Manual prático

Emacs + Org Mode + lazyblorg no Windows/WSL

Do zero até um blog estático local, com comandos, solução de erros e customização do template.

Este manual foi escrito para um uso direto: instalar as ferramentas, criar posts em Org Mode, gerar HTML com lazyblorg, abrir localmente no navegador e entender por que a primeira página aparece com a identidade do Karl Voit até você editar o template.

Resumo brutal: o lazyblorg funciona, mas não é ferramenta "bonitinha para iniciante". É poderoso, mas cheio de pressupostos do ambiente do autor. O jeito certo é aceitar isso e trabalhar com método.

Ambiente usado neste guia: Windows com WSL/Linux, GNU Emacs, Org Mode, Python 3, uv, pandoc e lazyblorg.

Sumário

1. O que você está montando

Você está criando um blog estático. Isso significa que não existe painel de administração, banco de dados, login, WordPress ou servidor dinâmico. Você escreve texto em arquivos .org, roda um comando, e o lazyblorg gera arquivos .html, .css, tags, feeds e páginas de arquivo.

```
Org Mode (.org)  ->  lazyblorg  ->  HTML/CSS estático  ->  navegador / GitHub Pages / servidor
```

Esse modelo é ótimo para blogs pessoais, notas técnicas, artigos e páginas leves. É péssimo se você espera uma interface visual moderna, edição por navegador ou temas prontos com um clique.

> Vantagem: tudo fica em texto puro e HTML estático. Fácil de versionar, copiar, hospedar e preservar.
>
> Desvantagem: você precisa mexer com terminal, arquivos de configuração e templates.
>
> Regra de ouro: primeiro faça funcionar feio. Depois deixe bonito.

2. Conceitos rápidos

Emacs

Emacs é um editor de texto extremamente extensível. A interface parece antiga porque ela é antiga mesmo; isso não é defeito visual, é fóssil funcional. Ele é usado aqui porque o Org Mode nasceu nele e funciona melhor nele.

Org Mode

Org Mode é um formato e um modo de edição para notas, tarefas, documentos e estrutura hierárquica em texto puro. Um arquivo Org usa extensão .org e headings com asteriscos.

```
* Projeto
** TODO Escrever rascunho
** DONE Publicar post
   CLOSED: [2026-05-14 Thu 12:00]
```

lazyblorg

lazyblorg é um gerador de blog estático que lê posts em Org Mode e gera HTML5. Segundo o README do projeto, ele foi feito para permitir esforço mínimo na criação de posts: escrever um heading, adicionar tag, ID, marcar como DONE e rodar o gerador.

Ponto importante: ele é explicitamente um projeto pessoal, “works-for-me”. Isso explica a sensação de que o software funciona, mas pressupõe que você esteja disposto a adaptar o fluxo do autor ao seu ambiente.

3. Instalação no Windows com WSL

Não rode isso direto no Windows nativo. O README do lazyblorg informa que ele não funciona nativamente no Microsoft Windows e sugere que Windows com WSL parece promissor. Então o caminho correto é WSL com Debian/Ubuntu.

3.1 Instalar o WSL

No PowerShell como administrador:

```
wsl --install
```

Reinicie se o Windows pedir. Depois abra o app Ubuntu/Debian pelo menu iniciar.

3.2 Instalar dependências Linux

```
sudo apt update
sudo apt install git python3 python3-pip python3-venv pandoc libgl1 libglib2.0-0 pipx
```

O pacote pandoc ajuda nas conversões de Org/HTML. libgl1 e libglib2.0-0 evitam erros do OpenCV/cv2 no WSL. pipx instala ferramentas Python sem quebrar o Python do sistema.

