

RELATÓRIO TÉCNICO

Implementação e Configuração de um Servidor Web
Abordagem Prática com PostgreSQL, NGINX, Apache e Aplicações de
Código Aberto

Fernanda Martins da
Silva^a

Gabriel Maia Miguel^a

Samuel Oliveira Lopes^a

^a Instituto Federal de São Paulo, Campus São João da Boa Vista

Abstract

Este relatório apresenta uma abordagem prática para a implementação e configuração de servidores web, utilizando SGBD, NGINX, Apache e aplicações de código aberto.

Keywords: servidor web, NGINX, Apache, SGBD, PostgreSQL, código aberto, implementação, configuração

Author Note

We have no conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Gabriel Maia Miguel,

1. Objetivos

1.1 *Objetivo Geral:*

Demonstrar a implementação e configuração passo a passo de um servidor web. Para isso, seguiremos os seguintes requisitos solicitados pelo professor responsável pela disciplina de Laboratório de Redes de Computadores. São eles:

1. Implementação de um SGBD (Sistema de Gerenciamento de Banco de Dados).
2. Hospedar uma aplicação Java Web que interaja com o SGBD.
3. Configurar um servidor HTTP com suporte a PHP e integração com o SGBD.
4. O Projeto deve estar acessível a partir da rede do Laboratório (LAN).
5. Utilizar e disponibilizar software de código aberto de uma das seguintes categorias:
 - ERP (Enterprise Resource Planning)
 - CMS (Content Management System)
 - CRM (Customer Relationship Management)
 - LMS (Learning Management System)
6. Documentar todo o processo de implementação e configuração do servidor web, incluindo detalhes técnicos, desafios enfrentados e soluções adotadas.
7. (Opcionalmente) Apresentar o projeto finalizado para avaliação, destacando as funcionalidades implementadas e a integração entre os componentes do servidor web por meio de vídeos demonstrativos.

1.2 *Objetivos Específicos*

- Compreender de forma prática os conceitos de redes de computadores aplicados na implementação de servidores web.
- Desenvolver habilidades técnicas na configuração de servidores HTTP, SGBD e aplicações web.
- Integrar diferentes tecnologias de código aberto para criar uma solução funcional e eficiente.
- Documentar o processo de implementação, facilitando a replicação e o aprendizado por parte de outros estudantes.
- Melhorar as habilidades de apresentação e comunicação técnica.

2. Considerações Gerais

Esta seção apresenta os principais conceitos de redes utilizados no desenvolvimento do projeto de forma breve. Nas seções seguintes, serão detalhadas as dificuldades enfrentadas e justificadas as escolhas das ferramentas adotadas pelo grupo. De modo a fornecer uma visão abrangente dos fundamentos teóricos aplicados, dos desafios superados e dos motivos que orientaram as decisões técnicas ao longo da implementação.

2.1 Trabalhos Correlatos

A implementação de servidores web é uma prática fundamental na infraestrutura de TI. Tradicionalmente, pilhas de soluções como LAMP (Linux, Apache, MySQL, PHP) ou LEMP (Linux, Nginx, MySQL, PHP) têm sido amplamente utilizadas para hospedar aplicações dinâmicas. Essas pilhas fornecem uma base sólida, integrando sistema operacional, servidor web, sistema de gerenciamento de banco de dados e linguagem de script.

No entanto, com a evolução da computação em nuvem e a necessidade de escalabilidade e portabilidade, abordagens baseadas em contêineres, como o Docker, ganharam destaque. Diferente das instalações tradicionais diretamente no sistema operacional (“bare metal”), o uso de contêineres permite o isolamento de dependências, facilitando a replicação do ambiente de desenvolvimento para a produção e evitando conflitos entre bibliotecas de diferentes aplicações. Este projeto adota essa abordagem moderna, utilizando Docker Compose para orquestrar múltiplos serviços interconectados.

