TR
Technical Report

AMazING

Plataforma de controlo de testes wireless com unidades móveis

André Santos, Jean Brito, Luís Costa, João Laranjo, Rui Coelho

universidade de aveiro

AMazING
Plataforma de controlo de testes wireless com unidades móveis

Projeto em Informática
Licenciatura em Engenharia Informática
Universidade de Aveiro
Junho 2020

André Santos, NMec 84811
andresgomes@ua.pt
Jean Brito, NMec 82783
jean18.jh@ua.pt
João Laranjo, NMec 91153
joao.laranjo@ua.pt
Luís Costa, NMec 85044
lmcosta98@ua.pt
Rui Coelho, NMec 86182
ruicoelho@ua.pt

Resumo

O AMazING (Advanced Mobile wIreless Network playGround) é uma plataforma de testes que tem como objetivo a capacidade de agendar e executar experiências de networking. A rede AMazING é composta por 24 nós fixos com capacidades wireless e um nó móvel (denominado de robô).
Através desta plataforma, de uma forma simples e intuitiva, podemos realizar experiências que podem variar entre o teste de novos protocolos a testes de interferência em redes wireless. Adicionalmente o robô, que possui um nó, está posicionado num carril o que o permite simular os movimentos de um utilizador à medida que se movimenta em relação aos 24 nós fixos.
Com principal objetivo deste projeto foi estabelecido que este seria o desenvolvimento de uma interface que facilite a interação com os nós acima mencionados e que permita, também, aos utilizadores fazer o agendamento e gestão das suas experiências na plataforma, bem como analisar e visualizar os resultados obtidos em cada uma das suas experiências.
É importante realçar que o trabalho a ser realizado com o robô ficou, após diálogo com os orientadores, para segundo plano. Isto devido a uma reestruturação eletrónica no âmbito de trabalhos a ser desenvolvidos em paralelo. Tais motivos tornaram o robô indisponível na realização deste projeto. Neste documento é apresentado o estudo efetuado relativamente a esta plataforma de testes bem como o seu processo de implementação. De seguida, é descrita a implementação da plataforma, a sua arquitetura e o seu modelo de dados.

Palavras-Chave

Experiência Testes realizados sobre os nós existentes na rede

APU: Advanced Processing Unit Hardware similar a um router utilizado para realizar as experiências de rede

SBC: Single Board Computer Computador construído numa única placa de circuitos. Possui elementos como um microprocessador(es), memória e I/O.

Nó APU configurada

Agradecimentos

Gostaríamos de agradecer aos supervisores de projeto, o Professor Flávio Meneses e ao Professor Daniel Corujo pela disponibilidade que demonstraram desde o primeiro momento para nos ajudar a chegar aos objetivos estabelecidos e a clarificar dúvidas que foram surgindo ao longo das várias fases do projeto.

Índice

Índice de Imagens

Índice de Tabelas

CAPÍTULO 1

Introdução

O desenvolvimento de plataformas que facilitem as operações de utilizadores de um dado sistema não é um conceito novo. A plataforma desenvolvida no decorrer deste projeto tem como público alvo utilizadores com experiência na área de redes e telecomunicações. Embora o contexto em que este projeto se insere seja bastante específico, é possível encontrar plataformas semelhantes à desenvolvida neste projeto. No entanto, a grande maioria das soluções encontradas são específicas à infraestrutura sobre a qual assentam e as restantes são demasiado generalistas, não considerando todos os detalhes necessários à realização de experiências sobre redes.
Deste modo, tornou-se necessário o desenvolvimento de uma plataforma capaz de responder às necessidades destes utilizadores, surgindo assim o AMazING.

1.1 Contexto

Este projeto foi desenvolvido no âmbito da unidade curricular de Projeto em Informática, inserido na Licenciatura em Engenharia Informática (LEI) da Universidade de Aveiro (UA). O foco do projeto é criação de uma interface que permita fazer o agendamento e gestão de experiências na plataforma de testes, sendo possível consultar os resultados após o término das mesmas. O presente trabalho foi desenvolvido durante o segundo semestre do ano letivo de 2019/2020.

1.2 Motivação

A plataforma AMazING constitui uma mais valia para qualquer pessoa que pretenda realizar testes sobre redes móveis dado que esta permite simular situações reais.
A motivação para este projeto é o desenvolvimento de uma plataforma que ajude essas mesmas pessoas a agendar e gerir as suas experiências e a obter os dados resultantes das mesmas, desse modo, facilitando todo o processo através de um GUI que lhes fornece tudo o que necessitam de forma fácil e intuitiva.

1.3 Objetivos

Este projeto teve como principal objetivo desenvolver uma plataforma intuitiva e de fácil utilização que permita aos seus utilizadores realizar as suas experiências com um nível menor de esforço no que toca à preparação e visualização destas.
Após uma reunião com os orientadores do projeto procedeu-se à definição de uma lista de objetivos na qual o desenvolvimento deste projeto se focou. Os objetivos definidos podem ser enunciados da seguinte forma:

- Criação de uma plataforma web que agrega todo o trabalho já existente;
- Tornar a plataforma intuitiva e fácil de usar como foi mencionado anteriormente;
- Tornar a plataforma de testes reconfigurável de modo a que esta se adapte às necessidades de cada utilizador.
- Desenvolvimento de uma ferramenta que permite a visualização de informação de cada nó existente na infraestrutura bem como recolher estatísticas e informações importantes numa dada experiência;
- A plataforma deve ser capaz de gerir ambientes de diferentes tipos de testes.

1.4 Impactos do Covid-19

Durante o desenvolvimento deste projeto houve a clara interferência causada pela situação do Covid-19 e como tal nós reunimos e delineamos não só os impactos mas também as soluções de modo a melhor nos prevenirmos e de modo a que o desenvolvimento não fosse afetado. Deste modo consideramos que os maiores impactos seriam:

- A não existência de reuniões presenciais o que dificulta a comunicação e interação do grupo;
- A impossibilidade de reuniões presenciais com os nosso orientadores, tornando todo o processo de transmissão de informação e esclarecimento de dúvidas mais complicado;
- A impossibilidade de aceder fisicamente ao IT o que impede as experiências com os equipamento lá presentes, nomeadamente o Switch onde estão conectados os nós, assim como acesso à rede privada de testes;
- Devido à impossibilidade de acesso à rede privada de testes foram disponibilizadas 5 APUs para nos possibilitar o avanço do trabalho sem a necessidade de deslocamento ao IT. Seria um para cada membro do grupo no entanto com a divisão de tarefas apenas dois membros do grupo ficaram com APUs - 3 e 2 para os 2 membros encarregues desta parte do projeto;

- Impacto da moral e motivação da equipa uma vez que não era possível reunirmo-nos presencialmente e ver os resultados da solução tão claramente.

Dado os impactos acima referidos, e após alguma deliberação, foram implementadas as seguintes soluções:

- Reuniões semanais presenciais foram substituídas por videoconferências de modo a manter o contacto com os orientadores mais "real";
- O contacto com os orientadores foi feito através de videoconferências e também através canal de slack mais ativo. Em alternativa, caso estes meios não estivessem a funcionar/disponíveis, o contacto foi mantido com a troca de emails;
- Os equipamentos foram enviados por correio e apenas quando absolutamente necessário foi feito um deslocamento com as devidas precauções para entrega e troca de algum material;
- Utilização de um switch comercial, de modo a substituir o switch Aruba apenas disponível no IT.

