

TECHNICAL DATA

Tools and Methods of the Canal-Keeper

June 2026 Edition

David Bories

Tried by the Flow — A Processual Ontology of the Datum, continuity volume
Technical Data — Tools and Methods for a Canal-Keeper

Conceived and directed by David Bories — Albi, France.

Written using generative artificial intelligence.

Translation produced by generative artificial intelligence, under the supervision of the author.

October 2026.

Aligned with the second edition of the trilogy.

First version: July 2026.

Version 0.13.0-en, pocket format.

© David Bories, 2026. Released under the Creative Commons Attribution 4.0 International licence (CC BY 4.0). Any use, including commercial use, is permitted provided the author and the source are credited.
<https://creativecommons.org/licenses/by/4.0/deed.en>

The full text of this volume and of the whole corpus is freely available at <https://artisanat-de-la-donnee.fr>

Table of Contents

Foreword — A Manual of Form	1
What This Book Is, and What It Is Not	1
For Whom	2
How This Manual Is Organized	2
The Rule of This Manual in One Sentence	3
Chapter 1 — The Canal, the Canal-Keeper, the Work	5
The River, the Canal, the Datum	5
The Intention and the Criteria	5
The Observational Optimum	6
The Three Phases, the Three Gestures	6
The Canal-Keeper’s Work	7
The Stance of Referent	7
Chapter 2 — The Hexagonal Method: Digging for Durability	9
The Principle: Isolating the Core	9
Why This Is the Tool of Durability	10
Infrastructure Obeys the Ports	10
The Canal-Keeper’s Gesture: Modelling Before Coding	12
The Concrete Organization of a Hexagonal Project	13
What the Hexagon Does Not Isolate on Its Own	13
Chapter 3 — The Core Without Dependencies: HTML, CSS, JavaScript	15
Why Native Rather Than a Framework	15
The Three Materials, and What Each Carries	16
Web Application, Installable Application: The Same Core	16
The Core’s Local Storage	17
Several Workbenches, a Single Work	17
Connecting the Outside World — Always Through the Hexagon	17
A Note of Honesty	18
Chapter 4 — Gamification: Making the Work Pleasant and Engaging	19

The Principle: Turning the Consolidated Datum into a Motivating Signal	19
The Mechanics, and the Craft Gesture They Serve	19
Thresholds: The Bridge Between Gamification and Steering . . .	20
The Safeguard: Gamifying Without Falsifying	21
The Fluid Experience, Beyond the Mechanics	21
Chapter 5 — Making the Datum: Drawing and Annotation	23
Mapping the Material: Which Data to Make	23
Setting the Use-By Limit	24
The Armature and Drawing: The Bed and the Water	24
The Adjacency List: Holding the Tree	25
The Two Edges: The Borne Link and the Reference	25
Nesting or Listing: The Criterion Is Search	26
The Armature Is Tended	26
Drawing Tools	27
Annotation Tools	28
Setting Drawing at the Optimum	29
Documenting One's Instruments	29
Chapter 6 — Circulating: Fixation and Transport	31
Fixation Tools: Where the Datum Holds	31
The Event System: Making the Act of Inscription Circulate . . .	32
Local Transport: Steering One's Activity	33
Remote Transport: Sharing the Datum	34
Multiplication: A Shared Fact Is No Longer a Unique Object . . .	35
Reusing Without Transmitting: The Objectal Moment, and How the Substance Holds It	35
The Incoming Flow: Drawing the Mail	36
Chapter 7 — Reading the Datum: The Steering Dashboard	39
The Reader Is Situated, Their Window Is Finite	39
Selectivity: Less, but Right	39
Indicators: Choosing the Right Ones	40
The Freshness of Reading: Irrigating the Dashboard	40
Reading Tools, and Their Biases	41
The Reader's Three Postures, and What They Dictate to the Interface	42
Tending the Capacity to Read	42
The Window Named, in the Signature	42
Chapter 8 — Local AI: The Canal-Keeper's Semantic Operator	43
What Generative AI Can Do for the Solopreneur	43
Why Local: Sovereignty, Cost, Control	44
Combining AI and Deterministic Rules	44
The Method of Progressive Automation	44
The Counterpart, Named	45
Chapter 9 — Tending: Maintenance, Security, Rituals	47
Maintenance Tools	47

Security and Digital Hygiene	48
The Reserve: The Double Withdrawn from the Cycle of Use . . .	48
The Rituals of the Workshop	49
The Signature of the Work	50
Living Documentation: Maintenance by the Client	50
Chapter 10 — The Digital Workshop: Hardware and Software	51
Hardware	51
Software	52
The Workshop as a Place, Physical or Virtual	53
The Workshop / Line Articulation	54
Chapter 11 — Equipment Itineraries: Assembling the Tools	55
The Procedure, Always the Same: Three Gestures	55
The Typical Itinerary, Stage by Stage	55
Three Illustrated Assemblies	57
The Thread That Holds Everything Together	57
Chapter 12 — The AI Agent Workshop: Industrializing the Itinerary	59
The Technical Base: Not Redoing What Is Proven	59
The AI Agents	60
Two Objects Not to Be Confused: The AI Agent Workshop and the Work	60
The Coding Agent: The Agents' Workbench	60
The Repository: Backup, Memory, and Distributed Base	61
The Sovereign Ports: Extending Without Depending	61
Steering the Workshop: Deterministic First, the Human at the Milestones	61
Putting It into Practice	62
What, in This Chapter, Does Not Go Stale	62
Annex — The Canal-Keeper's Toolbox	63
A. The Six Families of Tools, at a Glance	63
B. The Cross-Cutting Methods	63
C. Checklist of the Encountered Flow of Observation	64
D. Lapse Grid	64
E. Canal Review Template (Agenda)	64
F. Work Signature Sheet	65
G. Hardware and Software Reminder	65
H. The Safeguards, in Seven Sentences	65
I. Table of Correspondence — The Vocabulary of the Corpus and Existing Frameworks (Dated Form, July 2026)	66
Translation choices	66

Foreword — A Manual of Form

Note on the translation — This translation re-fixes the source text without displacing its substance. A few terms of the corpus have no obvious equivalent in English; the choices made, and how they are to be read, are set out in the section “Translation choices” at the end of the volume. This translation is not frozen: it will evolve with readers’ feedback and with the improvement of the means of translation. The French version is authoritative.

“The concept is a lamp; it does not make the chart of the reefs.” The whole trilogy *Tried by the Flow* held to this rule: saying what is true of the datum for a long time, and leaving to others the task of determining what should be done with it today. The present book makes the opposite wager, and owns it. It is a book of **form**.

Where *The River and the Canal* set out that canals must be dug in the flow of the real, where *The Data Artisan* named the canal-keeper and their six families of tools without prescribing their concrete use, this manual goes down one level: it says **with what** and **how** one digs, today, in 2026, a digital canal. Which languages, which machines, which architectures, which gestures. It is the canal-keeper’s workbench, open and stocked.

What This Book Is, and What It Is Not

This book **is** a user manual. It describes hardware and software tools available now, proven methods of making, and the way to keep them in the service of a single goal: offering the client — the solopreneur, the sole trader — a management of their activity that is fluid, legible, under control, and the serenity that comes with it.

This book **is not** a doctrine. Everything it names is dated. The languages cited will live and die; today’s local AI models will be replaced; the versions will change. This does not matter, on one condition: that the reader knows how to distinguish at all times **the substance** (what does not go stale — the stance of referent, the accuracy of the drawing, responsibility for the work) from **the form** (the tools, the languages, the platforms). The substance is inherited from the trilogy; this manual does not replay it, it

uses it. The form is what it adds, knowing that it is revisable.

When this manual writes “JavaScript”, “Docker” or “Ollama”, these words must be read as one would read, in a text from 1950, “a motor car can reach 100 km/h”: true at the date of writing, incidental to the reasoning, bound to be overtaken. What commits the manual is the *principle* that the tool serves — the separation of the business core from the infrastructure, local control, the durability of the medium — not the brand of the tool.

For Whom

For the **canal-keeper** — the data artisan (a craft yet to be created) — who wants to equip their digital workshop. For the apprentice who is learning the gestures. For the curious solopreneur who wants to understand what their tool does, and why. The manual takes the trilogy as acquired: it comes back to it only in brief reminders, where this is necessary to keep a gesture clear.

How This Manual Is Organized

The book follows the structure given by the *Workshop Book*: the canal-keeper’s **six families of tools**, arranged according to the **three phases of the datum**.

- **Making the datum** (phase 1): drawing and annotation. How the client’s activity produces an accurate datum, captured at the optimum.
- **Circulating the datum** (phase 2): fixation and transport. How the datum holds, travels locally to steer the activity, and far away to be shared.
- **Reading the datum** (phase 3): reading. How a dashboard makes the reader’s share practicable.
- **Tending**: maintenance. How a canal stays alive.

Before these families, two chapters set out the **methods** that run through all of them: **hexagonal architecture**, which makes the work durable by isolating the business core; and the **gamification of the experience**, which makes the work pleasant and engaging. After them, one chapter gathers the canal-keeper’s **workshop** — the hardware, the software, the rituals — and a last one proposes progressive **equipment itineraries**.

Each chapter stands on its own. One can come in through the tool one needs. But the order has a meaning: one reads well only what one has known how to make and to circulate.

The Rule of This Manual in One Sentence

Tools change; the craft — drawing accurately, making legible, making autonomous — remains. This manual arms the hand for today; it is for the hand to know that it will have to learn again tomorrow.

Chapter 1 — The Canal, the Canal-Keeper, the Work

This chapter teaches nothing new to anyone who has read the trilogy. It gathers, in a condensed language, the reference points that each tool of this manual uses. One can come back to it at a glance while reading.

The River, the Canal, the Datum

There are not, in the real, data waiting to be picked up. There is a **flow of the real** — what happens, permanently, indifferent to humans, which no one has dug and no one stops. Drawing is building a **flow of observation**: a canal that humans divert from the river through their intention, according to criteria they have chosen. The canal is infinitely poorer than the river; it contains only what its course has admitted. But the canal is something humans control: they can follow it, go back up it, replay it, link it to others, entrust it to a neighbour, fill it in when they no longer have any use for it.

The **datum of observation** is the grain of water one draws from this canal. One does not pick a datum; one digs a canal, and the datum is what flows out of it. Other data are born otherwise: an invoice issued, a quote signed are instituted by an act, under a rule; an indicator is derived, calculated from other data.

For the solopreneur, the river is their activity as it unfolds: worksites, trips, quotes, calls, deliveries, hours, kilometres, stocks that go up and down. All of this flows, and is lost, if nothing captures it. The canal-keeper digs, in this activity, the canals that bring back from it what steering needs — and nothing more.

The Intention and the Criteria

A canal is born of an **intention**: following a subject, answering a question, steering a decision. From the intention follow the **criteria**, which answer five questions:

- **What?** Which raw facts of the activity does one want to draw?
- **At what cadence?** How often does one draw?
- **With what fidelity?** What precision, what instrument, what acceptable margin of error?
- **With what corpus?** What context data accompany each drawing to make it interpretable (units, references, date)?
- **Over what scope?** What portion of the activity does one observe, and which does one leave aside?

These criteria are never neutral: they are the signature of the intention. Faced with any canal, the first question is not “what does it say?” but “what intention brought it into being?”. A canal never lies about the real; it shows only what its criteria allow it to show.

The Observational Optimum

More is not better. Between two symmetrical shoals — **under-drawing**, which falls below the threshold and loses the signal, and **over-engineering**, which draws far beyond what is necessary and gains nothing but cost — there is a zone where the canal is dug at its best. Its central point is the **observational optimum**: the joint setting of cadence, corpus and probity that maximizes the favourable gap between the benefit (the quality of the result) and the cost (drawing, transport, storage, computation, time).

There is no universal optimum. The question that determines it is always the same: *can the activity observed change significantly between two drawings?* If so, the cadence is too slow. If not, it is needlessly fast. The optimum is set again at each assignment. This is the golden rule of the digital canal-keeper: never over-equip a client, nor blind them.

The Nyquist threshold, plainly. For a series of drawings to render a movement faithfully, one must draw appreciably faster than the thing changes. A stock that varies every day cannot be tracked by an annual inventory. A monthly turnover is not steered minute by minute. Matching the cadence to the speed of the activity: that is the first setting.

The Three Phases, the Three Gestures

The datum has three phases, each set out by one tome. The digital canal-keeper equips all three.

1. **Making** (Tome 1) — the **act of inscription**: arresting a value in a flow that does not stop, and entrusting it to a medium. Three components: *what* (the object), *when* (the instant, the cadence), *how* (the instrument, the criterion).
2. **Circulation** (Tome 2) — the **act of exchange**: setting the work in movement. Transport is not the depositing of a datum somewhere; it

is a relation between two movements (the datum towards the reader, the reader towards the datum), accomplished only when the reader's share is practicable.

3. **Reception** (Tome 3) — the **act of reading**: *perceiving* → *distinguishing* → *deciphering*. Without reading, the datum remains constituted, but nothing is actualized. A work delivered that no one can read is an unfinished work.

The Canal-Keeper's Work

The work is not an isolated datum. It is a **canal at the optimum** — dug, sized, tended, rendering faithfully, for the intended use, what it was designed to capture — and, at the end of the chain, a **consolidated datum**: the synthesis that answers the question of use (an indicator, a status, a trend, an alert). A work is recognized by four features:

- it is **matched to the use** (neither over-calibrated, nor below the threshold);
- it is **durable** within the limits of what its nature allows;
- it is **signed** — its author is named, its date known, its criteria stated;
- it is **deliverable** — its reader's share is practicable, its reader's window is named.

The Stance of Referent

This is the axis around which everything turns, and the compass of each technical choice in this manual: **helping, making autonomous, not dispossessing**. Three concrete commitments follow from it, by which all the tools of the book will be judged.

