Universidade do Minho
Mestrado Integrado em Engenharia Informática
Departamento de Informática

System Deployment & Benchmarking

---

# Ghost

---

*Grupo 2*
Carlos Pinto Pedrosa - A77320
Eduardo Gil Ribeiro da Rocha - A77048
Manuel Gouveia Carneiro de Sousa - A78869
9 de Novembro de 2018

# Conteúdo



# 1 Introdução

Nos dias atuais, o mais simples serviço envolve uma série de componentes tecnológicos, equipas de manutenção, gestão, entre outros, tornando-se, por vezes, extremamente difícil gerir toda a infraestrutura que suporta o sistema.

No âmbito da Unidade Curricular de *System Deployment And Benchmarking*, foi-nos proposto, numa primeira fase, o estudo de um sistema relativamente simples e, numa fase posterior, o seu *deployment*.

Assim, nesta fase, apenas iremos detalhar o sistema escolhido.

# 2 Ghost

Ghost é uma plataforma *open source*, criada em 2013, com o intuito de construir e modernizar publicações *online*. A grande missão do *Ghost* é a de criar ferramentas abertas para todo o tipo de jornalistas e escritores, tendo assim um forte impacto no futuro dos *media*.

Assim, neste capítulo iremos apresentar a arquitetura e os componentes do sistema, os padrões de distribuição, as formas de comunicação entre elementos do sistema, os pontos de configuração e as operações críticas.

## 2.1 Arquitetura e Componentes da Aplicação

Do ponto de vista arquitetural, o *Ghost* possui 3 grandes ideais, os quais sustentam todo o Sistema:

- **Robusta *API* em *JSON* no seu núcleo;**
- **Aplicação sobre um cliente (ou web) para o Administrador;**
- **Um poderosíssimo e acessível *front-end*.**

Todas estes ideais funcionam de forma concreta e concisa, fornecendo ao utilizador uma panóplia de opções de customização.

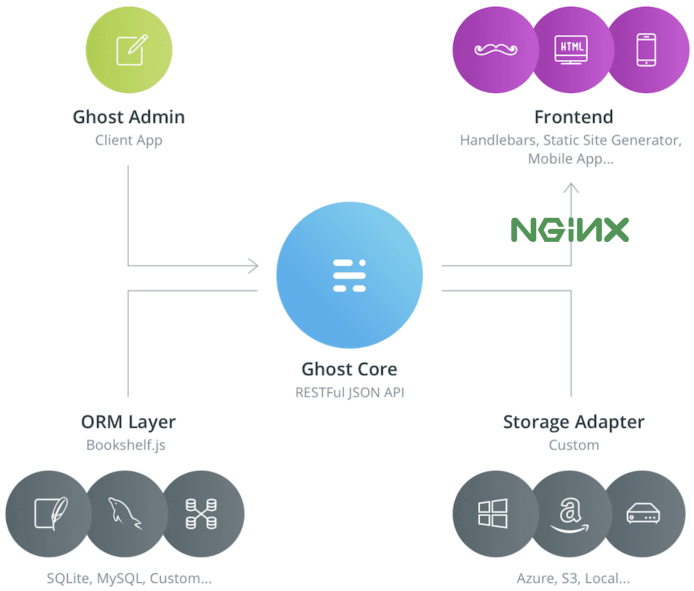


Figura 1: Arquitetura do Sistema Ghost

Assim, quando o administrador pretende, por exemplo, submeter uma nova publicação que já escreveu usando o front-end disponibilizado, esta é

transformada em *JSON* e enviada para o servidor web *NGINX*, que é depois interpretado pelo *Ghost Core*. No seguimento e consoante as configurações do *Ghost*, esta é depois guardada na Base de Dados ou Serviço de Armazenamento Externo.

Falaremos agora dos principais componentes presentes no *Ghost*.

### 2.1.1 Base de Dados & Sistema de Ficheiros

Em termos de Bases de Dados, o *Ghost* vem equipado com a biblioteca *Bookshelf.js ORM* que permite a utilização de inúmeros motores de bases de dados. No entanto, é sugerido *MySQL* no ambiente de produção.

Adicionalmente, apesar do *Ghost* utilizar o sistema de ficheiros local por defeito, este também permite a utilização de serviços que possibilitam colocar o sistema de ficheiros externamente, como na *Cloud*.