1.4.1 Estrutura do Documento

O restante documento possui uma estrutura dividida por cinco capítulos. O **capítulo 2** apresenta uma índole mais analítica, no qual são apresentados os trabalhos existentes nesta área identificando, assim, bases que poderão ser utilizadas para o presente projeto. São, ainda, discutidas as tecnologias que foram utilizadas bem como a explicação da sua utilização no âmbito do projeto. No **capítulo 3** é realizada uma análise e discussão da arquitetura do sistema bem como dos requisitos fundamentais para o seu bom funcionamento. No **capítulo 4** é realizada uma análise detalhada da implementação da plataforma abordando os vários pontos, isto é, explicação do frontend, da API de ligação entre o Frontend e a base de dados bem como a comunicação com os APUs e, por fim, a discussão da implementação do módulo de comunicação dos APUs (API Rest). Finalmente, no **capítulo 5** são enunciadas as conclusões tiradas do projeto bem como toda a aprendizagem associada ao mesmo, apontando ainda possível trabalho futuro que o grupo pensa fazer sentido no contexto do projeto.

CAPÍTULO 2

State of the Art

Devido ao facto de a temática deste projeto se inserir num nicho de mercado, é natural que não existam muitas soluções semelhantes. No entanto, no decorrer do processo de investigação associado ao projeto foi possível encontrar duas plataformas dignas de menção.

2.0.1 Projetos Relacionados

Tal como já foi referido, foram encontrados 2 projetos dignos de menção, sendo que um deles é um projeto previamente desenvolvido na UA. O segundo projeto encontrado é um trabalho desenvolvido e publicado numa parceria entre estudantes e investigadores do NICTA (National Information and Communications Technology Australia) e da Universidade Rutgers, na Austrália.

2.0.1.1 NICTA

Conforme o referido acima, este projeto foi desenvolvido por uma parceria entre estudantes e investigadores australianos. O paper publicado por esta parceria tem como titulo "OMF: A Control and Management Framework for Networking Testbeds". A partir da leitura do paper foi possível entender que este projeto procura resolver o mesmo problema que o AMazING, tendo adotado uma arquitetura bastante semelhante à da solução presente neste relatório.

2.0.1.2 AMazING 2019

O projeto AMazING foi iniciado em 2010, tendo sofrido várias atualizações ao longo dos anos, e tem como objetivo base uma plataforma de testes para redes sem fios em ambiente real que permita não só a gestão e monitorização de experiências em ambientes de redes sem fios, mas também a reprodutibilidade dos mesmos.
Este projeto foi abordado mais recentemente por um grupo de estudantes da universidade, no contexto da unidade curricular de Projeto em Informática que decorreu no ano letivo de 2018/19. O trabalho anteriormente realizado por este grupo encontra-se disponível no github. Embora a infraestrutura base, entenda-se o AMazING, seja a mesmo que a apresentada neste relatório, os focos dos dois projetos divergem ligeiramente.
A nível de trabalho realizado anteriormente, a edição do AMazING do ano letivo 2018/19

focou-se maioritariamente na construção da rede de nós em si (incluindo o "robô" mencionado previamente), deixando a interface e a plataforma web para segundo plano, o que levou a que essa mesma plataforma tivesse uma interface com o utilizador reduzida em funcionalidades. Em contraste, a edição de 2019/20 do AMazING tem como um dos objetivos a melhoria da plataforma web, de modo a torná-la mais acessível, intuitiva e com mais funcionalidades do que o projeto antecedente.

2.1 Tecnologias Utilizadas

Nesta secção apresentamos de forma sucinta as várias tecnologias que foram utilizadas no desenvolvimento do sistema.

2.1.1 Máquina Virtual

Na ciência da computação, uma máquina virtual consiste num software de ambiente computacional, capaz de executar programas como um computador real. Este processo é comumente referido como virtualização.
Foi disponibilizada uma Máquina Virtual no IT, acessível apenas na rede interna do departamento. Isto implica que o acesso à VM só possível dentro da rede do departamento ou através de uma VPN. Esta máquina virtual possibilitou a instalação de todas as instâncias utilizadas para o projeto em um ambiente único e disponível a todos os integrantes.

2.1.2 PostgreSQL

PostgreSQL é um sistema open-source de gestão de bases de dados relacionais (Relational Database Management System, RDBMS) coordenado pelo PostgreSQL Global Development Group. A sua criação teve como objetivo manter uma ferramenta open-source com a mesma fidelidade que o MySQL.
A estrutura funciona tendo como base tabelas que, por sua vez, têm atributos e recorre à Structured Query Language (SQL) de modo a permitir a manipulação dos dados armazenados nas tabelas.
No contexto deste projeto, o PostegreSQL foi utilizado para a manutenção de todas as informações associadas à plataforma, isto é, informações sobre as experiências, utilizadores, endereçamento e informações sobre os nós.

2.1.3 SQLite

SQLite é um RDBMS embutido dentro de uma biblioteca escrita na linguagem C. Isto significa que não existe a necessidade de instalar software adicional de modo a interagir

com a base de dados criada, visto que o SQLite cria o ficheiro de base de dados diretamente no disco.
A estrutura funciona tendo por base tabelas que fazem uso da linguagem SQL, tal como no RDBMS PostgreSQL.
Esta ferramenta foi utilizada em dois níveis distintos, o backend (na sua API Flask) e no Frontend.
No caso do backend a utilização desta ferramenta teve em vista o teste das features desenvolvidas uma vez que, devido ao covid, o desenvolvimento do projeto foi afetado. Tomando isto em consideração, foi necessário arranjar formas de armazenar informações de testes sem que fosse necessário configurar ambientes de desenvolvimento tão pesados e que pudessem ser facilmente replicados no computador do membro do grupo que possui os nós. Uma vez que o SQLite é um ficheiro de texto, bastava ao membro ter esse ficheiro no seu ambiente de desenvolvimento sem que fossem necessárias configurações extra evitando, assim, de cada vez que fosse necessário efetuar alterações nas tabelas de base de dados que fosse necessário este processo de implementação associado.
No caso do Frontend esta ferramenta foi essencialmente usada para a gestão de logs por parte do administrador do sistema.

2.1.4 Python

Python é uma linguagem de programação de alto nível criada por Guido van Rossum em 1991. De entre as suas características, destaca-se o facto de ser interpretada ao invés de ser compilada.
A sua simplicidade, bem como a facilidade de aprender a sintaxe, leva a que os programadores possam escrever código lógico e claro tanto para projetos de pequena como grande dimensão. Para além disso, o facto de não existir a etapa de compilação leva a que o seu ciclo edição-teste-debug seja muito mais rápido.
Esta linguagem foi utilizada em 3 camadas do sistema, na camada de apresentação, a framework Django 2.2.6, e em dois níveis da camada de lógica utilizando a framework Flask 2.2.5.

