RELATÓRIO TÉCNICO - MVP N2

Projeto: OMNISTACK: Solução Automatizada para Gestão de Laboratórios Virtuais
Disciplina: Ferramentas de Desenvolvimento para Web **Data:** 27/11/2025

1. Introdução e Escopo

Este relatório técnico documenta a entrega do MVP (Produto Mínimo Viável) para a avaliação N2. O escopo da entrega consiste na implementação containerizada da plataforma **AWX**, que atua como a interface web central e API de gerenciamento para a solução de orquestração de infraestrutura Omnistack.

O objetivo técnico foi demonstrar a aplicação dos conceitos de desenvolvimento web modernos, incluindo arquitetura de microsserviços, containerização e integração de sistemas (Frontend/Backend/Banco de Dados), conforme os requisitos da disciplina.

2. Arquitetura da Solução

Para atender aos requisitos de portabilidade e garantir a execução consistente em diferentes ambientes (Linux e Windows/WSL), optou-se por uma arquitetura de microsserviços orquestrada via **Docker Compose**. Esta abordagem desacopla os componentes do sistema, facilitando a manutenção e escalabilidade.

A pilha tecnológica (stack) é composta por três serviços interdependentes:

- **Serviço Web (Aplicação Principal):** Container baseado na imagem do AWX, responsável por servir a interface do usuário e processar as regras de negócio via API.
- **Serviço de Dados (Persistência):** Banco de dados relacional para armazenamento definitivo das informações do sistema.
- **Serviço de Cache e Mensageria:** Sistema em memória para gerenciamento de sessão, cache de aplicação e filas de tarefas assíncronas.

3. Tecnologias Implementadas

Abaixo, o detalhamento das tecnologias web e de infraestrutura utilizadas na construção da solução:

- **Frontend (Interface do Usuário):**
    - A interface web do sistema é uma SPA (*Single Page Application*) híbrida, utilizando **AngularJS** (para componentes legados) e **React** (para as novas visualizações de topologia e métricas).

    - A aplicação é servida pelo framework Django, proporcionando uma experiência de usuário responsiva e dinâmica para o gerenciamento de inventários e execução de *jobs*.

- **Backend & API:**
    - O núcleo do sistema é desenvolvido em **Python**, utilizando o framework **Django**.
    - Ele expõe uma **API RESTful** robusta que serve tanto ao frontend quanto a integrações externas, gerenciando autenticação, autorização (RBAC) e a orquestração das chamadas para o motor de automação (Ansible).
- **Banco de Dados:**
    - Utiliza-se o **PostgreSQL** (versão 12/13) como sistema gerenciador de banco de dados relacional (SGBD). Ele é responsável pela persistência crítica de dados, incluindo usuários, credenciais criptografadas, inventários de hosts e histórico de execução de tarefas.
- **Mensageria (Broker):**
    - O **Redis** (versão 6) é utilizado como *message broker* e cache. Ele gerencia a fila de tarefas assíncronas do *Task Manager*, permitindo que o servidor web delegue execuções pesadas (como o provisionamento de VMs) sem bloquear a interface do usuário.

4. Detalhes de Implementação e Desafios Superados

Durante o desenvolvimento do MVP, a adaptação do AWX (nativo para orquestração via Kubernetes) para um ambiente de desenvolvimento local "Docker puro" exigiu intervenções técnicas específicas para garantir a funcionalidade:

- **Injeção de Configuração:** Foi desenvolvido um arquivo settings.py customizado para definir manualmente os parâmetros de conexão com o banco de dados e Redis, injetado no container via *Volume Mount* do Docker.
- **Gerenciamento de Arquivos Estáticos:** Para viabilizar a execução fora de um servidor web reverso (como Nginx) em ambiente de desenvolvimento, foi necessário habilitar o modo DEBUG = True e configurar ALLOWED_HOSTS = ['*']. Isso permitiu que o próprio container Django servisse os ativos estáticos (CSS/JS) e aceitasse conexões locais, superando o problema de "tela branca" encontrado inicialmente.

- **Orquestração de Inicialização:** Implementou-se uma dependência condicional no Docker Compose, garantindo que a aplicação web só inicie após os serviços de dependência estarem operacionais.

5. Repositório e Distribuição

Conforme especificado na seção 2.6 e 2.7 do documento de requisitos :

- **Versionamento:** Todo o código de infraestrutura (docker-compose.yml, arquivos de configuração e documentação) está versionado no GitHub sob o repositório público: fatec-scs-fdw-2025-2-omnistack.
- **Imagens Docker:** As imagens utilizadas foram tageadas seguindo o padrão da disciplina (<usuario>/fatec-scs-imagem-omnistack-frontend e ...-db) e hospedadas no **DockerHub**, garantindo que o ambiente possa ser recriado pelo professor sem a necessidade de build local complexo.

6. Evidência de Funcionamento

A captura de tela abaixo comprova a execução bem-sucedida do ambiente na máquina de desenvolvimento local (localhost:8080), demonstrando o login de administrador efetuado e o acesso total ao Dashboard de gerenciamento.