### 2.1.2 Ghost Core

No núcleo desta aplicação, encontra-se uma *API* denominada de *RESTful JSON*, a qual é usada para que componentes do Ghost comuniquem (através de pedidos remotos). Usando uma arquitetura e *API REST*, usam então *JSON* como formato para troca de mensagens. Desta forma, é possível fazer o *marshalling* de mensagens entre os diversos componentes da aplicação.

### 2.1.3 Admin Client

A grande vantagem do *Ghost* é o facto de fornecer uma interface administrativa com ferramentas muito poderosas, permitindo assim a qualquer editor/escritor um fácil manuseamento do seu conteúdo e ao mesmo tempo da plataforma. De facto, o que normalmente acontece é que editores com ferramentas eficazes geralmente não são eficientes, e editores eficientes normalmente não são *user friendly*.

Com o *Ghost* podemos ter o melhor desses dois mundos, isto é, um editor com ferramentas poderosas e ao mesmo tempo altamente eficientes e fáceis de usar. Tudo isto é possível devido ao facto da aplicação do administrador ser completamente independente do núcleo do *Ghost*.

### 2.1.4 Web Server NGINX

Devido ao facto da plataforma funcionar em ambiente web, é necessário, pelo menos, um servidor capaz de lidar com os pedidos HTTP. Efetivamente, nesta plataforma em particular, é usado o servidor *NGINX*.

Adicionalmente, este servidor efetua o redirecionamento dos pedidos que recebera para o núcleo do sistema, onde são interpretados e construída uma resposta.

Posto isto, esta resposta é devolvida ao *NGINX*, que por sua vez a encaminha para o Cliente.

## 2.2 Pontos de Configuração

Neste capítulo iremos detalhar os pontos de configuração do sistema, isto é, todos os ficheiros que são necessário alterar para modificar certos aspetos do sistema, como por exemplo, o motor de base de dados e a sua localização na rede, entre outros.

De facto, o *Ghost* é fisicamente estruturado em duas diretorias principais:

1. core/, onde se localiza o núcleo do sistema, como código-fonte e configurações iniciais;

2. content/, onde se localizam os ficheiros que podem ser adicionados ou modificados pelos utilizadores, como por exemplo, imagens e temas.

```
carlos@Carlos:/var/www/ghost$ ls
config.production.json  content  current  system  versions
carlos@Carlos:/var/www/ghost$ _
```

Figura 2: Estrutura da Raiz do Sistema

```
carlos@Carlos:/var/www/ghost/versions/2.4.0/core$ ls
built  index.js  server
carlos@Carlos:/var/www/ghost/versions/2.4.0/core$
```

Figura 3: Estrutura da Diretoria core


Figura 4: Estrutura da Diretoria content

No entanto, se for necessário alterar algum dos componentes principais do sistema, existe um ficheiro de configuração onde é possível fazê-lo, *config.production.json*

### 2.2.1 Ficheiro de Configuração *config.production.json*


Figura 5: Exemplo de um Ficheiro de Configuração

É neste ficheiro, localizado na diretoria raiz do Ghost, onde se encontram configurados os principais componentes do sistema. Tudo desde o *url* do *blog* até às definições de *spam* podem ser definidos neste ficheiro.

| Name | Required? | Description |
|---|---|---|
| `url` | In production | Set the public URL for your blog |
| `database` | In production | Type of databased used (default: sqlite3) |
| `mail` | In production | Add a mail service |
| `admin` | Optional | Set the protocol and hostname for your admin panel |
| `server` | Optional | Host and port, or socket for Ghost to listen on |
| `privacy` | Optional | Disable features set in privacy.md |
| `paths` | Optional | Customise internal paths |
| `referrerPolicy` | Optional | Control the content attribute of the meta referrer tag |
| `useMinFiles` | Optionl | Generate assets url with .min notation |
| `storage` | Optional | Set a custom storage adapter |
| `scheduling` | Optional | Set a custom scheduling adapter |
| `logging` | Optional | Configure logging for Ghost |
| `spam` | Optional | Configure spam settings |
| `caching` | Optional | Configure caching settings |
| `compress` | Optional | Disable compression of server responses |
| `imageOptimization` | Optional | Configure image manipulation and processing |

Figura 6: Opções Possíveis no Ficheiro de Configuração

Para além das configurações possíveis de efetuar no ficheiro de configuração, é também possível definir encaminhamento dinâmico (*content/settings/routes.yaml*), redirecionamentos (*content/data/redirects.json*), *storage adaptors* e, por último, *custom schedules*.

