



UNIVERSIDADE  
LUSÓFONA

# Boxit

## Plataforma Geradora de Redes de Computadores Virtuais

Francisco Silva | 21705328

Trabalho Final de Curso | Licenciatura em Engenharia Informática | 24/11/2019

Orientado por: Miguel Tavares

[www.ulusofona.pt](http://www.ulusofona.pt)

## **Agradecimentos**

Aos professores Pedro Alves e Bruno Cipriano pelo início de curso que despertou em mim o interesse e curiosidade pela programação.

Ao professor orientador Miguel Tavares, que, para além de me ter proposto um excelente tema de trabalho final de curso, sempre me desafiou a aprender mais. O seu trabalho e exigência ensinaram-me a não estar satisfeito com o que sei e a procurar sempre melhorar.

## Resumo

O Boxit é uma plataforma que pretende virtualizar redes de computadores com o propósito de auxiliar o desenvolvimento de sistemas distribuídos.

Uma rede de computador é um conjunto de computadores que comunicam entre si. Estas redes são necessárias no desenvolvimento de um sistema distribuído uma vez que estes tipos de sistemas pretendem coordenar estes computadores de forma a realizarem um objetivo comum.

Tipicamente, no desenvolvimento de *software* existe pelo menos um ambiente de testes e um de produção, ou seja, no caso de um sistema distribuído estes correspondem a duas redes de computadores que podem ser muito dispendiosas uma vez que implicam a existência de vários computadores e a sua manutenção.

Tendo isto em conta, ao gerar virtualizações de redes de computadores é possível criar um ou mais ambientes de teste semelhantes ao de produção com um número menor de computadores o que também significa menos manutenção.

**Palavras-chave:** sistema distribuído, rede de computadores, plataforma, virtualização.

# Abstract

Boxit is a platform that aims to virtualize computer networks with the purpose of assisting the development of distributed systems.

A computer network is a set of computers that communicate with each other. These networks are necessary in the development of a distributed system since these types of systems intend to coordinate these computers in order to achieve a common goal.

Typically, in software development there is at least one test and one production environment, that is, in the case of a distributed system, these correspond to two computer networks that can be very expensive since they imply the existence of several computers and their maintenance.

Bearing this in mind, when generating virtualizations of computer networks it is possible to create one or more test environments similar to the production one with a smaller number of computers, which also means less maintenance.

**Keywords:** distributed systems, computer network, platform, virtualization.

# Conteúdo

Capítulo 1 Identificação do Problema .....	1
1.1 Descrição do Problema .....	1
1.2 Exemplo de um Caso Real.....	1
1.3 Estrutura do Documento .....	5
Capítulo 2 Viabilidade e Pertinência .....	6
Capítulo 3 Levantamento e Análise de Requisitos .....	7
3.1 Requisitos Funcionais.....	7
3.2 Requisitos Não Funcionais .....	10
Capítulo 4 Solução Desenvolvida .....	11
4.1 Tecnologias.....	11
4.1.1 Node.js.....	11
4.1.2 Docker .....	12
4.1.3 Node Package Manager (NPM).....	14
4.1.4 JSON Web Token (JWT) .....	14
4.2 Implementação.....	17
Na secção 4.2.1 Modos de Funcionamento .....	18
4.2.1 Modos de Funcionamento .....	19
4.2.2 Modelo de dados.....	19
4.2.3 Autenticação .....	21
Com as entidades mencionadas na secção 4.2.1 Modos de Funcionamento .....	21
4.2.4 Logger.....	22
4.2.5 Importar ficheiros ZIP .....	22
4.2.6 Criar os Docker Containers e a Rede de um Projeto .....	23
Capítulo 5 <i>Benchmarking</i> .....	24
Capítulo 6 Método e Planeamento .....	25
6.1 Proposta de Calendário .....	25
6.2 Cumprimento do Calendário .....	26
Capítulo 7 Resultados.....	27
Capítulo 8 Conclusão e Trabalho Futuro.....	30
Anexo 1 – Inquérito realizado a 38 alunos .....	33
Anexo 2 - Manual de instruções do Boxit .....	37
Glossário.....	39

## Lista de Figuras

Figura 1 - Arquitetura Cliente-Servidor [1].....	1
Figura 2 - Arquitetura do projeto de redes de computadores [2].....	2
Figura 3 - Erro ao iniciar um segundo cliente .....	3
Figura 4 - Erro de ligação do segundo cliente .....	3
Figura 5 - Solução do problema utilizando múltiplos computadores .....	4
Figura 6 - Solução do problema utilizando máquinas virtuais .....	4
Figura 7 - Funcionamento do Node.js [3] .....	12
Figura 8 - Aplicação desenvolvida em Java Spring vs. Node.js [15].....	12
Figura 9 – Dockerfile [4] .....	13
Figura 10 - Máquina virtual vs. Docker container [5] .....	14
Figura 11 - Exemplo JWT [7] .....	15
Figura 12 - Autenticação baseada em sessões [8] .....	16
Figura 13 - Autenticação baseada em JWT [8] .....	17
Figura 14 - Arquitetura da plataforma.....	18
Figura 15 - Diagrama Entidade Relação.....	19
Figura 16 - Funcionamento de uma função de <i>hash</i> .....	20
Figura 17 - Exemplo de <i>salt</i> .....	20
Figura 18 - Logger.....	22
Figura 19 - Utilização de hardware dos containers .....	27
Figura 20 - Servidor do projeto de redes a ser executado dentro de um container .....	28
Figura 21 - Cliente do projeto de redes a ser executado dentro de um container.....	28
Figura 22 Razões para os alunos não se deslocarem aos laboratórios.....	33
Figura 23 - Solução para testar o projeto de redes com múltiplos clientes .....	34
Figura 24 - Regularidade de utilização dos laboratórios .....	34
Figura 25 - Número máximo de clientes utilizados para testar o projeto de redes.....	35
Figura 26 - Razões para não ter testado o projeto com mais clientes.....	35
Figura 27 - Sondagem para saber os alunos queria ter testado os projetos com mais clientes.....	36

## **Anexos**

Anexo 1 – Inquérito realizado a 38 alunos .....	33
Anexo 2 - Manual de instruções do Boxit .....	37

# Capítulo 1

## Identificação do Problema

### 1.1 Descrição do Problema

Desenvolver um sistema distribuído levanta novos desafios tais como garantir a sincronização entre as componentes que o constituem assim como a sua comunicação, caso contrário, estaremos sujeitos a perder dados ou até bloqueios no sistema. Para evitar que isto aconteça é fundamental fase de testes seja intensiva e o mais abrangente possível.

Tipicamente, para testar este tipo de sistemas utilizam-se vários computadores ou máquinas virtuais que tentam simular totalmente os computadores. Contudo, nenhuma destas soluções é ótima, uma vez que ambas limitam o número de componentes do sistema que podem ser inicializados. Isto acontece porque nem sempre é possível arranjar vários computadores dado que são dispendiosos enquanto que, utilizar máquinas virtuais requer muitos recursos de *hardware* o que também nos leva ao problema anterior.

### 1.2 Exemplo de um Caso Real

Tipicamente, em alguns cursos de informática os alunos frequentam uma disciplina de redes de computadores. Em certos casos, nesta unidade curricular, os alunos enfrentam pela primeira vez o desafio de desenvolver um projeto que obedece a uma arquitetura cliente-servidor, por outras palavras, um projeto onde exista um servidor que recebe e responde a pedidos de um ou mais clientes, como mostra a Figura 1.

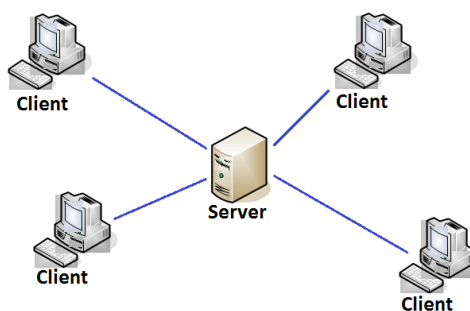


Figura 1 - Arquitetura Cliente-Servidor [1]



Na cadeira de redes de computadores da Universidade Lusófona de Humanidades e Tecnologias do ano letivo 2018/2019 os alunos tiveram de desenvolver um projeto que segue esta arquitetura. Este projeto consiste numa rede de leilões, onde o servidor representa um regulador e os clientes os licitadores.

O regulador é uma entidade cuja principal função é gerir os leilões e os licitadores existentes na rede, ou seja, o regulador está encarregue de criar, listar e terminar leilões.

O licitador possui um *plafond*, com o qual pode licitar. Para além disto, o licitador também pode pedir ao regulador que lhe crie ou liste leilões.

Para fins educativos, um dos requisitos impostos aos alunos foi que os pedidos dos clientes fossem enviados ao servidor através do *Transmission Control Protocol* (TCP) e que o servidor responde-se por *User Datagram Protocol* (UDP), como apresenta a Figura 2.

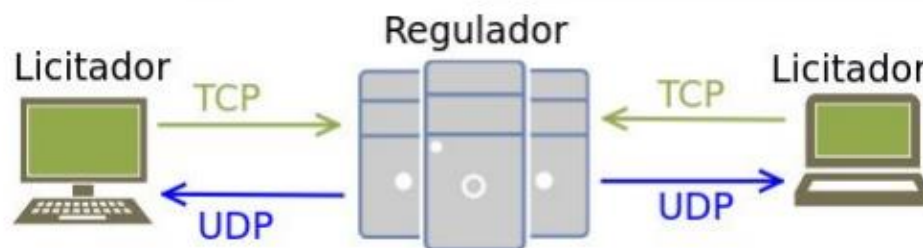


Figura 2 - Arquitetura do projeto de redes de computadores [2]

Tanto o TCP como o UDP são protocolos da camada de transporte cuja principal diferença é que no TCP o cliente negocia uma porta com o servidor para estabelecer uma ligação, enquanto que no UDP o cliente não necessita de estabelecer uma ligação com o servidor, ou seja, utiliza sempre a mesma porta. Esta diferença é a origem do problema.

Como em qualquer projeto que segue uma arquitetura cliente-servidor, os alunos têm o desafio de garantir que a comunicação entre o servidor e cada um dos clientes tem o mínimo de falhas possíveis, logo, uma das formas de testar este caso é iniciar manualmente o servidor e de seguida vários clientes. O problema é que ao iniciar mais que um cliente ocorre um erro como o apresentado na Figura 3.

```

java -jar leilao.jar localhost 3000
Exception in thread "main" java.net.BindException: Address already in use (Bind failed)
    at java.net.PlainDatagramSocketImpl.bind0(Native Method)
    at java.net.AbstractPlainDatagramSocketImpl.bind(AbstractPlainDatagramSocketImpl.java:93)
    at java.net.DatagramSocket.bind(DatagramSocket.java:392)
    at java.net.DatagramSocket.<init>(DatagramSocket.java:242)
    at java.net.DatagramSocket.<init>(DatagramSocket.java:299)
    at java.net.DatagramSocket.<init>(DatagramSocket.java:271)
    at leilao.licitador.Cliente.<init>(Cliente.java:27)
    at leilao.licitador.Main.main(Main.java:6)

```

Figura 3 - Erro ao iniciar um segundo cliente

Este erro acontece porque o primeiro e o segundo clientes foram iniciados no mesmo computador e estão a tentar usar o mesmo porto, ou seja, quando o primeiro cliente é iniciado, este ocupa um determinado porto, de seguida, quando o segundo cliente é iniciado, este tenta utilizar o mesmo porto o que é impossível. A Figura 4 demonstra este problema.

