European data space
for smart communities

Deliverable 5.2

The Cookbook: publishing and applying DT-aCoGo recipes in the Open Urban Platform

WP 5 – Communication and dissemination

Authors: Simay Batum (Future Insight), Corné Helmons (Gemeente Rotterdam)
Reviewers: Rick Klooster (Future Insight), Simay Batum (Future Insight), Corné Helmons (Gemeente Rotterdam)
Delivery date: 31 August 2026
Dissemination level: Public
Type: Report

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or of the granting authority. Neither the European Union nor the granting authority can be held responsible for them

Revision History

Author Name, Partner short name	Description	Date
Simay Batum, Future Insight Corné Helmons, Rotterdam	Initial draft: Cookbook principle, interface walkthrough and deployment wizard, based on the mockup at fi-group.github.io/dtacogo_recipes and the recipe logic and demosite report.	17 Aug 2026
Corné Helmons, Rotterdam	Added chapter 5 on the division of responsibility for scenario parameters, and the parameter-bounds finding in section 5.3.	31 Aug 2026
Simay Batum, Future Insight	Supplied the functional mockup and the underlying recipe fixtures.	Aug 2026
Corné Helmons, Rotterdam	Revision after OASC review: maturity terminology in the Executive Summary and chapter 9 (Table 9); status at project closure in the roadmap (Table 10) and in sections 3.4.1, 4.4 and 5.3; end-to-end overview added to section 1.2.	5 Oct 2026

Abbreviation	Definition	Abbreviation	Definition
API	Application Programming Interface	nLDT	National digital twin programme (Netherlands)
CIM	Common Information Model	OASC	Open & Agile Smart Cities
DCAT-AP-NL	Dutch application profile of the Data Catalogue Vocabulary	OGC	Open Geospatial Consortium
DGA	Data Governance Act	OUP	Open Urban Platform
DS4SSCC-DEP	Data Space for Smart and Sustainable Cities and Communities – Deployment	PPB	Purple Polar Bear (semantic interoperability layer)
DT-aCoGo	Digital Twinning as a Common Good	SDK	Software Development Kit
DTM	Digital Terrain Model	SSO	Single Sign-On
GPU	Graphics Processing Unit	TPG	The People Group
IAMA	Impact Assessment Human Rights and Algorithms	WFS	Web Feature Service
IAM	Identity and Access Management	WMS	Web Map Service
JSON	JavaScript Object Notation	WP	Work Package
LOD	Level of Detail	3D Tiles	OGC standard for streaming 3D geospatial content

Table of Contents

Revision History **2**
Executive Summary **5**
1 Introduction **7**
1.1 Purpose of this deliverable 7
1.2 Relation to the other deliverables 7
1.3 Scope, and the status of the mockup 8
1.4 How to read this document 8
2 The Cookbook principle **9**
2.1 From a recipe to a cookbook 9
2.2 Six design principles 9
2.3 Where the Cookbook sits in the architecture 9
3 The Cookbook interface **11**
3.1 Two ways in: guided search and the catalogue 11
3.2 The recipe record on screen 12
3.3 Constraining the shared catalogues 14
3.4 The four lifecycle tabs 14
3.4.1 Development — onboarding a provider 14
3.4.2 Deployment — standing the configuration up 15
3.4.3 User — the operational runbook 16
3.4.4 Orchestrator — the machine-readable projection 17
4 From recipe to running ecosystem: the deployment wizard **18**
4.1 Four steps, and why the order is fixed 18
4.2 Step 1 — Data ingredients 18
4.3 Step 2 — Tool and engine, and the capability indirection 20
4.4 Step 3 — Semantic alignment 21
4.5 Step 4 — Output and visualisation 23
4.6 The two-severity readiness model 23
4.7 The cost model 26
5 Where scenario parameters are set **27**
5.1 A boundary, not a gap 27
5.2 What the recipe publishes today 27
5.3 A defect in the contract 28
6 Publishing recipes **29**
6.1 The recipe record 29
6.2 The publication lifecycle 29
6.3 Metadata mapping 29
6.4 Versioning, and the relation to the nLDT Cookbook 30
7 Governance, ethics and trust **31**
7.1 Ownership and visibility 31
7.2 Roles are advisory 31
7.3 Licences travel with the recipe 31
7.4 Ethics and the Governance & Ethical Board 31
7.5 Logging 31
8 Replication across the three pilot cities **32**
8.1 Author once, consume three times 32
8.2 What changes, and what does not 32

8.3 The walkthrough as a worked example 33
9 Maturity, gaps and roadmap 34
9.1 What the mockup demonstrates 34
9.2 What it does not establish 35
9.3 Roadmap 35
10 Conclusions 36
Annex 1 — The four reference recipes 37
Annex 2 — Screenshot register 38
Annex 3 — Recipe record projection 39
About the European Data Space for Smart Communities 41

Executive Summary

Deliverable D4.1 defined what a recipe is: a declarative, reusable description of the datasets, tools, parameters and outputs needed to reproduce a sectoral analysis on an Open Urban Platform. This deliverable, D5.2, describes the **Cookbook** — the principle and the interface by which those recipes are published, found, and turned into a working configuration of data services, applications, visualisation and governance inside a hub in the OUP.

The Cookbook exists because a recipe on its own does not replicate anything. A planner in 's-Hertogenbosch who reads the Rotterdam stormwater recipe still has to work out which of their own datasets satisfy each requirement, whether their formats are acceptable, which locally registered tool can stand in for the one Rotterdam used, whether the attribute names line up, and what any of it will cost. The Cookbook is the layer that answers those questions against the consuming city's own hub, and that refuses to let a configuration be declared ready while any of them is unresolved.

Three design commitments shape the whole thing. First, the Cookbook is a **catalogue, not an installer**: it produces a checked, costed configuration and a resolved plan, and it stops there. Nothing is provisioned, no simulation is run, no money changes hands. Second, a recipe binds a **capability, not a vendor**: it declares that a flood simulation engine is required, and every registered application in the consuming hub that offers that capability is offered as a candidate. Third, the Cookbook is **advisory, not authorising**: the roles, phases and readiness signals it publishes help a user judge whether a recipe fits their situation, but access to the underlying datasets and applications remains governed by the hub's own access control, exactly as described in D2.3.

A working mockup of the proposed functionality has been built on the Open Urban Platform interaction model and is publicly available at https://fi-group.github.io/dtacogo_recipes/. It carries four reference recipes covering flooding, urban heat, mobility and emergency evacuation, and three city hubs — Rotterdam, 's-Hertogenbosch and Tampere — each holding an equivalent base dataset. The mockup is a design instrument: it is populated with fixtures and performs no live execution. Every screenshot in this document was taken from it on 31 August 2026.

The heart of the interface is a four-step deployment wizard. Its order — data ingredients, then tool and engine, then semantic alignment, then output and visualisation — is a genuine dependency rather than a convenience, because a dataset-and-tool pairing can only be checked for semantic alignment once both sides of the pairing exist. Readiness is tracked at two severities. A **blocked** requirement, where no decision has been taken at all, stops the user advancing. A **pending** requirement, where a plan exists but no correctly formatted resource has yet been registered, permits the user to continue planning but prevents deployment. This distinction is what lets a city budget for a recipe it cannot yet run, which is the normal case at the start of a project.

One boundary deserves to be stated plainly, because it is easy to mistake for a gap. The Cookbook does not provide a screen on which a planner sets a rain intensity or a simulation duration. That is deliberate. Scenario parameters are set in the visualisation layer — in the Unreal Digital Twin application, through a plug-in in which the user picks the calculation method and defines its variables — because that is where the result is seen and where iterating on a scenario actually happens. The Cookbook's job is to publish the parameter **contract**: which variables exist, of what type, in what unit, with what default. Chapter 5 sets out this division of responsibility, and identifies one concrete defect in it: the published record currently omits the bounds of each parameter, which the plug-in needs in order to render a control correctly.

