Ambiente de desenvolvimento SIGAA, SIGRH, SIPAC e SIGAdmin usando Docker

Repositório GIT: https://git.ufla.br/sig-ufrn/ambiente-docker

Este repositório contém os arquivos necessários para construir e executar ambientes Docker do SIGUFRN com JBoss 5.1.0.GA. A estrutura separa ambientes de desenvolvimento, homologação e produção, além de manter recursos compartilhados e scripts auxiliares para testes.

Pré-requisitos

- Docker instalado.
- Docker Compose instalado.
- Git LFS instalado, pois o repositório versiona arquivos binários grandes, como JBoss, JDK e agentes de monitoramento.

```
sudo apt-get install git-lfs
git lfs install
```

Estrutura Geral

```
.
├── dev/                # Ambiente de desenvolvimento
├── homolog/            # Ambiente de homologação
├── prod/               # Ambiente de produção
├── resources/          # Recursos compartilhados para build e monitoramento
├── testes/             # Scripts de apoio para carga, rate limit e PostgreSQL
├── README.md           # Visão inicial do repositório
└── MANUAL.md           # Este manual
```

Ambientes Disponíveis

Ambiente	Diretório	Container principal	Uso esperado
Desenvolvimento	dev/	jboss_dev	Execução local com debug remoto e volumes de workspace.
Homologação	homolog/	jboss-homolog	Ambiente com JBoss e PostgreSQL no Compose.
Produção	prod/	jboss	Execução do JBoss em produção.

Cada ambiente possui seus próprios arquivos ***Dockerfile***, ***compose.yaml***, ***env.conf***, ***README.md***, ***add_user_host.sh*** e ***limpar-jboss.sh***.

Componentes Principais

- ***Dockerfile***: constrói a imagem do JBoss customizado.
- ***compose.yaml***: define containers, portas, volumes, rede, usuário e limites de memória.
- ***env.conf***: sobrescreve ou define JAVA_OPTS usados pelo JBoss.
- ***add_user_host.sh***: cria o usuário/grupo sistemas no host e ajusta permissões dos diretórios montados.
- ***limpar-jboss.sh***: remove componentes desnecessários do JBoss durante o build da imagem.
- ***config/***: contém arquivos copiados para dentro da imagem no momento do build.
- ***ear_sistemas/***: contém EARs, WARs, JARs e arquivos de configuração das aplicações.
- ***logs/***: recebe logs do JBoss montados a partir do container.
- ***resources/***: contém recursos base, como jboss-5.1.0.GA-jdk6.zip, JDK e agentes de monitoramento.

Arquitetura Docker

O build usa duas etapas:

1. Uma etapa ***builder***, baseada em Alpine, que descompacta o JBoss, remove instâncias e bibliotecas desnecessárias e aplica arquivos de configuração.
2. Uma etapa final, baseada em ***amazoncorretto:8-alpine***, que recebe o JBoss preparado, instala dependências de fonte/timezone, cria o usuário ***sistemas*** e executa o servidor.

O JBoss é iniciado com:

```
/opt/jboss-5.1.0.GA/bin/run.sh -b 0.0.0.0 -p 8080 -c inst1
```

Diretórios Montados

Os ambientes montam diretórios do host dentro do container. Os principais são:

Host	Container	Finalidade
./ear_sistemas	/opt/sistemas/ear	Artefatos das aplicações e configurações
./pobox	/opt/pobox	Arquivos do SIAFI
./usersOnline	/var/jboss_config/usersOnline	Dados de usuários online.
./sistemas/comum	/var/sistemas/comum	Arquivos comuns dos sistemas.
./comum	/var/comum	Arquivos comuns adicionais
./logs	/opt/jboss-5.1.0.GA/server/inst1/log	Logs do JBoss.
./fonts	/usr/share/fonts	Fontes usadas pelas aplicações.

No ambiente ***dev***, também há volumes para workspace local:

```
- /caminho/para/LIBS:/sig-ufrn/eclipse-workspace/LIBS/
- /caminho/para/SharedResources:/sig-ufrn/eclipse-workspace/SharedResources/
- /caminho/para/Sistema:/sig-ufrn/eclipse-workspace/Sistema/
```

Esses caminhos devem ser ajustados no ***dev/compose.yaml*** conforme a máquina do desenvolvedor.

Configuração Inicial

1. Copiar Recursos

Cada ambiente espera uma pasta resources/ local para o build. Copie a pasta da raiz para o ambiente desejado:

```
cp -r ../resources ./
```

Execute o comando dentro de dev/, homolog/ ou prod/.

