

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Departamento de Informática



Relatório de Projeto

Análise Alarmística e Integração com Aplicações terceiras

CLARANET

Realizado por

Marcelo Filipe Pinto Ribeiro Silva

Orientador ESTGV: Francisco Ferreira Francisco

Orientador Empresa: António Baptista

Viseu, 2022

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Departamento de Informática

Relatório de Projeto
Cursos de Licenciatura em
Engenharia Informática

Análise Alarmística e Integração com Aplicações terceiras

CLARANET

Realizado por
Marcelo Filipe Pinto Ribeiro Silva

Ano Letivo 2021/2022

Viseu, 2022

Agradecimentos

Qualquer realização humana, mesmo que individual, é sempre o resultado de um conjunto diversificado de contextos, de instituições e, especialmente, de pessoas. O alcançar desta etapa não teria sido possível sem a colaboração, apoio, carinho e dedicação por parte de várias entidades ao longo de todo o percurso da minha formação. Por esta mesma razão, não quero deixar passar a oportunidade de agradecer a todos aqueles que, direta ou indiretamente, contribuíram para o meu sucesso e a minha chegada até aqui. Assim, dedico-lhes especialmente este projeto.

Quero começar por agradecer aos meus orientadores, ao Professor Francisco Ferreira Francisco e ao Eng.º António Baptista, pelas diretrizes que me forneceram no desenvolvimento deste trabalho e por me terem permitido atingir este objetivo. Também quero prestar o meu reconhecimento à Escola Superior de Tecnologia e Gestão de Viseu por todas as condições que me criou durante esta Licenciatura. Por último, estou de igual modo, muito grato à minha Família que sempre me ajudou e suportou nesta etapa.

Resumo

Neste relatório irá ser descrito o trabalho realizado na entidade acolhedora Claranet, no âmbito da unidade curricular “Projeto” do 2.º Semestre do 3.º ano de Licenciatura de Engenharia Informática na Escola Superior de Tecnologia e Gestão de Viseu do Instituto Politécnico de Viseu.

Desta forma, na unidade curricular é proposto aos alunos que desenvolvam um projeto a nível empresarial, inserido no ambiente profissional da entidade acolhedora, de forma que os alunos adquiram conhecimento do ambiente de trabalho empresarial e possam por à prova os conhecimentos adquiridos ao longo da licenciatura. Este projeto está dividido em duas partes, primeiramente o desenvolvimento de um sistema de automação de alertas, isto é, uma aplicação que acede a uma base de dados e quando é ultrapassado um limite de presenças numa zona escreve uma mensagem, que é enviada por *e-mail* e por SMS para um destinatário. Essa mesma aplicação vai estar enquadrada num agendador de tarefas que de 5 em 5 minutos executa a aplicação. Na segunda, parte é um sistema de serviços, onde foi criado uma interface (API) que vai invocar um procedimento armazenado da base de dados com o intuito de ser possível fazer pedidos de informação por parte de uma outra plataforma, de forma a poder visualizar o valor desse procedimento armazenado em tempo real de uma forma envolvente e interativa no *Power BI*.

Índice

Agradecimentos	1
Resumo	2
Índice de figuras	6
Listas de acrónimos e abreviaturas.....	8
1. Introdução	9
1.1. Contextualização do projeto	9
1.2. Motivação e objetivos.....	10
1.3. Estrutura do relatório	11
2. Descrição das entidades	12
2.1. ESTGV	12
2.2. Entidade de acolhimento	13
3. Tecnologias e ferramentas	14
3.1. Linguagens/Ferramentas utilizadas	14
3.1.1. C#	14
3.1.2. .NET Framework	15
3.1.3. SMTP	15
3.1.4. Windows task scheduler	16
3.1.5. Visual Studio Code	17
3.1.6. CMD	17
3.1.7. Power BI.....	18

3.1.8. <i>SQL Server Management Studio</i>	18
3.1.9. <i>SQL</i>	19
3.1.10. <i>Twilio</i>	20
3.1.11. <i>Draw.io</i>	21
3.1.12. <i>API</i>	21
4. Desenvolvimento	22
4.1. <i>Calendarização</i>	22
4.2. <i>Modelação UML</i>	23
4.2.1. <i>Diagramas de Atividades</i>	23
4.3. <i>Análise funcional</i>	25
4.3.1. <i>Mensagem enviada</i>	25
4.3.2. <i>E-mail</i>	26
4.3.3. <i>SMS</i>	26
4.3.4. <i>Power BI</i>	27
4.3.5. <i>Power BI resultado</i>	29
4.3.1. <i>Power BI resultado com valores nulos</i>	29
5. Implementação	30
5.1. <i>Detalhe técnico e arquitetural</i>	30
5.1.1. <i>Base de dados NSA</i>	30
5.1.2. <i>Console application</i>	32
5.1.3. <i>Arquitetura aplicação e procedimento</i>	34

5.1.4. <i>Configuração de Email</i>	35
5.1.5. <i>Configuração do SMS</i>	37
5.1.6. <i>Windows task scheduler</i>	38
5.1.7. <i>API</i>	39
5.1.8. <i>Configuração do Power BI</i>	40
5.2. <i>Desafios</i>	41
6. Conclusão	42
6.1. Principais resultados e contribuições.....	42
6.2. Trabalho futuro	43
Referências Bibliográficas	44

Índice de figuras

Figura 4-1 Diagrama de Atividades-1	23
Figura 4-2 Diagrama de Atividades-2	24
Figura 4-3 Cmd.....	25
Figura 4-4 E-mail	26
Figura 4-5 SMS	26
Figura 4-6 Inserir parâmetros	27
Figura 4-7 Tipos de Procedimento	28
Figura 4-8 Resultado	29
Figura 4-9 Resultado com valores nulos	29
Figura 5-1 Stored procedure	31
Figura 5-2 Lista Stored procedure	31
Figura 5-3 Resposta Stored procedure.....	31
Figura 5-4 Connection String	32
Figura 5-5 ADO.NET	33
Figura 5-6 Arquitetura geral	34
Figura 5-7 Configuração de E-mail	36
Figura 5-8 Arquitetura Twilio	37
Figura 5-9 Windows task scheduler	38
Figura 5-10 Dataset	39
Figura 5-11 Parâmetros	40

Figura 5-12 URL base	40
Figura 5-13 URL base com parâmetros.....	40

Listas de acrónimos e abreviaturas

ESTGV Escola Superior de Tecnologia e Gestão de Viseu

IPV Instituto Politécnico de Viseu

EA Entidade acolhedora

SMTP *Simple Mail Transfer Protocol*

C# C Sharp

SMS Serviço de mensagens curtas

API Interface de programação de aplicações

ISP *Internet Service Provider*

SSMS SQL Server Management Studio

REST *Representational State Transfer*

JSON *Java Script Object Notation*

HTTP *Hypertext Transfer Protocol*

TLS *Transport Layer Security*

TCP Protocolo de controle de transmissão

AP *Access points*

1. Introdução

1.1. Contextualização do projeto

Este relatório de estágio foi realizado no âmbito da disciplina de Projeto, lecionada no segundo semestre, do terceiro ano, da licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão de Viseu, em colaboração com a empresa Claranet. Em termos de tutoria, o docente Francisco Ferreira Francisco foi o orientador da instituição escolar, já o da entidade acolhedora foi o Engenheiro António Baptista, pertencente à Claranet.

O projeto está dividido em duas partes, sendo que a primeira tem o objetivo de criação de uma *console application* para o envio automático de *reports* via e-mail e via SMS. É uma aplicação programada em *Schedule Task*, executada de 5 em 5 minutos ou outro intervalo de tempo, que executa um *stored procedure* da base de dados. Este tem o propósito de retornar uma mensagem quando é ultrapassado um limite de pessoas numa determinada zona. Relativamente à segunda parte, criou-se uma interface (API) em ASP.NET Core 5, que tem como finalidade executar um *stored procedure*. Dentro desse existem outros procedimentos, que inicialmente é preciso passar todos os parâmetros necessários para executá-lo e depois disponibilizar um pedido para mostrar o resultado na plataforma Power BI. Nesta é essencial criar um ficheiro que vai conter uma *query* na qual passamos todos os parâmetros e o *URL* do pedido que foi criado na API. Esta última parte é feita para criar uma abstração para que os pedidos sejam feitos no servidor e não diretamente na base de dados.

No decorrer deste relatório é descrito o projeto, as suas características, as suas etapas de execução, os passos de criação e de desenvolvimento, os objetivos atingidos e todos os obstáculos na sua realização. Neste também está presente uma breve descrição das ferramentas e tecnologias necessárias à concretização do mesmo e envolventes na área em que é realizado. No final, é apresentada uma conclusão, onde foi feita uma reflexão do projeto e do seu percurso ao longo do estágio, assim como uma análise aos objetivos alcançados e trabalho a fazer no futuro.

1.2. Motivação e objetivos

Quando um consultor ou um programador precisa de aceder a *reports* de uma empresa ou outro qualquer espaço e necessita da possibilidade de definir parâmetros de análise, por exemplo, quantidade de dispositivos detetados por local, a partir dos quais será gerado um *report*. Este tipo de *reports* é relevante para despoletar ações, como, mobilização de brigadas de limpeza, reforço policial enviar esses tais ficheiros por email para uma lista de pessoas.

Surgiu então a ideia e a necessidade de construir esta aplicação que poupa tempo e o esforço humano nas suas tarefas diárias. Este projeto visa substituir esse trabalho adicional por um de forma automática, decorrido em *background*.