- **Living documentation**: each tool comes with a map that the client understands. They are the editor-in-chief of their data, not a tenant of them.
- **Co-design**: the client identifies for themselves what deserves automation and what must remain manual; they keep sovereignty over their craft gesture.
- **Tended capacity to read**: the tool makes the business owner more capable of reading their data, never dispenses them from it.

Verifiable criterion. At any moment, a client must be able to take back control or change provider without losing their data or their understanding. Any tool in this manual that violated this criterion is a bad tool, however efficient it may be.

The canal-keeper's recurring assignment is that of the blacksmith one calls back for a new piece, not that of the software one can no longer get out of. This principle, as will be seen, governs the whole architecture of what is made.

Chapter 2 — The Hexagonal Method: Digging for Durability

Of all the tools in the manual, this one is not a piece of software: it is a **method of layout**. It answers the hardest constraint of the digital canal-keeper — *tools change, the craft remains* — and it answers it through a discipline of organizing code. Hexagonal architecture (also called *ports & adapters*) is the direct technical translation of the stance of referent: it guarantees that a client can change technology without losing their business logic, and that the work does not go stale with the version of a library.

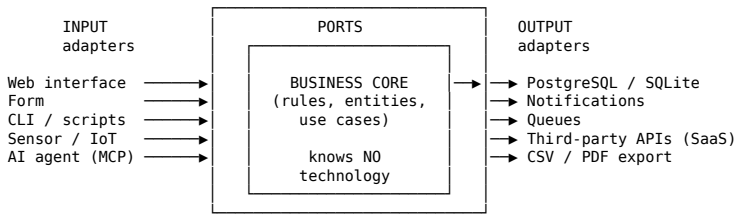
The Principle: Isolating the Core

A management application mixes two things of very different natures:

- **The business core** — the knowledge and the rules specific to the client's activity. *A load is optimal above 85% utilization. A quote not turned into an order within thirty days becomes a follow-up. VAT switches over when such and such a threshold is crossed.* This core depends on no technology. It is the craft gesture put into rules.
- **The infrastructure** — everything through which the core touches the outside world: the database, the user interface, third-party APIs, sensors, files, emails. This layer is **subject to external constraints of change**: it ages, is replaced, is reconfigured.

The ordinary error is to weave the two together — to write the business rule *inside* the code that talks to the database or that lays out the screen. The day the database changes, the rule is carried away with it.

Hexagonal architecture **forbids this mixing**. It places the business core at the centre, entirely ignorant of the outside world, and makes this centre communicate with the infrastructure through **contracts** called *ports*, of which each concrete technology is only an interchangeable *adapter* implementation.



On the left, the **input adapters**: everything that calls on the core (a click on the dashboard, a mobile form filled in on the worksite, a scheduled script, a sensor, an AI agent). On the right, the **output adapters**: everything the core activates to persist or to emit (saving to the database, sending a follow-up, publishing to a queue, calling an API). The core, at the centre, does not know whether the database is PostgreSQL or a file, nor whether the input comes from a human or a machine. It knows only its *ports*.

Why This Is the Tool of Durability

Three benefits, which are so many promises kept to the client.

One changes the form without touching the substance. The client starts with local storage in a file; six months later their volume grows, and one switches to a real database. On the side of the business core: *nothing changes*. One has only replaced one output adapter with another, behind the same port. The management rule that made the value of the work is intact. This is exactly the substance/form distinction of the trilogy, engraved in the code.

The craft is legible and transmissible. The business logic, isolated, can be written and reread in plain language before any code (see below). The client can validate it — *“yes, that is indeed how I calculate the performance of a vehicle”* — without knowing anything about programming. This is co-design made possible by the architecture.

One tests the core without switching anything on. Since the core depends on no database and no network, one can check it in isolation, with fake adapters. The canal is checked before being connected to the river.

Infrastructure Obeys the Ports

The usual order of reasoning is: *I have* a database, a mail server, an AI model — and I write code to use them. The infrastructure comes first; the code bends to it. The hexagon **reverses** this order. The port — the contract — comes first; the infrastructure obeys. One does not ask *“what can my infrastructure do?”* but *“what does the core need?”*, and the port

states it. Any concrete technology is then no more than an adapter that honours this contract, interchangeable — including over time.

The consequence goes further than a simple exchange of adapters. **One and the same port can take several adapters, and it is what is available that decides which one is mounted.** A port has its *degraded* adapter, when no real infrastructure is there, and its *amplified* adapter, when it is. A dated form, to fix ideas: a “sending” port — without a sending server, one adapter puts the request in a **queue** that the artisan takes up by hand; with a server, one adapter really sends. A “model consultation” port — without an engine, the adapter answers “*station required*”; with an engine, it answers. (*In the work of this edition, “the station” names this infrastructure that honours the ports; it is a form, dated and replaceable.*) The core, and the business rule that calls the port, **do not know which of the two adapters is connected.**

The decisive point is here: **the absence of infrastructure is not a breakdown, it is a case that the port provides for.** When the infrastructure is missing, the port does not break — it degrades *faithfully*. A request that cannot be honoured now is put in a queue (a deferred act, but recorded) or returns “*this is missing*” (a consultation that says what it is waiting for). The port keeps its contract even empty-handed: it does not lie, it does not fail silently. That the infrastructure obeys the port means that even its absence is a state provided for, named, that can be passed through.

Hence a **local and total** substitution. The day a real infrastructure arrives, one substitutes the adapter — port by port — without touching the core, nor the business rule, nor the interface. The function that calls “*send*” does not change depending on whether the sending waits in a queue or really goes out. This is the “*one changes the form without touching the substance*” already seen, extended: not only does one replace one infrastructure with another, but one sees it **appear and disappear** without the core noticing.

This reversal has a reach that goes beyond technique: it says *who is in command*. The artisan, keeper of the canal, sets through the ports what the core needs; the infrastructure — a database, a server, a model, a queue — is dated form: rented, replaceable, sometimes absent, never master. One owns the contract; one does not own the outside world, one connects to it — and if it is not there, one waits for it in a queue, without breaking anything.

The ports dictate the infrastructure — never the reverse. This is the maxim to engrave above the hexagon: the port is the substance (the lasting need of the craft), the adapter is the form (the technology of the day), and the durability of the work depends on the second never commanding the first.

The Canal-Keeper's Gesture: Modelling Before Coding

The procedure comes down to three gestures, in this order, always.

1. **Speaking the client's craft.** Distinguishing the **heart of the craft** — where computing dictates nothing, where the client's know-how resides — from the **support matrix**: the repetitive, administrative, automatable tasks, which do not call on it (follow-ups, tracking, accounting, data input). One never touches the heart of the craft; one models the matrix to free the heart of the craft. To the support matrix, common and generic, corresponds the **craft matrix**: the conditioned sequence of the work proper — the states of the workpiece and their conditions, never the gesture itself. Its principle is set out in the twin volume, the *Management System* (section III); *Technical Data* says with what it is equipped.
2. **Writing the business logic in plain language.** Before opening the code editor, one writes down the **entities** (the objects of the domain and their properties), the **rules** (the constraints), and the **use cases** (the sequences). In plain language, in pseudo-code, in a diagram. This document is validated with the client and will outlive all languages.
3. **Implementing in the chosen language.** The developer transcribes the validated logic. If the language changes one day, one transcribes again; the logic document, for its part, does not move.

An Example: The Logic for Optimizing a Load

Here is what the core layer looks like, written in plain language, independent of any technology. It could be implemented in any language without losing anything.

Entities of the domain

- *Item*: identifier, name, unit weight (kg), unit volume (m³), quantity, fragile (yes/no), priority (1 high → 3 low), stackable (yes/no).
- *Vehicle*: identifier, weight capacity (kg), volume capacity (m³), cost per kilometre, availability.

Fundamental business rules

- *BR1 — physical constraints.* The total weight and the total volume loaded never exceed the capacities of the vehicle; both apply simultaneously.
- *BR2 — fragility.* A fragile item increases by 10% the volume it occupies (safety).
- *BR3 — priorities.* High priority is handled first; when equal, high density (weight/volume) is preferred; as the last criterion, large quantity.
- *BR4 — efficiency threshold.* A load is optimal if its utilization reaches at least 85%, utilization being the minimum of the fill rate by weight and the fill rate by volume. Below that, the system raises an alert.

Use case — efficiency calculation (pseudo-code)

```

FOR a set of items and a vehicle:
  total_weight  = SUM(item.unit_weight × item.quantity)
  nominal_volume = SUM(item.unit_volume × item.quantity)
  adjusted_volume = nominal_volume
  FOR each fragile item:
    adjusted_volume += item.total_volume × 0.10
  weight_efficiency = total_weight / vehicle.weight_capacity
  volume_efficiency = adjusted_volume / vehicle.volume_capacity
  efficiency = MIN(weight_efficiency, volume_efficiency)
  limitation = IF weight_efficiency < volume_efficiency THEN "weight" ELSE "volume"
  RETURN { efficiency, limitation }

```

This text is the work, in the sense of substance. The JavaScript or other code that will implement it is only a dated embodiment of it. It is this inversion — the rule first, the technology afterwards — that makes hexagonal architecture, and that makes for durability.

The Concrete Organization of a Hexagonal Project

On disk, the separation can be read in the directory tree. The core (domain) never mentions a technology; the ports declare the contracts; the adapters implement them; infra hosts the services that can be outsourced.

```

client-project/
├── domain/           # CORE: no technological dependency
│   ├── entities/    # business objects (Item, Vehicle, Quote...)
│   ├── rules/       # management rules (BRL...BRN)
│   └── use-cases/    # sequences (optimizeLoad, followUpQuote...)
├── ports/           # CONTRACTS
│   ├── input/       # input ports (commands, queries)
│   └── output/      # output ports (repository, notification, publication)
├── adapters/        # interchangeable IMPLEMENTATIONS
│   ├── storage-sqlite/ # one adapter of the "repository" port
│   └── storage-postgres/ # another, which can be substituted without touching the
└── core
    ├── notify-email/
    └── export-pdf/
├── apps/            # client APPLICATIONS (see ch. 3 and 7)
│   ├── web/         # the dashboard (HTML/CSS/JS)
│   └── infra/        # base services (database, cache, secrets, logs)

```

Safeguard for sobriety. The hexagon is not an invitation to multiply layers. For a solopreneur, the core can fit in a few files and a single storage adapter. The method does not impose heaviness; it imposes *separation*. One stays at the optimum: enough structure for the client to be able to change tools one day, no more.

What the Hexagon Does Not Isolate on Its Own

The architecture protects the core, but it does not dispense with judgement. It is for the canal-keeper to decide *what* goes into the core (lasting business knowledge) and *what* remains on the periphery (dated choices).

Badly laid out, the boundary lets a business rule leak into an adapter, and the promise of durability is broken. The method is a lamp; the right layout remains an artisan's gesture, to be taken up again at each assignment.

Chapter 3 — The Core Without Dependencies: HTML, CSS, JavaScript

If hexagonal architecture decides *where* the code goes, this chapter decides *with what* one writes the core of the application that touches the client: its interface and its display logic. The answer of the canal-keeper concerned with durability is clear-cut and, at first sight, austere: **the native triplet of the web — HTML, CSS, JavaScript — without framework or dependency**. People often speak of *vanilla JS*. This choice is not conservatism; it is a strategy of duration and sovereignty.

Why Native Rather Than a Framework

Interface frameworks (React, Angular, Vue and their successors) render real services on large team applications. But for the work of a canal-keeper delivered to a solopreneur, they introduce three fragilities that the stance of referent cannot accept.

Accelerated lapse. A framework imposes its versions, its breaking changes, its chain of compilation tools. An application built on it requires permanent maintenance *just to stay in place*. The browser's native language, for its part, does not break: an HTML/CSS/JS page written cleanly today will still open in ten years. This is the difference between a canal that must be dug again each season and a masonry-lined canal.

Weight and dependency. A framework adds hundreds of kilobytes and a forest of third-party dependencies, each of which is a way in for a flaw or an abandonment. Native adds nothing: the browser is already there. The application remains light, fast, compatible with modest devices — which matters for an artisan on a low-end phone, in a van, off the network.

Reversibility. A work in native code is legible by any developer, without knowing a particular framework. The client can change provider without finding themselves a prisoner of an ecosystem. This is the verifiable criterion of chapter 1, applied to the interface layer.

Native is not the absence of method. Writing without a framework does not mean writing any old how. The native application core is organized into clear modules (one module per screen, one module per service), and it is precisely hexagonal architecture that gives it its backbone. Native provides the material; the hexagon provides the order.

The Three Materials, and What Each Carries

HTML — structure. HTML describes *what is*: the fields of an input form on the worksite, the zones of a dashboard, the list of quotes. Well written (semantic tags, explicit labels, logical order), it is legible by humans, by machines, and by assistive technologies. Structure is the skeleton of the work; it outlives any styling.

CSS — form and feel. CSS describes *how it appears*: colours, spacing, legibility, visual hierarchy, responsiveness to screen size (one and the same dashboard must be readable on a large workshop screen and on a phone). It is also, as the next chapter will show, the material of the animations that make the experience pleasant — the progress bar that fills up, the discreet *confetti* when an objective is reached. CSS carries a large part of the fluidity one feels, without a line of JavaScript.

JavaScript — behaviour. JS describes *what reacts*: calculating, updating an indicator when a datum comes in, switching a status, triggering an alert when a threshold is crossed. It is JS that drives the input adapter (data input) and the reading adapter (display), and that calls the business core. Kept clean and modular, it remains understandable and testable.

Web Application, Installable Application: The Same Core

The same HTML/CSS/JS core can take several forms according to need, without rewriting the business logic:

- **SPA** (*single-page application*) — a web application that behaves like software, without page reloads. Suited to the steering dashboard consulted at the office.
- **PWA** (*progressive web app*) — the same application, made **installable** on the phone or the computer, able to **work offline** and to store locally. This is often the ideal form for the mobile solopreneur: they “install” the tool on their home screen, enter a stock movement in the van without network, and synchronization happens when the connection returns.
- **Desktop** via a wrapper (Electron-type) or **hybrid mobile** via a wrapper (Cordova/Capacitor-type) — when the client wants a real installed application. These are only additional **input adapters** around the same core.