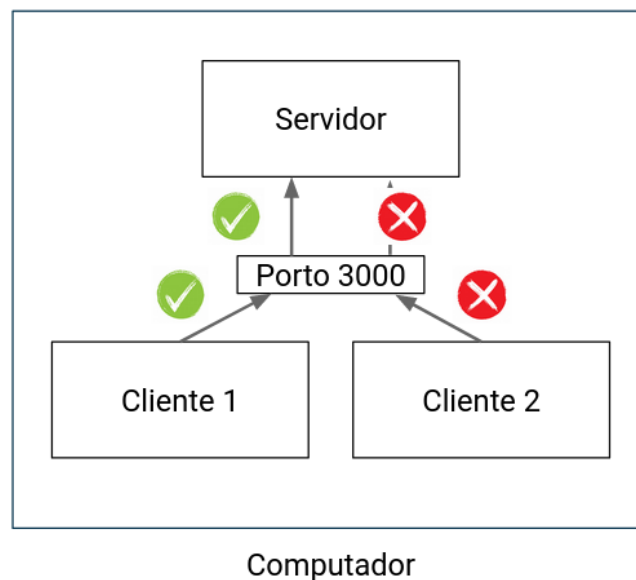


Figura 4 - Erro de ligação do segundo cliente

Para resolver este problema a resposta mais óbvia é utilizar mais do que um computador como apresenta a Figura 5. Esta solução tem desvantagens uma vez que, em geral, os alunos não possuem computadores suficientes, e apesar de existir a possibilidade de utilizar os laboratórios da faculdade, a maior parte dos alunos os alunos não os utiliza como se pode observar no Anexo 1.

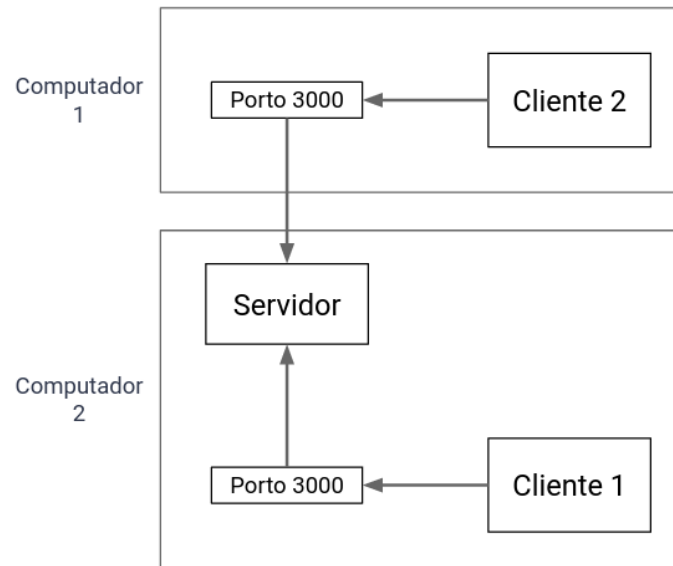


Figura 5 - Solução do problema utilizando múltiplos computadores

Outra solução seria utilizar máquinas virtuais tradicionais como apresenta a Figura 6. O problema desta solução é que utilizar várias máquinas virtuais requer algumas configurações de rede, que nem sempre os alunos dominam, e um computador com muitos recursos de hardware, o que muitas vezes não é possível.

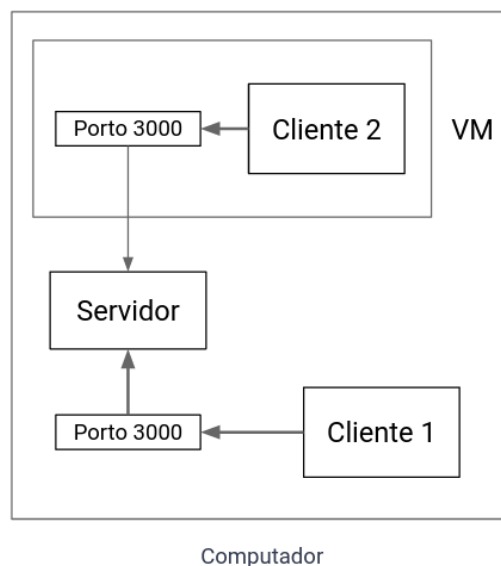


Figura 6 - Solução do problema utilizando máquinas virtuais

Os alunos não são os únicos a sofrer destes problemas. Qualquer programador que pretenda desenvolver um sistema como o do exemplo anterior terá a necessidade de fazer testes, e para tal, terá de arranjar uma solução como as apresentadas anteriormente.

Por isso, com esta plataforma, pretende-se criar uma resposta mais eficiente, isto é, uma plataforma que requeira poucos recursos de hardware em relação às máquinas virtuais e que permita que os programadores testem os seus sistemas distribuídos apenas com um computador.

## 1.3 Estrutura do Documento

Ao longo desta secção apresentarei a forma como este documento se encontra estruturado.

O objetivo do Capítulo 2 Viabilidade e Pertinência é apresentar o impacto que este trabalho tem na resolução do problema. Neste capítulo também se tenciona demonstrar que este projeto tem potencial para continuar o seu desenvolvimento após a conclusão do Trabalho Final de Curso.

No capítulo seguinte pretende-se listar requisitos funcionais e não-funcionais tendo em conta o problema a ser resolvido.

O Capítulo 4 Solução Desenvolvida tem como finalidade apresentar a solução desenvolvida para resolver este problema. Para tal, serão apresentadas as tecnologias utilizadas e o funcionamento da plataforma.

De seguida, o Capítulo 5 *Benchmarking* tem como propósito comparar as funcionalidades da solução proposta face às soluções já existentes no mercado.

O Capítulo 6 apresenta o planeamento efetuado para o desenvolvimento deste projeto.

No Capítulo 7 Resultados são apresentados todos os resultados tendo em conta o problema que se pretende resolver.

Por fim, no Capítulo 8 Conclusão e Trabalho Futuro pretende-se apresentar uma conclusão e novas funcionalidades que podem ser implementadas futuramente.

## Capítulo 2

### Viabilidade e Pertinência

Uma das técnicas mais utilizadas para escalar sistemas é dividi-los em várias componentes que funcionam como um só, ou seja, convertê-lo num sistema distribuído. Esta conversão acrescenta alguma complexidade como a comunicação e a sincronização desses componentes. Isto torna a realização de testes indispensável.

Contudo, este tipo de sistemas requer uma rede de computadores para o seu funcionamento, isto é, um conjunto de computadores interligados uns aos outros para iniciar cada um dos componentes. Por outros termos, dependendo do tamanho do sistema, poderá ser necessário uma quantidade enorme de computadores.

Como nem toda a gente possui mais que um computador, uma das formas de testar este sistema é com recurso a máquinas virtuais. O problema desta solução é que o número de componentes que podem ser inicializados está limitado ao número de máquinas virtuais que o computador suportar, e visto que uma máquina virtual pode necessitar muitos recursos de *hardware*, esse número poderá ser muito pequeno. Outra opção é comprar os computadores necessários, o que muitas vezes não é possível por falta de recursos financeiro. Para além disso, é um investimento que pode não ser justificado, como é o caso de um aluno comprar vários computadores para fazer um projeto de uma disciplina, ou arriscado, uma vez que, após a realização dos testes, pode-se chegar à conclusão que o sistema não é viável.

A plataforma que se pretende desenvolver neste projeto requisitará que o utilizador possua apenas um computador e permitir-lhe-á iniciar um número maior de componentes do que as máquinas virtuais.

## Capítulo 3

### Levantamento e Análise de Requisitos

A recolha e análise de requisitos deu origem a este capítulo que está dividido em duas secções, a 3.1 Requisitos Funcionais e a 3.2 Requisitos Não Funcionais.

Os 3.1 Requisitos Funcionais, apresentados na secção 3.1 Requisitos Funcionais, descrevem as ações que o software deve ser capaz de realizar.

Na secção 3.2 Requisitos Não Funcionais são listados os Requisitos Não Funcionais, estes especificam como é que o software deve realizar as ações.

### 3.1 Requisitos Funcionais

#### RF1 – Registo do utilizador

Pré-Condição:

O utilizador deve ter uma conta de *email* e acesso à mesma.

Requisito:

O utilizador deve inserir o seu *email*, *password*, primeiro nome e último nome para se registar no sistema.

Pós-Condição:

O sistema verifica se o *email* inserido já foi registado e caso não tenha sido, o sistema regista um novo utilizador com a conta por ativar e envia um *email* que contém um *link* que lhe permite ativar a conta, caso contrário, o sistema pede ao utilizador que insira um novo *email*.

#### RF2 – Ativar conta

Pré-Condição:

O utilizador deve ter executado o registo e recebido o *email* de ativação.

Requisito:

O utilizador ao carregar no *link* que lhe foi enviado previamente para o *email* deve ficar com a conta ativa.

Pós-Condição:

O sistema muda o estado da conta do utilizador para ativa.

### **RF3 – Autenticar utilizador**

Pré-Condição:

O utilizador deve estar registado para se poder autenticar e ter a conta ativada.

Requisito:

Para se autenticar, o utilizador insere o seu *email* e *password*.

Pós-Condição:

O sistema valida as credenciais inseridas, ou seja, verifica se o *email* inserido existe e se a *password* inserida corresponde à *password* associada a esse *email* no sistema.

Caso as credenciais do utilizador estejam incorretas, o sistema não autêntica o utilizador e permite ao utilizador tentar novamente.

### **RF4 – Recuperar *password***

Pré-Condição:

O utilizador deve estar registado e ter a conta ativada para poder recuperar a *password*.

Requisito:

O sistema deve permitir ao utilizador recuperar a *password*, para tal, o sistema pede ao utilizador que insira o *email* da conta da qual o utilizador pretende recuperar a *password*.

Pós-Condição:

Caso o utilizador insira um *email* existente no sistema, o sistema enviará um *email* para o *email* inserido com um *link* para o utilizador alterar a *password*.

### **RF5 – Permissões de acesso dos utilizadores**

Pré-Condição:

O utilizador deve estar registado e ter a conta ativada para poder recuperar a *password*.