Os objetivos para o desenvolvimento deste projeto incidem em: criar uma aplicação que esteja constantemente a funcionar. A aplicação invoca um procedimento armazenado e envia por *e-mail* e por SMS a resposta. Além disto, esta é executada de 5 em 5 minutos ou noutro intervalo de tempo dependendo da configuração. Um outro propósito é a integração com aplicações terceiras, ou seja, a implementação de uma camada de serviços, a API, para disponibilizar a informação a outros sistemas e fomentar a partilha de informação no *Power BI*.

1.3. Estrutura do relatório

Este relatório está organizado em sete capítulos.

O primeiro capítulo corresponde à Introdução, responsável por um enquadramento da cadeira, do projeto desenvolvido, as motivações e os objetivos do mesmo.

No que concerne ao segundo capítulo, nomeadamente, a Descrição das entidades, descreve-se o local de trabalho, assim como a Escola Superior de Tecnologia e Gestão de Viseu e a entidade de acolhimento, a Claranet.

Relativamente ao terceiro capítulo, Tecnologias e ferramentas, faz-se referência tanto as tecnologias subjacentes à elaboração do projeto, bem como à linguagem e às ferramentas utilizadas para o desenvolvimento deste.

No quarto capítulo, analisa-se o desenvolvimento do projeto, que descreve o modo de organização das tarefas de trabalho, o ponto inicial e a finalidade do próprio.

O quinto capítulo refere-se à implementação do projeto, que relata os detalhes técnicos da aplicação e os desafios passados na realização do mesmo.

Chegado ao sexto capítulo, a Conclusão, é apresentado uma reflexão sobre o trabalho desenvolvido e entidade acolhedora, uma análise do trabalho em geral, e também é posto em consideração o trabalho a realizar no futuro.

Por fim, no sétimo capítulo são apresentadas as Referências Bibliográficas consultadas para a realização do projeto e do relatório.

2. Descrição das entidades

2.1. ESTGV

Fundado no dia 26 de dezembro de 1979, o Instituto Politécnico de Viseu (IPV), é o primeiro e único estabelecimento de ensino superior público do distrito, assinalando assim um marco importante de desenvolvimento para a região de Viseu.

No total, o Instituto Politécnico de Viseu tem uma oferta formativa de 29 cursos de licenciatura, 29 mestrados, 13 pós-graduações, 6 pós-licenciaturas e 29 CTeSP, fazendo um total de 106 cursos e é composto por 5 Escolas Superiores.

A ESTGV foi criada em 1985, enquanto estabelecimento de ensino superior, é um centro de criação, difusão e transmissão de cultura, ciência e tecnologia, articulando as suas atividades nos domínios do ensino, da formação profissional, da investigação e da prestação de serviços à comunidade. A ESTGV rege-se por padrões de qualidade que asseguram formação adequada às necessidades da comunidade em que se insere. Cumpre formar, a nível superior, técnicos devidamente qualificados em setores carecidos na sua área de influência e promover atividades de investigação e desenvolvimento, visando a ligação entre o ensino superior e as empresas, como forma de contributo para solucionar os mais variados problemas.

2.2. Entidade de acolhimento

A Claranet, foi fundada em 1996, é especialista na modernização e gestão de aplicações críticas e infraestrutura 24x7. Com 26 anos de história e experiência, conta atualmente com uma equipa de mais de 700 profissionais, distribuídos por dois *data centers* e 4 escritórios em Portugal.

A Claranet inicialmente foi estabelecida em Londres como *Internet Service Provider* (ISP), pioneira em acesso pré-pago à Internet, depois entre 1997-2005 teve uma evolução para serviços geridos de Redes e *Hosting*, com expansão para a Europa, posteriormente entre 2006-2014 teve um aprofundamento do portfolio de serviços, com o reconhecimento internacional em Redes e *Cloud* e finalmente entre 2015-2021 expandiu-se para os EUA e América do Sul lançando o global da Claranet *Cyber Security*. A Claranet fornece serviços de rede, hospedagem e aplicativos gerenciados no Reino Unido, França, Alemanha, Holanda, Portugal, Espanha, Itália e Brasil.

3. Tecnologias e ferramentas

Neste capítulo irá ser abordado o estado de arte das tecnologias utilizadas. É realizada uma análise à arquitetura geral do projeto assim como as possíveis soluções que possam solucionar o problema apresentado.

Para finalizar irá ser explicado a escolha da solução a implementar recorrendo à análise de vantagens e desvantagens desta.

3.1. Linguagens/Ferramentas utilizadas

3.1.1. C#

C# ou *C Sharp* é uma linguagem de programação multiparadigma, de tipagem forte, desenvolvida pela Microsoft como parte da plataforma *.NET*. A sua sintaxe orientada a objetos foi baseada em *C++*, mas inclui muitas influências de outras linguagens de programação, como *Object Pascal* e, principalmente, *Java*. A linguagem *C#* destina-se a ser simples, moderna, de propósito geral. A linguagem e o compilador devem fornecer suporte para princípios de engenharia de *software*, tais como verificação de tipo forte, verificação dos limites do *array*, deteção de tentativas de usar variáveis não inicializadas e o *Garbage Collection* que liberta automaticamente o espaço de memória alocado após as ações relacionadas ao objeto classe são concluídas. A robustez do *software*, a durabilidade e a produtividade do programador são importantes.

Esta linguagem foi utilizada para o desenvolvimento da *console application*, sendo também usada para o desenvolvimento da API em *ASP.NET Core 5*.

3.1.2. .NET Framework

O *.NET Framework* é uma iniciativa da empresa Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para *.NET* pode ser executado em qualquer dispositivo que possua um *framework* de tal plataforma. Com ideia semelhante à plataforma *Java*, o programador deixa de escrever código para um sistema ou um dispositivo específico, e passa a escrever para a plataforma *.NET*. Permitindo também executar diversas linguagens criando interoperabilidade, ou seja, a capacidade de um sistema comunicar de forma transparente com outro sistema.

O *.NET* pode usar várias linguagens, *C#*, *F#*, ou *Visual Basic*, editores e bibliotecas para construir para *web*, *mobile*, *desktop*, jogos e *IoT (Internet of Things)*.

3.1.3. SMTP

SMTP (*Simple Mail Transfer Protocol*), ou seja, Protocolo de Transferência de Correio Simples, é o protocolo padrão de envio de mensagens de correio eletrônico através da Internet entre dois dispositivos computacionais (emissor e recetor), definido na RFC 821. É um protocolo simples, em texto plano, de somente de envio (semelhante a um carteiro), onde um ou vários destinatários de uma mensagem são especificados (e, na maioria dos casos, validados) sendo, depois, a mensagem transferida, por padrão via porta TCP 25 (ou 465 para conexão criptografada com SSL), podendo usar a porta alternativa 587. O funcionamento de um servidor SMTP pode ser dividido em duas etapas. A primeira inclui a concessão da permissão para o processo e a verificação da configuração do computador por meio do qual um *e-mail* é enviado. Na segunda, ele envia a mensagem e segue a entrega bem-sucedida do *e-mail*. Se, por algum motivo, o *e-mail* não for entregue, será devolvido ao remetente. O SMTP apenas tem a função somente de envio, isto é, não permite que um usuário descarregue/solicite as mensagens de um servidor. Algumas das vantagens são: um ambiente seguro para envio de *e-mails*, integração rápida e personalizável de *e-mail*, o *software* amigável e a análise em tempo real para controlar seus *e-mails*.

Foi usado o SMTP para que seja possível enviar a mensagem para o destinatário quando transcende o número limite de presenças.

3.1.4. Windows task scheduler

O *Task Scheduler* é um agendador de tarefas do Microsoft Windows que inicia programas do computador ou *scripts* em horários predefinidos ou após intervalos de tempo especificados. A Microsoft introduziu esta componente no *Microsoft Plus* para *Windows 95* como *System Agent*. A sua principal função é operar em *background* enquanto o *Windows* trabalha normalmente. A infraestrutura do *Task Scheduler* é a base para o recurso de trabalhos agendados do *Windows PowerShell* introduzido com o *PowerShell v3*. O *Task Scheduler* funciona gerenciando tarefas, entende-se tarefa, como a ação (ou ações) realizadas em resposta ao(s) gatilho(s) sendo uma definida pela associação de um conjunto de ações que pode incluir o lançamento de uma aplicação ou a execução de alguma ação personalizada, como um conjunto de gatilhos, que podem ser baseados em tempo ou em eventos.

Foi usado o *Task Scheduler* para que seja possível executar a *console application*.

3.1.5. Visual Studio Code

O *Visual Studio Code* é um editor de código desenvolvido pela Microsoft para *Windows*, *Linux* e *macOS*, foi lançado em 29 de abril de 2015. Este editor inclui um suporte de ligação a repositórios, como por exemplo o *Git*, complementação inteligente de código, realce de sintaxe, diversos *snippets*, e refatoração de código de diversas linguagens de programação. É uma ferramenta customizável para o utilizador, dado que permite editar atalhos, aspeto e preferências de ferramentas internas.

Esta ferramenta foi utilizada no meu projeto, para a produção e edição de código, usado também na linguagem *C#* para a criação da *console application* e da API.

3.1.6. CMD

Prompt de Comando (cmd.exe) é um interpretador de linhas de comando no OS/2 e de sistemas baseados no *Windows NT* (incluindo *Windows 2000*, *XP*, *Server 2003* e adiante até o mais recente *Windows 11*). É um comando análogo ao *command.com* do MS-DOS e de sistemas *Windows 9x*, ou de *shells* utilizados pelos sistemas *Unix*.

