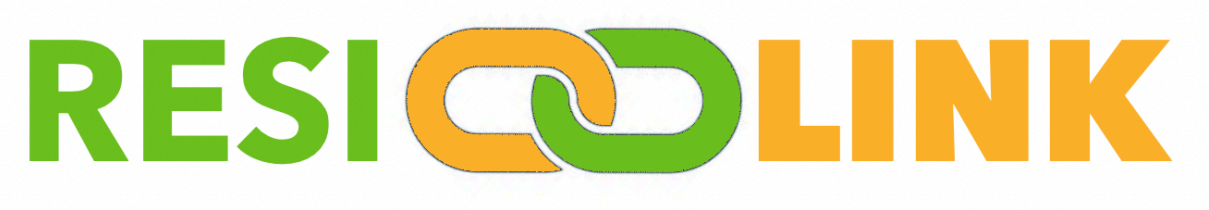


Increasing Resilience of Smallholders with Multi-Platforms Linking Localized Resource Sharing

Deliverable D4.2b

Final report software templates demonstrating RESILINK's open API

Responsible Editor:	UPPA
Contributors:	ORANGE
Document Reference:	RESILINK D4.2b
Distribution:	Public
Version:	1.1
Date:	May 2026

CONTRIBUTORS TABLE

DOCUMENT SECTION	AUTHOR(S)
SECTION 1	C. Pham (UPPA)
SECTION 2	A. Cazaux (UPPA)
SECTION 3	A. Cazaux (UPPA)
SECTION 4	M. Despland (ORANGE)

DOCUMENT REVISION HISTORY

Version	Date	Changes
V1.1	May 4th, 2026	PUBLIC RELEASE
V1.0	April 27th, 2026	FIRST DRAFT VERSION FOR INTERNAL APPROVAL
V0.1	April 20th, 2026	FIRST RELEASE FOR REVIEW

EXECUTIVE SUMMARY

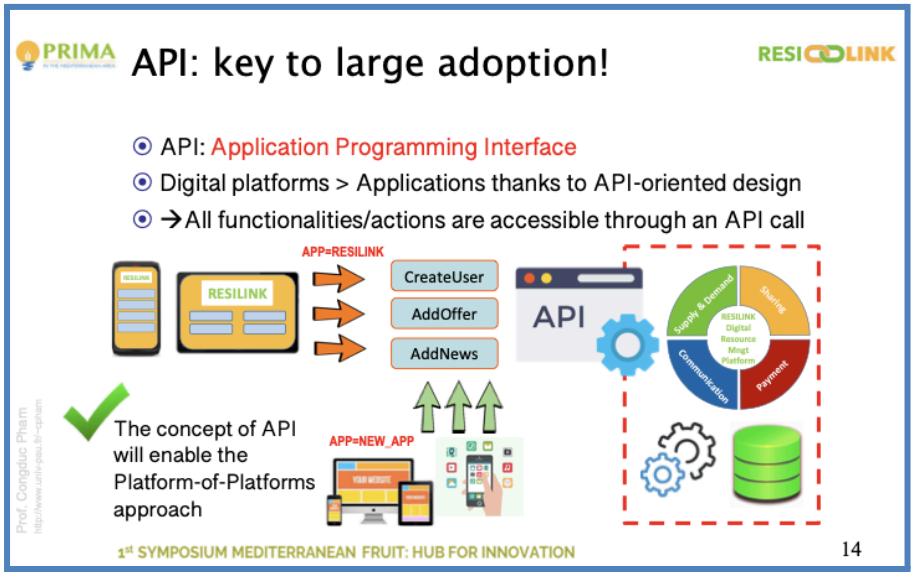
This Deliverable D4.2b presents how the developed core functionalities of the RESILINK platform can be used through its open API in other application domains to build resource sharing digital platforms.

Table of Contents

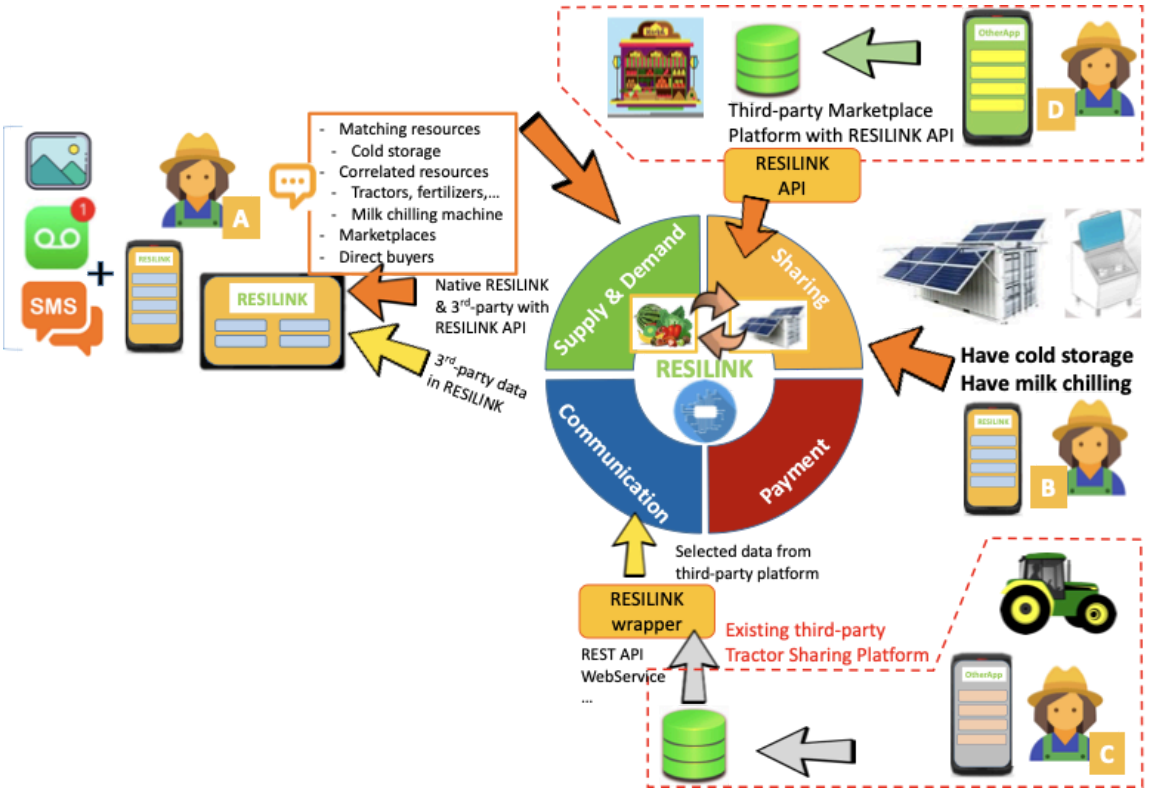
1. INTRODUCTION

RESILINK addresses local innovation capacity and facilitates technology appropriation by developing the digital resource management platform in open-source with an extensive open API to maximize re-utilisation and facilitate the integration of new platforms.

