Gmail **Fillipe Guerra <fillipe.backup@gmail.com>**

Brand Scanner

1 mensagem

Fillipe Guerra <fillipe.backup@gmail.com> 26 de outubro de 2025 às 12:31
Para: Fillipe Augusto Gomes Guerra <fillipe182@hotmail.com>, Fillipe Guerra <fillipe.backup@gmail.com>

Exemplo de Arquitetura Brand Scanner

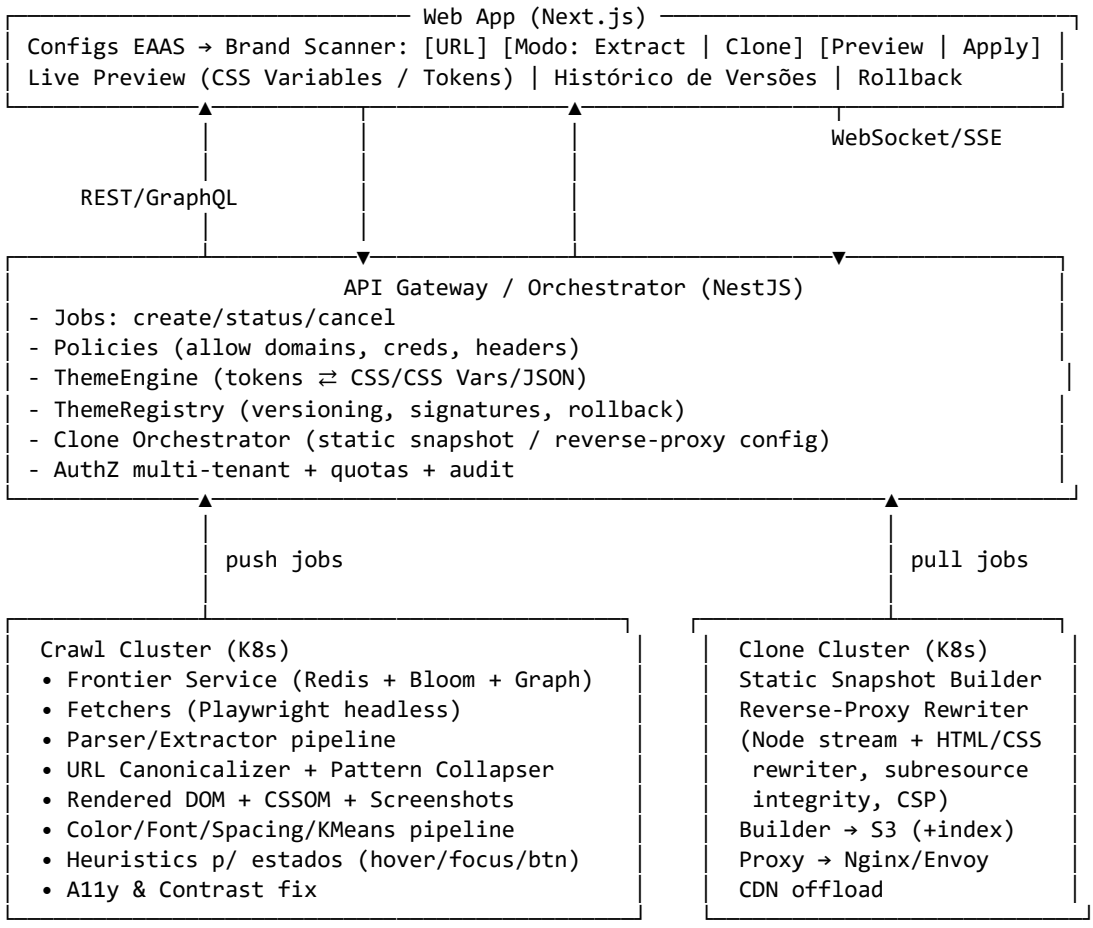
Visão 10/10 (SoTA)

Dois modos complementares:

1. **Brand Extract**
   Varrer *todo* o site para inferir **Design Tokens** (cores, tipografia, espaçamentos, componentes-chave, estados, ícones), **logo**, **favicons**, **gradientes**, **radius**, **elevation**, **densidade**; gerar **ThemeBundle** versionado; **Preview** instantâneo; **Aplicar**/"**Restaurar**".