What matters is the principle: *a single business core, several access shells*. One chooses the shell at the optimum of the need, never for the sake of fashion.

The Core's Local Storage

A native application has, in the browser, local reserves that allow it to work without a server:

- simple reserves for small preferences;
- a structured local database (of the *IndexedDB* type) to keep volumes of business data on the device, offline, and synchronize them afterwards.

For a great number of solopreneurs, **local is enough**: their data fit on their device, encrypted, backed up, and never leave the business. The remote server (chapter 6) is mobilized only when sharing or pooling requires it. This is the sobriety of the *local-first* architecture: processing as close as possible to the source, making travel only what must travel.

Several Workbenches, a Single Work

On the convergence of separate inscriptions. An artisan can keep their work on several workbenches. What the workbenches inscribe separately does not contradict itself: it follows on — each inscription remains a frozen act, a dated finding, carried by the workbench that produced it. Their reunion is therefore never a copy of state, where one would erase the other: it is a replay of acts, in an agreed order, where each act remains. The current reading follows the order of the replay; the superseded act is not destroyed — it stays in the journal, with its date and its workbench. And when two workbenches have inscribed the same thing during their separation, the divergence is reported to the artisan: the work ascertains, it does not settle in silence.

Connecting the Outside World — Always Through the Hexagon

The native core never speaks *directly* to an external service. Everything that comes from outside — a sensor, a public API, a SaaS service, an AI agent — comes in through an **adapter** connected to a **port**. This is the rule that makes the whole durable, and chapter 2 set out its principle. A few examples of common connections, each isolated behind its port:

- **Public reference APIs** (for example the French State's APIs to verify a company number or normalize an address): an input adapter enriches the datum entered.

- **Payment collection** (a payment provider): an output adapter, replaceable if the client changes provider.
- **Logistics, mapping, messaging**: so many adapters, never dependencies woven into the core.
- **Sensors and connected objects**: an input adapter translates the sensor's signal into a datum of the domain (see chapter 5).

The day one of these services closes, raises its prices or goes stale, one replaces the adapter concerned. The core — the client's craft gesture — does not move. This is the promise of sovereignty kept down to the smallest connection.

A Note of Honesty

All-native requires rigour: without a framework to impose a structure, it is for the artisan to hold it. This is precisely why this manual insists so much on hexagonal architecture, on modularity, and on living documentation. Native is not the easy choice; it is the choice of duration and control. For a signed work, which a client must be able to keep and understand for years, it is the right arbitration. Like any arbitration, it can be reopened: a project of quite another size could justify other choices — but it would no longer quite be the workshop, it would be the line.

Chapter 4 — Gamification: Making the Work Pleasant and Engaging

A work is of use only when received: if the solopreneur does not read their dashboard, does not enter their movements, abandons the tool after three weeks, the best-dug canal remains without use, and its value is not realized. Reception is not an extra; it is the condition of usefulness. **Gamification** — making a game of it — is the method that works on this reception: it makes use pleasant, motivating, almost effortless, so that the gesture of management, usually tedious, becomes fluid and even satisfying.

The finding that justifies it is documented: a significant share of solopreneurs abandon their management for **lack of visibility on their performance** and through **disinterest in tasks perceived as tedious**. Gamification attacks exactly these two causes: it makes performance *visible* and effort *rewarded*.

The Principle: Turning the Consolidated Datum into a Motivating Signal

Tome 1 names the consolidated datum: the synthesis with high value of use extracted from a canal. Gamification is the art of **presenting this consolidated datum in a form that speaks to the situated reader** — that informs them at a glance, situates them in relation to an objective, and encourages them. It makes no new datum; it dresses the true datum so that it is read.

The reference model is that of the applications that have made learning or habit-tracking playful: progress bars, levels, streaks, small rewards. Transposed to the management of an activity, it gives simple and honest mechanics.

The Mechanics, and the Craft Gesture They Serve

Mechanic	Concrete example	What it serves
Progress bar	"You have reached 80% of your monthly turnover objective."	Instant visualization of the state (consolidated datum legible at a glance)
Levels and badges	"Badge <i>Invoicing up to date</i> unlocked after 10 invoices issued on time."	Motivation to complete the repetitive tasks of the support matrix
Targeted notification	"Order no. 123 is late for delivery — to be dealt with."	Fewer things forgotten; alert at the right moment
Threshold animation	A discreet visual effect (<i>confetti</i>) when an objective is reached.	Strengthens the sense of accomplishment
Streak / regularity	"7 days of stock input without interruption."	Sets up the habit of drawing (the client tending the canal themselves)

All these mechanics are achieved in **native HTML/CSS/JS**, without any dependency: a progress bar is a `<div>` whose width follows an indicator; a *confetti* is a CSS animation; a notification is a highlight on screen or a system notification. Gamification does not make the work heavier; it makes it lively with the materials already in place.

Thresholds: The Bridge Between Gamification and Steering

The mechanic most useful to the solopreneur is not the badge; it is the **threshold**. A threshold is a quantified limit which, once crossed, triggers a signal. The glossary of the craft distinguishes three kinds, and all lend themselves naturally to gamification.

- **Low critical threshold** — below it, the activity is no longer sustainable ("*fewer than 5 orders this month*"). Alert signal.
- **High critical threshold** — beyond it, a restructuring is needed ("*more than 20 orders: consider delegating or hiring*"). Anticipation signal.
- **Intermediate threshold** — an objective the business sets itself for a direct benefit ("*15 orders: one more week of holiday*"). This is the reward threshold, the heart of motivating gamification.

The exemplary case is the **crossing of a tax threshold** (VAT, turnover ceiling). For the solopreneur, these thresholds are a source of anxiety because they are undergone: one discovers them after having exceeded them. A progress bar towards the ceiling, an alert at 70% of the threshold, turn anxiety into steering: *the threshold is no longer undergone, it is anticipated*. This is, literally, giving the business owner back a hold on their trajectory — the very object of the work.

The Safeguard: Gamifying Without Falsifying

Gamification is powerful, and therefore dangerous. Tome 1 set out that observation inserted into a decision loop **transforms what it observes**: a score that changes the behaviour it measures. Badly used, gamification can push towards bad decisions — closing a mediocre sale to “unlock a badge”, inflating a figure to make a bar move forward. Three rules hold this risk in check.

1. **Rewarding the right gesture, never the raw figure.** One gamifies “*invoicing on time*”, “*chasing an unpaid invoice*”, “*keeping one’s stock up to date*” — virtuous gestures — and not “*making sales at any price*”. The mechanic must serve the health of the activity, not flatter it.
2. **Never disguising a projected datum as a measured datum.** A bar that anticipates a trend must flag itself as a projection. A measured datum and an estimated datum are not displayed in the same way. The probity of what is shown takes precedence over the effect.
3. **Tending the capacity to read, not lulling it to sleep.** Gamification simplifies reading; it must not dispense from understanding. Behind each gamified indicator, the client must be able to access the raw datum and the way it is calculated. The animation is a way in to the datum, never a curtain in front of it.

The right spirit. Gamification is not an entertainment stuck onto management. It is a way of making **the reader’s share practicable** — exactly the definition that Tome 2 gives of accomplished transport. A solopreneur who had neither the time nor the taste to read their figures reads them, because they have been made legible and engaging. This is data literacy made accessible, not varnish.

The Fluid Experience, Beyond the Mechanics

Gamification in the narrow sense (badges, bars) belongs to a broader requirement: **user experience**. For the solopreneur, fluid means concrete:

- **Little input, well placed.** Replacing a forgotten notebook with a mobile interface where each movement updates an indicator; the artisan no longer “fills in a table”, they have an up-to-date inventory. Input merges into the craft gesture instead of being added to it.
- **Autocompletion and pre-filling.** Verifying a client, normalizing an address, taking up information already known: the less the solopreneur types, the more they stay in their craft.
- **Immediate legibility.** A dashboard that answers at a glance the three questions that count (*where am I? what must I do? what is drifting?*), without drowning them under the rest.
- **Serenity through the right alert.** Being warned of what is drifting, and *only* of what is drifting. A notification that shouts all the time no longer says anything; the sobriety of the alert makes its value.

The fluid, pleasant, controlled experience is not an incidental comfort: it is the criterion of success of the work. A canal one has no wish to read is a canal that will not be actualized. Gamification, held with probity, is the tool that keeps the client *in a position to observe their own activity* — and gives them back their serenity.

Chapter 5 — Making the Datum: Drawing and Annotation

First phase of the datum, first gesture: **inscription**. It is here that the client's activity — the river that flows — becomes a datum. Two families of tools serve this phase: **drawing tools**, which materialize the act of inscription, and **annotation tools**, which turn a signal into a qualified datum. This chapter sets them out in their digital form of 2026.

Mapping the Material: Which Data to Make

Before any tool, a sorting. Not all the data of an activity are of equal worth, and the glossary of the craft arranges them in three kinds — a distinction that governs security, storage and sharing.

- **Business datum** — directly linked to carrying out the activity: time spent on a worksite, materials consumed, orders in progress, kilometres travelled. This is the heart of drawing.
- **Ancillary datum** — collected indirectly, without direct impact on the activity but potentially of value later: GPS positions during trips, hours of use of a machine (for future predictive maintenance). To be drawn with discernment: useful, but never at the price of over-engineering.
- **Sensitive datum** — subject to regulation (GDPR) or critical: client contact details, financial data. It requires particular care in storage and access (chapter 9).

On this first sorting is superimposed a second axis, which no longer classifies the datum by its nature but by its **dependence**. Some data stand on their own; others exist only through another — a follow-up through its quote, an attached document through the record it documents. The first is called the **bearer**, the second **borne** (see, in the complementary notes, *The Bearer Datum and the Borne Datum*). Spotting this relation at the mapping stage avoids an error of model: one does not give a borne datum an autonomous existence that it does not have.

The simplest realization is **nesting**: the borne datum is lodged in the

record of its bearer (a field that contains its follow-ups, its attachments). It inherits its fate mechanically — it is read, backed up, and **goes down in a cluster** with its bearer when the latter passes into lapse (see “Setting the Use-By Limit”, further on in this chapter). One could hold the same relation through a linked table with cascade deletion: the means is substitutable, the fact of structure is not.

Drawing is modelled on the client’s **business process**, broken down into stable terms: *stage* (broad phase: quote, making, delivery), *task* (atomic action: cutting, sending an invoice), *operational sequence* (ordered group of tasks producing an identifiable result: “preparing a worksite”). One draws where steering needs to see, not everywhere.

Setting the Use-By Limit

Tome 3 sets out the principle: state lapse is not an instant but a threshold — the **use-by limit**, the point where the gap from the real exceeds what use tolerates. The substance says that such a threshold exists; the form says here how it is set. Three cases cover the workshop. **The intrinsic threshold**: the object of the datum fixes the term itself (a refused quote sees its use closed at once — zero duration, immediate passage into lapse; a dated offer expires on its date). It is inscribed as a field of the datum, set at the act of inscription. **The threshold of use**: nothing in the object fixes the term, but the use knows it (a stock entry serves the act for only a day; a supply price, for a season). It is set by type of datum, in the armature — it is a setting of the canal, reviewed at the canal review. **The age default**: when nothing is specified, a prudent default applies, according to the age of the datum and the cadence of its canal. At the term, the work carries out the change of state in the order that the substance imposes: the summable part is consolidated, the new state is written, and only then does the datum leave current use — lapse puts in order; it does not destroy. And the cluster follows its bearer: when the latter passes into lapse, its borne data go down with it.

The Armature and Drawing: The Bed and the Water

To the sorting by nature (business, ancillary, sensitive) and the axis of dependence (bearer, borne) is added a third reference point, which governs modelling. Not every datum is at the same level of the construction: some **structure** — the names, the types, the formats in which the others come to be arranged (“company”, “contact detail”, “type of contact detail”); others are what **is arranged** — the value of a SIRET number, a dialled number, an amount. The first form the **armature**; the second are the operational **drawings** (see, in the complementary notes, *Armature: the Canal Dug in Stable Sources*). The armature is the bed; the drawing is the water. Both are data — every datum is born of a drawing (observational route), of an act that institutes or of an operation that derives — but the armature is

drawn from **stable sources**, the agreed, whereas the water comes from the volatile real of the activity.

In form, this difference is not marked by a writing convention, but by a **field**. Each node of the model carries a role: armature or drawing. The canal-keeper thus reads, at a glance, where the bed is and where the water is — and the tool can treat differently what is dug once (the armature) and what flows unceasingly (the drawings).

The Adjacency List: Holding the Tree

The armature is not flat: a contact detail carries telephones, a telephone a designation and a number. It is a **tree**, and the simplest form for holding it is the **adjacency list**: each node knows its parent, and the tree is reconstituted by following the links. A node, schematically:

- **id** — a unique timestamped identifier (UUID v7, chapter 6): the identity of the node;
- **role** — armature or drawing (the bed or the water);
- **name** — the name carried by the node;
- **parent** — the id of the node it belongs to, or nothing if it is at the root;
- **value** — what is captured, for a drawing (empty for an armature node);
- **references** — a list of cross-references (see below);
- and the **metadata** that this chapter has already named: date, author, unit, context.

This flat list is **navigable** (one goes down, one goes back up, one looks for a node by its id) and **extensible**: digging one more canal is adding nodes, without touching the others.

The Two Edges: The Borne Link and the Reference

Two links connect the nodes, and they are not realized in the same way — this is the counterpart in form of the note *The Bearer Datum and the Borne Datum* (complementary notes), which distinguishes the **borne** link from **reference**.