Tanto as versões do OS/2 como do *Windows NT* do cmd.exe têm mais detalhes nas mensagens de erro do que o típico "*Bad command or file name*" (no caso de comando malformados) do *command.com*. Na versão OS/2, os erros são reportados no idioma do sistema, sendo que o texto é retirado dos ficheiros de mensagens do sistema. O comando *help* pode ser chamado com o código de erro para se obter informação mais detalhada.

O *cmd* é usado para facilitar a visualização das mensagens se são ou não enviadas e o porquê de não terem sido.

3.1.7. Power BI

O *Power BI* é uma coleção de serviços de *software*, aplicações e conectores que funcionam em conjunto para transformar as origens de dados não relacionadas em informações coerentes, visualmente envolventes e interativas. Os dados podem ser uma folha de cálculo do Excel ou uma coleção de armazéns de dados híbridos no local e com base na *Cloud*. O *Power BI* permite-lhe ligar-se facilmente às origens de dados, visualizar e descobrir o que é importante, bem como partilhar os seus conteúdos com qualquer pessoa. O *Power BI* consiste em vários elementos que funcionam em conjunto em três plataformas básicas:

- Num serviço SaaS (Software como Serviço) online chamado serviço *Power BI*;
- Nas aplicações móveis do *Power BI* para dispositivos *Windows*, *iOS* e *Android*;
- Numa aplicação para computadores com *Windows*, chamada *Power BI Desktop*.

Esta última plataforma foi a utilizada para a realização do projeto, para poder consultar os dados que provém da API.

3.1.8. SQL Server Management Studio

O SQL Server Management Studio (SSMS) foi lançado pela primeira vez com o Microsoft SQL Server. É o sucessor do *Enterprise Manager* no SQL 2000 ou anteriores. É um ambiente integrado para gerenciar qualquer infraestrutura de SQL. O SSMS fornece ferramentas para configurar, monitorar e administrar instâncias do SQL Server e bancos de dados. Usa-se o SSMS para implantar, monitorar e atualizar os componentes da camada de dados usados pelos seus aplicativos, além de criar consultas e *scripts*.

3.1.9. SQL

O SQL foi desenvolvido originalmente no início dos anos 70 nos laboratórios da IBM em San Jose, dentro do projeto *System R*, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional proposto por E. F. Codd. O nome original da linguagem era SEQUEL, acrônimo para “*Structured English Query Language*”. Esta é usada amplamente pela maioria de base de dados. Isto decorre da sua simplicidade e facilidade de uso. Diferencia-se de outras linguagens de consulta de base de dados, no sentido, em que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele. É uma linguagem declarativa em oposição a outras linguagens procedurais (Programação procedural). O que faz com que reduza o ciclo de aprendizagem daqueles que se iniciam na mesma.

3.1.10. Twilio

Twilio é uma empresa americana com sede em San Francisco, Califórnia, que fornece ferramentas de comunicação programáveis para fazer e receber chamadas telefônicas, enviar e receber mensagens de texto e realizar outras funções de comunicação usando API de serviço da web.

Twilio usa *Amazon Web Services* para hospedar a sua infraestrutura de comunicação via API seguindo um conjunto de princípios de design arquitetônico para proteger contra interrupções inesperadas. Em vez de usar protocolos padrão do setor, como SIP (*Session Initiation Protocol*) para controle de chamadas, o *Twilio* usa uma linguagem de marcação personalizada conhecida como *TwIML* para permitir a integração direta com seus serviços. *Twilio* e o cliente normalmente trocam documentos *TwIML* via HTTP *Webhook*, ou seja, pode haver alterações relativamente ao comportamento do retorno de chamada. Os retornos de chamada podem ser mantidos, modificados e gerenciados por usuários e/ou desenvolvedores de terceiros que podem não ser necessariamente afiliados ao aplicativo de origem. O formato geralmente é JSON. O *request* é feito como uma solicitação HTTP POST.

3.1.11. Draw.io

O Draw.io é uma ferramenta online e open-source, que permite a criação e edição de diagramas, gráficos, modelos E-R. Esta ferramenta foi utilizada para a criação de diagramas e esquemas.

3.1.12. API

ASP.Net Web API é um *framework* que facilita a construção de serviços REST HTTP que alcançam uma grande variedade de clientes incluindo dispositivos móveis, *browsers* e aplicações locais. É uma plataforma ideal para construção de serviços REST baseados em *.Net*. O REST (Representational State Transfer), sustentado totalmente em HTTP. Esta tem um formato padrão que é o JSON (Java Script Object Notation), é amplamente utilizado em qualquer plataforma (não apenas *.Net*). É um subconjunto da notação de objeto de *JavaScript*, mas o uso não requer *JavaScript* exclusivamente. Isso proporciona um potencial muito grande para os serviços HTTP, pois é um formato permutável, leve e livre de plataformas.

4. Desenvolvimento

Neste capítulo, será referida a organização realizada em termos de período temporal, para a realização das tarefas ao longo da concretização do projeto e será apresentada a aplicação.

4.1. Calendarização

Antes do projeto ser posto em execução, foi discutido um plano, para tornar mais fácil a organização das tarefas a nível temporal.

Primeiramente, foi apresentado o projeto completo de modo a familiarizar com o que ia ser feito. Passadas as semanas iniciais, foi exposto a primeira parte do projeto proposto sobre automação e envio de *reports* quando era excedido um número de presenças numa zona. A meio do semestre, foi dado a conhecer a segunda parte, que consistia em criar uma API e usar o *Power BI* para lhe fazer um pedido de forma a mostrar o resultado. Por fim, as duas últimas semanas resumiram-se a testar e introduzir alguns ajustes ao que já tinha sido desenvolvido e a corrigir problemas no *Power BI*. Neste período, procedi também à realização do relatório de projeto, assim como os outros documentos solicitados para avaliação.

4.2. Modelação UML

Com a planificação dos módulos do projeto, foi necessário idealizar as ações que cada aplicação iria realizar. Para analisar cada ação, foi desenvolvido dois diagramas de atividades.