Requisito:

Deve existir uma forma de controlar que tipo de acesso é que os utilizadores têm.

Pós-Condição:

O sistema deve dar acesso aos utilizadores tendo em conta as permissões que este possui.

## **RF6 – Importar componentes de um projeto através de ficheiros JAR (Java Archive)**

### **Pré-Condição:**

O utilizador deve estar autenticado, ter a conta ativada e possuir no seu computador um ficheiro JAR com o projeto que pretende importar.

### **Requisito:**

O sistema deve permitir ao utilizador importar ficheiros JAR que correspondem a uma componente do seu projeto.

### **Pós-Condição:**

O sistema verifica se os ficheiros são do tipo JAR, se não forem, o sistema pede ao utilizador que selecione novamente os ficheiros a importar, caso contrário importa os ficheiros.

## **RF7 – Importar projeto ZIP**

### **Pré-Condição:**

O utilizador deve estar autenticado, ter a conta ativada e ter um ficheiro ZIP que contenha o projeto, ou seja, todas as componentes do projeto.

### **Requisito:**

Tal como na importação de ficheiros JAR, o sistema de permitir ao utilizador importar ficheiros ZIP, para tal permite que o utilizador insira um ficheiro deste tipo.

### **Pós-Condição:**

O sistema verifica se o ficheiro importado é do tipo ZIP, caso seja, o sistema importa o ficheiro e descomprime-o, caso contrário o sistema pede ao utilizador que insira novamente o ficheiro.

## **RF8 – Importar projeto do GitHub**

### **Pré-Condição:**

O utilizador deve estar autenticado, ter a conta ativada, ter uma conta no GitHub e um repositório com um projeto.

### **Requisito:**

O sistema permite que ao utilizador que importe projeto de repositórios do GitHub.

### **Pós-Condição:**

O sistema verifica se o repositório existe e importa o projeto.



### **RF9 – Executar múltiplas instâncias de cada componente do projeto**

Pré-Condição:

O utilizador deve estar autenticado, ter a conta ativada e ter importado o projeto.

Requisito:

O utilizador pode selecionar o número de instâncias de várias componentes de um projeto que pretende executar.

Pós-Condição:

O sistema executa as instâncias selecionadas pelo utilizador.

### **RF10 – Acesso às instâncias executadas através do protocolo SSH (Secure Socket Shell), via *browser***

Pré-Condição:

O utilizador deve estar autenticado, ter a conta ativada, ter importado o projeto e executado múltiplas instâncias referentes a componentes de um projeto.

Requisito:

O utilizador deve ter acesso a cada uma das individualmente via *browser*, através do protocolo SSH.

Pós-Condição:

Para cada instância, o sistema abre um novo separador no *browser* permitindo ao utilizador interagir com as instâncias através do protocolo SSH.

## **3.2 Requisitos Não Funcionais**

### **RNF1 – Adaptação a diferentes camadas de acesso de dados**

O sistema não deve estar restrito a um único Sistema de Gestão de Bases de Dados Relacionais (SGBDR), ou seja, o sistema deve funcionar da mesma forma com um SGBDR MySQL, PostgreSQL, ou com outro tipo de SGBDR. Para além disso deve ser possível alterar o SGBDR através de um ficheiro de configurações.

### **RNF2 – Acesso ao sistema**

Apenas utilizadores autenticados, com uma conta ativa e com as devidas permissões têm acesso ao sistema.

## Capítulo 4

### Solução Desenvolvida

Neste capítulo serão apresentadas, na secção 4.1 Tecnologias, as tecnologias escolhidas para o desenvolvimento deste trabalho com a respetiva justificação. Seguidamente, na secção 4.2 Implementação explica-se o funcionamento da plataforma.

#### 4.1 Tecnologias

Nesta secção serão apresentadas as tecnologias utilizadas para o desenvolvimento deste projeto.

Na secção 4.1.1 Node.js pretende-se descrever o funcionamento do Node.js e as suas vantagens comparativamente a outras tecnologias do mesmo tipo.

De seguida, na secção 4.1.2 Docker pretende-se explicar o que é o Docker, uma Docker image, um Docker container e quais os benefícios de utilizar esta tecnologia em comparação com máquinas virtuais tradicionais.

Depois, a secção 4.1.3 Node Package Manager (NPM) destina-se a explicar a pertinência do uso do NPM neste projeto.

Por fim, na secção 4.1.4 JSON Web Token (JWT) é dado a conhecer o JSON Web Token e as suas características.

##### 4.1.1 Node.js

O Node.js é uma plataforma para construir *backends* escaláveis de alta *performance* usando JavaScript. A principal característica e diferença do Node.js em relação a outras tecnologias é a execução de requisições/eventos em apenas uma *thread* de forma assíncrona, como apresenta a Figura 7.



Figura 7 - Funcionamento do Node.js [3]

Para desenvolver este trabalho existem outras alternativas ao Node.js como o Java Spring MVC. A razão pela escolha do Node.js é a sua eficiência em relação a outras tecnologias. A Figura 8 ilustra um exemplo muito conhecido do PayPal que inicialmente utilizou Java Spring para desenvolver a sua aplicação e ao mudar para Node.js verificou que o número de páginas disponibilizadas por segundo quase duplicou.

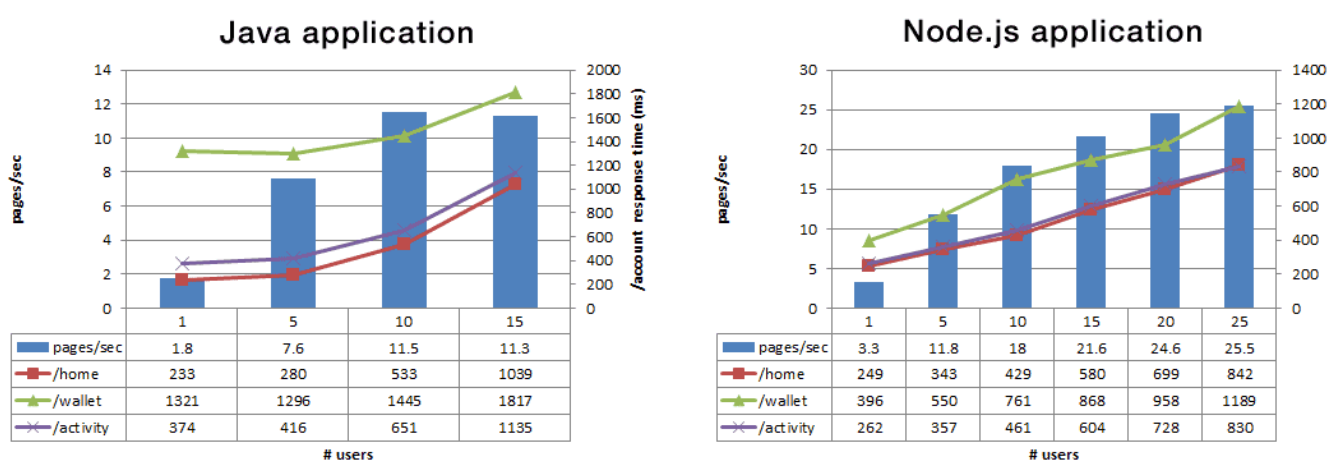


Figura 8 - Aplicação desenvolvida em Java Spring vs. Node.js [15]

## 4.1.2 Docker

O Docker é uma ferramenta *open source* de virtualização, que utiliza Docker *images* para guardar todas as instruções, configurações e dependências de uma aplicação.

Tal como um ficheiro ISO representa perfeitamente o conteúdo de um programa, uma Docker *image* representa o conteúdo de um Docker container, isto é, uma Docker

*image* guarda todas as instruções, configurações e dependências necessárias de um determinado Docker *container*. Essas instruções, configurações e dependências são definidas através de um Dockerfile apresenta a Figura 9. Nesse exemplo podemos observar que a Docker *image* gerada por este ficheiro é baseada em Ubuntu 16.04, de seguida será atualizada e ao gerar um Docker *container* irá executar o comando *bash*.

```
ARG CODE_VERSION=16.04
FROM ubuntu:${CODE_VERSION}
RUN apt-get update -y
CMD ["bash"]
```

Figura 9 – Dockerfile [4]

Um Docker *container* é uma instância de uma Docker *image*, ou seja, é uma virtualização com todas as configurações e dependências definidas pela Docker *image*.

Em comparação com uma máquina virtual, um *container* consome menos recursos de *hardware*. Tal como é apresentado na Figura 10, isto acontece porque, enquanto uma máquina virtual cria uma virtualização do sistema operativo e do *hardware*, um Docker *container* partilha o sistema operativo do *host* e não virtualiza o *hardware*, tornando desnecessária a existência do *Hypervisor*, que é o responsável pela gestão da virtualização do *hardware*.

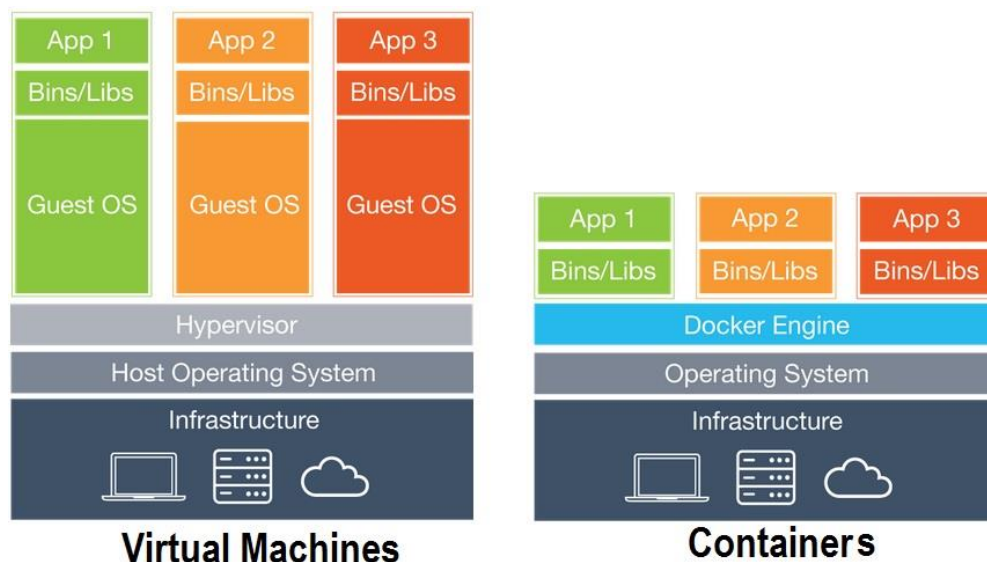


Figura 10 - Máquina virtual vs. Docker container [5]

### 4.1.3 Node Package Manager (NPM)

Uma vez que neste projeto se pretende tirar proveito de tecnologias como 4.1.1 Node.js e **Erro! A origem da referência não foi encontrada.**, é conveniente utilizar uma ferramenta como o NPM.