- **The parent link realizes the borne link** — composition, shared fate. A node belongs to its parent; it is born, lives and **goes down in a cluster** with it. Its deletion follows that of the parent: **cascade**. This is the edge of the tree. A contact detail without its company, a designation without its telephone no longer have any object: they are removed with it.
- **The references field realizes reference** — the cross-reference to a datum with a **life of its own**. In it one stores the id (UUID) of the target, accompanied by a **label** that states the nature of the link

{ label: "person responsible for the post", target: <UUID of an HR record> }). **No cascade:** deleting the target only breaks the cross-reference; deleting the node that refers does not touch the target, which subsists in its own tree. Reference links without tying existence together.

Keeping these two edges distinct is not a refinement: it is what governs, at deletion, **what dies in a cluster and what survives alone**. Confusing them — putting a reference link in cascade, or the reverse — corrupts the semantics of the two deaths.

Nesting or Listing: The Criterion Is Search

This chapter has already given, for the borne datum, a realization simpler still than the list: **nesting** — lodging the borne datum in the record of its bearer (a field that contains its follow-ups, its attachments). The means remains substitutable, but the choice is not indifferent, and one criterion settles it: **search**.

As long as a borne datum is **reached only through its bearer** — the lines of a quote, an issuer frozen on this quote —, nesting is suitable: one never searches for it alone, and the cluster is read as a block. But as soon as a datum must be **searched for or gone through for itself** — finding all the contact details of one type across the whole database, listing all the appointments of a week —, nesting makes it costly to reach: one would have to open each bearer to search its content. The **adjacency list** then holds it better: each node is a separate item, indexable, directly queryable, without going through its neighbours.

The rule is therefore: **nesting for the borne datum that is read only in a cluster; adjacency list as soon as the datum must be searched for alone**. Choosing one for the other does not change the fact of structure — the borne datum remains borne — but it can burden the functioning.

The Armature Is Tended

The role: armature nodes — the types, the formats — are not invented: they are drawn from the agreed (the socio-technical conventions that fix how one designates an address, a number, a legal form). They are data from a **tended stable source** (see the note *The Envelope: Quality as the Tending of the Agreed*, in the complementary notes): durable, but not eternal. In form, this has a consequence: the armature is **versioned** and **revised**, deliberately, when the convention shifts — the day an address becomes a triplet of coordinates, when the number gives way to an identifier. Keeping the armature attuned to the agreed is a gesture of tending, of the same nature as the canal review (chapter 9), but applied to the bed rather than to the water.

Safeguard on form. Adjacency list, UUID v7, nesting, references field: these are dated means of 2026. The substance — the armature as canal, the borne link and reference as two links with different fates, the agreed as a tended stable source — is set out in the three complementary notes; these pages only realize it. Another medium, another database, would realize the same facts otherwise.

At the optimum, always. The model **supports** depth; it does not **prescribe** it. For a small activity, one digs a tree **shallow but complete in canals**: one exercises the **two edges** — a borne datum in a cluster, a reference to another function — on a representative armature, so that the potential can be seen, without drowning the workshop under drawings it has no use for. Digging everything would be lining a ditch with masonry; it is under-drawing and over-engineering that the observational optimum keeps at a distance (chapter 1).

Drawing Tools

These are the instruments that arrest a value in the flow. In their digital version, they fall into four groups.

1. Collection Forms

The most common tool, and the most underestimated. A good input form — a mobile screen filled in on the worksite, at delivery, at the end of the day — *is* a drawing instrument, with its cadence and its fidelity. The principles:

- **Merging input into the craft gesture.** The artisan must not “fill in a table” on top of their work; they record a movement at the moment they make it. Fewer fields, better placed, at the right moment.
- **Capturing the corpus with the datum.** Each drawing carries its date, its author, its unit, its context — without which it will be uninterpretable later.
- **Pre-filling and constraining.** Choice lists rather than free text, default values, consistency checks at input: one prevents the error upstream rather than correcting it downstream.

2. Extraction Scripts

When the datum already exists in another system (accounting software, a spreadsheet, a mailbox, an existing database), one does not re-enter it: one **extracts** it with a script. Typical tools of 2026: scripts in JavaScript/Node.js or in shell, reading files (CSV, JSON), queries on an existing database. The extraction script is an input adapter: it translates the datum from a foreign format into the client’s domain.

Real case. A site director has a large management software package but does not get from it the indicators they need. The canal-keeper installs nothing new: they write **queries on the existing database** that extract exactly the view wanted, which the software then merely displays. These indicators are derived data, calculated from those of the software: making a datum, sometimes, is knowing how to query what is already there.

3. Sensors and Connected Objects (IoT)

For the physical magnitudes of the activity — temperature of premises, presence, level of a stock via tags (RFID), position of a vehicle — **sensors** draw continuously, at high cadence. They typically belong to the *ancillary datum* with strong potential. The canal-keeper knows their **calibration** (the measurement drifts over time) and their **biases** (what the sensor does not see). The raw signal of the sensor comes in through an input adapter that translates it into a datum of the domain; it never seeps into the business core.

4. Generative AI as an Instrument of Semantic Drawing

Here is the newest and most powerful instrument. A large language model (LLM) acts as a **semantic operator**: it knows how to read *unstructured* information — a client’s email, a photo of an order form, a transcribed voice message, a document — and **extract, qualify, structure** the datum from it. Where formerly a human was needed to read and copy, or dictionaries and deterministic rules heavy to maintain, the LLM directly extracts the elements wanted.

Typical uses for the solopreneur: extracting the information of an invoice received (amount, date, supplier); automatically sorting incoming emails; qualifying a client review; turning a photographed handwritten note into entered data. The model’s output is **structured** (in JSON, in a table) to enter the system. It is a derived datum: its provenance names the model and the original document.

This instrument is handled according to strict rules, detailed in chapter 8: a **local** model by preference (sovereignty), **human validation** on everything critical, and **combination** with deterministic rules where reliability comes first. The LLM does not replace judgement; it prepares the material that judgement will validate.

Annotation Tools

Drawing is not enough: one must **qualify**. An annotation grid turns a signal into a datum by imposing categories on it — which the flow of the real, of itself, does not contain. It is a decisive act and never a neutral one.

- **Grids and classifications.** The canal-keeper builds (or borrows) the categories: types of clients, order statuses (preparation, delivery, paid), kinds of expense, priority levels. They know what their grid **includes and excludes**. They borrow an existing grid when it is proven and comparable; they forge a new one when the client's use requires distinctions that exist nowhere else.
- **Annotation protocols.** Stable rules say *how* to qualify, so that one and the same fact always receives the same label — a condition for the series to be comparable over time.
- **AI-assisted annotation.** The LLM can propose a classification (sentiment analysis of a review, category of an expense); a human validates. One combines the semantic speed of the machine and the responsibility of the gesture.

The workshop **keeps its grids** in its library, with their history and the works they have made possible. A documented grid is a transmissible know-how; a lost grid is a canal one will no longer know how to reread.

Setting Drawing at the Optimum

The whole art lies in the setting, taken up again at each assignment. For each canal one digs, the canal-keeper goes through the grid of criteria of chapter 1:

- **What:** which business fact, which useful ancillary datum, which sensitive datum to protect.
- **Cadence:** matched to the speed of change of the activity. A stock with daily movement is drawn at each movement; client satisfaction is measured at a much slower rhythm.
- **Fidelity:** precision of the instrument, calibration, margin of error acceptable for the use.
- **Corpus:** the context carried along (date, unit, author, conditions).
- **Scope:** what one observes and what one deliberately leaves out of the field.

And the question that settles it: *can the activity change significantly between two drawings?* If so, one densifies; if not, one lightens. Neither blind nor drowned. It is this setting that distinguishes the canal-keeper's work from indiscriminate collection — and that spares the client the cost and the noise of a useless datum.

Documenting One's Instruments

A serious workshop keeps a **documented inventory** of its drawing tools: for each sensor, form, script or extraction prompt, its characteristics, its calibration, its known biases, its limits. This inventory is the memory of making. It makes it possible, later, to judge what a datum is worth — for judging a datum is first knowing with which instrument it was drawn.

Chapter 6 — Circulating: Fixation and Transport

A datum inscribed and never transmitted stays at the sole place of its inscription; making it circulate extends its availability and its possible uses. The second phase of the datum is that of circulation, and it rests on two families of tools: **fixation tools**, which keep the datum on a medium, and **transport tools**, which set it in movement. For the solopreneur, circulation has two quite distinct destinations: **local**, to steer their own activity, and **remote**, to share. This chapter treats both.

Fixation Tools: Where the Datum Holds

The medium is never innocent: it has its lifespan, its constraints of rereading, its technical dependencies. Badly chosen, it exposes the work to the **dead datum** — the case where the medium survives but where no one can any longer interpret what it carries, for want of a machine or a legible format. The canal-keeper chooses their medium according to the expected duration of use, the constraints of sharing, and the risk of lapse of the medium itself.

Formats: Preferring the Open and the Durable

The first rule of fixation is **not to shut oneself into a proprietary format** whose durability one does not control. One favours open formats, legible for a long time and by everyone:

- **JSON** and **CSV** for structured data for exchange and archival use: universal, readable by a machine as by the naked eye, re-importable everywhere.
- **Markdown** for documentation (and, via *pandoc*, the production of PDF): plain text that will outlive all word processors.
- **SQLite** as a database in a simple file: a complete database, without a server, transportable, open, legible in ten years. It is often the ideal medium for a solopreneur — robust and sober.

- **PostgreSQL** when volume, concurrent access or sharing require it: a proven and open server database.

The canal-keeper **documents the chosen format** so that the following generations — or simply the client, later — can still read what was written. A documented format is a reading key preserved.

Immutability: Keeping the History

For the data that commit (invoicing, traceability, legal tracking), a good practice of fixation consists in **never overwriting**, but **adding**. Rather than modifying a quote in place, one records a new version and archives the old one with its timestamp. Inspired by immutable registers (*ledgers*), this discipline keeps a **record** of everything that has happened, which is precious for tax compliance and for replaying the history. Technically, one identifies each record by a unique timestamped identifier (of the *UUID* v7 type) and one can seal each version with a fingerprint (a *hash* of its content), so that any alteration can be detected.

At the optimum, always. Immutability has a cost (volume, complexity). One reserves it for what deserves it — committing, traceable, durable data — and one keeps simple storage for the rest. No need to line a ditch with masonry.

The Event System: Making the Act of Inscription Circulate

In a living work, each gesture — creating a quote, following up a client, closing a due date — is an **act of inscription**: a fact of the activity, dated and fixed — most often instituted by the act itself (a quote issued, a due date closed) rather than measured. For a work to be more than the sum of its functions, these inscriptions must **circulate** between the modules without any one depending on another. That is the role of an event system.

The **event** is the software form of the act of inscription. It is given a name — `function.entity.action` —, a date, a source, and the data of the fact. It is, in the sense the trilogy meant, a **datum**: “*what is given — for a later occasion*”. A module emits the event without knowing who is listening to it; others subscribe to it in order to set off, from it, a new action — manual, automated or delegated. One and the same inscription can open several occasions.

Two organs carry it. The **bus** transports the inscription, here and now, from emitters to subscribers: it is local transport, **decoupled**, which makes it possible to graft on a function without touching the others. The **journal** keeps the succession of inscriptions: it is the flow of observation made durable, immutable, appended at the end (*append-only*) — the **reliable source** of the activity. The current state of a record is only a

convenient reading; it is the journal that is authoritative. In it one replays, one consolidates, one audits.

Each act opens and closes with an inscription. Before acting, one inscribes the intention (*init*) with its parameters: if the act fails hard, one goes back to its origin. The act accomplished, one inscribes its closure (*finish*); the act that fails in a controlled way, its failure (*fail*) — which a module can listen for in order to alert at once. The loop of the act is itself recorded.

Safeguard on form. All this is dated: a bus in memory, a local journal, for a single-station work. The substance remains the ontology — the act of inscription, the datum, the flow of observation. Tomorrow, the bus may be distributed and the journal archived elsewhere; the inscription, for its part, remains what it is: a fact given for a later occasion.

Archiving and Obligations

Some data must be kept by legal obligation (invoices for several years). **Archiving** is the long-term keeping of inactive but useful or obligatory data. It is distinct from current storage: it can be compressed, encrypted, put away separately, even have its **automatic deletion** scheduled at the legal term (for example an erasure after the planned retention period, for GDPR compliance). The canal-keeper inscribes these rules from the design stage.

Local Transport: Steering One's Activity

The most everyday circulation for the solopreneur goes nowhere far away: it connects their own drawings to their own dashboard. This is **internal** transport, which turns a scattered activity into legible steering.

Architecturally, it is the movement *from the business core towards reading*: the datum made (chapter 5) is consolidated — average, total, status, trend — then conveyed to the screen that will make it visible (chapter 7). In local-first, all this happens **on the client's device**, without a remote server: the raw datum does not leave the business, steering is instantaneous, and it works offline.

The typical mechanism: each movement entered updates a state indicator; the dashboard reads the consolidated indicator, never the raw mass. The solopreneur does not “consult a journal of all their operations” — they see *where they stand*. This is consolidation (Tome 1) in the service of the reader's window (Tome 3).

When several devices or several applications of the same client must coordinate on one and the same machine or one and the same local network, tools of internal circulation come into play: a shared database, a fast cache (of the Redis type) for frequently read states, a message queue (of the RabbitMQ type) to pass events from one service to another without coupling them, scheduled tasks (*cron*) for periodic consolidations. For a

solopreneur, most of the time, **none of this is necessary**: a database file and a little JavaScript are enough. One mobilizes these tools only when a threshold of complexity is really crossed — at the optimum, again.

Remote Transport: Sharing the Datum

The second use of circulation points outwards: transmitting to a third party (the chartered accountant, a client, a partner), or bringing one's datum into a broader exchange. Tome 2 gives the **minimal typology** of transport, and each mode calls for its tools.

- **Moving** — carrying the datum from one place to another (a backup, a migration). Tools: file export, encrypted copy.
- **Transmitting** — passing it to a designated recipient (sending the annual summary to the accountant). Tools: **CSV/PDF** export, message, secure drop.
- **Making accessible** — putting it at the disposal of whoever will come to query it. Tools: an **API** (programmable interface) with authentication.

