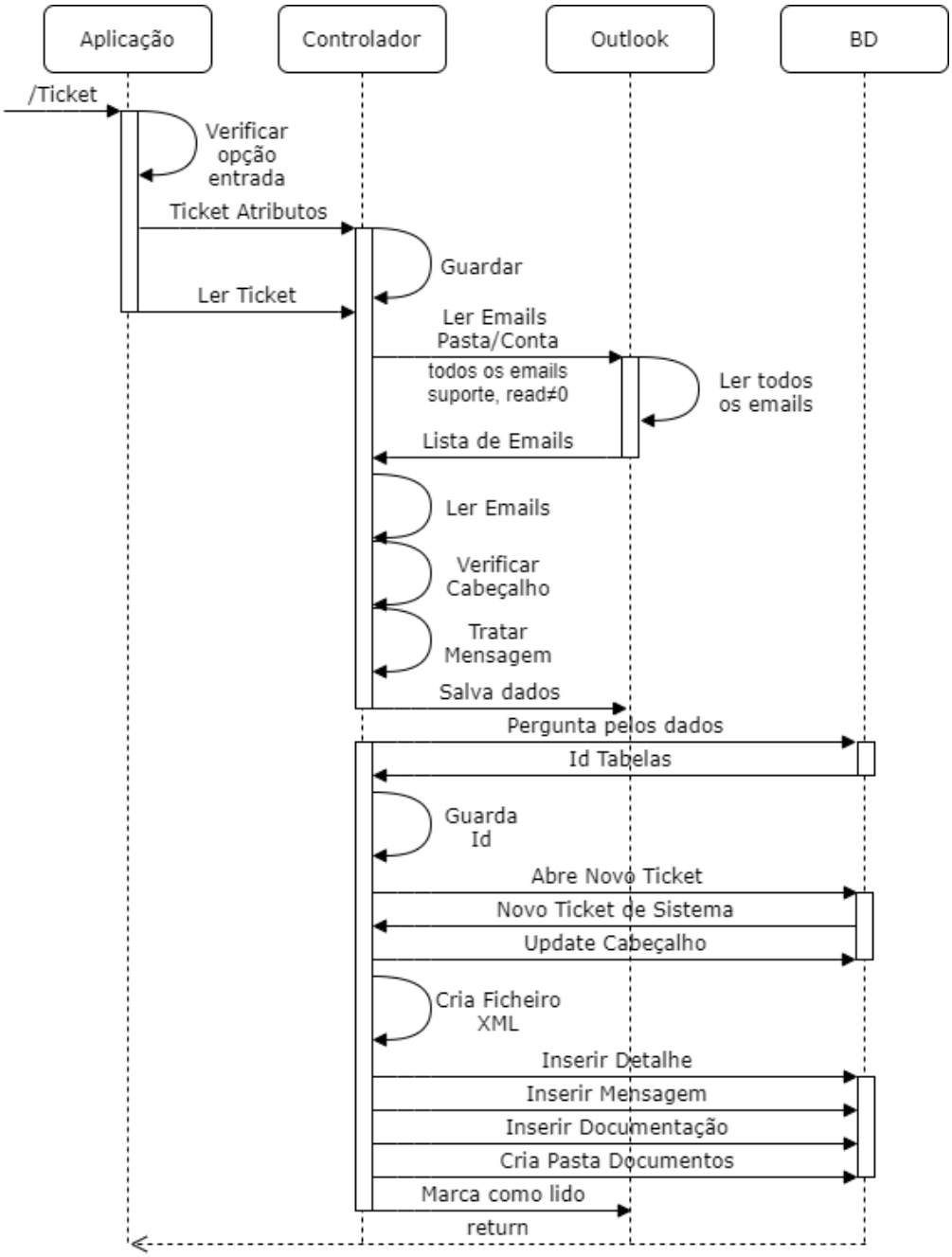

Figura 6 - Diagrama de sequência

4.4. Diagrama de Classe

Neste tópico visualizamos o diagrama de classes do sistema, representado pela figura 6. Um diagrama de classes representa uma estrutura e relações das classes, onde podemos identificar e agrupar os objetos. A estrutura de uma classe no diagrama é um retângulo com três espaços. A parte superior é o nome da classe, parte do meio é os atributos da classe e na parte inferior são os métodos e as operações da classe.[9]

Neste projeto são utilizadas três classes, sendo a classe MediaSisControlador a principal, onde são executados todos os métodos e operações para a sincronização dos e-mails. Depois temos também a classe Base Dados, que serve para fazer ligação entre a base de dados do software GPAC suporte e o cliente, para ir buscar informação, e por fim, temos a classe Outlook, que tem como função a leitura dos e-mails na conta de suporte.

Figura 7 - Diagrama de classe do sistema

4.5. Dicionário de dados

Um dicionário de dados é uma descrição de todos os campos do diagrama de classes, descrevendo o tipo de dados e a descrição.

Nome do campo	Tipo de dados	Descrição
_pasta	string	Define a pasta
_caixa	string	Define o correio eletrónico a usar
_dir	string	Define o diretório da empresa
findString	List<string>	Lista com todas as strings
obj	Outlook	Objetos do Outlook
rootFolder	MAPIFolder	Move uma pasta para a pasta de destino especifica
iPasta	List<Folder>	Lista de caminhos para as pastas
imail	List<MailItem>	Lista de mensagens do e-mail
ficheiros	List<string>	Lista dos ficheiros
ficheirosBody	List<string>	Lista do corpo dos ficheiros
idMail	List<string>	Lista de e-mails
_conta	string	Define a conta
_tabela	string	Define a tabela
_doc	string	Define o documento
_serie	string	Define a série
NSerie	DataSet	Início da série
NSerieN	int	Incrementa +1 ao número de série atual
NSerieT	int	Número de série na tabela
pasta	string	Pasta procurada
caixa	string	Correio eletrónico procurado
dir	string	Diretório da empresa
codigodocumento	string	Código de documento procurado
seriedocumento	string	Série de documento procurado

Odbc	string	Acesso base de dados
Conta	string	Conta procurada
Tabela	string	Tabela procurada

Tabela 7 – Dicionário de dados da classe MediaSisControlador

Nome do campo	Tipo de dados	Descrição
myId	int	Define o Id
MyconString	string	Define a palavra de conexão
myTransact	odbcTransaction	Define a transação com a base de dados
myConect	odbcConnection	Define a conexão com a base de dados
id	int	Id procurado
conString	string	Palavra de conexão procurada
CurrentTransaction	odbcTransaction	Atual transação

Tabela 8 - Dicionário de dados da classe Base Dados

Nome do campo	Tipo de dados	Descrição
Titulo	string	Define o assunto do e-mail
Cc	string	Define o campo CC do e-mail
To	string	Define o destinatário do e-mail
Bcc	string	Define o BCC do e-mail
Corpo	string	Define o corpo do e-mail
oItem	MailItem	Define a nova mensagem do e-mail
app	Application	Cria os itens necessários para fazer a catalogação do e-mail
Mapi	Store	Enumera e procura as pastas na sessão atual
Conta	Account	Define a conta atual
rootFolder	MAPIFolder	Move uma pasta para a pasta de destino especifica

_para	string	Destinatário procurado
_titulo	string	Assunto procurado
_cc	string	CC procurado
_bcc	string	BCC procurado
_corpo	string	Corpo procurado
textoEmail	string	Armazena o e-mail lido
assinatura	string	Define o corpo HTML de um e-mail

Tabela 9 - Dicionário de dados da classe Outlook

5. Tecnologias e Softwares utilizados

Neste capítulo são referenciadas as tecnologias que foram usadas para o desenvolvimento do projeto, bem como o software utilizado para a sua realização.

5.1. Tecnologias

C# (C Sharp)

O C# (C Sharp) é uma linguagem de programação desenvolvida pela Microsoft, orientada a objetos, fazendo parte da plataforma .NET. Baseada no C++, contém muitos elementos da linguagem Pascal e Java. Embora existam várias outras linguagens que suportam esta tecnologia (VB.NET, C++, J#), o C# (C Sharp) foi criado para facilitar o desenvolvimento e compreensão da plataforma .NET.[10]

Na listagem seguinte (Listagem 1) mostra um pequeno exemplo de código C# (C Sharp).

Listagem 1 – Exemplo de código C#

```
using System;

namespace Exemplo
{
  class OlaMundo
  {
    static void Main()
    {
      Console.WriteLine("Olá Mundo");
    }
  }
}
```

SQL Server

O SQL (Structured Query Language ou Linguagem de Consulta Estruturada), é uma linguagem de pesquisa declarativa para base de dados relacionais.[11] Para o desenvolvimento do projeto foi usado o Microsoft SQL Server por ser uma base de dados robusta que vem incluída no Microsoft Visual Studio (software utilizado na elaboração do projeto) e pelo facto da empresa utilizar na criação das suas bases de dados que utiliza no software GPAC.

Na listagem seguinte (Listagem 2) mostra um pequeno exemplo de código SQL.

