PM

IP Network Usage Accounting - Parte I

PROJETO DE MESTRADO

Daniel José Gomes Aguiar

MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

janeiro | 2015

IP Network Usage Accounting - Parte I

PROJETO DE MESTRADO

Daniel José Gomes Aguiar
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
Karolina Baras

CO-ORIENTADOR
Lina Maria Pestana Leão de Brito
Nélio José da Silva Vieira

IP Network Usage Accounting – Parte I

Daniel José Gomes Aguiar

Constituição do júri:

Prof. José Luís Silva, Presidente

Prof. Leonel Domingos Telo Nóbrega, Vogal

Prof.ª Karolina Baras, Vogal

Abril 2015

Funchal – Portugal

Resumo

Um Internet Service Provider (ISP) tem a seu cargo a gestão de redes compostas por milhares de clientes, onde é necessário haver um controlo rígido da largura de banda e evitar o congestionamento da mesma. Para isso os ISPs precisam de sistemas capazes de analisar os dados de tráfego e atribuir a qualidade de serviço (QoS) adequada com base nesses dados.

A NOS Madeira é o ISP líder na ilha da Madeira, com milhares de clientes por toda a Região. O sistema de controlo de largura de banda existente nesta empresa era obsoleto e levou à necessidade da criação de um novo sistema de controlo de largura de banda.

Este novo sistema, denominado *IP Network Usage Accounting*, é composto por três subsistemas: *IP Mapping System*, *Accounting System* e *Policy Server System*. Este relatório, fala sobre o desenho, a implementação e testes do primeiro subsistema, *IP Mapping System.*

O *IP Mapping System* é responsável por realizar a recolha de dados de tráfego realizado pelos clientes da NOS Madeira e a fornecê-los ao segundo subsistema, (Accounting System). Este, por sua vez, realiza uma análise desses mesmos dados e envia os resultados ao terceiro subsistema (Policy Server System) que aplica o QoS correspondente a cada IP dos clientes.

Palavras - Chave

Controlo de largura de banda

Recolha de dados

Tráfego Web

Base de Dados

Serviços Web

Abstract

An Internet Service Provider (ISP) is responsible for the management of networks made up of thousands of customers, where there must be strict control of bandwidth and avoid congestion of it. For that ISPs need systems capable of analyzing traffic data and assign the Quality of Service (QoS) suitable in reliance thereon.

NOS Madeira is the leading ISP in Madeira, with thousands of customers throughout the region. The existing bandwidth control system in this company was obsolete and led to the need to create a new bandwidth control system.

This new system, called *IP Network Usage Accounting*, consists of three subsystems: *IP Mapping System*, *Accounting System* and *Policy Server System*. This report talks about the design, implementation and testing of the first subsystem, *IP Mapping System*.

The *IP Mapping System* is responsible for performing the collection of traffic data held by customers of NOS Madeira and provide them to the second subsystem (*Accounting System*). This, in turn, performs an analysis of these data and sends the results to the third subsystem (*Policy Server System*) applying QoS corresponding to each of the clients IP.

Keywords

Bandwidth Throttling

Data collection

Network traffic

Databases

Web Services

Agradecimentos

Gostaria em primeiro lugar agradecer à minha orientadora Karolina Baras e à minha co-orientadora Lina Maria Pestana Leão de Brito por todo o apoio na elaboração deste projeto assim como a motivação prestada ao longo do mesmo.

Gostaria de agradecer ao meu co-orientador externo, Nélio José da Silva Vieira, assim como ao Francisco Azevedo por todo o apoio fornecido e conhecimentos transmitidos ao longo deste projeto.

Gostaria de agradecer aos meus colegas, Luís Ferreira e Jorge Canha, por toda ajuda fornecida colocando muitas vezes o próprio projeto de parte, e também pelos conhecimentos que transmitiram.

Gostaria de agradecer a todos os colaboradores da NOS Madeira, por toda a ajuda fornecida o longo do projeto.

Gostaria de agradecer a todos os professores ao longo do meu percurso académico, pelos conhecimentos transmitidos.

Gostaria de agradecer aos meus pais, e meus irmãos que sempre me apoiaram e motivaram na elaboração deste projeto.

Gostaria de agradecer aos meus amigos pela motivação e apoio dados ao longo deste projeto, nomeadamente ao Paulo Caldeira, Fábio Camacho, Andreia Guilherme, Joana Guilherme, Rui Ruel e Jéssica Freitas.

E por fim, gostaria de agradecer a todos aqueles que de uma maneira ou de outra deram apoio neste projeto.

Índice

Índice de Figuras

Lista de Acrónimos

API – Application Programming Interface

CMTS – Cable Modem Termination System

CPE – Customer Premises Management

DTS – Data Transformation Services

FTTH - Fiber To The Home

HFC – Hybrid Fiber-Coaxial

HTTP – Hyper Text Transfer Protocol

IP – Internet Protocol

ISP – Internet Service Provider

LAMP – Linux, Apache, MySQL, PHP

MAC – Media Access Control

MVC – Model-View-Controller

OLT – Optical Line Terminal

OSS – Operations Support System

QoS – Quality of Service

SCE – Service Control Engine

SOAP – Simple Object Access Protocol

SQL – Structed Query Language

UDDI – Universal Description, Discovery and Integration

URL – Uniform Resource Locator

WSDL – Web Services Description Language

XML – Extensible Markup Language

Capítulo 1 – Introdução

Hoje vivemos num mundo onde o acesso a internet é muito facilitado, podendo ser encontrado em casas particulares, ruas ou espaços comerciais. A largura de banda disponível necessita de ser repartida por todos os utilizadores. Um ISP é responsável por fazer essa gestão da largura de banda, de maneira a todos os utilizadores terem qualidade de serviço quando utilizam a internet. Para os utilizadores de casa, que contratam os serviços de um ISP, o processo tem que ser mais rígido de forma a garantir que o cliente tenha um serviço de qualidade, através de um uso moderado da largura de banda, o que muitas vezes não acontece.

A NOS Madeira é o principal ISP na ilha da Madeira, com um grande volume de clientes, oferecendo serviços de internet a partir de várias tecnologias, nomeadamente FTTH (Fiber To The Home), HFC (Hybrid Fiber-Coaxial) e redes móveis (3G/4G). A NOS Madeira, como qualquer outro ISP, tem de analisar o tráfego efetuado pelos clientes de maneira a verificar a utilização da largura de banda e apurar os pontos de melhoria. Muitas vezes um ISP encontra utilizadores com excesso de utilização da largura de banda, congestionando a mesma. Para resolver este problema, na NOS Madeira criou um sistema de controlo de largura de banda, que atribuía uma prioridade a um cliente, consoante a sua utilização da largura de banda, permitindo assim que aqueles que fossem mais moderados no uso da mesma fossem beneficiados, garantindo maior largura de banda e limitando a largura de banda aos clientes que excediam os valores aceitáveis definidos pela empresa. Este controlo de largura de banda também é muito importante para a NOS Madeira, uma vez que os custos de aluguer do cabo submarino são muito elevados e a realização de um controlo rígido de largura de banda permite à empesa minimizar estes custos.

O controlo de largura de banda na NOS Madeira, era um processo feito por três componentes: componente que recolhe dados estatísticos do uso da rede, componente de análise desses dados e componente de controlo da rede. Houve necessidade de implementação de um novo sistema, devido ao sistema antigo estar obsoleto. O sistema utilizava tecnologias que já estavam em desuso e cujas atualizações eram difíceis.

O novo sistema de controlo de largura de banda, denominado *IP Network Usage Accounting* é um sistema que permite um controlo de largura de banda de forma mais eficiente. A criação de novas regras de negócio é feita de forma mais amigável para o utilizador e oferece uma série de serviços web, que permite a NOS Madeira realizar uma análise mais precisa dos dados de tráfego efetuado pelos clientes. Este sistema é dividido em três subsistemas diferentes: *IP Mapping System*, *Accounting System* e *Policy Server System*. A implementação destes subsistemas foi dividida por três desenvolvedores. O *IP Mapping System* é o subsistema abordado neste relatório e que trata da recolha dos dados de tráfego dos clientes. O *Accounting System*, trata de fazer a análise dos dados recolhidos pelo subsistema anterior e realizar cálculos com esses dados para obter o valor de tráfego ponderado, de maneira a atribuir o QoS correspondente. Foi implementado pelo Luís Ferreira, e encontra-se descrito em [1]. Por fim, o *Policy Server System*, tem a responsabilidade de receber o par de dados QoS por

IP, e aplicar o mesmo aos equipamentos dos clientes. Foi implementado por Jorge Canha, e encontra-se descrito em [2]. Com a implementação deste novo sistema foi possível adiar um upgrade da largura de banda disponível do cabo submarino, minimizando assim os custos de aluguer do mesmo.

1.1 Objetivos

O objetivo principal deste projeto é a implementação de um sistema de controlo de banda larga para a empresa NOS Madeira, de forma a ultrapassar as limitações existentes no sistema anterior. Este novo sistema deve ter mais escalabilidade, flexibilidade, permitir a integração e modularidade. Este projeto irá ser dividido em três subsistemas: *IP Mapping System*, *Accounting System* e *Policy Server System.* Estes subsistemas irão ser desenvolvidos por três desenvolvedores diferentes. Este relatório trata do processo de desenvolvimento do primeiro subsistema, *IP Mapping System.*

O objetivo do primeiro subsistema, *IP Mapping System* é que seja capaz de recolher dados de tráfego dos clientes da NOS Madeira e enviar os mesmos para o subsistema *Accounting System,* mantendo os dados armazenados durante um ano como histórico, para fins legais.

1.2 Estrutura do Documento

Este documento é composto por seis capítulos:

- No primeiro capítulo, encontra-se a introdução, onde é feita uma breve descrição do sistema geral e os três subsistemas e da empresa NOS Madeira.
- No segundo capítulo é abordado o sistema anterior e a necessidade para implementar o novo sistema, sendo analisadas as tecnologias relevantes para este projeto.
- No terceiro capítulo faz-se referência à arquitetura geral do sistema, o subsistema *IP Mapping*, os requisitos funcionais e não-funcionais e a metodologia de desenvolvimento Scrum.
- No quarto capítulo é mencionada a implementação do sistema, onde são vistas as ferramentas utilizadas e a estrutura da base de dados.
- No quinto capítulo são descritos os resultados obtidos, os problemas encontrados ao longo do projeto e os tempos de execução.
- No sexto e último capítulo são apresentadas as conclusões do projeto.
- Nos anexos encontram-se o diagrama de fluxo de dados e o diagrama de sequência do subsistema *IP Mapping System*.

Capítulo 2 – Contexto Tecnológico

Neste capítulo iremos descrever e analisar sumariamente o sistema anterior existente na NOS Madeira para controlo de largura de banda e as suas limitações. Irão ser explicadas algumas das tecnologias relevantes para este projeto, tais como: Serviços Web, metodologia de desenvolvimento SCRUM e Sistema Gestor de Base de Dados Relacional (SGBD).

2.1 Sistema Anterior

Antes de descrever o sistema anterior é importante fazer referência a alguns conceitos para compreendermos melhor o que é e para que serve a regulação da largura de banda (bandwidth throttling).

A regulação de largura de banda é realizada com o intuito de aumentar ou diminuir intencionalmente a largura de banda de um serviço de internet por parte de um ISP, com o objetivo de fazer a gestão e controlo do uso da largura de banda no sentido de minimizar o congestionamento da rede. Isto permite que os utilizadores que realizem menos tráfego tenham maior largura de banda e que os utilizadores que realizem mais tráfego vejam a sua largura de banda reduzida, definidos pela empresa.

O sistema anterior existente na empresa NOS Madeira tinha muitas limitações, o que levou à decisão da implementação deste novo sistema de contabilidade de tráfego. As principais limitações verificaram-se nas tecnologias utilizadas, que eram obsoletas e pouco flexíveis, o que não permitia a realização de atualizações no sistema.

O sistema anterior era executado no Windows 2000 e Microsoft SQL Server 2000, o que já não têm qualquer tipo de suporte por parte da Microsoft, visto que são produtos descontinuados.

Este sistema era escrito em VBScript e DTS (Data Transformation Services). VBScript é uma linguagem que caiu em desuso, devido principalmente ao aparecimento do JavaScript. DTS é um conjunto de objetos e utilitários que permitem a automação de extração, transformação e operações de carregamento a partir de uma base de dados. DTS estão obsoletos, sendo substituído pelo SQL Service Integration Services. [3]

O sistema anterior tinha sete níveis de prioridade, sendo que o nível sete oferecia a maior largura de banda e o nível um oferecia a menor largura de banda. A limitação disto é que todos os produtos têm o mesmo tipo de QoS e independentemente do cálculo das regras, este aplica a largura de banda só com base no nível em que está e não com base no produto em si.

Outra limitação importante foi o facto de ser impossível alterar as regras de negócio e de terem de ser usadas as mesmas regras para todos os produtos. Um fator chave para a mudança também foi o facto do sistema anterior não guardar dados de tráfego para fins legais, que em qualquer momento podem ser pedidos pelas autoridades competentes.

Por último, o sistema não era capaz de dar uma resposta quando o SCE (Service Control Engine) não sabia a quem pertencia um determinado IP, o que provocava inconsistências no QoS aplicado a esse IP, visto que era aplicado um QoS por defeito.

2.2 Sistema Gestor de Base de Dados Relacional

Esta secção foi baseada na referência [4].

Sistemas gestores de base de dados (SGBDs) consistem num conjunto de aplicações de computador que são responsáveis pela gestão de uma base de dados. Um SGBD oferece uma interface ao utilizador de maneira a que este possa adicionar, alterar ou consultar dados previamente armazenados.

Existem diversos SGBDs disponíveis, mas neste projeto foi utilizado o MySQL, devido ao facto de ser o utilizado na empresa NOS Madeira e também porque era o que mais se adaptava as necessidades do projeto.

MySQL é um SGBD que utiliza linguagem SQL (Structed Query Language) como interface. As suas principais características são:

a) Portabilidade, visto que é suportada por qualquer plataforma atual;
b) Compatibilidade, porque existem drivers e módulos de interface para diversas linguagens de programação;
c) Excelente desempenho e estabilidade;
d) Pouco exigente no que diz respeito a recursos de hardware;
e) Facilidade de utilização;
f) É open source;
g) Contempla a utilização de vários motores de armazenamento (Storage Engines) como MyISAM, InnoDB, Falcon, BDB, etc.;
h) Suporta transações;
i) Suporta Triggers;
j) Suporta Cursors;
k) Suporta Stored Procedures e Functions;
l) Replicação é facilmente configurável;
m) Disponibilidade de interface gráfica de fácil utilização.