3.3 Instalar uv com pipx

Não use pip3 install uv global se o sistema reclamar de externally-managed-environment. Isso é proteção do Debian/Ubuntu moderno. Use pipx:

```
pipx ensurepath
# feche e abra o terminal se necessário
pipx install uv
uv --version
```

Se o comando uv não aparecer:

```
export PATH="$HOME/.local/bin:$PATH"
uv --version
```

4. Primeiro uso do Emacs e do Org Mode

4.1 Atalhos essenciais

No Emacs, C significa Ctrl e M significa Alt. Os comandos são sequenciais, não simultâneos. C-x C-f significa Ctrl+X e depois Ctrl+F.

Atalho	O que faz
C-x C-f	Abrir ou criar arquivo
C-x C-s	Salvar
C-x C-c	Sair do Emacs
C-g	Cancelar comando atual; botão de pânico
M-x	Executar comando pelo nome
TAB	Expandir/colapsar heading no Org Mode
C-c C-t	Alternar estado TODO/DONE

4.2 Criar um arquivo .org

```
C-x C-f
~/teste.org
Enter
```

Dentro do arquivo:

```
* Minha primeira nota
** TODO Aprender Org Mode
   DEADLINE: <2026-05-20>

** DONE Instalar Emacs
```

Salve com C-x C-s. Se o arquivo termina em .org, o Emacs deve entrar em Org Mode automaticamente.

5. Instalando o lazyblorg

5.1 Clonar o repositório

```
mkdir -p ~/projetos
cd ~/projetos
git clone https://github.com/novoid/lazyblorg.git
cd lazyblorg
```

5.2 Rodar o help

```
uv --project . run ./lazyblorg.py --help
```

Se aparecer a ajuda do programa, a instalação base está OK. Avisos sobre pastas de imagem do autor podem aparecer. Eles são feios, mas não são fatais.

5.3 Erro comum: libGL.so.1

Se aparecer:

```
ImportError: libGL.so.1: cannot open shared object file: No such file or directory
```

Resolva com:

```
sudo apt update
sudo apt install libgl1 libglib2.0-0
```

6. Criando o blog mínimo

6.1 Criar pastas

```
mkdir -p ~/meu-blog/org
mkdir -p ~/meu-blog/public
```

6.2 Criar o arquivo de posts

```
cat > ~/meu-blog/org/blog.org <<'EOF'
#+TITLE: Blog do Pablo

* Posts

** DONE Meu primeiro post com Org Mode :blog:orgmode:
CLOSED: [2026-05-14 Thu 12:00]
:PROPERTIES:
:ID: 2026-05-14-meu-primeiro-post
:CREATED: [2026-05-14 Thu 11:50]
:END:
:LOGBOOK:
- State "DONE" from "NEXT" [2026-05-14 Thu 12:00]
:END:

Este é meu primeiro post usando Org Mode com lazyblorg.

A ideia é simples:

- escrever em texto puro
- marcar posts com a tag =blog=
- gerar HTML
- publicar arquivos estáticos

*** Uma seção dentro do post

Aqui vai mais texto do artigo.
EOF
```

Detalhe importante: o post está em **, não em *. O heading * Posts é só uma categoria. O lazyblorg reconheceu melhor esse formato porque é parecido com o exemplo oficial.

6.3 Anatomia de um post válido

:blog: marca o heading como post de blog.

:orgmode: é uma tag extra. Na teoria não é obrigatória; na prática ajuda a evitar a tag cloud vazia nesse setup.

:ID: precisa ser único. Use data + slug: 2026-05-14-meu-primeiro-post.

DONE indica que o post está publicado.

CLOSED e LOGBOOK registram a data de publicação.

7. Gerando HTML e abrindo localmente

7.1 Primeiro build

```
cd ~/projetos/lazyblorg

rm -rf ~/meu-blog/public/*
rm -f ~/meu-blog/metadata.pk ~/meu-blog/lazyblorg-log.org

uv --project . run ./lazyblorg.py   --orgfiles ~/meu-blog/org/blog.org ~/projetos/lazyblorg/
templates/blog-format.org   --targetdir ~/meu-blog/public   --previous-metadata ~/meu-blog/metadata.
pk   --new-metadata ~/meu-blog/metadata.pk   --logfile ~/meu-blog/lazyblorg-log.org   --ignore-
missing-ids
```

Se der certo, você verá algo como “Generated … articles” e a pasta public terá index.html, feeds, tags e diretórios por ano.

