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**
**June 2020**

André Santos, NMec 84811
andresgomes@ua.ptD
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 robot 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. No entanto, cada um destes sistemas aborda uma tarefa ou comportamento que uma percentagem considerável da população adota enquanto que o nosso caso particular é um pouco mais "exclusivo". E por esse mesmo motivo há uma amostra menor de plataformas que têm como objetivo auxiliar estes utilizadores.

## 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 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 que 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 multi-teste.

## 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.

# 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. 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 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.

### 2.0.2 AMazING 2019

Este projeto foi desenvolvido anteriormente na UA 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 o projeto base, entenda-se o AMazING, seja o mesmo que o apresentado 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. 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 e intuitiva.

## 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 teve 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 desenvolvida uma vez que, devido ao covid, o desenvolvimento do projeto foi afetado. Tendo em vista isto, foi necessário arranjarmos 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 os 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 deployment 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 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.

## 2.2 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.

### 2.2.1 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.

### 2.2.2 OpenStack

O OpenStack é uma plataforma de computação em nuvem 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.

### 2.2.3 Preboot Execution Environment

O Preboot Execution Environment (PXE ou pixie) é um ambiente para inicializar computadores através da Interface da Placa de Rede, sem a dependência da disponibilidade de dispositivos de armazenamento (como Disco Rígidos) ou algum Sistema Operativo instalado. Desta forma, o Sistema Operativo do equipamento é carregado pela interface de rede toda vez que o mesmo é ligado, evitando assim o uso de unidades de armazenamento local e ou ação de atualização para cada equipamento.
Para o projeto, após a realização de uma experiência, o ambiente do de todas as APUs seria reciclado, evitando que experiências realizadas anteriormente afetem a realização de outras.
Para este projeto, não foi possível carregar os ambientes para as APUs através do PXE

devido às dificuldades de acesso a BIOS das APUs, o que era necessário para a inicializar as APUS com o sistema operacional alocado no servidor PXE.

### 2.2.4 Docker

Docker é um software contêiner da empresa Docker, Inc, que fornece uma camada de abstração e automação para virtualização de sistema operacional no Windows e no Linux, usando isolamento de recurso do núcleo do Linux como cgroups e espaços de nomes do núcleo, e um sistema de arquivos com recursos de união, como OverlayFS criando contêineres independentes para executar dentro de uma única instância do sistema operacional, evitando a sobrecarga de manter máquinas virtuais.
Foram criadas várias instâncias do docker para os distintos serviços a disponibilizar, isto é, foi criada uma instância para a Base de Dados PostgreSQL, uma para a API Flask que serve o Frontend, uma instância de Django no qual é executado o frontend e uma outra instância que disponibiliza o serviço de WebSSH serviço este que é baseado num container que corre uma instância de Python 3.7 com a devida instalação dos requisitos via Pip3.

### 2.2.5 APU

### 2.2.6 Switch Aruba

### 2.2.7 Github

O versionamento 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 neste caso específico, 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 diferentes diretórios sendo cada um destes diretórios um serviço. É de salientar que cada serviço foi desenvolvido no seu branch específico e após validação do mesmo o código seria colocado na master.

# 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.
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;

### 3.1.1 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. Importante referir que a execução de vários testes em simultâneo seria possível desde que fossem utilizados nós distintos. Por questões de simplicidade, sempre que um utilizador pretender reservar uma experiência, todos os nós ficam reservados para ele durante um determinado espaço de tempo. (é inexequível). No menu principal está disposta a grelha de nós e as várias operações possíveis estão divididas em secções na barra de navegação e cada secção pode ter subseções que permitem realizar funcionalidades mais específicas.

### 3.1.2 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.

### 3.1.3 Casos de Uso

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

#### 3.1.3.1 Utilizador/Tester

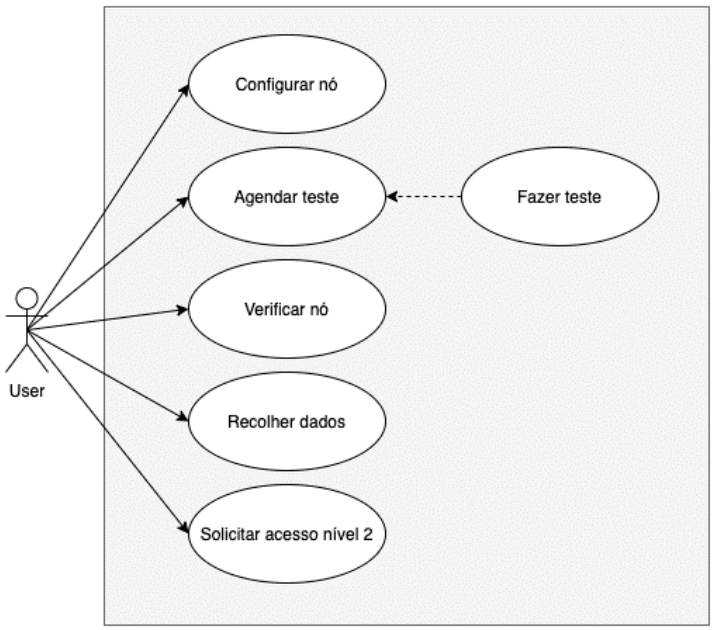


**Figura 3.1:** Casos de uso do utilizador/Tester.

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.