4.2.1. Diagramas de Atividades

Parte 1

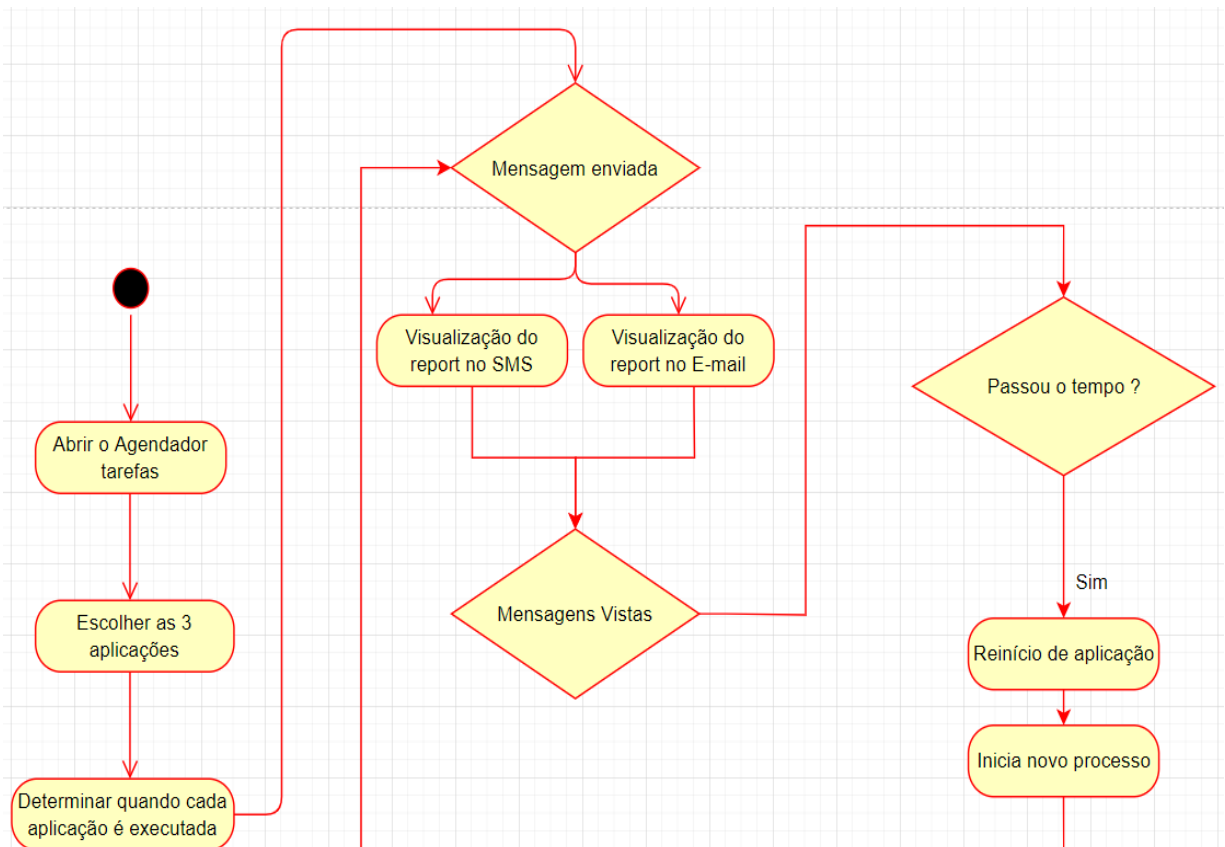


Figura 4-1 Diagrama de Atividades-1

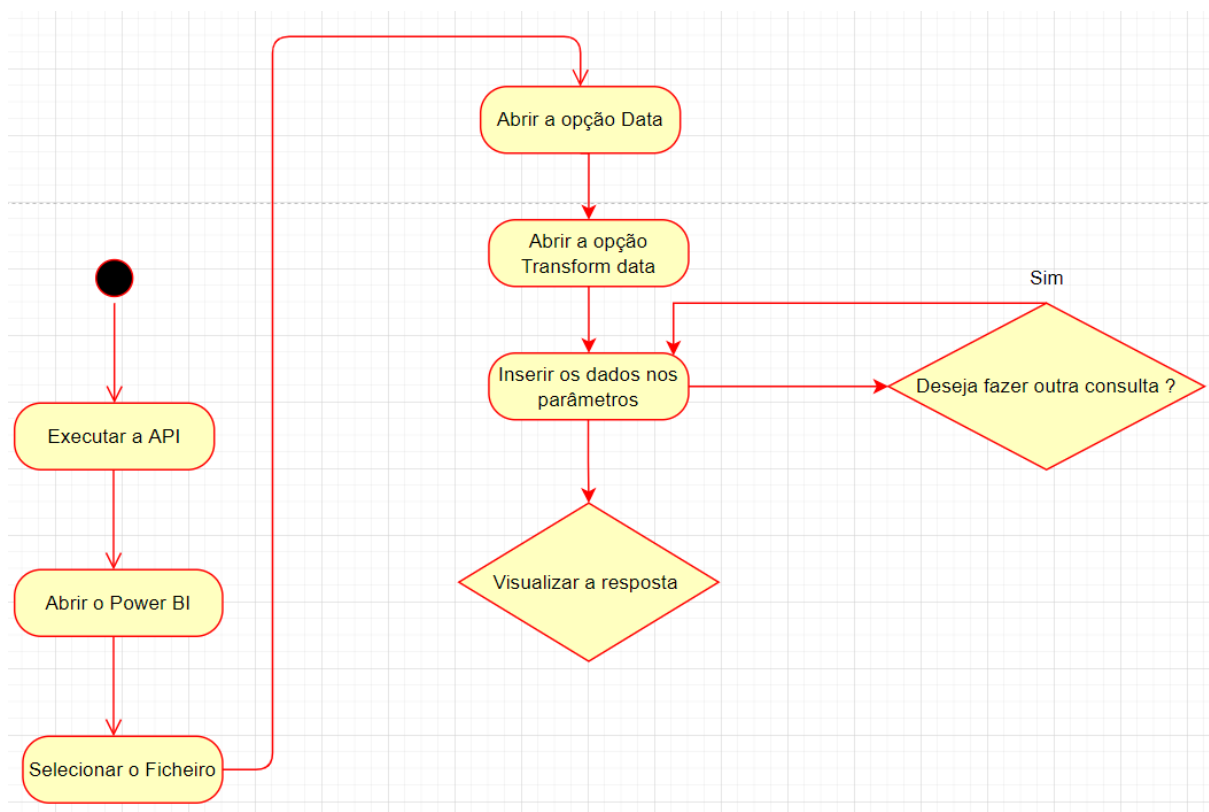


Figura 4-2 Diagrama de Atividades-2

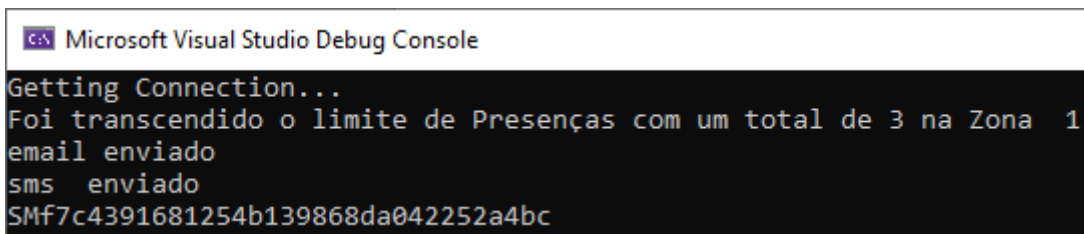
4.3. Análise funcional

Neste capítulo é descrito o funcionamento da aplicação na ótica do utilizador.

Através do agendador de tarefas, da *console application*, da API e do Power BI faz-se, todos os processos descritos seguidamente.

4.3.1. Mensagem enviada

Após ser executado a *console application* aparece no ecrã a mensagem de envio, se foi ou não enviado o *e-mail* e o SMS, como se vê Figura 4-3.



```
Microsoft Visual Studio Debug Console
Getting Connection...
Foi transcendido o limite de Presenças com um total de 3 na Zona 1
email enviado
sms enviado
SMf7c4391681254b139868da042252a4bc
```

Figura 4-3 Cmd

4.3.2. E-mail

Uma vez enviado o *e-mail* com sucesso, ao consultar o email destinatário aparece a mensagem, tal como na Figura 4-4.

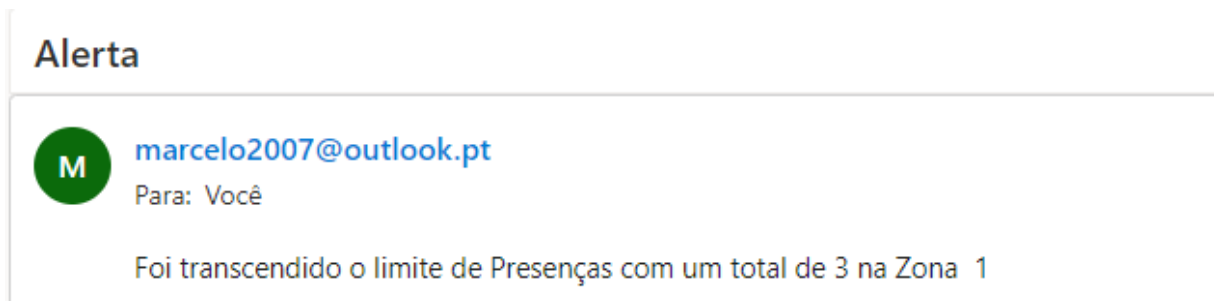


Figura 4-4 E-mail

4.3.3. SMS

Uma vez enviado o SMS com sucesso, ao consultar o número destinatário aparece a mensagem, tal como na Figura 4-5.

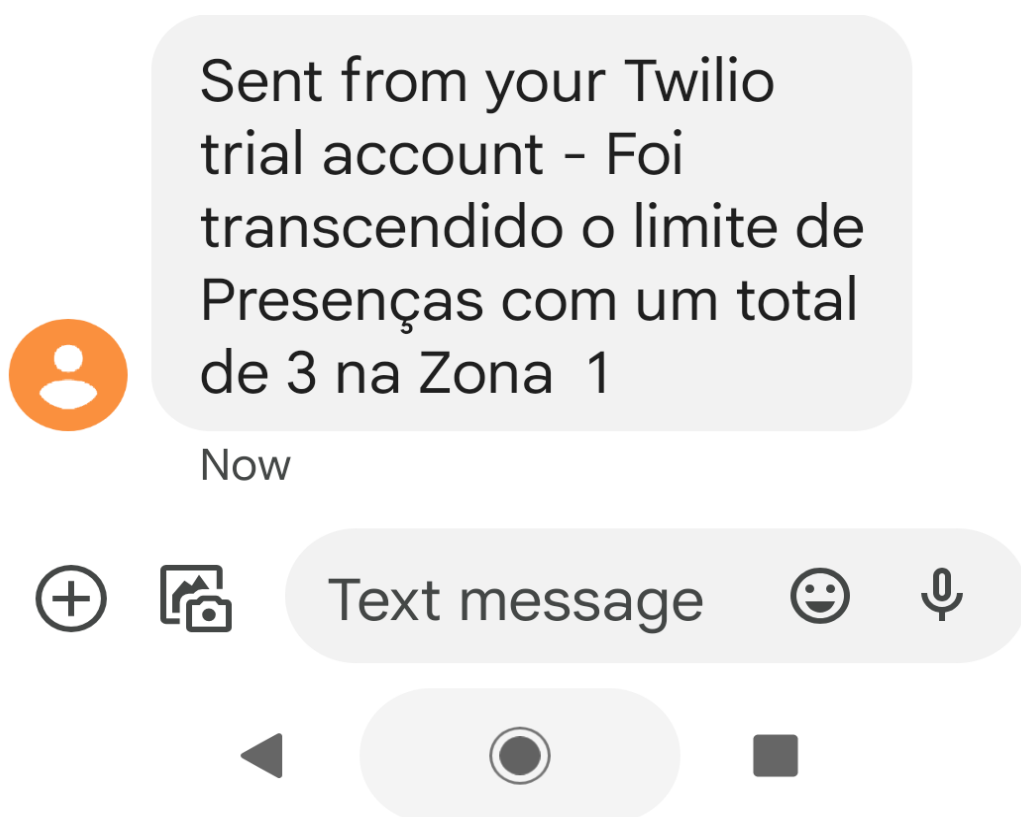
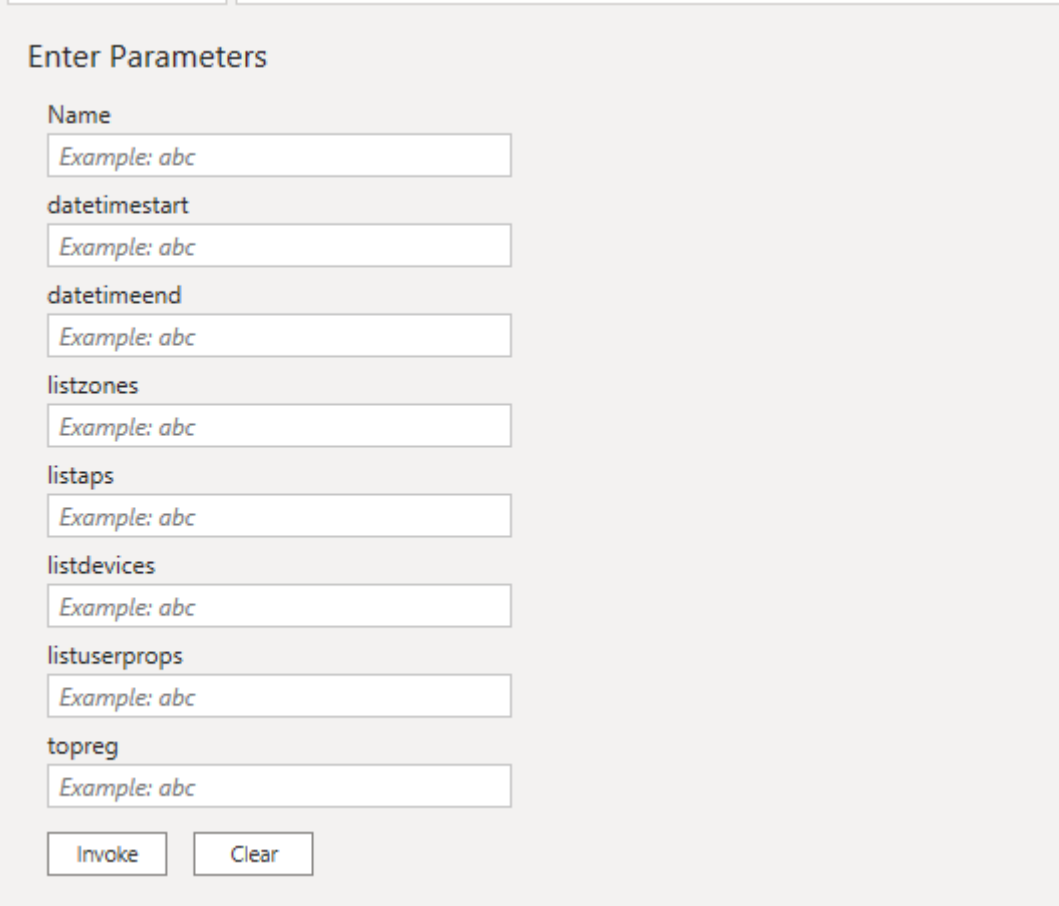