O NPM, como o nome indica é um gestor de pacotes, ou seja, é uma ferramenta que serve para instalar, desinstalar, atualizar e configurar pacotes.

Para além de gerir pacotes, o NPM também permite definir um conjunto de comandos a executar. Isto é bastante útil para tarefas como executar múltiplos testes, onde é possível selecionar todos os comandos necessários para concluir esta tarefa e associá-los a um comando mais simples que ao ser executado vai executar todos os comandos, previamente selecionados automaticamente.

### 4.1.4 JSON Web Token (JWT)

O JWT é um padrão IETF (RFC 7519 [6] ) que define um formato compacto e autocontido para propagar informação de forma segura entre entidades.

Um JWT é composto por três campos codificados em *Base64-URL* e separados por pontos:

- O *header*, que, tipicamente, consiste no tipo de *token* e no algoritmo a ser utilizado;
- O *payload* é a informação a ser transferida.
- A *signature* que é formado através da concatenação entre o *header* e o *payload* com um ponto a separá-los. De seguida, a string resultante é utilizada para gerar a *signature* através do algoritmo definido no *header* e uma chave privada.

Na Figura 11 é possível observar um JWT onde cada um dos três campos está representado com uma cor diferente.

The image shows a web-based JWT decoder interface. It is divided into two main sections: 'Encoded' and 'Decoded'.

**Encoded Section:** Labeled 'PASTE A TOKEN HERE', it contains a long string of base64-encoded characters: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHM I6arZ3Y922BhjWgQzWXcXNrZ0ogtVhfEd2o`. The string is color-coded: the first part (header) is red, the second part (payload) is green, and the third part (signature) is blue.

**Decoded Section:** Labeled 'EDIT THE PAYLOAD AND SECRET', it displays the decoded components of the token.

- HEADER: ALGORITHM & TOKEN TYPE:** Shows a JSON object: `{ "alg": "HS256", "typ": "JWT" }`. The `"alg"` value is red and `"typ"` is green.
- PAYLOAD: DATA:** Shows a JSON object: `{ "sub": "1234567890", "name": "John Doe", "iat": 1516239822 }`. The `"sub"` value is red, `"name"` is green, and `"iat"` is blue.
- VERIFY SIGNATURE:** Shows the HMACSHA256 algorithm being used to verify the signature. It includes a text input for the secret key, currently containing the word 'secret'. Below the input, there is a checkbox labeled 'secret base64 encoded' which is currently unchecked.

Figura 11 - Exemplo JWT [7]

O JWT vai ser utilizado na autenticação dos utilizadores. A vantagem disto é que o servidor não necessita de guardar o estado dos utilizadores, ao contrário do que acontece, por exemplo, num sistema de autenticação baseado em sessões. Neste tipo de sistema, o servidor cria sessões e guarda-as em memória. De seguida, envia o id da sessão ao utilizador que o armazena num *cookie*. Enquanto estiver autenticado, o utilizador envia este *cookie* ao servidor em todos os seus pedidos. O servidor compara a informação da sessão guardada em memória com o id da sessão guardada no *cookie*. **A Erro! A origem da referência não foi encontrada.** apresenta este processo.

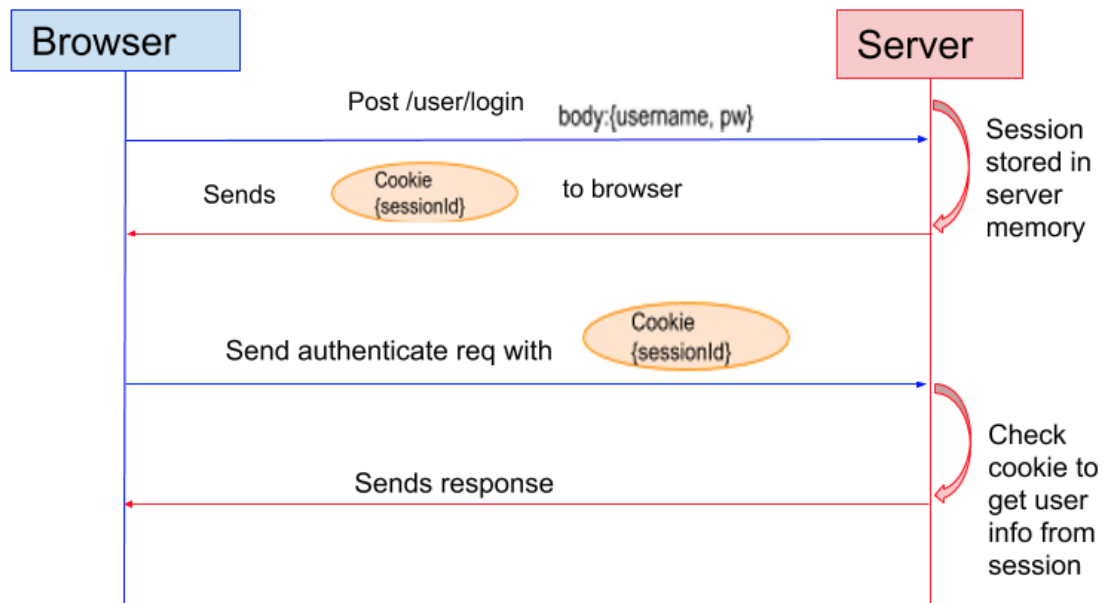


Figura 12 - Autenticação baseada em sessões [8]

No caso implementado neste projeto, representado na **Erro! A origem da referência não foi encontrada.**, o servidor gera um JWT e envia-o ao utilizador quando este se autentica. O utilizador guarda o JWT e envia-o em todos os pedidos que fizer. Uma vez que o servidor é a única entidade que possui a chave dos JWT dos utilizadores significa que estes não podem ser alterados por terceiros. Isto significa que o servidor apenas tem de verificar se os JWT sofreram modificações para confirmar a identidade dos utilizadores.

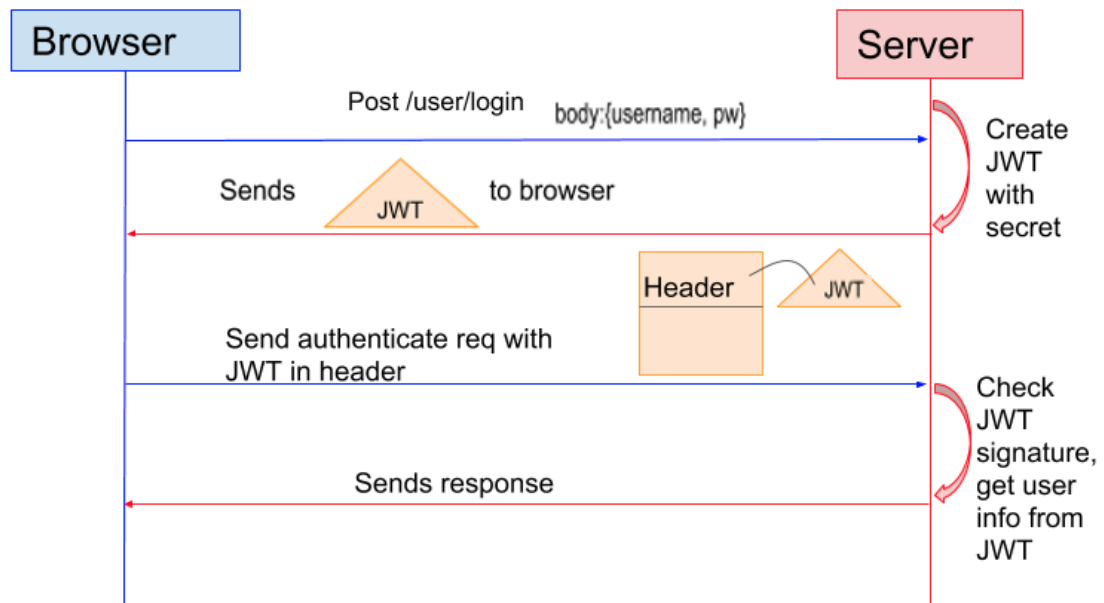


Figura 13 - Autenticação baseada em JWT [8]

## 4.2 Implementação

Esta secção está dividida em várias secções e tem como objetivo explicar o desenvolvimento do sistema. Ou seja, com esta secção pretende-se explicar todas o funcionamento do sistema e as decisões tomadas na sua construção.

Este sistema está dividido em três camadas como a Figura 14 apresenta.



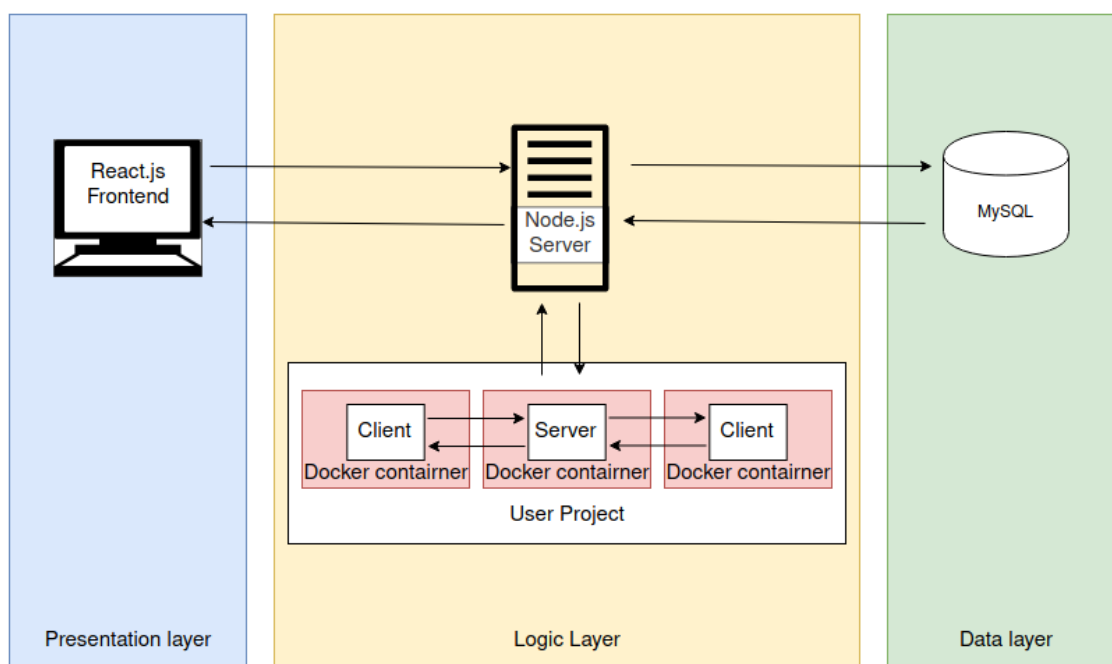


Figura 14 - Arquitetura da plataforma

Na primeira secção serão apresentados os modos de funcionamento desta plataforma e as suas diferenças.