2. **Brand Clone (Somente se aplica ao Marketplace dos Clientes, NAO se aplica a dashboard admin)**
   Substituir o visual do MArketplace dos clientes Fazendo **mirroring fiel** do site do cliente (estático + assets) **ou** operar como **Reverse-Proxy Rewriter** (espelho em tempo real) — com **URL rewriting**, **substituição de endpoints**, **injeção do nosso runtime** (barra de compra, auth, analytics, AB testing), e **política de cache/CDN**. Também com **Preview** seguro e **rollback**.

Arquitetura distribuída (alta escala, sem limites artificiais)



```
Storage & Infra:
- **Postgres** (ThemeVersion, ScanReport, CloneManifests)
- **Redis** (Frontier, filas BullMQ, WebSocket presence)
- **S3/MinIO** (screenshots, assets, WARC/ZIP snapshot)
- **OTel + Tempo/Jaeger + Loki + Grafana** (traces/métricas/logs)
- **Sentry** (app + workers)
- **CDN** (clone estático, assets)
```

Pontos-chave SoTA

- **Sem "limite de páginas"**: quem limita é **capacidade horizontal** (auto-scale); garantimos cobertura **com deduplicação inteligente**, **detecção de malhas infinitas/paginação** e **colapso de padrões** (explicado abaixo).
- **JS-rendering real**: SPA/SSR/CSR suportado (Playwright).
- **"Coverage" semântica**: sabemos **quando basta** (todas as variações de layout/tema capturadas) usando **assinaturas de DOM/CSS**.
- **Clone** confiável: 2 estratégias — **Snapshot build** (estático) **ou Reverse Proxy** (tempo real), ambas com **rewriting** e **injeção controlada**.

Algoritmos e técnicas cruciais

1) Frontier sem fim ≠ rastreamento cego

Para "tudo do site" sem travar:

- **Canonicalização de URL**: normaliza querystring (ordem, lower), remove trackers (`utm_*`, `fbclid`), **variantes de paginação** colapsadas (e.g. `?page=2..∞` → amostra [1,2, last]).
- **Equivalência de templates**: hash do **DOM shape** (sem conteúdo dinâmico) + **CSS ruleset signature**; se o hash já visto, **não recrawlear** variação igual (evita facet hell).
- **Detecção de loops**: Bloom filter + **limite por componente** (ex.: `/search?*` → amostra topN).
- **Sitemap seeds**: usar `sitemap.xml` para ampliar cobertura de forma barata.
- **Ordem de prioridade**: páginas que diversifiquem **estilos** (novas assinaturas) entram primeiro.

Código — Frontier (Redis + Bloom) e Canonicalização

```
// frontier.ts
import { createClient } from 'redis';
import BloomFilter from '@jacobbubu/bloom-filters'; // qualquer BF eficiente

const redis = createClient({ url: process.env.REDIS_URL });

export function canonicalize(u: URL): string {
  // 1) força https base
  const url = new URL(u.toString());
  // 2) remove trackers e normaliza query
  const blacklist = ['utm_source','utm_medium','utm_campaign','utm_term','utm_
content','fbclid','gclid','ref'];
  const qs = [...url.searchParams.entries()]
    .filter(([k]) => !blacklist.includes(k.toLowerCase()))
    .sort((a,b)=> a[0].localeCompare(b[0]));
  url.search = '';
  qs.forEach(([k,v]) => url.searchParams.append(k,v));
  // 3) remove trailing slash duplicado e âncoras
  url.hash = '';
  return url.toString();
}

export class Frontier {
  private seenBloom = BloomFilter.fromJSON(await redis.get('bloom:seen').then(x=>x && JSON.parse(x) ||
BloomFilter.create(1e7, 1e-6)));
```

```
  async enqueue(seed: string) {
    const key = 'queue:url';
    const canon = canonicalize(new URL(seed));
    if (this.seenBloom.has(canon)) return;
    await redis.lPush(key, canon);
  }

  async next(): Promise<string | null> {
    const key = 'queue:url';
    const url = await redis.rPop(key);
    if (!url) return null;
    this.seenBloom.add(url);
    await redis.set('bloom:seen', JSON.stringify(this.seenBloom));
    return url;
  }
}
```

