Destaques da Versão Atual

Resumo dos pontos-chave já implementados e pendências priorizadas.

• Favicon e estáticos

– Favicon servido por {% static 'favicon.ico' %} + rota opcional /favicon.ico (RedirectView).

– Remoção de duplicatas em static/ e limpeza de static/admin/.

• Templates e URLs

– Correções de caminhos e TemplateDoesNotExist/TemplateSyntaxError.

– Sem strings/triple-quotes dentro de urlpatterns.

• Formulário de Artigo (Criar/Editar)

– Página unificada de formulário com Bootstrap 5 e validações.

– Guardas para campos de arquivo (condicionais antes de .url).

• Conversão DOCX→PDF

– Conversão desativada em produção (sem LibreOffice/docx2pdf).

– Próxima versão: pipeline híbrido (converter local e publicar no storage remoto).

Pipeline Híbrido DOCX→PDF (Local → Produção) — Próxima Ve

Objetivo

• Converter DOCX localmente e publicar o PDF no storage de produção.
• Sem conversão automática no servidor (models/signals/admin).

Fluxo Operacional

• Origem: usar DOCX do artigo (local ou baixar do storage).
• Conversão local: Microsoft Word (docx2pdf) ou LibreOffice headless.
• Idempotência: calcular hash (SHA-256) do PDF; pular upload se não houver mudança.
• Upload: enviar para storage remoto (ex.: S3) em pdfs/artigos/<slug>.pdf.
• Atualização: setar Artigo.arquivo_pdf e salvar.
• Logs: registrar sucesso/falha; não derrubar o app por erro de conversão.

Checklist de Implementação

- [ ] Remover/confirmar remoção de conversão automática no servidor.
- [ ] Criar comando/endpoint para atualizar arquivo_pdf de modo seguro.
- [ ] PS1 orquestrador local: baixar DOCX (se preciso), converter, hash, upload, atualizar artigo.
- [ ] Flags de settings: ENABLE_DOCX_PDF_CONVERSION=False em produção.
- [ ] Templates com guardas para .url em campos de arquivo.
- [ ] Logging WARNING+ e sem access log do Gunicorn.
- [ ] Política de acesso no bucket (Get/Put/Delete restritos).
- [ ]

Boas Práticas

• Conversores e helpers em Apenas_Local/ ou app devtools (apenas DEBUG=True).
• Modo dry-run no PS1 antes de publicar.
• Documentar passos e rollback.

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 (≥1024px): 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.