2.2.1 Mecanismos de armazenamento (Storages Engines)

Esta secção teve como base a referência [5].

Um motor de armazenamento é um componente de software subjacente que um sistema gestor de base de dados (SGBD) usado para criar, ler, atualizar e excluir dados a partir de uma base de dados.

Existem inúmeros motores de armazenamento, tais como: Aria, BlitzDB, Falcon, InnoDB, MyISAM, etc. Neste documento iremos destacar dois dos mais usados que é o InnoDB e MyISAM.

2.2.1.1 InnoDB vs. MyISAM

InnoDB e MyISAM são os motores de busca mais usados atualmente. As suas principais diferenças são:

a) InnoDB é mais novo que MyISAM;
b) InnoDB é mais complexo que MyISAM;
c) InnoDB é mais estrito na integração de dados que o MyISAM;
d) InnoDB implementa row-level lock quando insere e atualiza dado enquanto MyISAM implementa table-level lock;
e) InnoDB suporta transações e MyISAM não suporta;
f) MyISAM suporta índice de procura no total do texto e o InnoDB não suporta.

Seguidamente é abordado as vantagens e desvantagens destes dois motores de armazenamento.

Vantagens do InnoDB:

a) InnoDB é utilizado quando a integridade dos dados é uma prioridade, porque este mecanismo cuida dos dados com ajuda de restrições de relacionamento e transações;
b) Rápido na escrita intensiva (inserções, atualizações) de tabelas, visto que este mecanismo utiliza bloqueio ao nível de linha (row-level lock) e apenas realiza alterações na mesma linha que está sendo inserida ou atualizada.

Desvantagens do InnoDB:

a) Consome mais requisitos do sistema, como a RAM;
b) Sem indexação de texto completo.

Vantagens do MyISAM:

a) Mais simples para desenhar e criar, logo é melhor para principiantes. Não se preocupa com as relações externas da tabela;
b) Especialmente bom para tabelas com leitura intensiva (select).

Desvantagens do MyISAM:

a) Não existe nenhuma verificação da integridade dos dados;
b) Não suporta transações.

2.3 Serviços Web (Web Services)

Esta secção foi baseada na referência [6].

Os serviços web surgem como uma solução para a integração de sistemas e na comunicação entre aplicações diferentes. Estes serviços permitem que as aplicações implementadas atualmente possam interagir com aplicações já existentes. Os serviços web trocam dados em formato XML (Extensible Markup Language), que é uma linguagem universal, permitindo assim a comunicação de aplicações implementadas em linguagens de programação diferentes.

As empresas ganham muito na utilização dos serviços web, porque permitem obter processos mais ágeis e ter uma comunicação entre os sistemas mais eficaz. Esta comunicação também é sobretudo dinâmica e segura, visto que, é feita sem a intervenção humana.

2.3.1 Tecnologias Utilizadas

Os serviços web utilizam três principais tecnologias, que são SOAP, WSDL e UDDI.

Seguidamente é apresentado um breve resumo das três tecnologias.

2.3.1.1 SOAP (Simple Object Access Protocol)

Esta secção foi baseada na referência [7].

SOAP é um protocolo de comunicação baseado em XML que permite realizar comunicação entre aplicações através da internet. Tem a particularidade de ser independente de qualquer sistema ou linguagem de programação. Permite contornar facilmente as firewalls.

SOAP é importante para o desenvolvimento de aplicações, porque permite a comunicação entre aplicações web. Este protocolo surge para permitir a realização da comunicação entre aplicações sobre HTTP, que é suportado por todos os navegadores e servidores, tornando-se assim independente de qualquer sistema operativo, com diferentes tecnologias e linguagens de programação.

Este protocolo é composto por três elementos: envelope, cabeçalho (header em inglês) e corpo (body, em inglês), como ilustrado na figura 1.

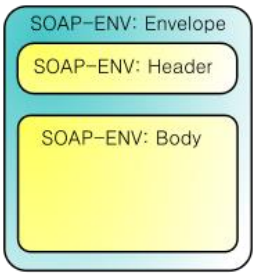

Figura 1 - Estrutura SOAP

O elemento envelope é o elemento raiz do SOAP, e é responsável por definir um documento XML como mensagem SOAP. Na figura 2 temos um exemplo de um envelope SOAP. [8]

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

Figura 2 - Elemento envelope do SOAP

O elemento opcional cabeçalho (header), contém informações específicas da aplicação sobre a mensagem SOAP. Se este elemento existir, este deve ser o primeiro elemento filho do elemento Envelope. Na figura 3 está um exemplo deste elemento. [9]

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">234
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

Figura 3 - Elemento header do SOAP

O elemento corpo (body), contém a mensagem SOAP a ser enviada ao destinatário. [10] Na figura 4 encontramos um exemplo:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Figura 4- Elemento body do SOAP

No exemplo presente na figura 4, trata-se de um pedido do preço das maçãs. O getPrice é um elemento específico da aplicação e não faz parte da estrutura da mensagem SOAP em si. De seguida, a figura 5 mostra uma possível resposta por parte do servidor. [10]

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Figura 5 - Exemplo de uma resposta Soap

2.3.1.2 WSDL (Web Services Description Language)

Esta secção foi baseada na referência [11].

WSDL é um documento XML usado para descrever serviços web e também serve para localiza-los. No conteúdo deste documento podemos encontrar a descrição completa de um determinado serviço web: especificação da localização do serviço (URL) e as operações ou métodos presentes no serviço. Na figura 6 encontramos um exemplo de documento WSDL.

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

Figura 6 - Documento WSDL

No elemento <types> é onde encontramos as definições de tipos de dados usados no serviço web, no elemento <message> é onde temos a definição dos dados que vão ser comunicados, no elemento <portType> encontramos o conjunto de operações suportadas e no elemento <binding> é a especificação do protocolo e o formato dos dados para um tipo de porta em particular.

Podemos encontrar mais elementos num documento WSDL, como elementos de extensão, e um elemento de serviço que torna possível agrupar as definições de vários serviços web num único documento WSDL.

2.3.1.3 UDDI (Universal Description, Discovery and Integration)

UDDI é uma framework de plataforma independente para descrever serviços web, descobrindo as empresas, e a integração de serviços de negócios usando a internet. UDDI é um diretório para armazenar informações sobre os serviços web que comunicam através do protocolo SOAP. [12]

UDDI pode trazer vários benefícios para uma empresa, tais como: torna possível descobrir o negócio certo dos milhões encontrados online; permite obter novos clientes e aumentar o acesso aos clientes existentes; permite que o cliente tenha uma participação rápida na economia global através da internet; permite descrever serviços e processos de negócio a partir de programação num ambiente único, aberto e seguro. [12]

2.4 Metodologia de desenvolvimento SCRUM

Esta secção, foi baseada na totalidade na referência [13].

SCRUM é uma framework iterativa e incremental de desenvolvimento ágil de software, para gerir a implementação de produtos. Esta define uma estratégia de desenvolvimento do produto, onde a equipa de desenvolvimento trabalha para alcançar um objetivo comum.

O princípio principal do SCRUM é que durante o projeto os clientes possam mudar os objetivos definidos, ou seja, os requisitos do projeto podem ser alterados durante o desenrolar do projeto.

De seguida irá ser abordado os papéis e os eventos na metodologia SCRUM.

2.4.1 Papéis do SCRUM

No SCRUM existem três papéis principais e um conjunto de papéis auxiliares. Os papéis principais são: *Product Owner*, *Development Team* e *Scrum master*.

Product Owner representa as partes interessadas, sendo o representante dos clientes. Ele é o responsável por garantir que a equipa oferece um valor ao negócio. O Product Owner descreve itens centrados no cliente, classifica e atribui uma prioridade e adiciona-os ao *product backlog*. A principal função do Product Owner é a comunicação e tem a capacidade de transmitir a prioridade aos membros da equipa, o que os torna muito importantes na melhor orientação do projeto.

Development Team (equipa de desenvolvimento, em português), é responsável pela entrega dos incrementos potencialmente utilizáveis do produto final de cada sprint. A equipa é normalmente composta entre três a nove elementos com habilidades multifuncionais que realizam o trabalho real.

Scrum Master é responsável pela remoção de impedimentos à capacidade da equipa de desenvolvimento entregar os objetivos do produto e resultados. Ele não é um líder de equipa ou gestor de projeto mas age como intermediário entre a equipa de

desenvolvimento e quaisquer distrações que surgem no decorrer do projeto. Este garante que o processo SCRUM corra como planeado, tendo a responsabilidade pela implementação de regras em todo o processo, assegurando motivação e áreas de melhoria por parte da equipa.

No contexto deste projeto, o Product Owner e o Scrum Master foram a mesma entidade, que neste caso foi a empresa NOS Madeira.

2.4.2 Eventos do SCRUM

No SCRUM existem três tipos de eventos: Sprint, Meetings e Extensions.

Sprint é visto como a unidade básica do Scrum onde está restrito uma duração específica, que costuma ser fixada previamente para cada sprint, sendo normalmente entre uma semana e um mês. Cada sprint começa por uma reunião de planeamento, em que o objetivo é a definição de um sprint backlog, onde é identificada as tarefas para o sprint. O sprint é encerrado por uma reunião de revisão e retrospetiva, apontando aspetos a melhorar, sendo todo o processo revisto e apresentado aos interessados.

Meetings (reuniões, em português), podem ser de três tipos distintos: Sprint planning meeting (reunião de planeamento do sprint), Daily scrum meeting (reunião diária de Scrum) e as End Meetings (reunião finais).

- ***Sprint planning meeting***, são reuniões marcadas no início de cada sprint com o objetivo de selecionar o trabalho a ser feito, produzem o Sprint Backlog, identificação e comunicação de como grande parte do trabalho irá ser feito.

- ***Daily Scrum meetting,*** são reuniões realizadas diariamente com objetivo de analisar as atualizações realizadas nesse mesmo dia. Também é visto se existe algum tipo de impedimento naquele sprint, por parte do Scrum Master.

- ***End Meetings,*** são reuniões realizadas no final de cada sprint com o intuito de rever o trabalho que foi concluído e o que ficou em atraso. Esta reunião também serve para os elementos da equipa fazer uma revisão do sprint anterior com o objetivo de encontrar aspetos a melhorar no próximo sprint.

Extentions, pode ser um Backlog refinement ou um Scrum of scrums.

- ***Backlog refinement*** é o processo em curso de revisão de itens do product backlog, e verificar se os mesmos estão devidamente priorizados e preparados de forma clara e executável pela equipa de desenvolvimento.

- ***Scrum of scrums,*** é uma técnica que permite escalar um Scrum para várias equipas que trabalham no mesmo produto, permitindo assim as equipas discutam o seu trabalho, sabendo coordenar a entrega de software, especialmente em áreas de sobreposição e integração.

No contexto deste projeto: As ***sprint planning meetings***, ocorriam periodicamente, onde eram definidos a conclusão de processos por data. A ***dayly scrum meeting***, não foram realizadas, visto não ser possível estar presente todos os dias na NOS Madeira. O ***daily scrum meeting*** era destinado aos elementos da equipa de desenvolvimento, o que neste caso é composta por um único elemento. A extensão ***Backlog Refinment*** e ***Scrum of scrums*** não foram utilizadas neste projeto.

2.4.3 Artefactos do SCRUM

No Scrum existem cinco artefactos que são: product backlog, sprint backlog, product increment, sprint burndown chart e release burndown chart.

- **Product backlog,** é uma lista ordenada de requisitos que é mantido por um produto. Que consiste de recursos, correções de erros, requisitos não-funcionais e o que precisa ser feito, a fim de entregar com sucesso um produto viável. O *product backlog*, é usado para verificar os pedidos de modificação de um produto e garantir que a equipa de desenvolvimento produza um produto que maximize o benefício do negócio para o cliente. Resumindo *product backlog* é uma lista priorizada de requisitos de alto nível.

- **Sprint backlog,** é uma lista de tarefas para a esquipa de desenvolvimento realizar durante um sprint.

- **Product Increment,** é a soma de todos os itens presentes no *product backlog*, que estão concluídos durante um sprint e sprints anteriores.

- **Sprint burndown chart,** é um gráfico exibido publicamente, mostrando o trabalho restante no *sprint backlog*. Este é atualizado todos os dias e dá uma visão simples do progresso do sprint.

- **Release burndown chart,** é um gráfico que é desenhado no final de um sprint, com o objetivo de mostrar o progresso de desenvolvimento do produto à equipa de desenvolvimento, ajudando a equipa a verificar as tarefas que faltam realizar.

No que diz respeito aos artefactos do Scrum, o ***product backlog*** neste projeto corresponde a listagem de requisitos funcionais e não-funcionais. O ***sprint backlog*** não era definido num documento as tarefas a realizar, mas sim era definido o que estaria de estar pronto até uma determinada data. Os restantes artefactos não foram utilizados neste projeto.

2.5 Conclusão

Neste capítulo ficamos mais esclarecidos quanto à necessidade de implementar um novo sistema de controlo de largura de banda para a empresa NOS Madeira. Verificamos as limitações no sistema que existia anteriormente.

Foi feita uma breve análise dos sistemas gestores de base de dados e qual a sua função, os mecanismos de armazenamento existentes e as principais diferenças, e por fim o que eram os serviços web e as tecnologias relacionadas.

Na escolha do mecanismo de armazenamento verificamos a importância ao nível da performance, tendo verificado a finalidade da base de dados, se vai ser escrita ou lida intensamente. A nível dos serviços web, percebemos que a partir dos mesmos conseguimos aceder a funcionalidades de aplicações diferentes de forma eficaz e segura.

A metodologia SCRUM, não foi utilizada na íntegra, visto que esta metodologia é principalmente utilizada quando existe uma equipa de desenvolvimento, o que não se aplica neste projeto, atendendo a que a equipa de desenvolvimento era composta por um único elemento.

