

BauzaGPT — Guía completa, estructura del repositorio y scripts listos para desplegar (GitHub Pages + API Node/Express)

Esta guía reúne **todo lo necesario** para poner en marcha el **frontend** con **GitHub Pages** y el **backend** con **Node/Express** (Render/Railway/Heroku o local). Incluye estructura del repo, archivos listos y comandos de despliegue.

0) Estructura final del repositorio

```
bauzagpt.com/
├─ README.md
├─ .gitignore
├─ docs/                # Sitio público (GitHub Pages)
│  ├─ CNAME             # "www.bauzagpt.com"
│  ├─ index.html        # Landing
│  ├─ precios.html
│  ├─ pago.html
│  ├─ login.html
│  ├─ privacidad.html
│  ├─ terminos.html
│  ├─ gracias.html
│  ├─ load-config.js    # Carga de config pública (sin secretos)
│  ├─ config.sample.json # Ejemplo de claves públicas (Firebase)
│  ├─ img/
│  │  └─ ojos-vigilantes.gif
│  └─ sitemap.xml
└─ api/                # Backend Node/Express
   ├─ package.json
   ├─ server.js
   ├─ .env.example
   ├─ src/
   │  ├─ payments/
   │  │  └─ methods.json # Catálogo dinámico de métodos de pago (sin
exponer números)
   │  ├─ routes/
   │  │  ├─ health.js
   │  │  ├─ orders.js
   │  │  ├─ payments.js
   │  │  └─ generate.js
   │  └─ services/
   │     ├─ linkChecker.js
   │     ├─ pdfReport.js
   │     └─ zipPack.js
```

```
|   └─ utils/
|       └─ tokens.js
└─ tmp/ (se crea en runtime)
```

Importante: No subas **secretos** al repo. Las claves públicas de Firebase pueden vivir en `docs/config.json`. Todo lo sensible (tokens/llaves privadas) va en **variables de entorno** del backend.

1) Frontend (`docs/`)

`docs/CNAME`

`www.bauzagpt.com`

`docs/load-config.js`

Archivo **JavaScript puro**. No uses etiquetas `<script>` aquí.


```
(async () => {
  try {
    // Carga configuración pública (por ejemplo, claves públicas de Firebase)
    const res = await fetch('/config.json', { cache: 'no-store' });
    if (!res.ok) throw new Error('No se pudo cargar config.json');
    window.BAUZA_PUBLIC = await res.json(); // disponible en el navegador
  } catch (e) {
    console.error('Error al cargar config:', e);
  }
})();
```

`docs/config.sample.json`

```
{
  "firebase": {
    "apiKey": "TU_API_KEY_PUBLICA",
    "authDomain": "tu-proyecto.firebaseio.com",
    "projectId": "tu-proyecto",
    "storageBucket": "tu-proyecto.appspot.com",
    "messagingSenderId": "000000000000",
    "appId": "1:000000000000:web:XXXXXXXXXXXXXXX"
  },
  "apiBase": "https://TU-BACKEND.onrender.com"
}
```