## Na secção 4.2.1 Modos de Funcionamento

O Boxit tem dois modos de funcionamento, o *single user* e o *multi user*. Este modo pode ser facilmente selecionado no ficheiro de configurações.

O modo *single user* permite que o utilizador execute o Boxit localmente, ou seja, todas as suas camadas são inicializadas no seu computador.

O modo *multi user* permite que vários utilizadores usufruam do Boxit. Neste modo as camadas de lógica e de dados são executadas num servidor e os utilizadores apenas utilizam a camada de apresentação para interagir com as outras camadas.

4.2.2 Modelo de dados é explicada a camada de dados. Nesta secção pretende-se dar a entender quais os dados a serem guardados e como é que estes são armazenados.

Na secção 4.2.3 Autenticação é apresentado o sistema de autenticação do projeto.

De seguida, a secção 4.2.4 Logger dá a conhecer o logger e os benefícios da sua implementação.

Por fim, na secção 4.2.5 Importar ficheiros ZIP é descrito o processo de importação de projetos dos utilizadores.

Todo o desenvolvimento do projeto pode ser acompanhado no seguinte repositório: <https://gitlab.com/boxit1/backend/-/tree/develop>

## 4.2.1 Modos de Funcionamento

O Boxit tem dois modos de funcionamento, o *single user* e o *multi user*. Este modo pode ser facilmente selecionado no ficheiro de configurações.

O modo *single user* permite que o utilizador execute o Boxit localmente, ou seja, todas as suas camadas são inicializadas no seu computador.

O modo *multi user* permite que vários utilizadores usufruam do Boxit. Neste modo as camadas de lógica e de dados são executadas num servidor e os utilizadores apenas utilizam a camada de apresentação para interagir com as outras camadas.

## 4.2.2 Modelo de dados

Tendo em conta os requisitos, foi possível identificar várias entidades como apresenta a Figura 15.

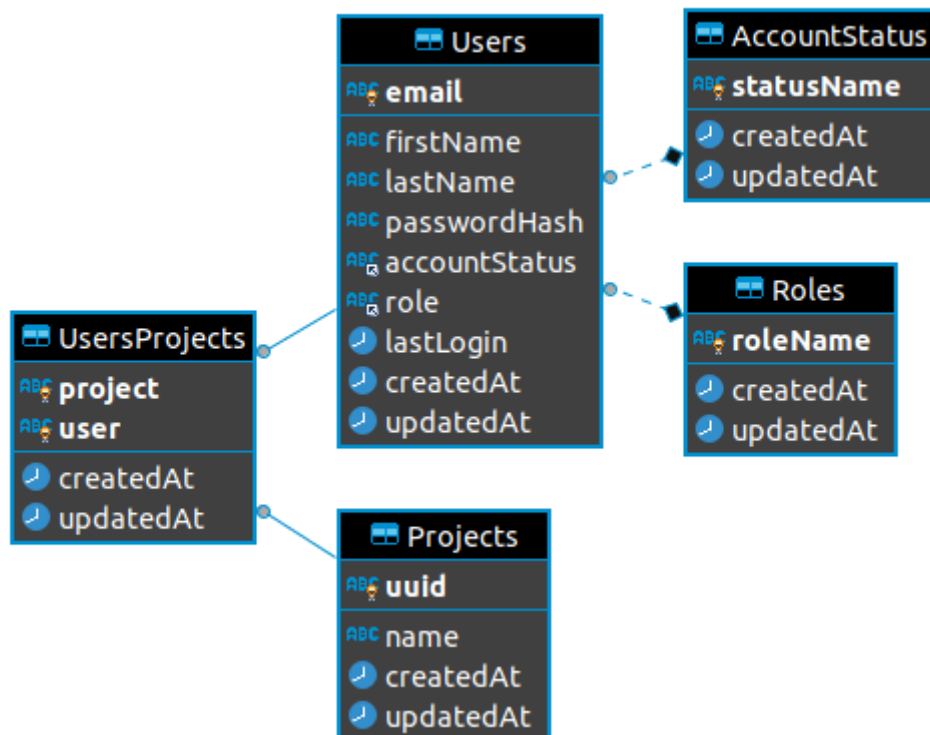


Figura 15 - Diagrama Entidade Relação

Nessa figura é possível observar uma entidade chamada *Users* que corresponde aos utilizadores. A chave primária desta tabela é o *email* uma vez que este é único e todos os utilizadores necessitam de um para se registar. Nesta tabela também existe uma coluna chamada *passwordHash* que, em conjunto com a coluna *email*, serve para autenticar os utilizadores. Os valores desta coluna não correspondem às *passwords* dos utilizadores, mas sim a palavras geradas através das *passwords* e de uma função de *hash*. Este tipo de função apenas recebe um valor e transforma-o noutra como apresenta Figura 16.

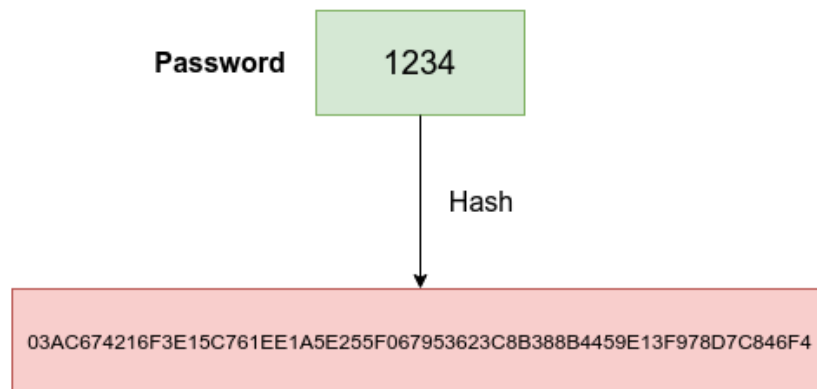


Figura 16 - Funcionamento de uma função de *hash*

Como medida de proteção adicional é acrescentado um *salt* às *passwords*, isto é, concatena-se à *password* com uma *string* aleatória, como apresenta a Figura 17 - Exemplo de *salt* Figura 17. O objetivo disto é que utilizadores que tenham *passwords* iguais não tenham o valor do *passwordHash* igual desta forma caso uma *password* e o *passwordHash* correspondente sejam comprometidas as restantes *passwords* continuem protegidas.

Salt value	String to be hashed	Hashed value = <b>SHA256</b> (Password + Salt value)
E1F53135E559C253	<b>password</b> 123E1F53135E559C253	72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8
84B03D034B409D4E	<b>password</b> 12384B03D034B409D4E	B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A

Figura 17 - Exemplo de *salt*

A tabela *Roles* tem como objetivo definir diferentes tipos de utilizadores que poderão ter diferentes permissões.

A tabela *AccountStatus*, como o nome indica, serve para identificar o estado da conta, ou seja, identificar se a conta está ativa ou por ativar. Esta tabela podia ser uma

coluna do tipo booleano na tabela *Users* mas acabaria por limitar a escalabilidade da plataforma. Um exemplo disso seria acrescentar um novo estado à conta como bloqueada, da forma como foi implementada basta acrescentar mais um elemento à tabela enquanto que se este campo fosse do tipo booleano era necessário fazer muito mais alterações.

Para além destas tabelas existe a tabela *Projects* que tem uma relação *many-to-many* com a tabela *Users*. Esta tabela corresponde aos projetos dos utilizadores e visto que cada utilizador pode ter vários projeto e um projeto pode pertencer a mais que um utilizador foi necessário criar a tabela *UsersProjects* para auxiliar esta relação.

Em todas as tabelas existem as colunas *createdAt*, *updatedAt* e no caso da *Users* existe também a *lastLogin*. Estas colunas são úteis para obter estatísticas, como o tráfego da plataforma, que ajudem a tomada de decisões no desenvolvimento.

### 4.2.3 Autenticação

## Com as entidades mencionadas na secção 4.2.1 Modos de Funcionamento

O Boxit tem dois modos de funcionamento, o *single user* e o *multi user*. Este modo pode ser facilmente selecionado no ficheiro de configurações.

O modo *single user* permite que o utilizador execute o Boxit localmente, ou seja, todas as suas camadas são inicializadas no seu computador.

O modo *multi user* permite que vários utilizadores usufruam do Boxit. Neste modo as camadas de lógica e de dados são executadas num servidor e os utilizadores apenas utilizam a camada de apresentação para interagir com as outras camadas.

4.2.2 Modelo de dados é possível implementar um processo de autenticação. Neste processo, o utilizador insere os dados necessários para o registo e o sistema trata de os validar e armazenar. Para além disto o sistema também envia um email ao utilizador para este ativar a sua conta.

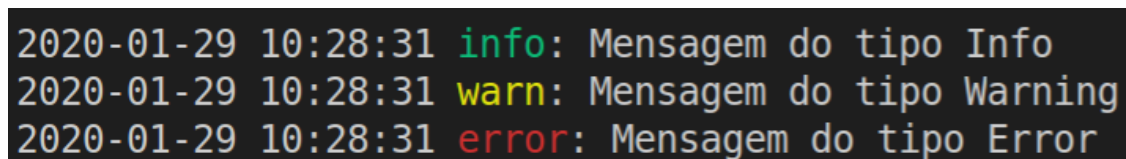
Assim que o utilizador esteja registado e com a conta ativa este pode autenticar-se, para tal, deve introduzir o seu *email* e *password*. De seguida, o sistema valida estes dados, ou seja, verifica se existe um *email* na base de dados que corresponda ao *email* introduzido e, caso exista, compara a cifra guardada na base de dados correspondente a esse *email* com a cifra da *password* inserida pelo utilizador. Caso o *email* não exista ou

as cifras não sejam iguais, o sistema envia uma mensagem de erro, caso contrário, o servidor envia um JWT que contém no seu *payload* um *hash* do seu email. Este JWT deve ser utilizado pelo o utilizador em todos os pedidos HTTP seguintes para garantir a sua identidade.

## 4.2.4 Logger

Uma vez que este sistema é *open source* é importante que o código seja simples, fácil de perceber e que exista boa documentação que explique o seu funcionamento.

Para além disso, qualquer ferramenta que possa ajudar a compreender o projeto pode ser importante, como tal, foi implementado um logger, isto é um sistema que envia mensagens de erro, avisos e informa em relação a qualquer operação executada. Um exemplo dessas mensagens é Figura 18.



```
2020-01-29 10:28:31 info: Mensagem do tipo Info
2020-01-29 10:28:31 warn: Mensagem do tipo Warning
2020-01-29 10:28:31 error: Mensagem do tipo Error
```

Figura 18 - Logger

O logger implementado neste projeto, para além de enviar as mensagens, também as guarda em ficheiros separados pelo tipo de mensagem e pela data da mensagem permitindo assim guardar um histórico de logs.

Este sistema pode ser facilmente desativado através de um ficheiro de configuração.