Capítulo 3 – Desenho e Modelação

Neste capítulo iremos analisar a arquitetura geral do sistema global e a modelação do subsistema implementado neste projeto, denominado *IP Mapping System*. Iremos tratar dos requisitos funcionais e não funcionais definidos para o subsistema em questão e discutir alguns diagramas que irão ajudar a descrever o funcionamento deste subsistema.

Ao nível de modelação e requisitos, este subsistema foi sofrendo algumas alterações ao longo do projeto, visto que houve necessidade de adicionar ou retirar algumas funcionalidades relevantes.

3.1 Arquitetura Geral do Sistema

O sistema que foi proposto implementar, tem o nome IP Network Usage Accounting, e tem como objetivo contabilizar o tráfego realizado pelos clientes da NOS Madeira e posteriormente aplicar um QoS ao produto de cada cliente de maneira a que este tenha uma utilização responsável do ISP. Sendo assim, este projeto é composto por três subsistemas: *IP Mapping System*, *Accounting System* e *Policy Server*.

O primeiro subsistema, *IP Mapping System*, é responsável pela recolha dos dados de tráfego do cliente, que posteriormente são associados com dados dos equipamentos, de maneira a saber que equipamento/cliente fez aquele tráfego e de seguida envia esses dados para o segundo subsistema *Accounting System*.

O segundo subsistema, *Accounting System*, é responsável por realizar cálculos com base nos dados de tráfego recolhido pelo primeiro subsistema, e aplicar o QoS de acordo com os cálculos efetuados, e posteriormente envia esses mesmos dados para o terceiro subsistema, *Policy Server*.

O terceiro subsistema, *Policy Server*, é responsável por aplicar o respetivo QoS aos clientes da rede, com base nos dados recebidos do segundo subsistema, *Accounting System*.

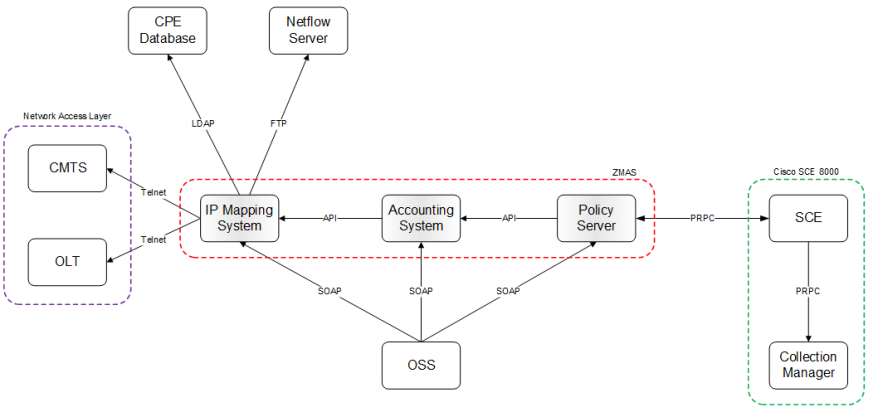

Figura 7 - Esquema componente-conetor do sistema versão 1

O esquema apresentado na figura 7 sofreu algumas alterações ao longo das reuniões realizadas com a NOS Madeira, onde ficou decidido que a recolha dos dados do CPE Database, CMTS e OLT não seriam feitas separadamente, mas seriam fornecidas de uma só vez através de um serviço web disponibilizado pelo OSS. Na figura 8, está apresentado o novo esquema.

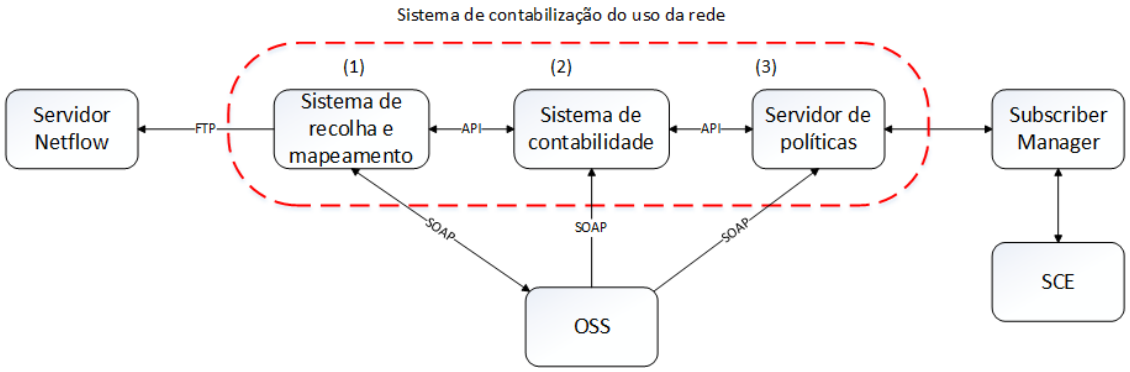

Figura 8 - Esquema componente-conector versão 2

De seguida encontra-se uma breve explicação de cada componente dos diagramas da figura 7 e 8.

Netflow Server, é onde são armazenados dados de tráfego, para serem analisados/processados. Daqui conseguimos ter conhecimento da quantidade de tráfego que um determinado IP realizou num determinado período.

CPE Database, tem informações sobre todos os CPEs e MAC Adresses da rede. É a partir desta informação que conseguimos fazer a associação do tráfego com um determinado cliente/equipamento.

CMTS (Cable Modem Termination System), é um componente que troca sinais digitais com cable modems numa rede de cabos. [14]

OLT (Optical Line Terminal), é um equipamento localizado no edifício principal do ISP e é o ponto final de uma rede ótica passiva (PON). [15]

OSS (Operational Support System), é um conjunto de aplicações que irão ajudar nas comunicações do ISP, como, monitorizar, analisar e gerir uma rede de telefones ou computadores. [16]

SCE (Service Control Engine), é um equipamento de rede especialmente desenhado para implementações de nível de operadora que exigem aplicações estáveis de alta capacidade, classificação baseada em secção e controlo do tráfego IP ao nível de aplicação por subscritor. [17]

3.2 Subsistema IP Mapping System

O *IP Mapping System*, é o primeiro subsistema deste projeto e tem como objetivo recolher as informações de tráfego efetuado pelos clientes da NOS Madeira, para posteriormente enviar esses mesmos dados para o segundo subsistema, *Accounting System*.

A recolha dos dados é feita em três etapas. A primeira etapa corresponde a recolha de dados do *Netflow Server*, onde conseguimos saber que um determinado IP realizou o respetivo tráfego num período de 15 minutos. Recolhida esta informação, passamos para a segunda etapa que corresponde a recolha de dados dos sistemas de informação que nos permite saber os IPs que estão atribuídos aos clientes. Sabendo os dados de tráfego de um determinado IP, e sabendo os IPs que estão atribuídos aos equipamentos dos clientes, a terceira etapa corresponde fazer a junção dos dados a partir do IP, assim ficando a saber os dados de tráfego realizado por um determinado cliente.

Os dados recolhidos pelo subsistema *IP Mapping System*, têm de ser enviados para o subsistema *Accounting System* para este poder realizar os cálculos necessários para atribuir um determinado QoS a um respetivo MAC Address.

Os dados recolhidos por este subsistema ficam armazenados numa base de dados relacional, onde podem ser consultados os dados de tráfego de um determinado cliente numa determinada data/hora.

3.2.1 Arquitetura do subsistema IP Mapping System

A arquitetura deste subsistema consiste em quatro módulos principais e um módulo que é responsável pelo fluxo de execução. Esses quatro módulos são: módulo de recolha de dados de tráfego, módulo de recolha de dados dos clientes, módulo de mapeamento dos dados, módulo de envio de dados.

Módulo de recolha de dados de tráfego, é o módulo responsável por fazer a recolha de dados do Netflow server, que corresponde aos dados de tráfego dos clientes num período de 15 minutos.

Módulo de recolha de dados dos clientes, é responsável pela recolha de dados dos sistemas de informação, para sabermos os IPs atribuídos aos equipamentos dos clientes e também termos informação do cliente.

Módulo de mapeamento dos dados, é o responsável por fazer o mapeamento dos dados que foram recolhidos, isto é, a partir desses dados irá fazer a junção dos dados recolhidos do Netflow server e dos dados dos sistemas de informação, conseguindo assim saber o tráfego realizado por um determinado MAC Address num período de 15 minutos.

Módulo de envio dos dados, é responsável por enviar os dados para o Accounting System. O envio dos dados é feito a partir de um serviço web disponibilizado pelo Accouting system.

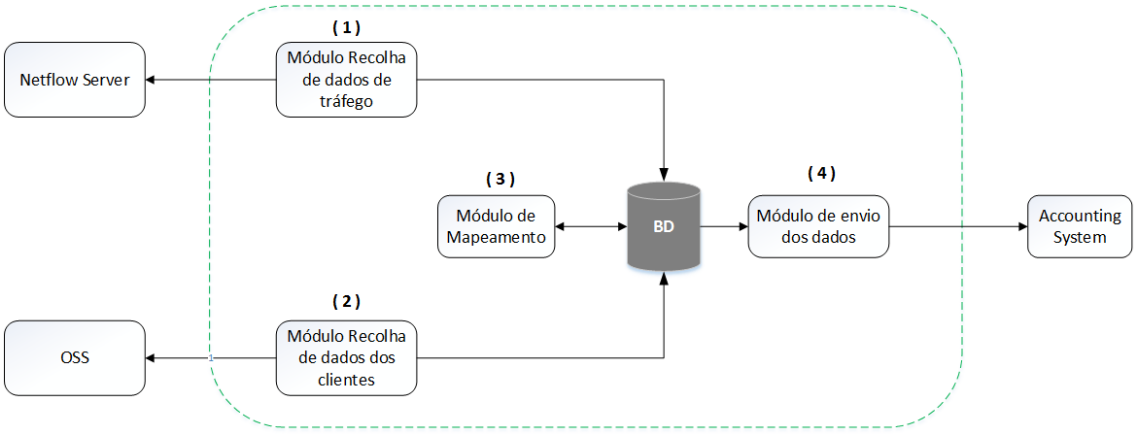

Figura 9 - Arquitetura do subsistema IP Mapping

A figura 9 ilustra a arquitetura deste subsistema.

Neste subsistema *IP Mapping* tudo funciona em redor da base de dados, onde os dados recolhidos do Netflow Server (1) e do OSS (2) são guardados nas tabelas correspondentes. Posteriormente é feita a junção dos dados pelo módulo de mapeamento (3) e por fim, os dados são guardados numa tabela para depois serem enviados (4) para o segundo subsistema *Accounting System*.

3.2.2 Diagrama de atividades do IP Mapping System

Na figura 10 está ilustrado o diagrama de atividades deste sistema, onde se pode perceber o funcionamento deste sistema. A execução deste sistema é feita automaticamente a cada 15 minutos, que corresponde ao período que o Netflow gera novos dados. A execução deste sistema é feita basicamente em quatro passos, nomeadamente:

1. Começa sempre pela recolha de dados do Netflow server e o armazenamento desses dados numa tabela da base de dados.

2. Seguidamente faz a recolha dos dados dos sistemas de informação através do serviço web disponibilizado, onde de seguida guarda os dados na tabela correspondente da base de dados.

3. Depois é feita a junção dos dados recolhidos. Esta junção é feita a partir do atributo IP que é o atributo em comum nas duas tabelas e é o que permite associar o tráfego realizado a um determinado cliente/equipamento. Os dados gerados dessa junção são armazenados em duas tabelas distintas. Isto é feito assim porque uma das tabelas servirá de histórico, onde a qualquer momento pode ser feita pesquisas para saber numa determinada data qual o cliente que

tinha um determinado IP. A tabela contém os mesmos dados, mas são apagados logo depois de enviar os mesmos para o *Accounting System*.

4. E por fim, os dados provenientes dessa junção são enviados para o Accounting System. Logo após o envio dos dados para *Accounting System*, este indica se recebeu os dados, e se a resposta for positiva os dados da tabela correspondente aos dados enviados é apagada, caso contrário termina o ciclo sem apagar os dados, que serão enviados juntamente com os dados do ciclo seguinte.

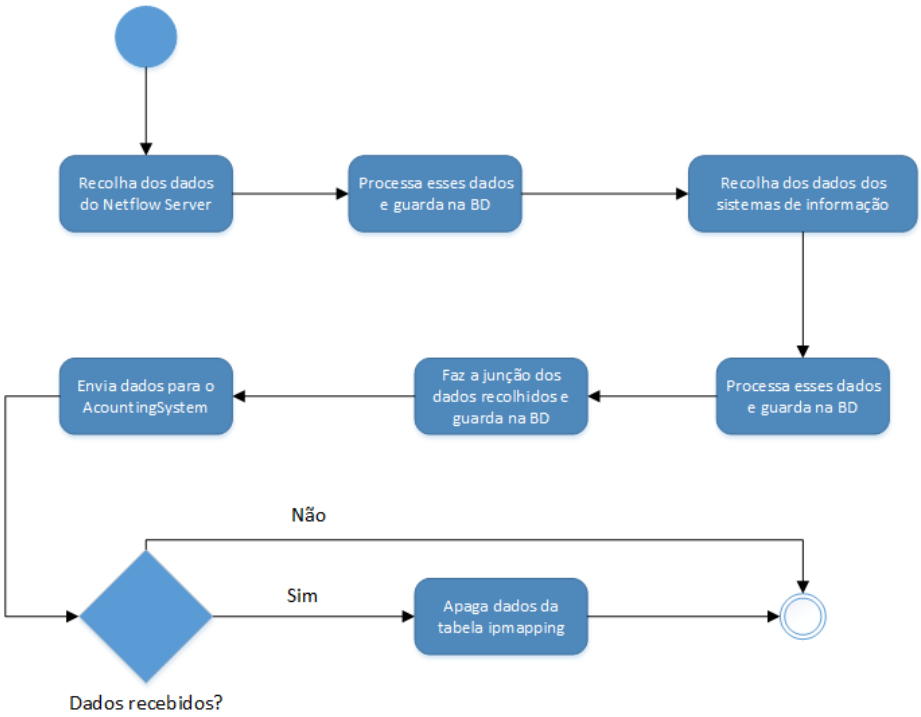

Figura 10 - Diagrama de Atividades do IP Mapping System

3.3 Requisitos

Aqui serão definidos os requisitos funcionais e não-funcionais do sistema. Estes sofreram algumas alterações ao longo do projeto, nomeadamente no que diz respeito a recolha de dados. Inicialmente a recolha de dados era feita em quatro etapas:

1. Recolha de dados do *Netflow Server*, onde tínhamos informações do tráfego efetuado por um determinado IP.
2. Recolha de dados dos equipamentos dos clientes (OLT e CMTS), onde tínhamos conhecimento do MAC Adress do modem do cliente e o IP atribuído.
3. Recolha de dados do CPE, onde tínhamos conhecimento do MAC Address do cable modem, o número do cliente e o serviço (produto e serviço do cliente).
4. Corresponde ao mapeamento de todos os dados recolhidos.