Producción: sube `` (mismo formato que el *sample*). No coloques secretos del backend aquí.

```

<!doctype html><html lang="es"><head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1"/>
<title>BAUZA GPT – OSINT  Pentest Ético</title>
<meta name="description" content="OSINT legal, pentesting ético y reportes automáticos.">
<link rel="preconnect" href="https://accounts.google.com">
<script src="/load-config.js" defer></script>
<style>
:root{--bg:#0b1220;--panel:#0f172a;--text:#e2e8f0;--muted:#94a3b8;--
accent:#22d3ee;--brand:#00ff7f}
*{box-sizing:border-box}body{margin:0;background:linear-
gradient(180deg,#07111e,#0b1220);color:var(--text);font-family:system-
ui,Segoe UI,Roboto,Ubuntu,sans-serif}
.hero{min-height:100svh;display:grid;place-items:center;padding:24px}
.card{max-width:960px;width:100%;background:rgba(15,23,42,.6);backdrop-
filter:blur(10px);border:1px solid #1e293b;border-radius:18px;padding:
28px;box-shadow:0 10px 35px rgba(0,0,0,.45)}
.badge{display:inline-block;padding:6px 12px;border:1px solid #1e293b;border-
radius:999px;color:var(--muted)}
.h1{font-size:clamp(28px,6vw,48px);line-height:1.1;margin:14px 0;font-weight:
800}
.lead{color:var(--muted);font-size:clamp(14px,3.5vw,18px)}
.grid{display:grid;gap:16px;grid-template-columns:repeat(auto-
fit,minmax(240px,1fr));margin-top:22px}
.tile{background:#0b1120;border:1px solid #1e293b;border-radius:16px;padding:
16px}
.btn{display:inline-block;margin-top:14px;padding:12px 16px;border-radius:
12px;border:1px solid #1e293b;background:#0b1120;color:#e2e8f0;text-
decoration:none;font-weight:700}
.btn.primary{background:linear-
gradient(90deg,#22d3ee,#00ff7f);color:#0b1220;border:none}
.footer{margin-top:22px;color:#64748b;font-size:14px}
</style>
</head><body>
<main class="hero">
  <section class="card">
    <span class="badge">BAUZA GPT</span>
    <h1 class="h1">OSINT legal + Pentesting ético con entrega <span
style="color:var(--brand)">PDF/ZIP</span></h1>
    <p class="lead">Regístrate con Google, elige plan (Básico $10 MXN / Pro
$20 MXN), sube tu consulta y recibe resultados verificados.</p>
    <div class="grid">
      <div class="tile"><b>Plan Básico</b><br>$10 MXN<br><a class="btn"
href="/precios.html">Ver detalles</a></div>
      <div class="tile"><b>Plan Pro</b><br>$20 MXN<br><a class="btn primary"
href="/pago.html">Pagar ahora</a></div>
      <div class="tile"><b>OSINT Legal</b><br>Teléfono, correo,

```

```

usuario.<br><a class="btn" href="/gracias.html">Ejemplo de entrega</a></div>
  <div class="tile"><b>Pentest Ético</b><br>Alcance con contrato.<br><a
class="btn" href="/terminos.html">Términos</a></div>
  </div>
  <p class="footer">© BAUZA GPT – Hecho con disciplina y café.</p>
</section>
</main>
</body></html>

```

docs/precios.html

```

<!doctype html><meta charset="utf-8"><title>Precios – BAUZA GPT</title>
<link rel="preconnect" href="https://accounts.google.com"><script src="/load-
config.js" defer></script>
<h1>Precios</h1>
<ul>
  <li><b>Básico ($10 MXN)</b>: PDF con enlaces verificados (HTTP &lt; 400).</
li>
  <li><b>Pro ($20 MXN)</b>: PDF + ZIP con JSON/CSV y verificaciones extra.</
li>
</ul>
<a href="/pago.html">Continuar al pago</a>

```

docs/pago.html

```

<!doctype html><meta charset="utf-8"><title>Pagar – BAUZA GPT</title>
<script src="/load-config.js" defer></script>
<h1>Pago</h1>
<p>Selecciona un método. Los números de cuenta <b>no</b> se muestran aquí; el
backend los gestiona y valida manualmente.</p>
<div id="metodos"></div>
<script>
(async () => {
  const base = (window.BAUZA_PUBLIC?.apiBase)||'';
  const r = await fetch(`${base}/payments/methods`);
  const data = await r.json();
  document.getElementById('metodos').innerHTML = data.methods.map(m => `
    <div style="border:1px solid #333;padding:8px;border-radius:10px;margin:
8px 0">
      <b>${m.nombre}</b><br>
      Tipo: ${m.tipo}<br>
      <small>${m.descripcion||''}</small><br>
      <button onclick="crearPedido('${m.id}')">Crear pedido</button>
    </div>`).join('');
})();
async function crearPedido(metodo){
  const base = (window.BAUZA_PUBLIC?.apiBase)||'';
  const r = await fetch(`${base}/orders`,{
    method:'POST',

```

```

    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify({plan: 'PRO', metodo})
  });
  const o = await r.json();
  alert(`Pedido #${o.orderId} creado. Sigue las instrucciones del método
seleccionado.`);
}
</script>

```

docs/login.html

```

<!doctype html><meta charset="utf-8"><title>Login – BAUZA GPT</title>
<script src="/load-config.js" defer></script>
<h1>Login con Google</h1>
<p>(Integraremos Firebase Auth en un segundo paso. Este HTML ya carga la
configuración pública.)</p>