Transport Is Accomplished Only If the Reader's Share Is Practicable

This is the golden rule, inherited from Tome 2: transmitting is not depositing. Delivering to the accountant a raw export that they will have to reprocess is not transmitting — it is depositing. The canal-keeper **consolidates** and **puts into the format expected by the recipient** so that the recipient can accomplish their share without effort. For the solopreneur who shares, this means: a clean export, dated, in the format the third party knows how to open, accompanied by what is needed to interpret it.

The API and the MCP Protocol: Opening One's Datum to AI Agents

A new need of 2026: making one's datum queryable by **AI agents**. More and more clients want an AI to be able to read their diary, their orders, their activity, in order to assist them. The technical answer is to develop a **client API** — access points that expose, in a controlled and authenticated way, the data wanted — and, so that it can dialogue with AI models according to a standard, to make it compliant with the **MCP** protocol (*Model Context Protocol*), which standardizes the interconnection between a source of data and an agent.

On the architecture side, this API is only one more **adapter**, at the input and the output of the hexagon: one adds it without touching the business core. It is the perfect illustration of durability through architecture: a twenty-year-old piece of software, whose data were prisoners of its inter-

face, finds a second life by exposing an MCP API — without rewriting its logic.

Alterations of Transport, to Be Prevented

Transport is not neutral: it can lose units, metadata, context; it takes time, during which the datum ages. The canal-keeper knows these alterations and prevents them: they carry the corpus along in the export, they times-tamp, they flag freshness, they check that a datum fresh at the source does not arrive stale. A workshop documents, for each type of work, the **proven mode of transport** and its conditions of reception.

Multiplication: A Shared Fact Is No Longer a Unique Object

A peculiarity of the datum: making it travel can **multiply its carriers** (a copy elsewhere without ceasing to exist here). This changes the way of reasoning about reliability, ownership, security. One does not secure a fact spread in multiple places as one guards a unique copy. Before sharing, the canal-keeper — and the client — ask themselves: *once out, this datum will not come back; who will be able to do what with it?* Remote circulation commits; it is decided consciously, not by default.

Reusing Without Transmitting: The Objectal Moment, and How the Substance Holds It

The typology of transport leaves one case aside: the one where the canal-keeper **reuses**, inside the work, a datum they already hold — without sending it to anyone. The quote that takes up a client's record is the example: the consolidated datum passes from one medium to another — from the record to the quote —, but it remains **with the same recipient**, the artisan. This is a spatial navigation (Tome 2), but an *internal* one and *without a third-party recipient*: no additional carrier is created towards the outside, the datum is not ceded, not even shared. One reuses; one does not transmit.

This gesture deserves to be named, for it touches the ontology. Reusing a consolidated datum as a stable reference point that one designates, copies, attaches, *is treating it, for the time of a use, as an object*. The **four pre-suppositions** that Tome 1 attributes to object ontology are verified locally here: the consolidated datum seems to **pre-exist** the quote, one holds it to be **stable**, one recognizes in it a **use value**, one **possesses** it and handles it. This is the **objectal moment** of the work.

Should one be alarmed by it? No — and that is the whole point. Tome 1 already said it: the *datum* is there “what can be **taken up** in the following occasion”, a **residue of process**. Taking up a consolidated datum does not

betray the processual: it is its very mechanism. The consolidated datum is, *by definition* (Tome 1), “reusable in other contexts”. The objectal moment is therefore not a relapse into the dominant ontology: it is a **dated form** that the processual **substance** provides for, authorizes — and **bounds**.

Three bounds, which the work makes visible where ordinary object ontology forgets them:

- **Lapse.** The consolidated datum reused is a residue detached from its flow of observation. The client’s name, frozen on the quote, no longer follows the living record. When the record disappears, the work **displays** it — a *broken link* — instead of pretending that the reference point is still valid. Lapse is shown, never hidden; the frozen name, for its part, remains exact for this quote.
- **The inappropriability of the flow.** One owns the **medium** and the **occurrence** — the quote, its consolidation —, never the flow from which they proceed (fourth reversal, Tome 1); this non-rivalry is ontological and does not prejudge the legal regime of the record or of the quote. Deleting the contact does not break the quote: one has only ever owned the residue.
- **Recontextualization.** The *datum* is **relational** (Tome 1): the identifier taken up means “this client” *only in this quote*, as a new relation — not as a thing transported intact.

The rule of practice fits in one sentence: *reuse the residue without ever forgetting that it is residue*. What separates the work from the dominant approach — *data lakes*, “my data” held to be a durable good — is not that it forbids itself to handle the consolidated datum as an object; it is that it never loses sight of the fact that the consolidated datum can go stale for the use it makes of it, that it does not own the flow, that it recontextualizes instead of transporting ready-made meaning. The objectal moment is legitimate *as long as the substance holds it*.

Safeguard on form. All this is dated: an identifier, a frozen name, a broken-link marker. The substance remains: *one reuses a residue, one does not own a flow; handling the datum as an object is permitted for the time of an occasion, on condition of checking that it is still valid for the use one makes of it*. It is here, at the point of reuse, that object ontology operates in the work — and it is here, exactly, that the processual substance must hold it.

The Incoming Flow: Drawing the Mail

Until now, what the work inscribed came from the flow of the workshop’s activity: a quote laid down, a follow-up, an invoice. But another flow arrives, from outside: the mail. A message received is not an object put away in a box; it is, it too, a flow of the real — words addressed to someone, dated, which will not be repeated identically. The work makes no exception of it: it treats it like any flow, through a **drawing**.

This drawing has a requirement of its own, which the dated form makes visible. The mail does not live in the work: it lives on a source, outside, which the work does not own. Drawing, here, is **reading without altering the source**. The station collects the messages read-only: it marks nothing, moves nothing, consumes nothing on the server; it takes a copy of them and leaves the flow intact. This is the **inappropriability of the flow** carried over to the input: one does not seize the mail, one collects an occurrence of it. The source remains someone else's; the work keeps only a deposit of it, dated, on its own medium — which says nothing, on its own, about what it has the right to do with it.

Once drawn, the message is **inscribed**. It enters the two times of the datum like any observation: a first time, the message as it was received — the sender, the subject, the body, intact; then a second time, where it is taken up in the following occasion — read, sorted, answered. The work keeps the history of this itinerary: is this message new, read, dealt with? The box outside does not know; the work, for its part, knows it, because it has inscribed not the mail, but *its reading by the artisan*.

There remains reading itself. Presented to the **situated reader**, the message calls for the act of reading: perceiving that it is there, distinguishing it from the others, deciphering it. On this last gesture, deciphering, support is possible — and it is here that one must be precise about what the support is, and is not. A consultation can propose a **qualification** (of what kind is this message?) and a **draft** reply. These proposals are not the message: they are **verisimilar** data, not observed data. The message received is observed; what is inferred from it is verisimilar, and must remain so — displayed as such, *to be validated*. The artisan rereads, adjusts, decides. The draft has effect only through their act; nothing goes out without their having made it their own. The support deciphers a first time; it does not read in the reader's place, and it is not the support that answers for what goes out.

All this belongs to the substance, and the passage through electronic mail is only its form. That the incoming flow is collected by one protocol today, by another tomorrow, changes nothing in what is said: a flow of the real arrives from outside; one draws it without altering it; one inscribes it with its history; a situated reader deciphers it, assisted but not replaced. The route is dated. The gesture is the one it has always been.

Chapter 7 — Reading the Datum: The Steering Dashboard

Third phase, third gesture: **reading**. It is there that the datum becomes useful, or remains unread. Tome 3 established it: *without reading, nothing is actualized*. The king tool of this phase, for the solopreneur, is the **steering dashboard**. This chapter says how to design it so that it is really read — that is, so that it respects the **reader's window**.

The Reader Is Situated, Their Window Is Finite

No reading is from nowhere. The solopreneur reads their activity from a precise situation: little time, an attention already called upon, a given corpus of competences, often a small screen between two tasks. Their **reader's window** is narrow. The best-dug canal, transported with the greatest fidelity, is of no use if what it delivers exceeds this window. *A datum one does not have the capacity to examine is not read, whatever one believes.*

The design of a dashboard is therefore, above all, a work on the window: showing only what can be read, in the time and with the attention the client has available. It is the exact counterpart, on the reception side, of the observational optimum on the making side: neither blind nor drowned.

Selectivity: Less, but Right

Reading is **selective** by nature: one does not read everything, one chooses. A good dashboard chooses *in the reader's place* what deserves their scarce attention. It answers, at a glance, three questions and those alone:

1. **Where do I stand?** The current state, consolidated into simple signals: turnover for the month against objective, cash position, orders in progress, margin.
2. **What must I do?** The actions that call for a decision: a quote to follow up, a late invoice, a stock below the threshold.

3. **What is drifting?** The alerts: a threshold approached, a trend that turns, an anomaly.

All the rest — the mass of raw drawings — remains accessible *in depth*, but is not displayed on the surface. The dashboard delivers the **consolidated datum**, not the journal. The canal-keeper's Lavoisier rule: *consolidate the activity into simple signals, rather than deliver an undecipherable raw mass*.

Indicators: Choosing the Right Ones

An **indicator** is a value calculated from the data, intended for steering and decision. The canal-keeper distinguishes its types and chooses them with the client, never in the client's place:

- **State indicators** — measure the instant: number of late orders, stock level, cash position.
- **Trend indicators** — measure movement: change in turnover, rate of conversion quote → order.
- **Threshold indicators** — situate in relation to a limit: progress towards the turnover ceiling, towards an objective (see gamification, chapter 4).

For activities where financial steering is too volatile or too late, one can rely on **physical indicators** rather than monetary ones — hours of work, volume produced, kilometres, utilization rate of a vehicle, energy consumed — which describe the activity in a stable and immediate way, the financial translation being made afterwards. This approach through the physical flow gives robust steering, not very sensitive to fluctuations, and often more telling for an artisan than accounting entries.

Real case. A transport company is losing money on badly filled deliveries. The right indicator is not global: it is the **unit performance of each vehicle according to its type of load** (utilization rate, cost/benefit). Well chosen, the indicator turns a vague intuition into a reasoned decision: which means to maintain, which to reduce. Choosing the indicator is already half the work.

The Freshness of Reading: Irrigating the Dashboard

A reading is taken at an instant; the flow, for its part, does not stop. A dashboard consulted then left as it is quickly loses its **freshness**: it shows a bygone state of the flow of observation while the activity, for its part, has gone on. The situated reader should neither have to wonder whether what they see is up to date, nor reload the page by hand to make sure. *A reading that lags behind the flow no longer quite reads the present: it reads a bygone state.*

The principle, on the reception side, is simple: the reading surface must be **irrigated** by the flow. What is inscribed appears, where the reader is looking. The dashboard is not a snapshot one refreshes from memory; it is a surface kept to the freshness of the flow of observation. It is the exact counterpart, in the third phase, of the observational optimum of the first: neither frozen nor flickering.

Concretely — and in a dated way — each act is inscribed as an event; the surface listens to the flow and repaints itself when the flow grows. Three safeguards keep this irrigation within the **reader's window**. First, *refresh only what is before one's eyes*: the window is finite, there is no point recalculating a view one is not looking at. Next, *never overwrite a gesture in progress*: the reader is also an actor; if their hand is in a field, the irrigation waits — it does not take back what they are in the middle of inscribing. Finally, *group the bursts*: one refresh per cycle, not one per event, failing which the flickering would exceed the window as surely as a raw mass.

Two additions complete the arrangement: a “**refresh**” within reach, for the reader who wants to force the update, and a refresh on **change of view**, since time may have passed. In this way the dashboard stays at the optimum: neither behind the flow — the silent lapse of reading —, nor exceeding the window.

Safeguard on form. All this is dated: a bus in memory, a local repaint. The substance remains: *reading must stay fresh with respect to the flow*. The means of irrigation may change; the requirement of freshness, for its part, does not go stale.

Reading Tools, and Their Biases

The tools that make things visible — graphs, visualizations, queries, cross-tabulations, descriptive statistics — **transform what they read**. A visualization chooses its axes, its scales, its aggregations; it can reveal a real trend or suggest a false one. The canal-keeper knows these biases as they know those of their sensors, and they inform the client of them. Concretely, for the solopreneur, one keeps to honest and legible representations:

- **key figures** brought forward (the strict minimum);
- simple **curves** for trends, on a scale that does not mislead;
- **progress bars** towards objectives and thresholds;
- **action lists** sorted by urgency;
- a sober **colour code** (for example: green/orange/red) for state and alert.

All this is achieved in **native HTML/CSS/JS** (chapter 3): no need for a heavy library to display a bar, a figure, a simple curve. The reading tool stays light, fast, openable everywhere, and reversible.

The Reader's Three Postures, and What They Dictate to the Interface

Tome 3 describes three ways of inhabiting over-throughput. Each translates into choices of interface.

- **Framing** — restricting or moving one's window to really read what matters. *In the interface:* letting the client choose their indicators, hiding the rest, going to the essential. The dashboard is an equipped framing.
- **Delegating** — entrusting reading to a device (an automatism, an AI agent) whose **situation one inherits**. *In the interface:* allowing delegation (an automatic summary, a smart alert) while signalling *who* read and with what limits. One delegates reading, not responsibility.
- **Accepting** — consenting with lucidity not to read everything. *In the interface:* sobriety. Not making the client feel guilty for not seeing everything; some data with brief freshness were not meant to be read, and this is not a failure.

The worst posture, the unwritten one, is the fourth: *believing one has read what one has merely seen pass*. A dashboard that gives the illusion of control without really giving it is harmful. The probity of the tool consists in making legible *what is*, and in giving access, beneath the surface, to the datum and to the way it is calculated.

