ibict

RELATÓRIO DE CUMPRIMENTO META 01

ATUALIZAÇÃO TÉCNICA E TECNOLÓGICA NA BIBLIOTECA DIGITAL DO TJDFT

PRESIDÊNCIA DA REPÚBLICA

Luiz Inácio Lula da Silva
Presidente da República

Geraldo José Rodrigues Alckmin Filho
Vice-Presidente da República

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

Luciana Santos
Ministra da Ciência, Tecnologia e Inovação

INSTITUTO BRASILEIRO DE INFORMAÇÃO EM CIÊNCIA E TECNOLOGIA

Tiago Emmanuel Nunes Braga
Diretor

Carlos Andre Amaral de Freitas
Coordenador de Administração - COADM

Ricardo Medeiros Pimenta
Coordenador de Ensino e Pesquisa em Informação para a Ciência e Tecnologia - COEPI

Henrique Denes Hilgenberg Fernandes
Coordenador de Planejamento, Acompanhamento e Avaliação - COPAV

Cecília Leite Oliveira
Coordenador Geral de Informação Tecnológica e Informação para a Sociedade - CGIT

Washington Luís Ribeiro de Carvalho Segundo
Coordenador Geral de Informação Científica e Técnica - CGIC

Hugo Valadares Siqueira
Coordenador Geral de Tecnologias de Informação e Informática - CGTI

Milton Shintaku
Coordenador de Tecnologias para Informação - COTEC

**MINISTÉRIO DA CIÊNCIA,
TECNOLOGIA E INOVAÇÃO**

Instituto Brasileiro de Informação
em Ciência e Tecnologia

RELATÓRIO DE CUMPRIMENTO META 01
ATUALIZAÇÃO TÉCNICA E TECNOLÓGICA NA BIBLIOTECA DIGITAL DO TJDFT

Coordenação de Tecnologia para Informação (COTEC)

Brasília

2023

EQUIPE TÉCNICA

Diretor do Instituto Brasileiro de Informação em Ciência e Tecnologia
Tiago Emmanuel Nunes Braga

Coordenador-Geral de Tecnologias de Informação e Informática – CGTI
Hugo Valadares Siqueira

Coordenador do Projeto
Milton Shintaku

Autores do relatório
Elton Mártires Pinto
Lucas Ângelo Silveira
Milton Shintaku
Mirele Costa

Revisão
Flavia Karla Ribeiro Santos
Rafael Teixeira de Souza

Normalização
Elton Mártires Pinto

Diagramação e projeto gráfico
Rafael Fernandez Gomes
Nuielle Medeiros

Este Relatório de Técnico é um produto do Projeto:

Ref. IBICT - Processo SEI nº 01302.000390/2020-38

Ref. FUNDEP 28331

Sumário

1. INTRODUÇÃO

O DSpace, criado pela Biblioteca do Massachusetts Institute of Technology (MIT) e Hewlett Packard (HP) e, posteriormente, mantido pela DuraSpace, foi baseado no Movimento de Acesso Aberto. Nesse sentido, essa ferramenta livre e de código aberto foi desenvolvida com o objetivo de criar repositórios institucionais e bibliotecas digitais, tal como disseminar a produção técnica e científica de instituições, sobretudo as voltadas para ensino e pesquisa.

No âmbito jurídico, o Superior Tribunal de Justiça (STJ) utilizou, em 2004, o DSpace para a criação da Biblioteca Digital Jurídica (BDJur). Sua criação influenciou diversos tribunais a criarem suas próprias bibliotecas digitais, possibilitando assim, a disseminação de suas produções intelectuais. Entre eles, o Tribunal de Justiça do Distrito Federal e dos Territórios (TJDFT).

Entretanto, tanto a implementação como a manutenção e atualização não são processos simples, pois requerem conhecimentos científicos, técnicos e tecnológicos. Além disso, é necessário amparo conceitual para criar um modelo de arquitetura da informação e elaborar e definir políticas que orientem o funcionamento da BD de acordo com a finalidade da biblioteca física e da instituição.

No TJDFT, a Biblioteca Digital (BD) é o portal de acesso ao acervo digital de interesse da comunidade do Tribunal. Por isso, reúne, preserva e divulga a produção intelectual dos magistrados, documentos de memória, bem como, disponibiliza o conteúdo das principais editoras jurídicas. A BD/TJDFT faz a Disseminação Seletiva da Informação (DSI), permitindo que os usuários recebam informações das atualizações das coleções.