2.1.5 Flask

Flask é uma framework web escrita em Python e baseada nas bibliotecas WSGI Werkzeug e Jinja2. O Flask possui a flexibilidade do Python e fornece um modelo simples para desenvolvimento web.
Esta framework foi utilizada em duas camadas lógicas do projeto, nas quais foram criadas:

- Uma REST API nos servidores disponibilizados. Permite a gestão do sistema, acesso à Base de dados, além de funcionar como um proxy de comunicação com as APUs;
- Uma REST API de controlo em cada um dos nós (APUs) utilizados nos projeto.

2.1.5.1 Flask Mail

O Flask Mail é uma ferramenta que fornece uma interface de configuração SMTP para a aplicação. Esta ferramenta permite o envio de emails através de views e scripts desta interface.
Esta ferramenta foi amplamente utilizada para o envio de emails de notificação acerca da realização das experiências. Estes emails são enviados no início e no fim da execução de uma experiência, independentemente dos resultados da mesma.

2.1.5.2 SQLAlchemy

SQLAlchemy é uma framework open source que fornece um conjunto de ferramentas que permitem mapear uma base de dados relacional em objetos na linguagem de programação Python.
Esta framework evita a escrita de queries SQL para aceder à base de dados, usando-se no seu lugar a interface do SQLAlchemy. Isto possibilita a migração entre diferentes bases de dados relacionais sem alteração do código fonte.
Esta framework revelou-se bastante útil no ambiente de testes, uma vez que facilita a utilização de uma base de dados secundária para a realização dos mesmos.

2.1.5.3 Swagger

O Swagger é uma framework open source apoiada por um grande ecossistema de ferramentas que ajuda os desenvolvedores a projetar, criar, documentar e consumir serviços da Web RESTful. O conjunto de ferramentas do Swagger inclui suporte para documentação automatizada, geração de código e geração de casos de teste.
A documentação da API foi construída com recurso ao Swagger, onde foram especificados todos os caminhos da API, exemplos de entrada e retorno de dados, para além de conter os possíveis status code de retorno.
Visto que a framework Swagger permite a realização de pedidos HTTP, é possível realizar pedidos ao servidor Flask através da mesma, inclusivé a caminhos que necessitem de JWT.

2.1.5.4 PyTest

A framework Pytest facilita a criação de pequenos testes em código Python, no entanto, também é capaz de suportar testes funcionais complexos para aplicações e bibliotecas.
O Pytest foi utilizado na API afim de se realizarem testes de integração sobre todos os caminhos existentes. Esta framework foi escolhida pois facilita a escrita de testes do tipo test client, de modo a testar os endpoints.

2.1.5.5 APScheduler

O Advanced Python Scheduler (APScheduler) é um programa que permite agendar pequenas tarefas (funções ou chamadas em código Python), de modo a serem executadas em ocasiões pré-determinadas.
Esta ferramenta foi utilizada para o arranque automático de experiências nos horários agendados pelos utilizadores. Esta ferramenta permite também auto-agendamento, isto é, após o término uma experiência, a seguinte é agendada automaticamente, independentemente do resultado da experiência.

2.1.6 Django

Django é uma framework para desenvolvimento rápido para web, escrito em Python, que utiliza o padrão model-template-view (MTV).
O WebSite do projeto foi construído com recurso a esta framework. A plataforma web consome os dados da API Flask e acede às APUs através de SSH (com recurso ao WebSSH). Para além disso possui uma base de dados própria em SQLite para realizar a gestão de acesso dos utilizadores.

2.1.6.1 Selenium

Selenium é uma framework utilizada para testes em aplicações WEB. O Selenium fornece uma ferramenta de reprodução que permite criar testes funcionais sem a necessidade de aprender uma linguagem de script de teste (Selenium IDE). Ele também fornece uma linguagem específica de domínio de teste (Selenese) para escrever testes em várias linguagens de programação, incluindo Groovy, Java, Perl, PHP, Python, Ruby e Scala. Os testes podem ser executados nos browsers mais comumente utilizados. O Selenium pode ser executado em Windows, Linux e macOS.
Selenium foi utilizado para realizar testes automatizados sobre a aplicação Web escrita em Django, garantindo que todas as funcionalidades das diversas partes da aplicação se encontram em conformidade com o esperado.

2.1.6.2 SSH

Secure Shell (SSH) é um protocolo criptográfico que permite estabelecer uma conexão segura sobre uma rede insegura. Este protocolo é capaz de criar canais seguros através de uma arquitetura cliente-servidor, em que o cliente SSH estabelece uma ligação ao servidor SSH pretendido. A sua utilização mais comum é na autenticação remota numa máquina, de modo a ser possível executar comandos no terminal.
No contexto deste projeto, o SSH foi utilizado de modo a permitir que um utilizador da plataforma consiga estabelecer uma ligação a uma APU através do seu browser. Para tal, foi utilizada a biblioteca de Python, WebSSH.
Esta biblioteca permite criar um SSH client ao qual o browser do utilizador se liga

utilizando o protocolo HTTP e websockets e por sua vez esse mesmo cliente ligar-se-á através de SSH ao servidor que for pretendido.

2.1.7 OpenStack

O OpenStack é uma plataforma de cloud computing, open source, implantada principalmente como Infraestrutura como Serviço (IaaS). A plataforma de software consiste num conjunto componentes inter-relacionados capazes de controlar pools de hardware de processamento, armazenamento e rede num único data center. A sua gestão é feita através de uma dashboard Web, por ferramentas de linha de comando ou por serviços Web RESTful.
Esta ferramenta foi utilizada para a criação de uma VM capaz de fazer deployment de todo o projeto permitindo, assim, a exposição da base de dados, API e Frontend. É de salientar que o deployment não foi completamente efetuado nesta máquina uma vez que o mesmo exige a utilização de nós e, neste momento, não existem nós disponíveis no edifício do IT. Deste modo, apenas a API Flask, o Frontend e o WebSSH se encontram deployed na VM.

2.1.8 Preboot Execution Environment

O Preboot Execution Environment (PXE ou pixie) é um ambiente que permite inicializar computadores através da Interface da Placa de Rede, independentemente da existência de dispositivos de armazenamento (como Disco Rígidos) ou de um Sistema Operativo. Deste modo, o Sistema Operativo do equipamento é carregado pela interface de rede sempre que o mesmo é ligado, evitando assim o uso de unidades de armazenamento local e a atualização dos equipamentos de forma individual.
No contexto do projeto, após a realização de uma experiência, o ambiente de todas as APUs deveria ser reciclado, evitando que experiências realizadas anteriormente afetem a realização e/ou resultados de experiências futuras.
Apesar dos esforços, não foi possível carregar os ambientes para as APUs através do PXE devido às dificuldades associadas à montagem de um servidor de PXE, ao acesso à BIOS de forma a alterar a ordem do boot e à dificuldade adicionar novas imagens com as configurações necessárias ao servidor. Estes impedimentos tornaram impossível inicializar as APUs com o sistema operativo alocado no servidor PXE.

2.1.9 Docker

Docker é um software container da empresa Docker, Inc, que fornece uma camada de abstração e automação para a virtualização de sistemas operacionais em Windows, Linux e macOS. Isto é alcançado através do isolamento de recursos do núcleo do Linux, tais como cgroups e namespaces, e da utilização de um sistema de arquivos com recursos de união, como o OverlayFS. Deste modo, o Docker é capaz de criar containers inde-

pendentes que executam dentro de uma única instância do sistema operativo, evitando assim a sobrecarga de manter máquinas virtuais.