Figura 4-5 SMS

4.3.4. Power BI

Ao utilizador é lhe fornecido um ficheiro que terá de ser aberto no *Power BI* onde o utilizador terá de ir a data e depois a *Transform Data*, onde já vêm a *query* para fazer o pedido à API onde vai ser possível inserir os 8 parâmetros que são pedidos no *stored procedure*, tal como na Figura 4-6.



The image shows a 'Enter Parameters' dialog box with the following fields and buttons:

- Name:
- datetimestart:
- datetimeend:
- listzones:
- listaps:
- listdevices:
- listuserprops:
- topreg:
- Buttons:

Figura 4-6 Inserir parâmetros

Existe 15 diferentes procedimentos que podem ser invocados basta alterar o parâmetro nome para um dos possíveis nomes, tal como na Figura 4-7.

Nome	UniqueVisits
	NewVisits
	NewSessions
	AverageVisitTime
	NumberSessions
	AverageSessionTime
	Top10Applications
	Top10AccessPoints
	Top10Zones
	Top10Devices
	PercentageSessions
	Top10Roles
	NumberUsers
	TopNetworks
	StayingVisits

Figura 4-7 Tipos de Procedimento

4.3.5. Power BI resultado

Um exemplo de execução do procedimento “UniqueVisits” com valores nos restantes parâmetros é, tal como na Figura 4-8.



Figura 4-8 Resultado

4.3.1. Power BI resultado com valores nulos

Um exemplo de execução do procedimento “UniqueVisits” com valores nos restantes parâmetros e com os últimos 3 sem qualquer valor é, tal como na Figura 4-9.



Figura 4-9 Resultado com valores nulos

5. Implementação

Neste capítulo irá ser descrita toda a implementação do sistema detalhadamente, desde a base de dados até ao *Power BI*, e acabando com os desafios passados na concretização do projeto.

5.1. Detalhe técnico e arquitetural

5.1.1. Base de dados NSA

Inicialmente disponibilizou-se um ficheiro para dar *BackUp* à base de dados onde contém as informações que vão ser utilizadas posteriormente. Para isso, foi necessário conceber um utilizador novo para aceder à base de dados que depois permitia fazer pedidos à mesma.

Para se ter conhecimento que numa certa zona foi ultrapassado ou não o limite de presenças previamente estipulado criou-se um *stored procedure*, no *SQL Server*.

Entende-se um *stored procedure*, como um grupo de uma ou mais instruções *Transact-SQL* (linguagem de consulta estruturada de transações usada para criar aplicativos e adicionar lógica de negócios) ou uma referência a um método CLR (*Common Language Runtime*) do *Microsoft .NET Framework*. Os procedimentos podem:

- Aceitar parâmetros de entrada e retornar vários valores no formulário de parâmetros de saída para o programa de chamada.
- Conter instruções de programação que executam operações no banco de dados. Estas incluem a chamada de outros procedimentos.
- Retornar um valor de *status* a um programa de chamada para indicar êxito ou falha (e o motivo da falha).

É demonstrado na Figura 5-1.

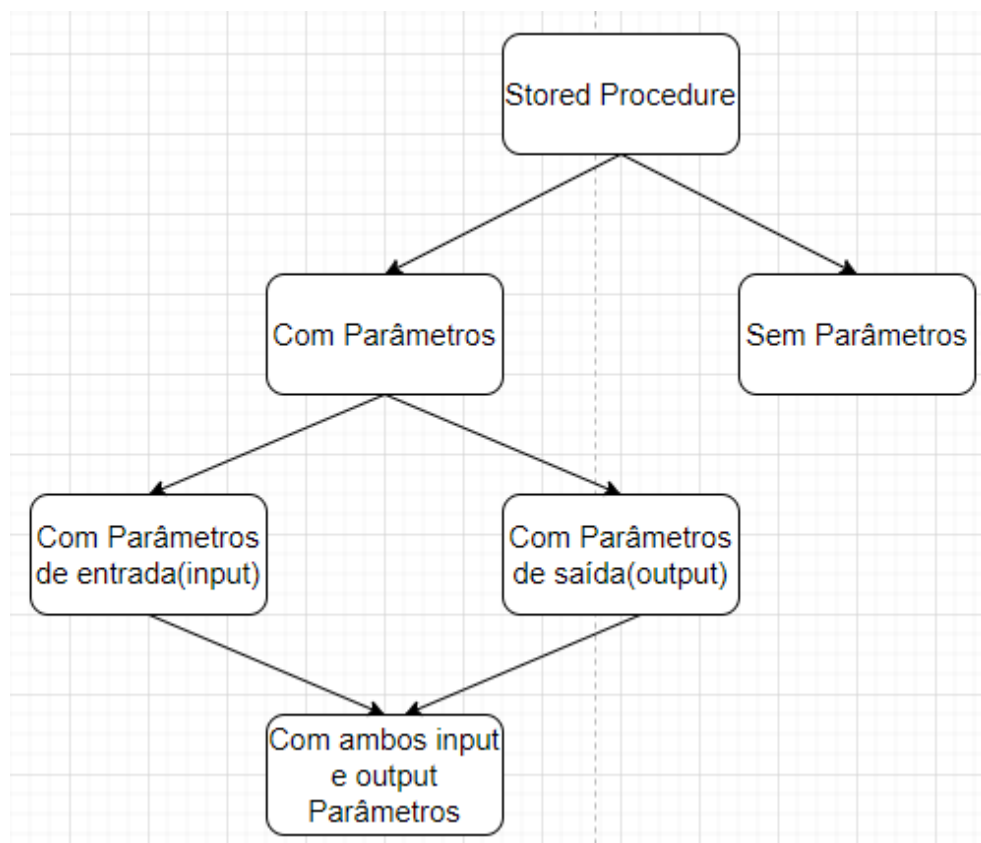


Figura 5-1 Stored procedure

Foi criado um *Stored procedure* para cada zona num total de 3 zonas, tal como na Figura 5-2.

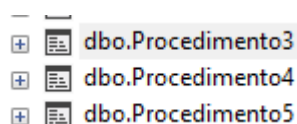


Figura 5-2 Lista Stored procedure

O *Stored procedure* contém 3 parâmetros de entrada, faz a verificação em que zona está e conta o número de presenças nesse local. A contagem é feita tendo em conta que todos os utilizadores estão ligados ao AP que está nessa zona e se a variável “datetime_end” é nula ou não. Se o valor dessa variável for maior que o limite, que neste caso é 1, ao ser executado o procedimento escreve que foi ultrapassado o limite de presenças nessa zona, tal como na Figura 5-3.

Presenças com um total de 4 na Zona 3

Figura 5-3 Resposta Stored procedure

5.1.2. Console application

A nível arquitetural, a *console application* tem como alvo consultar a base de dados, que neste caso é *SQL Server*. No entanto, precisam de se conectar antes, com recurso do *ADO.NET*, que fica entre a aplicação e a base de dados. Este é um componente mediador que vai conectá-la à aplicação, através da biblioteca: *System.Data.SqlClient* importada. Após isso, a aplicação já sabe que existe uma base de dados, *SQL Server*, e com recurso às classes que tem disponíveis concretiza os seguintes passos:

- *Connection*, providência uma conexão para a base de dados, que necessita de uma *connection string* com os 5 componentes tal como na Figura 5-4.