```

docs/privacidad.html

```

<!doctype html><meta charset="utf-8"><title>Privacidad – BAUZA GPT</title>
<h1>Aviso de Privacidad</h1>
<p>En el frontend no se exponen números de cuentas ni datos sensibles. La
validación de pago es manual desde el panel del backend.</p>

```

docs/terminos.html

```

<!doctype html><meta charset="utf-8"><title>Términos – BAUZA GPT</title>
<h1>Términos</h1>
<p>Servicio de OSINT legal con entrega por descarga segura mediante token. El
flujo no utiliza WhatsApp.</p>

```

docs/gracias.html

```

<!doctype html><meta charset="utf-8"><title>Gracias – BAUZA GPT</title>
<h1>Gracias</h1>
<p>Tu reporte está en proceso. Te avisaremos cuando esté listo para descargar
desde tu cuenta.</p>

```

docs/sitemap.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url><loc>https://www.bauzagpt.com/</loc></url>
  <url><loc>https://www.bauzagpt.com/precios.html</loc></url>
  <url><loc>https://www.bauzagpt.com/pago.html</loc></url>

```

```
<url><loc>https://www.bauzagpt.com/login.html</loc></url>
<url><loc>https://www.bauzagpt.com/privacidad.html</loc></url>
<url><loc>https://www.bauzagpt.com/terminos.html</loc></url>
<url><loc>https://www.bauzagpt.com/gracias.html</loc></url>
</urlset>
```

2) Backend (api/) — Node/Express listo

api/package.json

```
{
  "name": "bauzagpt-api",
  "version": "1.0.0",
  "type": "module",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "archiver": "^7.0.1",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "node-fetch": "^3.3.2",
    "pdfkit": "^0.13.0",
    "uuid": "^9.0.1"
  }
}
```

api/.env.example

```
PORT=8080
ADMIN_TOKEN=pon_un_token_fuerte
SITE_BASE=https://www.bauzagpt.com
```

api/server.js

```
import 'dotenv/config';
import express from 'express';
import path from 'node:path';
import { fileURLToPath } from 'node:url';
import health from './src/routes/health.js';
import orders from './src/routes/orders.js';
import payments from './src/routes/payments.js';
import generate from './src/routes/generate.js';

const app = express();
```

```

app.use(express.json());

// CORS básico
app.use((req, res, next)=>{
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');
  if(req.method==='OPTIONS') return res.sendStatus(200);
  next();
});

app.use('/health', health);
app.use('/orders', orders);
app.use('/payments', payments);
app.use('/generate', generate);

// Static para descargas temporales
const __dirname = path.dirname(fileURLToPath(import.meta.url));
app.use('/dl', express.static(path.join(__dirname, 'tmp'), { maxAge:
  '10m' }));

const PORT = process.env.PORT || 8080;
app.listen(PORT, ()=> console.log(`API lista en :${PORT}`));

```

api/src/routes/health.js

```

import { Router } from 'express';
const r = Router();
r.get('/', (_req, res)=> res.json({ ok:true, ts:Date.now() }));
export default r;

```

api/src/routes/orders.js

```

import { Router } from 'express';
import { v4 as uuid } from 'uuid';
const r = Router();

// En memoria (demo). En producción usa DB.
const ORDERS = new Map();

r.post('/', (req, res)=>{
  const { plan='PRO', metodo='manual' } = req.body||{};
  const orderId = uuid().slice(0,8);
  ORDERS.set(orderId, { orderId, plan, metodo, status: 'CREATED', at:
    Date.now()});
  res.json({ orderId, next: 'Sigue instrucciones del método
    seleccionado.' });
});

```

```

r.get('/:id', (req,res)=>{
  const o = ORDERS.get(req.params.id);
  if (!o) return res.status(404).json({error:'No existe'});
  res.json(o);
});

export default r;

```

api/src/routes/payments.js

```

import { Router } from 'express';
import methods from '../payments/methods.json' assert { type: 'json' };
const r = Router();