Foram criadas várias instâncias do Docker para os distintos serviços a disponibilizar. Tendo em vista este objetivo foram lançados containers para:

- A instância da base de dados do projeto - PostgreSQL
- Uma instância de Python 3.7 para a API Flask que serve o Frontend
- Uma instância de Python 3.7 para o Frontend
- Uma instância de Python 3.7 para o serviço de SSH over Browser (WebSSH)

2.1.10 APU

Unidade de Processamento Acelerado ou APU do inglês. Cada nó da plataforma AMazING é uma APU, uma unidade versátil e compacta capaz de desempenhar as funções de diferentes hardwares na área de redes e comunicações. O mesmo hardware pode comportar-se, por exemplo, como router, AP (Access Point), Switch, cliente e servidor. Cada APU é constituída por duas placas de rede wireless e três entradas ethernet, sendo duas delas usadas como interface de dados e outra usada como interface de controlo. Todo a configuração associada a cada APU é realizada através desta interface de controlo.
As APUs utilizadas no projeto são do modelo APU2C4, fabricado pela PC Engines. Estas APUs possuem as seguintes especificações:

- CPU - AMD GX-412TC, 1GHz quad Jaguar core;
- DRAM - 4GB DDR3-1333 DRAM;
- Armazenamento - O arranque pode ser feito a partir de um cartão SD, de um disco SSD ou através da porta USB;
- Conectividade - 3 portas Ethernet Gigabit (no site não fala sobre wifi, se souberem alguma coisa quanto a isso avisem para acrescentar aqui);
- Alimentação - Fonte de alimentação de 12V.

2.1.11 Switch Aruba

O Switch Aruba é o switch onde todos os nós da plataforma AMazING se encontram ligados. Este Switch suporta SDN (Software-defined networking), o que permite configurações dinâmicas e eficientes melhorando o desempenho e a monitorização de redes.
Como descrito anteriormente, cada APU possui uma interface de dados, esta é ligada entre ao Switch Aruba. Isto quer dizer que qualquer configuração proveniente do servidor é redirecionada para o switch que por sua vez encaminha para o determinado nó.

É importante mencionar que relacionadas a este switch estão as funcionalidades de PXE e de POE também descritas no documento presente.
Devido aos constrangimentos causados pelo Covid-19, o switch Aruba teve que ser substituído por um de grau comercial.

2.1.12 Github

O controlo de versões do código do projeto foi feito através do serviço disponibilizado pelo GitHub, que é uma plataforma de hosting de código-fonte com controle de versão usando o Git.
Optou-se por ter apenas um repositório central uma vez que, estando o grupo dependente de um hardware específico para o projeto funcionar, não seria possível instalar processos de CI/CD, não se justificando, assim, a utilização de vários repositórios como indicam as boas práticas do mesmo. Deste modo, todo o código escrito encontra-se nesse repositório central separado em diretórios diferentes, sendo cada um destes diretórios um serviço. É de salientar que cada serviço foi desenvolvido no seu branch específico e só após a validação do mesmo é que o código foi colocado na master.

2.1.13 Conclusão

A utilização destas ferramentas fornece um conjunto de características que foram vistas como vantajosas ao desenvolvimento do projeto, tendo como principal vantagem a familiaridade do grupo com as mesmas. É de salientar que a facilidade de desenvolvimento, dando especial ênfase às frameworks executadas sobre o código Python, também se revelou um fator decisivo aquando da escolha das tecnologias.
Todos os serviços utilizados no desenvolvimento do projeto foram hospedados numa máquina virtual localizada no IT. Recorreu-se ao Docker para a criação e execução dos containers contendo os serviços necessários, alcançando assim o isolamento pretendido. Este isolamento facilitou o desenvolvimento dos diversos serviços, bastando estar conectado ao ambiente da rede do departamento, através de, por exemplo, uma VPN, para poder consultar e utilizar o trabalho realizado pelos integrantes do grupo.

CAPÍTULO 3

Requisitos e Arquitetura do Sistema

3.1 Requisitos do Sistema

Esta secção apresenta as especificações gerais dos requisitos do sistema. As seguintes subseções descrevem, em primeiro lugar, o processo de levantamento de requisitos e de seguida o contexto, os atores e os respetivos casos de uso. Por fim, são especificados os requisitos não funcionais do sistema, bem como as dependências do sistema e suposições.

3.1.1 Levantamento de requisitos

O levantamento de requisitos foi realizado na primeira fase de desenvolvimento do projeto, na fase inicial de desenvolvimento. Foi realizada uma reunião com os Professore Flávio Meneses e o Professor Daniel Corujo na qual foram expostos os principais objetivos da plataforma.
Desse modo, foram tomadas as seguintes decisões:

- Uma plataforma web será o foco principal do projeto sendo que a possibilidade de trabalhar com o nó móvel (também denominado de robô) foi movida para trabalho futuro devido ao facto do mesmo estar a ser alvo de reestruturações mecânicas e eletrónicas;
- A utilização da plataforma está restrita para utilizadores autenticados.
  - Numa primeira fase serão criados, por um administrador do sistema, utilizadores próprios da plataforma (com um sistema de login com utilizador e palavra passe);
  - Numa fase posterior perspectiva-se a integração com as credenciais de colaboradores do IT através do LDAP.
- Permitir a reconfiguração da rede de nós através da plataforma assim como a visualização dos dados de cada nó e os resultados das variadas experiências;

3.1.2 Contexto

Redes sem fios e, mais em particular redes móveis, têm sido um assunto bastante discutido nos dias que correm. Trabalhos de investigação e de inovação têm-se deparado com dificuldades na realização de testes ou experiências em ambiente real. Tal pode acontecer

devido ao preço do material, condições ambientais ou difícil reprodução das condições ou ambientes sem fios. Para solucionar este problema surge a plataforma AMazING, uma plataforma de testes wireless com uma interface apelativa e fácil de utilizar.
A plataforma foi desenvolvida de modo a facilitar a configuração dos nós necessários para cada experiência. O utilizador poderá agendar e realizar a sua experiência num ambiente o mais real possível.
É importante referir que, a execução de vários testes em simultâneo seria possível desde que fossem utilizados nós distintos ou fosse garantido o isolamento, através de SDN, no switch. No entanto, devido à complexidade e aos conhecimentos de redes necessários para o desenvolvimento desta feature, foi considerado impossível a implementação da mesma no tempo disponível para a realização do projeto.
Por este motivo, sempre que um utilizador pretenda agendar uma experiência, todos os nós ficam reservados para ele durante um determinado espaço de tempo. No menu principal está disposta a grelha de nós e as várias operações possíveis estão divididas em secções na barra de navegação, sendo que cada secção pode ter subsecções que permitem realizar funcionalidades mais específicas.

3.1.3 Atores

A desenvolvimento da interface gráfica foi levado a cabo tendo em conta que a plataforma será utilizada por pessoas que à partida são experientes na sua área, porém mantendo a interface simples de modo a ser de uso fácil. De seguida, enumeramos uma classificação dos atores que irão utilizar a plataforma:

- **Utilizador - Begginer** – O João é estudante de Mestrado e está a realizar a sua tese no Instituto de Telecomunicações. Encontra-se num projeto de investigação que envolve testes de comunicação (com e sem fios) e desenvolvimento de novos protocolos de redes.
  O João precisa de realizar os seus testes num ambiente o mais real possível evitando simuladores virtuais.
  O João é um utilizador iniciante, (nível 1) e só lhe é permitido utilizar os templates ao utilizar o sistema. e carregar ficheiros de execução para as APUs.