Mesmo que siga o modelo da BDJur, a BD/TJDFT possui suas especificações, o que requer estrutura diferenciada e atendimento às orientações do Tribunal. Assim, em 2020 foi iniciado um projeto de pesquisa para atualizar o DSpace da versão 3 para a versão 6, bem como propor nova arquitetura da informação e layout, finalizado em dezembro de 2022.

Durante o período de atualização do software foi lançado o DSpace 7, versão que apresenta uma série de melhorias, como correção de *bugs*, melhor usabilidade e aprimoramento de recursos. E, após diversos testes da equipe do Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict) e o desejo de atualização da equipe do TJDFT foi proposto um aditivo para atualizar o software da versão 6.3 para a 7, bem como realizar ajustes na Revista de Doutrina Jurídica (RJD).

2. OBJETIVOS

2.1 Objetivo Geral

Levantamento dos novos desafios diante do cenário editorial decorrido das mudanças promovidas pela Capes e do lançamento da nova versão do DSpace.

2.2 Objetivos Específicos

- Levantar impactos ofertados na versão 7 do DSpace;
- Atualizar a biblioteca digital para a versão 7 do DSpace;
- Capacitar a equipe do TJDFT na versão 7 do DSpace.

3. RESULTADOS

3.1 Levantar impactos ofertados na versão 7 do DSpace

A versão 7 apresenta um conjunto de melhorias, como correção de *bugs*, melhorias de usabilidade e aprimoramento de recursos. Esta nova versão caracteriza-se pela formulação de sua arquitetura técnica, bem como pela formulação da arquitetura de informação baseada em entidades. É possível gerir informações relacionadas a um autor, uma revista ou qualquer outra entidade que tenha um perfil próprio. Com isso, será possível identificar de forma inequívoca os autores, bem como obter um nível de detalhe de informação que não existia nas versões anteriores (OPEN SCIENCE, 2020, online)[1].

3.1.1 Inovações da versão 7 do Dspace

Segundo a organização Duraspace, o DSpace 7 traz para o DSpace uma nova interface de usuário única e moderna baseada em uma tecnologia *open-source* utilizada para desenvolvimento de *front-end*[2], a plataforma de aplicações web Angular. No Dspace 7 essa interface de usuário criada no Angular substituiu as interfaces de usuário JSPUI (*Java Server Pages User Interface*) e XMLUI (*eXtented Mark Language User Interface*).

Nessa nova versão do DSpace o suporte para *Application Programming Interface* (API) *Representational State Transfer* (REST) foi aprimorado além de integrar padrões tecnológicos e as melhores práticas atuais segundo as recomendações do relatório *Next Generation Repository* (NGR). O Angular UI (em inglês, *User Interface*) possui a vantagem de permitir que a interface de usuário do DSpace 7 seja executada em um servidor separado do *back-end*[3] API REST, assim nessa nova versão, o *front-end* e *back-end* podem ser instalados separadamente. O DSpace 7 combina a nova interface de usuário com o *back-end* principal do DSpace 6.x aprimorado, resultando em um repositório enxuto, desacoplado e responsivo.

3.1.2 Angular UI

Angular UI é a segunda versão do *framework javascript* angular criado pelo Google, disponível em: https://angular.io/. É uma tecnologia baseada em *TypeScript*[4]. Para acompanhar a evolução tecnológica, os desenvolvedores do Angular perceberam que seria melhor criar um *framework* do zero. Da primeira versão o AngularJS, aproveitou-se unicamente a experiência obtida e as necessidades dos desenvolvedores. Já o Angular UI é uma plataforma para desenvolvimento de aplicações web e também *mobile*. Vale ressaltar que, aplicações desenvolvidas na primeira versão, não são compatíveis com Angular UI. Existem várias razões pelas quais o Angular UI chamou a atenção imediata quando foi lançado no final de 2016:

1 Disponível em: https://openscience.usdb.uminho.pt/?p=6341. Acesso em: 12 jan. 2023.

2 *Front-end* - camada da aplicação que interage diretamente com o usuário.

3 *Back-end* - camada de programação, desenvolvimento, estrutura tecnológica, suporte, etc. de um sistema que recebe os dados da interface do usuário.

4 *TypeScript*- superconjunto de *javascript* desenvolvido pela Microsoft que adiciona tipagem e alguns outros recursos a linguagem. A linguagem pode ser usada para desenvolver aplicações *JavaScript* tanto do lado cliente quanto do servidor.