// Lista pública (sin exponer números). Los detalles se muestran tras crear
pedido y validar manualmente.
r.get('/methods', (_req,res)=>{
  res.json({ methods: methods.map(m => ({ id:m.id, nombre:m.nombre,
  tipo:m.tipo, descripcion:m.descripcion }))) });
});

export default r;

```

api/src/routes/generate.js

```

import { Router } from 'express';
import { checkLinks } from '../services/linkChecker.js';
import { makePDF } from '../services/pdfReport.js';
import { packZip } from '../services/zipPack.js';
import { v4 as uuid } from 'uuid';
import fs from 'node:fs/promises';
import path from 'node:path';
import { fileURLToPath } from 'node:url';

const r = Router();
const __dirname = path.dirname(fileURLToPath(import.meta.url));
const TMP = path.join(__dirname, '..', '..', 'tmp');

// POST /generate => { query:"texto", plan:"BASICO|PRO", items:
["http...", ...] }
r.post('/', async (req,res)=>{
  const { query='', plan='PRO', items=[] } = req.body||{};
  await fs.mkdir(TMP, { recursive: true });
  const id = uuid().slice(0,8);

  // 1) Verifica enlaces (HEAD)
  const results = await checkLinks(items);

  // 2) Genera PDF

```



```

const pdfPath = path.join(TMP, `${id}.pdf`);
await makePDF({ query, results, outputPath: pdfPath });

if (plan === 'BASICO') {
  return res.json({ id, pdf: `/dl/${id}.pdf` });
}

// 3) Genera ZIP con JSON/CSV + PDF
const jsonPath = path.join(TMP, `${id}.json`);
const csvPath = path.join(TMP, `${id}.csv`);
await fs.writeFile(jsonPath, JSON.stringify(results,null,2));
const csv = ['url,status'].concat(results.map(r=>`${r.url},${r.status}`)).join('\n');
await fs.writeFile(csvPath, csv);

const zipPath = path.join(TMP, `${id}.zip`);
await packZip({ files:[pdfPath,jsonPath,csvPath], outputPath: zipPath });

res.json({ id, pdf: `/dl/${id}.pdf`, zip: `/dl/${id}.zip` });
});

export default r;

```

api/src/services/linkChecker.js

```

import fetch from 'node-fetch';

export async function checkLinks(urls=[]) {
  const work = urls.map(async (url) => {
    try {
      const r = await fetch(url, { method: 'HEAD' });
      return { url, status: r.status };
    } catch (e) {
      return { url, status: 0 };
    }
  });
  return Promise.all(work);
}

```

api/src/services/pdfReport.js

```

import PDFDocument from 'pdfkit';
import fs from 'node:fs';

export async function makePDF({ query, results, outputPath }){
  return new Promise((resolve,reject)=>{
    const doc = new PDFDocument({ margin: 40 });
    const stream = fs.createWriteStream(outputPath);
    doc.pipe(stream);

```

```

    doc.fontSize(18).text('BAUZA GPT – Reporte OSINT', { align: 'center' });
    doc.moveDown();
    doc.fontSize(12).text(`Consulta: ${query}`);
    doc.moveDown();
    doc.text('Enlaces verificados (HTTP < 400 => OK):');
    doc.moveDown(0.5);
    results.forEach(r=>doc.text(`• ${r.url} – status: ${r.status}`));
    doc.end();
    stream.on('finish', resolve);
    stream.on('error', reject);
  });
}

```

api/src/services/zipPack.js

```

import fs from 'node:fs';
import archiver from 'archiver';

export async function packZip({ files=[], outputPath }){
  return new Promise((resolve,reject)=>{
    const output = fs.createWriteStream(outputPath);
    const archive = archiver('zip', { zlib: { level: 9 } });
    archive.pipe(output);
    files.forEach(f=>archive.file(f, { name: f.split('/').pop() }));
    archive.finalize();
    output.on('close', resolve);
    archive.on('error', reject);
  });
}

```

api/src/payments/methods.json

```

[
  { "id": "bbva", "nombre": "BBVA Spin", "tipo": "transferencia",
    "descripcion": "Se muestran instrucciones tras crear pedido.", "visible":
    true },
  { "id": "nu", "nombre": "Banco Nu", "tipo": "transferencia",
    "descripcion": "Validación manual.", "visible": true },
  { "id": "oxxo", "nombre": "Depósito OXXO", "tipo": "efectivo",
    "descripcion": "Código y folio tras pedido.", "visible": true },
  { "id": "paypal", "nombre": "PayPal", "tipo": "digital",
    "descripcion": "@ivanbauza (handle)", "visible": true }
]

```