Nome do Parametro da String de conexão	descrição
Data Source	Identifica o servidor. Pode ser a maquina local, domínio ou endereço IP.
Initial Catalog	nome do banco de dados
Integrated Security	Define o SSPI para efetuar a conexão com usuário logado no Windows
User ID	Nome do usuário definido no SQL Server.
Password	Senha

Figura 5-4 Connection String

- *Command*, usa o *ExecuteReader* para criar o comando para executar a *query* com o fundamento de retornar linhas de dados, tais como o *SELECT*.
- *Data Reader*, lê os dados retornados pelo objeto *Command*. Permite aceder e percorrer os registos no modo de somente leitura e somente para frente, não oferecendo acesso desconectado e não permite alterar ou atualizar a fonte de dados original sendo usado para obter rapidamente dados de apenas leitura.

Após esta já ter feito o *Data Reader*, tal como a Figura 5-5, vamos adicionar a resposta a uma variável global chamada *abc* do tipo *string*.

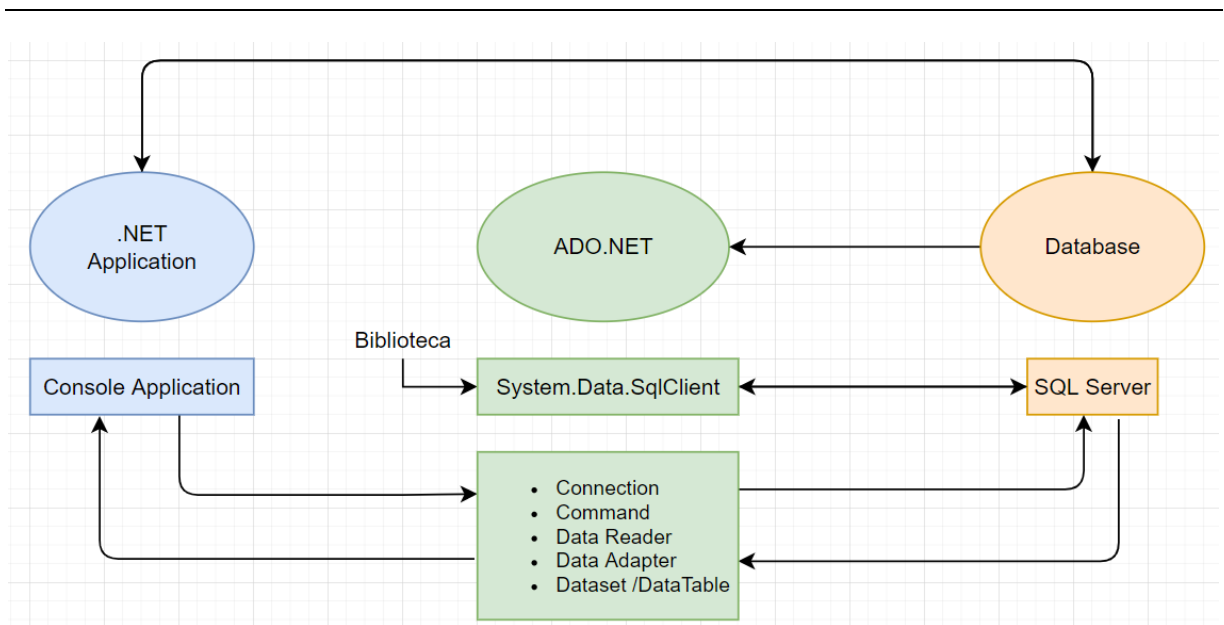


Figura 5-5 ADO.NET

5.1.3. Arquitetura aplicação e procedimento

Para enquadrar ainda melhor como funciona o relacionamento entre estes três elementos (*stored procedure*, a base de dados e a *console application*), o *stored procedure* realiza várias instruções na *query*, designadamente, o DML (*Data Manipulation Language*). As instruções que pode realizar são o *SELECT*, *INSERT*, *UPDATE* e *DELETE* que permitem que os usuários do SQL visualizem e gerenciem os dados.

Após o *stored procedure* ser criado, é armazenado na base de dados. Desta forma, e como mostrado anteriormente, é feita a ligação na *console application* para ter acesso à base de dados. Assim, pode-se invocar o *stored procedure*, o que permite que chame o mesmo sempre que a *console application* seja executada, criando um ciclo, tal como a Figura 5-6.

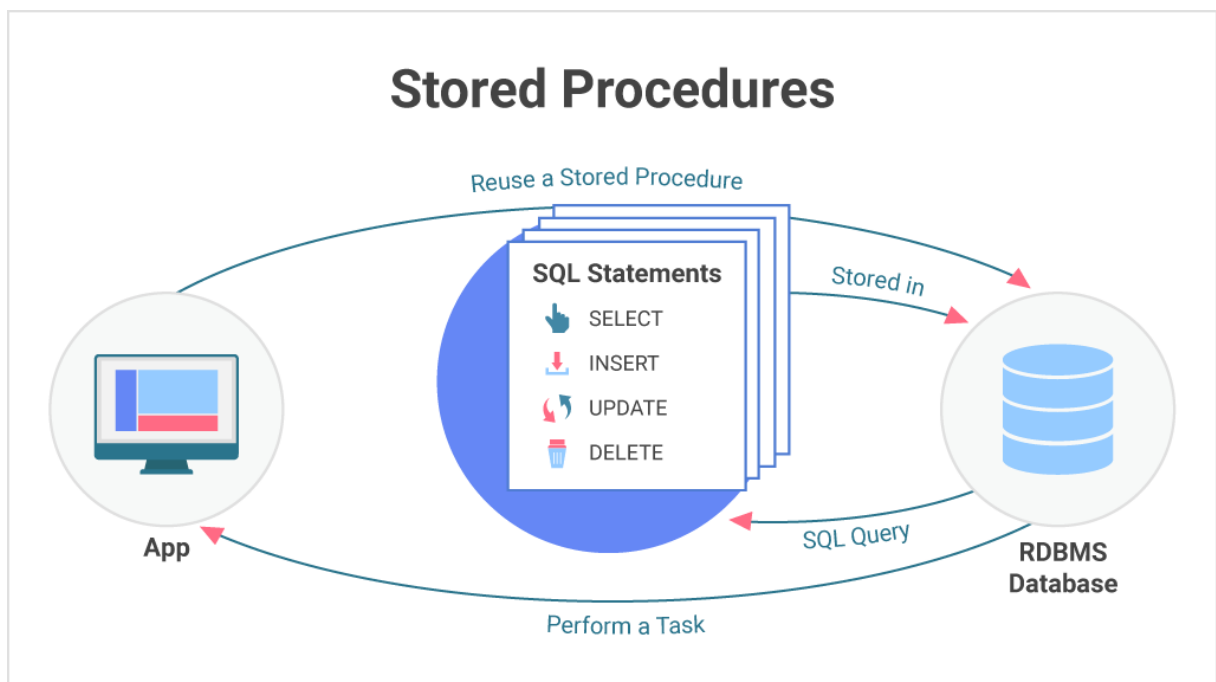


Figura 5-6 Arquitetura geral

5.1.4. Configuração de Email

Numa próxima fase, é usado o *SmtplibClient* de forma a conectar a aplicação a um serviço de *e-mails*. Para construir e enviar uma mensagem de *e-mail* usa-se o *SmtplibClient*, porém, é necessário especificar as seguintes informações:

- O servidor *host* SMTP que é usado para enviar *e-mail* e a porta (Outlook e porta 587).
- Credenciais para autenticação, exigidas pelo servidor SMTP.
- O endereço de *e-mail* do remetente.
- O endereço de *e-mail* do destinatário.
- O conteúdo da mensagem (*Body*) que é a variável global *abc*.

A conexão estabelecida pela instância atual da classe *SmtplibClient* com o servidor SMTP pode ser reutilizada se um aplicativo desejar enviar várias mensagens para o mesmo servidor SMTP. Isto é particularmente útil quando a autenticação ou a criptografia são usadas para estabelecer uma conexão com o servidor SMTP. Todavia o processo de autenticação e estabelecimento de uma sessão TLS pode ser uma operação dispendiosa. Para além disso, um requisito para restabelecer uma conexão para cada mensagem ao enviar uma grande quantidade de emails para o mesmo servidor SMTP pode ter um impacto significativo no desempenho. Assim, com recurso à implementação da classe *SmtplibClient*, que agrupa conexões SMTP, evita-se a sobrecarga de restabelecer uma conexão para cada mensagem com o servidor igual. Um aplicativo também pode reutilizar o objeto *SmtplibClient* para enviar muitos *e-mails* diferentes para o mesmo servidor SMTP e para vários SMTP distintos. Quando uma sessão SMTP é finalizada e o cliente deseja encerrar a conexão, é importante enviar uma mensagem *QUIT* ao servidor para indicar que não há mais informações para enviar, isto permite que o servidor libere recursos associados à conexão do cliente e processe as mensagens que foram enviadas pelo próprio. A classe *SmtplibClient* não tem um método para finalizar, portanto, um aplicativo deve chamar o *Dispose* para libertar recursos explicitamente. O método *Dispose* relaciona-se através de todas as conexões estabelecidas com o servidor SMTP especificado na propriedade *Host* e envia uma mensagem *QUIT* seguida do encerramento normal da conexão TCP. Figura 5-7.