Tending the Capacity to Read

A safeguard repeated because it is central: the reading tool makes the business owner **more** capable of reading their data, never dispenses them from it. Each indicator displayed can be “opened”: where does it come from, over what period, calculated how, with what margin. Delegating to the interface the fatigue of calculation is a good arbitration; delegating understanding is a bad one. The dashboard is a lamp the client learns to hold, not an oracle that thinks in their place.

The Window Named, in the Signature

Every work of reading is delivered by naming the **reader's window** it aims at: for whom this dashboard is intended, under what conditions it is legible, what one must know to interpret it. This field appears in the signature sheet (chapter 9). Designing for a situated reader, and saying so, is part of the delivery — it is what distinguishes a finished work from a datum thrown onto the screen.

Chapter 8 — Local AI: The Canal-Keeper’s Semantic Operator

Generative AI is, for the canal-keeper of 2026, the most powerful tool and the most delicate one. It runs through the three phases: it extracts and qualifies (making), it helps to transmit and summarize (circulation), it deciphers for the client what the client delegates to it (reception). Tome 1 situated it precisely: the large language model condenses into one entity the three roles historically separated — the **library** where information is shelved, the **librarian** who retrieves it, the **storyteller** who renders it. It is a **systemic leap**, with its benefits and its counterpart. This chapter says how to use it as an artisan: locally, with measure, and without dispossessing the client.

What Generative AI Can Do for the Solopreneur

As a **semantic operator**, an LLM processes unstructured information without one having to maintain dictionaries or complex rules. The management tasks it automates, wholly or in part:

- **Qualifying and classifying** — sorting emails, categorizing expenses, analysing a client review.
- **Extracting and structuring** — taking the useful fields from an invoice received or an order form, with usable **JSON** output.
- **Generating** — writing a draft quote, follow-up, standard reply, a report.
- **Synthesizing** — summarizing a file, a day of activity, a series of exchanges.

The guiding principle, everywhere: **the AI proposes, the human decides**. For repetitive, structured and low-risk tasks, automation can be complete. For everything that is critical, sensitive or legal — the validation of a quote, an invoice, a contract, a payslip — **decision and validation remain human**, without exception.

Why Local: Sovereignty, Cost, Control

The canal-keeper favours **local AI** — a model that runs on the client's or the workshop's machine, and not in a remote cloud. Three reasons, which are the three pillars of the craft:

- **Sovereignty and confidentiality.** The datum — often sensitive — **does not have to leave the business.** This is the *edge / local-first* architecture applied to AI: processing as close as possible to the source. The attack surface shrinks; compliance is simplified.
- **Control of costs.** No subscription per request, no bill that grows with use. The cost is that of the hardware, amortized.
- **Durability and independence.** One does not depend on the policy of a remote supplier who can change their prices, their conditions, or close down.

The tools of 2026 make this accessible: **open and light models** (of the Llama, Phi, Gemma and other families), able to run on a decent laptop or a small server; and **local runtimes** (of the *Ollama* type) that install and serve them in a few commands. One chooses the model to fit the need: a small model to classify emails, a somewhat more capable model for extraction or writing. No point aiming for the largest: at the optimum, again.

When the cloud remains legitimate. Local AI is the rule, not a dogma. A heavy and one-off task, or a need that the local hardware does not cover, can justify a measured recourse to a remote service — on condition of telling the client, of not passing sensitive data through it without their informed consent, and of isolating it behind a replaceable adapter. The question of arbitration arises, as always.

Combining AI and Deterministic Rules

AI is not always the right tool. For **structured, deterministic tasks, where reliability comes first** — a VAT calculation, the application of a management rule, a format validation — classic methods (explicit rules, automation of repetitive tasks of the RPA type, dictionaries) remain superior: predictable, verifiable, without hallucination. The canal-keeper **combines**: AI for the semantic and the fuzzy, the deterministic rule for the exact and the critical. Often, the output of an LLM is *checked* by a deterministic rule before being accepted — indicators, thresholds, consistency checks detect the anomaly.

The Method of Progressive Automation

Equipping a solopreneur with AI is not done all at once. The method is **progressive and collaborative**: start small, validate, extend. Six stages.

1. **Initial audit.** Through a guided interview, gather the daily, weekly, monthly tasks. Classify them according to four criteria — time consumed, arduousness, risk of error, impact on quality — and **prioritize** them in a matrix. Deliverable: a prioritized list of candidate tasks, validated by the entrepreneur. One targets first what is time-consuming, arduous and a source of errors.
2. **Mapping of the automatable tasks.** Cross the list with feasibility (data available, hardware, skills) and propose, for each priority task, the solution (suitable model, complementary tools, validation method). Deliverable: a table per task — solution, hardware required, benefits, risks.
3. **Prototyping and testing.** On **one or two** simple tasks with high impact (classifying emails, generating draft replies): install the tools, configure, **train the entrepreneur**, then test for two to four weeks while measuring time saved, errors, how it feels. Deliverable: a test report.
4. **Progressive deployment.** In waves — two new tasks every one to two months. Frequent targets: email management, client/supplier follow-up, stocks and prices, document creation, monitoring. Insist on **manual validation** and on understanding the limits. Deliverable: a follow-up dashboard.
5. **Optimization.** Refine the prompts, adjust the rules, automate more complex tasks (sales analysis, monthly reports), upgrade the hardware if needed. **Encourage autonomy:** the client knows how to modify simple rules, the documentation is clear.
6. **Maintenance.** Update the models, secure, back up, hold regular reviews. Deliverable: clear follow-up arrangements.

The verbatim that guides the relation with the client sums up the spirit: “AI is there to help you, not to replace you.” — “Start small, think big.” — “You decide what is automated and how.”

The Counterpart, Named

The canal-keeper tells the truth, including what a salesperson would keep quiet. Delegating the processing of a datum to an AI is **moving a little away from it**. Tome 1 set it out: the fusion of the three roles (library, librarian, storyteller) makes disappear what their separation made possible — traceability (knowing where a piece of information comes from), the return to the source, criticism, the counter-power of one role over another. Delegating one’s reading to a model is **inheriting its situation**: its frozen corpus, its blind spots, its freshness. *One exchanges submersion for a dependence, and everything depends on knowing what that on which one makes oneself dependent is worth.*

The safeguard is inscribed in the design: **the tool tends the business owner’s capacity to read, it never dispenses them from it**. An AI that summarizes must give access to what it has summarized. An AI that

classifies must allow its classification to be reviewed. The probity of delegation is knowing the situation one inherits — and never taking delegated reading for a reading without a situation, seen from above. This is the difference between a solopreneur who is equipped and a solopreneur who is dispossessed.

Chapter 9 — Tending: Maintenance, Security, Rituals

A flow of observation is too often designed, deployed, then abandoned to itself. This negligence degrades works: a sensor drifts, a format goes stale, a canal fills with data that no one reads any more. The sixth family of tools — **maintenance** — keeps the canal alive over time. It is coupled with a permanent requirement, **security**, and is structured by regular **rituals**. This chapter closes the loop of the six families.

Maintenance Tools

These are the devices that make it possible to keep a canal going over time. The canal-keeper **builds them in from the design stage**, never as an afterthought.

- **Calibration drift alerts.** A drawing instrument (sensor, script, extraction prompt) drifts. One monitors the upstream/downstream gap and flags when the measurement comes loose.
- **Freshness indicators.** Each datum carries its age. The system flags when a datum has exceeded its validity window for the use one makes of it, given the speed of its real.
- **Periodic re-drawing.** For data subject to lapse, one schedules renewal (scheduled tasks, *cron*): freshness is not recovered, it is made again.
- **Upstream/downstream consistency checks.** One checks that what comes out of the canal remains consistent with what goes into it, to detect a silent break.

For the solopreneur, these tools remain **sober**: a discreet alert, a freshness indicator on the dashboard, a night-time consolidation task. Maintenance is not a factory; it is the regular care of the carpenter who goes over their timber frame.

Security and Digital Hygiene

Security is not an option, especially for the **sensitive datum** (client contact details, financial data, data subject to the GDPR). **Digital hygiene** is the set of practices that keep data organized, secure and compliant. The pillars, in the *local-first* logic:

- **Encryption.** Sensitive data are encrypted at rest (on the disk, in the archive — for example with AES-256) and in transit (secure exchanges). An unencrypted backup is a leak waiting to happen.
- **3-2-1 backup.** Three copies of the data, on two different media, one of them off site. This is the rule that protects against **disappearance** — the second death of the datum (Tome 1), distinct from lapse: here the medium degrades, the format becomes illegible, or access is lost. An encrypted external disk and an encrypted remote copy are often enough.
- **Restricted access.** Each datum is accessible only to whoever needs it; authentication protects access (for example OAuth2 for connected services).
- **Traceability.** Keeping a record of who did what, when — which immutable fixation (chapter 6) makes easier.
- **Minimization and a set retention period.** Drawing and keeping only what serves; scheduling deletion at the legal term. Less data kept, less surface of risk — and this is sobriety too.

Sovereignty as security. The best safeguard remains structural: if the raw datum does not leave the business (local processing, controlled transfers), the attack surface is reduced to the source. Sober architecture is not only economical; it is safe.

The Reserve: The Double Withdrawn from the Cycle of Use

A datum lives in the work's cycle of use: there it serves, is consolidated, passes into lapse. As long as it is exposed there, it can be lost there — a wrong move, a failing medium. To guard against loss, one gives it a double withdrawn from this cycle: a **frozen** copy, written once and never rewritten, which one does not read in current use and which one mobilizes only to restore what has disappeared. This is the **reserve**.

The reserve is neither live nor lapsed: it is withdrawn from the cycle of use — not from the flow. There is no outside-the-flow (this is the substantive lesson of Tome 2): the content of the reserve, like any drawing, goes on moving away from the real, which has continued without it; what is suspended is its exposure to use and to the wrong move, not its lapse. It is distinguished from the 3-2-1 backup, which multiplies copies in distinct places to guard against material disaster. The reserve, for its part, aims at **continuity over time**: it keeps, at intervals, the whole state of the work,

and rotates its **media** — one keeps the last few generations (for example the current one and the two previous ones), at several temporalities (the day, the week, the month); beyond that, the oldest goes out. One and the same snapshot can serve several temporalities at once: one duplicates only once per period.

The canal-keeper's gesture here is to **set the cadence and the number of generations** according to what they can lose without harm, then to let the reserve hold on its own. Kept by the sovereign storage port, it obeys the same rules as the rest: without the station, it waits; once the station is back, it is deposited. The reserve never serves the present activity — that is its virtue: what does not serve does not risk being altered by use.

The Rituals of the Workshop

A workshop lives through its rituals — regular, articulated moments, in which arbitrations and care take place. Without them, the work drifts; with them, the canal stays alive. Two rituals structure tending.

The Canal Review

At a fixed interval (monthly, quarterly depending on the case), each tended canal is reviewed. Standard agenda:

1. Volume and quality of the drawings since the last review.
2. Calibration of the instruments — a drift ascertained?
3. Relevance of the criteria with regard to current use.
4. Reference corpus — still sufficient?
5. Possible loops — does the canal modify what it observes?
6. Signals of lapse — what to refresh?
7. Maintenance load — sustainable?
8. Decisions and arbitrations — who answers for them?
9. **Effective reception** — is the canal read by those for whom it is meant, or does its datum remain unread?

The Reception Review

The counterpart of the previous one, on the reader's side. Have the works delivered been **received**? Was their reader's share practicable? Did they arrive within their validity window, or did they go stale on the way? When a work is not read, one diagnoses the path:

- **incapacity** of the recipient — a remediable path: one makes the client more autonomous (training, simplification of the interface);
- **disinterest** — a path that calls for a decision of attention, or a re-design of the alert;
- **rapid lapse** — a path that calls for no answer: *not all unread data are failures*. A datum from a closed financial year sees its steering

window close: this is its normal lapse for this use, and it remains valid for the balance sheet and the archive.

This distinction spares the most costly error: treating as a failure a datum that was not meant to be read, or letting pass as normal a real non-reading.

The Signature of the Work

At each delivery — a canal built, a consolidated datum, a diagnosis — the canal-keeper **signs**. The signature is not vanity: it is an act of responsibility (the artisan answers for the work delivered) and of transparency (the user can judge). It is placed at the head of every delivery, on the model proven by the community of the craft.

WORK SIGNATURE

- Artisan: [name]
- Delivery date: [date]
- Type of work: [tended canal / consolidated datum / diagnosis]
- Question of use: [the question the work answers, in one sentence]
- Main criteria:
 - Scope: [...]
 - Cadence: [...]
 - Intended fidelity: [...]
 - Corpus used: [...]
- Observational optimum: [justification of the arbitration retained]
- Transport / fidelity expected at reception:
 - [what the recipient must be able to do for their share to be practicable; possible alterations on the way]
- Intended reader's window: [to whom it is legible, on what conditions]
- Known limits: [...]
- Estimated validity: [...]
- Conditions of maintenance / re-drawing: [...]

Two fields deserve attention, for they anchor the signature in circulation and reception. **Transport / fidelity expected** recalls that signing a work is making the reader's share practicable — not only depositing the datum. **Intended reader's window** recalls that a work is made for someone situated, and that saying so is part of the delivery. Without these two fields, one would sign a datum as if it did not have to be received.

Living Documentation: Maintenance by the Client

The best tending is the one the client can do themselves. The stance of referent requires it: **living documentation** — each tool comes with a map that the client understands, so that they are the editor-in-chief of their data, not a tenant of them. A work delivered without its documentation is a work that makes the client dependent; a documented work is a work they can maintain, develop, or entrust to someone else. This is the **verifiable criterion** of the craft, applied up to the last stage: at any moment, the client takes back control without losing anything.

Chapter 10 — The Digital Workshop: Hardware and Software

An artisan without a workshop is not an artisan: they are a theorist of craft. The workshop is the place where the gesture is made, where the tool is held, where the work is shown. This chapter describes the workbench of the digital canal-keeper: the **hardware** and the **software** of 2026. Everything in it is dated, and therefore revisable; what is not is the principle that guides each purchase — *matched to the need, neither blind nor over-engineered*.

