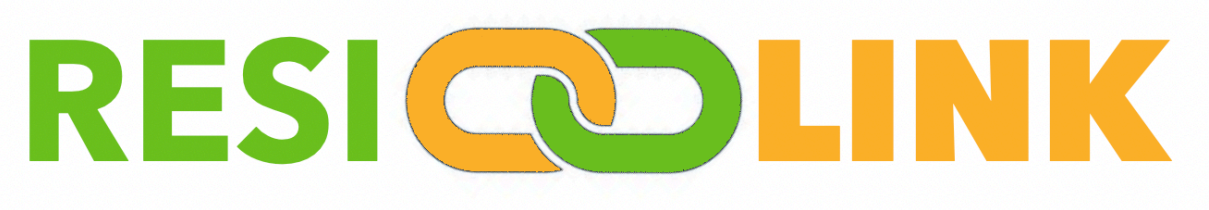


Increasing Resilience of Smallholders with Multi-Platforms Linking Localized Resource Sharing

Deliverable D2.2c

RESILINK resource sharing digital platform – v3

Responsible Editor:	ORANGE
Contributors:	UPPA
Document Reference:	RESILINK D2.2c
Distribution:	Public
Version:	1.1
Date:	Apr. 2026

CONTRIBUTORS TABLE

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

DOCUMENT REVISION HISTORY

Version	Date	Changes
V1.1	April 21st, 2026	PUBLIC RELEASE
V1.0	April 17th, 2026	FIRST DRAFT VERSION FOR INTERNAL APPROVAL
V0.1	April 10th, 2026	FIRST RELEASE FOR REVIEW

EXECUTIVE SUMMARY

Deliverable D2.2c describes the RESILINK platform v3 that runs behind the RESILINK mobile application to implement all the functionalities for sharing resources. It also describes the open API approach to enable third-party applications to use the RESILINK platform to build their own exchange platforms, implementing the “platform-of-platforms” that can be viewed technically as a multi-server federation system.

There is a companion document named D2.2c Annex that provides complete, step-by-step instructions for installing, configuring, and deploying the RESILINK platform, covering the 3 main components:

- The RESILINK Backend Server (Node.js + MongoDB)
- The RESILINK Mobile Application (Flutter/Android), including code configuration, APK/AAB build, signing, and release
- The mobile application public distribution via the Google Play Store

Table of Contents

1. INTRODUCTION

RESILINK develops a distributed digital resource management platform for real-time exchange of information on territorial resources and supplies & demands; connecting smallholders to new supply, sharing opportunities and distribution channels. In addition, RESILINK will incrementally use cutting-edge digital technologies to add intelligence in the agri-food value chain to provide simple application interfaces adapted to smallholders.

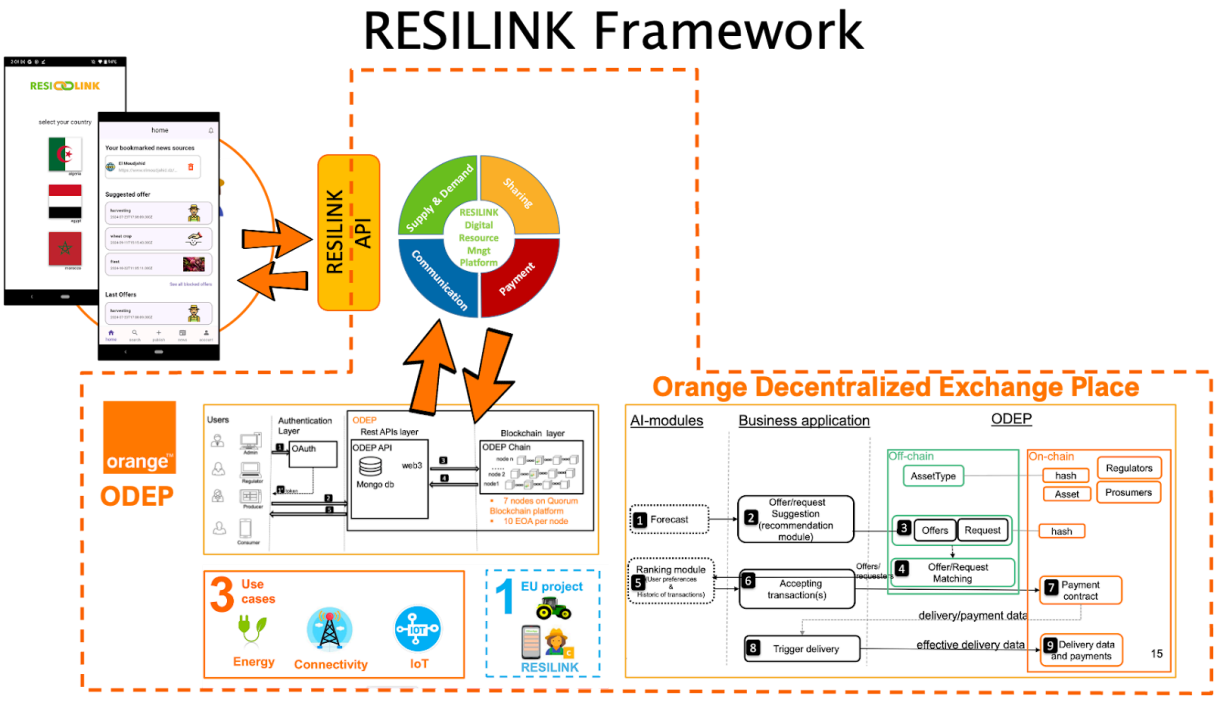
The lowest layer of the RESILINK digital resource management consists of the **Orange Decentralized Exchange Place (ODEP)** platform initially developed by Orange for energy and telecommunication ecosystems.