Listagem 2 – Exemplo de código SQL

```
SELECT A.CODIGO, A.DESCRICAO, B.DESCRICAO
FROM PRODUTOS A JOIN COMPONENTES B
ON (A.CODIGO = B.CODPRODUTO)
WHERE A.CATEGORIA = 1 OR A.CATEGORIA = 2
ORDER BY A.CATEGORIA, A.DESCRICAO
```

XML (eXtensible Markup Language)

O XML (eXtensible Markup Language), é uma linguagem utilizada para a criação de documentos com dados organizados hierarquicamente, como textos, bancos de dados e desenhos vetoriais. O XML tem como conceito a padronização de uma sequência de dados com o objetivo de organizar, separar o conteúdo e integrá-lo com outras linguagens. A grande característica do XML é a sua portabilidade, usando um arquivo XML dos conteúdos de uma base de dados pode ser lido facilmente noutra base de dados.[12]

Na listagem seguinte (Listagem 3) mostra um pequeno exemplo de código XML.

Listagem 3 – Exemplo de código XML

```
<?xml version="1.0">
<filmes>
    <filme id="1">
```

```
            <titulo>O XML veste prada</titulo>
              <resumo>O filme mostra a elegância da XML na
representação de dados estruturados e semi estruturados.</resumo>
            <genero>Aventura</genero>
            <genero>Documentário</genero>
            <elenco>
                <ator>Ator 1</ator>
                <ator>Ator 2</ator>
                <ator>Ator 3</ator>
            </elenco>
        </filme>
    </filmes>
```

5.2. Softwares

Microsoft Visual Studio 2019

O Microsoft Visual Studio é um programa de computador em ambiente de desenvolvimento integrado (IDE) da Microsoft para o desenvolvimento de softwares especialmente dedicado ao .NET Framework e às linguagens Visual Basic (VB), C, C++, C#. Também é um programa de desenvolvimento na área web, usando a plataforma do ASP.NET, como websites, aplicativos web, serviços web e aplicativos móveis.[13]

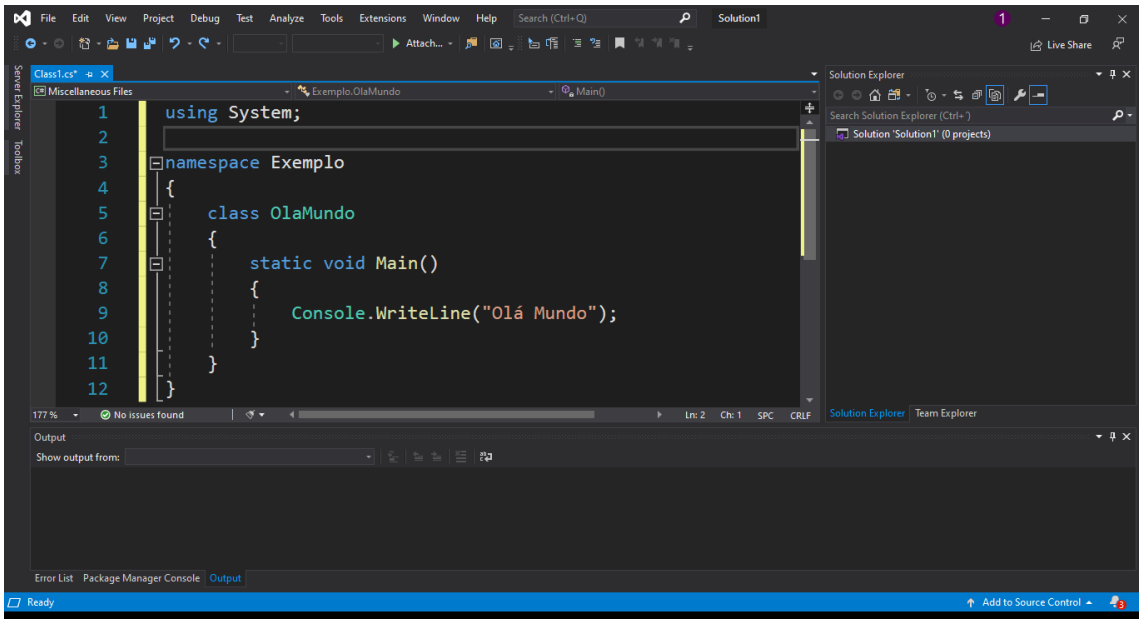

Figura 8 - Ambiente do Microsoft Visual Studio 2019

SQL Server Management Studio (SMSS)

O SQL Server Management Studio (SMSS) é uma ferramenta de interface gráfica do utilizador (GUI) que serve para aceder, configurar, gerenciar e administrar todos os componentes no Microsoft SQL Server, utilizando editores de scripts e ferramentas gráficas que trabalham com os objetos e recursos do servidor. Um dos recursos principais do SMSS é o Pesquisador de Objetos, que permite ao utilizador navegar, selecionar e manipular sobre qualquer objeto no servidor. Também pode ser usado para criar uma nova base de dados, alterar uma já existente, adicionar ou modificar tabelas e índices, como também para analisar o desempenho da base de dados.[14]

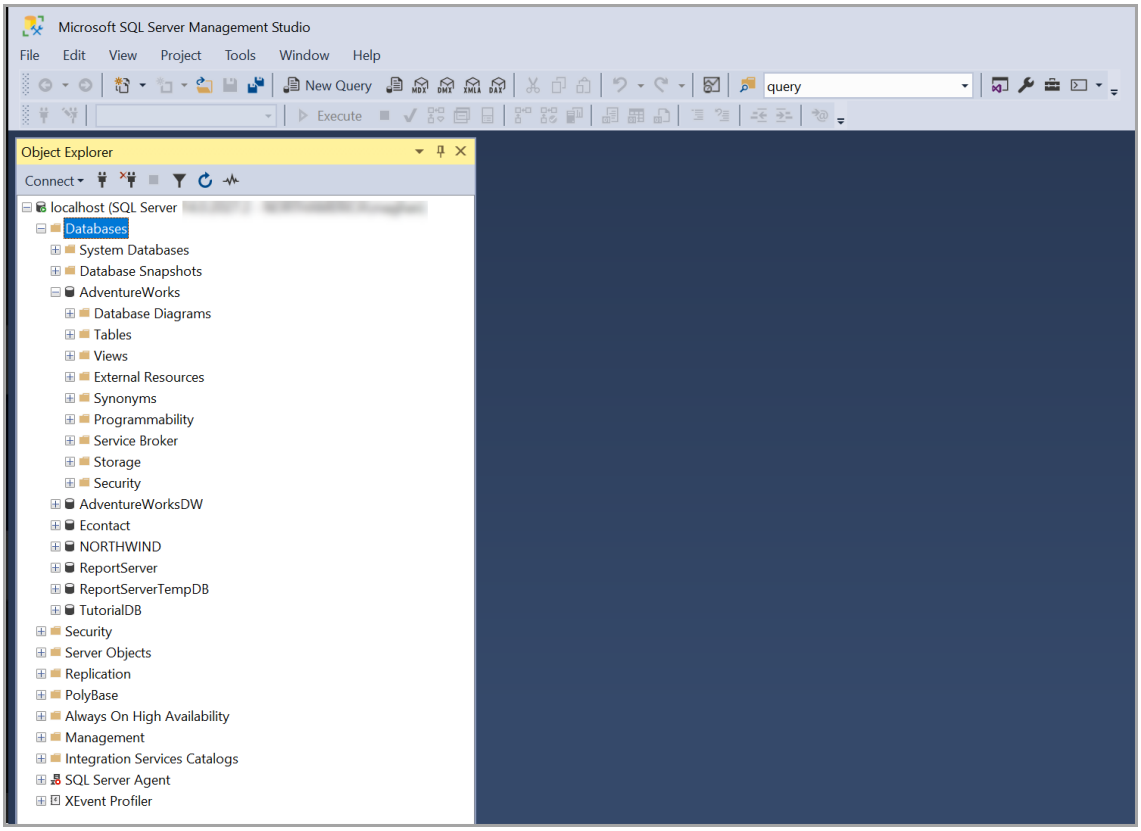

Figura 9 - Ambiente do SQL Server Management Studio

6. Implementação da Solução

Este capítulo apresenta a funcionalidade da aplicação desenvolvida através de alguns exemplos.

6.1. Início do *Ticket*

No desenvolvimento da aplicação a colaboração com o responsável pelo software GPAC foi importante para estar tudo interligado, tentando sempre procurar formas simples e eficazes para realizar o projeto. E tudo começa no software GPAC onde o cliente, quando reporta um erro ou até uma simples alteração, comunica através de um e-mail enviado no próprio GPAC.

O cliente através do atalho no teclado Shift + F1 , abre o pedido de assistência, como mostra a figura 10, para reportar o erro ou alterações desejadas para o e-mail suporte.