The deliverable closes with an assessment of the maturity of each aspect at project closure, and a roadmap. The interaction model, the readiness semantics, the capability indirection and the governance metadata have been demonstrated in the mockup, against fixture data and by its designers; they have not yet been implemented operationally or tested with municipal users. Outside the mockup, the pilot did make two of the underlying capabilities operational: the semi-automated delivery of data to the three models through the Common Information Model and the OUP, and the setting of scenario parameters in the Unreal Digital Twin application. The PPB mappings work in the PPB platform, but how to register them in the OUP is still open. Live catalogue federation, real execution through the Cook orchestrator, and the machine-readable publication of recipes against OGC API — Records are future work. User testing with planners moves to the first replication city, Groningen. The remaining items, including the two open defects identified in this deliverable (the missing parameter bounds and cross-dataset attribute matching), are taken up in the Geonovum Testbed 2026, which is to deliver a new version of the mockup and recommendations to the standards for real-world implementation.

1 Introduction

1.1 Purpose of this deliverable

This deliverable describes the Cookbook: the principle by which DT-aCoGo recipes are published and shared, the interface through which a user finds a recipe and applies it, and the way that interface guides the deployment of an ecosystem of data services, applications, visualisation and governance within a hub on the Open Urban Platform.

It has three audiences. For **OASC and the wider DS4SSCC community** it is a transferable description of a replication mechanism that is not specific to the three pilot cities. For **the pilot cities themselves** it is the operational account of how a recipe authored in one city is consumed in another. For **platform and model providers** it sets out what a provider must publish for their application to appear as a candidate in someone else's configuration.

1.2 Relation to the other deliverables

D5.2 sits at the end of a chain and depends on all of it. It does not restate the content of the earlier deliverables; it assumes them.

Deliverable	What D5.2 takes from it
D2.1 — Roles and responsibilities	The role vocabulary used in the recipe record's *targetRoles* field. The Cookbook publishes these as advisory metadata; it does not use them to grant access.
D2.2 — Use cases and requirements	The four sectoral use cases that the four reference recipes in the mockup instantiate: flooding, urban heat, mobility, emergency response.
D2.3 — Checks and guidance by the Governance & Ethical Board	The governance and ethics process that gates publication of a recipe, and the access-control model that the Cookbook defers to rather than duplicates.
D3.1 — Model integration and CIM	The Common Information Model and the Purple Polar Bear semantic layer. Step 3 of the wizard surfaces PPB mappings by name; chapter 9 of D3.1 is the narrative ancestor of the wizard's step order.
D3.2 — Model usage	The three analytical models — Rana, Tygron and dat.Mobility — and their API descriptions in Annexes A2–A4 of D3.1, which define what an execution contract must contain.
D3.6 — Ethical assessment of models	The ethical constraints that attach to a model, and therefore to any recipe that binds it.
D4.1 — Recipes	The recipe concept, the *conformsTo* conformance classes, and the four reference recipes. D5.2 is the publication and application layer for exactly these objects.
D5.1 — Communication and dissemination	The dissemination context. The Cookbook is the project's principal instrument for replication beyond the three pilots.

Table 1 — What D5.2 draws from each preceding deliverable.

Read end to end, a recipe follows five steps across D4.1 and D5.2. Steps 1 to 4 are what the Cookbook covers; step 5 lies outside it.

1. Author (D4.1). A city or provider describes an analysis as a recipe — the datasets, the capability, the parameters and the output — and checks it against the conformsTo conformance classes.

2. Review and publish (D2.3; chapters 6 and 7). The recipe passes the checks of the Governance & Ethical Board and is published in the Cookbook with its owner hub, licence, target roles and version.

3. Discover and inspect (chapter 3). A consuming city finds the recipe through guided search or the faceted catalogue, and reads its record.

4. Configure (chapter 4). The deployment wizard binds each requirement to the city's own datasets, registered applications, semantic mappings and viewer, and reports what is blocked, what is pending and what it will cost. The result is a checked, costed configuration.

5. Run and iterate (D3.2; chapter 5). The analysis runs with the bound models. The user sets scenario parameters in the Unreal Digital Twin application, within the parameter contract published with the recipe. The Cookbook does not perform this step; chapter 9 sets out what that means for maturity.

1.3 Scope, and the status of the mockup

What is in scope: the Cookbook principle; the publication model for recipes; the user interface for discovery, inspection and configuration; the readiness and cost model; the governance metadata carried on a recipe; and the replication path across the three pilot cities.

What is out of scope: the authoring of recipe content itself, which is D4.1; the internal workings of Rana, Tygron and dat.Mobility, which are D3.1 and D3.2; and the execution runtime, which does not yet exist in the project and is discussed only as a roadmap item in chapter 9.

The mockup is a **design instrument, not a product**. It runs entirely in the browser against fixture data. No request leaves the page, no simulation is performed, no invoice is raised, and every recipe card carries the label *Status: Mocked — no live execution*. Its purpose is to make the interaction model concrete enough to be criticised. Where the mockup's behaviour diverges from what a production Cookbook would need to do, this document says so rather than describing the intent as though it were built.

1.4 How to read this document

Chapter 2 sets out the Cookbook principle and where it sits in the OUP information architecture. Chapter 3 walks through the interface. Chapter 4 covers the deployment wizard in detail, including the readiness and cost models. Chapter 5 addresses where scenario parameters are set, which is the one place where responsibility passes out of the Cookbook and into the Digital Twin viewer. Chapter 6 describes publication, versioning and metadata. Chapter 7 covers governance and ethics. Chapter 8 works through replication across the three cities. Chapter 9 assesses the maturity of each aspect at project closure, and sets out the roadmap with the status of each item. Chapter 10 concludes.

Interface elements are named throughout exactly as they appear on screen, so that a reader with the mockup open in front of them finds the same words in both places. Where this deliverable proposes a change to a label, to a tab's content or to the published record, it says so explicitly and states what the current behaviour is.

Screenshots are taken from the public mockup and are captioned with the route they were captured from. Annex 2 lists them all with their routes so that any reader can reproduce them.

2 The Cookbook principle

2.1 From a recipe to a cookbook

D4.1 established the recipe as the unit of reuse: an ingredient list plus a method, declaring the datasets a given analysis needs, the capability that will process them, the parameters that shape the run, and the form the result takes. A recipe is deliberately abstract about the specific resources involved, because the whole point is that it should survive being carried from one city to another.

That abstraction is also the problem. A recipe that says "a digital terrain model in 3dterrain format" is portable precisely because it does not say which terrain model. Someone has to close that gap. Doing it by hand means reading the recipe, searching the local catalogue, guessing at whether a format is close enough, discovering three weeks later that an attribute the engine requires is absent from the local data, and having no record of any of those decisions.

The Cookbook closes that gap systematically. It is the register in which recipes are published, the search surface through which they are found, and the guided process through which a recipe's abstract requirements are bound to the concrete resources of a particular hub. Its output is not a running system. Its output is a **resolved, checked and costed configuration**, together with an explicit list of everything that is still missing.

2.2 Six design principles

Principle	What it means in practice
A catalogue, not an installer	The Cookbook produces a plan. It does not provision infrastructure, execute models, or transact. The final action available to a user is Review & Deploy, which hands a resolved configuration onward.
Capability, not vendor	A recipe binds a required capability such as *flood simulation*. The application named by the authoring city is used only to identify that capability and is then set aside; every locally registered application offering it becomes a candidate.
Derived, not duplicated	The Cookbook does not keep its own copy of the dataset and application catalogues. The Compatible Data and Compatible Apps views re-render the platform's own catalogues with the recipe's constraints locked on.
Advisory, not authorising	Roles, project phases and findability are published to help a user judge fit. They never substitute for the access control defined in D2.3.
Plan first, prove later	A configuration may be completed and costed while resources are still being sourced. Deployment is withheld until every requirement points at a real, correctly formatted resource.
Say what is missing, precisely	Every unmet requirement is reported against the specific slot, dataset pairing or attribute that caused it — never as a generic validation failure.

Table 2 — The design principles the Cookbook is built on.