In addition to ODEP, and to maximize RESILINK's usage by the end users, RESILINK will develop a mobile application, referred to as **RESILINK mobile app**, that will be the main interface to simply, quickly and intuitively manage resources. An intermediate platform, referred to as the **RESILINK platform**, will serve as the back-end server for the RESILINK mobile application.

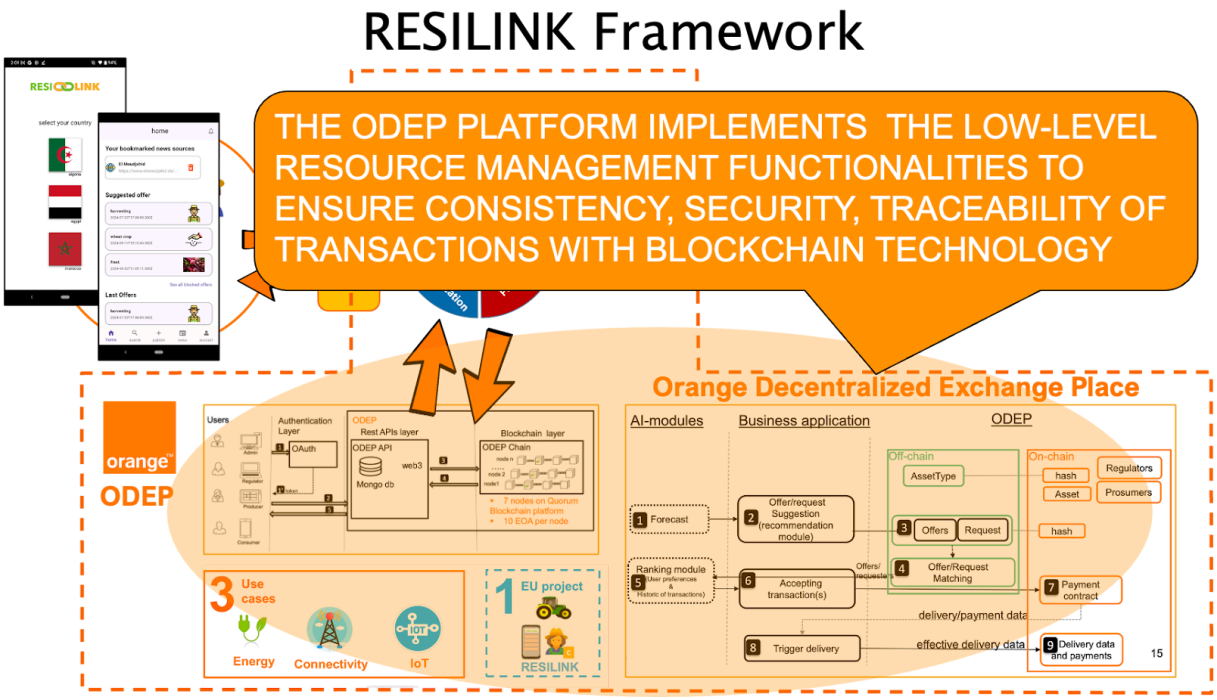
Due to ODEP's development constraints and internal architecture, it is difficult to add new functionalities in the time frame of the Living-Labs. Therefore, the RESILINK platform can run is 2 versions: (1) with ODEP at the lowest layer to implement smart contracts based on Blockchain technologies and (2) without ODEP and without smart contracts, but with many additional features that are useful for end-users that participate in the Living-Labs program. This 2 branches approach has been implemented since RESILINK platform – v2.