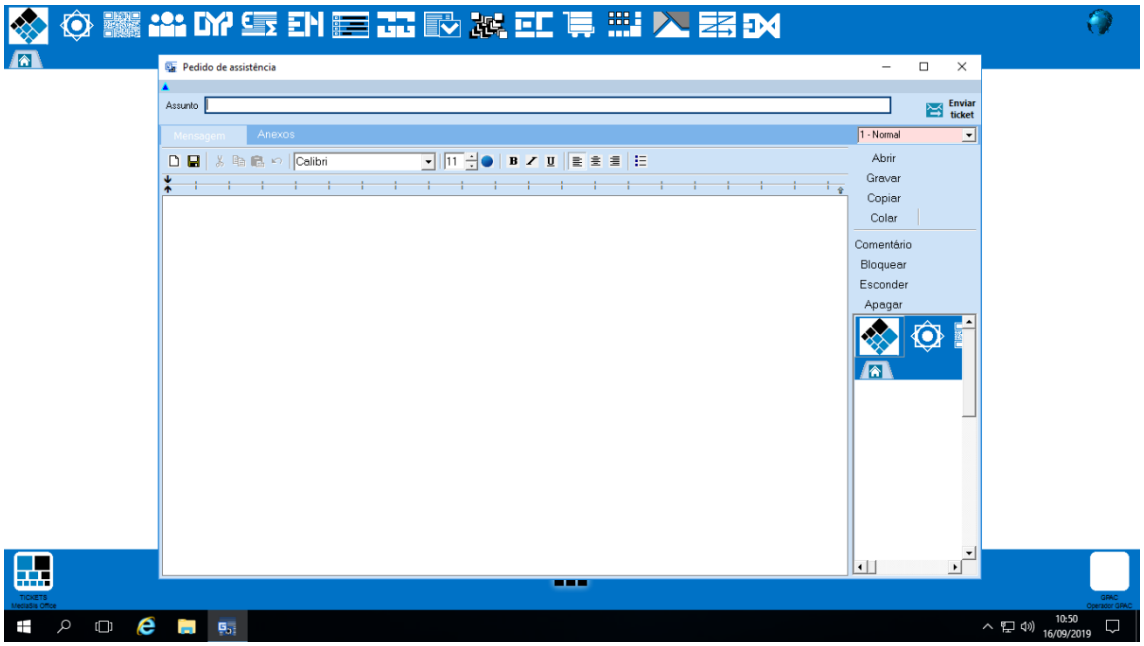

Figura 10 - Layout do pedido de assistência no GPAC

O cliente terá de preencher o campo assunto e o corpo da mensagem para informar o técnico do erro ou alteração desejada. Como normalmente os erros ocorridos dão informações no programa o cliente tem a possibilidade de tirar printscreen e ao abrir o assistente automaticamente é anexado o printscreen tirado pelo cliente, como se pode observar na figura 11.

Ainda é possível para o cliente colocar um grau de prioridade no ticket, dividido em quatro estados:

1-Normal;

2-Sugestão;

3-Urgente;

4-Impeditivo.

Estes estados servem para o responsável por cada cliente ter uma ideia da prioridade das tarefas a dar aos técnicos. Importa referir que dado o facto de a empresa ainda ter poucos colaboradores por vezes os técnicos coincidem com o responsável pelo cliente.

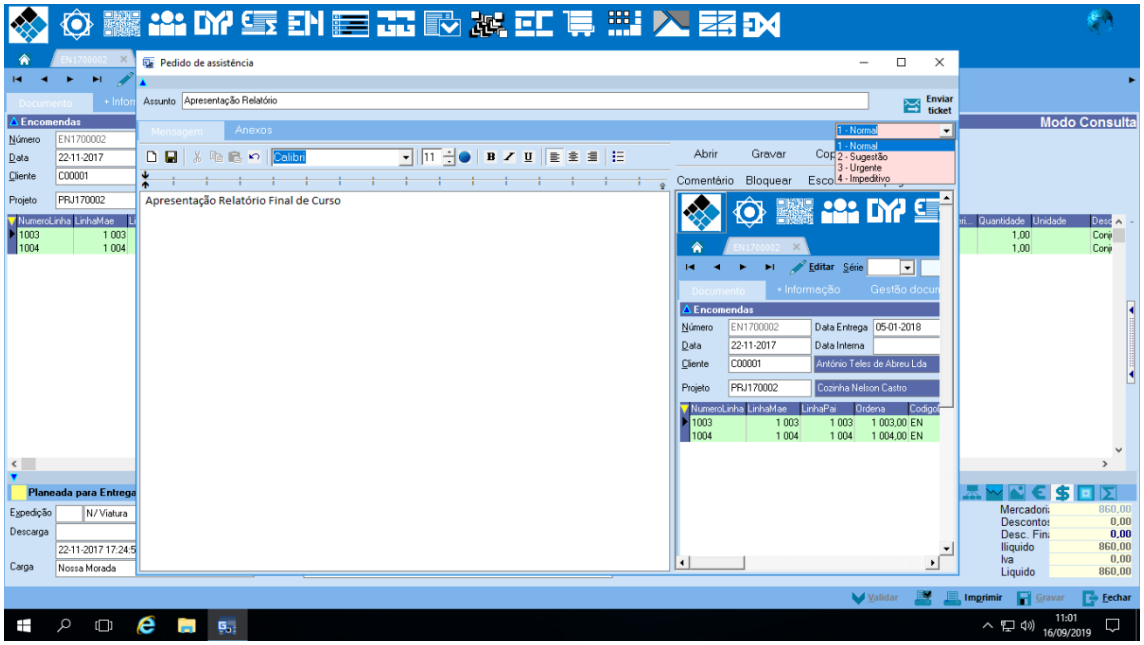

Figura 11 - Envio de E-mail através do GPAC

No mesmo pedido de assistência também é possível anexar alguns ficheiros, como mostra na figura 12. Normalmente os pedidos de alterações dos clientes são de logotipos ou de dados que querem alterar nos artigos da empresa, enviando ficheiros Excel e imagens para os técnicos alterarem no GPAC.

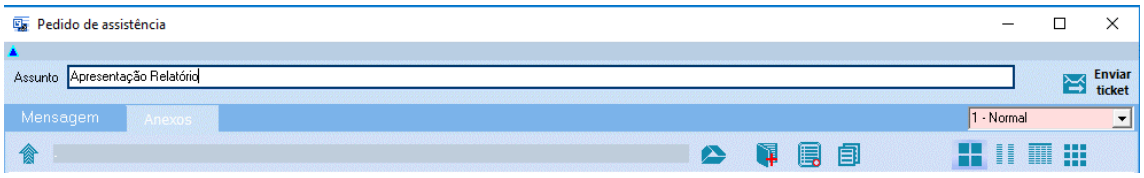

Figura 12 - Secção para anexar ficheiros externos

6.2. Sincronização dos E-mails

A sincronização dos e-mails fica a cargo da secretária da empresa que ao longo do dia verifica o e-mail da empresa de suporte e quando tem e-mails novos executa a aplicação que faz a sincronização com o GPAC. A figura 13 mostra o ícone da aplicação de sincronização MediaSisMail.

Figura 13 – Ícone da aplicação de sincronizaçãoMediaSisMail

Depois da execução da aplicação os resultados são mostrados em tabelas no GPAC, como representa a figura 14, onde a informação está guardada na Base Dados. A visualização dos tickets é feita num GPAC próprio para os tickets onde a Base Dados é específica para os mesmos. Contudo, cada cliente também tem a sua própria área de tickets e só visualiza os seus próprios tickets. A distinção é feita pelo código do cliente através de comandos SQL.

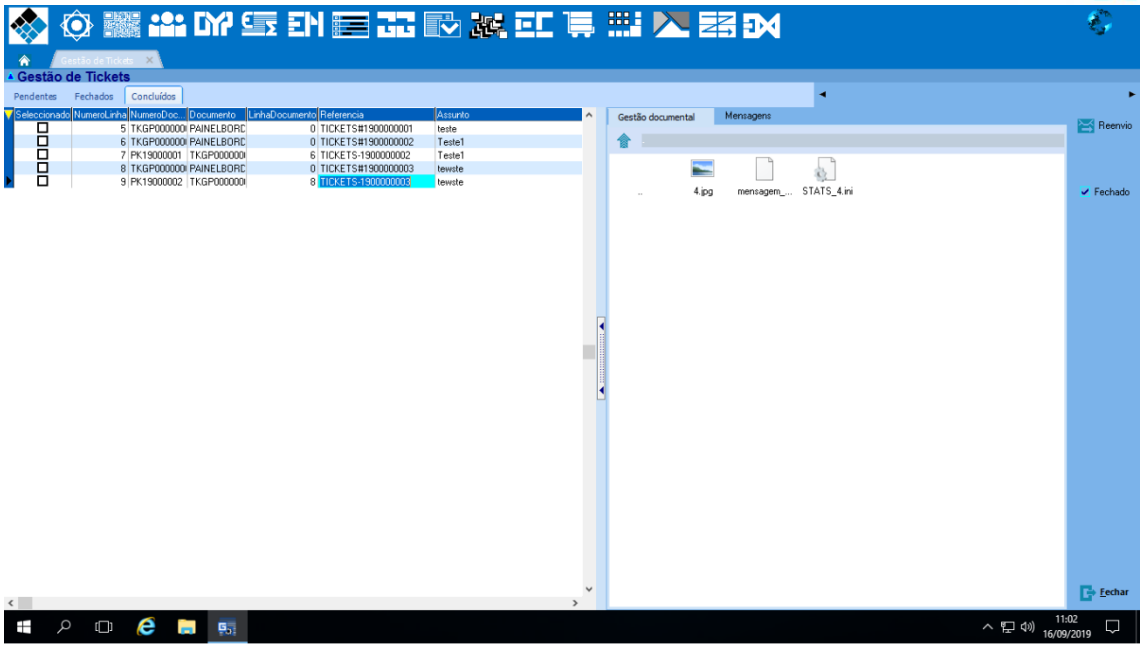

Figura 14 - Apresentação dos tickets

Os conteúdos enviados pelo cliente, imagens e outros ficheiros, são guardados na gestão documental que é uma tabela do GPAC para o técnico ter acesso mais facilitado aos anexos enviados pelo cliente.