```
api/src/utils/tokens.js
```

```
import { v4 as uuid } from 'uuid';  
export const makeToken = () => uuid().replace(/-/g, '').slice(0,24);
```

3) README (resumen en raíz)

```
# BAUZA GPT – OSINT & Pentest  
  
## Despliegue rápido  
  
### Frontend: GitHub Pages  
1. En **Settings → Pages** del repo: Source = **Deploy from a branch** →  
Branch = **principal** → Folder = **/docs**.  
2. `docs/CNAME` debe contener: `www.bauzagpt.com`.  
3. Sube `docs/config.json` (copia de `config.sample.json` con tu `apiBase`  
real).  
  
### Backend: Render/Railway/Heroku  
``bash  
cd api  
npm i  
# crea .env a partir de .env.example  
npm start
```

Configura la URL pública como `apiBase` en `docs/config.json`.

Comandos para subir

```
# Linux/WSL  
cd ~/bauzagpt.com  
git checkout principal || git checkout -b principal  
git add .  
git commit -m "feat: sitio + API inicial"  
git push -u origin principal
```

```
# Windows PowerShell  
cd $HOME/bauzagpt.com  
if (-not (git branch --show-current)) { git checkout -b principal }  
git add .  
git commit -m "feat: sitio + API inicial"  
git push -u origin principal
```

```
---

## 4) `.gitignore` (raíz)
```.gitignore
node_modules
.env
.env.*
api/tmp
.DS_Store
```

## 5) Scripts para crear todos los archivos (Linux/WSL)

Ejecuta en la **raíz** del repo.

```
set -e
mkdir -p docs img api/src/{routes,services,utils,payments} api/tmp

CNAME
printf "www.bauzagpt.com" > docs/CNAME

load-config.js (JS puro)
cat > docs/load-config.js <<'EOF'
(async () => {
 try {
 const res = await fetch('/config.json', { cache: 'no-store' });
 if (!res.ok) throw new Error('No se pudo cargar config.json');
 window.BAUZA_PUBLIC = await res.json();
 } catch (e) { console.error('Error al cargar config:', e); }
})();
EOF

config.sample.json
cat > docs/config.sample.json <<'EOF'
{
 "firebase": {
 "apiKey": "TU_API_KEY_PUBLICA",
 "authDomain": "tu-proyecto.firebaseio.com",
 "projectId": "tu-proyecto",
 "storageBucket": "tu-proyecto.appspot.com",
 "messagingSenderId": "000000000000",
 "appId": "1:000000000000:web:XXXXXXXXXXXXXXXX"
 },
 "apiBase": "https://TU-BACKEND.onrender.com"
}
EOF

index.html (placeholder corto; reemplaza por el completo de esta guía)
```

```

cat > docs/index.html <<'EOF'
<!doctype html><html lang="es"><head><meta charset="utf-8"/><meta
name="viewport" content="width=device-width,initial-scale=1"/><title>BAUZA
GPT – OSINT & Pentest Ético</title><meta name="description" content="OSINT
legal, pentesting ético y reportes automáticos."><script src="/load-
config.js" defer></script></head><body><h1>BAUZA GPT</h1><p>Landing mínima.
Reemplázala por la versión completa de la guía.</p></body></html>
EOF

Páginas básicas
for p in precios pago login privacidad terminos gracias; do echo "<!doctype
html><meta charset='utf-8'><title>${p^} – BAUZA GPT</title><h1>${p^}</h1>" >
docs/${p}.html; done

sitemap
cat > docs/sitemap.xml <<'EOF'
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
 <url><loc>https://www.bauzagpt.com/</loc></url>
</urlset>
EOF

Backend
cat > api/package.json <<'EOF'
{
 "name": "bauzagpt-api",
 "version": "1.0.0",
 "type": "module",
 "main": "server.js",
 "scripts": { "start": "node server.js" },
 "dependencies": {
 "archiver": "^7.0.1",
 "dotenv": "^16.4.5",
 "express": "^4.19.2",
 "node-fetch": "^3.3.2",
 "pdfkit": "^0.13.0",
 "uuid": "^9.0.1"
 }
}
EOF

cat > api/.env.example <<'EOF'
PORT=8080
ADMIN_TOKEN=pon_un_token_fuerte
SITE_BASE=https://www.bauzagpt.com
EOF

cat > api/server.js <<'EOF'
import 'dotenv/config';
import express from 'express';
import path from 'node:path';