2. QUICK REVIEW OF THE RESILINK FRAMEWORK

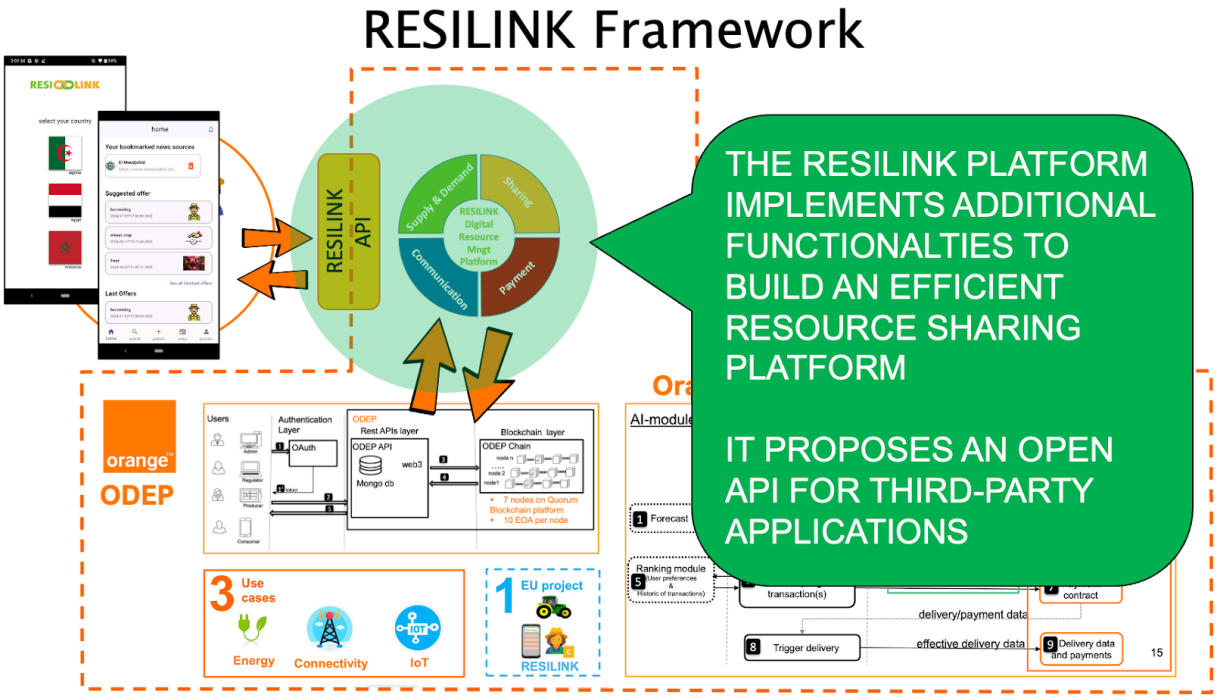
Big picture



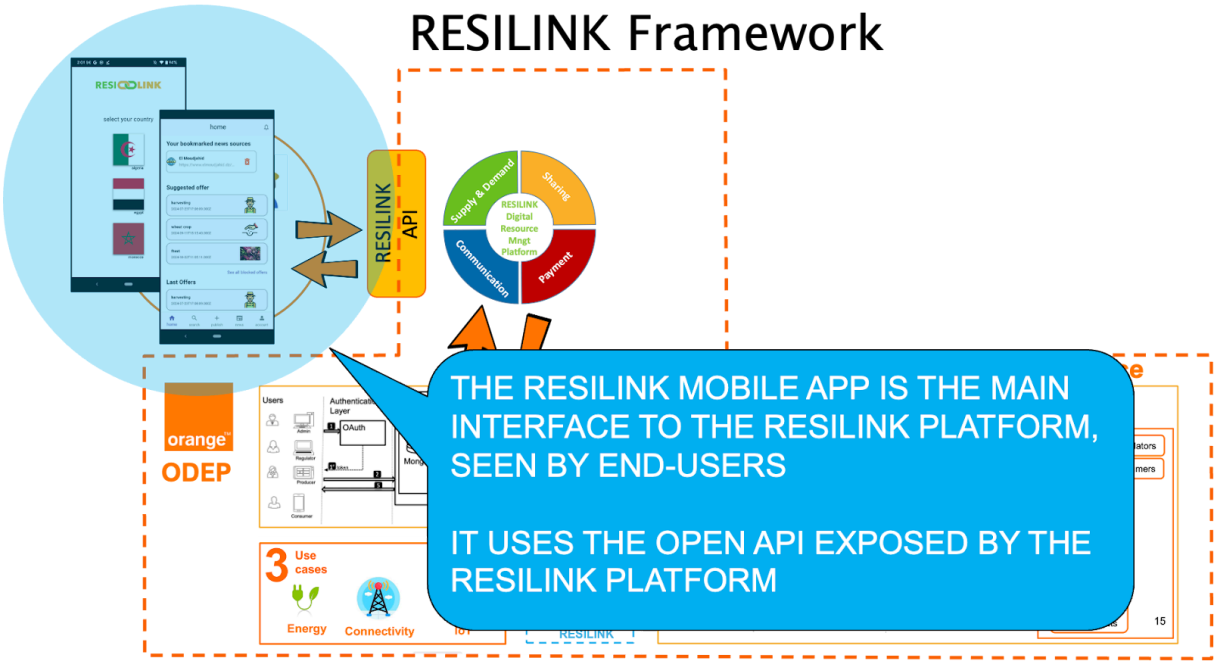
ODEP layer



RESILINK platform



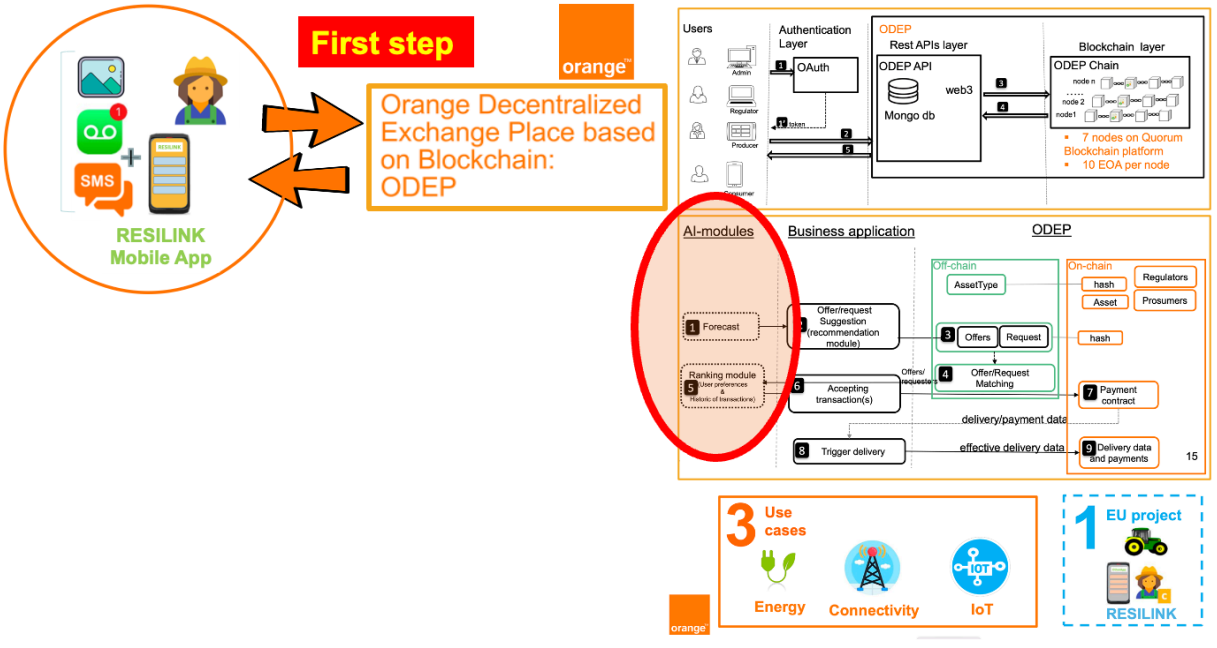
RESILINK mobile application



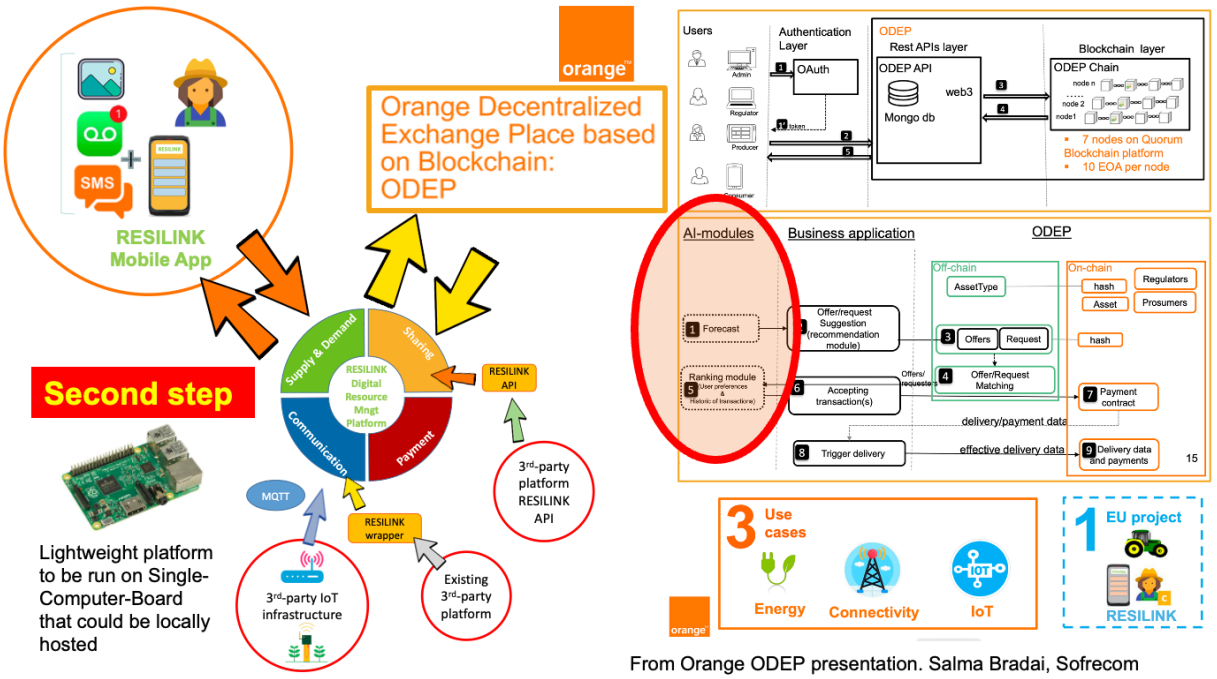
3. Evolution of the RESILINK platform

3.1. RESILINK platform – v1

RESILINK implements a dedicated intermediate RESILINK platform to serve as a front-end to the ODEP platform. The illustration below depicts this concept: the first step was the proof-of-concept demonstrator at the very beginning of the project – it was v0.



From Orange ODEP presentation. Salma Bradai, Sofrecom