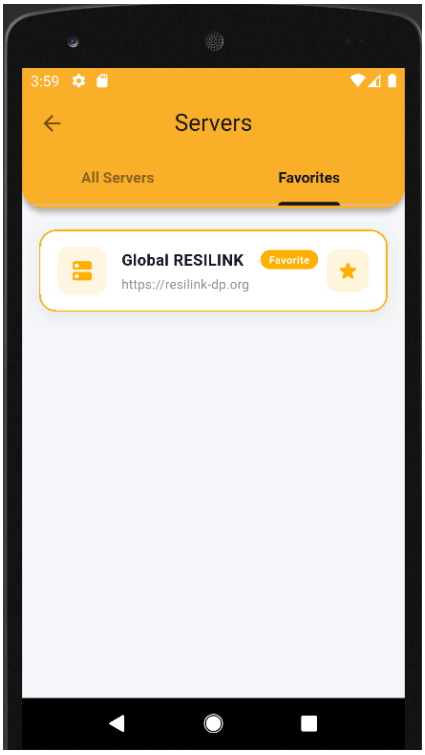
In a presentation given at the First Symposium of InovFarmer.MED, organized by INOVFARMER in collaboration with MED-LINKS, FAIRCHAIN and RESILINK at INRAE in Avignon, France, the importance of an open API was highlighted as illustrated by the following 2 slides.

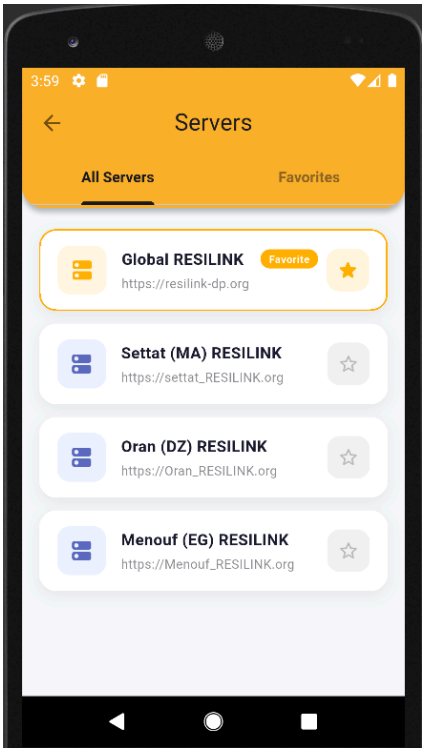


This open API will maximize re-utilisation and facilitate the integration of new platforms as illustrated by the following diagram where third-party applications can be developed using the RESILINK open API, thus enabling their resources to be discovered by other applications of the RESILINK ecosystem. This is the so-called “platform-of-platforms” approach that is promoted by RESILINK. From an implementation perspective, this “platform-of-platforms” approach will be deployed using Multi-Server Federation functionality.