```

```

import { fileURLToPath } from 'node:url';
import health from './src/routes/health.js';
import orders from './src/routes/orders.js';
import payments from './src/routes/payments.js';
import generate from './src/routes/generate.js';
const app = express();
app.use(express.json());
app.use((req,res,next)=>{res.setHeader('Access-Control-Allow-Origin','*');res.setHeader('Access-Control-Allow-Headers','Content-Type,Authorization');res.setHeader('Access-Control-Allow-Methods','GET,POST,OPTIONS');if(req.method==='OPTIONS')return res.sendStatus(200);next();});
app.use('/health', health);app.use('/orders', orders);app.use('/payments', payments);app.use('/generate', generate);
const __dirname = path.dirname(fileURLToPath(import.meta.url));
app.use('/dl', express.static(path.join(__dirname,'tmp'), { maxAge: '10m' }));
const PORT = process.env.PORT || 8080;app.listen(PORT, ()=> console.log(`API lista en :${PORT}`));
EOF

```

```

mkdir -p api/src/{routes,services,utils,payments}

```

```

cat > api/src/routes/health.js <<'EOF'
import { Router } from 'express';
const r = Router();
r.get('/', (_req,res)=> res.json({ ok:true, ts:Date.now() }));
export default r;
EOF

```

```

cat > api/src/routes/orders.js <<'EOF'
import { Router } from 'express';
import { v4 as uuid } from 'uuid';
const r = Router();
const ORDERS = new Map();
r.post('/', (req,res)=>{ const { plan='PRO', metodo='manual' } = req.body|| {}; const orderId = uuid().slice(0,8); ORDERS.set(orderId,{ orderId, plan, metodo, status:'CREATED', at:Date.now()}); res.json({ orderId, next:'Sigue instrucciones del método seleccionado.'});});
r.get('/:id', (req,res)=>{ const o = ORDERS.get(req.params.id); if(!o) return res.status(404).json({error:'No existe'}); res.json(o);});
export default r;
EOF

```

```

cat > api/src/routes/payments.js <<'EOF'
import { Router } from 'express';
import methods from '../payments/methods.json' assert { type: 'json' };
const r = Router();
r.get('/methods', (_req,res)=>{ res.json({ methods: methods.map(m=>({ id:m.id,nombre:m.nombre,tipo:m.tipo,descripcion:m.descripcion})));});});
export default r;