#### 3.1.3.2 Administrador

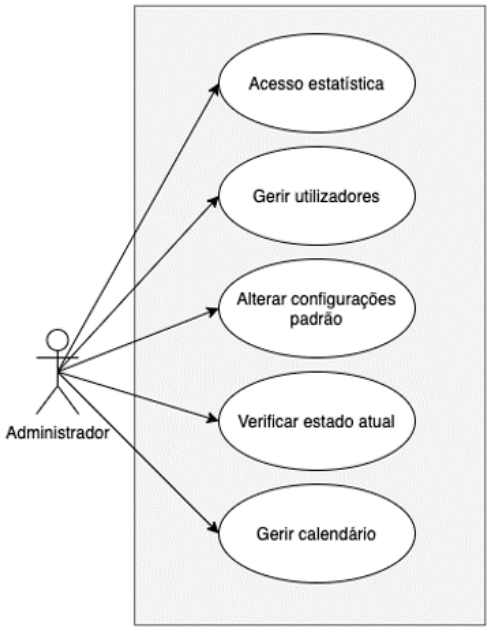


**Figura 3.2:** Casos de uso do Administrador.

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.

### 3.1.4 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.

#### 3.1.4.1 Suposições e Dependências

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

| **Hardware** | **Software** |
|---|---|
| • APUs | • Base de Dados Relacional de linguagem SQL<br>• Flask<br>- No servidor principal<br>- Em cada uma das APUs<br>• Django + SQLite<br>• Web SSH server |

**Tabela 3.3:** Componentes necessárias para o funcionamento.

### 3.1.5 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.

### 3.1.6 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.

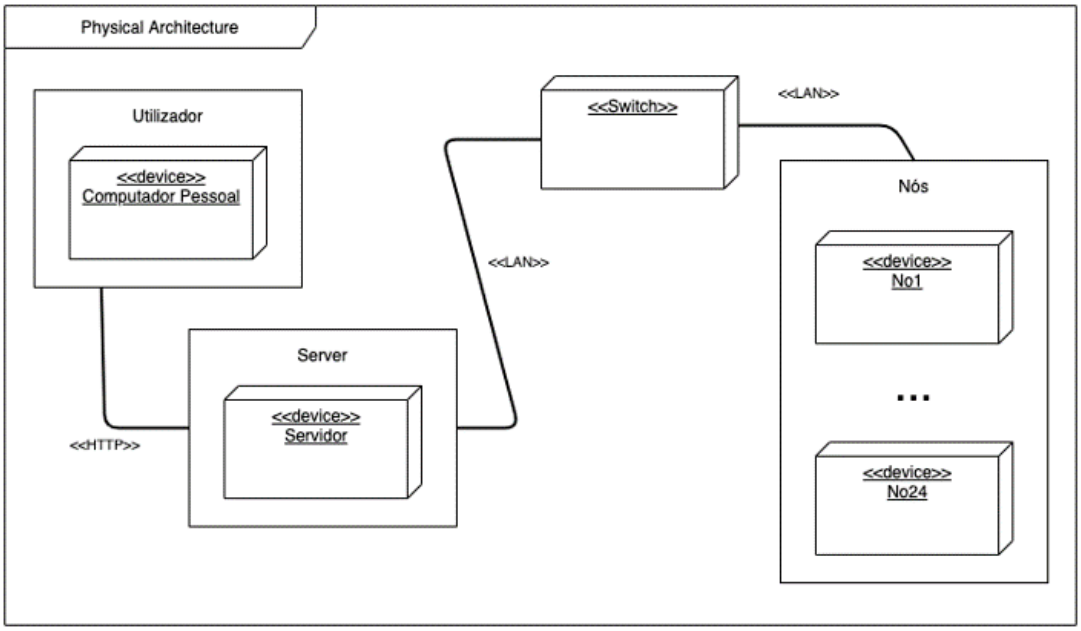


**Figura 3.3:** Modelo Físico.

### 3.1.7 Modelo Tecnológico

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

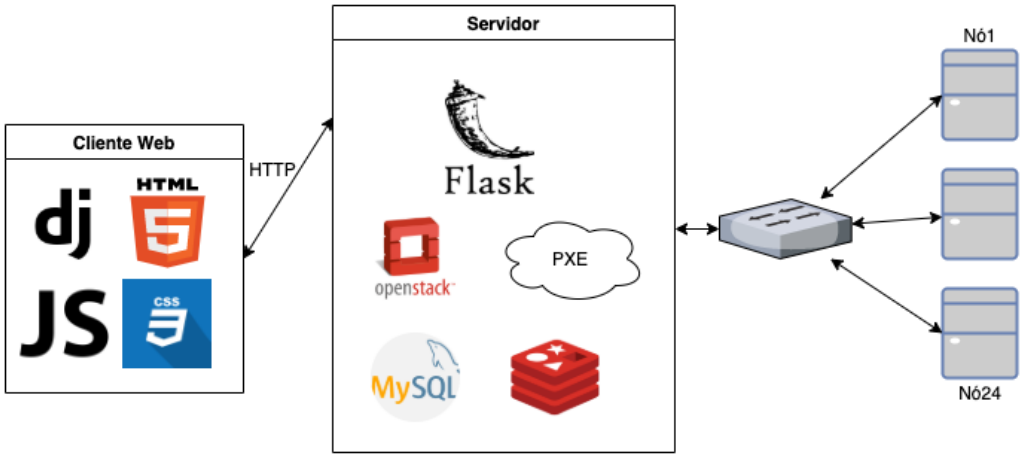


**Figura 3.4:** Modelo Tecnológico.

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

# Conclusão e Trabalho Futuro

Morbi pharetra ligula integer mollis mi nec neque ultrices vitae volutpat leo ullamcorper. In at tellus magna. Curabitur quis posuere purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Suspendisse tristique placerat feugiat. Aliquam vitae est at enim auctor ultrices eleifend a urna. Donec non tincidunt felis. Maecenas at suscipit orci.

# 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 |

universidade de aveiro