2.3 Where the Cookbook sits in the architecture

The DT-aCoGo architecture described in D3.1 runs from municipal source data, through the OUP hub, through the Purple Polar Bear semantic layer that translates source data into the Common Information Model, into the analytical models, and out into the Digital Twin viewer. The Cookbook is

not another stage in that pipeline. It sits above it, as the layer in which a configuration of that pipeline is described, published, discovered and resolved.

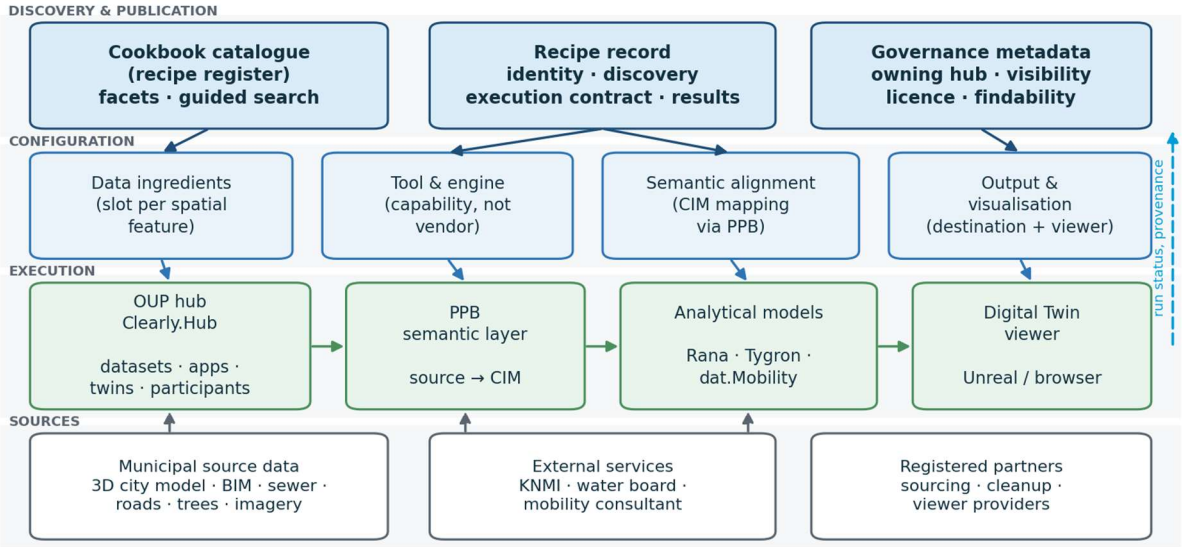


Figure 1 — The Cookbook in relation to the DT-aCoGo architecture. Discovery and publication sit above configuration, which resolves onto the execution stack described in D3.1. Run status and provenance return to the catalogue.

Within the Open Urban Platform's own information model, described in the Clearly.Hub documentation, the Cookbook introduces recipes as a **peer register** alongside hubs, datasets, applications and digital twins. A recipe is the object that binds the other four together: it references datasets by format and role, applications by capability, a hub as its owner and destination, and a digital twin as the surface on which its result is presented.

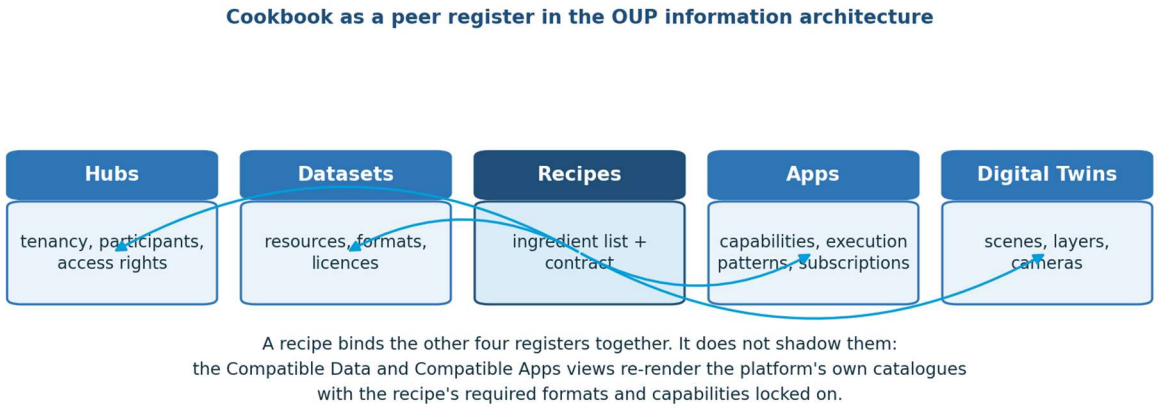


Figure 2 — Recipes as a peer register. A recipe constrains the platform's existing catalogues rather than shadowing them with a private copy.

3 The Cookbook interface

This chapter walks through the interface as a user meets it. Chapter 4 then treats the deployment wizard, which is substantial enough to warrant its own chapter. The actual mockup of the Cookbook can be found on https://fi-group.github.io/dtacogo_recipes/

3.1 Two ways in: guided search and the catalogue

The Cookbook has to serve two quite different users. One knows what they are looking for and wants the catalogue. The other has a problem — a neighbourhood that floods, a heat complaint, a development that will change traffic — and does not know that a recipe is the kind of thing that would help. Guided search is built for the second user. It asks about the problem, the project phase, the user's role and the theme, and reports a live count of how many recipes would still match, so the user can see the effect of each answer before committing to it. The whole sequence can be skipped.

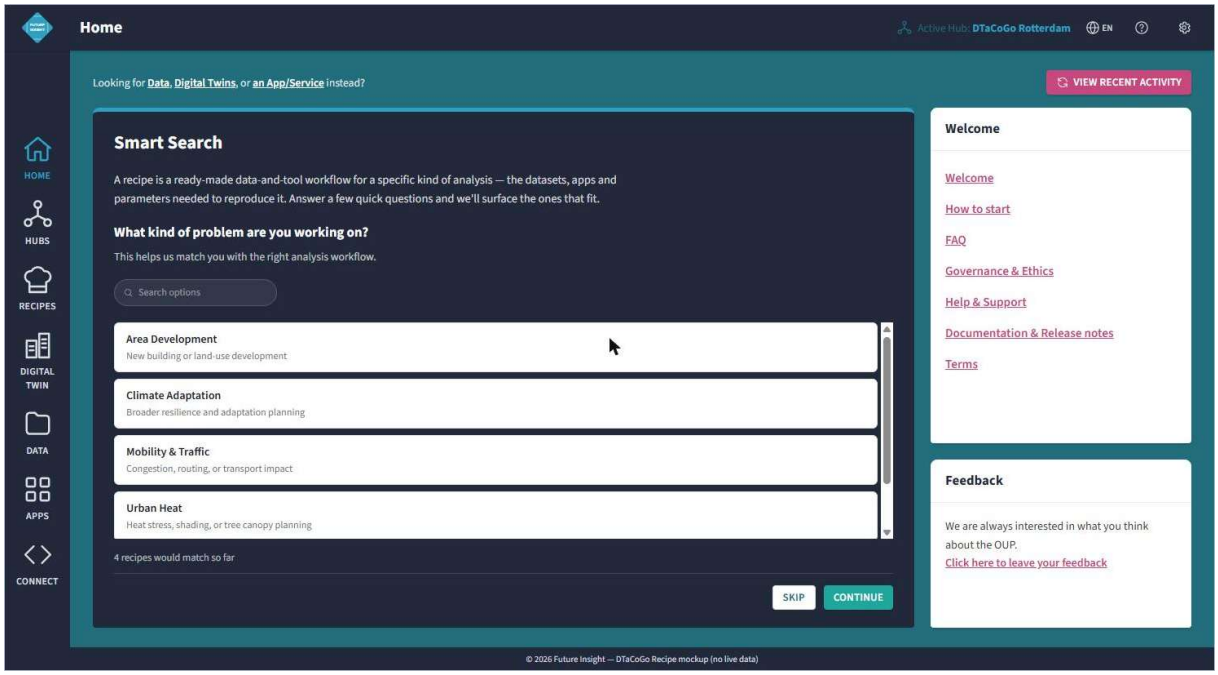


Figure 3 — Guided search on the home screen. The running count ("4 recipes would match so far") makes the effect of each answer visible. Route: #/

The catalogue is the direct route. Each recipe is a card carrying its title, a plain-language description, its owning hub, its licence, and its execution status. Nine facets narrow the list: owner hub, execution pattern, project phase, target roles, theme, location, input data format, output data format and findability. Project phase is rendered as a four-stop track — preparation, initiative, definition, design — because a recipe is typically useful across a span of phases rather than at a single point.