2) Extração robusta de identidade (Design Mining)

- **CSSOM completo**: coletar `:root` vars, `@font-face`, cores de `body`, `a`, `button`, `input`, **estados** `:hover/:focus/:disabled`.
- **Screenshots com K-Means** por **região sem texto** e **sem imagens de produto** (classificador rápido por `alt`, `role`, bounding boxes).
- **Gradient & shadow parsing**.
- **A11y auto-fix** (WCAG AA/AAA) com menor desvio perceptual (ΔE2000).
- **Tipografia tribal**: mapear `family`, `weights`, `scale` (clamp) e **hierarquia** (display/h1/h2/body/mono).

Código — coleta de hints + fontes + variáveis

```
// extractor/cssHints.ts (executa dentro do page.evaluate)
export async function collectCssHints(page: import('playwright').Page) {
  return page.evaluate(() => {
    function get(el: Element|null, prop: string, fallback?: string) {
      if (!el) return fallback;
      const v = getComputedStyle(el as Element).getPropertyValue(prop);
      return v || fallback;
    }
    const sel = (q: string) => document.querySelector(q);

    const body = getComputedStyle(document.body);
    const root = getComputedStyle(document.documentElement);
    const vars: Record<string,string> = {};
    for (let i=0;i<root.length;i++){
      const name = root[i];
      if (name.startsWith('--')) vars[name] = root.getPropertyValue(name).trim();
    }

    const fonts = Array.from(document.fonts || []).map(f => ({
      family: f.family, weight: Number(f.weight||400), style: f.style, stretch: f.stretch
    }));

    const roles = {
      bg: body.backgroundColor,
      fg: body.color,
      link: get(sel('a'),'color', body.color),
      buttonBg: get(sel('button'),'background-color', body.backgroundColor),
      buttonFg: get(sel('button'),'color', body.color),
      inputBg: get(sel('input'),'background-color', body.backgroundColor),
      inputFg: get(sel('input'),'color', body.color),
    };

    return { vars, fonts, roles, bodyFont: body.fontFamily };
  });
}
```

Código — paleta por screenshot + ΔE dedupe

```
// colors/palette.ts
import { getPixels } from './pixels'; // pega pixels ignorando texto (detecção simples via
contraste/bbox)
import { kmeans } from '@mljs/kmeans';
import deltaE from 'delta-e';

export async function paletteFromScreenshot(buf: Buffer, k=8): Promise<string[]> {
  const pixels = await getPixels(buf); // [[r,g,b], ...]
  const { centroids } = kmeans(pixels, k, { initialization: 'kmeans++', maxIterations: 60 });
  const rgb = centroids.map(c => `#${c.map(x => Math.round(x).toString(16).
padStart(2,'0')).join('')}`);
  // dedupe por percepção (ΔE2000)
  const keep: string[] = [];
  for (const hex of rgb) {
    if (!keep.some(h => deltaE.getDeltaE00(hexToLab(h), hexToLab(hex)) < 5)) keep.push(hex);
  }
  return keep;
}
```

Theme Tokens (única fonte de verdade)

```
export interface ThemeTokens {
  color: {
    primary: string; secondary: string; accent: string; neutral: string;
    bg: string; fg: string; link: string;
    success: string; warning: string; danger: string;
    surface?: string; subtle?: string;
  };
  font: {
    body: Typo; heading: Typo; mono?: Typo; cta?: Typo;
    scale?: { basePx: number; ratio: 1.125|1.2|1.25|1.333 };
  };
  radius: { sm:number; md:number; lg:number; xl?:number };
  spacing: { base:number; steps:number[] };
  shadow?: { sm:string; md:string; lg:string };
  border?: { width:number; style:'solid'|'dashed'; color?:string };
}
```