- Suporte para otimização de sites. As aplicações criadas podem ser facilmente indexadas/pesquisadas pelo Google ou pelo Google Scholar;

- Suporte a diretrizes de acessibilidade. Os leitores de tela (e similares) podem ter problemas com aplicações *Javascript* do lado do cliente. No entanto, a equipe desenvolvedora fez um estudo com a ajuda de especialistas em acessibilidade da Universidade do Kansas, descartando qualquer problema;

- Suporte para arquivamento na Web (por exemplo, coleta de arquivos na internet);

- Suporte para aplicações continuarem executando mesmo quando o *Javascript* está desativado. Para dar suporte a esse conceito, o Angular UI pré-compila o *javascript* em HTML "estático" no servidor. Assim, caso o *javascript* esteja desativado, seus usuários simplesmente solicitam novas páginas HTML estáticas pré-compiladas cada vez que clicam em links ou botões. Isso é feito por um módulo chamado Angular Universal;

- Suporte de plugins terceiros para criar/aprimorar aplicações;

- Suporte para experiência de usuário mais dinâmica e moderna. Embora isso também possa ser alcançado em tecnologias do lado do servidor (Java, Ruby, etc.), essas tecnologias necessitam a utilização de estruturas *javascript* (jQuery ou similares) para fornecer a mesma experiência;

- Suporte na separação entre a interface do usuário e o *back-end*. A criação da interface do usuário em uma tecnologia do lado do cliente forçando uma separação entre o código do servidor (*back-end*) e a interface do usuário. Embora essa separação não seja necessária, é uma prática recomendada;

- Suporte para API REST aprimorada. Embora o Dspace tenha uma API REST desde o Dspace 4.0, essa API REST é muito limitada em termos de funcionalidade e casos de uso práticos. A criação de uma aplicação do lado do cliente requer uma API REST estável e bem documentada que forneça todos os recursos da interface do usuário necessários. Uma API REST com todos os recursos também possui um benefício secundário de fornecer um caminho mais fácil para futuras integrações com o Dspace (por outras plataformas ou plugins de terceiros);

- Suporte para inovações tecnológicas. Nenhuma plataforma de software pode durar para sempre sem aprimoramentos contínuos baseados nas tecnologias mais recentes. O Dspace não é diferente, as duas interfaces de usuário JSPUI e XMLUI estão desatualizadas. (Embora tenham recebido uma refatoração de código recente, o JSPUI foi lançado inicialmente em 2002 e o XMLUI em 2008, mas conta com uma estrutura *Apache Cocoon*[5] não mantida e quase obsoleta).

3.1.3 Representational state transfer

Representational State Transfer (REST) é um modelo utilizado em projetos de software distribuído. O modelo REST descrito por um dos principais criadores do protocolo *Hypertext Transfer Protocol* (HTTP), inicialmente proposto para a evolução da arquitetura do protocolo, atualmente vem sendo aproveitado na implementação de *Web Services*[6], passando a utilizar o REST como uma alternativa ao SOAP. De forma resumida, SOAP é um protocolo

5 *Apache Cocoon- framework* para desenvolvimento web baseado em componentes e no conceito de separação de interesses.

6 *Web Service* é a solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes possibilitando interações compatíveis.

baseado em XML[7] para troca de informações num ambiente distribuído e descentralizado. Dessa forma, REST pode ser visto como um conjunto de princípios a serem utilizados em aplicações Web, os princípios e a forma correta de utilizá-los são apresentados na sequência.

3.1.3.1 Identificação de recursos

Aplicações geralmente são responsáveis por manter e gerenciar informações. Aplicações que lidam com *E-commerce* possuem informações sobre clientes, vendas, produtos e etc. No mundo REST, tais informações são chamadas de recursos. No REST, cada recurso possui um identificador único. Tal identificação é utilizada na aplicação para diferenciar qual recurso deve ser manipulado a cada solicitação. A identificação é feita utilizando o conceito de URI (*Uniform Resource Identifier*). Exemplos: http://servicorest.ibict .br/produto ; http://restservice.com.br/cliente e http://restservice.com.br/cliente/11.

