

# Backlog – Próxima versão (Projeto Pr. Albino Marks)

**Status:** versão atual congelada. Este documento reúne pendências e melhorias já combinadas para a próxima release.

---

## Checklist — Próximo Release (Pr. Albino Marks)

UI/UX da página do artigo

Layout 2 colunas ( $\geq 1024\text{px}$ ): conteúdo + lateral (destaques/ads).

Realce suave no corpo do texto (amarelo compatível com a home).

Destaques (3 itens) e slot para anúncio na sidebar.

Botões de navegação maiores que os de personalização (agrupados à direita).

Validação e robustez de uploads

arquivo\_word aceita somente .docx (validator + mensagem amigável).

Try/except no parser de DOCX (BadZipFile → ValidationError no form; nunca 500).

View /pdf/ segura: se sem arquivo → 404, se com arquivo → redirect para arquivo\_pdf.url.

Pipeline de PDF (local → S3 → BD Remoto)

Converter DOCX→PDF em background (ex.: LibreOffice headless).

Salvar PDF no storage padrão (S3) e atualizar Artigo.arquivo\_pdf.

Evitar reprocessar se já houver PDF idêntico (hash/mtime).

Logar falhas e exibir aviso no admin.

Autor duplicado no HTML

Remover autor do HTML gerado (não inserir no conteudo\_html).

Garantir exibição via template (visualizar\_artigo.html) ou views.py com o campo autor.

Testar casos: sem autor / com autor / múltiplos autores (se aplicável).

S3 / mídia

Confirmar AWS\_QUERYSTRING\_AUTH=True e AWS\_QUERYSTRING\_EXPIRE=86400.

MEDIA\_URL apontando para bucket/CDN (sem sobrescrever por "/media/").

IAM com s3::ListBucket, s3::GetObject, s3::PutObject, s3::DeleteObject apenas no bucket do projeto.

Git & deploy

.gitignore: media/ e staticfiles/ ignorados; índice limpo (git rm -r --cached).

Hooks: core.hooksPath apontando para pasta vazia ou desativado; usar --no-verify só em emergência.

Pós-deploy (Railway):

```
railway run "python manage.py migrate --noinput"
```

```
railway run "python manage.py collectstatic --noinput"
```

Smoke tests: acessar /admin/, publicar 1 artigo com imagem + DOCX; verificar slug, imagem (URL assinada), PDF e layout.

Testes rápidos (comandos úteis)

Verificar storage ativo:

```
from django.conf import settings; print(settings.DEFAULT_FILE_STORAGE, settings.MEDIA_URL)
```

Checar existência de PDFs no storage:

```
from django.core.files.storage import default_storage
```

```
from A_Lei_no_NT.models import Artigo
```

```
faltando=[(a.id,a.slug) for a in Artigo.objects.exclude(arquivo_pdf="") if not default_storage.exists(a.arquivo_pdf.name)]
```

```
len(faltando), faltando[:5]
```

---

**Observação:** manter esta lista viva – novos itens podem ser acrescentados conforme surgirem durante o trabalho no Projeto 21.

## Pipeline Híbrido DOCX→PDF (Local → Produção) — Próxima Versão

### Objetivo

- Converter DOCX localmente e publicar o PDF no storage de produção, sem conversão automática no servidor.

### Escopo

- PDF será enviado apenas quando gerado localmente; Artigo.arquivo\_pdf é atualizado após upload.
- Sem LibreOffice/docx2pdf em produção; sem conversão em models/signals/admin.

### Fluxo Operacional

- 1) Origem: usar DOCX do artigo (local ou baixar do storage).
- 2) Conversão local: Microsoft Word (docx2pdf) ou LibreOffice headless.
- 3) Idempotência: calcular hash (SHA-256) do PDF; pular upload se não houver mudança.
- 4) Upload: enviar para storage de produção (ex.: S3) em pdfs/artigos/<slug>.pdf.
- 5) Atualização: setar Artigo.arquivo\_pdf para o novo arquivo e salvar.
- 6) Log: registrar sucesso/falha; nunca derrubar o app por erro de conversão.

### Implementação (itens de ação)

- [ ] Remover lógica de conversão do models/signals/admin (já feito nesta release).
- [ ] Criar comando/endpoint apenas para atualizar arquivo\_pdf com segurança (autenticado).
- [ ] Criar script PowerShell (PS1) orquestrador local:
  - Baixar DOCX se necessário
  - Converter para PDF
  - Calcular hash e comparar
  - Upload para storage de PRODUÇÃO (com credenciais seguras)
  - Chamar endpoint/management para atualizar o Artigo
- [ ] Flags em settings: ENABLE\_DOCX\_PDF\_CONVERSION = False (produção); True somente em dev local.
- [ ] Templates: proteger acessos .url com “if campo” (evitar 500 quando vazio).
- [ ] Logging: nível WARNING+ no servidor; sem flood de logs.
- [ ] Segurança: usar variáveis de ambiente e perfil restrito no S3 (Get/Put/Delete no bucket do projeto).

### Boas Práticas

- Guardar conversores em Apenas\_Local/ (fora de apps instalados).
- Commands “devtools” habilitados só com DEBUG=True.
- “Dry-run” no PS1 para simular antes de publicar.
- Documentar passo a passo e pontos de rollback.