CSS Vars do tema (preview “ao vivo”)

```
export function tokensToCssVars(t: ThemeTokens) {
  const L = [':root {'];
  for (const [k,v] of Object.entries(t.color)) L.push(`--color-${k}:${v};`);
  L.push(`--font-body:${quote(t.font.body.family)},${t.font.body.fallbacks.join(',')};`);
  L.push(`--font-heading:${quote(t.font.heading.family)},${t.font.heading.fallbacks.join(',')};`);
  L.push(`--radius-sm:${t.radius.sm}px; --radius-md:${t.radius.md}px; --radius-lg:${t.radius.lg}px;`);
  L.push(`--space-base:${t.spacing.base}px;`);
  t.spacing.steps.forEach(s => L.push(`--space-${s}:${s*t.spacing.base}px;`));
  L.push('}');
  return L.join('\n');
}
```

Resultado: Preview muda instantaneamente (injetando `<style id="theme-vars">`).

---

3) Clone (Snapshot & Proxy), realmente “pixel-perfect”

3.1 Snapshot Builder (estático “congelado”)

- Rastreia todas as rotas renderizadas (como no crawler).
- **Baixa e reescreve** HTML/CSS/JS (URLs absolutas → relativas ao nosso bucket).
- **Injeta** nosso runtime (marketplace, auth), via **HTML Rewriter**.
- Gera **manifesto** (`clone.json`) e **sitemap clonado**.

- Empacota em **WARC** ou **ZIP + manifest** e publica em **S3 + CDN**.

Código — Rewriter de HTML streaming (Node + parse5)

```
import { parse, serialize, DefaultTreeDocument } from 'parse5';
import { Readable } from 'node:stream';

export function rewriteHtml(html: string, mapUrl: (u:string)=>string): string {
  const doc = parse(html) as DefaultTreeDocument;

  // 1) reescrever <a href>, <img src>, <link href>, <script src>
  const walk = (n: any) => {
    if (n.attrs) {
      for (const a of n.attrs) {
        if (['href','src','content'].includes(a.name)) {
          try { a.value = mapUrl(a.value); } catch {}
        }
      }
    }
    if (n.childNodes) n.childNodes.forEach(walk);
  };
  walk(doc);

  // 2) injetar nosso runtime antes de </head>
  const head = (doc.childNodes[1]?.childNodes || []).find((x:any)=>x.nodeName==='head');
  if (head) {
    head.childNodes.push({
      nodeName: 'script', tagName: 'script', attrs: [{name:'src', value:'/eaas/runtime.js'}],
      childNodes: [], namespaceURI: 'http://www.w3.org/1999/xhtml'
    });
    head.childNodes.push({
      nodeName: 'link', tagName: 'link', attrs: [{name:'rel', value:'stylesheet'}, {name:'href',
value:'/eaas/runtime.css'}],
      childNodes: [], namespaceURI: 'http://www.w3.org/1999/xhtml'
    });
  }

  return serialize(doc);
}
```

3.2 Reverse-Proxy Rewriter (tempo real)

- **Envoy/Nginx** recebe https://marketplace.cliente.com.
- Encaminha para https://site-oficial.com mas **reescreve**:
  - **Host/Origin/Cookie** (isolamento).
  - **Links/Assets** para ficar sob nosso domínio.
  - Injeção do runtime e **CSP** que permita nossos scripts.
- Cache agressivo (CDN), revalidação e **fallback** para snapshot quando upstream cai.