- **Utilizador - Advanced** - O João recebe acesso de utilizador avançado (nível 2) o que lhe permite ter acesso direto ao terminal das APUs durante a realização de sua experiência. Podendo assim realizar a instalação de drivers, implementação de protocolos e outros serviços necessários para a realização da experiência

- **Administrador** - O Manuel é professor e Doutorado em Redes e Telecomunicações. É um dos responsáveis pelo Instituto de Telecomunicações e tem experiência em vários projetos relacionados com o desenvolvimento e implementação de novos protocolos de redes e telecomunicações.
  O Manuel precisa de garantir o funcionamento e o acesso aos recursos disponibilizados no IT.

3.1.4 Casos de Uso

Nas imagens seguintes, apresentamos os modelos de casos de uso(CaU) da solução.

3.1.4.1 Utilizador/Tester

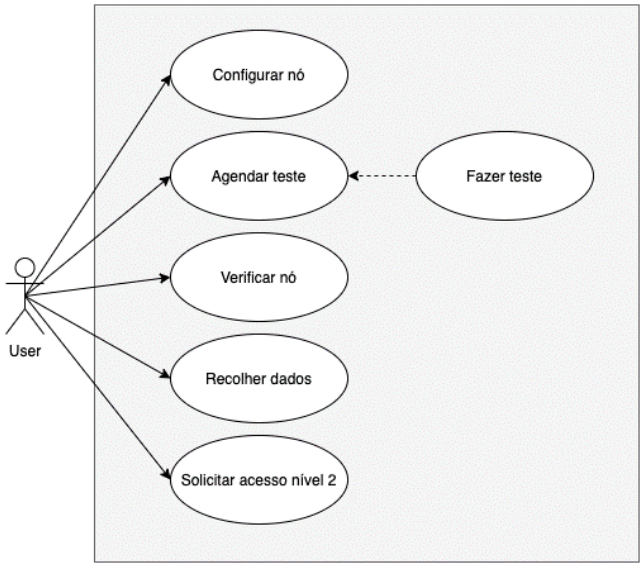

Figura 3.1: Casos de uso do utilizador/Tester.

Os casos de usos consoante a sua prioridade podem sintetizados de acordo com a seguinte tabela:

CaU	**Descrição**	**Prioridade**
Configurar Nó	Permite ao utilizador configurar um dado nó da rede como preferir.	Alta
Agendar/Fazer teste	Permite ao utilizador agendar a sua experiência e consequentemente efetuá-la no seu tempo alocado.	Alta
Verificar nó	Permite ao utilizador ver a informação de um nó (o seu estado, configuração, etc)	Alta
Recolher Dados	Permite ao utilizador recolher os dados provenientes da sua experiência.	Alta
Solicitar acesso de nível 2	Permite que utilizador solicite a um administrador uma elevação dos seus privilégios na plataforma.	Alta

Tabela 3.1: Casos de uso do utilizador/Tester.

3.1.4.2 Administrador

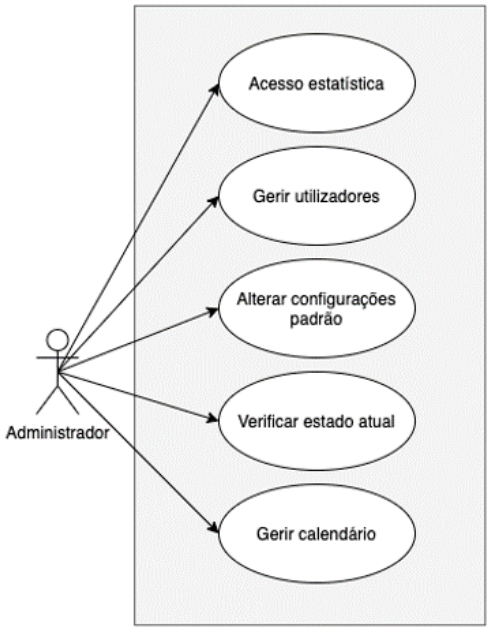

Figura 3.2: Casos de uso do Administrador.

Os casos de usos consoante a sua prioridade podem sintetizados de acordo com a seguinte tabela:

CaU	**Descrição**	**Prioridade**
Acesso a estatísticas	Permite ao administrador aceder às estatísticas da plataforma.	Alta
Gerir Utilizadores	Permite ao administrador gerir os utilizadores da plataforma assim como elevar o seu nível.	Alta
Adicionar/alterar Configurações padrões	Permite ao administrador adicionar/alterar configurações padrão dos nós	Alta
Verificar estado atual	Permite ao administrador verificar o estado atual da rede AMazING e dos seus nós.	Alta
Gerir Calendário	Permite ao administrador ver e gerir o calendário e os testes agendados.	Alta

Tabela 3.2: Casos de uso do Administrador.

3.1.5 Requisitos não funcionais

Abaixo, apresentamos os requisitos não funcionais do sistema.

- **Usabilidade:** dado o principal propósito da aplicação, é necessário que esta seja simples de aprender e utilizar;
- **Autenticação:** A utilização da aplicação deve ser disponível apenas para utilizadores do IT
  - Numa primeira fase, adicionados ao sistema por um administrador;
  - Numa segunda fase, possuindo autenticação pelas credenciais de colaboradores do IT através do LDAP.
- **Controlo:** O administrador do sistema deve ser capaz de configurar e realizar a manutenção do sistema;
- **Dados:** Os utilizadores do sistema deve conseguir aceder aos dados da experiência realizada.

3.1.5.1 Suposições e Dependências

Para o funcionamento completo do sistema, é necessário que sejam configurados os seguintes:

Hardware	**Software**

• APUs
• Switch Aruba
	• Base de Dados Relacional de linguagem SQL
	• Flask
	  - No servidor principal
	  - Em cada uma das APUs
	• Django + SQLite
	• Web SSH server

Tabela 3.3: Componentes necessárias para o funcionamento.

3.2 Arquitetura do sistema

Esta secção apresenta uma vista geral da arquitetura do sistema: os seus modelos de domínio, físico e tecnológico e a relação entre as componentes de cada um.

3.2.1 Modelo Físico

O modelo físico, representado na Figura 4, apresenta uma vista de alto nível da arquitetura física do sistema, as suas componentes, as suas interações e implementação.
Do lado do utilizador, através de um navegador o utilizador vai aceder à plataforma que está hospedada no servidor utilizando protocolos HTTP.
O servidor que está a hospedar a plataforma, este servidor seria uma VM localizada no IT, conecta-se então por LAN ao switch Aruba (componente descrito no capítulo 2), que por sua vez comunica com os nós presentes na rede de testes em causa, sendo que no nosso caso são as APUs que nos foram disponibilizadas.