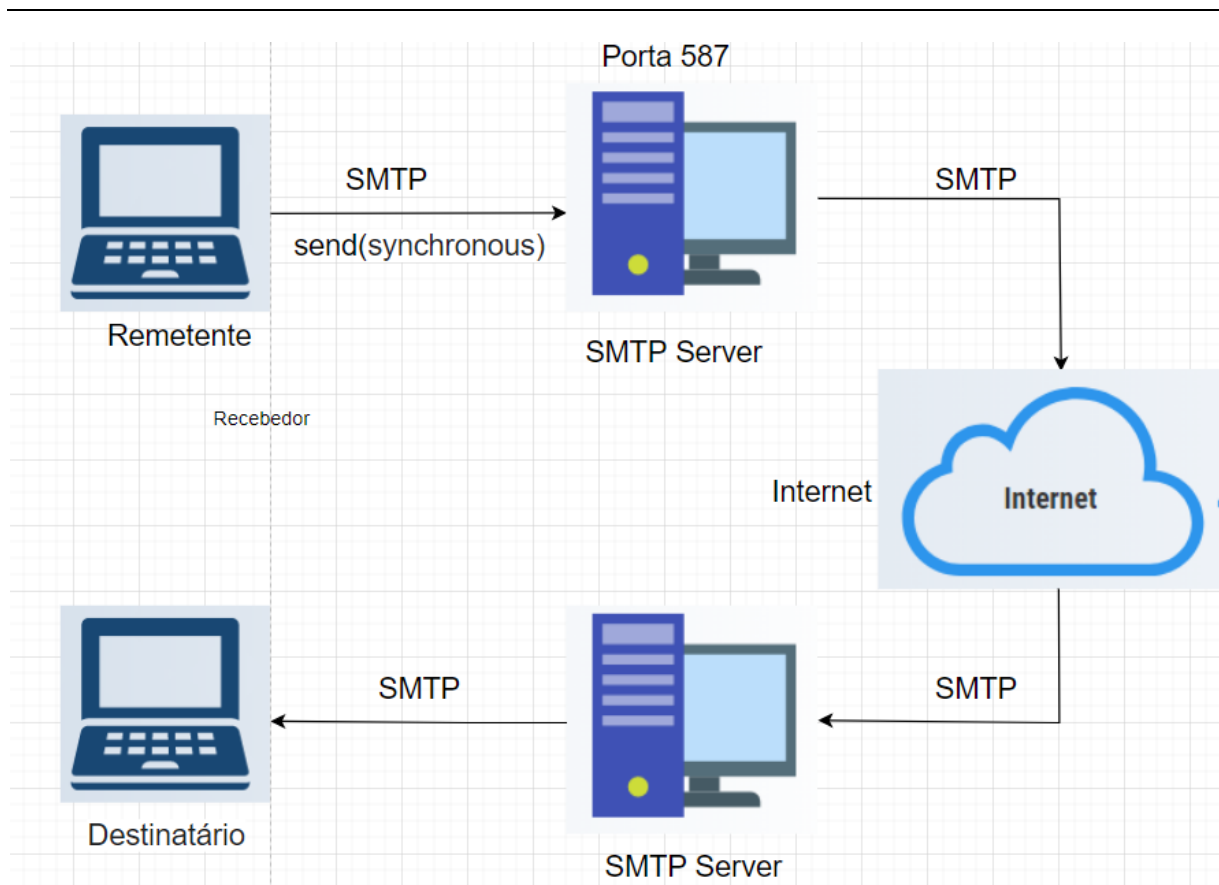


Figura 5-7 Configuração de E-mail

5.1.5. Configuração do SMS

Por fim é implementado o serviço *Twilio* para o envio do SMS, ao nível de configuração após estruturar todo o código necessário é imperativo a criação do número do remetente que é dado quando se regista a conta e compra-se um número no website *Twilio* e consequentemente o *accountSid* e o *authToken* é fornecido. Realizado os passos anteriores, é feita a substituição dos valores dos espaços reservados para os valores já fornecidos e acrescenta-se o corpo da mensagem que é igual à variável global *abc*.

Ao nível de arquitetura, quando o *Twilio* recebe a solicitação para enviar um SMS é feita uma verificação se foi incluído um número de telefone válido do *Twilio* no campo *from*. O *Twilio* então vai adicionar o SMS a uma *queue* ou retornará um erro HTTP na resposta à solicitação. Supondo que a solicitação não resultou em nenhum erro, a resposta do *Twilio* incluirá o SID da nova mensagem. Esse identificador exclusivo ajuda a referenciar esta mensagem. No código, é escrito o SID no terminal de forma é ser possível saber se foi ou não enviado o SMS. Figura 5-8.

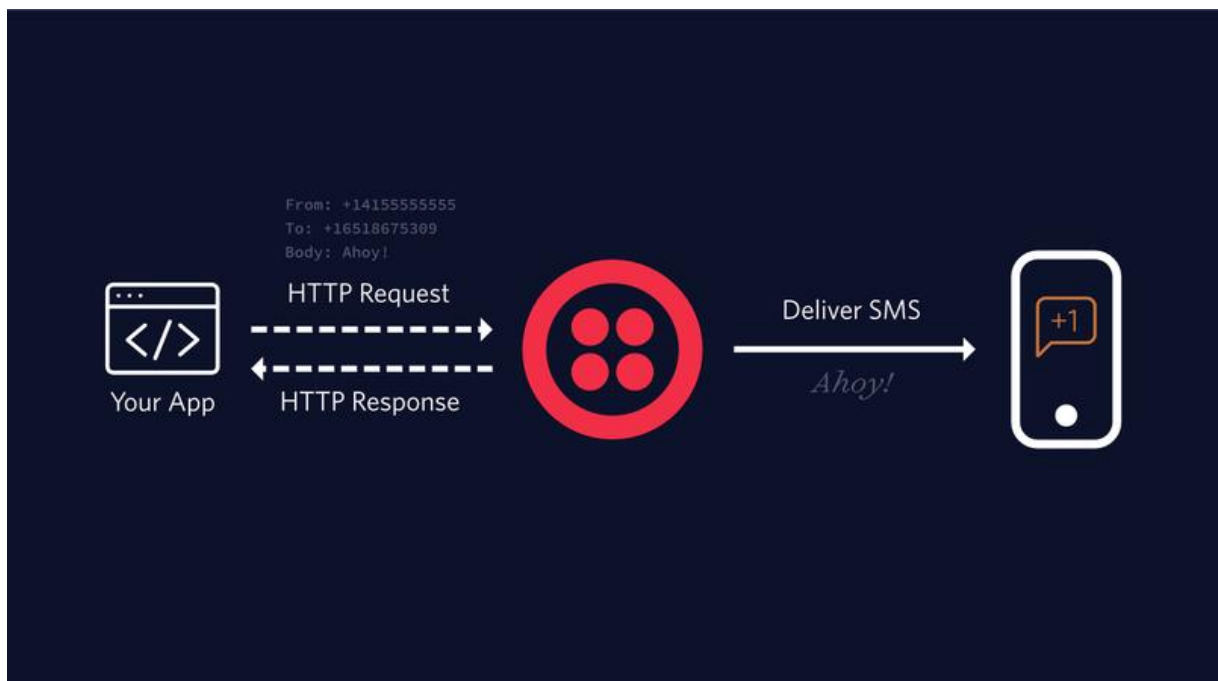


Figura 5-8 Arquitetura Twilio

5.1.6. Windows task scheduler

É com recurso ao *Windows task scheduler* que se vai programar como cada aplicação vai ser executada. O *task scheduler* necessita que o executável(.exe) da *console application* já esteja publicado, posto isto define-se uma tarefa com um nome e uma descrição, de quanto em quanto tempo vai ser acionado, ou seja, o *trigger* e qual a ação que vai ser acionada que neste caso é de uma das três *console application* e repete-se o mesmo processo para as restantes duas. Figura 5-9 .

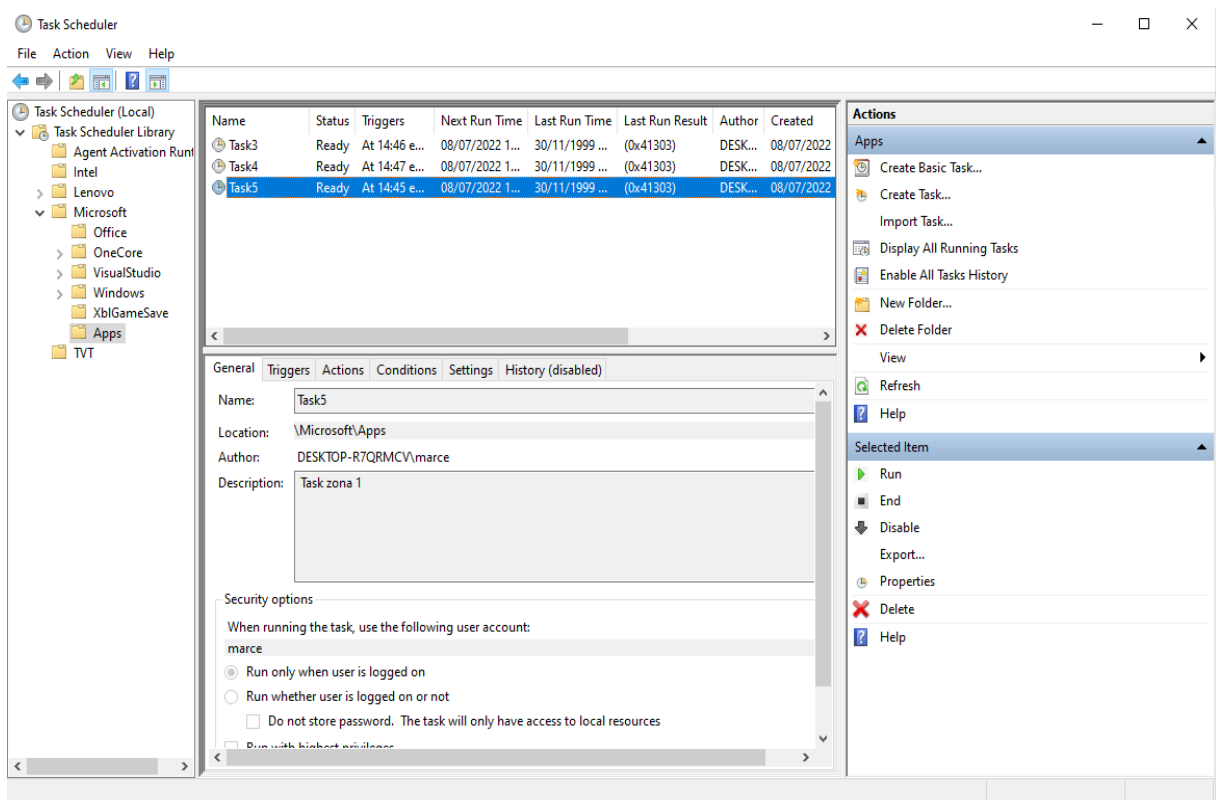


Figura 5-9 Windows task scheduler

5.1.7. API

Na segunda parte do projeto, a API tem uma arquitetura parecida ao que foi explicado no capítulo 5.1.2, mas a nível das classes difere, pois, após estabelecer a *string Connection*, a *class Command* vai ser definida como um comando do tipo *StoredProcedure*, vai ser adicionado os parâmetros que vem do procedimento, aos parâmetros da função *controller*. Não é usado o *Data Reader*, mas sim o *Data Adapter* e o *Dataset* :