Código — Rewriter de resposta (Node/Express + http-proxy)

```
import httpProxy from 'http-proxy';
import { rewriteHtml } from './rewriter';
const proxy = httpProxy.createProxyServer({ changeOrigin:true, selfHandleResponse: true });

app.use('/proxy', (req, res) => {
  proxy.web(req, res, { target: 'https://site-oficial.com' });
});

proxy.on('proxyRes', async (proxyRes, req, res) => {
  const chunks: Buffer[] = [];
  proxyRes.on('data', (c) => chunks.push(c));
  proxyRes.on('end', () => {
    const buf = Buffer.concat(chunks);
```

```
    const ct = proxyRes.headers['content-type'] || '';
    const headers = { ...proxyRes.headers };
    delete headers['content-security-policy'];           // reemitimos CSP compatível
    Object.entries(headers).forEach(([k,v]) => res.setHeader(k, v as any));

    if (String(ct).includes('text/html')) {
      const body = buf.toString('utf8');
      const mapped = rewriteHtml(body, absoluteToOurDomain);
      return res.status(proxyRes.statusCode||200).send(mapped);
    }
    return res.status(proxyRes.statusCode||200).send(buf);
  });
});
```

> **Obs.** Para sites SPA com roteamento em `history.pushState`, injetamos um **Service Worker** leve para reescrever fetches e manter tudo sob o nosso host.

---

4) Versionamento, rollback, assinatura

- **ThemeVersion**: imutável, com **HMAC** do bundle.
- **CloneManifest**: versão + lista de rotas + mapa de assets + carimbo do commit.
- **Rollback**: trocar ponteiro `active_version` por O(1); idem para **clone ativo** (snapshot vs proxy).
- **Diffs**: comparar tokens e gerar **changelog** legível.

Esquema (Prisma)

```
model ThemeVersion {
  id         String   @id @default(uuid())
  tenantId   String
  version    Int
  bundle     Json     // ThemeTokens + assets + signature
  createdAt  DateTime @default(now())
  createdBy  String?
  scanId     String?
  isActive   Boolean  @default(false)
  @@unique([tenantId, version])
}

model CloneManifest {
  id         String   @id @default(uuid())
  tenantId   String
  version    Int
  mode       String   // 'snapshot' | 'proxy'
  manifest   Json     // rotas, assets, rewrites
  createdAt  DateTime @default(now())
  isActive   Boolean  @default(false)
}
```

API Contratos (claros e didáticos)

```
// POST /brand-scanner/jobs
type CreateJobReq = {
  mode: 'extract' | 'clone';
  url: string;
  deep: 'full';                  // sem limite arbitrário
  cloneMode?: 'snapshot' | 'proxy';
  includeSubdomains?: boolean;
  headers?: Record<string,string>; // se cliente exigir login/token
  auth?: { type:'basic'|'bearer'; value:string };
};
type CreateJobRes = { jobId: string };

// GET /brand-scanner/jobs/:jobId
```

```
type JobStatus =
  | { status:'queued'|'running'; progress:number }
  | { status:'done'; reportId?: string; themeVersion?: number; cloneVersion?: number; previewUrl:
string }
  | { status:'failed'; error:string };

// POST /brand-scanner/theme/activate { version:number }
type ActivateThemeRes = { activeVersion:number };

// POST /brand-scanner/clone/activate { version:number }
type ActivateCloneRes = { activeVersion:number };
```

Erros padronizados (JSON: `code`, `message`, `hint`, `docsUrl`).

Observabilidade, SLO e Runbooks

- **SLO**:
  - *Extract*: p95 < **8 min** para sites até 10k páginas; contínuo acima disso (streaming da cobertura).
  - *Clone Snapshot*: p95 < **15 min** (com CDN warm).
  - *Proxy*: TTFB adicional < **40ms** p95.
- **Métricas**: páginas varridas/min, domínios/seg, % de novas assinaturas de layout, taxa de falhas por tipo (CSP, CORS, JS error), ΔE médio de auto-ajuste A11y.
- **Runbooks**:
  1. "**Coverage baixa**": verificar Frontier Graph; ajustar `includeSubdomains`; semantizar padrão de URL facetada.
  2. "**Layout quebrado no clone**": olhar `CSP` no proxy; revisar rewriter de módulos AMD/UMD; ativar modo "snapshot" como fallback.
  3. "**Fonts não renderizam**": copiar `@font-face` + `woff2`; reescrever URLs; garantir `Access-Control-Allow-Origin` no bucket.
  4. "**SPA não navega**": verificar Service Worker injetado e rotas do History API.