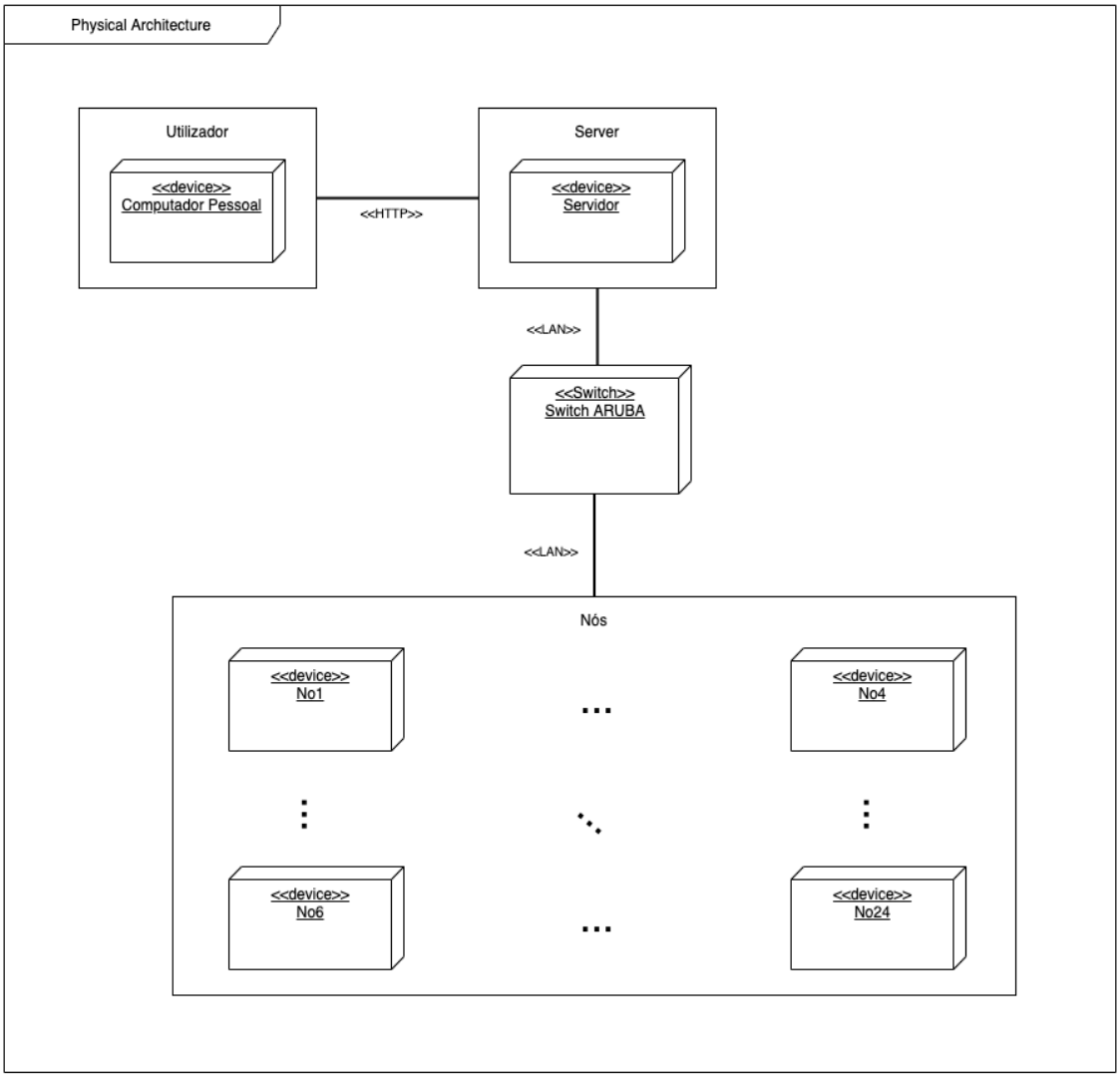

Figura 3.3: Modelo Físico.

É necessário que todas as instâncias sejam acessíveis entre si, com exceção das bases de dados, onde, a Relacional só necessita estar acessível para o servidor principal Flask. O SQLite só necessita estar acessível para o servidor Django. É de realçar que o Switch Aruba teve que ser substituído devido aos impactos causados pelo Covid-19.

3.2.2 Modelo Tecnológico

O modelo tecnológico fornece uma vista sobre as tecnologias utilizadas pelo sistema. A figura seguinte demonstra como estas se relacionam entre si.

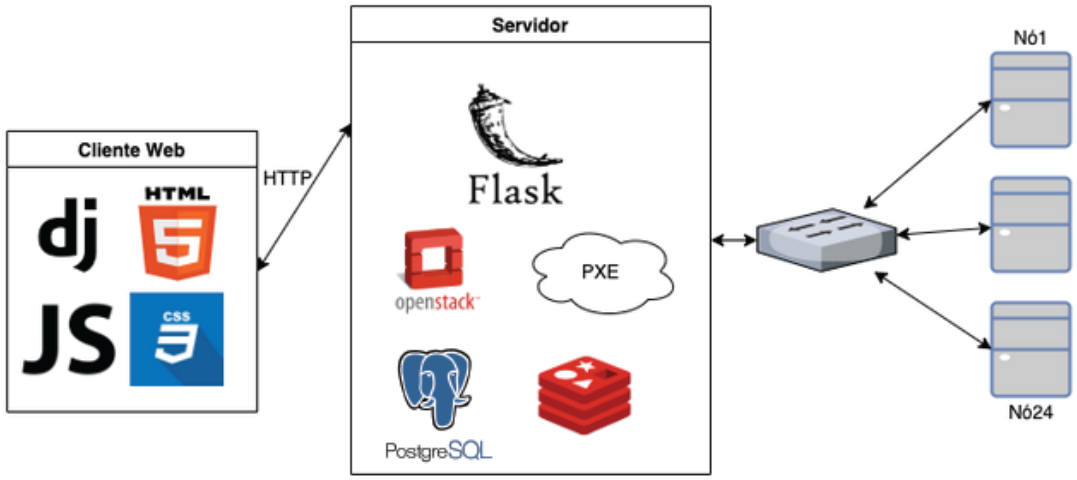

Figura 3.4: Modelo Tecnológico.

No lado do frontend, foi utilizada a framework Django que funciona sob a linguagem Python juntamente com HTML, JavaScript e CSS o que nos permitiu o desenvolvimento de uma plataforma web intuitiva e responsiva.
No que toca ao armazenamento de dados, este é feito do lado do servidor e através do PostgreSQL - projeto open-source baseado em MySQL.
Todas as ferramentas visíveis na imagem foram apresentadas e discutidas no capítulo 2. No que toca à sua implementação, razão de utilidade e a forma como interagem umas com as outras, no caso particular do projeto, serão discutidas abaixo.

CAPÍTULO 4

Implementação

Esta secção apresenta a implementação das ferramentas acima mencionadas e a forma como estas foram utilizadas no contexto do projeto, desde a sua interação com outras ferramentas até aos seus benefícios para a solução. As seguintes subseções falam, em primeiro lugar, da camada de dados e como esta está organizada de modo a suportar o armazenamento de dados necessitado para o projeto.
Em segundo lugar, descreve a camada de processamento e como a comunicação e a troca de dados é feita entre as várias secções da plataforma, como por exemplo, entre o frontend e a base de dados.
Por fim descreve o frontend e como este está estruturado de modo a cumprir os objetivos que foram definidos para o projeto e também fala acerca dos nós/APUs, como estes estão configurados e como eles são acedidos e reconfigurados pelo utilizador da plataforma que pretende realizar experiências.