Ao longo das reuniões na NOS Madeira, ficou decidido que não haveria recolha de dados dos equipamentos dos clientes nem do CPE Database, mas sim a recolha de dados de um único sítio, a partir de um serviço web disponibilizado pelo departamento de sistemas de informação. Sendo assim, as etapas um e dois anteriormente descritas passaram a corresponder a uma só etapa.

3.3.1 Requisitos Funcionais

1) O sistema deve ser capaz de recolher dados (contidos num ficheiro) do Netflow server e interpretar esses dados. Os dados presentes nesse ficheiro serão:
   a) IP origem;
   b) IP destino;
   c) Tráfego efetuado;

2) O sistema deve ser capaz de recolher dados dos sistemas de comunicação a partir de um serviço web e ser capaz interpretar esses dados. Os dados recolhidos são:
   a) IP do cliente (CPE);
   b) MAC Address do modem;
   c) MAC Address do cable modem;
   d) Número do cliente;
   e) Serviço (Produto e perfil de utilizador;

3) O sistema deve ser capaz de interagir com o módulo "Accounting System":
   a) O sistema deverá enviar para este módulo os seguintes dados:
      a) IP;
      b) Tráfego efetuado;
      c) MAC Address do modem;
      d) Período;
      e) Serviço;
      f) Nº do cliente;

   b) O sistema deve garantir que os dados enviados para o "Accounting System" não são dados repetidos;
   c) O sistema deve permitir as operações pull e push para trocar dados com o "Accounting System";
   d) O sistema deve estar sincronizado com o módulo "Accounting System";

4) O sistema deve guardar os dados numa base de dados relacional;

5) O sistema deve ter operações de consulta a base de dados:
   a) Dado um IP e um instante de tempo, o sistema deve ser capaz de devolver o IP e o MAC Address correspondente;
   b) Dado o número de cliente ou MAC Address ou IP e um intervalo de tempo, o sistema deve ser capaz de devolver o tráfego feito, o IP origem e o IP destino.

6) O sistema deve disponibilizar todas as suas funcionalidades como serviços:
   a) Estes serviços são disponibilizados através do protocolo SOAP;

3.3.2 Requisitos Não Funcionais

1) O sistema deve ser implementado em PHP, com recurso à Framework "Zend";
2) O sistema deve ser modular, de forma a permitir escalabilidade e facilitar a modificação do sistema;
3) O sistema deve efetuar um ciclo completo de processamento em menos de 15 minutos.

3.4 Conclusão

Neste capítulo foi abordada a arquitetura geral do sistema, onde vimos que o sistema é composto por três subsistemas: *IP Mapping System*, *Accounting System* e *Policy Server System*. A breve descrição de cada um permitiu perceber as funções de cada subsistema. Analisamos a arquitetura do primeiro subsistema, *IP Mapping System*, que é composto por quatro módulos: Módulo de recolha de dados de tráfego, Módulo de recolha de dados dos clientes, Módulo de mapeamento e Módulo de envio dos dados. A partir do diagrama de atividade percebemos a sua execução. Ainda analisamos os requisitos funcionais e não-funcionais definidos para este subsistema e analisamos a metodologia de desenvolvimento que foi o Scrum, embora com algumas diferenças.

Nos anexos encontram-se o diagrama de fluxo de dados e o diagrama de sequência. O diagrama de fluxo de dados, permite verificar os dados trocados entre os diversos subsistemas, assim como os equipamentos, nomeadamente o Netflow server e dados dos sistemas de informação. O diagrama de sequência, permite ver a sequência de execução de todo o sistema.

Capítulo 4 - Implementação

Neste capítulo irá ser abordada a implementação do subsistema IP Mapping System. Será especificada a implementação de cada módulo, assim como as ferramentas utilizadas para implementar este subsistema. Também, irá ter a especificação da base de dados, onde vamos descrever as tabelas existentes, replicação e partições na base de dados.

A nível de implementação, este projeto está dividido em quatro módulos principais, como especificado na secção 3.2.1. Um ciclo de processamento, é a execução total deste sistema, começando com a recolha de dados do Netflow server, a seguir a recolha de dados dos sistemas de informação, e tendo os dados recolhidos passa-se ao tratamento desses mesmos dados, onde é feita a junção dos dados, e por fim, esses mesmos dados são enviados para o Accounting System.

No que diz respeito à construção da base de dados, consiste numa tarefa que requer muita atenção, visto que, este subsistema trata grandes volumes de dados. Um dos objetivos deste subsistema é guardar os dados de tráfego, durante um ano para fins legais. Iremos verificar que foi implementada uma forma simples e eficaz de garantir a integridade e armazenamento dos dados e sempre salvaguardando a performance do sistema.

4.1 Estrutura de ficheiros do projeto

Neste projeto foram implementados sete ficheiros distintos. Ao longo desta secção será descrito a localização exata de cada ficheiro assim como o conteúdo de cada ficheiro. Ainda será analisado as pastas e o seu conteúdo, que sejam relevantes para o projeto.

4.1.1 Pasta principal do sistema

Na pasta **application**, é onde encontramos todos os ficheiros implementados, que serão descritos na secção 4.1.2. A pasta **cronlogs** é a pasta onde são guardados todos os logs do cron, ou seja, ao longo de cada ciclo de processamento é escrito o respetivo output da execução. A pasta **logs** é a pasta onde são guardados todos os logs deste projeto, isto é, todos os relatórios de erro, quando alguma falha ocorrer. As restantes pastas são pastas criadas pela Zend Framework.

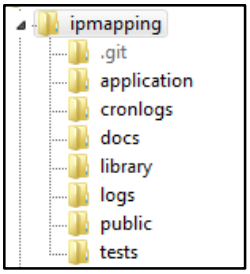

Figura 11 - Raiz do projeto

4.1.2 Pasta Application

Dentro da pasta **configs**, encontramos os ficheiros de configuração, onde neste caso encontramos dois ficheiros, denominados "application.ini" e "database.ini". O ficheiro "application.ini", é o ficheiro onde são colocadas as definições específicas do projeto, como caminhos (paths, em inglês), para a framework saber o caminho específico para alguns diretórios. O ficheiro "database.ini" é onde se encontram as definições do mysql e da base de dados.

Dentro da pasta **modules**, encontram-se os módulos implementados. No caso deste projeto encontra-se aqui somente o módulo responsável por disponibilizar o serviço web para consulta da base de dados. Dentro da pasta history, encontram-se duas pastas, **Controllers** e **Models**, que corresponde ao padrão MVC. O controlador chama-se "IndexController.php" e é onde encontramos as definições das ações SOAP e WSDL. O modelo corresponde ao ficheiro "Services.php" responsável por implementar a função checkclient, descrita na secção 4.4.9.

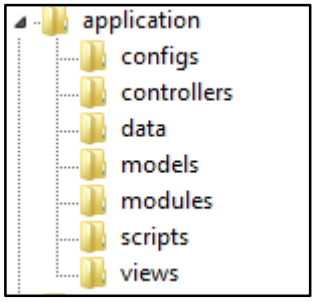

Figura 12 - Estrutura da pasta application

4.1.3 Pasta Scripts

A pasta **scripts** (figura 13) é a pasta principal deste projeto. Dentro desta pasta encontra-se uma pasta denominada **library** que contém todos os módulos do projeto. Esta pasta tem um ficheiro denominado "mapping.php" que corresponde ao ficheiro principal do sistema, descrito na secção 4.4.6. Ainda nesta pasta encontra-se um ficheiro denominado "lock.txt" que é o "lock file", descrito na secção 5.1.4.

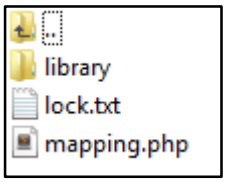

Figura 13 - Conteúdo da pasta scripts

4.1.4 Pasta library

Dentro desta pasta (figura 14) encontram-se quatro ficheiros, onde cada ficheiro corresponde a implementação de um módulo. O ficheiro "Netflow.php", contém o código do módulo de recolha de dados de tráfego, descrito na secção 4.4.2. O ficheiro

"Isdata.php", contém o código do módulo de recolha de dados dos clientes, descrito na secção 4.4.3. O ficheiro "Mapping.php" contém o código do módulo de mapeamento, descrito na secção 4.4.4. E por fim, o ficheiro "Senddata.php", contém o código do módulo de envio dos dados para o *Accounting System*, descrito na secção 4.4.5.

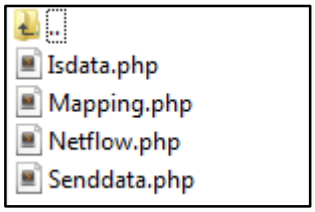

Figura 14 - Conteúdo da pasta library

4.2 Ferramentas Utilizadas

Para a implementação do sistema foi pedido que fosse utilizada a Zend Framework, que era a framework utilizada na NOS Madeira. Para o controlo de versões a ferramenta escolhida foi o GIT. Para a execução ocorrer automaticamente foi utilizada a ferramenta do Linux Cron, que funciona como as tarefas agendadas do Windows. Para instalação e configuração do servidor foi utilizado LAMP. Estas ferramentas são descritas ao longo desta secção.

4.2.1 Zend Framework 1.12

Zend Framework é uma framework open source para desenvolver aplicações web com PHP 5. Cada componente no Zend Framework é implementado com poucas dependências de outros componentes, obtendo assim um baixo acoplamento e permitindo assim os utilizadores utilizarem componentes individualmente. [18]

4.2.2 GIT

Controlo de versão é um sistema que regista alterações num ficheiro ou conjunto de ficheiros ao longo do tempo para podermos relembrar versões específicas mais tarde. Isto permite-nos, ao longo do projeto quando pretendemos adicionar uma nova funcionalidade, salvaguardar o projeto na eventualidade de ocorrer algum problema durante a implementação, podendo assim reverter uma versão anterior do projeto. [19]

O sistema de controlo de versão escolhido para este projeto foi o GIT. Este sistema destaca-se dos outros sistemas principalmente pelo facto de este armazenar e pensar nas informações muito diferente do que outros sistemas. A principal diferença do GIT e outros sistemas de controlo de versões, é na maneira que o GIT trata as informações como um conjunto de ficheiros e as mudanças feitas num arquivo ao longo do tempo, enquanto outros sistemas armazenam informações como uma lista de mudanças baseada num ficheiro.

4.2.3 LAMP

LAMP é um conjunto de ferramentas open source utilizado com o objetivo de obter servidores web instalados e a funcionar. São ferramentas necessárias para o desenvolvimento de aplicações web. A sigla LAMP, significa Linux, Apache, MySQL e PHP. Para este projeto em concreto, foi utilizada a distribuição Debian, onde foram instaladas e configuradas as restantes ferramentas, seguindo os passos descritos em [20].

4.3 Base de dados

O mecanismo de armazenamento utilizado em todas as tabelas é o InnoDB, embora inicialmente fosse MyISAM. Foi decidido mudar para InnoDB principalmente por permitir transações e também porque estávamos a trabalhar em tabelas que eram escritas constantemente.

A base de dados é composta por quatro tabelas: netflow_data, is_data, ipmapping e ipmapping_history. A tabela netflow_data guarda os dados referentes à recolha de dados do netflow. A tabela is_data guarda os dados referentes à recolha de dados dos clientes. A tabela ipmapping guarda os dados da junção dos dados das tabelas netflow_data e is_data. A tabela ipmapping_history é uma tabela igual a ipmapping, mas que guarda os dados de todos os ciclos de processamento, sem nunca os apagar durante um ano. Isto deve-se ao facto de ser preciso armazenar dados de tráfego durante esse período para fins legais, porque pode ser pedida essa informação pelas autoridades competentes. Foi decidido criar esta tabela porque os dados da tabela ipmapping são apagados a partir do momento que são enviados para o Accounting System, por questões de performance.

Resumindo o processo de armazenamento na base de dados funciona da seguinte forma: os dados recolhidos do Netflow server são armazenados na tabela netflow_data, de seguida os dados são recolhidos dos sistemas de informação e são armazenados na tabela is_data, depois de ter os dados nestas duas tabelas é feita a junção dos dados das mesmas e esses dados são armazenados nas tabelas ipmapping e ipmapping_history, de seguida os dados armazenados em ipmapping são enviados para o *Accounting System* e apagados desta tabela. A figura 15 ilustra a estrutura da base de dados.

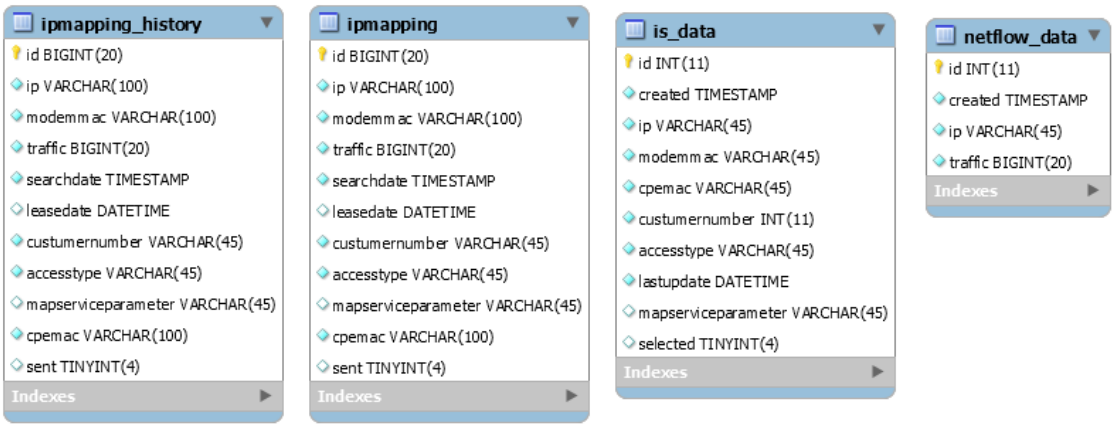

Figura 15 - Estrutura da base de dados

4.3.1 Tabela netflow_data

Na tabela netflow_data são armazenados os dados recolhidos do Netflow server. A partir desses dados é possível saber o tráfego efetuado por um determinado IP, num período de 15 minutos. Esta tabela além do atributo id, que serve para identificar o tuplo criado e o atributo created que corresponde a data de criação do tuplo, tem os seguintes atributos:

- ip, que corresponde ao IP que realizou um determinado tráfego.
- Traffic, que corresponde a quantidade em bytes de tráfego efetuado.