### 2.2.2 Base de Dados

O *Ghost* permite uma grande variedade de Bases de Dados, dando preferência, no entanto, a *MySQL* ou *SQLite3*. A configuração da Base de Dados é feita no ficheiro de configuração do subcapítulo anterior.

```
"database": {
    "client": "mysql",
    "connection": {
        "host": "127.0.0.1",
        "port": 3306,
        "user": "your_database_user",
        "password": "your_database_password",
        "database": "your_database_name"
    }
}
```

Figura 7: Configuração Para Base De Dados MySQL

```
"database": {
    "client": "sqlite3",
    "connection": {
        "filename": "content/data/ghost-test.db"
    },
    "useNullAsDefault": true,
    "debug": false
}
```

Figura 8: Configuração Para Base De Dados SQLite3

Uma outra opção de configuração para a base de dados é o número de conexões permitidas à Base de Dados.

```
"database": {
    "client": ...,
    "connection": { ... },
    "pool": {
        "min": 2,
        "max": 20
    }
}
```

Figura 9: Configuração Que Limita Número de Conexões à BD

Por último, o *Ghost* também permite configurar *SSL* para a Base de Dados através de um atributo no ficheiro *JSON*, o que é bastante útil para

ligar o *Ghost Core* à Base de Dados se estes não se encontrarem no mesmo servidor.

```
"database": {
    "client": "mysql",
    "connection": {
        "host": "your_cloud_database",
        "port": 3306,
        "user": "your_database_user",
        "password": "your_database_password",
        "database": "your_database_name",
        "ssl": true
    }
}
```

Figura 10: Configuração SSL da Base de Dados

### 2.2.3 Serviço de Mail

O componente mais importante depois de instalar o *Ghost* é configurar um serviço de mail no nosso sistema. Nesse sentido, tudo fica mais fácil com o ficheiro de configuração, pois com apenas algumas linhas extra conseguimos efetivar a sua configuração.

```
"mail": {
    "transport": "SMTP",
    "options": {
        "service": "Mailgun",
        "auth": {
            "user": "postmaster@example.mailgun.org",
            "pass": "1234567890"
        }
    }
}
```

Figura 11: Configuração de Mail

## 2.3 Operações Críticas

Neste capítulo iremos identificar gargalos do sistema, isto é, operações que efetuadas em grande quantidade podem afetar o desempenho do *Ghost*, causando lentidão ou até mesmo, falhas deste.

Assim, quando observamos atentamente o *Ghost*, verificamos que o seu principal objetivo é o de fazer chegar conteúdo ao cliente. Por sua vez, este conteúdo é criado pelo escritor/administrador do sistema. É relativamente fácil de concluir que existirá um número significativamente maior de operações de leitura do que escrita no sistema. Podemos afirmar então, que quando existir um elevado número de clientes do *blog*, este poderá passar por dificuldades de desempenho. É relevante dizer que o próprio administrador do *Ghost* para além de operações de escrita é também um cliente do seu *website*, pois a criação de conteúdo é realizada diretamente no site, efetuando também operações de leitura.

Um cliente quando pretende obter conteúdo de um site que corra *Ghost*, os seguintes passos acontecem:

- O Cliente, através do navegador, liga-se ao servidor *web* e pede o conteúdo do *website*;
- O *NGINX* redireciona os pedidos para o *Ghost Core*.
- O *Ghost Core*, faz o parsing do ficheiro *JSON* e retira as informações necessárias;
- Com estas informações, o *Ghost Core* recorre à base de dados para adquirir os dados necessários;
- *Ghost Core*, encapsula as informações e envia-as ao servidor *web*;
- O *NGINX* envia a informação para o Cliente;

Como se observa pelos passos acima descritos, só uma simples leitura envolve o *NGINX*, o *Ghost Core* e a Base de Dados, que são componentes essenciais do sistema, e por essa mesma razão não queremos que sofram sobrecargas.