2.2 Sistemas Gerenciadores de Banco de Dados

Os Sistemas de Gerenciamento de Banco de Dados (SGBD) são componentes críticos para a persistência e integridade dos dados em aplicações web. Neste projeto, optou-se pela utilização de dois SGBDs relacionais de código aberto amplamente adotados:

- **MariaDB:** Um fork do MySQL, conhecido por sua robustez e compatibilidade. É utilizado aqui como o banco de dados para o Moodle, garantindo suporte nativo e alto desempenho para as operações do LMS.
- **PostgreSQL:** Um SGBD objeto-relacional avançado, reconhecido por sua conformidade com padrões SQL e extensibilidade. É utilizado pela aplicação Spring Petclinic, demonstrando a capacidade do servidor de gerenciar múltiplos tipos de bancos de dados simultaneamente.

A escolha de utilizar contêineres separados para cada banco de dados reforça o princípio de desacoplamento, permitindo que cada aplicação tenha seu próprio ciclo de vida e configuração de armazenamento.

2.3 Sistemas Distribuídos vs. Monolíticos

A arquitetura de software tem transitado de modelos monolíticos para sistemas distribuídos e microsserviços.

- **Arquitetura Monolítica:** Tradicionalmente, aplicações web eram construídas como uma única unidade indivisível. Embora mais simples de desenvolver inicialmente, tornam-se difíceis de manter e escalar à medida que crescem, pois qualquer alteração requer a recompilação e implantação de todo o sistema.
- **Arquitetura Distribuída (Microsserviços):** A abordagem adotada neste projeto reflete conceitos de sistemas distribuídos. Ao separar o servidor web (NGINX), as aplicações (Moodle, Petclinic) e os bancos de dados em contêineres distintos, cria-se um ambiente modular. Isso permite, por exemplo, atualizar a versão do Java do Petclinic

sem afetar o funcionamento do Moodle, ou escalar horizontalmente o servidor web sem replicar desnecessariamente os bancos de dados. O Docker Compose atua como o orquestrador local, definindo a topologia dessa rede de serviços.

3. Implementação

A implementação do projeto foi realizada utilizando a tecnologia de contêineres Docker, orquestrada pelo Docker Compose. Essa abordagem permite isolar os serviços, facilitar a configuração e garantir a portabilidade do ambiente. A estrutura do projeto foi organizada em diretórios contendo as configurações específicas de cada serviço.

A seguir, detalhamos os passos e as configurações adotadas para cada componente da solução.

3.1 Arquitetura da Solução

A arquitetura é composta por cinco contêineres principais, divididos em redes internas para segurança e organização:

Table 1

Componentes da Arquitetura do Servidor

Serviço	Tecnologia	Função
web	NGINX	Proxy Reverso e Servidor Web
moodle	Moodle (Apache/PHP)	Sistema de Gestão de Aprendizagem (LMS)
moodle-db	MariaDB	Banco de Dados do Moodle
petclinic	Spring Boot (Java)	Aplicação de Exemplo (Java Web)
petclinic-db	PostgreSQL	Banco de Dados do Petclinic

3.2 Configuração do Proxy Reverso (NGINX)

O NGINX atua como a porta de entrada (Gateway) para o servidor. Ele escuta na porta 8080 (mapeada para a porta 80 do contêiner) e redireciona as requisições com base na URL acessada.

- **Rota /**: Serve arquivos estáticos HTML localizados em ./html.
- **Rota /moodle/**: Redireciona para o contêiner do Moodle.
- **Rota /petclinic/**: Redireciona para o contêiner da aplicação Spring Petclinic.

A configuração foi definida no arquivo `nginx/default.conf`, utilizando a diretiva `proxy_pass` para encaminhar o tráfego para os nomes de host dos serviços definidos no Docker Compose (`http://moodle` e `http://petclinic:8080`).

3.3 Implementação do LMS (Moodle)

Para atender ao requisito de software de código aberto (LMS), utilizamos o Moodle.

- **Banco de Dados**: Foi configurado um contêiner `mariadb:10.6` (`moodle-db`), com persistência de dados garantida pelo volume `moodle_db_data`.

- **Aplicação:** O contêiner `moodle` utiliza a imagem `brandkern/moodle:4.5-latest`.
- **Configuração:** As variáveis de ambiente (banco de dados, usuário, senha, URL base) foram externalizadas para os arquivos `config/moodle-db.env` e `config/moodle.env`.
- **Ajuste no Apache:** Foi necessário mapear um arquivo de configuração customizado (`apache/000-default.conf`) para definir o Alias `/moodle /var/www/html`, garantindo que a aplicação responda corretamente no subdiretório esperado pelo proxy reverso.

3.4 Implementação da Aplicação Java (Spring Petclinic)

A aplicação Java escolhida foi o **Spring Petclinic**, um exemplo clássico de aplicação corporativa.

- **Banco de Dados:** Utilizamos o PostgreSQL (`petclinic-db`) versão 15, com dados persistidos no volume `petclinic_db_data`.
- **Build da Aplicação:** Diferente das outras imagens prontas, a aplicação Java é construída localmente utilizando um `Dockerfile` de múltiplos estágios (Multi-stage build):
 1. **Estágio de Build:** Utiliza uma imagem JDK (Eclipse Temurin 25) para compilar o projeto com Maven (`./mvnw package`).
 2. **Estágio de Execução:** Utiliza uma imagem JRE mais leve para rodar o `.jar` gerado.
- **Context Path:** A aplicação foi configurada para rodar no contexto `/petclinic` através da propriedade `server.servlet.context-path` e variáveis de ambiente no `Dockerfile` e `config/petclinic.env`.

3.5 Orquestração com Docker Compose

O arquivo `compose.yml` na raiz do projeto define todos os serviços, redes e volumes.

- Redes:
 - ▶ `backend`: Conecta o NGINX, Petclinic e Petclinic-DB.
 - ▶ `moodle_net`: Conecta o NGINX, Moodle e Moodle-DB.

Essa segmentação impede, por exemplo, que o Moodle acesse diretamente o banco de dados do Petclinic e vice-versa.

- Volumes: Volumes nomeados (`moodle_data`, `moodle_db_data`, `petclinic_db_data`) garantem que os dados não sejam perdidos se os contêineres forem recriados.

Para iniciar todo o ambiente, utiliza-se o comando:

```
docker compose up -d --build
```

4. Conclusões e recomendações

O desenvolvimento deste projeto permitiu atingir com êxito o objetivo de implementar e configurar um servidor web funcional e robusto, integrando diversas tecnologias de código aberto. A utilização do Docker e Docker Compose provou ser uma escolha acertada, simplificando drasticamente o processo de **deploy** e configuração de ambientes complexos que envolvem múltiplas linguagens (PHP, Java) e bancos de dados (MariaDB, PostgreSQL).

Entre os principais desafios superados, destaca-se a configuração do proxy reverso com NGINX para o correto roteamento de tráfego e a gestão de redes internas do Docker para garantir a comunicação segura entre as aplicações e seus respectivos bancos de dados. A implementação do Moodle como LMS e do Spring Petclinic como aplicação Java demonstrou a versatilidade do servidor em hospedar diferentes tipos de cargas de trabalho.

Como recomendações para trabalhos futuros e melhorias no ambiente atual, sugere-se:

- **Implementação de HTTPS:** Configurar certificados SSL/TLS (por exemplo, com Let's Encrypt) no NGINX para garantir a segurança dos dados em trânsito.
- **Monitoramento e Logs:** Integrar ferramentas como Prometheus e Grafana para monitoramento de recursos e centralização de logs dos contêineres.
- **Backup Automatizado:** Implementar scripts de backup periódico para os volumes de dados dos bancos de dados e arquivos do Moodle.
- **CI/CD:** Estabelecer uma esteira de Integração e Entrega Contínua para automatizar a atualização das aplicações.

Este projeto serviu como uma excelente base prática para a consolidação dos conhecimentos em redes de computadores, sistemas operacionais e administração de sistemas.

Declaração de Conflito de Interesses

Os autores declaram não haver conflitos de interesse a declarar.