As URIs funcionam como um contrato que será utilizado entre serviço e cliente. Algumas boas práticas no uso de URIs são listadas abaixo:

- **Utilizar URIs legíveis:** nomes de fácil dedução relacionados ao domínio da aplicação;
- **Utilizar o mesmo padrão de URI:** criar um padrão de nomenclatura para as URIs;
- **Evitar operação na URI:** manipulação dos recursos exclusivamente por métodos do protocolo HTTP (GET, POST, PUT, DELETE). Evitar URIs tais como: http:// servicorest.ibict.br/produto/cadastrar e http:// servicorest.ibict.br/produto/1/deletar;
- **Evitar alterações nas URIs:** alterar a sintaxe causa impacto nos clientes que estavam utilizando o recurso.

3.1.3.2 Manipulação de recursos por métodos HTTP

O protocolo HTTP (HTTP, 2014) possui diversos métodos, sendo que cada um indica o tipo de manipulação a ser realizada, geralmente as aplicações REST apenas utilizam os métodos GET, POST, PUT e DELETE. Quadro 1 apresenta os cenários de utilização de cada um:

Quadro 1 - Recursos HTTP

Método HTTP	Semântica
GET	obtém os dados do recurso
POST	cria um novo recurso
PUT	substitui os dados de um recurso
DELETE	exclui o recurso

Fonte: Masar e Donohue (2023, online).

7 XML é uma recomendação da W3C para gerar linguagens de marcação para descrever diversos tipos de dados, cujo propósito principal é o compartilhamento de informações na internet.

O padrão de utilização dos métodos HTTP em um recurso chamado produto é apresentado no Quadro 2.

Quadro 2 - Recursos HTTP aplicados no REST.

Método	URI	Utilização
GET	/produtos	recuperar todos os produtos na aplicação
GET	/produtos/id	recuperar os dados de um determinado produto
POST	/produtos	criar um novo produto
PUT	/produtos/id	atualizar dados de um determinado produto
DELETE	/produtos/id	excluir um determinado produto

Fonte: Masar e Donohue (2023, online).

Como boa prática, evite utilizar apenas o método POST em requisições tais como: alteração e exclusão. Além disso, evita o método GET nas operações citadas, devido aos navegadores fazerem cache de requisições GET.

3.1.3.3 Representação dos recursos

Quando um recurso é solicitado por uma aplicação cliente, por exemplo uma requisição GET, elas não são removidas do servidor, como se estivesse sendo transferido para o cliente, mas sim, o que é transferido para o cliente é a representação do recurso. Atualmente, há inúmeros formatos para representar recursos, onde os mais populares são: XML, JSON[8], HTML e CSV. A comunicação entre aplicações é feita via transferência de representações dos recursos a serem manipulados gerando um desacoplamento entre cliente e servidor, algo benéfico, visto que, facilita a manutenção das aplicações.

Uma boa prática em aplicações REST é o suporte a múltiplas representações em um dado serviço, o que facilita a inclusão de novos clientes. Ao suportar múltiplas representações de recursos, espera-se que o cliente informe o formato desejado. No REST, essa negociação entre cliente X servidor é chamado de *Content Negotiation* e é feito via cabeçalho HTTP denotado como *accept*. Assim, ao fazer uma requisição ao serviço, o cliente adiciona na requisição o cabeçalho *accept*, indicando ao servidor o formato esperado da representação do recurso.

3.1.3.4 Evite manter dados de autenticação/autorização em sessão

A principal dificuldade em criar um serviço REST totalmente independente ocorre quando é necessário lidar com os dados de autenticação/autorização dos clientes. É natural os desenvolvedores armazenarem essas informações em sessão, método comum ao se desenvolver uma aplicação Web tradicional. A solução para resolver esse problema é a utilização de Tokens de acesso, que são gerados pelo serviço REST e devem ser armazenados pelos clientes, seja por *cookies* ou HTML 5 *Web Storage*[9]. Vale ressaltar que, o token deve ser enviado pelos clientes a cada nova requisição ao serviço. Existem diversas tecnologias e padrões para se trabalhar com Tokens, os mais comuns são:

8 JSON- acrônimo de *JavaScript Object Notation*, é um formato de padrão aberto de troca de dados simples e rápida entre sistemas que utiliza texto legível a humanos no formato atributo-valor.