Hardware

The canal-keeper's craft is of **low capital intensity**: it requires neither a rare machine nor a large investment. A few thousand euros are enough to equip a complete workshop. This is a strength of the craft — one sets up without going into debt.

The Workstation

- **A decent and mobile laptop**: it is the central tool, which serves to develop, to travel to the client's premises, to run the light AI models. A recent processor and generous memory count for more than extreme graphics power. Mobility allows **proximity** — going to the client, drawing on site, training in person.
- **One or two screens**: to keep the gesture right, reading the code and the work side by side.
- **An ergonomic workstation** (desk, chair): the gesture of reading a screen is long; the artisan's body is also a tool.

The Server

Depending on the needs of the client and of the workshop, a local **server** takes on several faces, sometimes combined on one and the same small

machine:

- **Web server** — to host the steering applications, locally at the client's premises or on a small workshop machine.
- **Database server** — when volume or sharing exceeds what a simple local file can hold.
- **Local LLM server** — a machine that serves the AI models internally, so that the datum does not leave the business. A modest server is enough for light models; a decent graphics card speeds up the more capable models.

The **edge / local-first architecture** comes in a continuum: from the client's simple terminal (where everything fits on their device) to a small workshop server pooled among several clients of one and the same territory. One sizes as close as possible to the real need.

Storage and Backup

- **A NAS server** or networked storage: to centralize and back up the works of the workshop.
- **One or more encrypted external disks:** for the **3-2-1** rule (three copies, two media, one off site). This is the guarantee against the disappearance of the datum.

Peripherals

A printer (often a professional one, for deliverables and documentation), a scanner if one digitizes existing documents. Modest, but useful for circulation and archiving.

Typical set-up budget. The experience of pilot businesses puts the complete initial equipment (laptop, screen, external disk, small server/NAS, peripherals, ergonomic workstation) in a range of a few thousand euros — a robust and low-risk financial profile, often lightened by business-creation grants.

Software

The canal-keeper's software is, as far as possible, **open, free of charge and durable** — consistent with the sovereignty they sell to their clients.

The Development Environment

- **A free operating system:** a Linux distribution (for example Debian) offers stability, no cost and control. It equips the workstation as well as the servers.
- **A code editor (IDE):** the tool in which the work is written. One chooses one that supports the multi-file work of hexagonal architecture, version control, and AI assistance if wanted.

- **A version control system (Git)** and repository hosting: to keep the history of the code, go back, and keep a record of the changes. Git is to the workshop what the immutable register is to the datum: the reliable memory of the work done.

Containerization

- **Docker and Docker Compose:** to package an application and its services (database, cache, queue) in reproducible **containers**. The benefit is decisive for the canal-keeper: what runs on their workstation runs identically at the client's premises, and the **environments** separate cleanly (Development, Test, *Staging*, Production). One delivers an assembly that installs and reinstalls without surprises.

The Language and the Services

- **JavaScript / Node.js** as main language: a single language from the business core to the extraction scripts and the services, which simplifies maintenance for a one-person workshop.
- **Open databases:** *SQLite* (a file, without a server, ideal for sobriety) or *PostgreSQL* (server, for volume and sharing).
- **Internal services** if needed, and only if needed: a fast cache (Redis), a message queue (RabbitMQ), an authentication service (OAuth2), log aggregation. These building blocks plug in as **adapters**; one adds them only when a threshold of complexity is really crossed.

Local AI

- **A runtime for local models** (of the *Ollama* type) and a few **light open models** (Llama, Phi, Gemma families...), chosen to fit the need of each task (classifying, extracting, writing, summarizing). See chapter 8 for their use.

Documentation and the Production of Deliverables

- **Markdown** to write the living documentation and the textual deliverables; **pandoc** to produce clean PDFs from them. Plain text, durable, versionable — which depends on no proprietary word processor.

The Workshop as a Place, Physical or Virtual

Beyond the machines, the workshop is a **place** endowed with functions that tools alone do not give:

- **a threshold** — an identified space one enters in order to *make* the datum (outside, one consumes it);

- **a workspace** where one can set down one's instruments and works in progress;
- **a library** — the memory of the workshop: reference books, annotation grids kept, archives of past flows, documentation of the works delivered;
- **a wall of the exhibited work** — to honour the work, teach by example, give points of reference.

When the workshop is **virtual** (online workspace, shared repository), it must reproduce these **functions**, not only store files: an identified access threshold, personal zones, a review space, organized archives, a gallery of works. Without that, it is no longer a workshop, it is a shared folder.

The Workshop / Line Articulation

In every organization, the workshop coexists with the **line** — the massive, low-cost processing of standardized flows. Both are necessary and do not serve the same needs. The line processes routine volumes; the workshop makes **by hand** the flows that deserve it: those that guide high-stakes decisions, those that must last, those whose traceability matters, those that must be made legible to a precise recipient. The passage from one to the other is decided explicitly. The canal-keeper knows when to bring out the workbench, and when to let the line do the work. This is again an arbitration at the optimum — the last, and the most strategic.

Chapter 11 — Equipment

Itineraries: Assembling the Tools

The previous chapters described the tools one by one. This one assembles them. For a tool is worth something only in a gesture, and a gesture only in an itinerary. Shown here is how the methods (hexagonal, gamification, native core, local AI) and the six families of tools follow on from one another in the service of a solopreneur — from the first contact to the tended work. Everything in it is illustration: another client will call for another assembly.

The Procedure, Always the Same: Three Gestures

Whatever the client, the canal-keeper follows three gestures, in order.

1. **Modelling before coding.** Speaking the client's craft, distinguishing their **heart of the craft** (untouchable) from their **support matrix** (automatable), and modelling the matrix to free the heart of the craft.
2. **Drawing to fit the need.** Locating the optimum between under-drawing and over-engineering, to be set again at each assignment.
3. **Delivering a legible and reversible work.** Documented, understood by the client, signed, and one over which they can take back control at any moment.

Development proceeds in stages, never all at once: **immersion** (installing the vital base), **adjustment** (the client gets used to it and identifies their real needs), **expansion** (one adds modules if the need justifies it).

The Typical Itinerary, Stage by Stage

Stage 0 — Observation

A free or light **audit** phase: talking with the business owner about how they work, their expectations, their problems, their opportunities. Mapping their business process (stages, tasks, sequences). Spotting the tasks that are time-consuming, arduous, sources of errors. Deliverable: an

overview of the current state and a prioritized action plan. *One proposes nothing before having listened.*

Stage 1 — Diagramming (Hexagonal Method)

Diagramming the data management system according to **hexagonal architecture**: isolating the business core, naming the ports, identifying the adapters (storage, interface, third-party services, sensors). The client obtains a clear view of what is going to be built, and can assess its timescales, budget, gains. It is here that durability is decided.

Stage 2 — The Business Logic (in Plain Language, Then in Code)

Writing the **entities, rules and use cases** in plain language (chapter 2), having them validated by the client, then implementing them — in **native JavaScript** (chapter 3). The core can be tested outside any technology.

Stage 3 — Drawing and Fixation

Setting up the **drawing tools** to fit the need (mobile forms, extraction scripts, sensors, semantic extraction by local AI) and the **annotation grids** (chapter 5). Choosing the open and durable **fixation medium** (chapter 6): often **local-first** storage on the client's device, with encryption.

Stage 4 — Steering (Reading + Gamification)

Building the **dashboard** (chapter 7): three questions — *where do I stand, what must I do, what is drifting* —, indicators chosen with the client, **thresholds** (tax, business, objectives). Dressing it in honest **gamification** (chapter 4) so that it is read and pleasant: progress bars, right alerts, anticipation of thresholds. The client moves from an activity undergone to a steered trajectory.

Stage 5 — Remote Circulation (If Needed)

If the client has to share (accountant, partner, AI agent), adding the **transport adapters**: consolidated CSV/PDF exports, or an **API (to the MCP standard)** for AI agents — without touching the core. The recipient reader's share must be practicable.

Stage 6 — Tending and Autonomy

Setting up **maintenance** (freshness alerts, 3-2-1 backup, re-drawings), establishing the **rituals** (canal review, reception review), **signing** the work, and — above all — delivering the **living documentation** that makes the

client autonomous. The recurring assignment is not a tended dependence: it is the blacksmith one calls back for a new piece.

Three Illustrated Assemblies

A. The Building Tradesperson Who Must Switch to Certified Invoicing

Need: a painter, a mason used to a spreadsheet must move to a compliant invoicing solution, without knowing how to choose. *Assembly:* little development, a lot of accompaniment. One audits, one presents the options, one installs and configures, one **trains** — ideally in a **pooled format** (a small group shares the need, exchanges, makes progress together). The work is not software: it is autonomy. Tools mobilized: audit, choice grid, training.

B. The Mobile Solopreneur Who Steers Their Stock and Their Cash

Need: replacing a forgotten notebook with up-to-date steering. *Assembly:* hexagonal core in native JS; **installable PWA** on the phone, working offline; encrypted **local-first** storage; input merged into the gesture (each movement updates an indicator); gamified dashboard with anticipation of the VAT threshold; 3-2-1 backup; PDF export for the accountant. The artisan no longer “fills in a table”: they have an inventory and a cash position legible at a glance.

C. The Business That Wants Tailor-Made Indicators in a Large Software Package

Need: a business owner has an expensive software package but does not get their indicators from it. *Assembly:* no replacement. The canal-keeper writes **queries** on the existing database (indicators derived from its data), which the software displays. Possibly, a business logic for decision support (for example the unit performance of each vehicle in a fleet), written in plain language and then implemented. Tools mobilized: extraction, hexagonal business logic, reading. Making a datum, sometimes, is knowing how to query what is already there.

The Thread That Holds Everything Together

Across these assemblies, one and the same thread: **giving the capacity, then the choice**. The canal-keeper does not sell a product that locks in; they transmit a capacity that sets free. Each tool of this manual — from the humblest form to the local AI model — is judged by the same measure:

does it make the client more master of their activity, or does it take them further from it? As long as the answer is the first, the work is right.

The artisan alone weighs nothing in the economy of the datum; the mesh does. Each equipped solopreneur, made able to read and share their datum on their territory, is a link in the mesh. Power comes from their federation — but it always begins with a canal well dug, at the optimum, signed, legible, and which a client can take back in hand. That is the object of this manual; the rest is a matter of practice, and of the hand that learns.

Federation requires no centre — it requires only one more link in the mesh, whose craft is to coordinate. The project manager of a worksite does not command the schedules of the trades: they hold the most complete reading, work out the dependencies, and propagate through exchange what each accepts in their own work. Coordinating through exchange costs fewer gestures than the bare mesh — and produces what the bare mesh does not produce: coherent order. It is a craft, not an authority; the day it becomes an authority, the mesh has become a pyramid again.

Chapter 12 — The AI Agent

Workshop: Industrializing the Itinerary

The previous chapter described the equipment itinerary as a canal-keeper conducts it by hand: observing, diagramming, modelling, drawing, steering, circulating, tending. This chapter describes how to **automate** this same itinerary, without changing anything in its gestures, by entrusting it to a team of specialized AI models that will be called the **AI Agents**. The canal-keeper knows how to manage them: it is their workbench, augmented.

Like everything in this manual, this chapter is **form**. The tools it names — a coding agent, a container engine, a repository host — are dated. The substance it serves — the itinerary, the three gestures, the stance of referent — does not go stale.

The Technical Base: Not Redoing What Is Proven

One part of the work never changes from one activity to another: the plumbing — serving files, working offline, updating itself, storing state. The canal-keeper does not rewrite it each time; they hold it to be a **proven base** and generate only the domain. This is the deterministic part of the itinerary: the less the workshop generates, the less it gets wrong.

On this base, one simple rule: **an external dependency is a debt and a risk**. When a built-in and proven primitive does the job, one prefers it — encryption through the browser's native interface rather than a third-party library loaded from elsewhere, a server of a few lines rather than a package to install. On what is sensitive, this preference is not negotiable: a dependency can be compromised, and the work must be able to run without installing anything, without network, and remain reversible. A single module system, consistent links, tests that pass before any handover: this is what distinguishes a work over which the client takes back control from an assembly that comes apart.

The applicable detail — which interfaces, which format, which checks — belongs to form and lives outside this manual, in the technical constraints that the agents read. The principle, for its part, does not go stale.

The AI Agents

An AI Agent is an AI model held by a **charter**: a role, an artefact it reads, an artefact it produces, constraints, and what must be true before it hands over. To each stage of the itinerary corresponds an agent. They work in a chain: not a document that grows longer, but a **series of distinct documents**, versioned and readable by a human, each the work order of the next.

The singularity of the procedure fits in one sentence: **the upstream documents are not mere technical specifications — they are the chapters of the quality manual of the activity.** The activity manual is not produced alongside the software; it is the upstream part of the chain. Hence a boundary, operative here:

- above the line, the description of the activity (actors, products, processes, traceability, flows, data): this is the manual, the **substance**, the durable asset;
- below it, the hexagonal folder, the code, the tests: **dated form**, which can be regenerated from the substance.

The asset is not the software. It is the specification of the activity.

Two Objects Not to Be Confused: The AI Agent Workshop and the Work

The AI agent workshop is the production workshop. It brings together, on the local machine, the model engine, the agents, a sandbox for execution and testing, a repository. One can isolate it in a container (Docker): it is the production tool, operated by the artisan.

The work is the software produced. It obeys chapters 2 and 3: a native hexagonal core, local-first, installable, reversible, which the client can take back. **It does not require the AI agent workshop in order to function.** One does not deliver a workshop to a client: one delivers to them a piece of furniture of which they have the plans. Confusing the two — asking the entrepreneur to operate every day the machinery that served to make it — would betray the gesture of reversibility.