7.2 Conferir arquivos gerados

```
ls -la ~/meu-blog/public
```

7.3 Servir localmente

```
cd ~/meu-blog/public
python3 -m http.server 8000
```

No navegador do Windows, abra:

```
http://localhost:8000
```

Se aparecer um site com “public voit” mas também seu post no meio, isso é sucesso parcial: o conteúdo foi gerado, mas o template ainda é do autor.

8. Customizando a página para deixar de ser “public voit”

O lazyblorg usa o arquivo blog-format.org como template. O arquivo padrão vem com textos, links e identidade do Karl Voit. Se você usa o template original diretamente, seu blog fica com a cara dele.

8.1 Copiar o template para seu blog

```
cp ~/projetos/lazyblorg/templates/blog-format.org ~/meu-blog/org/blog-format.org
```

8.2 Procurar textos do autor

```
grep -n "Karl\|Voit\|public voit\|homepage" ~/meu-blog/org/blog-format.org
```

8.3 Editar o template

```
nano ~/meu-blog/org/blog-format.org
```

Troque textos como:

```
public voit
Karl Voit
This is the homepage of Karl Voit.
```

Por algo seu:

```
Blog do Pablo
Pablo Murad
Este é o meu blog pessoal.
```

8.4 Gerar usando seu template editado

```
cd ~/projetos/lazyblorg

rm -rf ~/meu-blog/public/*
rm -f ~/meu-blog/metadata.pk ~/meu-blog/lazyblorg-log.org

uv --project . run ./lazyblorg.py   --orgfiles ~/meu-blog/org/blog.org ~/meu-blog/org/blog-format.
org   --targetdir ~/meu-blog/public   --previous-metadata ~/meu-blog/metadata.pk   --new-metadata ~/
meu-blog/metadata.pk   --logfile ~/meu-blog/lazyblorg-log.org   --ignore-missing-ids
```

Depois suba novamente o servidor local e recarregue o navegador.

9. Criando novos posts

Você pode adicionar novos posts no mesmo arquivo ~/meu-blog/org/blog.org. Use sempre o mesmo padrão.

```
** DONE Segundo post :blog:linux:
CLOSED: [2026-05-15 Fri 10:00]
:PROPERTIES:
:ID: 2026-05-15-segundo-post
:CREATED: [2026-05-15 Fri 09:50]
:END:
:LOGBOOK:
- State "DONE" from "NEXT" [2026-05-15 Fri 10:00]
:END:

Texto do segundo post.
```

Depois rode novamente o comando de geração. O lazyblorg reprocessa o site inteiro a cada run; isso é normal.

9.1 Modelo mental correto

Editar .org não publica nada sozinho.

Salvar o arquivo .org não gera HTML.

Você precisa rodar o comando do lazyblorg para regenerar o site.

Você precisa copiar/subir a pasta public para publicar na internet.

10. Publicando na internet

10.1 GitHub Pages - forma simples

Crie um repositório no GitHub, por exemplo meu-blog. Depois envie o conteúdo da pasta public.

```
cd ~/meu-blog/public
git init
git add .
git commit -m "Primeira versão do blog"
git branch -M main
git remote add origin https://github.com/SEU_USUARIO/meu-blog.git
git push -u origin main
```

No GitHub: Settings -> Pages -> Deploy from branch -> main. Depois aguarde o endereço ficar disponível.

10.2 Servidor próprio/VPS

Se você tem VPS ou hospedagem com SSH, pode copiar com rsync:

```
rsync -avz --delete ~/meu-blog/public/ usuario@servidor:/var/www/meu-blog/
```

Não rode lazyblorg no servidor se não precisar. Gere localmente e publique só os HTMLs. Mais simples, menos coisa para quebrar.