9 HTML 5 *Web Storage*- aplicações Web podem armazenar dados localmente no navegador do usuário.

- OAUTH: padrão aberto para autorização utilizado para permitir validação de usuários na Internet sem expor suas senhas em sites de terceiros usando contas Google, Facebook, Microsoft, Twitter, etc;

- JWT (*JSON Web Token*): um método RCT 7519, padrão da indústria para realizar autenticação entre duas partes por meio de um token assinado que tem como objetivo autenticar uma requisição web. O token é um código em Base64 que armazena objetos JSON com os dados que permitem a autenticação da requisição;

- Keycloack: é um produto de software de código aberto que permite login único com o gerenciamento de identidades e gerenciamento de acesso.

3.1.3.5 Utilização correta dos códigos HTTP

Toda requisição HTTP a um servidor deve resultar em uma resposta. Além disso, cada resposta devolve ao cliente um código informando o status da requisição. Há dezenas de códigos HTTP, cada um com uma semântica específica (HTTP, 2014). Os códigos HTTP são agrupados em classes, conforme demonstrado a seguir no Quadro 3:

Quadro 3 - Classe de códigos http, fonte

Classe	Semântica
2xx	requisição processada com sucesso
3xx	indica uma ação ao cliente para que a requisição seja concluída
4xx	indica erro(s) na requisição causado(s) pelo cliente
5xx	Indica que a requisição não pode ser concluída devido a erro(s) no servidor

Fonte: Masar e Donohue (2023, online).

A boa prática consiste em conhecer os principais códigos HTTP e utilizá-los de maneira correta. Quadro 4 apresenta os principais códigos HTTP e quando os utilizar:

Quadro 4: Principais códigos HTTP, quando os utilizar

Código	Descrição	Forma de utilizar
200	*OK*	requisições GET, PUT e DELETE executadas com sucesso
201	*Created*	requisições POST, indicando um novo recurso criado com sucesso
206	*Partial Content*	requisições GET que devolvem apenas uma parte do conteúdo de um recurso
302	*Found*	requisições feitas à URIs antigas que foram alteradas
400	*Bad Request*	requisições com informações enviadas pelo cliente inválidas
401	*Unauthorized*	requisições que exigem autenticação, mas seus dados não foram fornecidos
403	*Forbidden*	requisições que o cliente não tem permissão de acesso ao recurso solicitado

Código	Descrição	Forma de utilizar
404	*Not Found*	requisições para uma URI com recurso inválido
405	*Method Not Allowed*	requisições onde o método HTTP informado pelo cliente não é suportado
406	*Not Acceptable*	requisições com formato de representação do recurso requisitado pelo cliente não é suportado
415	*Unsupported Media Type*	requisições cujo formato da representação do recurso enviado pelo cliente não seja suportado
429	*Too Many Requests*	recurso possui limite de requisições para um mesmo cliente, e o cliente o atingiu
500	*Internal Server Error*	requisições onde um erro tenha ocorrido no servidor
503	*Service Unavailable*	requisições feitas a um serviço fora do ar.

Fonte: Masar e Donohue (2023, online).

É importante utilizar o código correto para cada tipo de situação. A prática de utilizar um mesmo código genérico para todas as situações, pode ocasionar complicações na manutenção, caso a aplicação apresente problemas futuros.

3.2 Atualizar Biblioteca Digital para a Versão 7 do DSpace

A atualização está sendo feita inicialmente em ambiente de desenvolvimento. Os arquivos estão armazenados no git-lab fornecido pelo TJDFT na url: https://gitlab.tjdft.jus.br/externos/ibict/dspace.git

Lembrando que o acesso é restrito a usuários do TJDFT e para prestadores de serviços cadastrados.

3.2.1 Criação do ambiente

Ambiente de container a nível de desenvolvimento chamado de stage no TJDFT:

- pod para o solr
- pod para o postgreSQL
- pod para o *front-end*
- pode para o *back-end*

3.2.2 Dependências do *back-end*

- Linguagem de programação Java versão 11
- servidor web tomcat versão 9
- Gerenciador de dependências Maven versão 3.86

- Deploy da aplicação com Ant versão 1.10.12
- Solr versão 8.11.2 para indexação dos metadados
- Persistência com postgreSQL 13