- *Data adapter*, vai também recuperar o resultado como o *Data Reader*, mas tem a função preencher a lacuna entre o objeto do *Dataset* e a base de dados, por outras palavras criar uma ponte entre um objeto de dados e a base de dados.
- *Dataset* é, uma tabela que contém uma coleção de objetos da tabela de dados e cada objeto é uma coleção de linhas(rows) e colunas(columns) de dados, ou seja, ter uma coleção de linhas para diferentes tabelas e uma coleção de colunas de diferentes tabelas.

Figura 5-10.

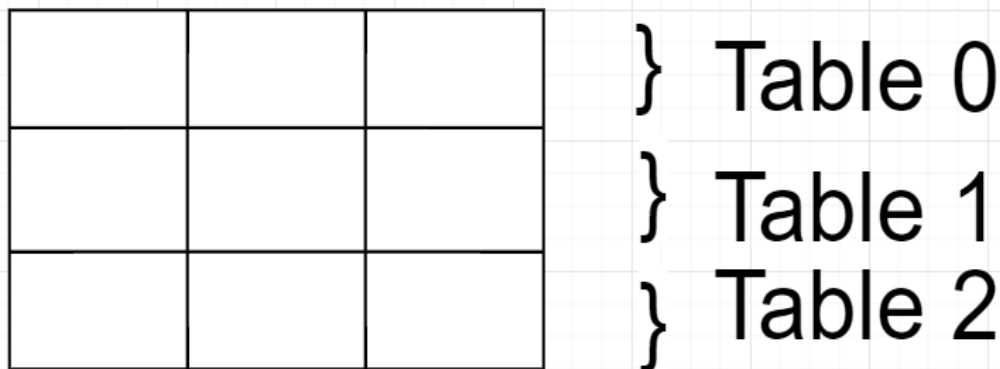


Figura 5-10 Dataset

Após finalizado todo o processo é assim feito o *HttpGet* retornando uma *jsonString* que é na verdade um *URL* para poder ser possível fazer pedidos *GET* numa outra plataforma que é o *Power BI*.

5.1.8. Configuração do Power BI

Ao nível de arquitetura, o *Power BI* é preciso escolher inicialmente, a fonte de dados. O *Power BI* tem o recurso de obtenção de dados com o qual se pode conectar a diferentes tipos de fontes de dados, como arquivos, bases de dados, *Cloud*, conexões a API etc. As conexões de dados são estabelecidas dessas fontes de dados para ferramentas de autoria, como o *Power BI Desktop*.

Com recurso ao *Power BI Desktop*, para que seja possível executar todos os 15 procedimentos é criado uma *query* que receba os 8 parâmetros todos do tipo *texto*. Figura 5-11.

```
((Name as text,datetimestart as text, datetimeend as text, listzones as text, listaps as text, listdevices as text, listuserprops as text, topreg as text) as table =>
```

Figura 5-11 Parâmetros

É preciso depois passar o *URL* base após ter-se escolhido que a fonte de dados era JSON. Figura 5-12.

```
let  
Source = Csv.Document(Web.Contents("https://localhost:44310/api/Emp?Name="
```

Figura 5-12 URL base

E finalmente passar o sufixo do *URL* base que contém os parâmetros. Figura 5-13.

```
& Name & "&datetimestart=" & datetimestart & "&datetimeend=" & datetimeend & "&listzones=" & listzones & "&listaps=" & listaps & "&listdevices=" & listdevices & "&listuserprops=" & listuserprops & "&topreg=" & topreg))
```

Figura 5-13 URL base com parâmetros

5.2. Desafios

Uma das dificuldades enquanto desenvolvia a aplicação, foi a implementação dos procedimentos para determinar o número de presenças, porque demorei um pouco a perceber como estava composta a base de dados e como que um utilizador estava ou não na zona.

Outro incómodo foi, no final da segunda fase ao introduzir o URL que provem da API era me imposto pelo Power BI que eu coloca-se os parâmetros todos com os respetivos valores e que depois ao alterar esses valores dava sempre erro. Com muita insistência da minha parte, encontrei uma forma de fazer tudo manualmente e consequentemente já conseguia passar os parâmetros dinamicamente, no entanto há parâmetros que é necessário passar valores null e ao desenvolver a API consegui arranjar uma maneira de ser possível em vez de passar null bastava não colocar qualquer valor no parâmetro. Resultou para toda a plataforma para testar API como o Swagger, Postman, SoapUI, Katalon Studio, no entanto no Power BI não, ao continuar reparei que era possível alterar os parâmetros no URL que aparecia após ter feito um pedido com sucesso e alterando os parâmetros para, "", "", """) começou a funcionar tudo corretamente.

6. Conclusão

6.1. Principais resultados e contribuições

A realização deste projeto permitiu-me, de uma forma prática, aplicar os meus conhecimentos adquiridos ao longo de todo o percurso académico, assim como adquirir novos conhecimentos para a realização deste projeto.

Para mim, foi uma experiência enriquecedora, pois tive contacto com o mundo empresarial, onde me permitiu tornar melhor profissional, adquirir e melhorar novas metodologias de trabalho em áreas com a qual nunca tinha entrado em contacto, tal como o Power BI.

O projeto foi desenvolvido na linguagem C# e SQL, onde contei com a ajuda da entidade acolhedora, que sempre se mostrou disponível e à disposição para me ajudar, ensinar e transmitir conhecimentos em todas as dúvidas que foram surgindo. Agradeço, mais uma vez, por toda a disponibilidade que me foi dispensada. Sinto-me grato por ter tido a oportunidade de poder facilitar o processo de consultoria informática, mais propriamente pela Claranet.

6.2. Trabalho futuro

O desenvolvimento do projeto nunca está completo pois existe sempre alguma coisa que pode vir a ser acrescentada, de forma a aumentar a eficiência, qualidade e usabilidade do projeto.

Como trabalho futuro pretendo desenvolver um método que verifique cada *report* e compare com o anterior, para caso não haja alterações no seu conteúdo, não ser enviado o email, mas sim uma notificação do porquê de não ser enviado o suposto *report*. Outro aspeto interessante a integrar, seria guardar e atualizar o conteúdo desses *reports* para uma base de dados. Para finalizar, os *reports* serem enviados por outro meio que não por email e por SMS, tornando assim o sistema mais completo e eficiente.

Referências Bibliográficas

- [1] Language Support in Visual Studio Code. (s.d.). Obtido de Visual Studio Code:
<https://code.visualstudio.com/docs/languages/overview>
- [2] Visual Studio Code FAQ. (s.d.). Obtido de Visual Studio Code:
https://code.visualstudio.com/docs/supporting/faq#_how-to-disable-telemetry-reporting
- [3] Claranet. (s.d.). Obtido de Claranet:
<https://www.claranet.pt/>
- [4] SmtpClient Class. (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/api/system.net.mail.smtpclient?view=net-6.0>
- [5] Documentation. (s.d.). Obtido de Selenium:
<https://www.selenium.dev/documentation/>
- [6] C# Documentation. (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/csharp/>
- [7] O que é o .NET Framework? Quais as vantagens? (s.d.). Obtido de Enotas:
<https://enotas.com.br/blog/net-framework/>
- [8] Process Class (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process?view=net-6.0>
- [9] Directory Class (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/api/system.io.directory?view=net-6.0>
- [10] Path Class (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/api/system.io.path?view=net-6.0>
- [11] File Class (s.d.). Obtido de Microsoft Docs:
<https://docs.microsoft.com/en-us/dotnet/api/system.io.file?view=net-6.0>

[12] Why Serilog? (s.d.). Obtido de Serilog:

<https://serilog.net/>

[13] C# - Usando o Serilog (s.d.). Obtido de Macoratti:

https://www.macoratti.net/18/05/c_serilog1.htm

[14] How to get execution directory of console application (7 de janeiro de 2021). Obtido

de StackOverflow: <https://stackoverflow.com/questions/10993721/how-to-get-execution-directory-of-console-application>

[15] How to Execute C# Program on cmd (command-line)? (30 de janeiro de 2020). Obtido

de GeeksforGeeks: <https://www.geeksforgeeks.org/how-to-execute-c-sharp-program-on-cmd-command-line/>

[16] Send SMS and MMS Messages in (23 de Julho de 2021). Obtido de twilio:

<https://www.twilio.com/docs/sms/tutorials/how-to-send-sms-messages-csharp>