Também é possível haver troca de mensagens com o operador que abriu o ticket do lado do cliente, como se pode ver na figura 15.

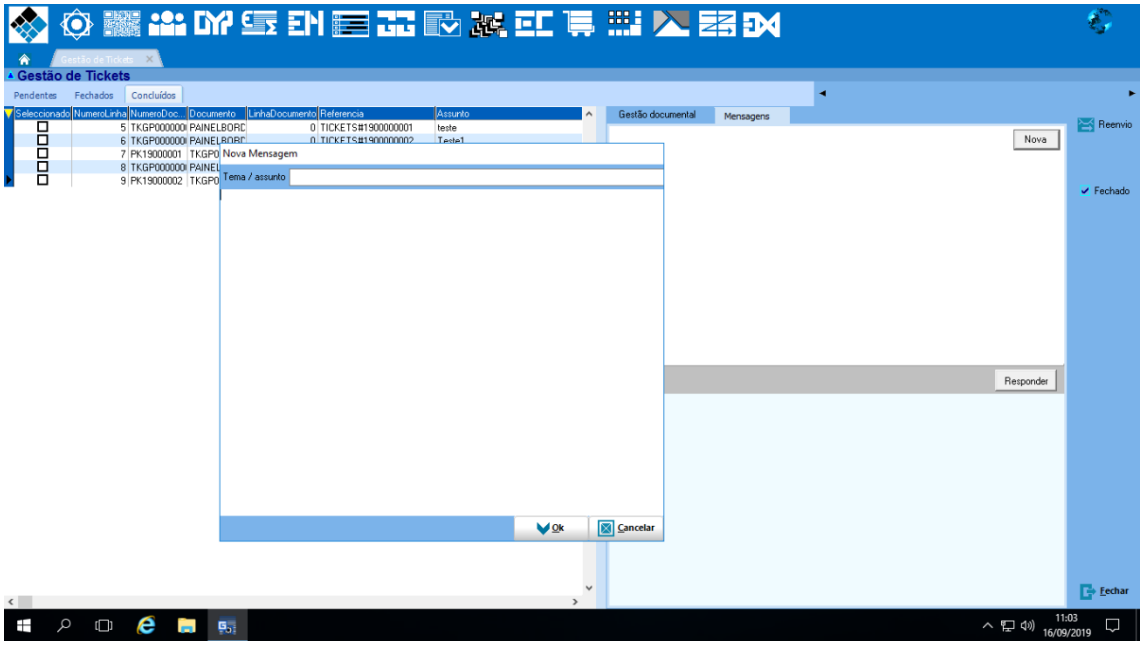

Figura 15 - Troca de mensagens

6.3. Finalização do Ticket

A finalização do ticket tem três fases:

Fase 1:

O ticket quando entra no sistema fica pendente à espera que o técnico o comece a resolver.

Fase 2:

Após a possível resolução do ticket, o técnico coloca o estado de fechado, ficando à espera que o cliente o valide.

Fase 3:

Com a validação do cliente o ticket pode passar novamente ao estado pendente, se o erro continuar a ocorrer ou a alteração proposta não foi exatamente a efetuada, mas se tudo correu como planeado o cliente tem de colocar o ticket como concluído, este estado só pode ser feito no lado do cliente.

7. Conclusão

No desenrolar deste projeto, foi desenvolvida uma aplicação de sincronização do e-mail suporte para o software GPAC.

No início do projeto foram delineados objetivos para a realização do trabalho. Foi feita alguma pesquisa sobre aplicações semelhantes, até analisando um trabalho de um colega, foi analisado que devido a uniformização pretendida e ao uso de uma base de dados externa, optou-se por fazer uma aplicação de raiz e assim fácil para personalizar a pedido da empresa.

Depois da análise de requisitos feita, o desenvolvimento da aplicação foi mais simples e fácil, devida à compreensão do problema.

Com a metodologia ágil utilizada, os erros e problemas que surgiram ao longo do projeto foram mais fáceis de corrigir devido aos testes e avaliações que se iam fazendo com os elementos envolvidos no projeto.

A aplicação foi desenvolvida para ser de fácil acesso e compreensão, para que a pessoa encarregada de fazer a sincronização dos e-mails a consiga fazer sem precisar de um supervisor do projeto. Com uma simples execução do aplicativo, os e-mails são sincronizados em tabelas no software GPAC, fazendo catalogação dos e-mails a partir do cabeçalho e do cliente de origem.

Os testes efetuados ao projeto foram bem sucedidos. Usando uma conta GPAC que a empresa tem para testes, foram enviados alguns pedidos. Com a execução da aplicação desenvolvida os e-mails enviados para o e-mail de suporte da empresa para teste foram sincronizados com o GPAC, possibilitando assim a visualização dos pedidos efetuados, tal como era pretendido.

Foi um projeto muito gratificante de realizar, permitiu ter uma experiência no mundo do trabalho e conhecer melhor a maneira de trabalhar e lidar com vários clientes que a empresa tem. Além disso foi aprofundada a aprendizagem de várias linguagens como por exemplo SQLServer e a utilização de vários programas como por exemplo SQL Server Management Studio.

Para o futuro, a empresa tem interesse em generalizar a aplicação, deixando de ser específica para o e-mail suporte da empresa, Outlook, podendo usar outros serviços

webmail. Também era interessante implementar tempos para ter uma forma de mostrar aos clientes o tempo que o colaborador demora a resolver o ticket e a atribuição de técnico para cada ticket.

Bibliografia

[1] LOURENÇO, Leonardo. *MestreClique GF-Tickets*. 2019. [Acedido: 2 de Fevereiro de 2020]

[2] "Metodologia de desenvolvimento de Software" [Online]. Em: https://www.devmedia.com.br/metodologia-de-desenvolvimento-de-software/1903. [Acedido: 5 de Dezembro de 2019]

[3] SILVEIRA, Maria Clara. *Processos Software*. Engenharia Software II. [Acedido: 5 de Dezembro de 2019]

[4] MACHADO, Felipe Nery Rodrigues. *Análise e Gestão de Requisitos de Software Onde nascem os sistemas*. Saraiva Educação S.A., 2018. [Acedido: 5 de Dezembro de 2019

[5] SOMMERVILLE, Ian. *Engenharia de Software / Ian Sommerville* ; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves ; revisão técnica Kechi Hirama. 9º ed. São Paulo : Pearson Prentice Hall, 2011. [Acedido: 16 de Dezembro de 2019]

[6] "UML" [Online]. Em: https://pt.wikipedia.org/wiki/UML. [Acedido: 16 de Dezembro de 2019

[7] "Caso de Uso" [Online]. Em: https://pt.wikipedia.org/wiki/Caso_de_uso. [Acedido: 16 de Dezembro de 2019]

[8] NUNES, Mauro; O'NEILL, Henrique. *Fundamental de UML*. 2º ed. FCA. [Acedido: 18 de Dezembro de 2019]

[9] "O que é um diagrama de classe UML?" [Online]. Em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml. [Acedido: 20 de Dezembro de 2019]

[10] "C# - C Sharp – Linguagem de Programação – InfoEscola" [Online]. Em: https://www.infoescola.com/informatica/c-sharp/. [Acedido: 10 de Janeiro de 2020]

[11] "SQL" [Online]. Em: https://pt.wikipedia.org/wiki/SQL. [Acedido: 10 de Janeiro de 2020]

[12] “O que é o XML?” [Online]. Em: https://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm. [Acedido: 10 de Janeiro de 2020]

[13] “Microsoft Visual Studio” [Online]. Em: https://pt.wikipedia.org/wiki/Microsoft_Visual_Studio

[14] “Microsoft SQL Server” [Online]. Em: https://pt.wikipedia.org/wiki/Microsoft_SQL_Server. [Acedido: 10 de Janeiro de 2020]

[15] “Microsoft.Office.Interop.Outlook Namespace” [Online]. Em: https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.outlook?view=outlook-pia. [Acedido: 08 de Agosto de 2019]

Anexos

No seguinte anexo apresenta algumas partes de código usado no desenvolvimento da aplicação, que me foi autorizado a mostrar.

Temos em primeiro o controlador que faz toda a ação da aplicação, onde vai pesquisar todas as informações que necessita para fazer sincronização com o GPAC.