UX do Preview (polido e sofisticado)

- **Comparador Lado-a-Lado** (padrão × novo) com *split handle*.
- **Editor fino** antes de aplicar (trocar primária/secundária, ajustar tons até passar AAA nos componentes críticos).
- **Histórico** com miniaturas de paletas e fontes.
- **Switch instantâneo** (CSS Vars) sem recarregar a página.
- **Botões**: "**Aplicar Tema**", "**Ativar Clone** (snapshot/proxy)", "**Restaurar Padrão**".

Trechos essenciais adicionais

Worker de Crawl com Playwright-Cluster (auto-scale)

```
import { Cluster } from 'puppeteer-cluster'; // similar para Playwright cluster
import { Frontier, canonicalize } from './frontier';
import { extractPageSignature, extractBrandHints } from './extract';

const frontier = new Frontier();

const cluster = await Cluster.launch({
```

```
  concurrency: Cluster.CONCURRENCY_CONTEXT,
  maxConcurrency: Number(process.env.MAX_WORKERS||64),
  puppeteerOptions: { headless: 'new', args: ['--no-sandbox','--disable-dev-shm-usage'] }
});

await cluster.task(async ({ page, data: url }: {page:any, data:string}) => {
  await page.goto(url, { waitUntil: 'networkidle2', timeout: 30000 });

  // 1) coleta links internos e enfileira
  const links: string[] = await page.$$eval('a[href]', as => as.map(a => (a as
HTMLAnchorElement).href));
  for (const l of links) {
    const u = new URL(l);
    if (u.host === new URL(url).host) await frontier.enqueue(canonicalize(u));
  }

  // 2) extrai assinatura e hints
  const signature = await extractPageSignature(page); // hash de DOM/CSS
  const hints = await extractBrandHints(page);        // tokens parciais
  await savePartial(url, signature, hints);
});

await frontier.enqueue(canonicalize(new URL(seedUrl)));
while (true) {
  const next = await frontier.next();
  if (!next) break;
  cluster.queue(next);
}
await cluster.idle();
await cluster.close();
```

HTML Rewriter — mapa de URL (proxy/snapshot)

```
function absoluteToOurDomain(orig: string): string {
  try {
    const u = new URL(orig, 'https://site-oficial.com');
    // assets estáticos → /_assets/..., rotas → /_page/...
    if (/\.(png|jpe?g|gif|svg|webp|woff2?|css|js|map)$/i.test(u.pathname))
      return `/assets${u.pathname}${u.search}`;
    return `/page${u.pathname}${u.search}`;
  } catch { return orig; }
}
```

Documentação (rica e didática)

Inclua no repositório:

- `docs/01-overview.md`: o que é Brand Extract/Clone, diagramas, SLO.
- `docs/02-architecture.md`: serviços, filas, clusters, storages, rede, segurança.
- `docs/03-contracts.md`: DTOs, exemplos reais de **ScanReport**, **ThemeBundle**, **CloneManifest**.
- `docs/04-algorithms.md`: Frontier, canonicalização, equivalência de templates, K-Means, ΔE, A11y.
- `docs/05-runbooks.md`: como debugar, fallbacks, toggles de feature.
- `docs/06-api.md`: endpoints, exemplos curl, erros.
- `docs/07-deploy.md`: Helm charts, autoscaling, limites de recursos, CDN.
- `docs/08-security.md`: isolamento, secrets, auditoria, assinatura.
- `docs/09-ux-preview.md`: como o preview aplica tokens e como editar.
- `docs/10-testing.md`: matriz de sites canário (SPA, SSR, e-commerce, blog, portal), testes E2E.

Cada doc com **diagramas mermaid**, **trechos de código comentados**, **checklists** e **“gotchas”**.