IMPLEMENTAÇÃO DO MONITORAMENTO DE TAXA DE OBSOLESCÊNCIA DOS FORMATOS

1. Infraestrutura Base

Banco de Dados de Formatos Crie um repositório central para armazenar informações sobre formatos:

- Nome e extensão do formato
- Versão
- Especificações técnicas
- Status (ativo, obsoleto, em observação)
- Data de inclusão no sistema
- Ferramentas compatíveis
- Nível de risco de obsolescência

Integração com Ferramentas de Identificação

- **DROID**: Automatize varreduras regulares dos repositórios digitais para identificar formatos em uso
- **JHOVE**: Configure validações automáticas de integridade e conformidade dos arquivos
- Integre os resultados dessas ferramentas ao seu banco de dados central

2. Sistema de Monitoramento Automatizado

Scripts de Auditoria Periódica

```
Frequência: Mensal ou trimestral
Ações:
```

```
- Varredura completa do acervo digital
- Identificação de novos formatos
- Validação de formatos existentes
- Geração de relatórios comparativos
```

Métricas a Coletar

- Total de formatos únicos no acervo
- Formatos suportados vs. não suportados
- Distribuição de arquivos por formato
- Idade média dos formatos
- Formatos em risco de obsolescência

3. Processo de Revisão e Atualização

Fluxo de Trabalho Sugerido:

1. **Auditoria Automática** (mensal)
   - Execução das ferramentas DROID/JHOVE
   - Consolidação dos dados no banco central
2. **Análise Técnica** (trimestral)
   - Revisão de formatos não suportados
   - Avaliação de riscos de obsolescência
   - Identificação de necessidades de migração
3. **Revisão Estratégica** (anual)
   - Atualização da política de formatos
   - Planejamento de migrações necessárias
   - Avaliação de novas tecnologias

4. Indicadores e Dashboards

KPIs Principais:

- Taxa de cobertura: (Formatos suportados / Total de formatos) × 100
- Taxa de crescimento de formatos: Variação percentual entre auditorias
- Índice de obsolescência: Percentual de arquivos em formatos de alto risco
- Tempo médio para inclusão de novos formatos

Visualizações Recomendadas:

- Gráfico de tendência temporal do número de formatos
- Mapa de calor de distribuição de formatos por coleção

- Alertas para formatos críticos
- Timeline de migrações realizadas e planejadas

5. Ferramentas e Tecnologias

Para Implementação:

- **Banco de dados**: PostgreSQL ou MongoDB para armazenar metadados de formatos
- **Orquestração**: Apache Airflow ou cron jobs para automatizar auditorias
- **Dashboards**: Grafana, Power BI ou Tableau para visualização
- **Alertas**: Sistema de notificações (email, Slack) para formatos críticos

Ferramentas Open Source Recomendadas:

- DROID (The National Archives, UK)
- JHOVE (Harvard University Library)
- Apache Tika (identificação de tipos de arquivo)
- PRONOM (registro de formatos de arquivo)

6. Documentação e Governança

- Crie uma política formal de preservação digital
- Documente procedimentos de auditoria
- Estabeleça critérios claros para avaliação de risco
- Defina responsáveis por cada etapa do processo
- Mantenha um registro histórico de todas as decisões

7. Plano de Ação Inicial

Fase 1 (1-2 meses): Levantamento inicial

- Inventário completo do acervo
- Instalação e configuração das ferramentas
- Criação do banco de dados de formatos

Fase 2 (2-3 meses): Automação

- Desenvolvimento de scripts de auditoria
- Configuração de processos automatizados
- Criação de dashboards básicos

Fase 3 (Contínua): Operação e melhoria

- Execução regular das auditorias
- Análise de tendências
- Refinamento do processo

Essa abordagem permite começar de forma simples e escalar conforme necessário, sempre mantendo visibilidade sobre o estado tecnológico do seu acervo digital.