A chave primária desta tabela é o atributo id. Ao início de cada ciclo de processamento, os dados existentes nesta tabela são apagados, isto é, antes do início de uma nova recolha de dados do Netflow server, os dados existentes são eliminados.

4.3.2 Tabela is_data

A tabela is_data é onde são armazenados os dados que são recolhidos dos sistemas de informação. A partir destes dados, é possível saber os IPs atribuídos aos equipamentos dos clientes naquele momento. Esta tabela além dos atributos comuns, id e created, tem os seguintes atributos:

- ip, que corresponde ao IP atribuído ao cable modem/equipamento do cliente;
- modemmac, que é o MAC Address do cable modem do cliente;
- cpemac, que é o MAC Address do equipamento do cliente;
- custumernumber, que corresponde ao número do cliente em questão;
- accesstype, representa o produto do cliente para fins de cálculos;
- lastupdate, é a ultima data em que o IP foi atualizado;
- mapserviceparameter, representa o produto do cliente para fins de faturação;

A tabela is_data usa o atributo id como chave primária. Tal como acontecia na tabela netflow_data, os dados são eliminados quando se inicia um novo ciclo.

4.3.3 Tabela ipmapping

Na tabela ipmapping encontramos os dados provenientes da junção das duas tabelas anteriores, a tabela netflow_data e is_data. A junção é feita a partir do atributo ip que têm em comum, onde podemos saber a quantidade de tráfego em bytes, realizada por um determinado cliente (MAC Address), no período de 15 minutos. Os dados desta tabela são os que são enviados para o *Accounting System*. Depois de enviar com sucesso os dados são eliminados, com exceção do último tuplo criado. Além dos atributos comuns (id), e os já falados (ip, modemmac, traffic, customernumber, accesstype, mapserviceparameter e cpemac), esta tabela tem os seguintes atributos:

- searchdate, corresponde a data em que foi feita a recolha dos dados;
- leasedate, é a data em que o IP foi "libertado" daquele modem;

- sent, funciona como uma flag (0 ou 1), que indica se aquele tuplo já foi enviado.

4.3.4 Tabela ipmapping_history

A tabela ipmapping era escrita sempre com um grande volume de dados a cada 15 minutos, e era pedido para guardar os dados dos ciclos todos, durante um ano para fins legais. Mas isto iria trazer um problema relacionado com a performance do sistema, porque a seleção dos dados a enviar para o Accounting System tornava-se mais lenta, devido a quantidade de dados da tabela. Devido a isto, a solução encontrada foi criar uma tabela igual a ipmapping que apenas guarda os dados, assim sendo, as tabelas ipmapping e ipmapping_history guardam os dados, mas a tabela ipmapping depois de enviar os dados e receber confirmação do *Accounting System*, irá apagar os dados contidos nela e a tabela ipmapping_history manterá sempre os dados de todos os ciclos de processamento.

4.3.4.1 Partições

Para aumentar a performance da base de dados foram criadas partições para facilitar as pesquisas nas tabelas da base de dados. Neste caso foram criadas partições na tabela ipmapping_history por mês, isto significa que os dados a medida que são guardados são divididos em diversas partições com base no mês do atributo "searchdate", como podemos ver na figura 16.

```
PARTITION BY HASH (MONTH(searchdate))
PARTITIONS 12;
```

Figura 16 - Criação das partições

Nas outras tabelas não houve necessidade de criar estas partições, visto que as mesmas não eram alvo de muitas pesquisas.

4.3.4.2 Índices

Os índices servem para aumentar a velocidade das operações de consulta da base de dados. Na base de dados do *IP Mapping System*, estes índices foram criados na tabela ipmapping_history, que é a tabela onde o número de consultas é elevado. Na figura 17 podemos ver em que atributos foram criados esses índices.

Table	Non_unique	Key_name	Seq_in_index	Column_name
ipmapping_history	0	PRIMARY	1	id
ipmapping_history	0	PRIMARY	2	searchdate
ipmapping_history	1	modem	1	modemmac
ipmapping_history	1	ip	1	ip
ipmapping_history	1	data	1	searchdate

Figura 17 - Índices criados na tabela ipmapping_history

4.4 Implementação do sistema

Nesta secção iremos relatar como foram implementados os módulos do sistema. O sistema é composto por quatro módulos como foi abordado na secção 3.2.1. De seguida é descrito o funcionamento de cada um dos módulos, as funções implementadas e os serviços web que são fornecidos.

Este projeto não utilizou a metodologia MVC, visto que a mesma não exigia nenhuma interface do utilizador perante as diversas funcionalidades do sistema, logo foi decidido contruir o sistema a partir de scripts PHP, que são chamados a partir de um ficheiro que definia a sequência de execução do sistema.

Inicialmente as funcionalidades do sistema estavam todas associadas a um serviço web, ou seja, quando queríamos executar o sistema tinha que ser executado o URL do serviço web, onde estava definido todos os procedimentos a realizar, mas este processo era muito lento daí a decisão da criação de scripts onde a execução era muito mais rápida.

4.4.1 Funções gerais implementadas

Neste sistema foi criada uma função utilizada em todos os módulos, chamada InsertDB, responsável por inserir dados numa tabela da base de dados. A mesma recebe como parâmetros duas variáveis: $data, que corresponde a um array que contém os dados a inserir; e $table, que corresponde a uma string que contém o nome da tabela na qual queremos inserir os dados. A função retorna verdadeiro se a inserção for bem sucedida, caso contrário retorna falso. Na figura 18 encontramos a implementação da função.

```
public function insertDB($data, $table) {
    $db = Zend_Db_Table::getDefaultAdapter();
    $db->beginTransaction();
    try {
        //SQL Query to insert data in table
        $query = "INSERT IGNORE INTO $table (modemmac, ip, cpemac, custumernumber,
            accesstype, lastupdate) VALUES ";

        foreach ($data as $value) {
            $value1 = $value['modemmac'];
            $value2 = $value['ip'];
            $value3 = $value['cpemac'];
            $value4 = $value['custumernumber'];
            $value5 = $value['accesstype'];
            $value6 = $value['lastupdate'];

            $query .= "('$value1', '$value2', '$value3', '$value4', '$value5', '$value6'),";
        }
        $query = substr($query, 0, -1);
        $db->query($query);

        $db->commit();
        return true;
    } catch (Zend_Exception $e) {
        $db->rollBack();
        throw new Exception($e);
        return false;
    }
}
```

Figura 18 - Função InsertDB

Na figura 19, encontramos um exemplo de uso da função InsertDB, onde nesse caso foi a inserção dos dados nas tabelas ipmapping e ipmapping_history da base de dados.

```
$this->insertDB($result, 'ipmapping');
$this->insertDB($result, 'ipmapping_history');
```

Figura 19 - Exemplo de uso da função InsertDB

4.4.2 Módulo de recolha de dados de tráfego

Este módulo é responsável por fazer a recolha de dados do Netflow Server e armazenar os mesmos na tabela netflow_data da base de dados. A recolha dos dados é feita a partir da função getflowData. De seguida será descrita a implementação deste módulo.

O primeiro passo, é eliminar os dados presentes na tabela, fazendo uma simples query SQL que é responsável por apagar os dados da tabela netflow_data. De seguida inicia-se o acesso ao Netflow Server utilizando o protocolo FTP *(File Transfer Protocol)* onde será transferido um ficheiro que contém as informações de tráfego. Este processo é ilustrado na figura 20.

```
try {
    //Set up basic ftp connection to collector
    $conn_id = ftp_connect('url') or die("Couldn't connect to ftp server");

    // try to login
    if (ftp_login($conn_id, 'user', 'password')) {
        echo "Connection sucessfull\n";
    } else {
        echo "Couldn't connect\n";
    }

    $local_file = APPLICATION_PATH . '/data/dataflow.txt';
    $ftp_file = './netflow_export.txt';

    // try to download $server_file and save to $local_file
    if (ftp_get($conn_id, $local_file, $ftp_file, FTP_ASCII)) {
        echo "Successfully written to $local_file\n";
    } else {
        echo "There was a problem\n";
    }
    //Close the connection
    ftp_close($conn_id);
```

Figura 20 – Transferência do ficheiro do Netflow Server

Depois de termos transferido o ficheiro com os dados necessários, passamos a ler esse ficheiro. Essa leitura é feita a partir de uma função denominada parseData, que será especificada posteriormente. Depois de lidos os dados do ficheiro, os mesmos estão contidos num array e são inseridos na tabela netflow_data da base de dados através da função InsertDB, como ilustrado na figura 21.

```php
//Reading file and insert data in DB
$file = file(APPLICATION_PATH . '/data/dataflow.txt');
$data = $this->parseData($file);
$this->insertDB($data, 'netflow_data');
```

Figura 21 - Leitura do ficheiro e inserção na base de dados

4.4.2.1 Funções Implementadas

Neste módulo foi implementada uma função denominada parseData que tem o objetivo de fazer a leitura do ficheiro transferido do Netflow Server. Esta leitura é feita a partir da definição de uma expressão regular de acordo com a estrutura do ficheiro. Isto foi feito desta forma, principalmente porque a estrutura do ficheiro é exatamente a mesma em todas as recolhas, embora seja uma desvantagem no futuro se houver uma mudança da estrutura do ficheiro.

```php
public function parseData($file) {
    $start_line = 19; //define the first search line
    $ip_pos = 0; //ip position
    $number_pos = 2; //octets position
    $fields = array(
        $ip_pos => 'ip',
        $number_pos => 'traffic'
    );

    $pattern = '/[.\d]+\s/i'; //regular expression
    $data = array();
    $index = 0;
    //search data according the regular expression
    for ($i = $start_line; $i < sizeof($file); $i++, $index++) {
        preg_match_all($pattern, $file[$i], $values);
        for ($j = 0; $j < sizeof($values[0]); $j++) {
            if ($j == $ip_pos || $j == $number_pos)
                $data[$index][$fields[$j]] = $values[0][$j];
        }
    }
    return $data;
}
```

Figura 22 - Função parseData

4.4.3 Módulo de recolha de dados dos clientes

O módulo dos sistemas de informação é o módulo responsável por fazer a recolha de dados dos sistemas de informação. A recolha de dados é feita a partir de um serviço web disponibilizado pelos sistemas de informação da NOS Madeira.

Ao chamar a função getIpMapping presente no serviço web disponibilizado, conseguimos obter os dados, que são serializados e guardados num ficheiro. Os dados são serializados de maneira a depois poder chama-los em qualquer lugar e converte-los facilmente naquilo que pretendemos.

```php
//Soap Server Client declaration
$client = new Zend_Soap_Client("http://ws.zonmadeira.pt/ipmappingws/wsdl");
$data = $client->getIpMapping();  // function to get data
$file = APPLICATION_PATH . '/data/isData.txt';
file_put_contents($file, serialize($data)); // write data in file
```

Figura 23 - Definição do cliente SOAP

Tendo os dados escritos num ficheiro, estes são interpretados da seguinte forma: verificado que o ficheiro existe, é obtido o conteúdo do ficheiro e guardado num array de objetos, onde cada posição do array corresponde a um tuplo da tabela da base de dados. De seguida, é feito um mapeamento dos nomes dos atributos presentes no array para os atributos presentes na base de dados, chamando a função mappingFields, descrita na secção 4.4.3.1. Tendo feito o mapeamento, o sistema guarda os dados obtidos na base de dados. Todo este processo é ilustrado na figura 24.

```php
if (file_exists($file)) {
    $objData = file_get_contents($file);
    $obj = unserialize($objData);
    if (!empty($obj)) {
        $array = $obj->mapping->Struct;
        //mapping fields from file
        for ($i = 0; $i < sizeof($array); $i++) {
            $fields = $array[$i];
            $f = array($this->mappingFields($fields));
            $this->insertDB($f, 'is_data');
        }
    }
}
```

Figura 24 - Leitura do ficheiro e inserção dos dados na base de dados

4.4.3.1 Funções Implementadas

Neste módulo foi implementada a função mappingFields que recebe um array, e faz o mapeamento com os nomes dos atributos presentes no array para os nomes dos atributos da base de dados. Esta função teve de ser implementada porque havia necessidade de mudar o nome dos atributos vindos dos sistemas de informação, para os nomes dos atributos da tabela is_data da base dados. Esta função retorna um array, com os nomes dos atributos de acordo com a tabela da base de dados e os dados prontos a inserir. A figura 25 ilustra a implementação desta função.

```php
public function mappingFields($data) {
    $mappingFields = array(// my => theirs
        'modem_mac' => 'modemmac',
        'cpe_ip' => 'ip',
        'cpe_mac' => 'cpemac',
        'customer_nr' => 'custumernumber',
        'access_type' => 'accesstype',
        'last_update' => 'lastupdate',
    );
    $mappedData = array();
    foreach ($data as $index => $row) {
        if (array_key_exists($index, $mappingFields))
            $mappedData[$mappingFields[$index]] = $row;
        else
            $mappedData[$index] = $row;
    }

    return $mappedData;
}
```

Figura 25 - Função mappingFields

4.4.4 Módulo de mapeamento dos dados

Este módulo é responsável por fazer o mapeamento dos dados, isto é, irá fazer a junção dos dados recolhidos no módulo Netflow e no módulo dos sistemas de informação. Está junção é feita através do atributo em comum nas duas tabelas, ip, através da qual conseguimos assim relacionar o cliente com o tráfego efetuado. Isto é conseguido fazendo um Select dos dados pretendidos das duas tabelas, com a condição (where), de ter o mesmo ip. Assim obtêm-se os dados que são guardados num array que depois são inseridos na tabela ipmapping e ipmapping_history da base de dados. A figura 26 ilustra este processo.

```php
//SQL query to join data
$sql = "SELECT netflow_data.ip, netflow_data.traffic, '$date' AS searchdate, is_data.ip,
    is_data.modemmac, is_data.cpemac, is_data.custumernumber, is_data.accesstype
     FROM netflow_data, is_data
     WHERE netflow_data.ip=is_data.ip";
$result = $this->ipMappingDb->fetchAll($sql);
$this->insertDB($result, 'ipmapping');
$this->insertDB($result, 'ipmapping_history');
```