A implementação desta ferramenta serviu também para uma melhor familiarização com a linguagem de programação em causa.

## 4.2.5 Importar ficheiros ZIP

Nesta plataforma foi desenvolvida uma solução para importar ficheiros zip. Para tal, os utilizadores fazem um pedido HTTP ao servidor do tipo POST. Esse pedido deve conter o ficheiro a ser importado e, como explicado na secção 4.2.3 Autenticação um

JWT para efeitos de autenticação. O servidor ao receber estes pedidos, cria uma diretoria para o projeto do utilizador onde os ficheiros zip serão extraídos.

## **4.2.6 Criar os Docker Containers e a Rede de um Projeto**

Para criar os Docker containers primeiro é necessário definir as Docker images necessárias. Para tal, o sistema trata de criar um Dockerfile para cada componente do projeto. Nesse Dockerfile estão especificadas todas as dependências e configurações da respetiva componente e as configurações do acesso por SSH.

Uma vez criados os Dockerfiles já é possível criar as Docker images e de seguida os Docker containers no entanto é necessário definir e configurar a rede na qual estes serão inicializados, para tal, será criado um ficheiro docker-compose.yaml. Neste ficheiro é possível especificar a ordem que cada container é inicializado, quais os portos que devem ser expostos e quais os argumentos passados a cada uma das componentes do projeto. Para além disto é possível definir uma rede de container, isto é, especificar quais o container que podem comunicar entre si.

## Capítulo 5

### *Benchmarking*

Tal como já foi indicado, no desenvolvimento de projetos com múltiplos processos utilizam-se diversas máquinas virtuais, ou mais do que um computador, no entanto, nenhuma destas soluções é ótima.

Utilizando o exemplo do Capítulo 1 onde os alunos necessitam desenvolver um projeto que obedece a uma arquitetura cliente-servidor, é possível identificar as razões destas soluções não serem as melhores.

1. Utilizar múltiplas máquinas virtuais, requer um computador com muitos recursos, o que muitas vezes não é possível.
2. Utilizar mais do que um computador, também não é a melhor opção porque implica que os alunos possuam mais que um computador, o que nem sempre é o caso, em alternativa existe a possibilidade de os alunos utilizarem os laboratórios da faculdade, mas normalmente não se mostram interessados.
3. Existe uma aplicação da Microsoft chamada Generator-docker [9] que apenas gera Dockerfiles e um ficheiro executável para criar uma Docker image e Docker containers. Apesar desta ser uma melhor solução por utilizar menos recurso de *hardware* e menos computadores o utilizador ainda necessita de configurar toda a rede de containers, isto é, o utilizador tem de garantir que todos os containers que irão executar as componentes do seu projeto comunicam.
4. Outra opção poderia ser utilizar um projeto chamado Alfresco Installer [10] que permite criar *templates* de docker-composes.yaml. O problema desta opção é que não permite criar Dockerfiles, ou seja, não permite criar novas Docker images, desta forma os tipos de projetos ficariam limitados.

## Capítulo 6

### Método e Planeamento

#### 6.1 Proposta de Calendário

Este capítulo tem como propósito definir um calendário para o desenvolvimento deste projeto. A Tabela 1 apresenta o calendário proposto.

Período de Desenvolvimento	Objetivo
25 de novembro a 1 de dezembro	Análise de requisitos
2 de dezembro a 8 de dezembro	Modelação da Base de Dados
9 de dezembro a 22 de dezembro	Criação de um sistema de autenticação
23 de dezembro a 5 de janeiro	Importar projetos do utilizador em formato JAR
6 de janeiro a 12 de janeiro	Importar projetos do utilizador em formato ZIP
13 de janeiro a 17 de janeiro	Escrever relatório intermédio
18 de janeiro a 2 de fevereiro	Importar projetos do utilizador de repositórios Git
3 de fevereiro a 16 de fevereiro	Gerar Dockerfiles com todas as dependências dos projetos
17 de fevereiro a 1 de março	Gerar Docker images
2 de março a 8 de março	Exportar projetos com todas as configurações
9 de março a 22 de março	Importar projetos gerados pela plataforma
23 de março a 12 de abril	A partir das Docker images criar Docker containers
13 de abril a 19 de abril	Consultar logs dos Docker containers
20 de abril a 26 de abril	Escrever relatório intercalar
27 de abril a 17 de maio	Acesso aos Docker containers por SSH via browser
18 de maio a 31 de maio	Testar com utilizadores
1 de junho a 7 de junho	Correção de bugs
8 de junho a 14 de junho	Aplicar sugestões de acordo com a opinião dos utilizadores
14 de junho a 19 de junho	Escrever relatório final

Tabela 1 - Calendário proposto



## 6.2 Cumprimento do Calendário

Uma vez definido o calendário é necessário criar um plano para o cumprir, para tal, foi elaborada uma lista de tarefas para atingir cada um dos objetivos. Esta lista de tarefas era atualizada sempre que uma das tarefas era concluída ou caso surgisse algum imprevisto como um *bug*.

Apesar deste planeamento, o calendário não foi concluído na totalidade. Devido ao número de tecnologias utilizados neste projeto, o tempo de aprendizagem foi superior ao previsto. Este atraso impossibilitou o desenvolvimento da camada de apresentação que terá de ser realizado futuramente.

## Capítulo 7

### Resultados

Tendo o projeto de redes descrito no Capítulo 1 Identificação do Problema como exemplo, é possível observar que ao utilizar o Boxit os problemas que este apresentava aos alunos na inicialização de vários clientes são evitados.

Como podemos observar pelo inquérito do Anexo 1 a maioria dos alunos testou os seus projetos com apenas dois clientes, as razões disto acontecer são os alunos não possuírem vários computadores e os computadores que possuem não tem recurso suficientes para suportar várias máquinas virtuais. Na Figura 19 podemos observar uma lista de trinta e um container a executar o projeto de redes.

NAME	CPU %	MEM USAGE / LIMIT
19ef5658-5ceb-4420-afc9-45ae9332208a_client26	0.15%	14.82MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client23	0.13%	14.91MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client5	0.15%	15.07MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client30	0.15%	15.34MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client4	0.17%	15.06MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client25	0.17%	14.83MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client16	0.15%	15.09MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client6	0.13%	15.07MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client12	0.18%	15.16MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client21	0.16%	15.03MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client13	0.14%	15.06MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client3	0.16%	14.81MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client28	0.16%	15.12MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client15	0.17%	14.64MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client2	0.14%	14.99MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client7	0.14%	15.05MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client29	0.15%	14.85MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client20	0.14%	14.8MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client9	0.14%	14.5MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client19	0.12%	14.97MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client27	0.15%	14.89MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client14	0.12%	14.89MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client1	0.15%	14.78MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client11	0.13%	14.95MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client18	0.16%	14.77MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client10	0.13%	15.22MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client24	0.12%	15.04MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client22	0.12%	15.07MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client8	0.12%	14.82MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_client17	0.14%	14.85MiB / 30.82GiB
19ef5658-5ceb-4420-afc9-45ae9332208a_server	0.13%	14.87MiB / 30.82GiB

Figura 19 - Utilização de hardware dos containers

Nesta lista, um dos containers corresponde ao servidor como podemos observar na Figura 20 e os restantes trinta aos clientes que comunicam com o servidor como é o caso da Figura 21 que apresenta um cliente a pedir ao servidor para criar um novo leilão.

```

madruga@ubuntu:~$ docker logs 16885e2d-44dd-4dfc-9ec8-15b0bb7b93ca_server && docker attach 16885e2d-44dd-4dfc-9ec8-15b0bb7b93ca_server
Insira um username
User1
Insira uma password
1234
Insira o plafond inicial
100
Licitador registrado
Insira um username
User2
Insira uma password
1234
Insira o plafond inicial
200
Licitador registrado
Insira um username
0 utilizador User1 foi autenticado.
0 utilizador User2 foi autenticado.
Foi criado um leilao com os seguintes dados.
1 Bola 19/08/2020 10:00:43 10.0 User1

```

Figura 20 - Servidor do projeto de redes a ser executado dentro de um container

```

madruga@ubuntu:~$ docker logs 16885e2d-44dd-4dfc-9ec8-15b0bb7b93ca_client1 && docker attach 16885e2d-44dd-4dfc-9ec8-15b0bb7b93ca_client1
Escolha uma opção
1 - Modo manual
2 - Modo automatico
3 - Quit
1
Insira um username
User1
Insira uma password
1234
Utilizador verificado
Escolha uma opção
1 - Licitar
2 - Criar Leilão
3 - Listar Leilões
4 - Plafond
5 - Quit
2
Descreve o objeto que quer leiloar
Bola
Insira a data de fecho
Insira o dia de fecho do leilão
19
Insira o mes de fecho do leilão
8
Insira o ano de fecho do leilão
2020
Insira a hora de fecho do leilão
10
Insira o minuto de fecho do leilão
00
Insira o valor inicial do leilão
10
0 seu leilão foi criado com sucesso com ID 1.

```

Figura 21 - Cliente do projeto de redes a ser executado dentro de um container

Como podemos observar na lista da Figura 19 cada um destes containers utiliza aproximadamente 15 MiB de memória e cerca 0.15% do CPU, o que são valores muito reduzidos que a maioria dos computadores pessoais suporta. Isto permite um aumento substancial no número de clientes que cada aluno poderia iniciar para testar o seu projeto ao utilizar o Boxit localmente.

O Boxit para além de poder ser executado localmente também pode ser inicializado num servidor. Tendo em conta o mesmo exemplo do projeto de redes, a Universidade poderia disponibilizar um servidor para executar o Boxit, desta forma os alunos autenticavam-se com as suas credenciais e poderiam enviar os seus projetos para o servidor que trataria de criar todos os containers necessários para inicializar o projeto com o número de clientes pretendidos. De seguida os alunos poderiam aceder via SSH aos containers pertencentes ao seu projeto.

## Capítulo 8

### Conclusão e Trabalho Futuro

Em suma, o desenvolvimento deste projeto foi muito mais desafiante do que qualquer outro projeto de outra disciplina do curso. Para além de necessitar de um planeamento mais detalhado, exigiu que uma grande adaptação dos conhecimentos adquiridos durante o curso para aprender novas tecnologias como o Docker e Node Js e construir uma solução adequada ao problema.

Apesar da plataforma já ser capaz de resolver o problema a que se propôs, esta ainda pode ser aperfeiçoada. Para tal, é necessário desenvolver toda a camada de apresentação tendo em conta documentação criada.

Para além disto a plataforma apenas suporta projetos Java, logo uma melhoria seria aumentar a quantidade de tipos de projetos que a plataforma suporta.

Uma vez que na maior parte do desenvolvimento de *software* os programadores utilizam ferramentas de controlo de versões como o Git em conjunto com plataforma que permitem a criação de repositórios remotos como o GitHub, outra melhoria uma boa ideia integrá-los com o Boxit, isto é, permitir que os utilizadores pudessem importar projetos do GitHub.

