Website Generator App - Design, Structure, and Tags

OVERVIEW

Background

The existing website generator application uses a single JSON file structure where content, layout, and design elements are tightly coupled. This approach creates several challenges:

- Business users can break pages by making syntax errors in layout configuration
- Content cannot be reused across different page layouts
- Design changes require editing content files, risking data corruption
- No clear separation of concerns between what business users edit vs. what developers control
- Difficult to maintain consistency across pages with similar layouts
- No validation to prevent invalid configurations

The current system evolved organically without a formal architecture, resulting in technical debt and maintenance challenges that limit scalability.

Purpose

The WebGen Phase 1 project establishes a clean separation between form and function by refactoring the JSON architecture into distinct layers:

Configuration Layer
Site-wide settings, layout templates, navigation patterns (admin-managed)

Content Layer
Pure content blocks with no layout information (business user-managed)

Mapping Layer
Page configurations that connect content to layouts (admin-managed)

Navigation Layer
Menu structure and hub/spoke relationships (admin-managed)

This separation ensures business users can safely edit content without risk of breaking the website, while administrators maintain full control over design, layout, and navigation patterns.

SCOPE

In Scope

✅ Define standardized JSON file types with clear responsibilities
✅ Create reusable layout templates and page patterns
✅ Establish hub/spoke directory structure (hub00-hub99)
✅ Separate content files (.content.json) from page configurations (.page.json)
✅ In addition to JSON, support TOML formatted content files (.content.txt)
✅ Implement site-wide configuration system (./site/config.json)
✅ Define secondary navigation patterns for hub/spoke pages
✅ Create validation rules and error handling
✅ Document migration strategy from legacy format
✅ Maintain backward compatibility during transition

Out of Scope

❌ Visual content editor UI (Phase 2)
❌ Multi-language content management system (Phase 2)
❌ Version control and content approval workflows (Phase 2)
❌ Automated testing suite (Phase 2)
❌ Performance optimization (Phase 3)
❌ API for external integrations (Phase 3)

For Administrators (Developers/Designers)

🎯 Centralized control - Change all page layouts from one file
🎯 Reusable templates - Define once, use everywhere
🎯 Easy maintenance - Fix layout bugs in one place
🎯 Clear ownership - Know exactly who should edit what
🎯 Better validation - Catch errors before deployment

For the Organization:

🎯 Reduced errors - Fewer broken pages in production
🎯 Faster time-to-market - Business users can update content independently
🎯 Lower training costs - Business users only learn content structure
🎯 Scalability - Easy to add new pages and layouts
🎯 Consistency - All pages follow standardized patterns

WEBSITE DEV STRUCTURE

`_ips-v4/`	PROJECT ROOT
`├── package.json`	NPM config (ROOT LEVEL)
`├── .gitignore`	Git ignore file
`├── README.md`	Documentation
`│`	
`├── build/`	BUILD system (Node.js scripts)
`│   ├── config.json`	BUILD settings file
`│   ├── generator.js`	Main build orchestrator
`│   ├── renderer.js`	HTML generator
`│   ├── watcher.js`	File watcher daemon
`│   └── helpers.js`	Utility functions
`│`	
`├── templates/`	Handlebars templates
`│   ├── page.hbs`	Main page wrapper
`│   ├── partials/`	
`│   │   ├── header.hbs`	
`│   │   ├── footer.hbs`	
`│   │   └── navigation.hbs`	
`│   └── layouts/`	
`│       ├── alternating-image.hbs`	
`│       ├── feature-grid.hbs`	
`│       ├── text-block.hbs`	
`│       ├── form.hbs`	
`│       └── hero-banner.hbs`	
`│`	
`├── site/`	SITE definition files (READ by build)
`│   ├── config.json`	SITE settings file
`│   ├── _layouts.json`	Page section layouts
`│   ├── _page-templates.json`	Page patterns with navigation
`│   ├── _header-layouts.json`	Page header layouts
`│   ├── _navigation-layouts.json`	Page secondary navigation patterns
`│   └── _menu.json`	Single source of truth for hub and spoke site navigation and sitemap
`│`	

├── src/	CONTENT files (READ by build)
│ └── en/	
│ ├── hub00/	Top level hub and spoke pages
│ │ ├── home.content.json(or .txt)	Page and content defintion pairs used to render final
│ │ ├── home.page.json	HTML page in ./_dist/pages/ directory
│ │ ├── services.content.json	
│ │ ├── services.page.json	
│ │ ├── contact.content.json	
│ │ └── contact.page.json	
│ ├── hub01/	Markets hub and spoke pages
│ │ ├── markets.content.json	
│ │ ├── markets.page.json	
│ │ ├── mlm.content.json	
│ │ ├── mlm.page.json	
│ │ ├── partyplan.content.json	
│ │ ├── partyplan.page.json	
│ │ ├── b2b.content.json	
│ │ ├── b2b.page.json	
│ │ ├── b2c.content.json	
│ │ └── b2c.page.json	
│ └── hub02/	Features hub and spoke pages
│ └── (features files)	
│	
├── styles/	CSS files (COPIED to dist)
│ ├── layout.css	
│ ├── content.css	
│ ├── theme.css	
│ └── responsive.css	
│	
├── scripts/	OLD SPA scripts (REFERENCE ONLY)
│ ├── contentRenderer.js	OLD (not used in build)
│ └── uiHelpers.js	OLD (not used in build)
│	
├── assets/	Images/media (COPIED to dist)
│ └── (images)	
│	
├── index.html	OLD SPA entry (REFERENCE ONLY)
├── _app.js	OLD SPA controller (REFERENCE ONLY)
├── _styles.css	CSS imports (COPIED to dist)
│	

WEBSITE RUNTIME STRUCTURE

`_ips-v4/`	PROJECT ROOT
`└── _dist/`	GENERATED OUTPUT (deployed to server)
`    ├── index.html`	Generated by build (using home page and content defintions)
`    ├── pages/`	
`    │   ├── services.html`	Generated by build
`    │   ├── contact.html`	Generated by build
`    │   ├── markets.html`	Generated by build
`    │   ├── mlm.html`	Generated by build
`    │   ├── partyplan.html`	Generated by build
`    │   ├── b2b.html`	Generated by build
`    │   ├── b2c.html`	Generated by build
`    │   └── … (other site pages)`	Generated by build
`    │`	
`    ├── _styles.css`	Copied from root
`    ├── styles/`	Copied from root
`    │   ├── layout.css`	
`    │   ├── content.css`	
`    │   ├── theme.css`	
`    │   └── responsive.css`	
`    ├── scripts/`	NEW optional enhancement
`    │   └── app.js`	Optional client-side navigation
`    └── assets/`	Copied from root
`        └── (images)`	

📋 QUICK REFERENCE

WHO EDITS WHAT?

File/Folder	Root	Dist	Purpose	Admin	Business User
package.json	✅	❌	NPM config (ROOT)	✅	❌
build/	✅	❌	Build scripts	✅	❌
templates/	✅	❌	Handlebars templates	✅	❌
site/	✅	❌	Config JSON (read by build)	✅	❌
src/	✅	❌	Content JSON/TOML (read by build)	❌	✅
styles/	✅	✅	CSS files (COPY)	✅	❌
scripts/	✅	❌	Legacy SPA (reference)	✅	❌
assets/	✅	✅	Images/media (COPY)	✅	✅
index.html	✅	❌	OLD SPA (reference)	✅	❌
_app.js	✅	❌	OLD SPA (reference)	✅	❌
_styles.css	✅	✅	CSS imports (COPY)	✅	❌
_dist/	✅	N/A	Output directory	❌	❌
_dist/index.html	❌	✅	Generated index page	❌	❌
_dist/_app.js	❌	✅	Optional enhancement (reference)	❌	❌
_dist/_styles.css	❌	✅	COPY Style imports	❌	❌
_dist/pages/	❌	✅	Generated HTML pages	❌	❌
_dist/scripts/	❌	✅	Optional enhancement (COPY)	❌	❌
_dist/styles/	❌	✅	CSS files (COPY)	❌	❌
_dist/assets/	❌	✅	Images/media (COPY)	❌	❌

HOW IT WORKS

Single Page Update (95% of time)		Full Site Rebuild (5% of time)
SME edits mlm.content.json	time-1	Admin edits _layouts.json
↓		↓
Watcher detects change	time-2	Watcher detects global change
↓		↓
Spawns: npm run cms-regen-page:file="mlm.content.json"	time-3	Spawns: npm run cms-regen-site
↓		↓
Worker generates mlm.html	time-4	Worker generates all 50 HTML files
↓		↓
Browser auto-refreshes	time-5	Browser auto-refreshes
Time: ~1-2 seconds		Time: ~15-25 seconds

Website Generator App - Tags,

1. WEBSITE CONFIGURATION (_site.json)

- **Required**: ✅ = Required, ⚪ = Optional

Tag	Definition	Values	Required	Managed By
siteId	Unique website identifier	`String (alphanumeric)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
type	Website category	`"business", "family", "community"`	✅	👨‍💼
tone	Communication style	`"professional", "casual", "fun"`	✅	👨‍💼
theme.mode	Color scheme	`"light", "dark", "seasonal"`	✅	👨‍💼
theme.primaryColor	Brand primary color	`Hex color (#d90429)`	✅	👨‍💼
theme.secondaryColor	Brand secondary color	`Hex color (#a10000)`	✅	👨‍💼
theme.accentColor	Accent/hover color	`Hex color (#ff0000)`	✅	👨‍💼
theme.fontFamily	Typography	`Font stack string`	✅	👨‍💼
branding.siteUrl	Website URL	`URL string`	✅	👨‍💼
branding.siteName	Website name	`String`	✅	👨‍💼
branding.tagLine	Site tagline	`String`	⚪	👨‍💼
branding.logoPath	Logo base URL	`URL string`	⚪	👨‍💼
directories.hubPattern	Hub naming pattern	`"hub{00-99}"`	✅	👨‍💼
directories.maxHubs	Maximum hubs	`Integer (100)`	✅	👨‍💼
directories.paths.styles	Styles root directory	`Path string`	✅	👨‍💼
directories.paths.meta	Framework root directory	`Path string`	✅	👨‍💼
directories.paths.data	Data root directory	`Path string`	✅	👨‍💼
directories.paths.media	Media base URL	`URL string`	✅	👨‍💼
languages.default	Default language	`"en", "es", "fr"`	✅	👨‍💼
languages.supported	Supported languages	`Array of language codes`	✅	👨‍💼
contact.companyName	Company legal name	`String`	✅	👨‍💼
contact.address	Physical address	`String`	✅	👨‍💼
contact.phone	Contact phone	`String`	✅	👨‍💼
contact.email	Contact email	`Email string`	✅	👨‍💼
footer.copyrightYear	Copyright year	`Integer (2025)`	✅	👨‍💼
footer.copyrightHolder	Copyright owner	`String`	✅	👨‍💼

2. LAYOUT TEMPLATES (_layouts.json)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
lastUpdated	Last modification date	`ISO date string (YYYY-MM-DD)`	✅	👨‍💼
layouts.{id}	Unique layout identifier	`"alternating-image", "hero-banner", "feature-grid", etc.`	✅	👨‍💼
layouts.{id}.name	Human-readable name	`String`	✅	👨‍💼
layouts.{id}.description	Layout purpose	`String`	✅	👨‍💼
layouts.{id}.defaultSettings	Default configuration	`Object`	✅	👨‍💼
layouts.{id}.defaultSettings.split	Width split ratio	`"30-70", "40-60", "50-50", "60-40", "70-30"`	⚪	👨‍💼
layouts.{id}.defaultSettings.imageSide	Image position	`"left", "right"`	⚪	👨‍💼
layouts.{id}.defaultSettings.columns	Column count	`1, 2, 3, 4`	⚪	👨‍💼
layouts.{id}.supportedSplits	Valid split values	`Array of split strings`	⚪	👨‍💼
layouts.{id}.supportedSides	Valid side values	`["left", "right"]`	⚪	👨‍💼
layouts.{id}.contentSlots	Required content fields	`Object defining slots`	✅	👨‍💼
contentSlots.{slot}.required	Is field required	`true, false`	✅	👨‍💼
contentSlots.{slot}.type	Field data type	`"text", "richtext", "array", "media", "link"`	✅	👨‍💼

3. PAGE TEMPLATES (_page-templates.json)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
lastUpdated	Last modification date	`ISO date string (YYYY-MM-DD)`	✅	👨‍💼
pageTemplates.{id}	Template identifier	`"alternating-sections-left-start", "hub-with-tab-menu", etc.`	✅	👨‍💼
pageTemplates.{id}.name	Template name	`String`	✅	👨‍💼
pageTemplates.{id}.description	Template purpose	`String`	✅	👨‍💼
pageTemplates.{id}.headerLayout	Header style	`Reference to _header-layouts.json ID`	✅	👨‍💼
pageTemplates.{id}.secondaryNav	Navigation pattern	`Reference to _navigation-layouts.json ID`	✅	👨‍💼
pageTemplates.{id}.contentLayout.pattern	Content pattern	`"repeating-alternating", "dynamic-sections", "static"`	✅	👨‍💼
pageTemplates.{id}.contentLayout.firstSide	Starting image side	`"left", "right"`	⚪	👨‍💼
pageTemplates.{id}.contentLayout.defaultSplit	Default width split	`"30-70", "50-50", "70-30"`	⚪	👨‍💼

4. HEADER LAYOUTS (_header-layou

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
lastUpdated	Last modification date	`ISO date string (YYYY-MM-DD)`	✅	👨‍💼
headerLayouts.{id}	Header style identifier	`"standard", "minimal", "breadcrumb-header", etc.`	✅	👨‍💼
headerLayouts.{id}.name	Header name	`String`	✅	👨‍💼
headerLayouts.{id}.description	Header purpose	`String`	✅	👨‍💼
headerLayouts.{id}.alignment	Text alignment	`"center", "left", "right", "split"`	✅	👨‍💼
headerLayouts.{id}.components	Displayed elements	`Array: ["title", "subtitle", "breadcrumb", "divider", "actions"]`	✅	👨‍💼
headerLayouts.{id}.titleSize	Title font size	`CSS size string ("42px", "2em")`	✅	👨‍💼
headerLayouts.{id}.subtitleStyle	Subtitle styling	`"uppercase-red", "normal", "italic"`	⚪	👨‍💼
headerLayouts.{id}.dividerStyle	Divider line style	`"full-width", "centered", "none"`	⚪	👨‍💼
headerLayouts.{id}.integrated	Integrated with hero	`true, false`	⚪	👨‍💼

5. NAVIGATION LAYOUTS (_navigation-layouts.json)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
lastUpdated	Last modification date	`ISO date string (YYYY-MM-DD)`	✅	👨‍💼
navigationLayouts.{id}	Nav pattern identifier	`"tab-bar-sibling", "sidebar-left", "none", etc.`	✅	👨‍💼
navigationLayouts.{id}.name	Navigation name	`String`	✅	👨‍💼
navigationLayouts.{id}.type	Navigation type	`"secondary", "none"`	✅	👨‍💼
navigationLayouts.{id}.style	Display style	`"horizontal-tabs", "vertical-sidebar", "breadcrumb"`	✅	👨‍💼
navigationLayouts.{id}.position	Screen position	`"below-header", "left", "right", "above-header"`	✅	👨‍💼
navigationLayouts.{id}.sticky	Sticky scroll behavior	`true, false`	⚪	👨‍💼
navigationLayouts.{id}.behavior.showHubLink	Display hub link	`true, false`	⚪	👨‍💼
navigationLayouts.{id}.behavior.showSiblings	Display sibling links	`true, false`	⚪	👨‍💼
navigationLayouts.{id}.behavior.highlightActive	Highlight current page	`true, false`	⚪	👨‍💼
navigationLayouts.{id}.behavior.centerAlign	Center navigation	`true, false`	⚪	👨‍💼
navigationLayouts.{id}.visibility.hubPage	Show on hub pages	`true, false`	✅	👨‍💼
navigationLayouts.{id}.visibility.spokePage	Show on spoke pages	`true, false`	✅	👨‍💼
navigationLayouts.{id}.width	Sidebar width	`CSS size string ("250px")`	⚪	👨‍💼
navigationLayouts.{id}.cssClasses	CSS class names	`Array of strings`	⚪	👨‍💼

6. MENU CONFIGURATION (_menu.json)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
version	Configuration version	`"2.0"`	✅	👨‍💼
lastUpdated	Last modification date	`ISO date string (YYYY-MM-DD)`	✅	👨‍💼
menuItems.{id}	Menu item identifier	`"markets", "features", "services", etc.`	✅	👨‍💼
menuItems.{id}.label	Display name	`String`	✅	👨‍💼
menuItems.{id}.type	Page type	`"hub", "spoke"`	✅	👨‍💼
menuItems.{id}.hub	Parent hub ID	`"hub00", "hub01", "hub02", etc.`	✅	👨‍💼
menuItems.{id}.dataFile	Content file path	`"hub01/markets.json"`	✅	👨‍💼
menuItems.{id}.order	Display order in menu	`Integer`	⚪	👨‍💼
menuItems.{id}.secondaryNav.layout	Nav layout ID	`Reference to _navigation-layouts.json`	✅	👨‍💼
menuItems.{id}.secondaryNav.showOnHub	Show on hub home	`true, false`	⚪	👨‍💼
menuItems.{id}.secondaryNav.showOnSpokes	Show on spoke pages	`true, false`	⚪	👨‍💼
menuItems.{id}.defaultSection	Default hub section	`String (section ID)`	⚪	👨‍💼
menuItems.{id}.children	Child pages (spokes)	`Array of page objects`	⚪	👨‍💼
children[].id	Child page ID	`String`	✅	👨‍💼
children[].label	Child display name	`String`	✅	👨‍💼
children[].dataFile	Child content file	`"hub01/mlm.json"`	✅	👨‍💼

7. CONTENT FILE (*.content.json) and toml as (*.content.txt)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👤
contentId	Unique content identifier	`String matching filename`	✅	👤
language	Content language	`"en", "es", "fr"`	✅	👤
hub	Hub directory	`"hub00", "hub01", "hub02"`	✅	👤
lastUpdated	Last edit date	`ISO date string (YYYY-MM-DD)`	✅	👤
author	Content author	`String`	⚪	👤
blocks	Content blocks container	`Object`	✅	👤
blocks.{blockId}	Individual content block	`Unique block identifier`	✅	👤
blocks.{blockId}.id	Block identifier	`String (matches key)`	✅	👤
blocks.{blockId}.source	Source of content block	`"file", "database" (Defaults to "file" if omitted.)`	⚪	👤
blocks.{blockId}.title	Section title	`String`	✅	👤
blocks.{blockId}.paragraph	Main text content	`String (plain text or markdown)`	⚪	👤
blocks.{blockId}.listItems	Bullet point list	`Array of strings`	⚪	👤
blocks.{blockId}.media	Media object	`Object`	⚪	👤
blocks.{blockId}.media.type	Media type	`"image", "carousel", "video"`	✅	👤
blocks.{blockId}.media.src	Image filename	`"165507.jpg"`	✅	👤
blocks.{blockId}.media.alt	Alt text	`String`	✅	👤
blocks.{blockId}.callToAction	Link object	`Object`	⚪	👤
blocks.{blockId}.callToAction.text	Link text	`"READ MORE", "Learn More"`	✅	👤
blocks.{blockId}.callToAction.targetId	Link destination	`Page ID ("mlm", "features")`	✅	👤

8. PAGE CONFIGURATION (*.page.json)

Tag	Definition	Values	Required	Managed By
siteId	Website identifier	`String (matches _site.json)`	✅	👨‍💼
pageId	Unique page identifier	`String matching filename`	✅	👨‍💼
pageType	Page classification	`"hub", "spoke"`	✅	👨‍💼
language	Page language	`"en", "es", "fr"`	✅	👨‍💼
hub	Hub directory	`"hub00", "hub01", "hub02"`	✅	👨‍💼
pageTemplate	Page template ID	`Reference to _page-templates.json`	✅	👨‍💼
headerLayout	Header layout ID	`Reference to _header-layouts.json`	✅	👨‍💼
secondaryNav	Navigation layout ID	`Reference to _navigation-layouts.json`	✅	👨‍💼
contentSource	Content file reference	`Content file ID (without extension)`	✅	👨‍💼
metadata	Page metadata	`Object`	✅	👨‍💼
metadata.title	Page title	`String`	✅	👨‍💼
metadata.subtitle	Page subtitle	`String`	⚪	👨‍💼
metadata.description	SEO description	`String`	⚪	👨‍💼
metadata.keywords	SEO keywords	`Array of strings`	⚪	👨‍💼
contentBlocks	Ordered content list (simple ten	`Array of block IDs from content file`	⚪*	👨‍💼
sections	Section definitions (manual conti	`Array of objects`	⚪*	👨‍💼
sections[].sectionId	Section identifier	`String`	✅	👨‍💼
sections[].layoutTemplate	Section layout	`Reference to _layouts.json`	✅	👨‍💼
sections[].layoutSettings	Layout overrides	`Object`	⚪	👨‍💼
sections[].layoutSettings.split	Width split override	`"30-70", "50-50", "70-30"`	⚪	👨‍💼
sections[].layoutSettings.imageSide	Image position override	`"left", "right"`	⚪	👨‍💼
sections[].contentMapping	Content-to-layout map	`Object`	✅	👨‍💼
sections[].contentMapping.contentBlock	Block ID reference	`String from content file`	✅	👨‍💼
sections[].contentMapping.slots	Slot assignments	`Object mapping slots to content`	✅	👨‍💼

Note: Use contentBlocks OR sections, not both. Required field depends on pageTemplate.contentLayout.pattern:

- Use contentBlocks when pattern = "repeating-alternating" (auto-generates sections)

- Use sections when pattern = "dynamic-sections" (manual section control)

9. COMMON PATTERNS BY FILE TYPE

Patten Name	Content File	Page File
Simple Spoke Page (e.g., Services)	Pure content blocks	`pageTemplate: "alternating-sections-right-start"`
		`headerLayout: "standard"`
		`secondaryNav: "none"`
		`contentBlocks: ["block1", "block2", "block3"]`
Hub Page (e.g., Markets)	Overview content blocks	`pageTemplate: "alternating-sections-left-start"`
		`headerLayout: "standard"`
		`secondaryNav: "none"`
		`contentBlocks: ["mlm-overview", "partyplan-overview", "b2b-overview"]`
Spoke with Sibling Nav (e.g., MLM)	Detail content blocks	`pageTemplate: "alternating-sections-right-start"`
		`headerLayout: "standard"`
		`secondaryNav: "tab-bar-sibling"`
		`contentBlocks: ["mlm-detail-1", "mlm-detail-2"]`
Hub with Tab Menu (e.g., Features)	Multiple section groups	`pageTemplate: "hub-with-tab-menu"`
		`headerLayout: "standard"`
		`secondaryNav: "tab-bar-hub-sections"`
		`sections: { "home": [...], "frontend": [...], "backoffice": [...] }`

10. VALIDATION RULES

Check	Rule	Error Message
Site consistency	siteId matches across all files	`"Site ID mismatch - file belongs to different project"`
File naming	{pageId}.content.json matches contentId	`"Content ID must match filename"`
File naming	{pageId}.page.json matches pageId	`"Page ID must match filename"`
Hub reference	hub value exists in _site.json definitions	`"Invalid hub reference"`
Template reference	pageTemplate exists in _page-templates.json	`"Template not found"`
Layout reference	layoutTemplate exists in _layouts.json	`"Layout not found"`
Content reference	All contentBlocks exist in content file	`"Content block not found"`
Required slots	All required slots defined in layout are mapped	`"Missing required content slot"`
Image side	imageSide is "left" or "right"	`"Invalid imageSide value"`
Split ratio	split matches pattern \d+-\d+ and sums to 100	`"Invalid split ratio"`

11. DEPRECATED ASSETS (Phase 1 Migration)

Tags/Patterns No Longer Used

Old Tag/Pattern	Used In	Replaced By	Migration Notes
commonData	Old JSON structure	`Content blocks in *.content.json`	`Extracted to separate content files; shared content now managed through content block`
imageSide: "image-left"	Old section definitions	`imageSide: "left"`	`Simplified to "left"/"right" only for consistency`
imageSide: "image-right"	Old section definitions	`imageSide: "right"`	`Simplified to "left"/"right" only for consistency`
Embedded media base paths	Old JSON files	`_site.json → directories.paths.media`	`Centralized in site configuration for easier updates`
pageType: "spoke" without hub	Old menu structure	`hub: "hub00" required`	`All pages must explicitly specify hub directory`
Inline carousel configurations	Old sections	`Standardized media object`	`All media now uses consistent structure with type, src, alt`
Mixed content/layout JSON	Old single-file structure	`Separated .content.json + .page.json`	`Clean separation of concerns achieved`
tagline (lowercase)	_site.json	`tagLine (camelCase)`	`Standardized casing for consistency`

Files No Longer Used

Old File	Purpose	Replaced By	Status
markets.json (single file)	Combined content + layout	`markets.content.json + markets.page.json`	`Keep for Phase 1 compatibility`
services.json (single file)	Combined content + layout	`services.content.json + services.page.json`	`Keep for Phase 1 compatibility`
features2.json	Test/alternate features page	`features.content.json + features.page.json`	`Can be deleted after migration`
Old _menu.json format	Navigation without hub references	`New _menu.json with hub structure`	`Deprecated - do not use`

Structural Changes

Old Pattern	New Pattern	Reason
Single JSON per page	Two files: .content.json + .page.json	`Separation of content from layout`
Layout embedded in content	Layout defined in page templates	`Reusability and consistency`
No site-level configuration	_site.json with global settings	`Centralized configuration`
Ad-hoc section layouts	Defined in _layouts.json	`Standardized components`
Inline navigation config	_navigation-layouts.json patterns	`Reusable navigation templates`

Website Generator App - Cost Analysis

Analyze page generation cost and pros/cons for the options:

a) dynamically generate the page on demand
b) generate once then serve static pages until something changes to force page regeneration
c) an alternate solution

Option A: Dynamic Generation (Client-Side SPA)

Architecture:

```
User Request → index.html + app.js → Load 6 config files → Parse JSON →
Generate HTML → Render to DOM
```

Cost Breakdown:

Factor	Cost	Impact
Initial Load	6-8 HTTP requests for config files (~50-200KB total)	⚠️ Slow first page load (1-2 seconds)
Every Page Switch	Parse JSON + render HTML (10-50ms)	✅ Fast navigation within site
Browser Work	JavaScript execution (heavy)	⚠️ Poor on low-end devices
SEO	Content not in HTML source	❌ Poor Google indexing
Server Load	Minimal (just serving files)	✅ Very light server
Caching	Must re-download on browser close	⚠️ Repeat visitors re-download
Updates	Immediate (edit JSON, refresh)	✅ No build step

Performance Profile:

First Visit:
- Load index.html: 50ms
- Load 6 config files: 300ms (6 requests × 50ms)
- Load app.js: 100ms
- Parse all JSON: 50ms
- Render home page: 30ms
- TOTAL: ~530ms ⚠️

Subsequent Pages (same session):
- Parse page JSON: 10ms
- Render content: 20ms
- TOTAL: ~30ms ✅

Return Visit (closed browser):
- Same as First Visit: ~530ms ⚠️

User Experience:

First Load: [████████░] Slow
Navigation: [██░░░░░░░░] Fast
Return Visit: [████████░] Slow
Mobile/Tablet: [████████░] Slow (more JS parsing)

Option B: Static Site Generation (Pre-Built HTML)

Architecture:

```
Build Time: Load configs → Generate all pages → Write HTML files
User Request: index.html (already rendered) → Fast display
```

Cost Breakdown:

Factor	Cost	Impact
Build Time	5-30 seconds (generate all pages)	⚠️ Developer wait time
Initial Load	1 HTTP request for full HTML (~50-100KB)	✅ Very fast first page
Every Page Switch	1 HTTP request for next page	✅ Fast, but full page reload
Browser Work	Minimal (just render HTML)	✅ Great on all devices
SEO	Full HTML in source	✅ Perfect Google indexing
Server Load	Minimal (static files)	✅ Very light server
Caching	Browser + CDN caching	✅ Extremely fast repeat visits
Updates	Must rebuild + deploy	⚠️ Extra step to publish changes

Performance Profile:

First Visit:
- Load home.html: 80ms (includes all content)
- Render HTML: 20ms
- TOTAL: ~100ms ✅

Subsequent Pages:
- Load services.html: 70ms
- Render HTML: 15ms
- TOTAL: ~85ms ✅ (full page refresh)

Return Visit (browser cache):
- Load from cache: 0ms
- Render: 15ms
- TOTAL: ~15ms ✅✅✅

User Experience:

First Load: [██░░░░░░░░] Fast

Navigation: [███░░░░░░░] Fast (but page flash)

Return Visit: [█░░░░░░░░░] Very Fast (cached)

Mobile/Tablet: [██░░░░░░░░] Fast (no JS parsing)

Option C: Hybrid Approach (Recommended)

Architecture:

```
Build Time:
      Generate HTML with embedded JSON
User Request:
      HTML loads instantly
      + Progressive enhancement via JS
```

Benefits:

✅ **Instant first render** (HTML already there)

✅ **SEO perfect** (Google sees full HTML)

✅ **Fast navigation** (JS can intercept clicks for SPA-style)

✅ **Works without JS** (fallback to standard links)

✅ **Best of both worlds**

User Experience:

First Load	[▉▒▒▒▒▒▒▒▒▒] Very Fast (HTML renders immediately)
Navigation	[▉▉▒▒▒▒▒▒▒▒] Fast (SPA-style if JS enabled)
Return Visit	[▉▒▒▒▒▒▒▒▒▒] Very Fast (cached HTML)
Mobile/Tablet	[▉▒▒▒▒▒▒▒▒▒] Very Fast (HTML-first)

```
<!-- index.html (generated at build time) -->
<!DOCTYPE html>
<html>
<head>
  <title>inPowerSuite</title>
  <script type="application/json" id="site-
config">
    { "siteId": "inpowersuite", ... }
  </script>
  <script type="application/json" id="page-
data">
    { "pageId": "home", "contentBlocks": [...]
}
  </script>
</head>
<body>
  <!-- Pre-rendered HTML -->
  <header class="app-header">
    <h1>inPowerSuite</h1>
  </header>
  <main class="page-content">
    <section class="static-section">
      <h2>Niche Group</h2>
      <p>Recipient of...</p>
    </section>
  </main>
  <footer>...</footer>

  <!-- Optional: Progressive enhancement -->
  <script src="app.js" defer></script>
</body>
</html>
```

Cost Comparison Table

Metric	Dynamic (SPA)	Static (SSG)	Hybrid
First Load (Cold)	530ms	100ms	120ms
First Load (Warm)	530ms	15ms	20ms
Navigation	30ms	85ms	35ms
SEO Score	Poor (3/10)	Perfect (10/10)	Perfect (10/10)
Mobile Performance	Fair (5/10)	Excellent (10/10)	Excellent (9/10)
Update Process	Edit JSON → Refresh	Edit → Build → Deploy	Edit → Build → Deploy
Update Time	0 seconds	10-30 seconds	10-30 seconds
Server CPU	Low	None	Low
CDN Cacheable	Config files only	All pages	All pages
Works Offline	No (needs configs)	Yes (cached HTML)	Yes (cached HTML)