Implementação Prática com DROID e JHOVE

Vou mostrar como configurar e automatizar essas ferramentas:

1. DROID - Identificação de Formatos

Instalação e Configuração

Download e Setup:

```
# Download do DROID
wget https://github.com/digital-preservation/droid/releases/download/6.9.9/droid-binary-6.9.9-bin.zip

# Descompactar
unzip droid-binary-6.9.9-bin.zip -d /opt/droid

# Tornar executável
chmod +x /opt/droid/droid.sh
```

Uso via Interface Gráfica

1. Execute: `./droid.sh`
2. Crie um novo perfil
3. Adicione diretórios para varredura
4. Execute a análise
5. Exporte relatórios (CSV, PDF, DROID Report)

Automação via Linha de Comando

Script básico de varredura:

```
#!/bin/bash
# scan_droid.sh

# Variáveis
DROID_HOME="/opt/droid"
REPOSITORY_PATH="/caminho/para/seu/repositorio"
OUTPUT_DIR="/var/droid/reports"
DATE=$(date +%Y%m%d_%H%M%S)
PROFILE_NAME="scan_${DATE}.droid"
REPORT_NAME="report_${DATE}.csv"

# Criar perfil e executar varredura
$DROID_HOME/droid.sh -a "$REPOSITORY_PATH" -p "$OUTPUT_DIR/$PROFILE_NAME"

# Gerar relatório CSV
$DROID_HOME/droid.sh -p "$OUTPUT_DIR/$PROFILE_NAME" -e "$OUTPUT_DIR/$REPORT_NAME"

# Gerar relatório detalhado
$DROID_HOME/droid.sh -p "$OUTPUT_DIR/$PROFILE_NAME" -r "$OUTPUT_DIR/detailed_${DATE}.pdf"

echo "Varredura concluída: $OUTPUT_DIR/$REPORT_NAME"
```

Tornar executável:

```
chmod +x scan_droid.sh
```

Análise dos Resultados do DROID

Script Python para processar relatórios CSV:

```
#!/usr/bin/env python3
# analyze_droid_report.py

import pandas as pd
import json
```

```
from datetime import datetime
from collections import Counter

def analyze_droid_report(csv_file):
    """Analisa relatório CSV do DROID"""

    # Ler CSV
    df = pd.read_csv(csv_file)

    # Análise de formatos
    formats = df['FORMAT_NAME'].dropna()
    format_counts = Counter(formats)

    # Análise por extensão
    extensions = df['EXT'].dropna()
    ext_counts = Counter(extensions)

    # Estatísticas gerais
    stats = {
        'data_analise': datetime.now().isoformat(),
        'total_arquivos': len(df),
        'total_formatos_unicos': len(format_counts),
        'total_extensoes_unicas': len(ext_counts),
        'top_10_formatos': dict(format_counts.most_common(10)),
        'top_10_extensoes': dict(ext_counts.most_common(10)),
        'arquivos_sem_identificacao': len(df[df['FORMAT_NAME'].isna()]),
        'tamanho_total_mb': df['SIZE'].sum() / (1024 * 1024)
    }

    # Identificar formatos em risco
    formatos_obsoletos = identify_obsolete_formats(format_counts)
    stats['formatos_obsoletos'] = formatos_obsoletos

    # Salvar análise
    output_file = f"analysis_{datetime.now().strftime('%Y%m%d_%H%M%S')}.json"
    with open(output_file, 'w') as f:
        json.dump(stats, f, indent=2)

    print(f"Análise salva em: {output_file}")
```

```
    return stats

def identify_obsolete_formats(format_counts):
    """Identifica formatos potencialmente obsoletos"""

    # Lista de formatos conhecidos como obsoletos ou em risco
    obsolete_keywords = [
        'WordPerfect', 'Lotus', 'dBase', 'PageMaker',
        'Harvard Graphics', 'QuarkXPress'
    ]

    obsolete = {}
    for format_name, count in format_counts.items():
        for keyword in obsolete_keywords:
            if keyword.lower() in format_name.lower():
                obsolete[format_name] = count
                break

    return obsolete

if __name__ == "__main__":
    import sys
    if len(sys.argv) < 2:
        print("Uso: python analyze_droid_report.py <arquivo.csv>")
        sys.exit(1)

    analyze_droid_report(sys.argv[1])
```