No próximo capítulo, descreveremos as opções tomadas para evitar que tais eventos aconteçam.

## 2.4 Padrões de Distribuição e Formas de Comunicação

Com o intuito de solucionar os desafios enunciados no subcapítulo anterior, resolvemos criar o seguinte modelo, de forma a demonstrar uma possível distribuição para o sistema.

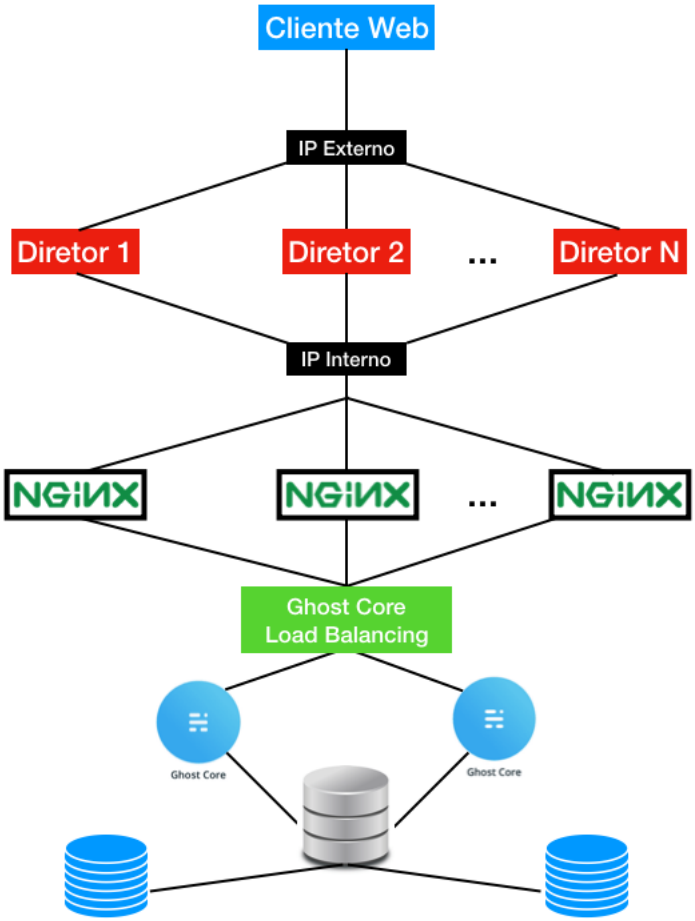


Figura 12: Possível Distribuição do Sistema

De facto, quando o Cliente tenta estabelecer uma conexão ao servidor, está, na verdade, a ligar-se a um *IP* externo disponibilizado pelo serviço ***KeepAlive*** [1]. Assim, depois de verificado qual o "Diretor" ativo, este escolhe

[1] O KeepAlive é serviço que permite gerir processos (diretores) que por sua vez gerem um conjunto de servidores *web*, neste caso.

o servidor *NGINX* mais adequado para lidar com o pedido e a informação é redirecionada. Por sua vez, o servidor *web* irá necessitar de encaminhar o tráfego para o *Ghost Core*. No entanto, existirão algumas instâncias independentes deste componente, pelo que, é necessário um serviço capaz de balancear a carga destes. Posto isto, o servidor aplicacional escolhido terá de interrogar a Base de Dados. Por fim, estará a executar um serviço capaz de monitorizar o estado da Base de Dados, e em caso de alguma falha, automaticamente redirecionar as conexões para uma Base de Dados auxiliar que seja uma réplica da principal.

# 3 Conclusão

De acordo com os principais objetivos desta primeira fase do trabalho prático, concluímos que estes foram atingidos com sucesso. De facto, fomos capazes de identificar os diversos componentes do sistema, e como estes se interligam.

Em suma, esta primeira fase deu-nos bases para identificar os diversos componentes de sistemas distintos e relativamente complexos. Numa próxima fase, passaremos para o *deployment* do *Ghost* tendo como base o modelo anteriormente apresentado.

# Referências


[1] https://docs.ghost.org/concepts/architecture/

[2] https://docs.ghost.org/concepts/core/

[3] https://docs.ghost.org/concepts/admin/

[4] https://docs.ghost.org/concepts/config/