Guia “Colar e Rodar” — Passo a passo

- Trocar os caminhos para as quais estão no seu computador !

Abra 3 janelas do PowerShell e rode cada bloco em uma janela.

Segue o “colar e rodar”. Abra 3 janelas do PowerShell e rode cada bloco em uma janela.

Terminal 1 — API (FastAPI na 8010)

```
cd "C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc\server"
python -m venv .venv .\.venv\Scripts\Activate.ps1 pip
install -r requirements.txt

# FOODS_DB a partir do repo (ajusta automático)
$repo="C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc"
$db=(Get-ChildItem -Recurse -File -Filter alimentos.sqlite $repo | Select-Object -First 1 -
Exp
$env:FOODS_DB = ($db -replace '\\','/')
"FOODS_DB=$($env:FOODS_DB)" | Set-Content -Encoding UTF8 .env

uvicorn app.main:app --host 127.0.0.1 --port 8010
```

Terminal 2 — Caddy (site + proxy /api)

```
@' :8080 {
encode zstd gzip

    # API primeiro     handle_path
/api/* {         reverse_proxy
127.0.0.1:8010     }

    # Site estático (dist do client)     root *
"C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc\client\dist"
try_files {path} /index.html     file_server }
'@ | Set-Content -Encoding UTF8 C:\caddy\Caddyfile

C:\caddy\caddy.exe run --config C:\caddy\Caddyfile --watch
```

Terminal 3 — Túnel público (Cloudflare)

```
cloudflared tunnel --url http://localhost:8080
```

Pronto. Não feche essas três janelas.

Se ainda não tiver a pasta `dist` do cliente, gere antes (uma 4ª janela):

```
cd "C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc\client"
"EXPO_PUBLIC_API=/api" | Set-Content -Encoding UTF8 .env
npx expo export -p web
```

Guia rápido: subir o projeto depois que reiniciar o PC (Windows/PowerShell)

CASO VOCÊ JA TENHA FEITO OS COMANDOS A CIMA !

> Use três janelas do PowerShell:

> Terminal 1 = API (porta 8010)

> Terminal 2 = Caddy (porta 8080)

> Terminal 3 = Cloudflare Tunnel (link público)

Terminal 1 — API (FastAPI na 8010)

```
cd "C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc\server"

# venv (cria se não existir) e ativa if (-not (Test-
Path .\.venv)) { python -m venv .venv }
.\.venv\Scripts\Activate.ps1

# dependências pip install -r
requirements.txt

# define FOODS_DB apontando para o alimentos.sqlite dentro do repo
$repo = "C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc" $db = Get-ChildItem -Recurse -File
-Filter alimentos.sqlite $repo | Select-Object -First 1 -Ex if (-not $db) { Write-Error
"alimentos.sqlite não encontrado em $repo"; break } $env:FOODS_DB = ($db -replace '\\','/')
"FOODS_DB=$($env:FOODS_DB)" | Set-Content -Encoding UTF8 .env

# libera a porta 8010 se algo ficou preso $portPid = (Get-NetTCPConnection -LocalPort 8010 -
ErrorAction SilentlyContinue).OwningProcess if ($portPid) { Stop-Process -Id $portPid -Force
}

# sobe a API
python -m uvicorn app.main:app --host 127.0.0.1 --port 8010 --log-level info
```

Deixe essa janela aberta. A linha final deve mostrar `Uvicorn running on http://127.0.0.1:8010`.

Terminal 2 — Caddy (site + proxy /api)

```
# mata Caddy antigo, se houver
Get-Process caddy -ErrorAction SilentlyContinue | Stop-Process -Force

# escreve o Caddyfile (API primeiro, SPA em seguida)
@' :8080 {
encode zstd gzip

    route {          # /api -> encaminha
para a API          handle_path /api/* {
reverse_proxy 127.0.0.1:8010         }

        # site estático + fallback do SPA         handle {              root *
"C:\Users\roger\Desktop\TrabalhoFinal\TCC-II\tcc\client\dist"
try_files {path} /index.html             file_server         }
    }
}
'@ | Set-Content -Encoding UTF8 C:\caddy\Caddyfile

# inicia e deixa aberto
C:\caddy\caddy.exe run --config C:\caddy\Caddyfile --watch
```

Deixe essa janela aberta. Você deve ver algo como `server running ... :8080`.

Terminal 3 — Cloudflare Tunnel (link público)

```
cloudflared tunnel --url http://localhost:8080
```

Ele vai imprimir uma URL do tipo `https://xxxxx.trycloudflare.com`.

Enquanto essa janela ficar aberta, seu site e a API ficam públicos nesse endereço.

Testes rápidos

Você pode rodar estes testes em qualquer janela separada.

Local

```
# site
Invoke-WebRequest "http://127.0.0.1:8080" | % StatusCode   # deve ser 200

# categorias via Caddy -> JSON
Invoke-RestMethod "http://127.0.0.1:8080/api/v1/categories"

# exemplo de consulta por categoria
$encCat = [System.Uri]::EscapeDataString("Frutas e derivados")
Invoke-RestMethod
"http://127.0.0.1:8080/api/v1/foods?category=$encCat&limit=3&columns=descrip
```

Público (substitua pela sua URL do túnel)

```
$URL = "https://sys-component-trained-march.trycloudflare.com"
Invoke-RestMethod "$URL/api/v1/categories"

$encCat = [System.Uri]::EscapeDataString("Frutas e derivados")
Invoke-RestMethod
"$URL/api/v1/foods?category=$encCat&limit=3&columns=description,energy_kcal,
```

Dica de acentuação no PowerShell (só para visualização no terminal)

Isso não afeta o site, apenas como o terminal imprime:

```
chcp 65001 [Console]::InputEncoding  =
[System.Text.Encoding]::UTF8
[Console]::OutputEncoding = [System.Text.Encoding]::UTF8
```

Acesso pela rede local (opcional)

Se quiser abrir o site para outros dispositivos da sua LAN em `http://SEU_IP:8080`:

```
New-NetFirewallRule -DisplayName "Caddy 8080" -Direction Inbound -Protocol TCP -LocalPort
```

`8080` A API pode continuar só em `127.0.0.1:8010`, que é mais seguro.

Observações

* Mantenha as três janelas abertas: API, Caddy e Cloudflare.
* Se o PC dormir ou reiniciar, repita este guia.
* A URL `trycloudflare.com` pode mudar a cada execução. Para um endereço fixo, useum túnel nomeado com um domínio seu no Cloudflare.

Se quiser, eu te passo um “modo automático” com três scripts `.ps1` e agendamento no Windows para subir tudo sozinho ao logar.

CASO DE ERRO NA HORA DE ACESSAR O SITE FAÇA O SEGUINTE EM CADA TERMINAL

```
ctrl + C e faça os comandos em cada terminal
```

Terminal 1 ```Ctrol + c

```
python -m uvicorn app.main:app --host 127.0.0.1 --port 8010 --log-level debug
```

Terminal 2 ```Ctrol + c

```
C:\caddy\caddy.exe run --config C:\caddy\Caddyfile --watch
```

Terminal 3 ```Ctrol + c

```
``` cloudflared tunnel --url
http://localhost:8080
```
```