Figura 26 - Mapeamento dos dados

4.4.5 Módulo de envio dos dados para o Accounting System

Este módulo é responsável por enviar os dados da tabela ipmapping da base de dados para o *Accounting System*. Isto é feito recorrendo a um serviço web fornecido pelo *Accounting System*. O primeiro passo foi fazer a definição do serviço web específico, como ilustrado na figura 27, onde é chamada a função pushBulk (figura 29), para enviar os dados correspondentes obtidos a partir de uma pesquisa a base de dados, como podemos ver na figura 28, onde são obtidos os dados da tabela ipmapping. Também é fornecido um serviço web que simplesmente retorna o último id recebido, conseguindo assim saber que dados eliminar da tabela ipmapping, assim garantindo que os dados eliminados são somente os dados já recebidos pelo *Accounting System*.

```
$client = new Zend_Soap_Client("http://netaccounting.nosmadeira.pt/ipmapping/index/wsdl");
```

Figura 27 - Definição do cliente SOAP

```
    //Select data of IPmapping
    $result1 = $this->ipMappingDb->fetchAll("SELECT id AS '0', ip AS '1', modemmac AS '2',
        traffic AS '3', searchdate AS '4', custumernumber AS '5', accesstype AS '6' "
            . "FROM ipmapping "
            . "WHERE sent='0' "
            . "ORDER BY id ASC "
            . "LIMIT 0, 40000");
```

Figura 28 - Query SQL para selecionar os dados a enviar

```
//Sent data to Accounting
try {
    $result = $client->pushBulk($result1);
    $lastId = $client->requestLastInsertedId();
    if ($result) {
        $this->ipMappingDb->delete('ipmapping', 'id < ' . $lastId);
        $this->ipMappingDb->update('ipmapping', array('sent' => 1), 'id = ' .$lastId);
    }
```

Figura 29 - Envio dos dados para o Accounting System

```
<portType name="Ipmapping_Model_ServicesPort">
    <operation name="pushBulk">
        <documentation>
            PushBulk method Receives an array of raw traffic
            data and inserts it into the database
        </documentation>
        <input message="tns:pushBulkIn"/>
        <output message="tns:pushBulkOut"/>
    </operation>
    <operation name="requestLastInsertedId">
        <documentation>
            RequestLastInsertedId method Returns the last
            inserted id from the traffic table
        </documentation>
        <input message="tns:requestLastInsertedIdIn"/>
        <output message="tns:requestLastInsertedIdOut"/>
    </operation>
</portType>
<message name="pushBulkIn">
    <part name="rawData" type="soap-enc:Array"/>
</message>
<message name="pushBulkOut">
    <part name="return" type="xsd:boolean"/>
</message>
<message name="requestLastInsertedIdIn"/>
<message name="requestLastInsertedIdOut">
    <part name="return" type="xsd:int"/>
</message>
```

Figura 30 - Excerto do documento WSDL do Accounting System

Na figura 30 pode ver-se um excerto do documento WSDL do *Accounting System* onde podemos ver os serviços web disponibilizados ao *IP Mapping System.* Esses serviços oferecem duas operações, uma denominada "pushBulk" e outra "requestLastInsertedId". Na tag <message> podemos ver o que cada operação recebe e o que retorna. A operação pushBulk, basicamente irá receber um array com os dados enviados, e retornar um booleano, que terá o valor true se os dados forem recebidos com sucesso ou false caso contrário. A operação "requestLastInsertedId", retorna o último id da tabela "traffic" do *Accounting System.*

Neste módulo teve de ser utilizado o mecanismo "lock file" para evitar que no decorrer da execução haja dois processos a enviar dados. Se um processo estivesse a enviar dados, e outro processo estivesse a ser executado este último já não iria enviar os dados, visto que, já existia um processo a realizar esse mesmo envio. Logo que o processo acabasse de enviar esses mesmos dados, desbloqueava o acesso ao ficheiro, permitindo assim um novo processo enviar os dados. Este assunto será aprofundado na secção 5.1.4.

4.4.6 Ficheiro principal do sistema

O ficheiro principal do sistema é onde são chamados todos os módulos, isto é, onde é definida a sequência de execução. A sequência é a seguinte (figura 31): primeiro chama a função getflowData, que corresponde à função declarada no módulo de Netflow, de maneira a recolher dados de tráfego; de seguida é chamada a função getisData, que é a função declarada no módulo dos sistemas de informação, para recolher os dados dos sistemas de informação da NOS Madeira; de seguida é realizada a junção dos dados recolhidos anteriormente, através de uma query que realiza a junção dos dados através do atributo ip; e por último, são enviados os dados para o Accounting System através da função sendData, presente no módulo de envio dos dados para o Accounting System.

```php
//Collect Netflow Data
$flow_time = $flow->getflowData(0);

//Collect Data for Information Systens
$isdata_time = $isdata->getisData($flow_time);

//Mapping data of Netflow and Information Systens
$mapping->joinData($isdata_time);

$lock = fopen('/var/vhost/ipmapping/application/scripts/lock.txt', 'w');
if (flock($lock, LOCK_EX | LOCK_NB)) {

    try {
        //Send Data to Accounting
        $send->sendData();
    } catch (Exception $e4) {
        file_put_contents(APPLICATION_PATH . '/../logs/logIpmapping.txt',
            "\n\n" . date('Y-m-d H:i:s') . " - Senddata Problem\n\n" . $e4, FILE_APPEND);
    }
}
```

Figura 31 - Ficheiro Principal do Ficheiro

4.4.7 Recuperação de falhas do sistema

A recuperação de falhas foi pensada principalmente para que a mesma não demorasse mais de 15 minutos. Assim sendo, a solução apropriada foi utilizar o tempo de execução do sistema para o sistema decidir se tinha tempo para executar novamente alguma parte do código que falhasse. Esta recuperação do sistema foi implementada em todos os módulos, com exceção do módulo de envio de dados para o Accounting System, onde o mesmo se ocorre um erro, mantém os dados na tabela ipmapping e envia-os no próximo ciclo de processamento. A figura 32 ilustra a estrutura base da recuperação de erros do sistema.

```php
try {

    //Module code here

    //End time and calculate execution time
    $time_end = microtime(true);
    $exec_time = $time_end - $time_start;

    //Return execution time
    return $exec_time;
} catch (Exception $ex) {
    //End Time and calculate execution time
    $time_end = microtime(true);
    $exec_time = $time_end - $time_start;
    //Wait 120 seconds and add this in execution time
    sleep(120);
    $exec_time = $exec_time + 120;
    //Verify if execution time not exceed 540 seconds, if not, call function again
    if($exec_time <= 540) {
        $this->getflowData($exec_time);
    }
    //If execution time exceed 540 seconds write problem in log
    else {
        //write in log file
        die;
    }
}
```

Figura 32 - Recuperação de falhas do sistema

Para recuperação de erros são usados os blocos "try" e "catch", que basicamente permitem que um determinado código contido em "try" seja executado e se alguma coisa falhar, entra dentro de "catch" e executa o código existente lá. Neste caso concreto o código do módulo em si está contido no bloco "try" e este tenta executar o código correspondente e verifica a hora quando inicia a execução utilizando a função disponibilizada pelo PHP, microtime e no final da execução calcula o tempo de execução e retorna esse mesmo valor, para que o próximo módulo a executar saiba qual o tempo de execução decorrido. No caso de alguma falha durante a execução, este passa para o código contido em "catch", onde calcula o tempo de execução decorrido até esse momento e de seguida faz um compasso de espera de dois minutos para então verificar se o tempo de execução excede ou não os nove minutos. Se exceder os nove minutos o sistema interrompe o processo e escreve o erro num ficheiro de log, e se não exceder os nove minutos este tenta executar a função novamente. O tempo limite de execução do sistema é nove minutos, devido à definição do cron do Linux. O Netflow Server gera novos dados a cada 15 minutos, mas pode haver algum atraso de alguns minutos, daí o cron ser definido sempre com cinco minutos de atraso para existir margem para o

Netflow Server criar o ficheiro. Sendo assim para a execução do sistema restam dez minutos, mas foi decidido que o limite para execução do sistema seria de nove minutos. A definição do cron é explicada na secção seguinte, 4.4.8.

4.4.8 Definição do Cron no Linux

Cron é o nome da aplicação que permite os utilizadores do UNIX executar comandos ou scripts automaticamente num horário ou data específica. Neste projeto o mesmo foi utilizado para que executasse este sistema em horários específicos que eram de 15 em 15 minutos, concretamente: hh:05, hh:20, hh:35 e hh:50, onde hh são as diversas horas de um dia. A periodicidade deste sistema deve-se principalmente ao Netflow server, visto que este gera novos dados num período de 15 minutos, começando as 00:00 e acabando as 23:45. O cron é definido com 5 minutos de tolerância em relação ao Netflow, devido ao facto de muitas vezes haver atrasos na geração de dados do Netflow.

A estrutura de definição de cron é a seguinte:

```
minute hour dom month dow user cmd
```

Figura 33 - Estrutura de definição do Cron

minute, corresponde a que minutos da hora que o sistema irá ser executado, e é definido num intervalo de 0 a 59;

hour, corresponde a que horas do dia o comando irá ser executado, e é definido num intervalo de 0 e 23;

dom, corresponde ao dia do mês que o comando irá ser executado;

month, corresponde o mês que o comando irá ser executado;

dow, é o dia da semana que o comando vai ser executado, e é definido de 0 a 7.

user, é o utilizador que executa o comando;

cmd, é o comando que irá ser executado.

Na figura 34 encontra-se a definição do cron do sistema, onde executa o ficheiro principal.

```
# m h  dom mon dow   command
5 * * * * php /var/vhost/ipmapping/application/scripts/mapping.php > /var/vhost/ipmapping/cronlogs/cron1.txt
20 * * * * php /var/vhost/ipmapping/application/scripts/mapping.php > /var/vhost/ipmapping/cronlogs/cron2.txt
35 * * * * php /var/vhost/ipmapping/application/scripts/mapping.php > /var/vhost/ipmapping/cronlogs/cron3.txt
50 * * * * php /var/vhost/ipmapping/application/scripts/mapping.php > /var/vhost/ipmapping/cronlogs/cron4.txt
```

Figura 34 – Definição do Cron

4.4.9 Serviços Web disponibilizados

O subsistema IP Mapping disponibiliza um único serviço web, com o objetivo de realizar uma consulta à base de dados, de maneira a saber a que cliente pertence um determinado IP numa determinada data/hora. De seguida veremos a sua implementação.

De futuro ainda podem ser implementadas novas operações, onde podemos fazer pesquisas à base de dados e obter informações relevantes, como por exemplo, saber que MAC Address do equipamento de um determinado cliente. Para isso basta acrescentarmos novas operações ao serviço web disponibilizado.

A função que realiza a operação do serviço web é denominada de checkclient, que recebendo um IP e uma data, retorna o número do cliente que tinha esse IP atribuído naquela data. A implementação desta função é uma simples pesquisa à base de dados com os parâmetros definidos. Na figura 35 está representada uma pergunta a este serviço e na figura 36 está a respetiva resposta.

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:soap="http://ipmapping.nosmadeira.pt/history/index/soap">
   <soapenv:Header/>
   <soapenv:Body>
      <soap:checkclient soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <ip xsi:type="xsd:string">95.172.184.138</ip>
         <date xsi:type="xsd:string">2015-01-15 09:40:00</date>
      </soap:checkclient>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 35 - Pedido SOAP ao serviço web

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:ns1="http://ipmapping.nosmadeira.pt/history/index/soap"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:checkclientResponse>
         <return xsi:type="xsd:string">1611516</return>
      </ns1:checkclientResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 36 - Resposta SOAP do serviço web

Pretende-se saber qual o número de cliente atribuído ao IP “95.72.184.138” na data e hora “2015-01-15 09:40:00”. A resposta do servidor foi que esse IP pertencia ao cliente número “1611516”.

4.5 Conclusão

Neste capítulo foram abordados os assuntos relacionados com a implementação do subsistema *IP Mapping*. A primeira secção refere-se às ferramentas utilizadas para a implementação deste subsistema, onde foi utilizada a Zend Framework 1.12 para a implementação do código em si. Para controlo de versões usamos o GIT e para ferramentas de rede do Linux usamos o LAMP.

A segunda secção refere-se à base de dados do subsistema *IP Mapping*, onde foi analisada a estrutura das quatro tabelas, assim como as partições existentes e os índices criados.

A terceira secção refere-se à implementação dos quatro módulos e do ficheiro principal do sistema, onde é definida a sequência de execução do sistema. Foi vista a implementação da recuperação de erros do sistema e por fim, foi descrita a definição do cron, que permite a execução automática do sistema.

Capitulo 5 – Testes e Resultados

Neste capítulo vão ser abordados os problemas encontrados ao longo da implementação do sistema, os testes efetuados e as soluções encontradas. Alguns desses problemas foram resolvidos em pouco tempo sem alterar muito o sistema, enquanto para resolver outros foi preciso reformular o sistema. Uma preocupação na implementação deste sistema foi a performance, de maneira a garantir que o ciclo de processamento não ultrapassava os 15 minutos. Um grande desafio foi a recuperação automática de falhas, onde cabe ao desenvolvedor pensar em todas as falhas possíveis o que nem sempre é fácil.

Os testes foram feitos ao longo do projeto, onde testávamos o sistema à medida que íamos aumentando as funcionalidades do sistema. Desde o momento em que o sistema estava em produção, as novas funcionalidades eram testadas muito bem numa máquina de *staging* antes de passar para a máquina de *produção*.

Numa fase inicial o sistema entrou em produção, embora o sistema anterior se mantivesse sempre em funcionamento, para a NOS Madeira poder analisar os resultados do novo sistema. A empresa pode assim verificar se os cálculos não estavam muito longe dos do sistema anterior, visto que se trata de um sistema de grande responsabilidade porque abrange muitos utilizadores.