3.2.3 Deploy do solr

A integração com o openshift é feita via git-lab. O ambiente de desenvolvimento está em ambiente de container e o dockerfile para implantar o solr é dado a seguir:

```
ARG SOLR_VERSION=8.11

FROM solr:${SOLR_VERSION}-slim

ENV AUTHORITY_CONFIGSET_PATH=/opt/solr/server/solr/configsets/authority/conf
\
    OAI_CONFIGSET_PATH=/opt/solr/server/solr/configsets/oai/conf \
    SEARCH_CONFIGSET_PATH=/opt/solr/server/solr/configsets/search/conf \
    STATISTICS_CONFIGSET_PATH=/opt/solr/server/solr/configsets/statistics/
conf

USER root
# Define a senha do usuário (substitua pela senha desejada)
RUN echo 'solr:solrsenha' | chpasswd
#RUN usermod -s shell solr

RUN mkdir -p $AUTHORITY_CONFIGSET_PATH && \
    mkdir -p $OAI_CONFIGSET_PATH && \
    mkdir -p $SEARCH_CONFIGSET_PATH && \
    mkdir -p $STATISTICS_CONFIGSET_PATH

COPY ./Dockers/dspace-solr/solr/authority/conf/* $AUTHORITY_CONFIGSET_PATH/
COPY ./Dockers/dspace-solr/solr/oai/conf/* $OAI_CONFIGSET_PATH/
COPY ./Dockers/dspace-solr/solr/search/conf/* $SEARCH_CONFIGSET_PATH/
COPY ./Dockers/dspace-solr/solr/statistics/conf/* $STATISTICS_CONFIGSET_
PATH/

RUN chown -R solr:solr /opt/solr/server/solr/configsets

USER solr
```

3.2.4 Deploy do postgreSQL

A integração com o openshift é feita via git-lab. O ambiente de desenvolvimento está em ambiente de container e o dockerfile para implantar o postgreSQL é dado a seguir:

```
ARG POSTGRES_VERSION=13
FROM postgres:${POSTGRES_VERSION}
# Defina as variáveis de ambiente para o banco de dados, usuário e senha
ENV POSTGRES_DB=dspace
ENV POSTGRES_USER=dspace
ENV POSTGRES_PASSWORD=root
```