4.1 Camada de dados

Nesta secção vamos discutir com detalhe a implementação da camada de dados. Em primeiro lugar, demonstramos o modelo de dados da base de dados.
No bloco dos utilizadores, temos as seguintes tabelas:

- **Profile** – esta tabela modela um utilizador e as suas informações;
- **Experience** – esta tabela modela uma experiência como por exemplo a sua data de início e de fim;
- **APU_Config** - define a configuração a fornecer a APU para uma dada experiência;
- **APU** – tabela que contém a informação acerca de cada APU da rede como por exemplo o IP;
- **Role** – cada utilizador pode ser de vários roles como por exemplo “Advanced” e “Beginner”

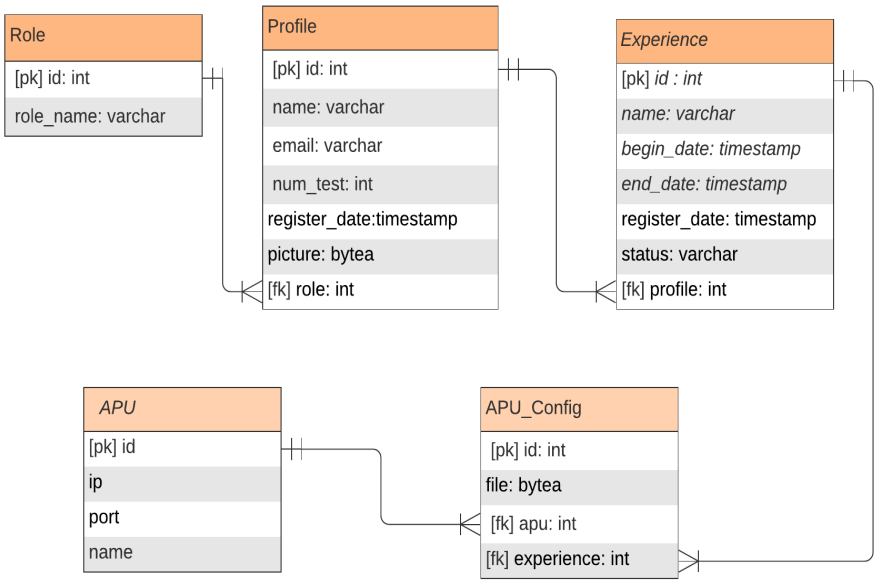

Figura 4.1: Modelo de dados (bloco do utilizador).

4.2 Camada de processamento

Esta secção apresenta o servidor principal da plataforma construído utilizando a framework Flask. Esta camada é responsável pela gestão do sistema, prover e receber dados do FrontEnd, consumir e atualizar a base de dados. Além de interagir com a Base de Dados, esta camada também interage com as APUs, inicializando automaticamente experiências e atuando como um proxy para disponibilizar informações das APUs em tempo real.

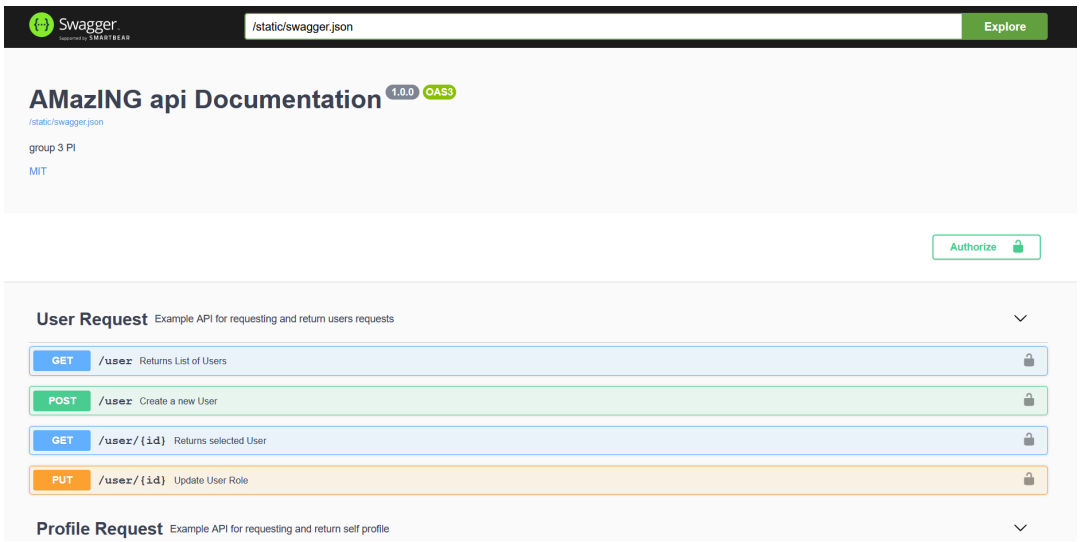

Figura 4.2: Fragmento da documentação do Servidor utilizando o swagger.

4.3 Fluxo de dados

Esta secção, contém os fluxos de informação que passam ou se iniciam pelo servidor. Inicializando pelos pedidos tipo GET, que buscam informação da base de dados e retornam os dados , tratada de acordo com o necessário. Este tipo de operação não realiza alteração na informação da base de dados.

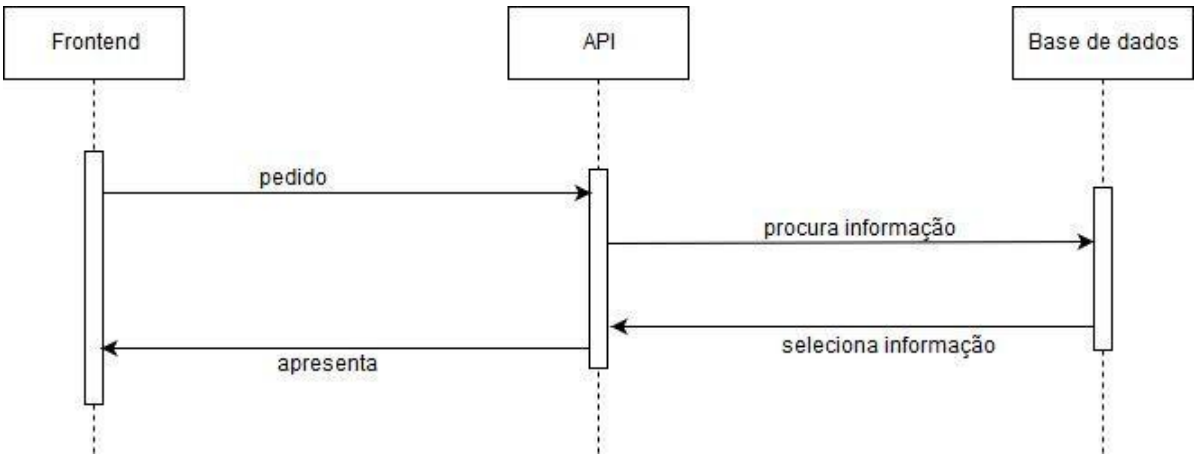

Figura 4.3: Fluxo de comunicação para o método GET.