## Bibliografia

- [1] sumouli.choudhary, “geeksforgeeks,” [Online]. Available: <https://practice.geeksforgeeks.org/problems/explain-client-server-architecture>. [Acedido em 21 11 2019].
- [2] H. M. Tavares, “Projeto de Redes de Computadores”.
- [3] “StackOverflow,” [Online]. Available: <https://stackoverflow.com/questions/32373645/node-js-non-blocking-io-vs-java-thread-pool-pattern-using-nio-unclear-scheduli>. [Acedido em 25 Abril 2020].
- [4] C. Canvas, “Udemy,” [Online]. Available: <https://www.udemy.com/course/docker-essentials/learn/lecture/12339826#overview>. [Acedido em 24 11 2019].
- [5] “aqua,” [Online]. Available: <https://www.aquasec.com/wiki/display/containers/Docker+Containers+vs.+Virtual+Machines>. [Acedido em 21 11 2019].
- [6] “IETF,” [Online]. Available: <https://tools.ietf.org/html/rfc7519>. [Acedido em 2020 Abril 25].
- [7] “JSON Web Tokens,” [Online]. Available: <https://jwt.io/>. [Acedido em 2020 Abril 25].
- [8] S. Hsu, “Session vs Token Based Authentication,” [Online]. Available: <https://medium.com/@sherryhsu/session-vs-token-based-authentication-11a6c5ac45e4>. [Acedido em 25 Abril 2020].
- [9] “Node.js,” [Online]. Available: <https://pt.wikipedia.org/wiki/Node.js>. [Acedido em 21 11 2019].
- [10] “What is a Container?,” [Online]. Available: <https://www.docker.com/resources/what-container>. [Acedido em 21 11 2019].
- [11] “What is Docker?,” [Online]. Available: <https://opensource.com/resources/what-docker>. [Acedido em 21 11 2019].
- [12] M. Hamedani, “React vs. Angular: The Complete Comparison,” 5 Novembro 2018. [Online]. Available: <https://programmingwithmosh.com/react/react-vs-angular/>. [Acedido em 21 11 2019].

- [13] G. Santos, “O que é, porque usar e primeiros passos,” [Online]. Available: <https://medium.com/thdesenvolvedores/node-js-o-que-%C3%A9-por-que-usar-e-primeiros-passos-1118f771b889>. [Acedido em 21 11 2019].
- [14] TechMagic, “React vs Angular vs Vue.js — What to choose in 2019?,” [Online]. Available: <https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d>. [Acedido em 21 11 2019].
- [15] P. Engineering, “Node.js at PayPal,” [Online]. Available: <https://medium.com/paypal-engineering/node-js-at-paypal-4e2d1d08ce4f>. [Acedido em 21 11 2019].
- [16] “Planning to hire Node JS developers? Here are 3 cases to help you decide,” 30 Janeiro 2017. [Online]. Available: <https://dedicatedteams.io/planning-to-hire-node-js-developers/>. [Acedido em 21 11 2019].
- [17] C. Academy, “Top 32 Sites Built With ReactJS,” 21 6 2016. [Online]. Available: <https://medium.com/@coderacademy/32-sites-built-with-reactjs-172e3a4bed81>. [Acedido em 23 11 2019].
- [18] S. Holla, “Tech Diagonal,” [Online]. Available: [https://www.techdiagonal.com/reactjs\\_courses/beginner/reactjs-component-state/](https://www.techdiagonal.com/reactjs_courses/beginner/reactjs-component-state/). [Acedido em 24 11 2019].
- [19] T. Eneh, “Most popular CI/CD pipelines and tools,” [Online]. Available: <https://medium.com/faun/most-popular-ci-cd-pipelines-and-tools-ccfdce429867>. [Acedido em 28 1 2020].
- [20] “Node Organization-Distributed Model,” [Online]. Available: [https://3.bp.blogspot.com/-KEFZ9hdIg68/VJ0vLxPgT3I/AAAAAAAAAIs/YILov-6Sg1E/s1600/Node\\_Organization-Distributed\\_Model%40allicient.png](https://3.bp.blogspot.com/-KEFZ9hdIg68/VJ0vLxPgT3I/AAAAAAAAAIs/YILov-6Sg1E/s1600/Node_Organization-Distributed_Model%40allicient.png). [Acedido em 22 Fevereiro 2020].
- [21] [Online]. Available: <https://github.com/microsoft/generator-docker>. [Acedido em 15 Julho 2020].
- [22] [Online]. Available: <https://github.com/Alfresco/alfresco-docker-installer>. [Acedido em 15 Julho 2020].

## Anexo 1 – Inquérito realizado a 38 alunos

Este inquérito tem como objetivo compreender qual o interesse dos alunos em deslocar-se aos laboratórios da universidade para realizarem os seu projetos, tal como as razões deste pelas quais estes não utilizam os laboratórios. Para além disto com este inquérito pretende-se perceber como é que os alunos testaram os seu projetos de redes com múltiplos clientes e com quantos clientes é que testaram.

Na pergunta da Figura 22 perguntou-se aos alunos quais as razões que os desmotiva a deslocarem-se aos laboratórios onde a maior parte respondeu que é inconveniente.

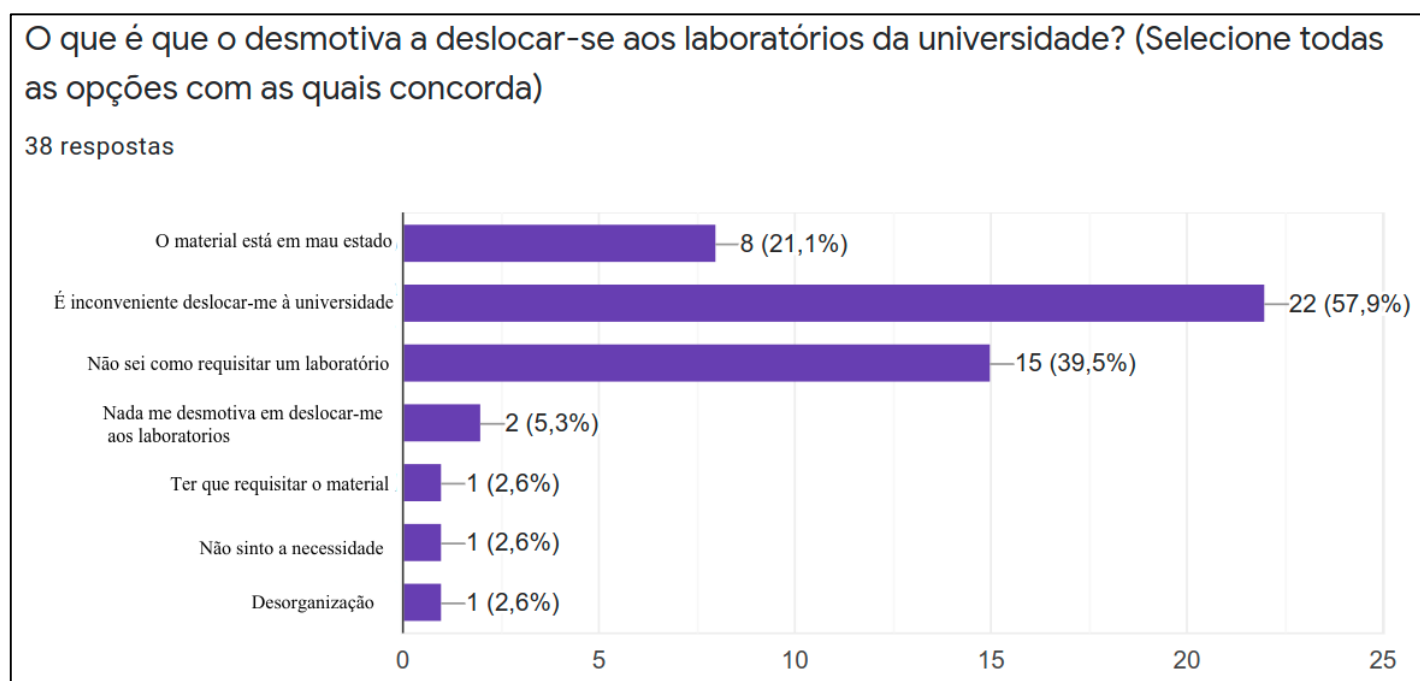


Figura 22 Razões para os alunos não se deslocarem aos laboratórios

Na questão seguinte apresentada na , perguntou-se ao alunos com que regularidade se deslocam à faculdade para utilizar os laboratórios. Neste questão a maioria respondeu que nunca. Tendo em conta a resposta desta questão e da anterior é possível compreender que os alunos preferem não ter de se deslocar de propósito aos laboratórios da faculdade para trabalharem nos seus projetos.



Com que regularidade se desloca à universidade para utilizar os laboratórios no desenvolvimento dos seus projetos? (Selecione apenas uma opção)

38 respostas

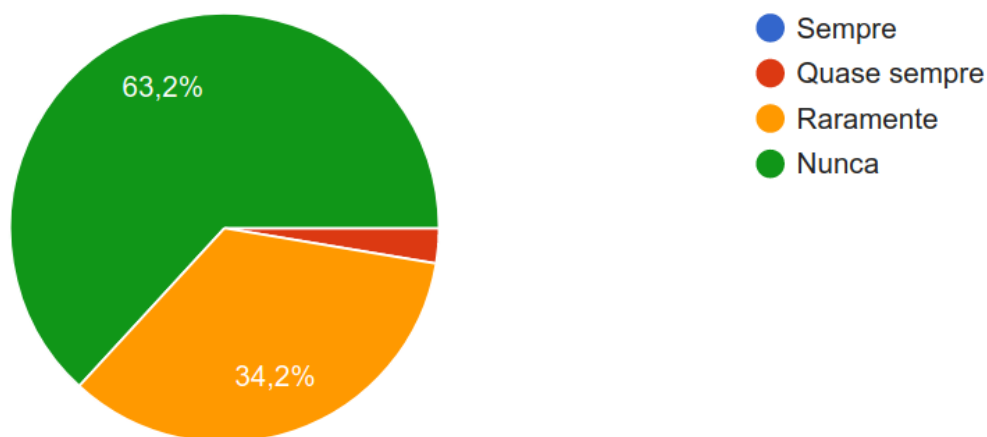


Figura 24 - Regularidade de utilização dos laboratórios

De seguida perguntou-se aos alunos como é que tinham testado os seus projetos com mais do que um cliente, como apresenta a Figura 23. Nesta pergunta houve uma maior variação nas respostas, mas apesar disso, é possível observar que nenhum dos alunos se deslocou aos laboratórios da faculdade e que a maior parte utilizou máquinas virtuais ou mais do que um computador

Como é que testou o seu projeto com mais do que um cliente? (Selecione todas as opções com as quais concorda)