5.1 Problemas encontrados

Ao longo do projeto foram encontrados alguns problemas na implementação dos diversos módulos. Alguns desses problemas foram prontamente resolvidos na sua ocorrência e outros tiveram de ser analisados para perceber a sua causa, requerendo assim algum tempo para a resolução dos mesmos. Ao longo desta secção irão ser descritos os problemas encontrados na implementação dos quatro módulos.

5.1.1 Módulo de recolha de dados de tráfego

Neste módulo surgiu um problema no acesso ao *Netflow server* a partir de FTP, devido às permissões. Depois de verificar as permissões por parte da NOS Madeira foi verificado que o servidor não estava a permitir a conexão. A solução foi permitir a conexão a máquina onde se encontrava alojado o sistema.

Um dos grandes desafios deste módulo foi a interpretação do ficheiro, onde o caminho seguido apresenta algumas desvantagens, principalmente se a estrutura do documento transferido do *Netflow server* mudar. A interpretação do ficheiro foi feita com base na estrutura do mesmo, que foi usada uma expressão regular onde reconhece os dados.

5.1.2 Módulo de recolha de dados dos clientes

Neste módulo o grande desafio foi a recolha dos dados em si, que foi feita a partir de um serviço web fornecido pelos sistemas de informação da NOS Madeira. Um dos problemas encontrados neste módulo era ausência de dados, onde foi revisto se o problema era do módulo implementado ou era dos sistemas de informação. Depois de realizar testes foi verificado que em alguns ciclos o serviço web dos sistemas de informação não retornava qualquer resposta, o que fazia com que esse ciclo ficasse sem dados de tráfego. O problema foi resolvido pelos responsáveis da NOS Madeira.

5.1.3 Módulo de mapeamento dos dados

Os problemas encontrados neste módulo foram a nível de performance na inserção dos dados. O processo de inserção inicialmente demorava em média cinco minutos, porque da forma que a função insertDB estava implementada, a mesma fazia um pedido de inserção por cada tuplo da tabela. A solução foi construir a query de maneira a que juntasse os valores todos e só depois fizesse uma única inserção na base de dados, assim tornando a inserção muito mais rápida, demorando em média dois segundos.

5.1.4 Módulo de envio de dados para o Accounting System

Este módulo foi o mais problemático, principalmente ao enviar os dados para o *Accounting System.*

Um dos problemas encontrados neste módulo foi relacionado com a performance, visto que para selecionar os dados da tabela ipmapping era cada vez mais demorada com o passar do tempo, devido á quantidade de dados que esta tinha presente. A solução para este problema foi a criação de uma tabela com estrutura igual a ipmapping, denominada ipmapping_history que guarda o histórico do tráfego efetuado pelos clientes, libertando assim a tabela ipmapping que é apagada logo depois de ser confirmado que o *Accounting System* recebeu os dados do respetivo ciclo de processamento.

Outro problema estava relacionado com o envio dos dados, onde o pedido SOAP expirava devido ao tempo que demorava o envio dos dados. Uma das soluções tentadas, foi aumentar os tempos limite de execução dos pedidos, mas sem sucesso. Depois de rever vários fatores, o problema foi encontrado no *Accounting System*, onde eram usados determinados índices na tabela responsável por guardar os dados recebidos do *Ipmapping System*, que tornavam o processo de inserção dos dados muito mais lento gerando timeouts. A solução encontrada foi a remoção dos índices desnecessários, tornando o processo de inserção muito mais rápido e assim também o envio dos dados tornou-se num processo rápido e sem problemas.

Foi detetado um problema no momento em que o MySQL se reiniciou, que foi com o atributo id da tabela ipmapping da base de dados, que era definido como auto_increment. Como esta tabela era apagada sempre que enviava dados para o

Accounting System, quando o MySQL reiniciou também o id foi reiniciado, enviando ids repetidos para o *Accounting System*. A solução encontrada foi criar um novo atributo na tabela ipmapping, denominado *sent*, que funciona como uma flag que indica se um determinado tuplo da tabela já foi enviado ou não. Quando o *Ipmapping System* recebe a confirmação que os dados foram recebidos pelo *Accounting System*, este apaga todos os dados da tabela ipmapping com exceção ao último tuplo enviado, e muda o atributo *sent* para um que indica que aquele tuplo já tinha sido enviado e não seleciona o mesmo no próximo ciclo. Assim, mesmo que o MySQL seja reiniciado por alguma razão, a tabela terá sempre um tuplo com o ultimo id recebido pelo *Accounting System*. Para saber o último id enviado foi disponibilizado um serviço web pelo *Accounting System*, que retorna o ultimo id recebido; assim os dados apagados na tabela ipmapping correspondem somente aos dados recebidos por parte do *Accounting System*.

Foi notado que sempre que o IP Mapping System enviava dados para o Accounting System, a máquina ficava lenta, e com muitos processos a executar. Isto devia-se principalmente ao facto de serem feitas múltiplas inserções na base de dados, e todas dos mesmos dados. A solução encontrada para resolver este problema foi a utilização de um trinco, que permitia ou bloqueava o acesso de um processo, neste caso uma nova inserção.

No decorrer do ciclo de execução do *IP Mapping System*, antes de enviar os dados é verificada a existência de processos a enviar os dados. Se existisse algum processo a enviar os dados nesse momento o ciclo era dado como terminado. Se não existisse nenhum processo a enviar os dados, este iniciava um novo processo.

Para verificar a existência de processos a correr foi usado um "lock file", onde basicamente antes de entrar numa determinada área do código, é verificado o conteúdo do ficheiro, se está bloqueado ou desbloqueado, permitindo ou não a execução desse determinado código. Na figura 37, podemos ver onde é verificado este ficheiro.

```
$lock = fopen('/var/vhost/ipmapping/application/scripts/lock.txt', 'w');
if (flock($lock, LOCK_EX | LOCK_NB)) {

    try {
        //Send Data to Accounting
        $send->sendData();
    } catch (Exception $e4) {
        file_put_contents(APPLICATION_PATH . '/../logs/logIpmapping.txt',
          "\n\n" . date('Y-m-d H:i:s') . " - Senddata Problem\n\n" . $e4, FILE_APPEND);
    }
}
```

Figura 37 - Lock file

Resumindo, o flock coloca um trinco num ficheiro. Esse ficheiro funciona como um recurso que enquanto estiver aberto por um programa ou instância de um programa, nenhum outro poderá aceder.

5.2 Tempos médios de execução

Os tempos de execução foram um ponto muito importante neste projeto, porque tinham de ser rígidos no que diz respeito à performance. O tempo de execução total dos três subsistemas nunca deveria exceder os 15 minutos, que correspondem ao tempo que o Netflow server mantém os dados de um ciclo. Passando esse intervalo de tempo são gerados novos dados. O resultado final para o subsistema descrito neste relatório foi muito bom, com o processo total a demorar em média 98 segundos.

Ao longo desta secção iremos analisar os tempos médios de execução dos módulos implementados e ainda o tempo médio de execução de todos os módulos. A nível de cálculos foi obtida a média, o desvio padrão e o intervalo de confiança.

Os testes encontrados nesta secção foram realizados, recolhendo o timestamp atual no início e no final da execução do módulo correspondente através da função *microtime,* disponível no PHP. Tendo estes dois tempos, basta fazermos um simples cálculo do tempo total de execução, fazendo a subtração do tempo inicial ao tempo final. Na figura 38 está presente o código utilizado para obter os tempos, assim como para realizar a respetiva subtração. Este teste foi repetido durante cinco ciclos consecutivos, obtendo assim os valores presentes nas tabelas desta secção.

```
$time_start1 = microtime(true);

//Module Code Here

$time_end = microtime(true);
$execution_time = ($time_end - $time_start);
```

Figura 38 – Implementação do teste para obter tempos de execução

5.2.1 Módulo de recolha de dados de tráfego

	TEMPOS (SEGUNDOS)
	12.99
	12.82
	12.89
	12.22
	12.06
MÉDIA	**12.60**
S	**0.42**
μ	**12.06 < μ < 13.12**

Tabela 1 - Tempos médios de execução do módulo de recolha de dados dos clientes

Na tabela 1 podemos observar que o tempo médio de execução do módulo de recolha de dados de tráfego é de 12.60 segundos, realizando os cálculos de desvio padrão (S) obtemos o valor de 0.42 segundos e com 95% de confiança podemos afirmar que o módulo de recolha de dados de tráfego é executado entre 12.06 e 13.12 segundos.

5.2.2 Módulo de Recolha de Dados dos Clientes

	TEMPOS (SEGUNDOS)
	61.87
	61.07
	60.29
	58.73
	58.90
MÉDIA	**60.17**
S	**1.36**
μ	**58.48 < μ < 61.86**

Tabela 2 - Tempos médios de execução do módulo de recolha de dados de tráfego

Na tabela 2 podemos observar que o tempo médio de execução do módulo de recolha de dados dos clientes é de 60.17 segundos, realizando os cálculos de desvio padrão (S) obtemos o valor de 1.36 segundos e com 95% de confiança podemos afirmar que o módulo de recolha de dados dos clientes é executado entre 58.48 e 61.86 segundos.

5.2.3 Módulo de Mapeamento

	TEMPOS (SEGUNDOS)
	6.38
	6.43
	5.40
	4.60
	5.64
MÉDIA	**5.69**
S	**0.76**
μ	**4.75 < μ < 6.63**

Tabela 3 - Tempos médios de execução do módulo de mapeamento

Na tabela 3 podemos observar que o tempo médio de execução do módulo de mapeamento é de 5.69 segundos, realizando os cálculos de desvio padrão (S) obtemos o valor de 0.76 segundos e com 95% de confiança podemos afirmar que o módulo de recolha de dados dos clientes é executado entre 4.75 e 6.63 segundos.

5.2.4 Módulo de Envio dos Dados

	TEMPOS (SEGUNDOS)
	18.45
	17.57
	16.59
	27.42
	17.74
MÉDIA	**19.55**
S	**4.45**
μ	**14.03 < μ < 25.07**

Tabela 4 - Tempos médios de execução do módulo de envio dos dados

Na tabela 4 podemos observar que o tempo médio de execução do módulo de envio dos dados é de 19.55 segundos, realizando os cálculos de desvio padrão (S) obtemos o valor de 4.45 segundos e com 95% de confiança podemos afirmar que o módulo de recolha de dados dos clientes é executado entre 14.03 e 25.07 segundos.

5.2.5 Ciclo Completo

	TEMPOS (SEGUNDOS)
	99.69
	97.89
	95.17
	102.97
	94.34
MÉDIA	**98.01**
S	**3.50**
μ	**93.67 < μ < 102.35**

Tabela 5 - Tempos médios de execução de um ciclo completo

Na tabela 5 podemos observar que o tempo médio de execução do ciclo completo é de 98.01 segundos, realizando os cálculos de desvio padrão (S) obtemos o valor de 3.50 segundos e com 95% de confiança podemos afirmar que o ciclo completo é executado entre 93.67 e 102.35 segundos.

5.3 Resultados Obtidos

Nesta secção vai ser feita uma análise aos requisitos cumpridos ao longo deste projeto e também ver o resultado final para o cliente, que neste caso é a NOS Madeira.

A nível dos requisitos funcionais foram cumpridos na sua totalidade, como se descreve de seguida:

- O requisito funcional 1) foi cumprido com a implementação do módulo Netflow conforme descrito na secção 4.4.2.
- O requisito funcional 2) foi cumprido com a implementação do módulo Sistemas de Informação conforme descrito na secção 4.4.3.
- O requisito funcional 3) foi cumprido através da implementação do módulo de envio de dados para o *Accounting System* e o módulo de mapeamento, conforme descrito nas secções 4.4.4 e 4.4.5.
- O requisito funcional 4) foi cumprido na criação de uma base de dados relacional, que armazena todos os dados gerados, conforme descrito na secção 4.3.
- Os requisitos funcionais 5) e 6) foram cumpridos na implementação do serviço web que permite a consulta à base de dados, conforme descrito na secção 4.3.9.

A nível de requisitos não-funcionais também foram cumpridos na sua totalidade:

- O requisito não-funcional 1) foi cumprido; foi utilizada a Zend Framework 1.12 na implementação deste projeto, conforme descrito na secção 4.2.1.
- O requisito não-funcional 2) foi cumprido, visto que a implementação do sistema foi dividida em diversos módulos conforme descrito ao longo da secção 4.4, o que torna a deteção de erros mais fácil e permite mais escalabilidade ao sistema.
- O requisito não-funcional 3) foi cumprido, já que o sistema demora em média dois minutos a realizar o ciclo completo de execução, conforme descrito na secção 5.2.

Cumpridos os requisitos, o subsistema *IP Mapping* funcionou como esperado no início do projeto, e foi integrado juntamente com os dois subsistemas, *Accounting System* e *Policy Server System*. Não houve problemas devido aos subsistemas terem sido implementados já com base na integração dos outros sistemas, pois os mesmos foram implementados tendo sempre em conta os parâmetros que tinham de enviar para os outros subsistemas, neste caso concreto o *Accounting System*. O sistema entrou em produção e é atualmente usado pela NOS Madeira para controlo de largura de banda dos seus clientes.

5.4 Conclusão

Neste capítulo foram analisados os problemas encontrados ao longo da implementação do subsistema IP Mapping e apontamos as soluções encontradas para resolvê-los. Foram verificados os resultados obtidos e foi analisado como foram cumpridos os requisitos funcionais e não-funcionais e foi feita uma breve análise ao resultado final. Os testes foram realizados ao longo de todo o projeto, consoante a implementação de novas funcionalidades.

Os resultados obtidos foram muito favoráveis, com o subsistema IP Mapping a realizar todas as funcionalidades que foram propostas pela NOS Madeira. A nível de performance, os resultados foram bons, tendo em conta que este realiza o ciclo de processamento muito mais rápido do que o limite permitido.

Capitulo 6 - Conclusões

O projeto aqui apresentado foi muito importante para a empresa NOS Madeira, visto que tinham como objetivo substituir o sistema responsável pela regulação de largura de banda dos clientes, devido ao facto de já ser um sistema obsoleto e com tecnologias muito antigas que não permitiam atualizações. Com este projeto, torna-se mais fácil todo o processo porque o mesmo é dividido em três subsistemas, *IP Mapping System*, *Accounting System* e *Policy Server System* permitindo assim uma maior facilidade ao nível da deteção de problemas, mantendo as funcionalidades separadas.