2. JHOVE - Validação de Integridade

Instalação

```
# Download do JHOVE
wget http://software.openpreservation.org/rel/jhove-latest.jar

# Configurar PATH
export JHOVE_HOME=/opt/jhove
export PATH=$PATH:$JHOVE_HOME/bin
```

Script de Validação Automatizada

```
#!/bin/bash
# validate_jhove.sh

JHOVE_HOME="/opt/jhove"
REPOSITORY_PATH="/caminho/para/seu/repositorio"
OUTPUT_DIR="/var/jhove/reports"
DATE=$(date +%Y%m%d_%H%M%S)
REPORT_FILE="$OUTPUT_DIR/jhove_report_${DATE}.xml"

# Criar diretório de saída
mkdir -p "$OUTPUT_DIR"

# Executar JHOVE recursivamente
$JHOVE_HOME/bin/jhove -h XML -o "$REPORT_FILE" -kr "$REPOSITORY_PATH"

echo "Validação concluída: $REPORT_FILE"

# Análise básica do relatório
grep -c "Status: Well-Formed and valid" "$REPORT_FILE" > "$OUTPUT_DIR/valid_count_${DATE}.txt"
grep -c "Status: Not well-formed" "$REPORT_FILE" > "$OUTPUT_DIR/invalid_count_${DATE}.txt"
```

Script Python para Análise do JHOVE

```
#!/usr/bin/env python3
# analyze_jhove_report.py

import xml.etree.ElementTree as ET
import json
from datetime import datetime
from collections import defaultdict

def analyze_jhove_report(xml_file):
    """Analisa relatório XML do JHOVE"""

    tree = ET.parse(xml_file)
    root = tree.getroot()
```

```
# Namespace do JHOVE
ns = {'jhove': 'http://hul.harvard.edu/ois/xml/ns/jhove'}

results = {
    'data_analise': datetime.now().isoformat(),
    'total_arquivos': 0,
    'validos': 0,
    'invalidos': 0,
    'bem_formados': 0,
    'erros_por_formato': defaultdict(list),
    'formatos_analisados': defaultdict(int)
}

# Processar cada arquivo
for repInfo in root.findall('.//jhove:repInfo', ns):
    results['total_arquivos'] += 1

    # Obter status
    status = repInfo.find('.//jhove:status', ns)
    formato = repInfo.find('.//jhove:format', ns)
    filepath = repInfo.get('uri', 'unknown')

    if status is not None:
        status_text = status.text

        if 'valid' in status_text.lower():
            results['validos'] += 1
        else:
            results['invalidos'] += 1

            # Coletar mensagens de erro
            messages = repInfo.findall('.//jhove:message', ns)
            for msg in messages:
                error_msg = msg.text
                formato_nome = formato.text if formato is not None else 'unknown'
                results['erros_por_formato'][formato_nome].append({
                    'arquivo': filepath,
                    'erro': error_msg
                })
```

```
            if 'well-formed' in status_text.lower():
                results['bem_formados'] += 1

        # Contar formatos
        if formato is not None:
            results['formatos_analisados'][formato.text] += 1

    # Converter defaultdict para dict normal
    results['erros_por_formato'] = dict(results['erros_por_formato'])
    results['formatos_analisados'] = dict(results['formatos_analisados'])

    # Calcular percentuais
    if results['total_arquivos'] > 0:
        results['percentual_validos'] = (results['validos'] / results['total_arquivos']) * 100
        results['percentual_invalidos'] = (results['invalidos'] / results['total_arquivos']) *
100

    # Salvar análise
    output_file = f"jhove_analysis_{datetime.now().strftime('%Y%m%d_%H%M%S')}.json"
    with open(output_file, 'w') as f:
        json.dump(results, f, indent=2)

    print(f"Análise JHOVE salva em: {output_file}")
    print(f"Total de arquivos: {results['total_arquivos']}")
    print(f"Válidos: {results['validos']} ({results.get('percentual_validos', 0):.2f}%)")
    print(f"Inválidos: {results['invalidos']} ({results.get('percentual_invalidos',
0):.2f}%)")

    return results

if __name__ == "__main__":
    import sys
    if len(sys.argv) < 2:
        print("Uso: python analyze_jhove_report.py <arquivo.xml>")
        sys.exit(1)

    analyze_jhove_report(sys.argv[1])
```