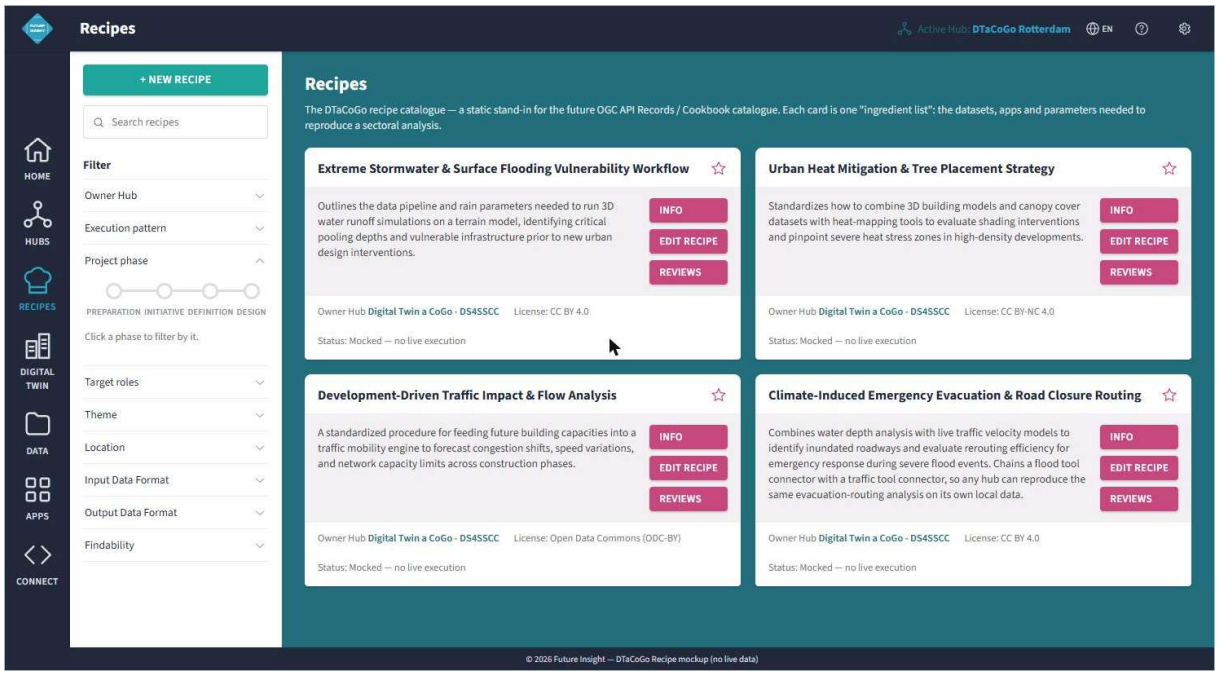


Figure 4 — The recipe catalogue with its nine facets. Each card is one ingredient list. Route: #/recipes

Two facets deserve comment because they are governance instruments rather than search conveniences. **Owner hub** is what makes replication legible: a user in Tampere can see at a glance that a recipe originates in Rotterdam and reason about how far its assumptions travel. **Findability** exposes the visibility setting of the recipe record itself, so that a user understands whether what they are looking at is public, shared with a named set of hubs, or private to their own.

3.2 The recipe record on screen

Opening a recipe presents its full record. The upper half is identity and discovery metadata: title, owning hub, findability, description, execution pattern, project phases, target roles, themes and location.

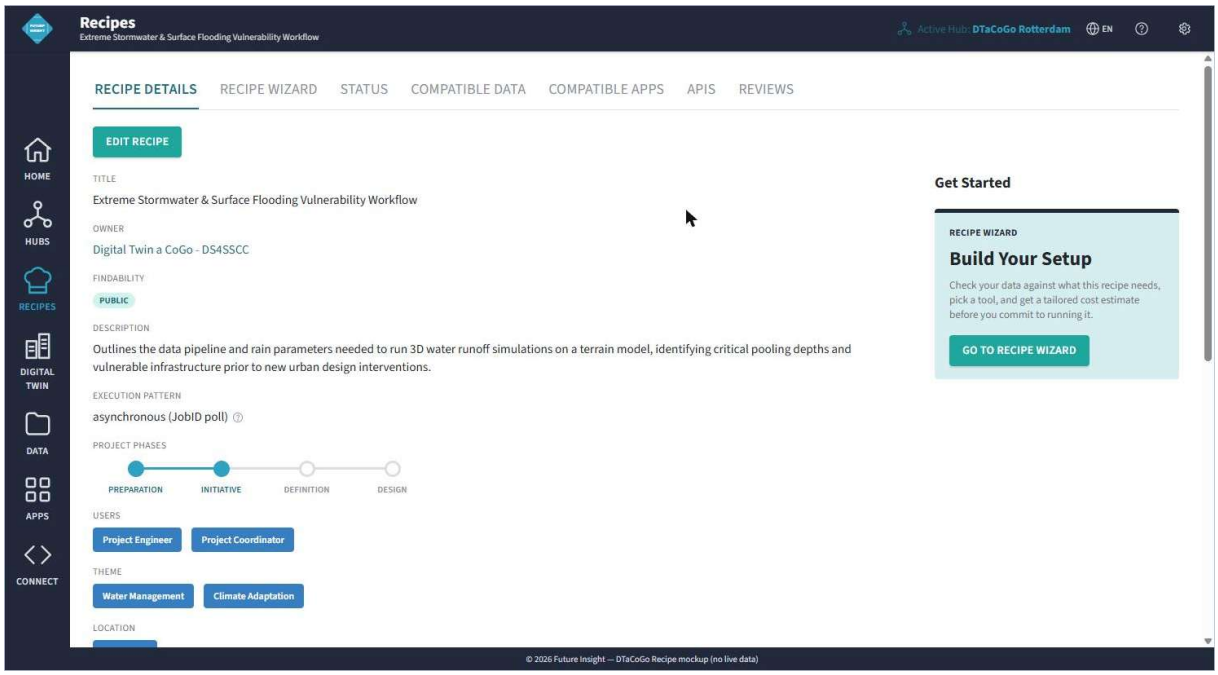


Figure 5 — The recipe detail view for the stormwater recipe. Route: #/recipes/recipe-rana/details

The lower half is the execution contract: the required input data expressed as named roles, the input and output formats, and a preview of what the result looks like. The preview in Figure 6 is itself informative — it is a frame from the Unreal Digital Twin viewer showing an extreme rainfall scene, with the scenario control visible in the viewer's own toolbar. That is the division of responsibility described in chapter 5, made visible.