11. Troubleshooting dos erros que apareceram

11.1 pip3 install uv -> externally-managed-environment

Causa: Debian/Ubuntu não deixa você instalar pacote Python global com pip porque isso pode quebrar o Python do sistema.

```
sudo apt install pipx
pipx ensurepath
pipx install uv
```

Não use --break-system-packages para isso. É pedir para criar problema besta.

11.2 ImportError: libGL.so.1

Causa: OpenCV/cv2 precisa de libGL no ambiente Linux/WSL.

```
sudo apt install libgl1 libglib2.0-0
```

11.3 “no suitable template data could be parsed”

Causa: você rodou lazyblorg sem passar o template blog-format.org no --orgfiles. O programa precisa do seu arquivo de posts e do template.

```
--orgfiles ~/meu-blog/org/blog.org ~/projetos/lazyblorg/templates/blog-format.org
```

11.4 ValueError: max() iterable argument is empty

Causa provável: o lazyblorg não reconheceu nenhum post válido/tag válido, então a tag cloud ficou vazia.

Confirme que o post está em heading de nível 2: ** DONE ...

Confirme que tem tag :blog: e uma tag extra, por exemplo :orgmode:

Confirme que tem :ID:, CLOSED e LOGBOOK.

Confirme que você passou o template no --orgfiles.

11.5 Aparece “public voit” no navegador

Isso não é erro de build. É o template padrão do autor. Copie o template para seu projeto e edite os textos.

```
cp ~/projetos/lazyblorg/templates/blog-format.org ~/meu-blog/org/blog-format.org
nano ~/meu-blog/org/blog-format.org
```

12. Checklist final e comandos de bolso

12.1 Checklist de build

WSL funcionando.

- uv funcionando: uv --version.
- lazyblorg respondendo: uv --project . run ./lazyblorg.py --help.
- Arquivo ~/meu-blog/org/blog.org existe.
- Template copiado e editado em ~/meu-blog/org/blog-format.org.
- Comando de build aponta para blog.org + blog-format.org.
- Pasta ~/meu-blog/public contém index.html.
- Servidor local roda em http://localhost:8000.

12.2 Comando único de rebuild

```
cd ~/projetos/lazyblorg && rm -rf ~/meu-blog/public/* && uv --project . run ./lazyblorg.py   --
orgfiles ~/meu-blog/org/blog.org ~/meu-blog/org/blog-format.org   --targetdir ~/meu-blog/public   --
previous-metadata ~/meu-blog/metadata.pk   --new-metadata ~/meu-blog/metadata.pk   --logfile ~/meu-
blog/lazyblorg-log.org   --ignore-missing-ids
```

12.3 Comando para abrir localmente

```
cd ~/meu-blog/public
python3 -m http.server 8000
```

12.4 Estrutura final recomendada

```
~/meu-blog/
├── org/
│   ├── blog.org
│   └── blog-format.org
├── public/
│   ├── index.html
│   ├── 2026/
│   ├── tags/
│   └── feeds/
├── metadata.pk
└── lazyblorg-log.org
```

13. Fontes e notas

Este manual foi baseado no fluxo prático executado no terminal e na documentação oficial do lazyblorg no GitHub.

- lazyblorg é um gerador de HTML5 estático a partir de Org Mode e usa Python 3.
- O README informa que o projeto não funciona nativamente no Windows, mas WSL parece promissor.
- O fluxo de post recomendado pelo README é: escrever entry, usar tag :blog:, adicionar ID, marcar DONE e ter LOGBOOK com timestamp.
- O próprio README deixa claro que é um projeto pessoal, feito principalmente para o workflow do autor.

URLs úteis:

```
https://github.com/novoid/lazyblorg
https://orgmode.org/
https://www.gnu.org/software/emacs/
```

Conclusão honesta: se você quer só publicar rápido, Hugo/Astro/Jekyll podem ser mais fáceis. Se você quer um fluxo de notas em Org Mode virando blog estático, lazyblorg é interessante - desde que você aceite mexer em template e terminal.