Para o desenvolvedor este projeto também foi muito importante, pois teve a oportunidade de aplicar muitos dos conhecimentos adquiridos ao longo do curso, adquirindo experiência na área, nomeadamente em contexto real de uma empresa, como a NOS Madeira, que disponibiliza serviços 24 horas por dia durante sete dias por semana. Este projeto também permitiu ao desenvolvedor familiarizar-se com novas tecnologias, nomeadamente os serviços web, novas ferramentas de implementação, tais como a Zend Framework e o GIT.

Quanto ao subsistema *IP Mapping*, descrito neste documento, trata-se de um sistema que deve garantir a integridade dos dados, pois é a partir dos dados recolhidos neste subsistema que são realizados os cálculos que definem o QoS dos clientes. Uma má recolha dos dados implicaria que os cálculos eram feitos com os dados errados o que levaria à atribuição de um QoS errado ao respetivo cliente.

No futuro este subsistema pode disponibilizar mais serviços web, capazes de fazer pesquisas à base dados, nomeadamente à tabela ipmapping_history, onde é guardada a informação de todos os ciclos efetuados. Estes serviços web irão permitir o conhecimento de informações relevantes, quer sobre o tráfego efetuado, quer sobre os próprios clientes ou equipamentos. Um dos pontos importantes deste subsistema é a quantidade de dados gerados, que ocupa um grande espaço de armazenamento, exigindo assim uma monitorização constante do espaço de armazenamento disponível. Devido principalmente ao tempo limitado para a implementação do sistema, não foi possível a realização de testes intensivos, de maneira a encontrar erros e corrigi-los. Também não foi possível implementar alarmes SNMP que permite a notificação do utilizador da ocorrência de uma eventual falha em algum ponto da execução do sistema.

Este relatório descreve o primeiro subsistema deste projeto, denominado *IP Mapping System*. Ao longo dos capítulos é descrito todo o processo de desenvolvimento do projeto, passando principalmente por quatro fases: definição de requisitos, desenho e modelação, implementação e por fim os testes e resultados do sistema.

No capítulo um foi feita uma breve introdução acerca de todo o enquadramento do projeto, nomeadamente no que se refere à planificação do desenvolvimento.

No capítulo dois foi feita uma abordagem sobre o sistema utilizado anteriormente, responsável pelo controlo de largura de banda da NOS Madeira, assim como as tecnologias relevantes utilizadas neste projeto, tais como, os sistemas gestores de base de dados, os serviços web e a metodologia de desenvolvimento Scrum.

No capítulo três foi descrito o processo de desenho e modelação do sistema, que é um processo fundamental na criação de qualquer projeto de software, porque mostra a arquitetura geral do sistema a implementar, assim como a definição dos requisitos funcionais e não-funcionais.

No capítulo quatro encontra-se a descrição do processo de implementação do sistema, desde a estrutura de ficheiros do projeto, as ferramentas utilizadas, a base de dados implementada e por fim a implementação do sistema em si.

No capítulo cinco encontram-se descritos todos os testes realizados e os resultados obtidos, assim como os problemas encontrados ao longo da implementação do sistema.

O resultado final deste projeto foi a obtenção de um sistema funcional, que garante a estabilidade e que está pronto para identificar e resolver uma eventual falha. Este foi um ponto importante neste projeto, visto tratar-se de um sistema que lida com o desempenho da rede dos clientes da NOS Madeira, onde uma instabilidade no sistema poderia trazer algumas reclamações por parte dos clientes.

Todas as funcionalidades pretendidas foram implementadas e os resultados finais foram favoráveis quer para a NOS Madeira, quer para o desenvolvedor.

Referências

[1] L. Ferreira, "IP Network Usage Accounting - Parte II," Universidade da Madeira, Funchal, Portugal, 2014.

[2] J. Canha, "IP Network Usage Accounting - Parte III," Universidade da Madeira, Funchal, Portugal, 2014.

[3] T. Chaffin, Mark Knight, Brian Robinson, *Professional SQL Server 2000 DTS*. Wiley Publishing, Inc., 2003.

[4] M. Axmark, David Widenius, "1.3 Overview of the MySQL Database Management System," in *MySQL 5.7 Reference Manual*, Oracle, pp. 4 – 9.

[5] M. Axmark, David Widenius, "Storage Engines," in *MySQL 5.0 Reference Manual*, .

[6] wikipedia, "Web service." [Online]. Available: http://pt.wikipedia.org/wiki/Web_service. [Accessed: 30-Dec-2014].

[7] w3schools.com, "SOAP Introduction." [Online]. Available: http://www.w3schools.com/webservices/ws_soap_intro.asp. [Accessed: 02-Jan-2015].

[8] w3schools.com, "SOAP Envelope Element." [Online]. Available: http://www.w3schools.com/webservices/ws_soap_envelope.asp. [Accessed: 02-Jan-2015].

[9] w3schools.com, "SOAP Header Element." [Online]. Available: http://www.w3schools.com/webservices/ws_soap_header.asp. [Accessed: 02-Jan-2015].

[10] w3schools.com, "SOAP Body Element." [Online]. Available: http://www.w3schools.com/webservices/ws_soap_body.asp. [Accessed: 02-Jan-2015].

[11] w3schools.com, "Introduction to WSDL." [Online]. Available: http://www.w3schools.com/webservices/ws_wsdl_intro.asp. [Accessed: 02-Jan-2015].

[12] w3schools.com, "WSDL and UDDI." [Online]. Available: http://www.w3schools.com/webservices/ws_wsdl_uddi.asp. [Accessed: 02-Jan-2015].

[13] "Scrum (software development)." [Online]. Available: http://en.wikipedia.org/wiki/Scrum_(software_development). [Accessed: 09-Jan-2015].

[14] M. Rouse, “Cable modem termination system (CMTS),” *Setembro*, 2005. [Online]. Available: http://searchnetworking.techtarget.com/definition/cable-modem-termination-system. [Accessed: 02-Jan-2015].

[15] M. Rouse, “optical line terminal (OLT),” *Abril*, 2014. [Online]. Available: http://searchnetworking.techtarget.com/definition/Optical-line-terminal-OLT. [Accessed: 02-Jan-2015].

[16] M. Rouse, “operational support system (OSS),” *Julho*, 2007. [Online]. Available: http://searchtelecom.techtarget.com/definition/operational-support-system. [Accessed: 02-Jan-2015].

[17] cisco, “Cisco SCE 8000 Series Service Control Engine.” [Online]. Available: http://www.cisco.com/c/en/us/products/service-exchange/sce-8000-series-service-control-engine/index.html. [Accessed: 02-Jan-2015].

[18] “Introduction to Zend Framework,” 2005. [Online]. Available: http://framework.zend.com/manual/1.12/en/introduction.overview.html. [Accessed: 10-Jan-2015].

[19] S. Chacon and B. Straub, “Pro git,” Second Edi., Apress, 2014, p. 27.

[20] E. Severdlov, “How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Debian,” *October, 4*, 2012. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-debian. [Accessed: 24-Mar-2014].

Anexos

Anexo 1 – Fluxo de Dados

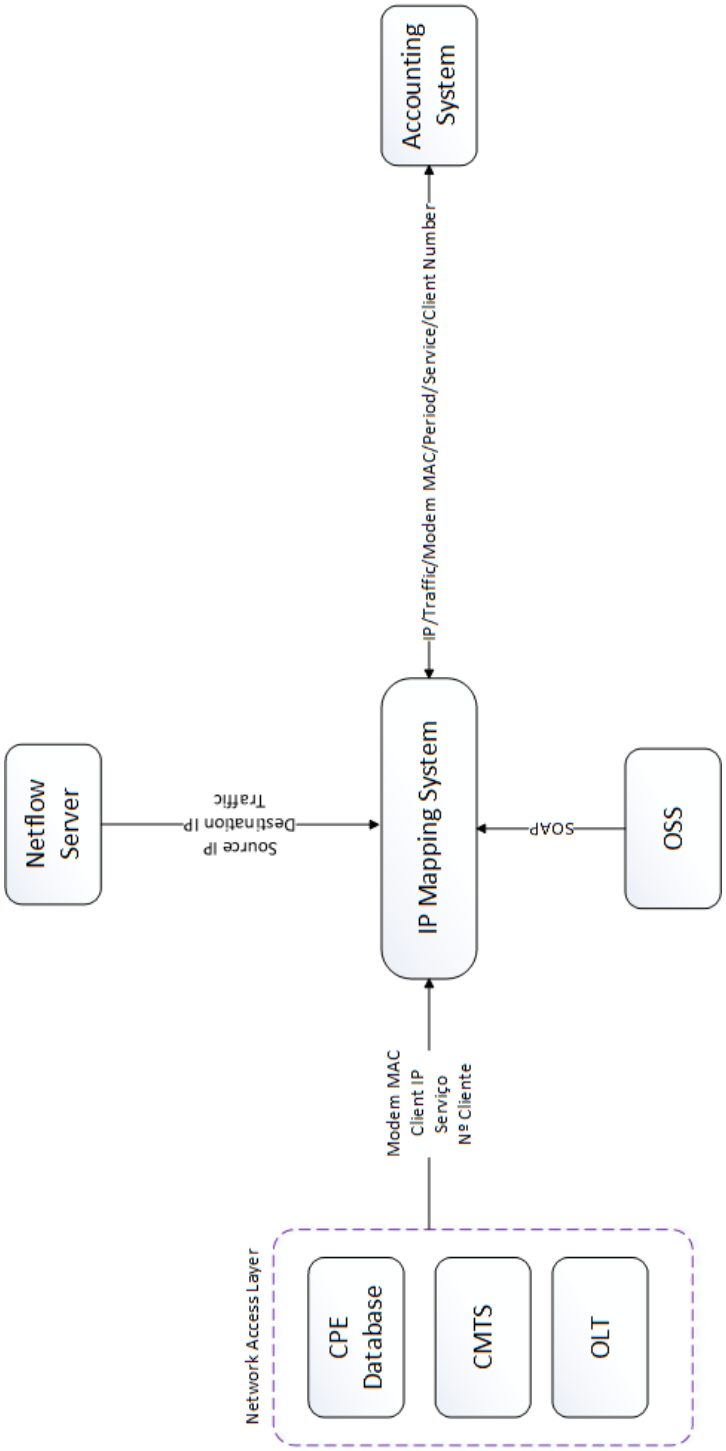

Figura 39 - Diagrama de fluxo de dados o IP Mapping System

Anexo 2 – Diagrama de Sequência

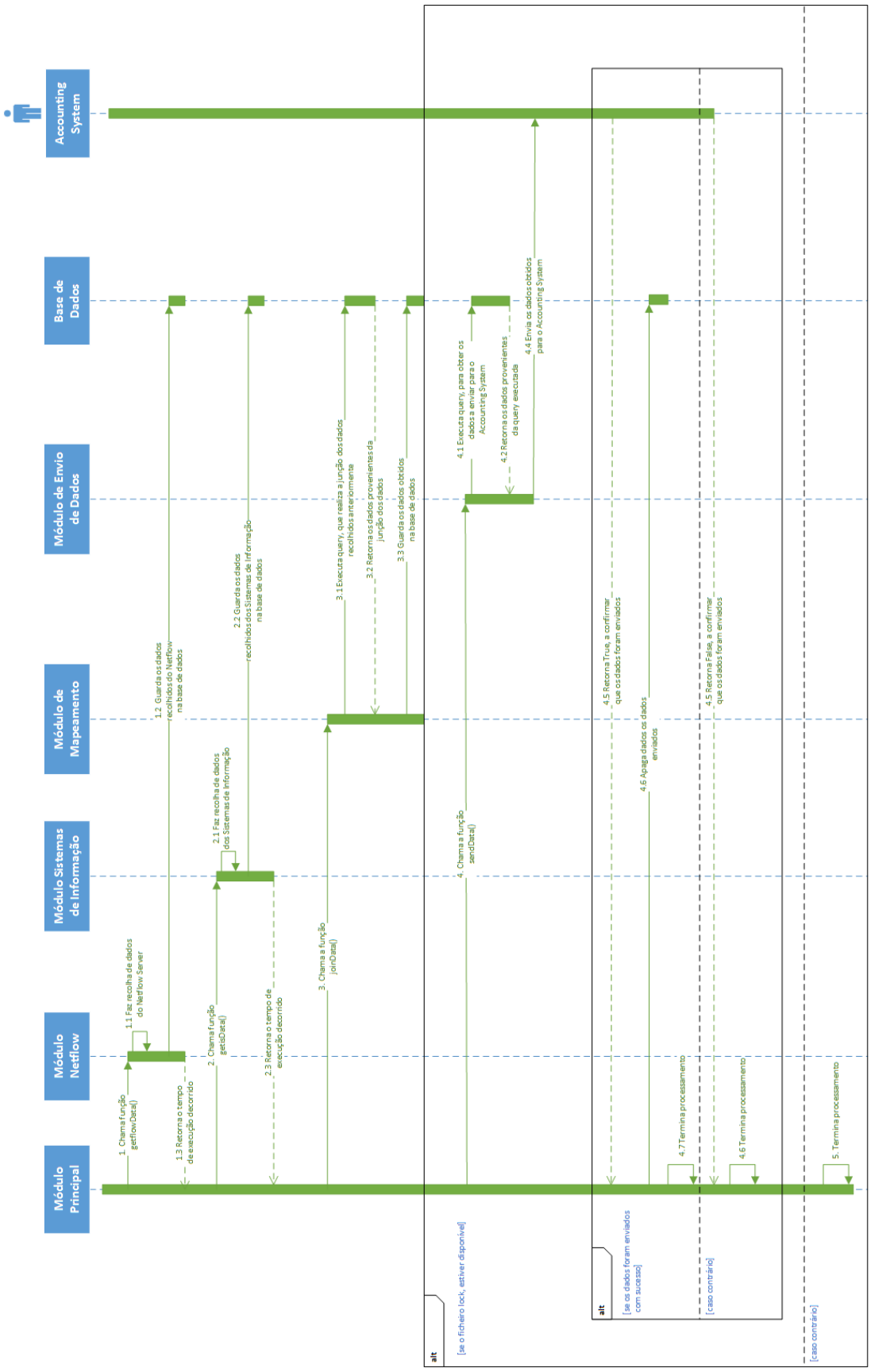

Figura 40 - Diagrama de sequência do IP Mapping System