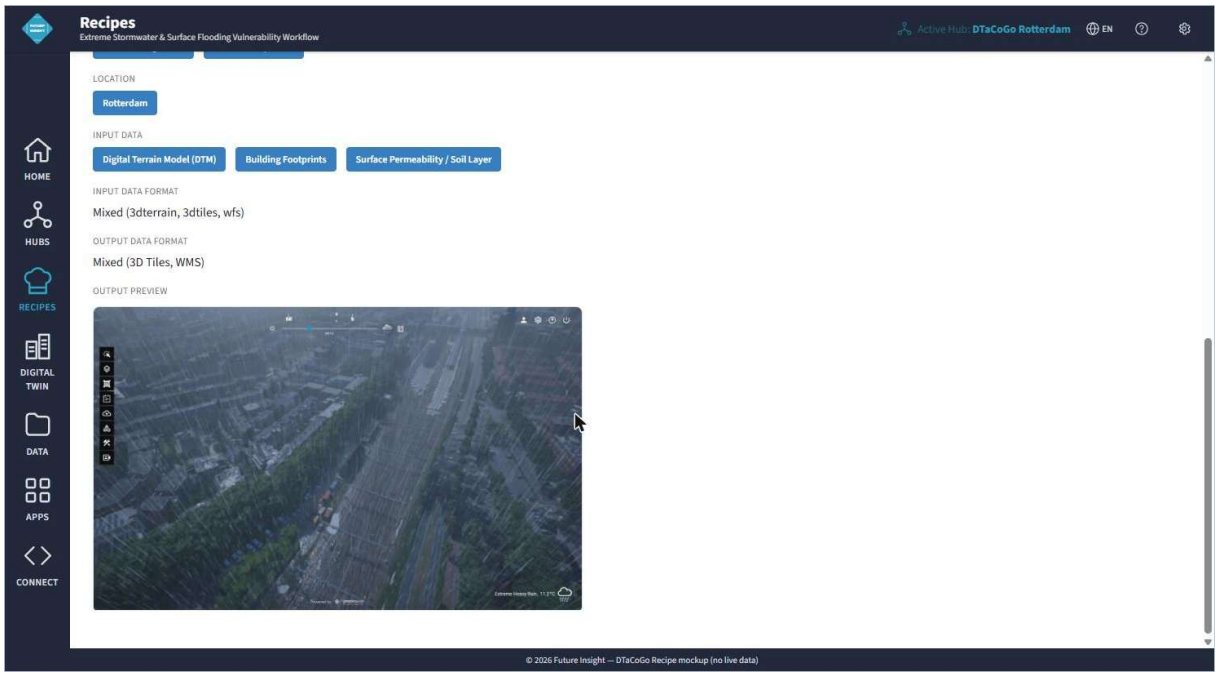


Figure 6 — Input roles, formats and output preview. The preview is a frame from the Unreal Digital Twin viewer. Route: #/recipes/recipe-rana/details

The execution pattern field — here *asynchronous (JobID poll)* — is worth singling out. It tells an integrator how the bound model must be driven, and it comes directly from the API descriptions in Annexes A2 to A4 of D3.1. A recipe that declares an asynchronous pattern cannot be satisfied by a tool that only supports synchronous invocation, however well its capability matches.

3.3 Constraining the shared catalogues

The Compatible Data view is the clearest expression of the derived-not-duplicated principle. It is the platform's ordinary dataset catalogue, with the recipe's required formats pre-selected and locked. In Figure 7 the format facet shows *3dterrain*, *3dtiles* and *wfs* checked, and a marker indicates that the constraint comes from the recipe rather than from the user. The user can still search, sort and filter on everything else.

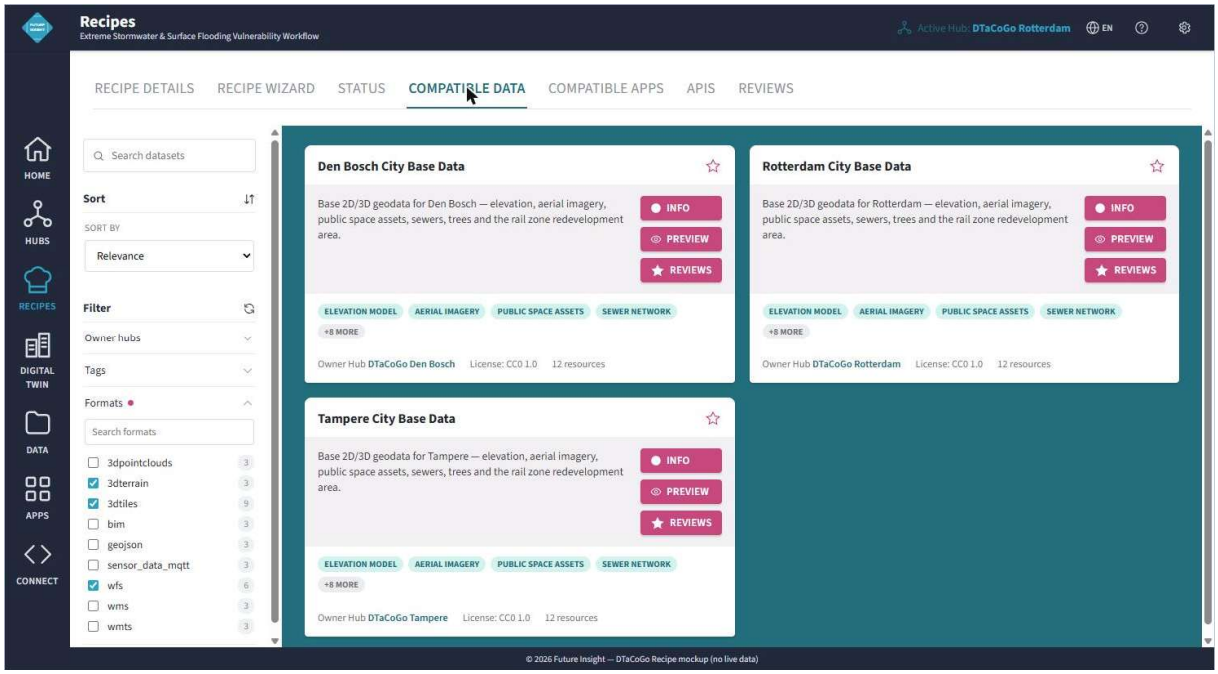


Figure 7 — Compatible Data: the shared dataset catalogue with the recipe's formats locked on. Route: #/recipes/recipe-rana/data

The result shown here also previews chapter 8. All three city base datasets — Den Bosch, Rotterdam and Tampere — satisfy the stormwater recipe's format requirements. The recipe is portable across the three hubs at the level of format before any city-specific work is done. Compatible Apps behaves identically for applications, filtered by the capability the recipe requires.

3.4 The four lifecycle tabs

Behind the recipe's edit view sit four tabs that describe the recipe across its lifecycle rather than at the moment of configuration. They answer four different questions asked by four different people.

3.4.1. Development — onboarding a provider

The Development tab, headed **Connect apps to the OUP**, lists what an organisation must do to connect an application to the OUP so that it can appear as a candidate in anyone's recipe: set up the app hub, authentication, subscription plans and billing; publish the APIs; describe the SDK-light used to implement a plug-in in a visualisation application; connect single sign-on; and implement identity and access management.

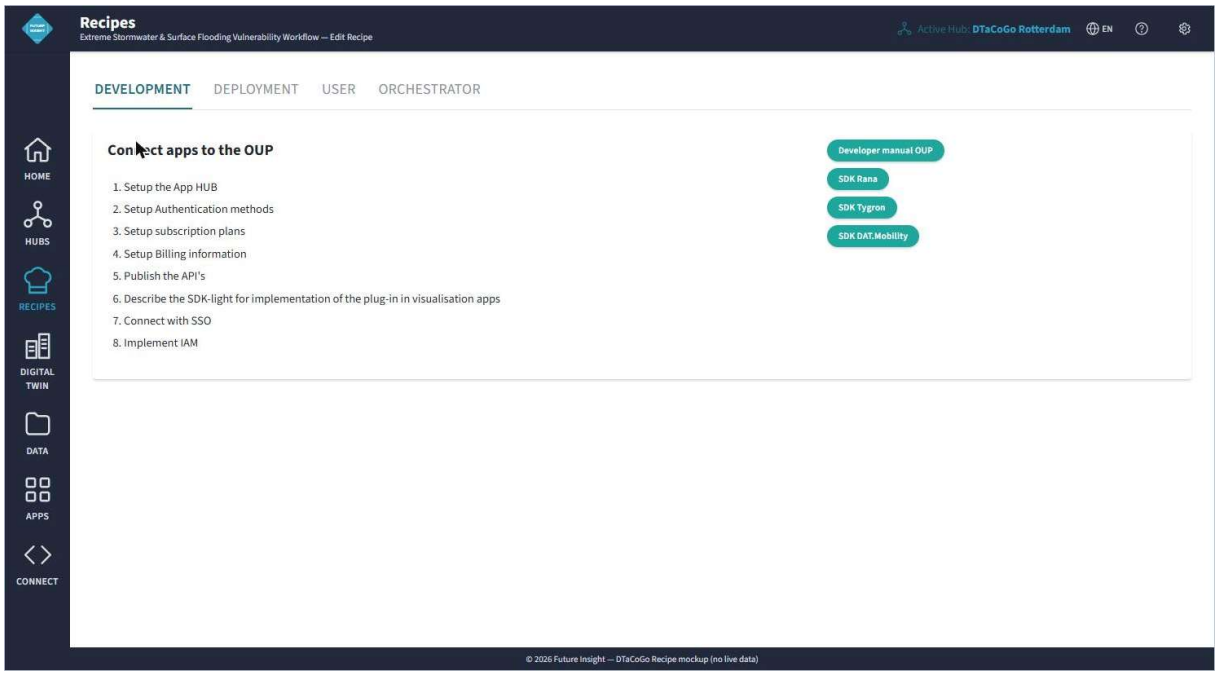


Figure 8 — The Development tab: a provider onboarding checklist, with links to the developer manual and the three model SDKs. Route: #/recipes/recipe-rana/edit