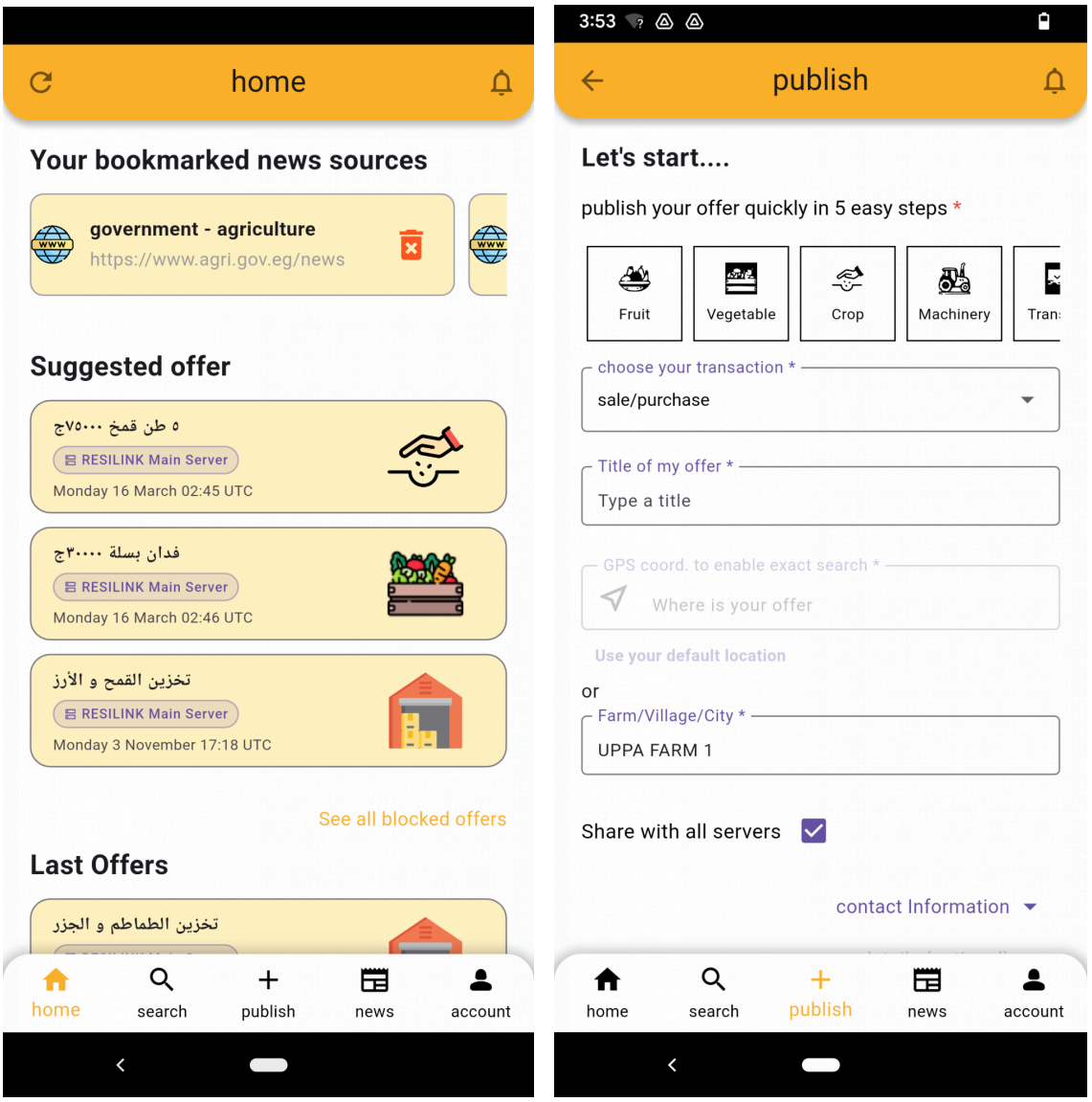


Although the RESILINK Mobile Application will be discussed in more detail in D2.4b “Final report on test and validation of the full RESILINK framework”, below are the Multi-Server Federation functionality integrated into the mobile application.





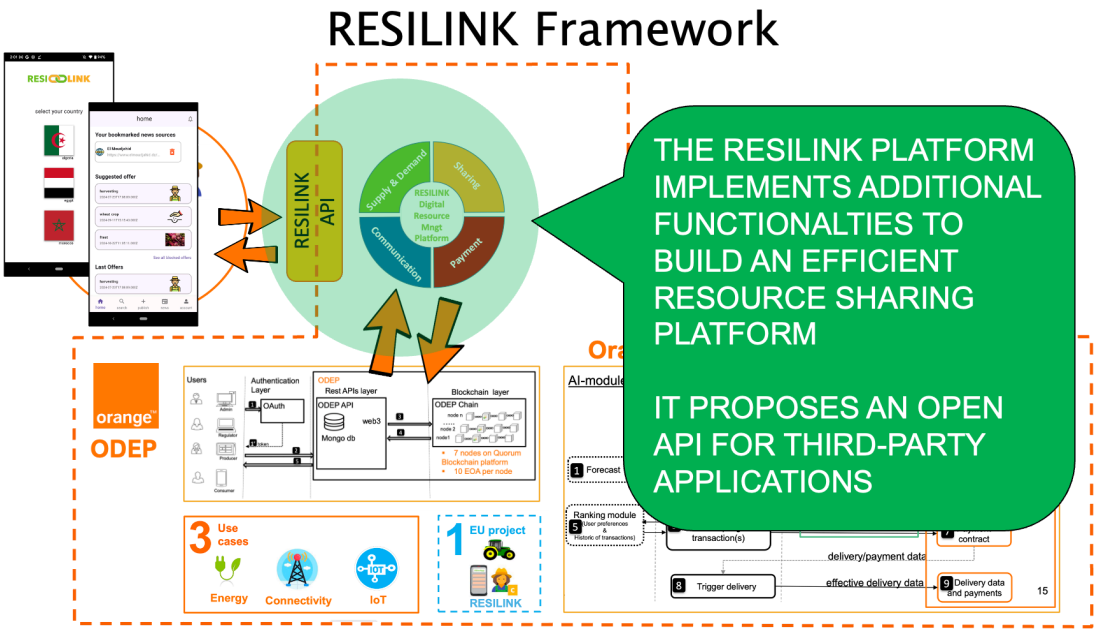
Offers are then tagged with the server they come from and when publishing an offer, one can select the offer to be shared on other servers.



2. THE RESILINK'S OPEN API

2.1. Overview

The so-called RESILINK platform is implemented as a back-end server to the RESILINK mobile application as illustrated in the figure below. We will refer to it as "RESILINK Back-end Server" in this document.