2. Configurar Fontes

A pasta fonts/ deve conter as fontes esperadas pelo Compose, incluindo arquivos como:

- Arimo-Italic-VariableFont_wght.ttf
- Arimo-VariableFont_wght.ttf
- Century Gothic.ttf
- EnglishTowne.ttf
- arquivos da pasta static/

3. Configurar ear_sistemas

A pasta ear_sistemas/ deve conter os artefatos e configurações necessários para o ambiente.

Para desenvolvimento, normalmente são necessários:

- libs.jar/remotehosts.properties
- libs.jar/ldap.properties
- postgres-ds.xml
- EARs opcionais, como admin.ear

Para homologação e produção, normalmente são necessários:

- libs.jar/remotehosts.properties
- libs.jar/ldap.properties
- postgres-ds.xml
- barramentoservicos.ear
- dto.jar
- servicos.ear
- shared.ear
- sigaa.ear
- sipac.ear
- sigrh.ear
- admin.ear
- sigpp.ear
- probe.war

4. Ajustar env.conf

Revise JAVA_OPTS conforme os recursos disponíveis na máquina.

Parâmetros importantes:

- -Xms: memória inicial da JVM.
- -Xmx: memória máxima da JVM.
- -XX:MetaspaceSize: tamanho inicial do Metaspace.
- -XX:MaxMetaspaceSize: tamanho máximo do Metaspace.
- -Dbr.ufrn.jboss.instanceName: nome da instância JBoss.
- -Dfile.encoding=ISO-8859-1: encoding usado pela aplicação.

No dev, o debug remoto já fica habilitado na porta 8787.

Permissões

O container executa com o usuário sistemas. Para evitar problemas de escrita nos volumes, crie o mesmo usuário/grupo no host e ajuste a propriedade dos diretórios montados.

Dentro do diretório do ambiente, execute:

```
sudo ./add_user_host.sh
```

O script cria UID/GID 8080 e aplica *chown* nos diretórios:

- comum
- ear_sistemas
- fonts
- logs
- pobox
- sistemas
- usersOnline

No dev/compose.yaml, os argumentos de build atuais usam TARGET_UID=1000 e TARGET_GID=1000. Ajuste esses valores ou o script conforme o UID/GID necessário no host.

Build da Imagem

Entre no diretório do ambiente desejado e execute:

```
docker compose build --no-cache
```

Alternativamente, sem Compose:

```
docker build -t sigufrn --progress=plain --no-cache .
```

Alterações em Dockerfile, config/ ou limpar-jboss.sh exigem novo build da imagem.

Execução

Entre no diretório do ambiente e suba os containers:

```
docker compose up -d
```

Para acompanhar os logs:

```
docker logs -f jboss_dev
```

Em homologação:

```
docker logs -f jboss-homolog
```

Em produção:

```
docker logs -f jboss
```

Para acessar o shell:

```
docker exec -it jboss_dev /bin/sh
```

Ou ajuste o nome do container conforme o ambiente.

Portas

Ambiente	**Porta**	**Serviço**
dev, homolog, prod	8080	Aplicação JBoss
dev	8787	Debug remoto Java
homolog	15432	PostgreSQL exposto no host
opcional	9404	Métricas JMX Exporter
opcional	4000	Glowroot

opcional dev	8025	Interface MailPit
opcional dev	1025	SMTP MailPit
opcional dev	8079	Dozzle

Ambiente de Desenvolvimento

O ambiente dev/ usa a imagem ufla/jboss-5.1.0.ga-dev:2.1.0 e expõe a porta 8787 para debug remoto.

Para configurar o debug no Eclipse:

1. Acesse **Run > Debug Configurations....**
2. Crie uma configuração em **Remote Java Application**.
3. Use localhost ou 127.0.0.1 como host.
4. Use a porta 8787.
5. Clique em **Apply** e depois em **Debug**.

O dev/compose.yaml também contém blocos comentados para MailPit e Dozzle.

MailPit

Para interceptar e-mails, descomente o serviço mailpit no dev/compose.yaml e configure a aplicação para usar:

- host: mailpit-sigufrn
- porta SMTP: 1025
- interface web: http://127.0.0.1:8025/

Dozzle

Para visualizar logs pelo navegador, descomente o serviço dozzle no dev/compose.yaml e acesse:

```
http://127.0.0.1:8079/
```

Ambiente de Homologação

O ambiente homolog/ sobe dois serviços:

- jboss, com container jboss-homolog.
- postgres, com container postgres.

O PostgreSQL usa a imagem postgres:18-alpine, volume Docker pg_data e arquivo postgres-env.conf.