This content is **identical for every recipe**, and that is correct: it is the onboarding obligation of a provider joining the platform, not a property of any one analysis. The tab label is therefore narrower than its content. This deliverable keeps the label the interface uses — Development — and describes its content for what it is, a provider onboarding checklist, so that the document and the screen agree. The project has decided to rename the tab Provider Onboarding to match its content. The rename had not been applied to the mockup at project closure, so the screenshots and this chapter still show the current label (Table 10).

Item six on that list is load-bearing for chapter 5. The SDK-light is the mechanism by which a model provider ships the plug-in through which a user later sets that model's variables inside the viewer.

3.4.2. Deployment — standing the configuration up

The Deployment tab is a two-part checklist, headed **In the Hub** and **In the Analyses Models**, covering what must be done in the hub — generic settings, authentication, billing, connecting datasets and applications, adding subscriptions, describing the digital twin's camera position and available layers, and inviting users and assigning their roles — and what must be done inside the analytical model: set up the project, load the data from the OUP, configure the calculation model, publish the simulation.

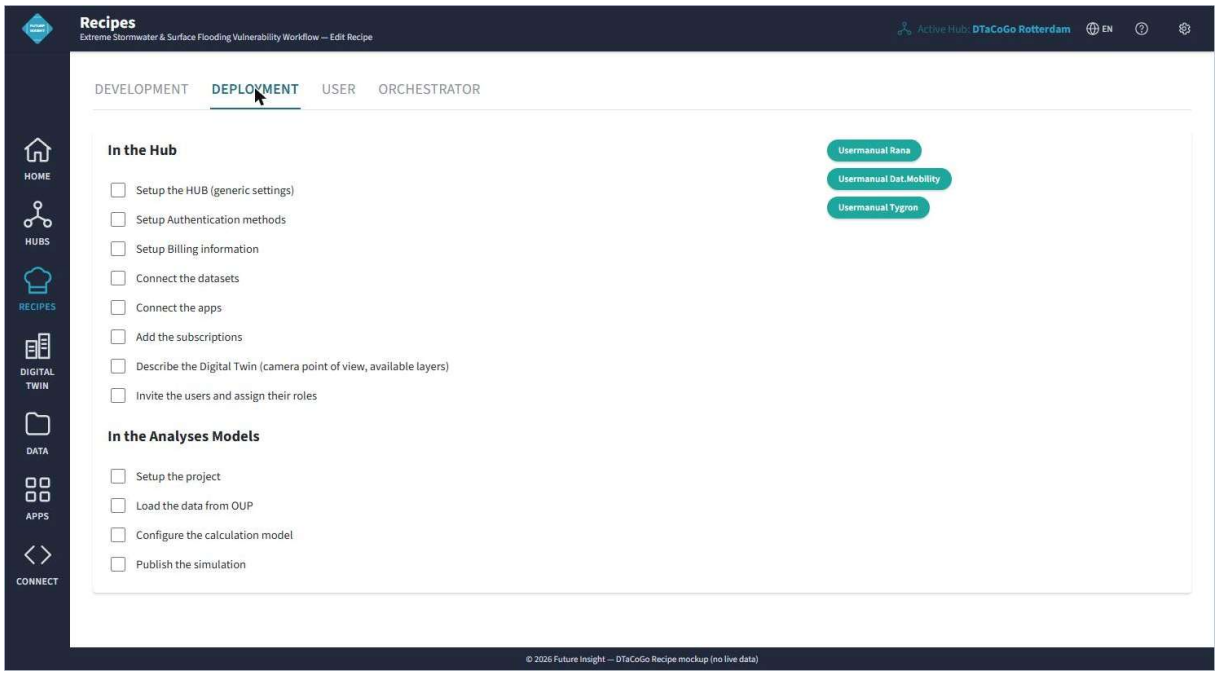


Figure 9 — The Deployment tab, split between hub configuration and model configuration, with links to the three model user manuals. Route: #/recipes/recipe-rana/edit/deployment

3.4.3. User — the operational runbook

The User tab is the seven-step runbook an end user follows once everything is in place: start the Unreal or web viewer, log in with OUP credentials and select the hub, load the digital twin, start the available plug-ins, start Rana, pick the calculation model and define the variables, and run the simulation.

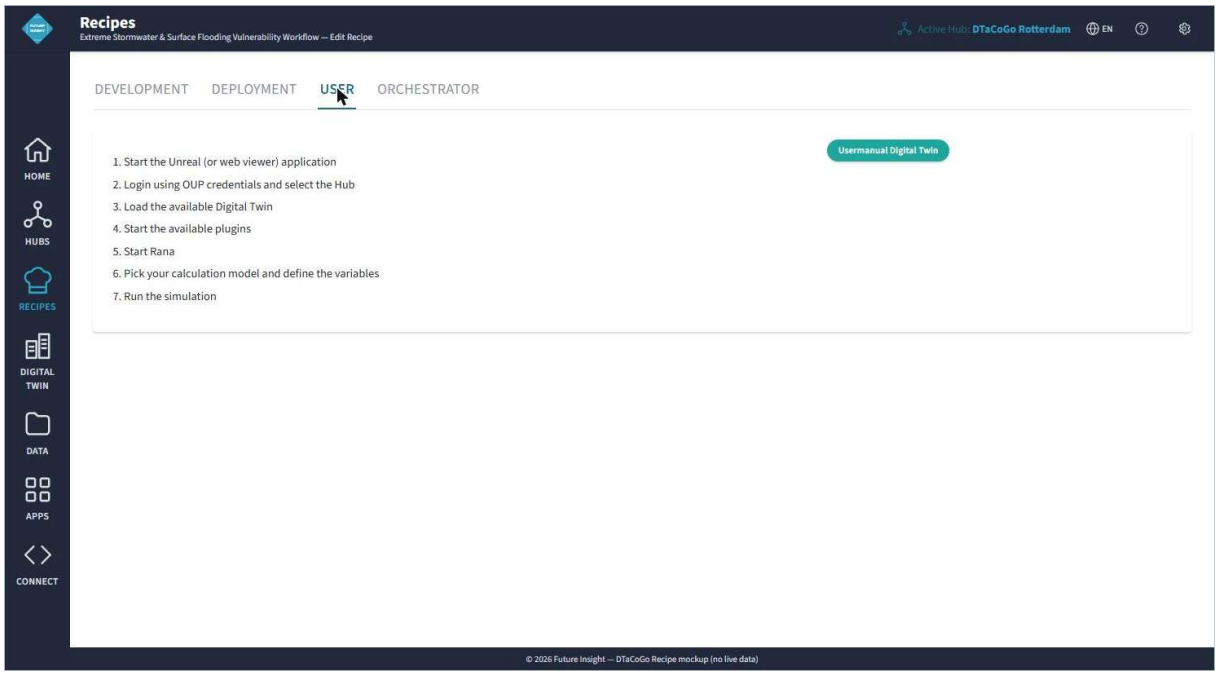


Figure 10 — The User tab. Step 6 is where scenario variables are set — in the viewer, not in the Cookbook. Route: #/recipes/recipe-rana/edit/user

Step six is the single most important sentence in the interface for understanding the boundary of the Cookbook, and chapter 5 is devoted to it.