3. Integração e Automação Completa

Script Master de Orquestração

```
#!/bin/bash
# master_preservation_scan.sh

set -e

# Configurações
BASE_DIR="/var/preservation"
REPOSITORY="/caminho/para/repositorio"
DROID_SCRIPT="/opt/scripts/scan_droid.sh"
JHOVE_SCRIPT="/opt/scripts/validate_jhove.sh"
PYTHON_ANALYZE="/opt/scripts/analyze_droid_report.py"
JHOVE_ANALYZE="/opt/scripts/analyze_jhove_report.py"
LOG_FILE="$BASE_DIR/logs/scan_$(date +%Y%m%d_%H%M%S).log"

# Criar estrutura de diretórios
mkdir -p "$BASE_DIR"/{logs,reports,droid,jhove,database}

echo "=== Iniciando Varredura de Preservação Digital ===" | tee -a "$LOG_FILE"
echo "Data: $(date)" | tee -a "$LOG_FILE"

# 1. Executar DROID
echo "--- Executando DROID ---" | tee -a "$LOG_FILE"
$DROID_SCRIPT 2>&1 | tee -a "$LOG_FILE"

# Encontrar último relatório DROID
LATEST_DROID=$(ls -t /var/droid/reports/report_*.csv | head -1)

# 2. Analisar relatório DROID
echo "--- Analisando relatório DROID ---" | tee -a "$LOG_FILE"
python3 $PYTHON_ANALYZE "$LATEST_DROID" 2>&1 | tee -a "$LOG_FILE"

# 3. Executar JHOVE
echo "--- Executando JHOVE ---" | tee -a "$LOG_FILE"
$JHOVE_SCRIPT 2>&1 | tee -a "$LOG_FILE"
```

```
# Encontrar último relatório JHOVE
LATEST_JHOVE=$(ls -t /var/jhove/reports/jhove_report_*.xml | head -1)

# 4. Analisar relatório JHOVE
echo "--- Analisando relatório JHOVE ---" | tee -a "$LOG_FILE"
python3 $JHOVE_ANALYZE "$LATEST_JHOVE" 2>&1 | tee -a "$LOG_FILE"

# 5. Consolidar resultados
echo "--- Consolidando resultados ---" | tee -a "$LOG_FILE"
python3 /opt/scripts/consolidate_reports.py 2>&1 | tee -a "$LOG_FILE"

echo "=== Varredura Concluída ===" | tee -a "$LOG_FILE"
```

Script de Consolidação

```
#!/usr/bin/env python3
# consolidate_reports.py

import json
import glob
from datetime import datetime
import sqlite3

def consolidate_reports():
    """Consolida análises do DROID e JHOVE em banco de dados"""

    # Conectar ao banco de dados
    conn = sqlite3.connect('/var/preservation/database/preservation.db')
    cursor = conn.cursor()

    # Criar tabelas se não existirem
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS scans (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            data_scan TIMESTAMP,
            total_arquivos INTEGER,
            total_formatos INTEGER,
            arquivos_validos INTEGER,
            arquivos_invalidos INTEGER,
```

```
        tamanho_total_mb REAL
    )
''')

cursor.execute('''
    CREATE TABLE IF NOT EXISTS formatos (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        scan_id INTEGER,
        nome_formato TEXT,
        quantidade INTEGER,
        FOREIGN KEY (scan_id) REFERENCES scans(id)
    )
''')

# Encontrar últimas análises
latest_droid = max(glob.glob('analysis_*.json'), key=lambda x: x)
latest_jhove = max(glob.glob('jhove_analysis_*.json'), key=lambda x: x)

# Carregar dados
with open(latest_droid) as f:
    droid_data = json.load(f)

with open(latest_jhove) as f:
    jhove_data = json.load(f)

# Inserir scan
cursor.execute('''
    INSERT INTO scans (data_scan, total_arquivos, total_formatos,
                       arquivos_validos, arquivos_invalidos, tamanho_total_mb)
    VALUES (?, ?, ?, ?, ?, ?)
''', (
    datetime.now(),
    droid_data['total_arquivos'],
    droid_data['total_formatos_unicos'],
    jhove_data['validos'],
    jhove_data['invalidos'],
    droid_data['tamanho_total_mb']
))

scan_id = cursor.lastrowid
```

```
    # Inserir formatos
    for formato, qtd in droid_data['top_10_formatos'].items():
        cursor.execute('''
            INSERT INTO formatos (scan_id, nome_formato, quantidade)
            VALUES (?, ?, ?)
        ''', (scan_id, formato, qtd))

    conn.commit()
    conn.close()

    print(f"Dados consolidados no banco de dados (Scan ID: {scan_id})")

if __name__ == "__main__":
    consolidate_reports()
```