```

EOF

```
cat > api/src/routes/generate.js <<'EOF'
import { Router } from 'express';
import { checkLinks } from '../services/linkChecker.js';
import { makePDF } from '../services/pdfReport.js';
import { packZip } from '../services/zipPack.js';
import { v4 as uuid } from 'uuid';
import fs from 'node:fs/promises';
import path from 'node:path';
import { fileURLToPath } from 'node:url';
const r = Router();
const __dirname = path.dirname(fileURLToPath(import.meta.url));
const TMP = path.join(__dirname, '..', '..', 'tmp');
r.post('/', async (req,res)=>{ const { query='', plan='PRO', items=[] } =
req.body||{}; await fs.mkdir(TMP,{recursive:true}); const id =
uuid().slice(0,8); const results = await checkLinks(items); const pdfPath =
path.join(TMP, `${id}.pdf`); await makePDF({ query, results, outPath:
pdfPath }); if (plan==='BASICO') return res.json({ id, pdf:`dl/$
{id}.pdf` }); const jsonPath = path.join(TMP, `${id}.json`); const csvPath =
path.join(TMP, `${id}.csv`); await fs.writeFile(jsonPath,
JSON.stringify(results,null,2)); const csv =
['url,status'].concat(results.map(r=>`${r.url},${r.status}`)).join('\n');
await fs.writeFile(csvPath, csv); const zipPath = path.join(TMP, `${
{id}.zip`); await packZip({ files:[pdfPath,jsonPath,csvPath], outPath:
zipPath }); res.json({ id, pdf:`dl/${id}.pdf`, zip:`dl/${id}.zip` });});
export default r;
EOF
```

```
cat > api/src/services/linkChecker.js <<'EOF'
import fetch from 'node-fetch';
export async function checkLinks(urls=[]) { return Promise.all(urls.map(async
(url)=>{ try{ const r=await fetch(url,{method:'HEAD'}); return { url,
status:r.status }; }catch(e){ return { url, status:0 }; }))); }
EOF
```

```
cat > api/src/services/pdfReport.js <<'EOF'
import PDFDocument from 'pdfkit';
import fs from 'node:fs';
export async function makePDF({ query, results, outPath }){ return new
Promise((resolve,reject)=>{ const doc=new PDFDocument({margin:40}); const
stream=fs.createWriteStream(outPath); doc.pipe(stream);
doc.fontSize(18).text('BAUZA GPT – Reporte OSINT',{align:'center'});
doc.moveDown(); doc.fontSize(12).text(`Consulta: ${query}`); doc.moveDown();
doc.text('Enlaces verificados (HTTP < 400 => OK):'); doc.moveDown(0.5);
results.forEach(r=>doc.text(`• ${r.url} – status: ${r.status}`)); doc.end();
stream.on('finish', resolve); stream.on('error', reject); }); }
EOF
```

```
cat > api/src/services/zipPack.js <<'EOF'
import fs from 'node:fs';
```

```

import archiver from 'archiver';
export async function packZip({ files=[], outputPath }){ return new
Promise((resolve,reject)=>{ const output=fs.createWriteStream(outputPath); const
archive=archiver('zip',{ zlib:{level:9}}); archive.pipe(output);
files.forEach(f=>archive.file(f,{ name:f.split('/').pop() }));
archive.finalize(); output.on('close', resolve); archive.on('error',
reject); }); }
EOF

cat > api/src/payments/methods.json <<'EOF'
[
 { "id":"bbva", "nombre":"BBVA Spin", "tipo":"transferencia",
"descripcion":"Se muestran instrucciones tras crear pedido.", "visible":
true },
 { "id":"nu", "nombre":"Banco Nu", "tipo":"transferencia",
"descripcion":"Validación manual.", "visible": true },
 { "id":"oxo", "nombre":"Depósito OXXO", "tipo":"efectivo",
"descripcion":"Código y folio tras pedido.", "visible": true },
 { "id":"paypal","nombre":"PayPal", "tipo":"digital",
"descripcion":"@ivanbauza (handle)", "visible": true }
]
EOF

README breve
cat > README.md <<'EOF'
(Consulta la guía principal en este documento/canvas para pasos completos.)
EOF

.gitignore
cat > .gitignore <<'EOF'
node_modules
.env
.env.*
api/tmp
.DS_Store
EOF

printf "\nListo. Ahora:\n cd api && npm i && npm start\n\nLuego sube todo
con git (rama principal /docs).\n"

```

## 6) Subir a GitHub — comandos finales

### WSL/Linux

```

cd ~/bauzagpt.com
git checkout principal || git checkout -b principal
git add .

```



```
git commit -m "feat: frontend Pages + backend API inicial"
git push -u origin principal
```

### PowerShell (Windows)

```
cd $HOME/bauzagpt.com
if (-not (git branch --show-current)) { git checkout -b principal }
git add .
git commit -m "feat: frontend Pages + backend API inicial"
git push -u origin principal
```

En **GitHub** → **Settings** → **Pages**: selecciona **Branch: principal** y **Folder: /docs**.

## 7) Prueba rápida (sanity check)

- Frontend: `https://vanerbauza.github.io/bauzagpt.com/` (o tu dominio `www.bauzagpt.com`).
- Backend local: `http://localhost:8080/health` → `{ ok: true }`.
- Endpoint demo: `POST /generate` con body `{"query": "prueba", "plan": "PRO", "items": ["https://example.com"]}`.

## 8) Notas y buenas prácticas

- **Cero secretos en el frontend.**
- Administra métodos de pago en `api/src/payments/methods.json` o vía un endpoint/panel admin.
- Las entregas se publican temporalmente en `/dl/{id}.pdf` y `/dl/{id}.zip`.
- Si sirves el sitio bajo subruta (sin CNAME), ajusta rutas absolutas (`/load-config.js` → `./load-config.js` y `/config.json` → `./config.json`).

¡Listo para producción, gallo! 🚀