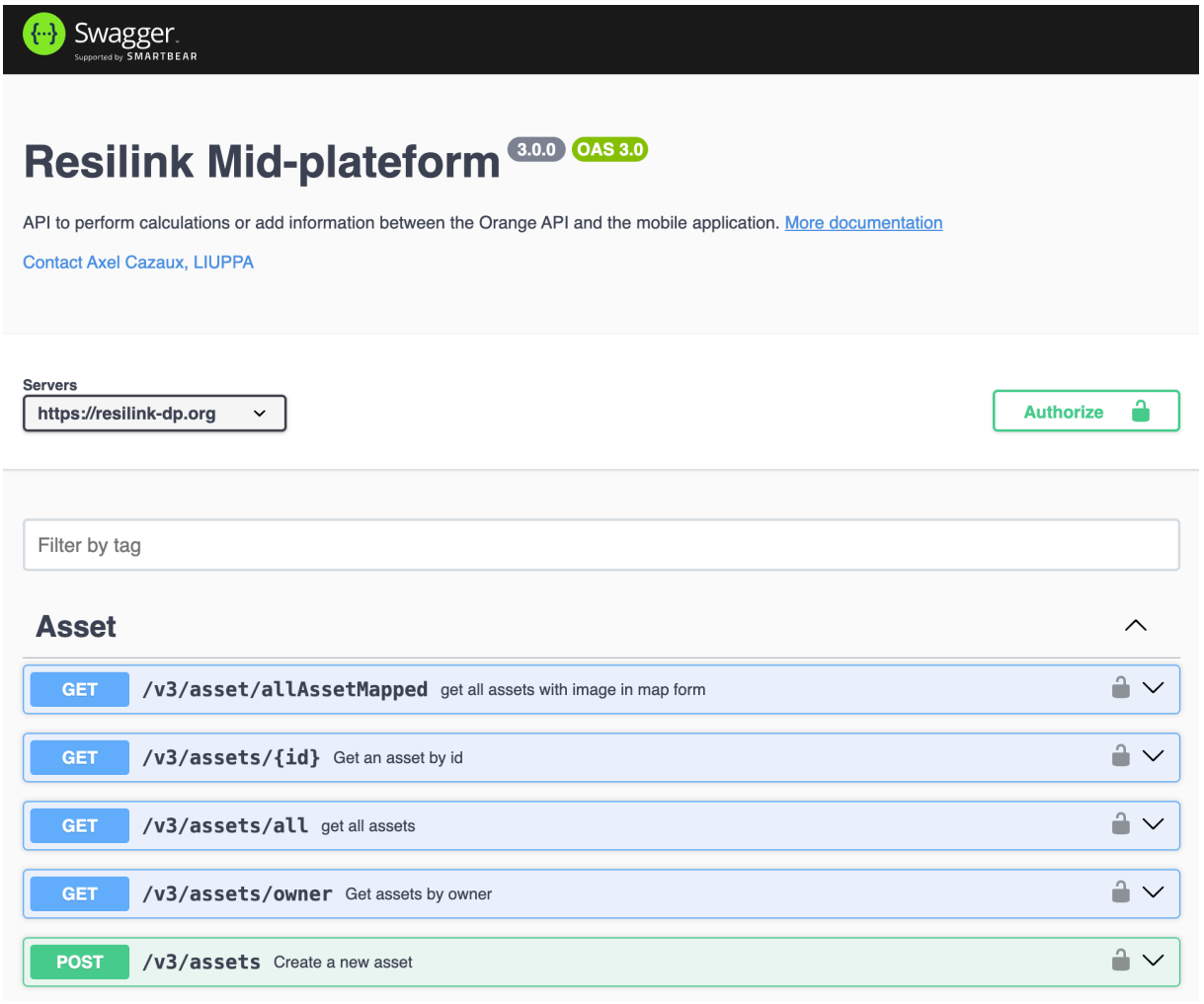


The API v3 has over 80 endpoints across 13 route groups. The following table summarizes the route groups and their scope.

Route Group	Endpoints	Scope
User	10	Account creation, authentication, profile
Prosumer	15	Prosumer profiles, blocked offers, bookmarks
Asset	7	Asset CRUD, image upload/storage
AssetType	6	Asset type definitions
Offer	12	Local & federated offers, filtering
News	5	Community news and announcements
Rating	6	User ratings and feedback
Contract (stub)	4	Preserved for ODEP compatibility
Request (stub)	2	Preserved for ODEP compatibility
Regulator (stub)	2	Preserved for ODEP compatibility
RecommendationStats	5	Personal & global recommendations
RegisteredServers	5	Federation server registry
FavoriteServers	5	User favorite server management

2.2. List of API functions by categories

This section lists all available endpoints exposed by the RESILINK REST API, organised by entity. Each table describes the HTTP method, the route, a brief description of the operation, and the authentication requirement. All endpoints are also accessible interactively via the Swagger UI at https://resilink-dp.org/v3/api-docs/.



User endpoints:

Method	Endpoint	Description
POST	/v3/users/auth/sign_in/	Retrieve user data and access token (login)
POST	/v3/users/	Create a new user
GET	/v3/users/	Return the list of all users
GET	/v3/users/:userId	Find a user by ID
GET	/v3/users/getUserByEmail/:userEmail	Get a user by email
GET	/v3/users/getUserByUserName/:userName	Get a user by username
PUT	/v3/users/password/	Change the password of the authenticated user
PUT	/v3/users/:userId	Update an existing user
DELETE	/v3/users/:userId/	Delete a user
DELETE	/v3/users/allData/:userName/	Delete a user, all their data, and logs
DELETE	/v3/users/logs/:userName/	Delete a user's logs only

Prosumer endpoints:

Method	Endpoint	Description
POST	/v3/prosumers/new/	Create a new user and their prosumer profile simultaneously
POST	/v3/prosumers/	Create a new prosumer profile (user must already exist)
GET	/v3/prosumers/all	Get all prosumers
GET	/v3/prosumers/:id/	Get a prosumer by ID
PUT	/v3/prosumers/:prosumerId/	Update an existing user and their prosumer data
PATCH	/v3/prosumers/:id/balance	Credit a prosumer's balance
PATCH	/v3/prosumers/:id/activityDomain	Update a prosumer's activity domain
PATCH	/v3/prosumers/:id/sharingAccount	Credit a prosumer's sharing account
PUT	/v3/prosumers/:id/addBookmark	Add an ID to the prosumer's bookmark list
POST	/v3/prosumers/:id/blocked-offers/server	Block an offer from a specific server
GET	/v3/prosumers/:id/blocked-offers/server	Get blocked offers for a specific server

GET	/v3/prosumers/:id/blocked-offers/all	Get all blocked offers across all servers
DELETE	/v3/prosumers/:id/blocked-offers/server/:offerId	Unblock an offer from a specific server
DELETE	/v3/prosumers/delBookmark/id/	Remove an ID from the bookmark list
DELETE	/v3/prosumers/:id	Delete a prosumer

AssetType endpoints:

Method	Endpoint	Description
POST	/v3/assetTypes/	Create a new asset type
GET	/v3/assetTypes/all	Get all asset types
GET	/v3/assetTypes/all/mapFormat	Get all asset types in map format (key/value pairs)
GET	/v3/assetTypes/:id/	Get an asset type by ID
PUT	/v3/assetTypes/:id/	Update asset type attributes
DELETE	/v3/assetTypes/:id/	Delete an asset type

Asset endpoints:

Method	Endpoint	Description
POST	/v3/assets/	Create a new asset
POST	/v3/assets/img/	Register an image for an asset
GET	/v3/assets/all	Get all assets
GET	/v3/assets/owner	Get assets belonging to the authenticated owner
GET	/v3/assets/:id/	Get an asset by ID
GET	/v3/asset/allAssetMapped/	Get all assets in map form
PUT	/v3/assets/:id/	Update an asset's attributes
DELETE	/v3/assets/:id/	Delete an asset
DELETE	/v3/assets/img/:id/	Delete the images associated with an asset