4. Agendamento com Cron

```
# Editar crontab
crontab -e

# Adicionar linha para execução mensal (dia 1 às 2h da manhã)
0 2 1 * * /opt/scripts/master_preservation_scan.sh

# Ou para execução semanal (toda segunda às 3h da manhã)
0 3 * * 1 /opt/scripts/master_preservation_scan.sh
```

5. Dashboard de Visualização

```
#!/usr/bin/env python3
# generate_dashboard.py

import sqlite3
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime

def generate_dashboard():
    """Gera dashboard com métricas de preservação"""
```

```
conn = sqlite3.connect('/var/preservation/database/preservation.db')

# Evolução temporal
df_scans = pd.read_sql_query('''
    SELECT data_scan, total_formatos, arquivos_validos,
           arquivos_invalidos, tamanho_total_mb
    FROM scans
    ORDER BY data_scan
''', conn)

# Criar gráficos
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Gráfico 1: Evolução de formatos
axes[0, 0].plot(df_scans['data_scan'], df_scans['total_formatos'], marker='o')
axes[0, 0].set_title('Evolução do Número de Formatos')
axes[0, 0].set_xlabel('Data')
axes[0, 0].set_ylabel('Total de Formatos')
axes[0, 0].grid(True)

# Gráfico 2: Validade dos arquivos
axes[0, 1].plot(df_scans['data_scan'], df_scans['arquivos_validos'],
                label='Válidos', marker='o')
axes[0, 1].plot(df_scans['data_scan'], df_scans['arquivos_invalidos'],
                label='Inválidos', marker='x')
axes[0, 1].set_title('Integridade dos Arquivos')
axes[0, 1].set_xlabel('Data')
axes[0, 1].set_ylabel('Quantidade')
axes[0, 1].legend()
axes[0, 1].grid(True)

# Gráfico 3: Top formatos do último scan
df_formatos = pd.read_sql_query('''
    SELECT nome_formato, quantidade
    FROM formatos
    WHERE scan_id = (SELECT MAX(id) FROM scans)
    ORDER BY quantidade DESC
    LIMIT 10
''', conn)
```

```
    axes[1, 0].barh(df_formatos['nome_formato'], df_formatos['quantidade'])
    axes[1, 0].set_title('Top 10 Formatos (Último Scan)')
    axes[1, 0].set_xlabel('Quantidade')

    # Gráfico 4: Tamanho total
    axes[1, 1].plot(df_scans['data_scan'], df_scans['tamanho_total_mb'],
                    marker='o', color='green')
    axes[1, 1].set_title('Evolução do Tamanho Total (MB)')
    axes[1, 1].set_xlabel('Data')
    axes[1, 1].set_ylabel('Tamanho (MB)')
    axes[1, 1].grid(True)

    plt.tight_layout()

plt.savefig(f'/var/preservation/reports/dashboard_{datetime.now().strftime("%Y%m%d")}.png',
                dpi=300)
    print("Dashboard gerado com sucesso!")

    conn.close()

if __name__ == "__main__":
    generate_dashboard()
```

Revision #1
Created 23 October 2025 19:45:43 by Rondineli G. Saad
Updated 23 October 2025 19:55:41 by Rondineli G. Saad