Revise principalmente:

```
POSTGRES_USER=postgres
POSTGRES_PASSWORD=password
POSTGRES_DB=postgres
POSTGRES_HOST_AUTH_METHOD=md5
```

Não mantenha senha padrão em ambiente compartilhado.

Ambiente de Produção

O ambiente prod/ sobe somente o serviço JBoss definido no compose.yaml.

Cuidados recomendados:

- Validar env.conf antes de subir o ambiente.
- Validar permissões dos volumes antes do deploy.
- Garantir que ear_sistemas/ contenha os artefatos corretos da versão a ser publicada.
- Não versionar logs, EARs, dados de runtime ou segredos.
- Recriar o container em janelas controladas de manutenção.

Para recriar o container:

```
docker compose stop jboss
docker compose pull
docker compose up -d jboss --force-recreate
```

Observabilidade

O repositório contém agentes em resources/monitoring/:

- Glowroot para traces.

- JMX Exporter para métricas Prometheus.

Para habilitar:

1. Copie resources/monitoring/ para o ambiente desejado.
2. Descomente o volume ./monitoring/:/opt/monitoring/ no compose.yaml.
3. Exponha as portas 4000 e 9404 no Dockerfile e no compose.yaml.
4. Descomente os agentes no env.conf.

As métricas do JMX Exporter ficam em 9404. O arquivo padrão resources/monitoring/jmx_exporter/config.yaml usa autenticação básica com usuário admin e senha admin; altere antes de usar fora de ambiente local.

O Glowroot fica disponível em 4000. No primeiro acesso, cadastre usuário e senha em Administration > Users > Add new.

Testes e Scripts de Apoio

Teste de Carga

Diretório: testes/teste_carga/

Simula vários usuários registrando ponto no SIGRH.

Configuração básica:

```
python3 -m venv venv
source venv/bin/activate
python3 -m pip install -r ./requirements.txt
cp .env.sample .env
```

Depois, ajuste o .env e configure usuários de teste no banco conforme o README do diretório.

Rate Limit

Diretório: testes/ratelimit/

Scripts disponíveis:

```
python3 teste-rate-limit.py
python3 teste-rate-limit-navegacao.py
```

Consultas PostgreSQL

Diretório: testes/consultas_postgres/

Contém SQL para salvar periodicamente consultas em execução no PostgreSQL.

Exemplo de execução:

```
export PGPASSWORD='password'
watch -n 1 psql -h 127.0.0.1 -d <bd> -U <user> -f salvar_consultas.sql
```

Arquivos Ignorados

O .gitignore exclui diretórios e arquivos de runtime, como:

- dev/ear_sistemas/, homolog/ear_sistemas/, prod/ear_sistemas/
- logs/
- resources/ copiado para cada ambiente
- fonts/ por ambiente
- pobox/, usersOnline/, comum/, sistemas/comum/
- .env dos testes
- ambientes virtuais Python

Isso evita versionar artefatos grandes, dados sensíveis, logs e arquivos específicos de máquina.

Fluxo Recomendado de Uso

Para configurar um ambiente do zero:

1. Entre no diretório do ambiente: dev/, homolog/ ou prod/.
2. Copie ../resources para o diretório atual.
3. Prepare fonts/.
4. Prepare ear_sistemas/ com EARs, WARs, JARs e arquivos de configuração.
5. Ajuste env.conf.
6. Ajuste volumes e portas no compose.yaml.
7. Configure permissões com sudo ./add_user_host.sh.
8. Gere a imagem com docker compose build --no-cache.
9. Suba o ambiente com docker compose up -d.
10. Valide logs com docker logs -f <container>.

Solução de Problemas

Erro de permissão em volumes

Execute novamente sudo ./add_user_host.sh no diretório do ambiente e confira se o UID/GID do container corresponde ao usuário do host.

Alteração em arquivos de config/ não aparece no container

Arquivos de config/ são copiados durante o build. Gere a imagem novamente com docker compose build --no-cache.

JBoss não encontra EARs ou arquivos de configuração

Confira se ear_sistemas/ existe, se os arquivos foram copiados corretamente e se os volumes estão apontando para o caminho correto no compose.yaml.

Debug remoto não conecta no desenvolvimento

Confira se o container jboss_dev está em execução, se a porta 8787 está exposta e se o Eclipse usa localhost:8787.

Métricas ou traces não aparecem

Confira se os agentes foram copiados, se o volume monitoring está montado, se as portas foram expostas e se as linhas do env.conf foram descomentadas.

Revisão #4
Criado 2026-08-02 11:52:05 UTC por OBEDE JESSE CARVALHO
Atualizado: 2026-08-02 12:35:09 UTC por OBEDE JESSE CARVALHO