Offer endpoints:

Method	Endpoint	Description
POST	/v3/offers/	Create a new offer
POST	/v3/offers/createOfferAsset/	Create a new offer and its asset
POST	/v3/offers/byIds	Retrieve specific offers by their IDs
POST	/v3/offers/local/all/filtered	Get filtered offers from the local server only
POST	/v3/offers/federated/all/filtered	Get filtered offers from local and favorite external servers
GET	/v3/offers/local/all	Get all valid offers from the local server only (no federation)
GET	/v3/offers/federated/all	Get federated valid offers from local and user's favorite servers
GET	/v3/offers/suggested	Get valid and suggested offers for the authenticated user
GET	/v3/offers/LimitedOffer	Get a limited range of valid offers
GET	/v3/offers/owner/blockedOffer/local/	Get blocked offers from the local server only
GET	/v3/offers/owner/blockedOffer/federated/	Get blocked offers from local and favorite external servers
GET	/v3/offers/owner/	Get all offers from the authenticated prosumer
GET	/v3/offers/owner/purchase/	Get all offers purchased by the authenticated prosumer
GET	/v3/offers/all/	Get all offers (admin view)
GET	/v3/offers/:id/	Get an offer by ID
PUT	/v3/offers/:id/	Update an offer's attributes
PUT	/v3/offers/:id/updateOfferAsset/	Update both an offer's and its asset's attributes
DELETE	/v3/offers/:id/	Delete an offer

Contract endpoints (ODEP version only):

Method	Endpoint	Description
POST	/v3/contracts/	Create a new contract
GET	/v3/contracts/all	Get all contracts (admin view)
GET	/v3/contracts/owner/ongoing/	Get ongoing contracts of the authenticated owner
GET	/v3/contracts/owner/	Get all contracts by owner
GET	/v3/contracts/	Get a contract by ID
PATCH	/v3/contracts/measurableByQuantityContract/:id/	Update state of a purchase contract for a material asset and adjust payment
PATCH	/v3/contracts/measurableByTimeContract/:id/	Update state of a rent contract for a material asset and adjust payment
PATCH	/v3/contracts/cancelContract/:id/	Cancel a contract before or after its execution

News endpoints:

Method	Endpoint	Description
POST	/v3/news/	Create a news source
POST	/v3/news/:id/	Create a personal news source and add it to the creator's favorites
PUT	/v3/news/:id/	Update a news source
GET	/v3/news/all	Get all news
GET	/v3/news/country	Get all news from a country
GET	/v3/news/ids	Get news from an ID list
GET	/v3/news/owner/:id/	Get all news from a specific prosumer
GET	/v3/news/countryOwner	Get all news within a country excluding the user's subscribed news
DELETE	/v3/news/:id/	Delete a news item

Rating endpoints:

Method	Endpoint	Description
POST	/v3/rating/	Create a rating
PUT	/v3/rating/:userId/	Update a rating by user ID
GET	/v3/rating/all	Get all ratings
GET	/v3/rating/average	Get the average rating
GET	/v3/rating/:userId	Get the rating of a specific user
DELETE	/v3/rating/:userId/	Delete a rating

Registeredservers endpoints (without ODEP version only):

Method	Endpoint	Description
POST	/v3/registeredservers/	Register a new server
POST	/v3/registeredservers/global	Register a new server in the main/central server
GET	/v3/registeredservers/	Retrieve all locally registered servers
GET	/v3/registeredservers/global	Retrieve all servers registered in the main server
GET	/v3/registeredservers/:serverName/	Retrieve a registered server by name
PUT	/v3/registeredservers/:serverName/	Update a registered server by name
DELETE	/v3/registeredservers/:serverName/	Delete a registered server

FavoriteServers endpoints (without ODEP version only):

Method	Endpoint	Description
POST	/v3/favoriteServers/	Create a favorite server list for a user
POST	/v3/favoriteServers/:username/add	Add a server to a user's favorite list
GET	/v3/favoriteServers/	Get all favorite server lists
GET	/v3/favoriteServers/:username	Get a specific user's favorite server list
PUT	/v3/favoriteServers/:username	Replace the entire favorite server list for a user
DELETE	/v3/favoriteServers/:username/remove	Remove a server from a user's favorite list
DELETE	/v3/favoriteServers/:username	Delete a user's entire favorite server list

Regulator endpoints (ODEP version only):

Method	Endpoint	Description
POST	/v3/ODEP/regulators	Create a new regulator
GET	/v3/ODEP/regulators/all	Get all regulators
GET	/v3/ODEP/regulators/:id	Get a regulator by ID
PATCH	/v3/ODEP/regulators/:id	Update the list of asset types a regulator is accountable for
DELETE	/v3/ODEP/regulators/:id	Delete a regulator

Request endpoints (ODEP version only):

Method	Endpoint	Description
POST	/v3/ODEP/requests/	Post a new request
GET	/v3/ODEP/requests/all	Get all requests
GET	/v3/ODEP/requests/:id/	Get a request by ID
PUT	/v3/ODEP/requests/:id/	Update a request's attributes
DELETE	/v3/ODEP/requests/:id/	Delete a request

Recommendationstats endpoints (without ODEP version only):

Method	Endpoint	Description
POST	/v3/recommendationstats/	Create a new recommendation stats document
GET	/v3/recommendationstats/	Retrieve all recommendation stats documents
GET	/v3/recommendationstats/:name/	Retrieve a recommendation stats document by name
PUT	/v3/recommendationstats/:name/	Update a recommendation stats document by name
PATCH	/v3/recommendationstats/:name/increment/:assetType	Increment the count of a specific asset type in a recommendation stats document
DELETE	/v3/recommendationstats/:name/	Delete a recommendation stats document by name

3. Installing and Deploying instance of the RESILINK platform

The detailed description of the installation procedures for all RESILINK platform components are presented in:

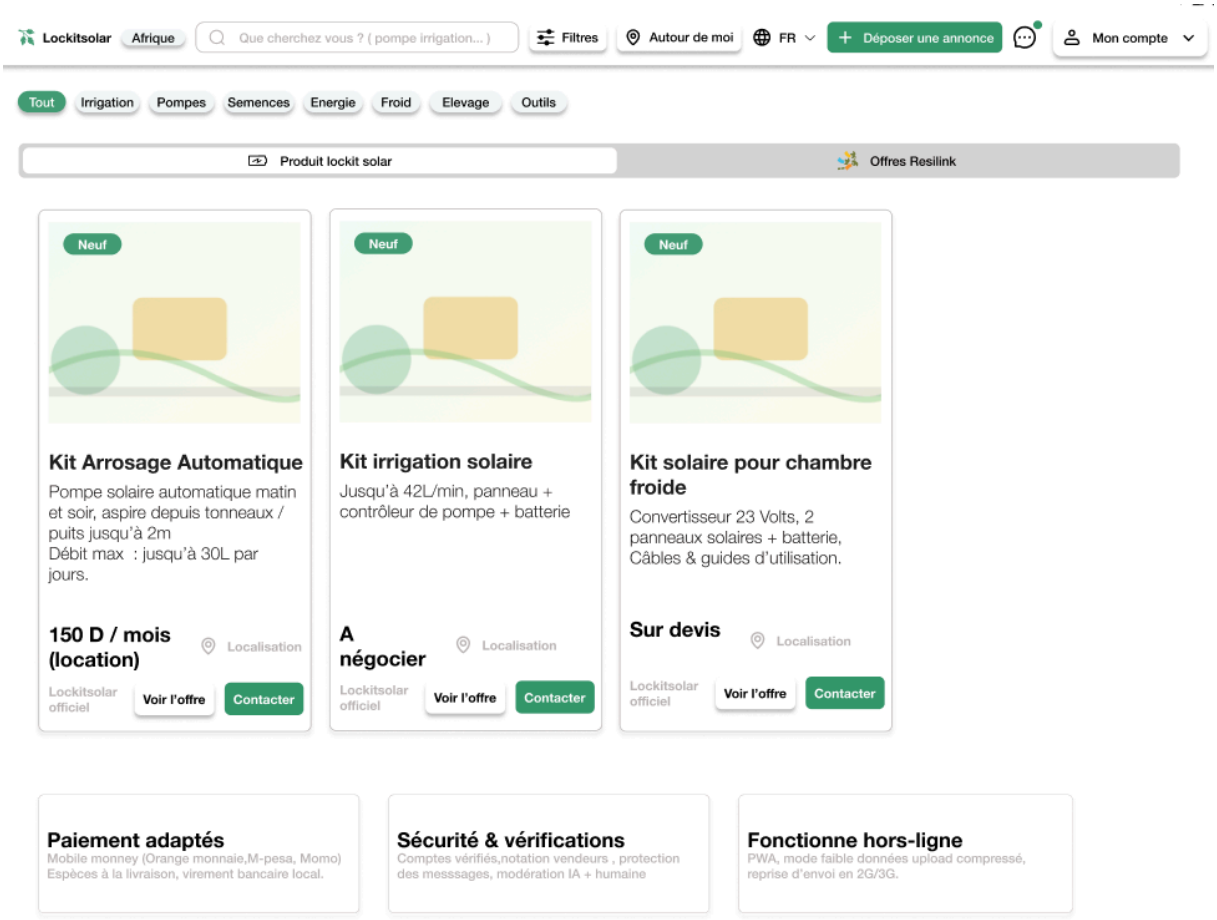
D2.2c - Annex “RESILINK resource sharing digital platform – v3”
https://resilink.eu/wp-content/uploads/2026/04/D2.2c-annex.pdf

4. Use Case: LocKitSolar by Orange

4.1. LocKitSolar

LocKitSolar is a demonstration application showcasing how to leverage the RESILINK API for building a solar product marketplace. It enables users to authenticate, browse a curated catalog of photovoltaic solutions, and complete purchases seamlessly. Built with Vue.js, the platform harnesses the RESILINK API to orchestrate data retrieval and core business operations.

LocKitSolar Example Application



More concretely, LocKitSolar streamlines the rental of solar-powered agricultural equipment for farmers. Designed with accessibility at its core, the intuitive interface empowers farmers to quickly locate and rent equipment tailored to their specific needs. On the operational side, the platform provides LocKitSolar with a secure, efficient system for inventory management and rental tracking. Constructed with modern web technologies, it ensures a responsive experience accessible from any device.

Beyond this use case, LocKitSolar fulfills a pedagogical mission: serving as a blueprint for developers seeking to build their own services on the RESILINK API. Its modular, extensible architecture facilitates the addition of future features. The platform also functions as a

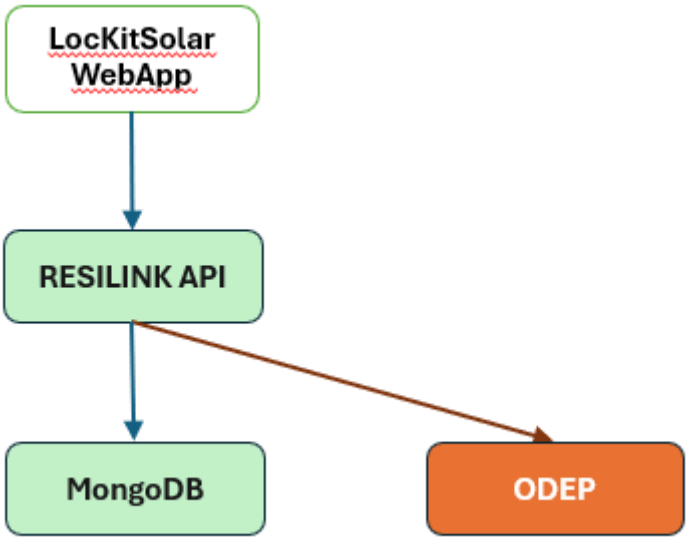
reference implementation, documenting best practices for user authentication, data retrieval, and API operations.