3.4.4. Orchestrator — the machine-readable projection

The Orchestrator tab shows the same recipe as a JSON document intended for a future Cook orchestrator. It opens with the conformance classes defined in Annex 1 of D4.1, then carries identity, governance, discovery metadata, the parameter block, the bound input datasets and the compatible applications.

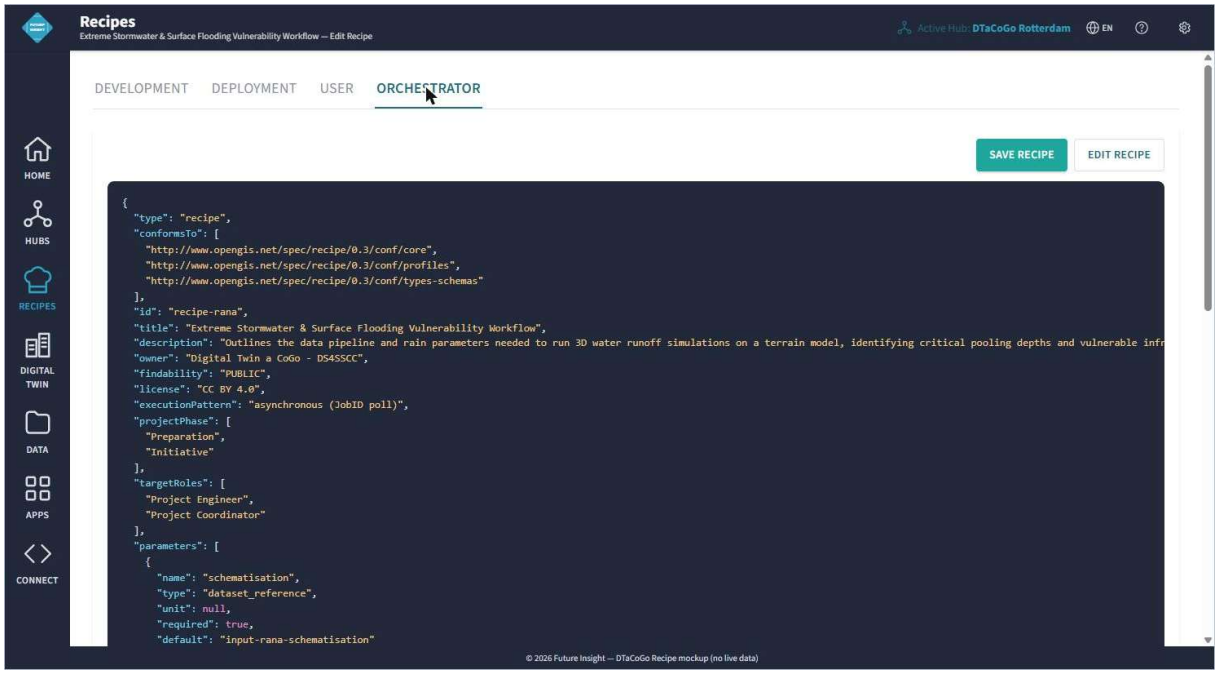


Figure 11 — The Orchestrator tab: the recipe as a machine-readable record, conforming to the classes defined in D4.1 Annex 1. Route: #/recipes/recipe-rana/edit/orchestrator

This projection matters more than its unassuming presentation suggests. It is the artefact that would be published to an OGC API — Records catalogue, consumed by an orchestrator, and version-controlled. Section 6.3 sets out its metadata mapping, and section 5.3 identifies a defect in it.

4 From recipe to running ecosystem: the deployment wizard

The wizard is where a recipe stops being a document and becomes a configuration. It takes the abstract requirements of the recipe and binds each one to something concrete in the consuming hub, refusing to declare the result ready while anything is unresolved.

4.1 Four steps, and why the order is fixed

On screen the steps are labelled **Data Ingredients**, **Tool & Engine**, **Semantic Alignment** and **Output & Visualization**; this document refers to them in running prose as data ingredients, tool and engine, semantic alignment, and output and visualisation. The sequence is a dependency, not a matter of taste. Semantic alignment checks whether a particular dataset can be read by a particular tool, which is only a meaningful question once both have been chosen. Choosing an output destination sensibly requires knowing what the tool will produce. Any other order would ask the user to answer questions whose terms do not yet exist.

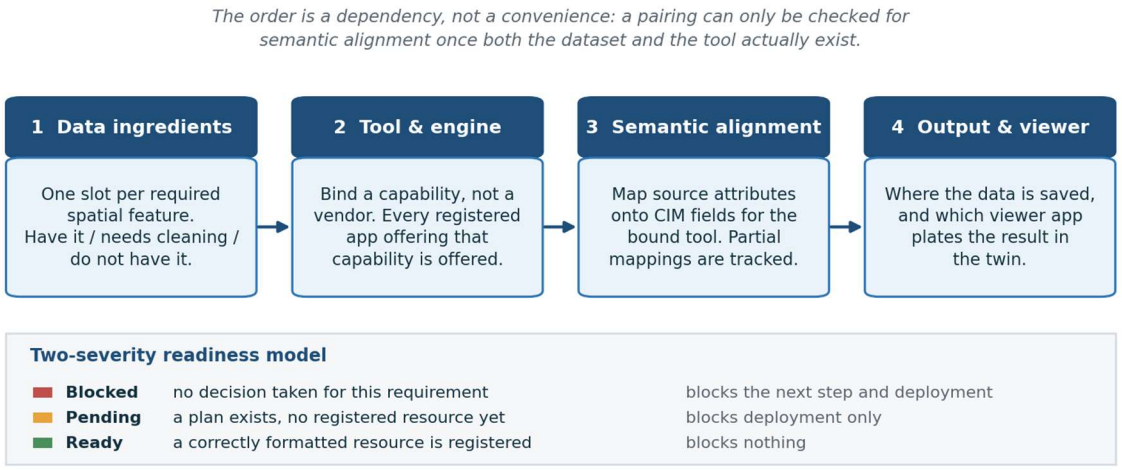


Figure 12 — The four wizard steps and the two-severity readiness model.

4.2 Step 1 — Data ingredients

The recipe's input requirements are presented as numbered slots, one per required spatial feature. Each slot names the feature, states the required format as a badge, and explains in plain language what the data is for and why the model needs it. Alongside the slots the step displays the recipe's data protocols and formats and its execution constraints.