The Coding Agent: The Agents' Workbench

The AI Agents need an environment that gives them, under supervision, access to the file system, to the shell and to the tools: this is the role of a **coding agent**. In 2026, one uses for this a tool such as Vibe Code: it reads

the files, runs commands, writes code, all under the operator’s control. Nothing is written without supervision — which makes the milestone of human validation not an option, but the very mechanics of the tool. This tool is form: it will be replaced. The principle it serves — agents that act on the workbench under the eye of the referent — remains.

The Repository: Backup, Memory, and Distributed Base

The work is versioned in a **git** repository, for two reasons already known from chapter 10: **backup** (the work does not depend on a single disk) and **memory** (the evolution can be reread, compared, undone). To this is added a third function, specific to the AI agent workshop: the repository serves as a **distributed base**. Everything needed for the launch — the corpus as knowledge base, the charters, the templates, the start-up procedure — fits in a repository that one clones or downloads, and that one copies onto the local machine. The artisan thus has a complete base, ready for use, which they develop for their territory.

The Sovereign Ports: Extending Without Depending

The local model is enough for most tasks, and it is sovereign by construction. Where it is not enough, the workshop — like the work — puts **high-performing sovereign services** to work, connected through the hexagon (chapters 3 and 6, “connecting the outside world — always through the hexagon”). Three uses arise:

- a **sovereign frontier model** for precise semantic tasks out of reach of the local one;
- **forecasting** with time-series foundation models (demand, stocks, load);
- **inter-professional sources of data**, for collective monitoring.

Absolute rule, inherited from distributed sovereignty: these ports are **non-vital**. The core works when the external link breaks; the sovereign service is an enrichment, never a dependence. An adapter, therefore, and always substitutable.

Steering the Workshop: Deterministic First, the Human at the Milestones

A local model is weaker than a frontier model at producing reliable code. Chapter 8 gave the counter: **combining AI and deterministic rules**. The workshop implements it at the scale of the entire software. It is **mostly deterministic** — templates, scaffolding, bounded generation of the gaps alone on a pre-built and tested base — and mobilizes the model only where it excels: describing the craft, mapping, writing. This is not

total autonomy; it is an assisted itinerary with checkpoints, where the referent validates before each handover.

Putting It into Practice

The concrete detail — folder structure, documents to supply, order in which to launch the agents, commands — belongs to a living **getting-started** guide, kept in the repository and revised with it. This manual does not take it up: instructions for use that name commands and an address would go stale within a few seasons, and a book of substance does not have to age at the rhythm of a tool. The manual says the principle; the repository says the gesture of the day.

What, in This Chapter, Does Not Go Stale

The coding agent, the container engine, the repository host, the sovereign services named: all that will pass. What remains is that an equipment itinerary can be entrusted to agents held by charters; that the quality manual of the activity is its durable part and the software its regenerable part; that one delivers not the workshop but the work; and that no external service must ever become vital. The canal-keeper who holds to this will hold any tool that tomorrow presents to them.

Annex — The Canal-Keeper’s Toolbox

This annex gathers, as a quick reference, the tools of questioning and the reference points of the manual. A tool is not a rule: the rule dispenses with thinking, the tool opens a question. None of these sheets delivers an automatic verdict — they serve to *forget nothing essential*, never to replace situated judgement.

A. The Six Families of Tools, at a Glance

Phase of the datum	Family	Digital tools of 2026
Making	Drawing	Mobile forms, extraction scripts, sensors/IoT, local AI (semantic extraction)
Making	Annotation	Grids and classifications, protocols, AI-assisted annotation
Circulation	Fixation	Open formats (JSON, CSV, SQLite), immutability (UUID v7, hash), encrypted archiving
Circulation	Transport	Local: consolidation, database, cache, queues. Remote: CSV/PDF export, API (MCP)
Reception	Reading	Native dashboard (HTML/CSS/JS), indicators, sober visualizations, gamification
Tending	Maintenance	Freshness/calibration alerts, re-drawing, 3-2-1 backup, consistency checks

B. The Cross-Cutting Methods

- **Hexagonal architecture** — isolating the business core; ports & adapters; changing the form without touching the substance; modelling before coding.

- **Core without dependencies** — native HTML/CSS/JS; SPA/PWA; local-first; durability, lightness, reversibility.
- **UX gamification** — making the consolidated datum legible and engaging; thresholds; bars, badges, right alerts; safeguard: rewarding the right gesture, never the raw figure.
- **Local AI** — semantic operator; sovereignty, cost, control; the AI proposes, the human decides; combining with deterministic rules.

C. Checklist of the Encountered Flow of Observation

To be asked in front of a data set, an indicator, a figure.

Making — 1. From what act was this datum born: a drawing from a flow of the real, an act that institutes under a rule, or a calculation from other data? 2. Who dug it, with what intention? 3. What criteria — cadence, scope, instrument? 4. What units, norms, implicit references? 5. What was not drawn and could be relevant? 6. What is the date of the inscription, and its functional age? 7. Is the flow in a loop — does it modify what it observes? 8. What biases does the class of instrument generate?

Circulation — 9. Can the upstream chain be followed back to the source? Through how many hands, copies, reprocessings — and what may it have undergone there?

Reception — 10. For my use, is this flow suited? 11. Do I have the means and the time to read it, given my window? If not, is it incapacity, disinterest or rapid lapse?

D. Lapse Grid

For each datum and the use one makes of it, situate its temporality (trigger of a question, not a verdict): lapse is judged for a use. A perishable datum is not a bad datum: it is a datum with a short validity window for this use, which can remain valid for another (balance sheet, history).

Datum	Fragment of the real	Speed of change	Functional age	Probable path of non-reading
Demographic datum	Population	Slow	Several years	—
Economic indicator	Market	Fast	Days to weeks	Rapid lapse
Environmental measurement	Air, water	Variable	Hours to months	Depending on use
Behavioural score	Person	Fast	Days	Rapid lapse

E. Canal Review Template (Agenda)

1. Volume and quality of the drawings.

2. Calibration of the instruments — drift?
3. Relevance of the criteria with regard to use.
4. Reference corpus — sufficient?
5. Loops — effects observed?
6. Signals of lapse — what to refresh?
7. Maintenance load — sustainable?
8. Decisions and arbitrations — who answers for them?
9. Effective reception — read, or unread (incapacity / disinterest / rapid lapse)?

F. Work Signature Sheet

To be placed at the head of every delivery (detail in chapter 9).

WORK SIGNATURE

- Artisan: [name]
- Delivery date: [date]
- Type of work: [tended canal / consolidated datum / diagnosis]
- Question of use: [the question the work answers, in one sentence]
- Main criteria:
 - Scope: [...]
 - Cadence: [...]
 - Intended fidelity: [...]
 - Corpus used: [...]
- Observational optimum: [justification of the arbitration retained]
- Transport / fidelity expected at reception:
 - [what the recipient must be able to do for their share to be practicable; possible alterations on the way]
- Intended reader's window: [to whom it is legible, on what conditions]
- Known limits: [...]
- Estimated validity: [...]
- Conditions of maintenance / re-drawing: [...]

G. Hardware and Software Reminder

Hardware — laptop, screen(s), ergonomic workstation; server (web / database / local LLM); NAS and encrypted external disks (3-2-1 backup); printer/scanner.

Software — free OS (Linux Debian); IDE + Git; Docker & Docker Compose (Dev/Test/Staging/Prod environments); Node.js/JavaScript; SQLite or PostgreSQL; if needed Redis, RabbitMQ, OAuth2; Ollama + light open models (Llama/Phi/Gemma); Markdown + pandoc.

H. The Safeguards, in Seven Sentences

1. The optimum is set again at each assignment: neither blind nor drowned.
2. The business core depends on no technology; the form is replaceable, the substance is not.
3. Process as close as possible to the source; make travel only what must travel.

- 4. The AI proposes, the human decides; validation remains human on everything critical.
- 5. Gamify the right gesture, never the raw figure; do not disguise a projection as a measurement.
- 6. The tool tends the client’s capacity to read, it never dispenses them from it.
- 7. At any moment, the client can take back control without losing anything.

I. Table of Correspondence — The Vocabulary of the Corpus and Existing Frameworks (Dated Form, July 2026)

The substance owes nothing to the standards of the moment; the practitioner, for their part, will meet them. This table, revisable and dated, gives working equivalences — without identity of meaning.

Corpus	Existing framework	What coincides / what differs
Chain of traceability; attributed act of inscription	Provenance / lineage (W3C PROV)	Same concern to go back up the chain; PROV models entities and activities, the corpus starts from the act and its subject.
Immutable fixation; change of state written before withdrawal	Event sourcing; append-only logs	Same primacy of the inscribed event over the current state; the corpus adds state lapse set by use.
Reader’s share practicable; reference corpus attached	FAIR principles (findable, accessible, interoperable, reusable)	FAIR equips circulation; the corpus recalls that reception cannot be decreed — a FAIR datum can remain mute for a given reader, or simply unread.
Freshness; use-by limit	TTL, data freshness / staleness	Same threshold mechanics; the corpus grounds the threshold in the reader’s use, not in technique.
The agreed; tended stable sources	Reference frameworks, controlled vocabularies, master data	Same function of shared agreement; the corpus insists: tended, and therefore perishable.

Translation choices

These notes say how to read the English words that render the terms of the corpus; they are fixed for the whole English translation by the *Translation Charter of the Corpus*.

- **datum / data.** The corpus treats the datum as an individual: constituted by an act, dated, carried by media. The translation therefore says *datum* in the singular and *data* as its plural.

- **drawing / to draw** (*prélèvement, prélever*): the gesture of drawing from the flow, as one draws water. *Draw* is kept for this gesture; **under-drawing** and **re-drawing** follow it. An **entry** (*relevé*) is the value inscribed; **input** renders *saisie*, the gesture of keying a datum in. The **apparatus** (*dispositif*) is what observes and draws; *device* keeps its everyday sense (a phone, a tablet).
- **canal; canal-keeper; workbench** (*canal; canalier; établi*): the canal dug in the river, never a *channel*; the canal-keeper is the data artisan (a craft yet to be created) who digs and tends it. The **workshop** works by hand what the **line** (*la chaîne*) processes in bulk.
- **tending / to tend** (*entretien, entretenir*): the care that keeps a canal, a flow or the agreed alive. **Maintenance** and *maintain* render *main-tenance* and *maintenir*, the family of tools and the technical act.
- **lapse; stale** (*péremption; périmée*): a relation, not a decay; a datum goes stale *for a use*. **State lapse** and the **use-by limit** (*durée limite d'utilisation*) name the threshold beyond which the gap from the real exceeds what use tolerates; the **validity window** is the time within it.
- **bearer / borne** (*portante / portée*): a datum that others exist only through, and those data; the borne datum **goes down in a cluster** (*descend en grappe*) with its bearer. The **borne link** (the parent field) is distinct from **reference** (the references field). *Bear* is kept for this pair; *carry* and *concern* render the other senses of *porter*.
- **armature; the agreed; tended stable source** (*armature; le convenu; source stable entretenue*): the bed of the canal, drawn from shared conventions, as against the water of the operational drawings. **Safeguard on form** (*Garde-fou de forme*) is the warning that the means described are dated.
- **the reserve** (*la réserve*): the frozen double withdrawn from the cycle of use, distinct from the 3-2-1 backup.
- **objectal moment** (*moment objectal*): the time during which a consolidated datum is handled as an object; **object ontology, residue of process** and **the inappropriability of the flow** follow Tome 1.
- **render; verisimilar** (*restituer; vraisemblable*): the storyteller **renders** what the stock holds; a **verisimilar datum** is inferred or estimated, as against an observed datum.
- **trace** (*trace*): reserved for the mark that the real leaves of itself. *Record* renders the ordinary sense; *traceability* renders *traçabilité*.
- **path; itinerary; route** (*chemin; parcours; voie*): *path* is kept for the three paths of non-reading; **itinerary** renders *parcours* (the **equipment itineraries**); *route* renders *voie*. The three stages of development are **immersion, adjustment, expansion**.
- **business core; heart of the craft; craft gesture** (*cœur métier; cœur de métier; geste métier*): the business core is the centre of the hexagon; the heart of the craft is the client's know-how, which computing does not touch. The **support matrix** and the **craft matrix** render *matrice de soutien* and *matrice de métier*. The **station** ren-

ders *le poste*, the sovereign server of the datum; *workstation* keeps its everyday sense. The **journal** renders *le journal* (of acts).

- **Code and examples.** The pseudo-code, the field names, the comments of the directory tree and the labels of the hexagon diagram are given in English, as they would be written in an English-language workshop; the business rules *RG1-RG4* (*règles de gestion*) become *BR1-BR4*. French identifiers with no English equivalent (the SIRET number) are kept. Where the French text says that business logic is written “in French”, the translation says “in plain language”.
- **Quotations.** Heraclitus is given in a current English form (“One never steps into the same river twice”). The formulas of the trilogy follow their English translation (“No reading is from nowhere”, “Without reading, nothing is actualized”, “Not all unread data are failures”).
- **Titles.** *Tried by the Flow* renders *À l’épreuve du flux*; this volume is *Technical Data*. The *Workshop Book* is *The Data Artisan’s Workshop Book*; the *Management System* is the *SGQA — Quality and Activity Management System*. The tools of the annex keep the titles fixed in *The Canal and the Workshops* (*Checklist of the Encountered Flow of Observation*, *Lapse Grid*, *Canal Review Template*, *Work Signature Sheet*).

Every Theory Gains in Practice

For the canal-keeper who wants to stock their workbench, for the apprentice who is learning the gestures, for the solopreneur who wants to understand what their tool does – and why.

One never steps into the same river twice. But one can learn to dig canals from it. After the trilogy *Tried by the Flow*, which spoke the substance, *Technical Data* gives the form: the **canal-keeper's technical manual**.

Tools and methods for digging, today, a digital canal — building what lasts, making the datum, making it circulate, making it legible and tending it. In the service of a single goal: offering the solopreneur a management of their activity that is fluid, legible, under control, and the serenity that comes with it. Not software one can no longer get out of, but a capacity that sets free.

June 2026 Edition

Tools change; the craft remains