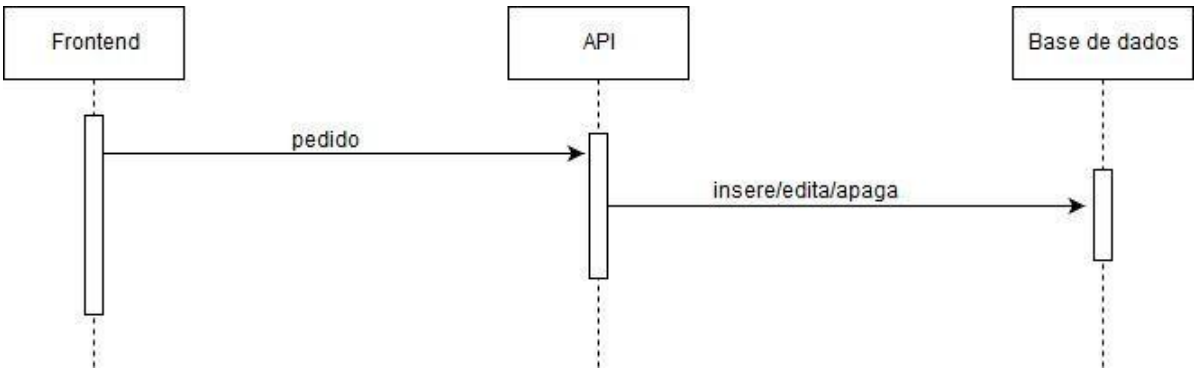

Figura 4.4: Fluxo de comunicação para os métodos POST, PUT e DELETE.

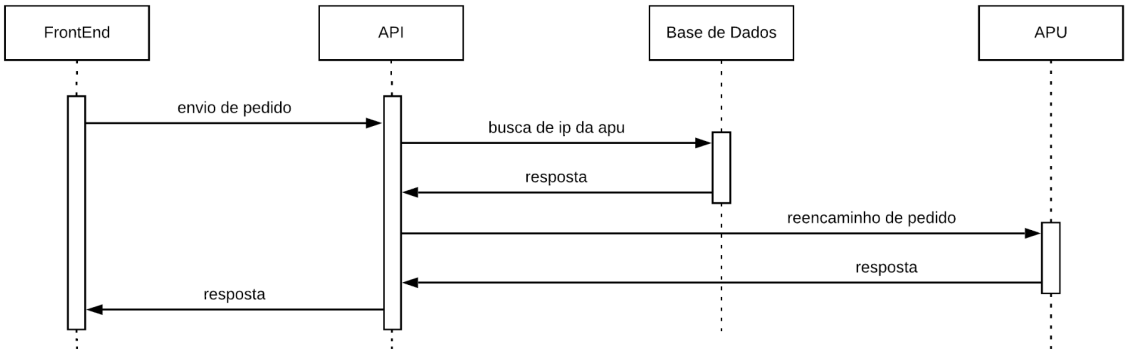

Figura 4.5: Fluxo de comunicação Proxy, para os métodos GET e POST.

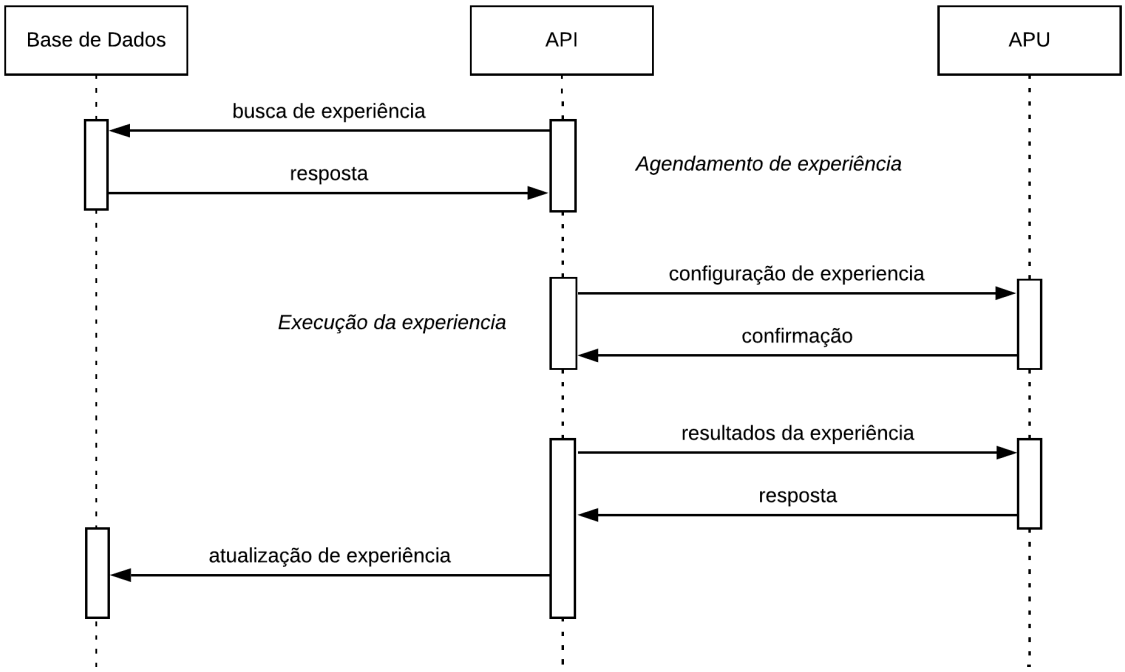

Figura 4.6: Fluxo de comunicação Proxy, para os métodos GET e POST.

4.4 Frontend

INSERIR TEXTO

4.5 Nós/APU

ISERIR

CAPÍTULO 5

Conclusão e Trabalho Futuro

O AMazING é uma plataforma de testes para redes em ambiente real. Permite a realização de experiências por utilizadores com diferentes níveis de conhecimento, utilizando configurações predefinidas ou configurando manualmente.
Estando o projeto num estado concluído, a equipa sente que os principais objetivos foram concretizados. Foram adquiridos conhecimentos durante a realização do projeto que foram considerados uma mais valia para o futuro. Em particular, na área de redes, configuração das mesmas e no desenvolvimento de APIs.
Apesar de terem surgido alguns desafios como, por exemplo, a pandemia, a equipa sente-se satisfeita com o trabalho realizado e com o conhecimento adquirido no decorrer deste projeto. Contudo, existem algumas funcionalidades que poderiam ser melhoradas e exploradas futuramente:

- Aperfeiçoar a página de estatísticas;
- Implementar o suporte a PXE e PoE;
- Implementar a comunicação com o Switch ARUBA e a rede presente no IT;
- Testar a API quando esta utiliza todos os nós;
- Implementar isolamento para um ambiente multi-teste;

ANEXO A

Acrónimos

LEI	Licenciatura em Engenharia Informática
UA	Universidade de Aveiro
RDBMS	Relational Database Management System
SQL	Structured Query Language
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
WWW	World Wide Web
UI	User Interface
CaU	Casos de Uso
REST	Representational State Transfer
API	Application Programming Interface
IT	Instituto de Telecomunicações
VM	Máquina Virtual
IaaS	Infrastructure as a service (Infraestrutura como Serviço)
GUI	Graphic User Interface (Interface Gráfica do Utilizador)
POE	Power over Ethernet
JWT	Json Web Token
SDN	Software-defined networking

universidade de aveiro