3.2.5 Deploy do *back-end*

A integração com o openshift é feita via git-lab. O ambiente de desenvolvimento está em ambiente de container e o dockerfile para implantar o *back-end* é dado a seguir:

```
FROM danielucb/ubuntu:22.04
MAINTAINER lucasangelo <lucasangelo@ibict.br>

# This Dockerfile uses JDK11 by default
 ARG JDK_VERSION=11
 ARG TOMCAT_VERSION=9-jdk11-openjdk
 ARG SERVER_TJDFT=Arquitetura/server.xml
 ARG DSPACE_CONFIG=Arquitetura/dspace.cfg
 ARG DEST_DIR=/usr/local
 ARG DSP_DIR=/dspace
 ARG DSPACE_TJDFT=Arquitetura/DSpaceBack.tar.gz
 ARG SOURCE_BACK=DSpace-dspace-7.4
 ARG SOURCE_SOLR=Arquitetura/solr-8.11.2.tgz

RUN apt-get update && \
    apt-get install -y wget && \
    apt-get install -y cron && \
    apt-get install -y openjdk-$JDK_VERSION-jdk && \
    apt-get install -y nano && \
    apt-get install -y git && \
    apt-get install -y nano && \
    apt-get install -y curl && \
    apt-get clean

RUN mkdir -p “/tmp/solr8”
COPY “$SOURCE_SOLR” “/tmp”
RUN tar xzf “/tmp/solr-8.11.2.tgz” -C “/tmp/solr8”    --strip-components=1
&& \
     rm “/tmp/solr-8.11.2.tgz”

RUN mkdir -p “/tmp/DSpace-dspace-7.4”
COPY “$DSPACE_TJDFT” “/tmp”
RUN tar xzf “/tmp/DSpaceBack.tar.gz” -C “/tmp/DSpace-dspace-7.4”    --stri-
p-components=1 && \
     rm “/tmp/DSpaceBack.tar.gz”
# Configurando as variáveis de ambiente do Java
ENV JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
ENV PATH=$PATH:$JAVA_HOME/bin

# Definição da versão do Tomcat
ARG TOMCAT_VERSION=9.0.76

# Download e instalação do Apache Tomcat 9
RUN apt-get install -y curl && \
    curl -O https://downloads.apache.org/tomcat/tomcat-9/v${TOMCAT_VER-
SION}/bin/apache-tomcat-${TOMCAT_VERSION}.tar.gz && \
    tar -xvf apache-tomcat-${TOMCAT_VERSION}.tar.gz && \
    rm apache-tomcat-${TOMCAT_VERSION}.tar.gz && \
    mv apache-tomcat-${TOMCAT_VERSION} /opt/tomcat
```

```
# Definindo a variável de ambiente CATALINA_HOME
ENV CATALINA_HOME=/opt/tomcat
COPY “$SERVER_TJDFT” “$CATALINA_HOME”/”conf/”
RUN chmod +x “$CATALINA_HOME”/”conf/server.xml”

#  MVN ENV
ARG  MVN_TARGZ=Arquitetura/apache-maven-3.8.6.tar.gz
COPY “$MVN_TARGZ” “$DEST_DIR”
ENV MAVEN_HOME   “/usr/share/maven”
ENV MAVEN_CONFIG “/root/.m2”
ENV MAVEN_OPTS   “-Dhttps.protocols=TLSv1,TLSv1.1,TLSv1.2”
RUN  mkdir -p “$MAVEN_HOME”       && \
     mkdir -p “$MAVEN_HOME”/”ref” && \
     mkdir -p “$MAVEN_CONFIG”     && \
     tar xzf “$DEST_DIR”/”apache-maven-3.8.6.tar.gz” -C “$MAVEN_HOME”
--strip-components=1 && \
     ln -snf “$MAVEN_HOME”/”bin/mvn” /usr/bin/mvn  && \
     rm “$DEST_DIR”/”apache-maven-3.8.6.tar.gz”

# Definição da versão do Apache Ant
ARG  ANT_TARGZ=Arquitetura/apache-ant-1.10.12-bin.tar.gz
#  ANT ENV
COPY “$ANT_TARGZ” “$DEST_DIR”
ENV ANT_HOME   “/usr/share/ant”
RUN  mkdir -p “$ANT_HOME”       && \
     tar xzf “$DEST_DIR”/”apache-ant-1.10.12-bin.tar.gz” -C “$ANT_HOME”
--strip-components=1 && \
     chmod 775 “$ANT_HOME”/”bin/ant” && \
     ln -snf “$ANT_HOME”/”bin/ant” /usr/bin/ant  && \
     rm “$DEST_DIR”/”apache-ant-1.10.12-bin.tar.gz”

# Compilando o back-end

RUN  cd “/tmp/$SOURCE_BACK” && \
mvn package

#Copiando o arquivo de configuração do back-end
COPY “$DSPACE_CONFIG” “/tmp/$SOURCE_BACK/dspace/target/dspace-installer/
config/”

#script executado para implantar o back e iniciar tomcat e solr
copy iniciar.sh /tmp/iniciar.sh
RUN chmod +x /tmp/iniciar.sh

# DEPLOY no tomcat
RUN ln -snf “$DSP_DIR”/webapps/server “$CATALINA_HOME”/webapps/server

# Expondo a porta do tomcat e do solr
EXPOSE 8080 8983

ENTRYPOINT /tmp/iniciar.sh
```

Conteúdo do arquivo iniciar.sh:

```
cp -R /tmp/solr8 /dspace/
chmod -R 775 /dspace/solr8

#start solr
cd /dspace/solr8/bin/ && ./solr start -force

cd /tmp/DSpace-dspace-7.4/dspace/target/dspace-installer/ &&
     ant fresh_install

ln -s /dspace/solr/authority /dspace/solr8/server/solr/configsets/
ln -s /dspace/solr/oai /dspace/solr8/server/solr/configsets/
ln -s /dspace/solr/search /dspace/solr8/server/solr/configsets/
ln -s /dspace/solr/statistics /dspace/solr8/server/solr/configsets/

#restart solr
cd /dspace/solr8/bin/ && ./solr restart -force

#start tomcat
/opt/tomcat/bin/catalina.sh run
```

Acesso a aplicação é dentro do domínio do TJDFT por questões de segurança na url https://ibict-dspace.apps.tjdft.jus.br/server