Para o desenvolvimento do projeto foram usados os métodos da Microsoft com o Outlook, *Microsoft.Office.Interop.Outlook.*[15]

MediaSisControlar.cs

```
1.  using Microsoft.Office.Interop.Outlook;
2.  using System;
3.  using System.Collections.Generic;
4.  using System.Data;
5.  using System.Globalization;
6.  using System.IO;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Windows.Forms;
10.
11.
12. namespace MediaSisMail
13. {
14.     class MediaSisControler:ActiveRecord
15.     {
16.         private string _pasta;
17.         private string _caixa;
18.         private string _dir;
19.         private string _odbc;
20.         private List<string> findString;
21.         private Outlook obj;
22.         private Microsoft.Office.Interop.Outlook.MAPIFolder rootFolder;
23.         private List<Microsoft.Office.Interop.Outlook.Folder> iPastas;
24.         private List<Microsoft.Office.Interop.Outlook.MailItem> imail;
25.         private List<string> ficheiros;
26.         private List<string> ficheirosBody;
27.         private List<string> idMail;
28.         private string _conta;
29.         private string _tabela;
30.         private string _doc;
31.         private string _serie;
32.         private DataSet NSerie;
33.         private int NSerieN;
34.         private int NSerieT;
35.
36.
37.         /*################################################ PROPRIEDADES ###
    ##########################################################################*/
38.
39.         public string pasta{ get { return _pasta; } set { _pasta = value; }}
```

```
40.         public string caixa{ get { return _caixa; } set { _caixa = value; }}
41.         public string dir { get { return _dir; } set { _dir = value; } }
42.         public string codigodocumento { get { return _doc; } set { _doc = valu
    e; } }
43.         public string seriedocumento { get { return _serie; } set { _serie = v
    alue; } }
44.         public string odbc
45.         {
46.             get { return _odbc; }
47.             set { _odbc = value; conString = string.Format("DSN={0};Uid=******
    *;Pwd=******", odbc);
48.             }
49.         }
50.         public string Conta { get { return _conta; } set { _conta = value; } }

51.
52.         public string Tabela{get { return _tabela; } set{_tabela = value;}}
53.
54.         /*################################################## CONSTRUTOR #####
    ##########################################################################*/
55.         public MediaSisControler()
56.         {
57.             ficheiros = new List<string>();
58.             idMail = new List<string>();
59.             ficheirosBody = new List<string>();
60.             findString = new List<string>();
61.             obj = new Outlook();
62.         }
63.         /*################################################## FUNÇÕES PRIVADAS
     ##########################################################################*/

64.
65.         public void SetPasta()
66.         {
67.             iPastas = obj.GetPastas(rootFolder, _pasta);
68.         }
69.
70.
71.         private void ExportData()
72.         {
73.             if (ficheiros.Count > 0)
74.             {
75.
76.                 var csv = new StringBuilder();
77.                 foreach (Microsoft.Office.Interop.Outlook.MailItem mail in ima
    il)
78.                 {
79.                     try
80.                     {
81.                         string c4 = mail.UnRead.ToString();
82.                         string c1 = mail.EntryID;
83.                         string c2 = mail.SenderEmailAddress;
84.                         string c3 = mail.Subject;
85.
86.                         var newline = String.Format("{0};{1};{2};{3}", c1, c2,
     c3, c4);
87.                         csv.AppendLine(newline);
88.                     }
89.                     catch(System.Exception ex)
90.                     {
91.
92.                     }
93.
94.                     try
```

```
95.                     {
96.                         File.WriteAllText(ficheiros[0].ToString(), csv.ToStrin
    g());
97.                     }
98.                     catch (System.Exception ex)
99.                     {
100.                                DialogResult rs = MessageBox.Show(ex.Message, "
    Não foi possivel gerar ficheiro",MessageBoxButtons.OK, MessageBoxIcon.Error);

101.                            }
102.                        }
103.
104.                    }
105.                }
106.
107.                private void ImportData()
108.                {
109.                    foreach(string nome in ficheiros)
110.                    {
111.                        try
112.                        {
113.                          var reader = new StreamReader(File.OpenRead(nome));
114.                          while (!reader.EndOfStream)
115.                            {
116.                                var line = reader.ReadLine();
117.                                idMail.Add(line);
118.                            }
119.                        }
120.                        catch(System.Exception ex)
121.                        {
122.                            DialogResult rs = MessageBox.Show(ex.Message, "Não
    foi possivel encontrar o ficheiro", MessageBoxButtons.OK, MessageBoxIcon.Error
    );
123.                        }
124.
125.                    }
126.
127.                }
128.
129.                public string GetCaminhoEmpresa()
130.                {
131.                    string sql1 = "SELECT  Caminho,Codigo FROM dbo.Empresas";
132.
133.                    try
134.                    {
135.                        DataSet tbempresa = ExecuteQuery(sql1);
136.                        string sistema = (string)tbempresa.Tables[0].Rows[0].It
    emArray[0];
137.                        string empresa = (string)tbempresa.Tables[0].Rows[0].It
    emArray[1];
138.                        return sistema + "\\" + empresa + "\\";
139.                    }
140.                    catch (System.Exception ex)
141.                    {
142.                        DialogResult rs = MessageBox.Show(ex.Message, "Erro", M
    essageBoxButtons.OK, MessageBoxIcon.Information);
143.                        Console.WriteLine(ex.Message);
144.                        return null;
145.                    }
146.
147.
148.                }
149.
```

```
150.                 private string getDocumento(string vossaref, string cliente, st
    ring nossaref, string linha)
151.                 {
152.                     string sql1 = "SELECT CAST(L.NumeroLinha AS VARCHAR(100)) F
    ROM dbo.TKLDocs L" +
153.                                   " INNER JOIN TkDocs V ON (V.Numero=L.NumeroDocu
    mento)" +
154.                                   " WHERE V.Terceiro='" + cliente + "' AND V.Voss
    aRef='" + vossaref + "' and V.NossaRef='" + nossaref + "' AND L.LinhaDocumento
    =" + linha + "";
155.
156.                     try
157.                     {
158.                         DataSet tbempresa = ExecuteQuery(sql1);
159.                         List<string> linhas = new List<string>();
160.                         for (int i = 0; i < tbempresa.Tables[0].Rows.Count; i++
    )
161.                         {
162.                             linhas.Add((string)tbempresa.Tables[0].Rows[i].Item
    Array[0]);
163.                         }
164.
165.                         return string.Join(",", linhas);
166.                     }
167.                     catch (System.Exception ex)
168.                     {
169.                         Console.WriteLine(ex.Message);
170.                         return null;
171.                     }
172.                 }
173.
174.                 private void RegTicket()
175.                 {
176.                     string ndocumento = null;
177.                     List<Microsoft.Office.Interop.Outlook.MailItem> SortedList
    = imail.OrderBy(o => o.ReceivedTime).ToList();
178.                     foreach (Microsoft.Office.Interop.Outlook.MailItem it in So
    rtedList)
179.                     {
180.                         string[] assunto = it.Subject.Split(']');
181.                         string[] tipo = assunto[0].Split('#');
182.                         if (tipo.Length == 8)
183.                         {
184.                             string vossaRef = tipo[0].Replace("[", string.Empty
    ) + "-" + tipo[1].Replace("[", string.Empty);
185.                             assunto[0] = assunto[0].Replace("[", string.Empty);

186.                             tipo[0] = tipo[0].Replace("[", string.Empty);
187.
188.                             switch (tipo[4].ToUpper())
189.                             {
190.                                 case "CONCLUÍDO":
191.                                     ndocumento = getDocumento(vossaRef, tipo[0]
    , tipo[2], tipo[3]);
192.                                     Tik_concluido(ndocumento, it.ReceivedTime.T
    oString("yyyyMMdd HH:mm:ss"), it);
193.                                     break;
194.                                 case "FECHADO":
195.                                     ndocumento = getDocumento(vossaRef, tipo[0]
    , tipo[2], tipo[3]);
196.                                     Tik_fechado(ndocumento, it.ReceivedTime.ToS
    tring("yyyyMMdd HH:mm:ss"), it);
197.                                     break;
198.                                 case "NOVO":
```

```
199.                                     Tik_aberto(vossaRef, tipo[0], tipo[2], tipo
    [3], assunto[1], tipo[6], it.ReceivedTime.ToString("yyyyMMdd HH:mm:ss"), it, _
    dir);
200.                                     break;
201.                                 case "NOVA MENSAGEM":

202.                                     Tik_NovaMensagem(vossaRef, tipo[0], tipo[2]
    , tipo[3],tipo[7], assunto[1],it);
203.                                     break;
204.                                 default:
205.                                     break;
206.
207.                             }
208.                         }
209.                     }
210.                 }
211.
212.                 private void Tik_NovaMensagem(string numb, string cliente, stri
    ng Documento, string linhanumero, string orig, string assunto, Microsoft.Offic
    e.Interop.Outlook.MailItem item)
213.                 {
214.                     string ndocumento = getDocumento(numb, cliente, Documento,
    linhanumero);
215.
216.                     if (ndocumento == "" || ndocumento == null)
217.                         return;
218.
219.                     string[] ndocumentoIndividual = ndocumento.Split(',');
220.                     string[] linhaOrigem = orig.Replace("M:","").Replace("P:","
    ").Split('|');
221.                     string[] cabecAssunto = assunto.Replace(">", "").Split('<')
    ;
222.                     string body = "";
223.                     if (item.Body != null)
224.                     {
225.                         body = item.Body.Replace("'",string.Empty);
226.                     }
227.                     string[] vbody = body.Split(new string[] { "<INICIO_MENSAGE
    M>" }, StringSplitOptions.None);
228.                     string[] fbody = vbody[1].Split(new string[] { "<FIM_MENSAG
    EM>" }, StringSplitOptions.None);
229.                     string vdata = vbody[0].Substring(33, 10) + " " + vbody[0].
    Substring(17, 8);
230.
231.
232.
233.
234.                     foreach (string numerolinha in ndocumentoIndividual)
235.                     {
236.
237.                         try
238.                         {
239.                             string sql = "SELECT L.* FROM dbo.LinColabora L" +"
    \n"+
240.                                          "INNER JOIN dbo.TkLDocs C ON (C.NumeroDo
    cumento=L.Documento AND C.NumeroLinha=L.LinhaDocumento) " +"\n"+
241.                                          "WHERE C.NumeroLinha=" + numerolinha + "
    \n" +
242.                                          "AND L.NumSeqOrigem="+linhaOrigem[1]+"";

243.
244.                             DataSet tbempresa = ExecuteQuery(sql);
245.                             if(tbempresa.Tables[0].Rows.Count==0)
246.                             {
```

```
247.                                 sql = "INSERT INTO dbo.LinColabora(Documento,Li
    nhaDocumento,LinhaMae,TipoColaborador,Colaborador,Descricao,Memo,Data,HoraInic
    ial,HoraFinal,Tipo,NumSeqOrigem,LinhaMaeOrigem)" + "\n" +
248.                                     "SELECT TOP 1 C.NumeroDocumento,C.NumeroLin
    ha,0 LinhaMae,'GP' TipoColaborador,'"+cabecAssunto[1]+"' Colaborador,'" + cabe
    cAssunto[0] + "' Descricao,CAST('"+fbody[0]+"' AS VARCHAR(MAX)) Memo," + "\n"
    +
249.                                     "CONVERT(VARCHAR(8),CAST('"+vdata+ "' AS DA
    TETIME),112),CAST('" + vdata + "' AS DATETIME),CAST('" + vdata + "' AS DATETIM
    E),'MSG'," + linhaOrigem[0]+","+linhaOrigem[1]+" \n" +
250.                                     "FROM dbo.TkLDocs C" + "\n" +
251.                                      "WHERE C.NumeroLinha=" + numerolinha + "";

252.
253.                                 ExecuteNonQuery(sql);
254.                             }
255.                             else
256.                             {
257.                                 sql = "INSERT INTO dbo.LinColabora(Documento,Li
    nhaDocumento,LinhaMae,TipoColaborador,Colaborador,Descricao,Memo,Data,HoraInic
    ial,HoraFinal,Tipo,NumSeqOrigem,LinhaMaeOrigem)" + "\n" +
258.                                     "SELECT TOP 1 C.NumeroDocumento,C.NumeroLin
    ha,L.NumSeq LinhaMae,'GP' TipoColaborador,'"+cabecAssunto[1]+"' Colaborador,'"
     + cabecAssunto[0] + "' Descricao,CAST('" + fbody[0] + "' AS VARCHAR(MAX)) Mem
    o," + "\n" +
259.                                     "CONVERT(VARCHAR(8),CAST('" + vdata + "' AS
     DATETIME),112),CAST('" + vdata + "' AS DATETIME),CAST('" + vdata + "' AS DATE
    TIME),'MSG'," + linhaOrigem[0]+","+linhaOrigem[1]+"\n" +
260.                                     "FROM dbo.TkLDocs C" + "\n" +
261.                                     "INNER JOIN dbo.LinColabora L ON (L.Documen
    to=C.NumeroDocumento AND L.LinhaDocumento=C.NumeroLinha AND L.NumSeqOrigem="+l
    inhaOrigem[1]+") \n"+
262.                                      "WHERE C.NumeroLinha=" + numerolinha + "";

263.
264.                                 ExecuteNonQuery(sql);
265.                             }
266.
267.                             item.UnRead = false;
268.
269.
270.
271.                         }
272.                         catch (System.Exception ex)
273.                         {
274.                             DialogResult rs = MessageBox.Show(ex.Message, "Erro
    ", MessageBoxButtons.OK, MessageBoxIcon.Information);
275.                             return;
276.                         }
277.
278.                     }
279.
280.                 }
281.         /*##################################################### FUNÇÕES PUBLICAS
    ##############################################################################*/

282.
283.                 public void SetCaixa()
284.                 {
285.                     rootFolder = obj.DefineCaixaCorreio(_caixa);
286.                 }
287.
288.                 public void SetFindString(string s)
289.                 {
```

```
290.                     string[] xs = s.Split(';');
291.                     foreach(string str in xs)
292.                     {
293.                         findString.Add(str.Trim('"'));
294.                     }
295.                 }
296.
297.                 public void SetFicheiro(string s)
298.                 {
299.                     string[] xs = s.Split(';');
300.                     foreach (string str in xs)
301.                     {
302.                         ficheiros.Add(str.Trim('"'));
303.                     }
304.                 }
305.
306.                 public void SetBodyFile(string s)
307.                 {
308.                     string[] xs = s.Split(';');
309.                     foreach (string str in xs)
310.                     {
311.                         ficheirosBody.Add(str.Trim('"'));
312.                     }
313.                 }
314.
315.
316.                 public void SetTitulo(string s)
317.                 {
318.                     obj.Titulo = s;
319.                 }
320.
321.                 public void SetTo(string s)
322.                 {
323.                     obj.To = s;
324.                 }
325.
326.                 public void SetCC(string s)
327.                 {
328.                     obj.Cc = s;
329.                 }
330.
331.                 public void SetBCC(string s)
332.                 {
333.                     obj.Bcc = s;
334.                 }
335.
336.
337.                 public void FindByName()
338.                 {
339.                     imail = obj.FindBy(findString, iPastas);
340.                 }
341.
342.                 public void GetData()
343.                 {
344.                     SetPasta();
345.                     FindByName();
346.                     ExportData();
347.                     Console.Write(iPastas.Count.ToString() + " pasta(s) encontr
    ada(s) e " + imail.Count.ToString() + " email(s) encontrado(s)");
348.                 }
349.
350.                 public void SaveData()
351.                 {
352.                     string[] caminho = ficheiros[0].Split('\\');
```

```
353.                    string pasta="";
354.                    string aux="";
355.                    for (int i=0;i<caminho.Length-1;i++)
356.                    {
357.                        aux = string.Copy(pasta);
358.                        pasta = string.Copy(aux + caminho[i] + "\\");
359.                    }
360.
361.                    DirectoryInfo d = new DirectoryInfo(pasta);
362.                    FileInfo[] fil = d.GetFiles("*.msg");
363.
364.                    foreach(FileInfo f in fil)
365.                    {
366.                        f.Delete();
367.                    }
368.
369.
370.                    SetCaixa();
371.                    ImportData();
372.                    obj.SaveMailMsg(idMail,pasta);
373.                }
374.
375.                public void SaveEmail()
376.                {
377.                    obj.SaveMail(ficheiros, ficheirosBody);
378.                }
379.
380.                public void Ticket()
381.                {
382.                    SetPasta();
383.
384.                    imail = new List<Microsoft.Office.Interop.Outlook.MailItem>
    ();
385.                    foreach (string s in findString)
386.                    {
387.                        obj.FindByAccount(s,iPastas,imail);
388.                    }
389.
390.                    RegTicket();
391.
392.                }
393.
394.                  public void SetConta()
395.                {
396.                    obj.SetConta(_conta);
397.                }
398.
399.                public void SetCaminhoEmpresa()
400.                {
401.                    dir = GetCaminhoEmpresa();
402.                }
403.
404.            }
405.        }
```

De seguida temos a implementação da classe Base Dados, que tem como nome ActiveRecord que faz as ligações as bases de dados tanto do cliente como a da empresa.

```
1.  //------------------------------------------------------------------------------
    --
2.  // <auto-generated>
3.  //     This code was generated by a tool
4.  //     Changes to this file will be lost if the code is regenerated.
5.  // </auto-generated>
6.  //------------------------------------------------------------------------------
    --
7.  namespace MediaSisMail
8.  {
9.      using System;
10.     using System.Data;
11.     using System.Data.SqlClient;
12.     using System.Configuration;
13.     using System.Data.Odbc;
14.
15.     public abstract class ActiveRecord
16.     {
17.
18.         protected int myId;
19.         protected static string MyconString;
20.         public int id { get { return myId; } }
21.
22.
23.
24.         public static string conString { get { return MyconString; } set { Myc
    onString = value; } }
25.         private static OdbcTransaction myTransact;
26.         private static OdbcConnection myConect;
27.         protected static OdbcTransaction CurrentTransaction { get { return myT
    ransact; } }
28.
29.
30.         private static void DisplaySqlErrors(OdbcException ex)
31.         {
32.             for (int i = 0; i < ex.Errors.Count; i++)
33.             {
34.                 //escrever no ficheiro;
35.             }
36.         }
37.
38.         protected static OdbcConnection GetConnection(bool open)
39.         {
40.             try
41.             {
42.
43.                 OdbcConnection cnx = new OdbcConnection(MyconString);
44.                 if (open)
45.                     cnx.Open();
46.                 return cnx;
47.             }
48.             catch (Exception ex)
49.             {
50.                 if (ex is OdbcException)
51.                 {
52.                     DisplaySqlErrors((OdbcException)ex);
53.                 }
54.
55.                 throw new ApplicationException(ex.Message, ex);
56.             }
57.         }
58.
59.         protected static DataSet ExecuteQuery(OdbcConnection cnx, string sql)
```

```
60.         {
61.             try
62.             {
63.                 OdbcDataAdapter da = new OdbcDataAdapter(sql, cnx);
64.                 DataSet ds = new DataSet();
65.                 da.Fill(ds);
66.                 return ds;
67.             }
68.             catch (Exception ex)
69.             {
70.                 if (ex is OdbcException)
71.                 {
72.                     DisplaySqlErrors((OdbcException)ex);
73.                 }
74.
75.                 throw new ApplicationException(ex.Message, ex);
76.             }
77.         }
78.
79.         protected static int ExecuteNonQuery(OdbcConnection cnx, string sql)
80.         {
81.             try
82.             {
83.                 OdbcCommand cmd = new OdbcCommand(sql, cnx);
84.                 return cmd.ExecuteNonQuery();
85.             }
86.             catch (Exception ex)
87.             {
88.                 if (ex is OdbcException)
89.                 {
90.                     DisplaySqlErrors((OdbcException)ex);
91.                 }
92.
93.                 throw new ApplicationException(ex.Message, ex);
94.             }
95.         }
96.
97.         protected static int ExecuteNonQuery(OdbcTransaction tx, string sql)
98.         {
99.             try
100.                {
101.                    OdbcCommand cmd = new OdbcCommand(sql, tx.Connection, t
    x);
102.                    return cmd.ExecuteNonQuery();
103.                }
104.                catch (Exception ex)
105.                {
106.                    if (ex is OdbcException)
107.                    {
108.                        DisplaySqlErrors((OdbcException)ex);
109.                    }
110.
111.                    throw new ApplicationException(ex.Message, ex);
112.                }
113.            }
114.
115.            protected static int ExecuteNonQuery(OdbcTransaction tx, OdbcCo
    mmand cmd)
116.            {
117.                try
118.                {
119.                    cmd.Transaction = tx;
120.                    return cmd.ExecuteNonQuery();
121.                }
```

```
122.                     catch (Exception ex)
123.                     {
124.                         if (ex is OdbcException)
125.                         {
126.                             DisplaySqlErrors((OdbcException)ex);
127.                         }
128.
129.                         throw new ApplicationException(ex.Message, ex);
130.                     }
131.                 }
132.
133.                 protected static DataSet ExecuteQuery(string sql)
134.                 {
135.                     try
136.                     {
137.                         if (myConect == null)
138.                             myConect = GetConnection(false);
139.
140.                         OdbcDataAdapter da = new OdbcDataAdapter(sql, myConect)
    ;
141.                         DataSet ds = new DataSet();
142.                         da.Fill(ds);
143.                         return ds;
144.                     }
145.                     catch (Exception ex)
146.                     {
147.                         if (ex is OdbcException)
148.                         {
149.                             DisplaySqlErrors((OdbcException)ex);
150.                         }
151.
152.                         throw new ApplicationException(ex.Message, ex);
153.                     }
154.                 }
155.
156.                 protected static DataSet ExecuteTransactedQuery(string sql)
157.                 {
158.                     try
159.                     {
160.                         if (myConect == null)
161.                             myConect = GetConnection(false);
162.
163.                         OdbcDataAdapter da = new OdbcDataAdapter(sql, myConect)
    ;
164.                         DataSet ds = new DataSet();
165.                         da.Fill(ds);
166.                         return ds;
167.                     }
168.                     catch (Exception ex)
169.                     {
170.                         if (ex is OdbcException)
171.                         {
172.                             DisplaySqlErrors((OdbcException)ex);
173.                         }
174.
175.                         throw new ApplicationException(ex.Message, ex);
176.                     }
177.                 }
178.
179.                 protected static int ExecuteNonQuery(string sql)
180.                 {
181.                     try
182.                     {
183.
```

```
184.                     OdbcConnection cnx = GetConnection(true);
185.                     int r = ExecuteNonQuery(cnx, sql);
186.                     cnx.Close();
187.                     return r;
188.                 }
189.                 catch (Exception ex)
190.                 {
191.                     if (ex is OdbcException)
192.                     {
193.                         DisplaySqlErrors((OdbcException)ex);
194.                     }
195.
196.                     throw new ApplicationException(ex.Message, ex);
197.                 }
198.             }
199.
200.             protected static int ExecuteTransactedNonQuery(string sql)
201.             {
202.                 try
203.                 {
204.                     OdbcCommand cmd = new OdbcCommand(sql);
205.                     return ExecuteTransactedNonQuery(cmd);
206.
207.                 }
208.                 catch (Exception ex)
209.                 {
210.                     if (ex is OdbcException)
211.                     {
212.                         DisplaySqlErrors((OdbcException)ex);
213.                     }
214.
215.                     throw new ApplicationException(ex.Message, ex);
216.                 }
217.             }
218.
219.             protected static int ExecuteTransactedNonQuery(OdbcCommand cmd)

220.             {
221.                 try
222.                 {
223.                     cmd.Transaction = CurrentTransaction;
224.                     cmd.Connection = CurrentTransaction.Connection;
225.                     return cmd.ExecuteNonQuery();
226.                 }
227.                 catch (Exception ex)
228.                 {
229.                     if (ex is OdbcException)
230.                     {
231.                         DisplaySqlErrors((OdbcException)ex);
232.                     }
233.
234.                     throw new ApplicationException(ex.Message, ex);
235.                 }
236.             }
237.
238.
239.
240.             protected static void BeginTransaction()
241.             {
242.                 try
243.                 {
244.                     if (myTransact == null)
245.                         myTransact = GetConnection(true).BeginTransaction()
    ;
```

```
246.                     }
247.                     catch (Exception ex)
248.                     {
249.                         if (ex is OdbcException)
250.                         {
251.                             DisplaySqlErrors((OdbcException)ex);
252.                         }
253.
254.                         throw new ApplicationException(ex.Message, ex);
255.                     }
256.                 }
257.
258.                 protected static void CommitTransaction()
259.                 {
260.                     if (myTransact != null)
261.                     {
262.                         OdbcConnection cnx = myTransact.Connection;
263.                         try
264.                         {
265.                             myTransact.Commit();
266.                         }
267.                         finally
268.                         {
269.                             cnx.Close();
270.                             myTransact = null;
271.                         }
272.                     }
273.                 }
274.
275.                 protected static void RoolbackTransaction()
276.                 {
277.                     if (myTransact != null)
278.                     {
279.                         OdbcConnection cnx = myTransact.Connection;
280.                         try
281.                         {
282.                             myTransact.Rollback();
283.                         }
284.                         finally
285.                         {
286.                             cnx.Close();
287.                             myTransact = null;
288.                         }
289.                     }
290.                 }
291.
292.                 protected static int ExecuteTransactionScalar(string sql)
293.                 {
294.                     try
295.                     {
296.                         OdbcCommand cmd = new OdbcCommand(sql);
297.                         return ExecuteTransactedScalar(cmd);
298.                     }
299.                     catch (Exception ex)
300.                     {
301.                         if (ex is OdbcException)
302.                         {
303.                             DisplaySqlErrors((OdbcException)ex);
304.                         }
305.
306.                         throw new ApplicationException(ex.Message, ex);
307.                     }
308.                 }
309.
```

```
310.                protected static int ExecuteTransactedScalar(OdbcCommand cmd)
311.                {
312.                    try
313.                    {
314.                        cmd.Transaction = CurrentTransaction;
315.                        cmd.Connection = CurrentTransaction.Connection;
316.                        var result = cmd.ExecuteScalar();
317.                        if (result == null)
318.                            return 0;
319.                        else
320.                            if (result.ToString() == "")
321.                            return 0;
322.                        else
323.                            return Int32.Parse(result.ToString());
324.
325.                    }
326.                    catch (Exception ex)
327.                    {
328.                        if (ex is SqlException)
329.                        {
330.                            DisplaySqlErrors((OdbcException)ex);
331.                        }
332.
333.                        throw new ApplicationException(ex.Message, ex);
334.                    }
335.                }
336.
337.
338.
339.            }
340.        }
```

Por fim temos a classe Outlook, que tem como responsabilidade a leitura dos e-mails e a sua estruturação para depois ser enviado para o controlador.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using Microsoft.Office.Interop.Outlook;
6.  using System.Windows.Forms;
7.  using System.IO;
8.  using System.Xml;
9.  using System.Data.Odbc;
10. using System.Globalization;
11. using System.Data;
12.
13. namespace MediaSisMail
14. {
15.     class Outlook
16.     {
17.         private Microsoft.Office.Interop.Outlook.MailItem  oItem;
18.         private Microsoft.Office.Interop.Outlook.Application app;
19.         private Microsoft.Office.Interop.Outlook.NameSpace mapi;
20.         private Microsoft.Office.Interop.Outlook.Stores stores;
21.         private Microsoft.Office.Interop.Outlook.Account conta;
22.         private MAPIFolder rootFolder;
23.
24.
```

```
25.         private string _para;
26.         private string _titulo;
27.         private string _cc;
28.         private string _bcc;
29.         private string _corpo;
30.         private string textoEmail = "";
31.         private string assinatura;
32.
33.
34.         /*################################################## PROPRIEDADES ###
    ##########################################################################*/
35.         public string Titulo { get { return _titulo; } set { _titulo = value;
    } }
36.         public string Cc { get { return _cc; } set { _cc = value; } }
37.         public string To { get { return _para; } set { _para = value; } }
38.         public string Bcc { get { return _bcc; } set { _bcc = value; } }
39.         public string Corpo { get { return _corpo; } set { _corpo = value; } }

40.
41.
42.
43.         /*################################################## CONSTRUTOR #####
    ##########################################################################*/
44.
45.         public Outlook()
46.         {
47.             app = new Microsoft.Office.Interop.Outlook.Application();
48.             mapi=app.GetNamespace("MAPI");

49.             stores = app.Session.Stores;
50.             conta = null;
51.         }
52.
53.         /*################################################## FUNÇÕES PRIVADAS
     ##########################################################################*/

54.
55.         private void FindPastas(Folders fl,string nome,List<Folder> lista)
56.         {
57.             foreach(Folder f in fl)
58.             {
59.                 if(string.Equals(f.Name.ToUpper(),nome.ToUpper()))
60.                 {
61.                     lista.Add(f);
62.                 }
63.                 FindPastas(f.Folders, nome,lista);
64.             }
65.         }
66.
67.         private void PesquisaEmail(List<MailItem> litem,string conta,Folder f)

68.         {
69.             for (int i=1;i<=f.Items.Count;i++)
70.             {
71.                 try
72.                 {
73.                     MailItem item = (MailItem)f.Items[i];
74.                     if(item.UnRead)
75.                     {
76.                         string[] rec = item.To.Split(';');
77.                         foreach(string srec in rec)
78.                         {
79.                             if(string.Equals(srec.ToUpper(),conta.ToUpper()))
```

```
                                {
                                    litem.Add(item);
                                    break;
                                }
                            }

                        }

                    }
                    catch(System.Exception ex)
                    {
                        string x = ex.Message;
                    }
                }
            }

            /*############################################### FUNÇÕES PUBLICAS ##############################################################################*/

            public void SetConta(string nome)
            {
                    foreach (Account c in app.Session.Accounts)
                    {
                        if (string.Equals(c.SmtpAddress.ToUpper(), nome.ToUpper()))
                        { conta = c; break; }
                    }
                }

                public void SaveMail(List<string> ficheiro, List<string> body)

                {

                    oItem = app.CreateItem(0);

                    oItem.GetInspector.Activate();


                    //mailItem.GetInspector.Activate();
                    assinatura = oItem.HTMLBody;
                    oItem.Recipients.Add(To);
                    oItem.BCC = Bcc;
                    oItem.CC = Cc;
                    oItem.Subject = Titulo;

                    foreach (string s in body)
                    {
                        ReadFile(s);
                    }


                    string[] tbody = textoEmail.Split('\n');
                    oItem.HTMLBody = "";
                    for (int i = 0; i < tbody.Length; i++)
                    {
                        if (tbody[i] == "\r")
                            oItem.HTMLBody = string.Format("{0}<p/>{1}", oItem.HTMLBody, tbody[i]);
                        else
                            oItem.HTMLBody = string.Format("{0} {1}", oItem.HTMLBody, tbody[i]);
                    }

```

```
138.                     for (int i = 0; i < 5; i++)
139.                         oItem.HTMLBody = oItem.HTMLBody + "<p/>";
140.                     oItem.HTMLBody = oItem.HTMLBody + assinatura;
141.
142.
143.
144.
145.                     foreach (string s in ficheiro)
146.                     {
147.                         try
148.                         {
149.                             oItem.Attachments.Add(s);
150.                         }
151.                         catch (System.IO.FileNotFoundException ex)
152.                         {
153.                             DialogResult rs = MessageBox.Show("Ficheiro : " + s
    + " - " + ex.Message + " Este ficheiro não será adicionado", "", MessageBoxBu
    ttons.OK, MessageBoxIcon.Information);
154.                         }
155.
156.                     }
157.
158.                     oItem.Save();
159.                 }
160.
161.                 public MAPIFolder DefineCaixaCorreio(string nome)
162.                 {
163.
164.                     foreach (Microsoft.Office.Interop.Outlook.Store store in st
    ores)
165.                     {
166.                         string name = store.DisplayName;
167.                         if (string.Equals(nome.ToUpper(), name.ToUpper()))
168.                         {
169.                             rootFolder = store.GetRootFolder();
170.                             return rootFolder;
171.                         }
172.                     }
173.
174.                     return null;
175.                 }
176.
177.                 public List<Folder> GetPastas(MAPIFolder mFolder, string nome)

178.                 {
179.                     List<Microsoft.Office.Interop.Outlook.Folder> lista = new L
    ist<Microsoft.Office.Interop.Outlook.Folder>();
180.                     FindPastas(mFolder.Folders, nome, lista);
181.                     return lista;
182.
183.                 }
184.
185.                 public List<MailItem> FindBy(List<string> nomes, List<Folder> p
    astas)
186.                 {
187.                     List<MailItem> litem = new List<MailItem>();
188.                     foreach (string nome in nomes)
189.                     {
190.                         foreach (Folder f in pastas)
191.                         {
192.                              for(int i=1;i<=f.Items.Count;i++)
193.                              {
194.                                 try
195.                                 {
```

```
196.                                     Microsoft.Office.Interop.Outlook.MailItem i
    tem =(Microsoft.Office.Interop.Outlook.MailItem) f.Items[i];
197.                                     string[] dominio = item.SenderEmailAddress.
    Split('@');
198.
199.
200.                                     if (string.Equals(item.SenderEmailAddress.T
    oUpper(), nome.ToUpper()) || string.Equals(dominio[1].ToUpper(),nome.ToUpper()
    ))
201.                                     {
202.                                        litem.Add(item);
203.                                     }
204.                                 }
205.                                 catch (System.Exception ex)
206.                                 {
207.                                     string x = ex.Message;
208.                                 }
209.                             }
210.
211.                         }
212.                     }
213.
214.                     return litem;
215.
216.                 }
217.
218.                 public MailItem FindById(string id)
219.                 {
220.                     try
221.                     {
222.                         oItem = mapi.GetItemFromID(id, rootFolder.StoreID);
223.                         return oItem;
224.                     }
225.                     catch (System.Exception ex)
226.                     {
227.                         string x = ex.ToString();
228.                         return null;
229.                     }
230.
231.                 }
232.
233.                 public void FindByAccount(string conta, List<Folder> pasta, Lis
    t<MailItem> litem)
234.                 {
235.
236.                     foreach (Folder f in pasta)
237.                     {
238.                         PesquisaEmail(litem, conta, f);
239.                     }
240.                 }
241.
242.                 public void SavePst(MailItem mail,string path)
243.                 {
244.                     mail.SaveAs(path, Microsoft.Office.Interop.Outlook.OlSaveAs
    Type.olMSG);
245.                 }
246.
247.                 private void ReadFile(string path)
248.                 {
249.                     textoEmail = null;
250.                     try
251.                     {   // Open the text file using a stream reader.
```

```
252.                         FileStream fs = new FileStream(path, FileMode.Open, Fil
    eAccess.Read);
253.                         byte[] buffer;
254.                         buffer = new byte[fs.Length];
255.                         fs.Read(buffer, 0, (int)fs.Length);
256.                         Encoding seuEncoding = Encoding.GetEncoding("Windows-
    1252");
257.                         Encoding cp850 = Encoding.GetEncoding(850);
258.
259.
260.                         textoEmail = seuEncoding.GetString(buffer);
261.
262.
263.
264.                     }
265.                     catch (IOException ex)
266.                     {
267.                         DialogResult rs = MessageBox.Show("Ficheiro : " + path
    + " - " + ex.Message + " Ficheiro assinatura não encontrado", "", MessageBoxBu
    ttons.OK, MessageBoxIcon.Information);
268.                     }
269.                 }
270.
271.                 public void SaveMailMsg(List<string> idMail, string pasta)
272.                 {
273.                     int np = 1;
274.
275.                     foreach (string str in idMail)
276.                     {
277.                         string[] id = str.Split(';');
278.                         MailItem mail = null;
279.                         mail =FindById(id[0]);
280.                         if (mail != null)
281.                         {
282.                             string nome = pasta + mail.SenderEmailAddress.Repla
    ce("@", string.Empty).Replace(".", string.Empty).Replace(" ", string.Empty) +
    DateTime.UtcNow.ToString("yyyyMMddHHmmss") + "_" + np.ToString() + ".msg";
283.                             SavePst(mail, nome);
284.                             Console.Write("Ficheiro gerado: " + nome + "\n");
285.                             np++;
286.                         }
287.                     }
288.                 }
289.             }
290.         }
```