Deliberately streamlined, this demonstration focuses exclusively on core functionality. It deliberately sidesteps unnecessary complexity—user registration, payment processing, order management—in favor of pedagogical clarity. By emphasizing authentication, data retrieval, and fundamental operations (product browsing, basic purchases), it provides developers with a solid, immediately comprehensible foundation. The codebase, well-structured and thoroughly documented, adheres to web development standards, facilitating maintenance and future enhancements.

LocKitSolar thus stands as an invaluable resource: a pragmatic starting point for anyone looking to build applications on the RESILINK API.

4.2. The architecture

LocKitSolar's architecture comprises two complementary layers: a Vue.js 3 frontend and the RESILINK backend API. The frontend delivers an intuitive interface enabling farmers to browse equipment and manage rentals, while the backend orchestrates authentication, data persistence, and business logic. This separation of concerns ensures clean code organization and independent scalability.



The frontend prioritizes user experience—responsive design, intuitive navigation, and accessibility—while the backend emphasizes security and performance, protecting sensitive data and processing requests efficiently. Together, they form a robust, maintainable system capable of handling growth and future feature additions.

For this demonstration, we employ the RESILINK version operating independently of ODEP. However, the application's modular architecture ensures seamless adaptation should ODEP integration become necessary. Rather than hardcoding dependencies, the codebase

abstracts API interactions, allowing developers to swap RESILINK versions or integrate alternative backends with minimal refactoring. This flexibility is documented through clear examples and integration guides, enabling developers to extend the platform according to their specific requirements.

Built on modern web technologies and adhering to industry best practices, LocKitSolar delivers reliability, maintainability, and consistent performance under load. It serves as both a functional marketplace and a comprehensive reference implementation—demonstrating not merely how to use the RESILINK API, but how to architect scalable, adaptable applications that evolve with changing needs.

LocKitSolar's Vue.js 3 architecture follows a modular, layered structure:

Entry Point & Core Setup

- `src/main.ts` bootstraps the application, initializing Pinia for state management, configuring Vue Router for navigation, and mounting the root component.

Root Layout

- `src/App.vue` serves as the application shell, rendering the navigation header, page content via `RouterView`, and initializing authentication state on load.

Components & Navigation

- `src/components/` houses reusable UI elements:
  - `TheNavigation.vue` — top-level navigation bar with role-based admin access
  - `TheWelcome.vue`, `WelcomeItem.vue` — welcome screen components
  - Icon components — shared iconography

Pages & Views

- `src/views/` contains full-page screens:
  - `LoginView.vue` — user authentication
  - `HomeView.vue` — equipment catalog and rental requests
  - `AdminView.vue` — asset, offer, and contract management
  - `AboutView.vue` — application information

State Management

- `src/stores/authStore.ts` (Pinia) manages:
  - User authentication state
  - Role detection (admin vs. customer)
  - Session persistence via `localStorage`
  - Login and logout operations

API Integration

- `src/services/apiService.ts` — Axios-based API client abstracting:
  - Authentication headers
  - RESILINK endpoints for offers, assets, contracts, and rental requests

Styling

- `src/assets/ contains global stylesheets (base.css, main.css) applied across the application.`

4.3. A service developed using IA

We leveraged an AI agent to accelerate development, providing specifications and receiving high-quality, maintainable code. The agent successfully generated the frontend architecture—Vue.js components, API service layer, and authentication logic—following web development best practices and producing well-structured, readable code.

However, **code generation is not a turnkey process**. The AI agent produced a solid foundation, but developers encountered real issues requiring debugging and refinement. The agent cannot anticipate all edge cases or perfectly interpret nuanced requirements. This is not a limitation unique to AI; it reflects the gap between specification and implementation that exists in all software development.

The practical value lies in understanding AI's role: a productivity accelerator, not a replacement for human expertise. The agent excels at generating boilerplate, scaffolding components, and implementing standard patterns. Developers remain essential for code review, testing, architectural decisions, and adapting outputs to specific contexts.

The Specification as Living Documentation

The specifications provided serve multiple purposes beyond code generation:

- **Reference**: Developers can quickly understand application requirements and feature scope
- **Roadmap**: Clear guidance for planned enhancements and architectural evolution
- **Living document**: Designed for iterative updates as requirements and user needs evolve

This approach—combining AI-assisted development with comprehensive specification documentation—demonstrates a pragmatic methodology: leverage automation where it adds value while maintaining human oversight and expertise at critical decision points.

SPECIFICATIONS.md

```
# LocKitSolar V2 - Solar Agricultural Equipment Rental Application

## 📋 Project Overview

LocKitSolar is a modern web application developed with Vue.js 3 that allows farmers to rent solar-powered agricultural equipment via the RESILINK platform.

The project is developed with Vue.js.

The development server runs in Docker.

The source files are mounted directly from the host machine into the container.

The project was initialized from Docker using `npm create vue@latest`.

## API Configuration

- Authentication: http://resilink_render_server:8080/v1/users/auth/sign_in
- API Base URL: http://resilink_render_server:8080/v1/
- Main account ID: "lockitsolar"

The first page is the home page.

## Login

To use LocKitSolar, users must authenticate with RESILINK to obtain an API access token.

To do this, send a POST request to the Authentication URL with the following body:

``` json
{
        userName: username.value,
        password: password.value
}
```

The response must:

* return HTTP 200
* contain `data.userName` with the value of the Main account ID
```

```
* contain a `data.accessToken` field, which is stored and used in subsequent API calls


If login is successful, the user is redirected to Home; otherwise, an error message is displayed.

The Login page must use the same visual style as the Home page.


On the Login page, users can log in with any username and password. There is also a button, "Login as lockitsolar admin", which automatically logs in with the lockitsolar admin credentials stored in the `.env` file, and another button, "Login as lockitsolar customer", which automatically logs in with the lockitsolar customer credentials stored in the `.env` file.


## HOME

The mockup ![HOME](./design/home-produit-lockitsolar.png)


As long as the user is not authenticated, the offers list is not displayed.

To retrieve the lockitsolar offers, call `GET {API Base URL}/ODEP/offers/all` with the Authorization header, filter offers where `offerer` equals `lockitsolar`, then call `GET /assets/{id}` for each offer (`id = offer.assetId`).

The response is a list of JSON objects representing products.