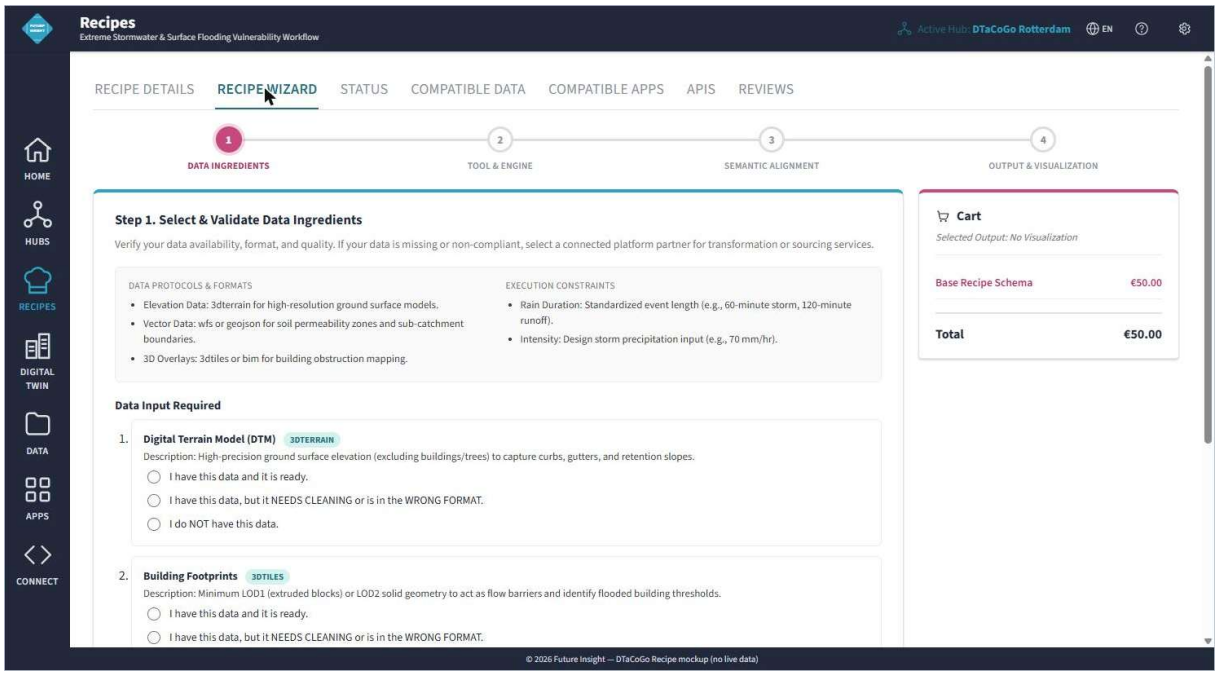


Figure 13 — Step 1. Three ingredient slots, the recipe's protocol and constraint summary, and the cart. Route: #/recipes/recipe-rana/wizard

Each slot offers exactly three answers, and this restraint is the design's main insight. **I have this data and it is ready** opens a resource picker. **I have this data, but it needs cleaning or is in the wrong format** asks the user first to flag which resource is deficient and then offers transformation partners. **I do not have this data** offers sourcing partners, or an explicit declaration that the user will handle it outside the platform. Three answers, because that is the real decision space; anything finer would be a taxonomy the user has to learn.

The resource picker inherits the recipe's constraint. In Figure 14 the format facet is locked to *3dterrain* for the terrain slot, and the user drills into a city dataset to reach the individual resource. A user cannot bind a resource that does not satisfy the recipe, because such resources are never shown.

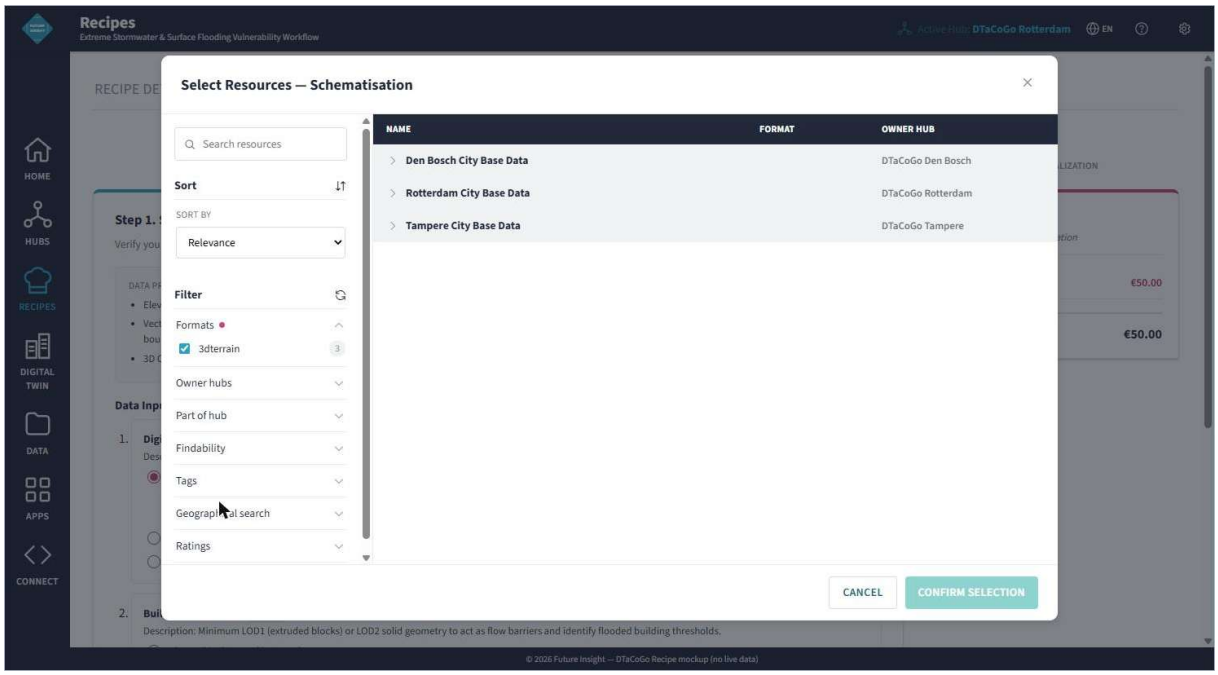


Figure 14 — The resource picker, locked to the format the slot requires. Route: #/recipes/recipe-rana/wizard

Where the user has neither the data nor a way to prepare it, the platform's registered partners are offered inline with their prices, and the choice is recorded as part of the configuration rather than left as an action item in someone's notes.

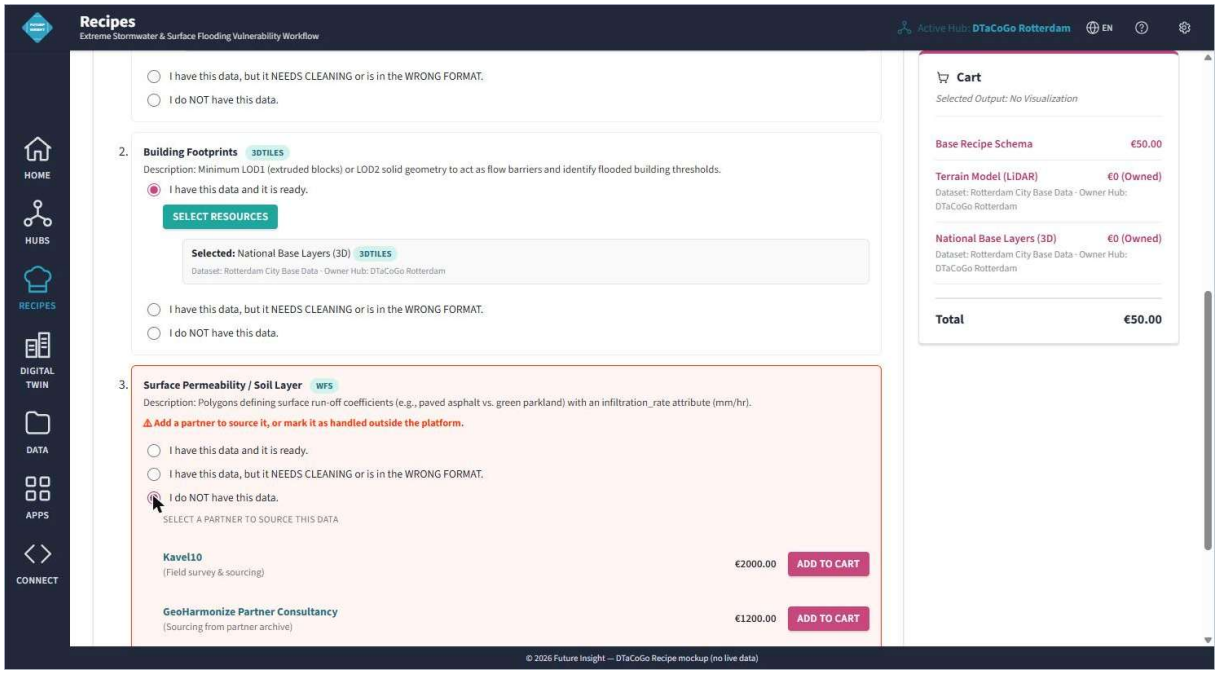


Figure 15 — Sourcing partners offered against an unmet requirement, with an explicit opt-out for sourcing outside the platform. Route: #/recipes/recipe-rana/wizard

4.3 Step 2 — Tool and engine, and the capability indirection

The recipe requires a flood simulation engine. It does not require Rana. The distinction is the mechanism on which replication depends.