35 respostas

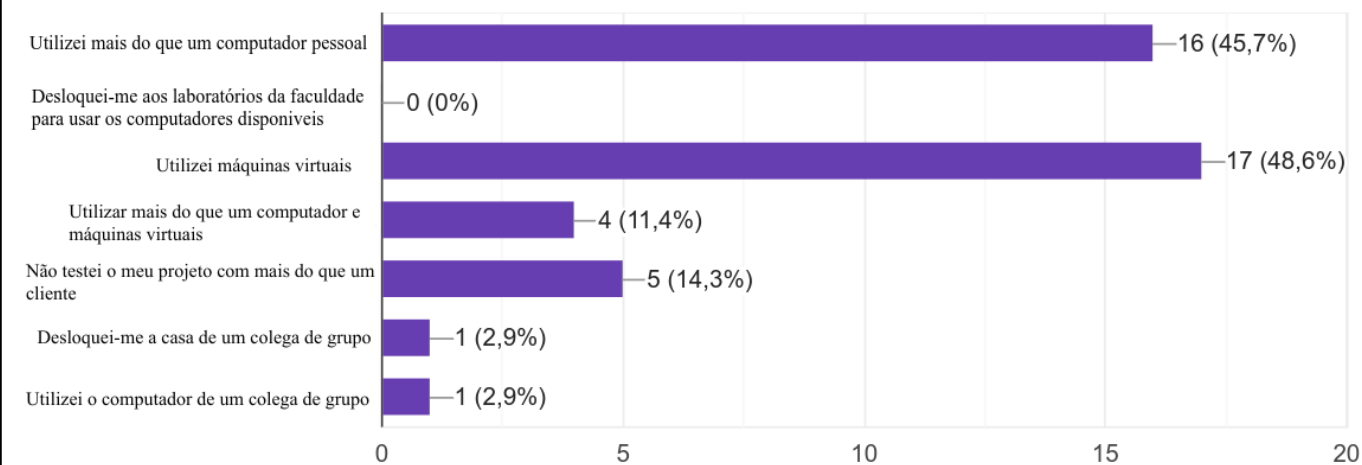


Figura 23 - Solução para testar o projeto de redes com múltiplos clientes

Depois perguntou-se aos alunos qual o máximo de clientes que utilizaram para testar o projeto como apresenta a Figura 25. Como podemos observar a grande maioria testou com 2 clientes.

Qual foi o máximo de clientes que utilizou para testar o seu projeto? (Selecione apenas uma opção)

33 respostas

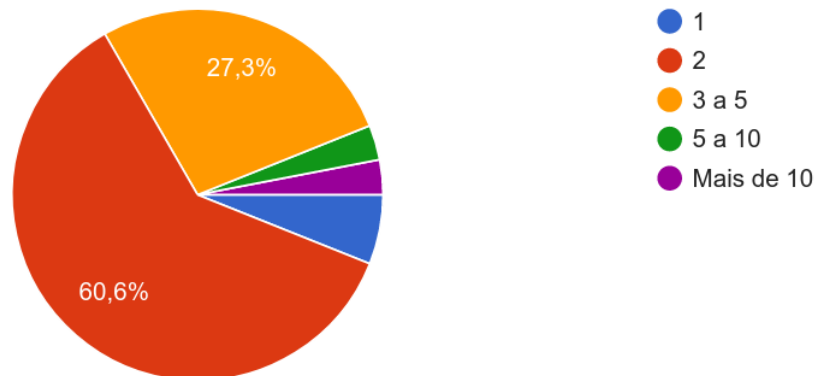


Figura 25 - Número máximo de clientes utilizados para testar o projeto de redes

Na pergunta seguinte pretendeu-se descobrir a razão pela qual os alunos não testaram os seus projeto com mais clientes. Apesar de metade do alunos responder que não queria testar com mais clientes uma grande parte sentiu-se limitado por não ter muitos computadores ou por o computador que tem não suportar muitas máquinas virtuais.

Porque é que não testou o seu projeto com mais clientes? (Selecione apenas uma opção)

32 respostas

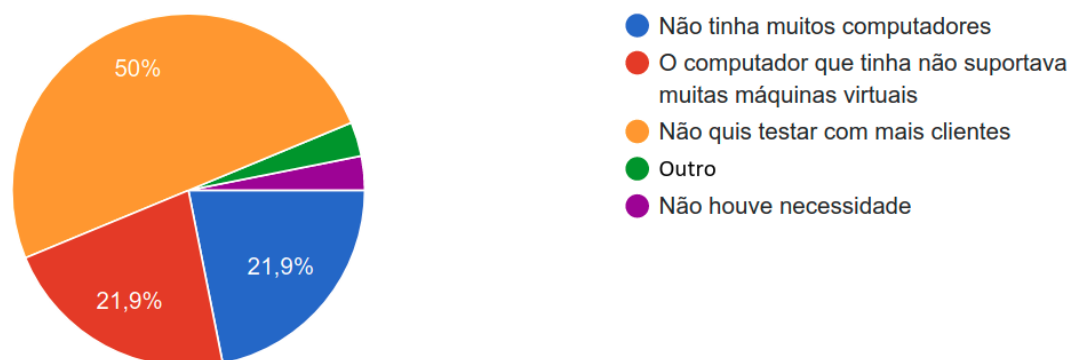


Figura 26 - Razões para não ter testado o projeto com mais clientes

Por fim perguntou-se aos alunos se gostariam de ter testado o seu projeto com mais clientes ao que a maioria respondeu que sim como apresenta a . Desta pergunta é possível concluir que existe uma procura para uma solução deste problema.

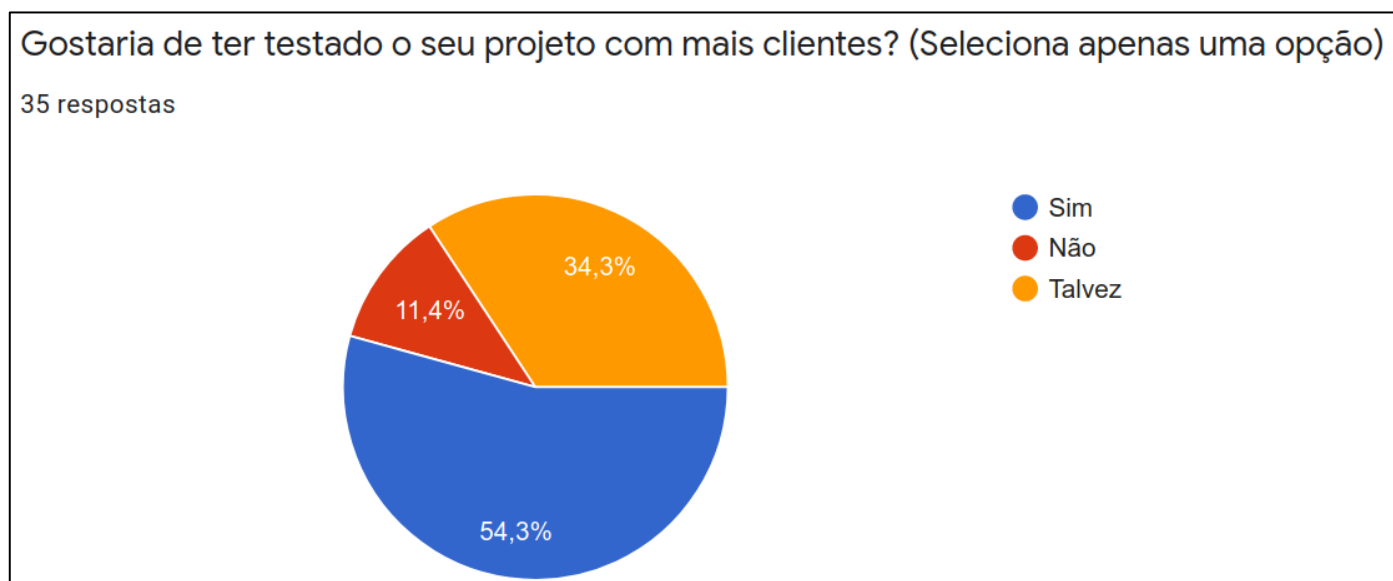


Figura 27 - Sondagem para saber os alunos queria ter testado os projetos com mais clientes

## Anexo 2 - Manual de instruções do Boxit

### Requisitos para iniciar o Boxit

- Node Js
- NPM
- Docker
- SGBDR (os suportados são MySQL, MariaDB, PostgreSQL e MsSQL)

### Iniciar o Boxit

1. Acrescentar ao projeto um ficheiro chamado .env com a seguinte estrutura.

```
# App
DEV_APP_PORT=5000
APP_NAME=Boxit_Backend
NODE_ENV=development
SINGLE_USER=false

# Logger
LOG_INFO=true
LOG_ERROR=true
LOG_WARN=true

#SSH
SSH_PASSWORD=Boxit

# Database
DB_PORT=3306
DB_NAME=Boxit_DB
DB_PASS=password123
DB_USER=boxit_user
DB_HOST=127.0.0.1
DB_DIALECT=mysql #options: mysql, mariadb, postgres, mssql
DB_LOGGING=false
DB_MIGRATION_STORAGE_TYPE=none #options: none, json, sequelize
DB_MIGRATION=migrations.json
DB_SEEDER_STORAGE_TYPE=none #options: none, json, sequelize
DB_SEEDER=seeders.json

# Authentication
JWT_SECRET=this is my secret phrase
JWT_EXPIRES_IN=1d
JWT_ALGORITHM=HS512
SALT_ROUND=10
```

```
REFRESH_TOKEN_SECRET=this is another secret phrase  
REFRESH_TOKEN_EXPIRES_IN=2d  
EMAIL_TOKEN_SECRET=my secret email phrase  
EMAIL_TOKEN_EXPIRES_IN=1h
```

```
# Email  
EMAIL_SERVICE=Gmail  
EMAIL_USER=example.email@gmail.com  
EMAIL_PASSWORD=1234  
EMAIL_NAME=Boxit non-reply
```

2. Instalar todas as dependências do projeto, para tal basta executar o seguinte comando na raiz do mesmo:

```
npm install
```

3. Criar a base de dados e de seguida iniciar o Boxit, para tal existe um *script* que pode ser executado através do seguinte comando:

```
npm run start
```

Uma vez que todos estes passos sejam concluídos é possível verificar a documentação dos pedidos suportados pelos Boxit através de um browser em:

[http://localhost:<DEV\\_APP\\_PORT>/api-docs](http://localhost:<DEV_APP_PORT>/api-docs)

onde <DEV\_APP\_PORT> corresponde à variável definida no ficheiro .env.

# **Glossário**

UI – User Interface

JAR – Java Archive

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

SSH – Secure Socker Shell

NPM – Node Package Manager

JWT – JSON Web Token

IETF - Internet Engineering Task Force

HMAC - Hash-based Message Authentication Code