3.2.6 Instalação do *front-end*

A integração com o openshift é feita via git-lab. O ambiente de desenvolvimento está em ambiente de container e o dockerfile para implantar o *front-end* é dado a seguir:

```
FROM danielucb/ubuntu:22.04
#FROM ubuntu:22.04
MAINTAINER lucasangelo <lucasangelo@ibict.br>

  ARG DSPACE_TJDFT=Arquitetura/dspace-angular-dspace-7.5.tar.gz
  ARG CONFIG_TJDFT=Arquitetura/dspace-ui.json
  ARG CONFIGFRONT_TJDFT=Arquitetura/config.prod.yml
  ARG VERSION_SOURCE=7.5
  ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y wget && \
    apt-get install -y cron && \
    apt-get install -y nano && \
    apt-get install -y git && \
    apt-get install -y nano && \
    apt-get install -y curl && \
    apt-get clean

# Atualize os pacotes do sistema
RUN apt-get upgrade -y

# Download e instalação do Node.js versão 16
RUN curl -fsSL https://deb.nodesource.com/setup_16.x | bash -
RUN apt-get install -y nodejs

# Instale o npm versão 9
RUN npm install -g npm@9
```

```
# Instale o PM2 globalmente
RUN npm install -g pm2

# Faça o download e instale o Yarn
RUN curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add -
RUN echo “deb https://dl.yarnpkg.com/debian/ stable main” | tee /etc/apt/
sources.list.d/yarn.list
RUN apt-get update && apt-get install -y yarn


#COPIANDO O FONTE
RUN mkdir -p “/tmp/dspace-angular-dspace-$VERSION_SOURCE”
COPY “$DSPACE_TJDFT” “/tmp”
RUN tar xzf “/tmp/dspace-angular-dspace-${VERSION_SOURCE}.tar.gz” -C “/tmp/
dspace-angular-dspace-$VERSION_SOURCE”    --strip-components=1 && \
     rm “/tmp/dspace-angular-dspace-${VERSION_SOURCE}.tar.gz”

#COPIANDO O ARQUIVO DE CONFIG DO DEPLOY
COPY “$CONFIG_TJDFT” “/tmp”

# Navegue até o diretório da aplicação
WORKDIR /tmp/dspace-angular-dspace-$VERSION_SOURCE
# Execute o comando yarn install para baixar as dependências
RUN yarn install

RUN chmod +x -R “/tmp/dspace-angular-dspace-$VERSION_SOURCE”

COPY “$CONFIGFRONT_TJDFT” “/tmp/dspace-angular-dspace-${VERSION_SOURCE}/
config”
# Defina a variável de ambiente NODE_OPTIONS para aumentar o limite de memó-
ria
RUN NODE_OPTIONS=”--max-old-space-size=4096” yarn build:prod

#Execute o comando yarn build:prod para criar a versão de produção
RUN yarn build:prod

copy iniciarFront.sh /tmp/iniciarFront.sh
RUN chmod +x /tmp/iniciarFront.sh

# Exponha a porta 4000
EXPOSE 4000

ENTRYPOINT /tmp/iniciarFront.sh
```

Conteúdo do iniciarFron.sh:

```
#!/bin/bash

cp -R /tmp/dspace-angular-dpace-7.5/dist /dspace-ui-deploy/
cp -R /tmp/dspace-angular-dpace-7.5/config /dspace-ui-deploy/
cp /tmp/dspace-ui.json /dspace-ui-deploy/
chmod -R 775 /dspace-ui-deploy/
cd /dspace-ui-deploy/ && pm2 start dspace-ui.json
```

Acesso a aplicação é dentro do domínio do TJDFT por questões de segurança na url https://ibict-dspace.apps.tjdft.jus.br/server

4. CONSIDERAÇÕES FINAIS

Após uma série de ataques de hackers, o TJDFT revogou acessos de prestadores de serviços até reforçar suas regras de segurança visando proteger seus ambientes e dados sensíveis. No entanto, a equipe do Ibict ficou incapacitada de atuar durante cerca de 4 meses, uma vez que para a implantação se faz necessário o acesso ao ambiente. As novas regras implementadas resultaram em restrições rigorosas, exigindo novos meios de autenticação, o que também impactou consideravelmente no processo de atualização, resultando no atraso e na impossibilidade de realizar a capacitação antes da entrega deste relatório de meta. Entretanto, o Ibict garante a realização do treinamento uma vez que a BD (versão 7) esteja disponível e funcionando corretamente.

REFERÊNCIAS

MASAR, Ivan; DONOHUE, Tim. DSpace release 7.0 status, 2023. Disponível em: https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status#DSpaceRelease7.0Status-MajorbenefitsofDSpace7. Acesso em: 13 nov. 2023.

OPEN SCIENCE. DSpace: a nova geração de repositórios, 2020. Disponível em: https://openscience.usdb.uminho.pt/?p=6341. Acesso em: 13 nov. 2023.

ibict
Instituto Brasileiro de Informação em Ciência e Tecnologia

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO