Cartnet — The Web, in Cartridges

Version 0.1 — discussion draft | November 23, 2026

Developed by Pablo Murad — November 23, 2026 — pablo@pablomurad.com — https://rsw.works

Experimental Project Specification. This document is an experimental specification of the Really Small Web project. It is a work in progress and has not been published as an RFC or approved as an Internet Standard.

License: specifications CC0 1.0 (public domain); future reference implementation and official tools MIT.

CARTNET — THE WEB, IN CARTRIDGES

Experimental conception and architecture document

Version: 0.1 — discussion draft

Date: November 23, 2026

Developed by Pablo Murad

pablo@pablomurad.com

Provisional name: Cartnet

Format: Web Cartridge

Protocol: Cart Protocol (provisional name)

Reader: Cart Browser

Command-line tool: Cart CLI

This document gathers and develops the ideas discussed about a portable, collectible, interconnected web independent of a mandatory central server. The technical decisions are proposals, not a definitive specification nor an existing implementation.

01. The Central Idea

What if an entire website could be distributed as a PNG image?

Not a screenshot, but an image containing the site itself: its texts, pages, styles, resources, identity, code, and links.

A person could save that image, send it by email, host a copy, share it on a forum, or archive it on their computer. Opening it in a compatible reader would make the content navigable, even without an internet connection.

The idea is inspired by PICO-8 cartridges, self-contained files, webrings, and the Small Web sharing culture.

The next step is to let these cartridges discover other cartridges, exchange messages, receive updates, and participate in a distributed network. It is not the PNGs that communicate directly: it is the readers and services that interpret the cartridges and implement the protocol.

The result would be a web composed of transportable digital objects, which can be collected, copied, preserved, and shared.

One sentence summarizes the proposal:

```
The web, in cartridges.
```

02. Philosophy and Objectives

- Offline-first: the basic content must work without the internet.
- Decentralization: no central server should be mandatory.
- Portability: the cartridge must be a copyable, preservable file.
- Domain-independent identity: verifiable authorship through keys.

- Public specification, documented and implementable by third parties.
- Simplicity, low consumption, and compatibility between implementations.
- Privacy and security by default, with explicit permissions.
- Freedom to publish, host, archive, and choose readers.
- Compatibility with the conventional web, when it makes sense.
- No mandatory blockchain, cryptocurrency, or proprietary platform.
- Personal and experimental aesthetics, inspired by the Small Web, IndieWeb, tildeverse, pubnix, the demoscene, and video-game cartridge culture.

The image is not the most efficient way to distribute files. A pure ZIP would suffice. The choice of PNG has a cultural and functional purpose: offering a recognizable visual cover that also transports a functional digital object, and that can be opened as an ordinary image.

03. How Data Lives Inside the PNG

The PNG is organized into blocks called chunks. The initial signature identifies the format; IHDR describes the image; IDAT contains the pixels; IEND ends the file. Other blocks can store additional information that an ordinary viewer does not need to interpret.

Two relevant techniques:

A. Data in chunks. tEXt, zTXt, and iTXt store textual metadata. Private ancillary chunks can hold arbitrary binary data, such as a ZIP. A PNG reader that does not understand an ancillary chunk can ignore it. The pixels do not need to be modified.

B. Data in the pixels (LSB). The least significant bits of the color channels can represent additional information. PICO-8 uses two bits of each of the four RGBA channels, storing one byte per pixel. Its 160 x 205 pixel cartridge offers 32,800 bytes of raw capacity in the pixels, although there are internal limits of its own for the game's data and code.

For Cartnet, technique A is initially recommended: a private ancillary chunk containing a ZIP. The LSB technique can be explored as an experimental option for small, artistic, or fixed-size cartridges.

An example of a provisional chunk name is caRT. The first lowercase character indicates an ancillary chunk; the second, a private chunk; the third must be uppercase (reserved bit); the fourth uppercase indicates that editors not recognizing it should not copy it indiscriminately when modifying critical chunks. The final definition requires precise documentation and compatibility tests.

The structural PNG limit is 2,147,483,647 bytes of data per chunk. This does not mean a reader should accept cartridges of that size. Practical limits must be much smaller and configurable.

Initial proposal: 10 MiB for the complete PNG, 32 MiB of expanded ZIP content, and 512 files per cartridge. These are illustrative limits, to be validated with the use cases. The reader must also limit image dimensions, processing time, and memory consumption.

IMPORTANT: social networks, CDNs, and editors can remove chunks or alter pixels. A PNG displayed correctly may no longer contain the cartridge. Distributing the original file and verifying its integrity are fundamental requirements. A separate ZIP backup can be useful.

04. Anatomy of a Web Cartridge

Example file name: midnight.cart.png

```
midnight.cart.png
■■■ Cover image visible normally
■■■ Private chunk with payload.zip
    ■■■ manifest.json
    ■■■ index.html
```

```
■■■ style.css
■■■ app.js              (optional, later phases)
■■■ signature.sig
■■■ assets/
■    ■■■ logo.svg
■    ■■■ photo.jpg
■■■ data/
     ■■■ posts.json
```

The manifest describes the cartridge. The remaining files represent its contents. The reader verifies and opens the entrypoint.

In the first version, HTML and CSS are sufficient. JavaScript can be included later, under an appropriate isolation policy.

Local resources use relative paths. The reader must not depend on remote fonts, scripts, or images for the basic offline experience. Mutable data, such as messages and user preferences, must live in separate storage, not silently modify a cartridge signed by the author.

05. Manifest and Metadata

Illustrative manifest.json example:

```
{
  "protocol": "cart/1.0",
  "name": "midnight-journal",
  "title": "The Midnight Journal",
  "description": "A little corner of the network.",
  "type": "website",
  "version": 12,
  "author": "pablo",
  "public_key": "ed25519:CHAVE_PUBLICA_AQUI",
  "entrypoint": "index.html",
  "created_at": "2026-11-23T00:00:00Z",
  "license": "CC-BY-4.0",
  "previous_release": "sha256:HASH_DA_VERSAO_ANTERIOR",
  "peers": [
    "https://example.org/alice.cart.png",
    "https://example.net/bob.cart.png"
  ],
  "capabilities": ["discovery", "updates"],
  "permissions": [],
  "tags": ["journal", "smallweb", "photography"]
}
```

The final schema must separate mandatory and optional fields, define UTF-8 encoding, limits, accepted types, and version compatibility.

Capabilities indicate features offered by the cartridge. Permissions represent accesses requested from the reader. An announced capability does not automatically grant a permission.

Possible types: website, profile, blog, zine, library, game, forum, application, archive, catalog, and others defined by the community.

The signature cannot be computed over a ZIP that contains the signature itself, as that creates a circular dependency. It is recommended to sign a manifest in canonical representation that includes the file list and their hashes, excluding the signature file. The final format must define exactly which bytes are signed, how to reject extra files, and how to treat duplicate names and invalid paths.

06. Identity, Addressing, and Versioning

An Ed25519 identity can allow the author to sign cartridges without their identity depending exclusively on a DNS domain.

Conceptual address of a cartridge:

```
cart://&lt;public-key&gt;/midnight-journal
```

Conceptual address of an immutable version:

```
cart://sha256/&lt;payload-hash&gt;
```

The first form identifies the project across its updates. The second identifies specific bytes of a package. They are not URLs recognized by conventional browsers without a handler or gateway.

A signed publication record could bind the author's key, the project name, the version number, the payload hash, and its URLs or other distribution locations.

Cartridges with the same name but different keys are independent projects. Friendly names are aliases, not identity proof.

Each new version can reference the previous one, creating a history chain. The reader must flag old or conflicting versions, reject unauthorized updates, and avoid automatic regressions. A valid signature proves integrity and control of the key used; it does not prove that the author is trustworthy or that the code is safe.

The payload hash and the whole-PNG hash are distinct concepts: the first identifies the transported package; the second identifies the exact image file. The protocol must say which is used in each operation. If the cover needs to be authenticated, it must also be bound to the signature by an explicit rule, without creating circularity.

Key management, rotation, recovery, and revocation need to be designed. Initially, one key per author and a local trust model are sufficient for experimentation. Key loss cannot be magically solved by a decentralized system.

07. The Cartridge Network

Cartridges are passive objects. The Cart Browser, the Cart CLI, and optional services are the network's active participants.

Each reader owns a local cartridge library and can know other participants, locate publications, download new versions, send messages, and replicate content, according to the chosen policy.

Example topology:

```
[Reader A: blog + profile] ---- [Optional relay]
        |                              |
[HTTP server A]              [Reader B: zine]
        |                              |
        +------ [Mirror C] --------+
```

The network can evolve in layers:

Layer 1: links and HTTP(S). Cartridges announce URLs of other cartridges. The reader downloads the file, verifies the content, and adds it to the library. It is a kind of dynamic webring. It requires no own protocol server.

Layer 2: signed records and discovery. Servers or relays announce identities, versions, hashes, and download locations. A reader queries announcements and resolves the newest version it can verify. It can use several independent servers.

Layer 3: communication and synchronization. Readers exchange signed events, follow authors, receive comments and messages, and synchronize content according to the offered capabilities.

Layer 4: optional P2P replication. Participants distribute verifiable copies among themselves, without depending exclusively on the original server. This requires discovery, connectivity, storage, limits, and replication policies.

The protocol must distinguish location, identity, and content. An HTTP address can change without changing the author's identity; the same version can be hosted in many places.

Availability is not guaranteed by decentralization alone. If nobody keeps an accessible copy, the content becomes unavailable through the network, although offline copies may still exist.

08. Discovery and Navigation

A cartridge could announce relations such as:

- Other cartridges by the same author.
- Friends and recommended sites.
- Dependencies or complementary resources.
- Community directories, webrings, and collections.
- Cartridges that cite or comment on its publications.

The reader would present these relations visually. The user could open a stored cartridge or search for a copy on the network.

An initial discovery network can be built with static JSON files published over HTTPS. Optional directories can index manifests, categories, tags, and descriptions.

Automatic discovery does not imply automatic download or execution. The reader must allow inspection, confirmation, and limits, including not to turn links into involuntary tracking or abusive consumption.

09. Communication Between Cartridges

Conceptual set of operations (not commands standardized today):

```
DISCOVER   Locate new cartridges and identities.
FETCH      Obtain an identified cartridge or resource.
HELLO      Present an identity or session.
ANNOUNCE   Publish information about a cartridge or version.
FOLLOW     Follow an author or project.
MESSAGE    Send a message to a recipient.
COMMENT    Publish a comment linked to specific content.
SYNC       Synchronize events, versions, and records.
VERIFY     Verify identity and integrity locally.
```

Transport can be HTTPS, WebSocket, or later a P2P network. The operations are semantic; they need not correspond to HTTP verbs or literal commands on a network connection.

Conceptual event example:

```
{
  "protocol": "cart/1.0",
  "type": "message",
  "id": "HASH_DO_EVENTO",
  "from": "CHAVE_PUBLICA_DO_REMETENTE",
  "to": "CHAVE_PUBLICA_DO_DESTINATARIO",
  "created_at": "2026-11-23T18:40:00Z",
  "body": "hello from another cartridge",
  "signature": "ASSINATURA_DO_EVENTO"
}
```

The definitive schema must define canonicalization, signature, covered fields, identifiers, limits, deduplication, repeated-event handling, and offline behavior.

Signature is not encryption. Private messages require their own end-to-end encryption mechanism and key management. One must not invent an ad hoc cryptographic construction.

Events are separate from the original cartridge. A comment or message does not rewrite another person's PNG. Readers can store events locally, forward them, and display them alongside the referenced content.

The application communication API must be mediated by the reader. A cartridge requests a limited operation, and the reader decides whether to execute it. The cartridge must not directly access the user's private keys or internal network.

10. Cart Browser

The Cart Browser would be a reader and library manager, not necessarily a browser built from scratch.

Possible interface:

```
+-----------------------------------------------------------+
| &lt;  &gt;  Update | cart://author/midnight-journal            |
+-----------------+-----------------------------------------+
| CART LIBRARY    | THE MIDNIGHT JOURNAL                    |
|                 |                                         |
| [ ] Journal     | Cover, text and cartridge content       |
| [ ] Alice       |                                         |
| [ ] Archives    | Signature: valid | Version: 12          |
| [ ] Zines       |                                         |
|                 | Connected cartridges: Alice, Archives   |
| Import PNG      |                                         |
| Discover network|                                         |
+-----------------+-----------------------------------------+
| Offline ready | 4 local cartridges | Network connected   |
+-----------------------------------------------------------+
```

Features:

- Import files by drag-and-drop or by entering a URL.
- Display cover, name, description, authorship, and version.
- Inspect manifest, integrity, signature, and permissions.
- Navigate internal pages and links between cartridges.
- Keep a local library, tags, favorites, and history.
- Export or share the original PNG.
- Discover new cartridges through links and directories.
- Update cartridges manually or under an authorized policy.
- Preserve old versions when desired.
- Display optional comments, messages, and notifications.
- Allow offline reading and indicate which resources require a connection.

Possible stack: Tauri 2 + Rust + HTML/CSS/JS for the interface + SQLite for the local catalog. The web mechanism must separate the reader's privileged interface from untrusted content rendering. Tauri commands must not be granted to imported content.

A static web reader can also be created for HTML/CSS-only cartridges, with local inspection and rendering in the browser. A CLI reader would allow inspecting, verifying, and extracting packages without a graphical interface.

11. Cart CLI

Command-line tool for creation, publication, and verification.

Examples of a future interface (commands not yet implemented):

```
cart init midnight-journal
cart keygen
cart pack ./midnight-journal --cover cover.png --out journal.cart.png
cart inspect journal.cart.png
cart verify journal.cart.png
cart extract journal.cart.png --out ./extracted
cart fetch https://example.org/journal.cart.png
cart publish journal.cart.png
cart discover journal.cart.png
cart list
cart update midnight-journal
```

Rust is a good option for distributing a standalone binary and using established PNG, ZIP, hash, signature, and HTTP libraries. A Python prototype would also allow quickly validating the format. The protocol's core library must be independent of the interface.

12. Security: A Fundamental Requirement

An ordinary PNG containing JavaScript does not automatically execute that code. Risk arises if a program extracts and executes content, misinterprets a file, or contains security flaws.

As the Cart Browser intends to interpret third-party content, any received cartridge must be considered untrusted, even if it has a valid signature.

Security proposal:

- First version: static HTML and CSS; JavaScript disabled.
- Validate the PNG structure, chunks, and sizes before processing.
- Verify hashes, signature, and manifest schema.
- Limit compressed size, expanded size, files, dimensions, memory, time, and operation rate.
- Reject absolute paths, ../, dangerous symbolic links, duplicate names, and other means of escaping the extraction directory.
- Prevent ZIP bombs and images with excessive resource consumption.
- Render content in an environment separate from the privileged interface.
- Disable external network and scripts by default; block remote resources, forms, navigations, and other unauthorized exits.
- Apply sandbox, Content Security Policy (CSP), and origin isolation.
- If JavaScript is added, offer minimal, mediated APIs; do not expose system commands, filesystem, private keys, or broad IPC.
- Separate each cartridge's local data from the reader's global catalog.
- Require consent for sensitive capabilities, without accepting security decisions coming from the cartridge's own manifest.
- Confirm actions such as publication, message sending, and downloads.
- Prevent downloads started by the reader from reaching internal services or private addresses without explicit authorization; handle redirects.
- Do not execute code during package import, verification, or extraction.
- Keep dependencies updated and test malformed files (fuzzing).

Conceptual permissions:

```
render.static       HTML/CSS and internal resources.
script.execute      Restricted JavaScript (future).
storage.local       The cartridge's own isolated storage.
network.discover    Cartridge discovery, mediated by the reader.
network.message     Authorized message sending.
network.external    Limited, authorized external connections.
```

Terminal access, private keys, and personal files must not be offered to ordinary cartridges. An eventual file selection must be specific, user-initiated, and limited to the selected item.

A sandboxed webview is not by itself a complete guarantee. The architecture needs to be tested on each platform and reviewed before arbitrary applications can be safely executed.

When hosting PNGs on the web, serve them with Content-Type: image/png, X-Content-Type-Options: nosniff, and preferably separate user-provided content from the authenticated-services origin. Do not trust the file extension as validation. Offer the original for download.

13. Publication and Updates

Author publication flow:

1. Create a self-contained site in a directory.
2. Choose or create the cover PNG image.
3. Generate the manifest with version and identity.
4. Compute hashes and sign the representation defined by the protocol.
5. Package the files into a ZIP.
6. Insert the payload into the PNG using the defined private chunk.
7. Validate the result in the Cart CLI and in the reader.
8. Host the PNG on a static HTTP server.
9. Optionally publish a signed announcement record of the version.
10. Share the address or the file directly.

Reader flow:

1. Receive the PNG or find it through an announcement.
2. Validate limits, structure, and content.
3. Verify signature and integrity.
4. Present identity, trust, and requested permissions.
5. Store a local copy without altering the original.
6. Render the content in an isolated environment.
7. Check for updates only according to the user's preference.

Updates must be received as new complete versions in the MVP. Deltas, selective synchronization, and block distribution may be introduced only if the need justifies the complexity.

14. Existing Technologies and Inspirations

There is no need to reinvent all internet layers.

```
PICO-8
https://www.lexaloffle.com/pico-8.php
Direct inspiration: PNG cartridges containing games and resources.

PNG Specification
https://www.w3.org/TR/png/
Normative base for the container structure and chunks.

Excalidraw
https://excalidraw.com/
Example of an exported image that can carry an editable scene.

TiddlyWiki
https://tiddlywiki.com/
Self-contained documents and applications, useful offline inspiration.

Nostr
https://github.com/nostr-protocol/nips
Signed events, identities, and relay-based communication. Could serve
as an optional transport for announcements, comments, and messages.
Nostr uses secp256k1 keys and Schnorr signatures, not Ed25519. An
integration requires separate identities or a verifiable binding,
without presuming that keys or signatures are interchangeable.

Blossom
https://github.com/hzrd149/blossom
Hash-identified blob distribution, with Nostr integration.
Can serve as optional storage of the original PNGs.
```

```
IPFS / IPNS
https://ipfs.tech/
https://docs.ipfs.tech/concepts/ipns/
Content-addressed distribution and signed mutable references.

Hypercore
https://github.com/holepunchto/hypercore
P2P replication of verifiable logs and data.

IndieWeb
https://indieweb.org/
Content ownership, personal publishing, and linking principles.

Webrings, tildeverse, pubnix, Gemini, and Gopher are also cultural
inspirations for a small-scale network and independent communities.
```

The existing Nostr and Blossom infrastructure can be used in a future experiment, without making it a mandatory dependency of the format. The MVP must work with local files and common HTTP servers.

15. Use Cases

A. PORTABLE BLOG A PNG containing a blog or a personal journal edition. The reader can navigate articles, photographs, and references offline.

B. PROFILE / DIGITAL IDENTITY Cartridge with biography, links, public keys, contact means, and identity proofs. Never include private keys in the public package.

C. ZINES AND MAGAZINES An illustrated cover carrying all texts and resources of an edition. The collection of images becomes a library of publications.

D. ANSI / ASCII ARCHIVE An image displays the rendered art and transports the original ANSI file, SAUCE information, description, authorship, and additional resources. An application especially interesting for ASCII.zone.

E. SMALL APPLICATIONS AND GAMES Tools, interactive experiences, games, and small web programs executed inside a restricted environment, when the runtime allows.

F. DIGITAL POSTCARD An image containing a letter. The content can be separately encrypted with age or OpenPGP and opened by the authorized recipient. The cover remains public. The PNG offers no secrecy by itself.

G. DISTRIBUTED COMMUNITY Profile and publication cartridges, with mutual discovery, comments, followers, and messages through separate, verifiable events.

H. LIBRARIES AND DIGITAL PRESERVATION Collections of cartridges preserved on independent computers and servers. The local library allows consultation even if the original server ceases to exist. Replication and licenses must be respected.

I. CATALOGS, DIRECTORIES, AND WEBRINGS A cartridge gathers references to other cartridges, collections, or communities. It can be updated without requiring a central platform.

16. Implementation Roadmap

PHASE 1 — CARTRIDGE FORMAT

Objective: prove that an image can carry an offline site.

- Document the initial format version, manifest, and private chunk.
- Create minimal packer and extractor in Python or Rust.
- Use an internal ZIP and a conventional PNG cover.
- Create real example cartridges with HTML/CSS, without scripts.
- Define limits, hashes, signature, and validation rules.

- Write round-trip tests and invalid files.

Criterion: an original file can be transported, validated, extracted, and read offline, without altering its cover appearance.

PHASE 2 — CART CLI

Objective: let third parties create and publish cartridges.

- Implement init, pack, inspect, verify, and extract.
- Generate and protect local keys.
- Add fetch and static HTTP(S) publication.
- Create valid and invalid fixtures for interoperability.
- Document commands, format, and public examples.

Criterion: two users can create and verify their own cartridges, without depending on a central platform.

PHASE 3 — CART BROWSER

Objective: turn cartridges into a navigable experience.

- SQLite library with import, favorites, and history.
- Tauri interface or local web prototype.
- Isolated rendering of static content.
- Navigation between local and remote cartridges.
- Visual inspection of authenticity and permissions.
- Security and isolation tests before enabling JavaScript.

Criterion: any user can open a compatible PNG and navigate the stored cartridges even without internet.

PHASE 4 — CART NETWORK

Objective: connect the cartridges into a distributed publication network.

- Discovery through links, directories, and signed announcements.
- Resolution of identities and versions.
- Verified updates, mirrors, and authorized replication.
- Signed events for comments and messages.
- Moderation, blocking, spam, and usage-limit policies.
- Optional adapters for Nostr, Blossom, or other networks.

Criterion: two independent readers discover cartridges, verify publications, and exchange events without requiring a centralized account.

PHASE 5 — RUNTIME AND EXPERIMENTS

Objective: offer interactive applications and advanced features.

- Restricted JavaScript and documented capabilities APIs.
- Reader-mediated messaging between cartridges.
- Persistent, isolated storage.
- Optional P2P replication and more efficient synchronization.
- Alternative readers, gateways, and community tools.

Criterion: applications from different authors coexist without accessing each other's resources or the system's outside authorized permissions.

17. Recommended MVP: Start Small

Do not start with DHT, blockchain, an own transport protocol, chat, message encryption, or a complete browser.

The first experiment must have only:

- A PNG packer and extractor with an internal ZIP.
- A small, documented manifest.
- Self-contained HTML/CSS sites.
- Verifiable integrity and signature.
- A static-content reader with isolation.
- Links between cartridges downloaded or hosted via HTTPS.
- Three demonstration cartridges, created by distinct identities.

Example: a personal blog, a zine, and a directory referencing both. Publish the PNGs on different servers and demonstrate that the reader keeps navigating after disconnecting from the internet.

The first network is born from the links. Real-time communication is not necessary to validate the concept of a portable, interconnected web.

18. Open Questions and Limitations

- Definitive name and availability of domain, trademarks, and repositories.
- Exact choice and documentation of the chunk and payload structure.
- Manifest canonicalization, signature, and content identification.
- How to bind cover, content, and identity verifiably.
- Cartridge size limits and external-resource policy.
- cart:// address resolution and discovery bootstrap.
- How to detect conflicting updates and resolve key rotation.
- How to encourage replication without replicating content indiscriminately.
- How to reduce spam, abuse, tracking, and unwanted content.
- Blocking policies, local moderation, licenses, and removal requests.
- How to handle message privacy and communication metadata.
- Interoperable sandbox and security of JavaScript applications.
- How to treat incompatible versions and preserve content long-term.
- How to distinguish a historical copy from a current verified version.
- How to keep complexity, consumption, and infrastructure cost low.

A decentralized network does not guarantee anonymity, availability, confidentiality, or absence of censorship. These are distinct properties, requiring their own technical and operational decisions.

The PNG approach adds overhead and limits compatibility with services that recompress images. The experience must justify this choice through portability, visual presentation, and collection culture.

19. Possible Names and Identity

```
Cartnet — provisional preferred name for the network.
Cartweb — alternative more directly tied to the web.
Pocketweb — emphasis on portability.
Pixelnet — emphasis on images.
Cartspace — emphasis on small independent digital spaces.
Cartridge Protocol / CRTP — alternative name for the specification.
```

The names have not been verified against trademarks, existing projects, or domain availability. This research must precede the choice.

Possible phrases:

> The web, in cartridges.
> Download a website. Keep a world.
> A little web you can carry.
> Collect the web. Connect the cartridges.

20. Final Vision

Imagine browsing a library of images that are, at the same time, small independent websites. Each cartridge has its visual identity, its pages, and its connections. You can receive it from a friend, save it to a disk, publish it on another server, or discover an entire community through it.

The content does not depend on being online all the time. The signature allows distinguishing authentic publications from altered versions. Links and events let the small digital universes connect. The infrastructure can be replaced, but the files remain.

The most important aspect is not hiding JavaScript inside a PNG. It is creating an open format and a network where people can own, share, preserve, and connect small web spaces without necessarily handing their content to a central platform.

Cartnet would be a cultural and technical experiment: a web of collectible, verifiable, interconnected digital objects.

> The web, in cartridges.

Author's Address

Pablo Murad
Independent
pablo@pablomurad.com
https://pablomurad.com
Official project website: https://rsw.works

Small files. Independent worlds.