``` json
{
    "id": "40",
    "name": "Kit solaire Irrigation",
     "description": "Kit solaire de 50w pour l'irrigation composé d'une pompe d'un débit de 5l/h",
    "assetType": "AgriculturalMachine",
    "owner": "lockitsolar",
    "multiAccess": false,
    "totalQuantity": 12,
    "remainingQuantity": 8,
    "regulatedId": "",
    "specificAttributes": [
      {
        "attributeName": "type",
        "value": "kit solaire"
      },
```
```

```
      {
        "attributeName": "moteur",
        "value": "0"
      },
      {
        "attributeName": "energy",
        "value": "solar"
      },
      {
        "attributeName": "location",
        "value": ""
      }
    ]
}
```

The product category is the value of the parameter where `attributeName` equals `type`.

The "+ Post an ad" button does not appear.

When the user is `lockitsolar adminAccount.username`, this button is replaced by Administration.

Rent a product

When the user clicks a product, a modal window appears with asset and offer details, and the user has two buttons: "Cancel" and "Rent".

* "Cancel": closes the window
* "Rent": creates a contract

To create a contract, first create a request by calling `POST /ODEP/requests` with:

``` json
{
  "requestor": the username of the logged user,
  "beginTimeSlot": "2026-02-09T13:54:48.399Z",
  "endTimeSlot": "2026-02-09T13:54:48.399Z",
  "validityLimit": "2026-02-09T13:54:48.399Z",
  "transactionType": "rent",
  "offerIds": [
```

```
    the offer id
  ]
}
```
The response will look like:
``` json
{
"requestId": integer,
"message": "string",
"availableOffersCount": integer,
"availableOffersIds": [
integer
]
}
```
Then create the contract with `POST /contracts`:
``` json
{
"offerId": 0,
"requestId": 0
}
```

Admin

The Admin page follows the same visual style as the Home page.
It allows users to manage assets and offers for the admin account.
It also allows users to view and manage the list of ongoing contracts.

For assets
Allows retrieving the asset list with `GET /assets/owner`.
Allows creating a new asset with `POST /assets` and a body like:
``` json
{
      "name": "un nom",
                "description": "Kit solaire pour une chambre froide",
                "assetType": "agriculturalMachine",
```

```
                "transactionType": "rent",
                "totalQuantity": 20,
                                                                "images"    :
["iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mP8/5+hHgAHg
gJ/PchI7wAAAABJRU5ErkJggg=="],
                "totalQuantity": 22,
                "remainingQuantity": 17,
                "specificAttributes": [
                    {
                        "attributeName": "type",
                        "value": "kit solaire"
                    },
                    {
                        "attributeName": "moteur",
                        "value": "0"
                    },
                    {
                        "attributeName": "energy",
                        "value": "solar"
                    },
                    {
                        "attributeName": "location",
                        "value": ""
                    },
                    {
                        "attributeName": "lks",
                        "value": "0"
                    }
                ]
            }
```

`assetType` is always `AgriculturalMachine`.

`transactionType` is always `rent`.

`attributeName = energy` can take the values: `solar`, ` electric`, ` gasoil`, ` manual` (the leading space before some values is intentional).

`images` is an array containing one element: a Base64-encoded image string selected by the user on their computer.

Allows deleting an asset via `DELETE /assets/{id}`.

```
### For offers


      Allows listing offers via `GET /ODEP/offers/all` and filtering on
offers where `offerer` equals `lockitsolar`.

     Allows creating an offer via `POST /offers` with a body like:
``` json
{
"offerer": "lockitsolar",
"assetId": createdAssetId,
"beginTimeSlot": new Date().toISOString(),
"endTimeSlot": "2026-03-25T15:46:02.675Z",
"validityLimit": "2023-06-25T15:46:02.675Z",
"offeredQuantity": 10,
"price": 2,
"deposit": 10,
"cancellationFee": 5,
"rentInformation": {
    "delayMargin": 10,
    "lateRestitutionPenality": 40,
    "deteriorationPenality": 50,
    "nonRestitutionPenality": 100
    }
}
```
`createAssetId` must be one of the existing `assetId` values for the
owner, selected from a list of existing asset names.
`assetId` must be passed as a number (the ID of a previously created
asset).

Allows deleting an offer with `DELETE /offers/{id}`.

For Contracts

Allows retrieving the contract list with `GET
/contracts/owner/lockitsolar`.


``` json
[
  {
```
```

```
    "idContract": 0,
    "offer": "string",
    "Request": "string",
    "asset": "string",
    "state": "string",
    "quantityToDeliver": 0,
    "deliveredQuantity": 0,
    "consumedQuantity": 0,
    "creationDate": "2026-02-09T13:45:35.945Z",
    "transactionType": "string",
    "offerer": "string",
    "requester": "string",
    "assetType": "string",
    "price": 0,
    "deposit": 0,
    "cancellationFee": 0,
    "beginTimeSlot": "2026-02-09T13:45:35.945Z",
    "endTimeSlot": "2026-02-09T13:45:35.945Z",
    "effectiveBeginTimeSlot": "2026-02-09T13:45:35.945Z",
    "effectiveEndTimeSlot": "2026-02-09T13:45:35.945Z",
    "rentInformation": {
      "delayMargin": 0,
      "lateRestitutionPenalty": 0,
      "deteriorationPenalty": 0,
      "nonRestitutionPenalty": 0
    }
  }
]
```

Display the result as a table with `requester`, `beginTimeSlot`, `endTimeSlot`, `asset`, and `state`.

When the user clicks a table row, a modal window appears with details of the corresponding asset, offer, and `assetType`.

Acronyms List

Acronym	Explanation
API	Application Programming Interface
HTTP	Hyper Text Transport Protocol
ODEP	Orange Decentralized Exchange Place
UI	User Interface
URL	Universal Resource Locator

Project Co-ordinator Contact

Pr. Congduc Pham

University of Pau

Avenue de l'Université

64000 PAU

FRANCE

Email: Congduc.Pham@univ-pau.fr

ACKNOWLEDGEMENT

This document has been produced in the context of the PRIMA RESILINK project. The RESILINK project consortium would like to acknowledge that the research leading to these results has received funding from the European Union through the PRIMA program.