From Orange ODEP presentation. Salma Bradai, Sofrecom

In the second step (RESILINK platform v1) The RESILINK platform will function as an API intermediary between the ODEP API and the RESILINK mobile application.

3.2. RESILINK platform – v2

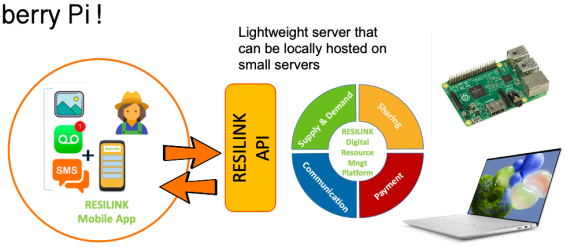
The main motivations behind the RESILINK platform v2 are:

1. Provide a very lightweight server that could be deployed out-of-the-box on laptops or even single board computers such as a Raspberry Pi
2. This lightweight server should be able to manage most of RESILINK platform core functionalities: news feeds, publication of offers, search for offers and contact users
3. Enable the deployment of several lightweight RESILINK servers, where each server could have a geographical scope to efficiently manage crisis in a very local manner
4. Incremental deployment of servers could then be also enabled to provide a very level of flexibility
5. Enable a full platform-of-platforms approach where the RESILINK server can be deployed, possibly adapted and used, through the proposed open API, for other applications, thus stimulating local innovation

These motivations were actually defined from the very beginning of the project as illustrated by the following 3 slides that were presented during various dissemination events and workshops.

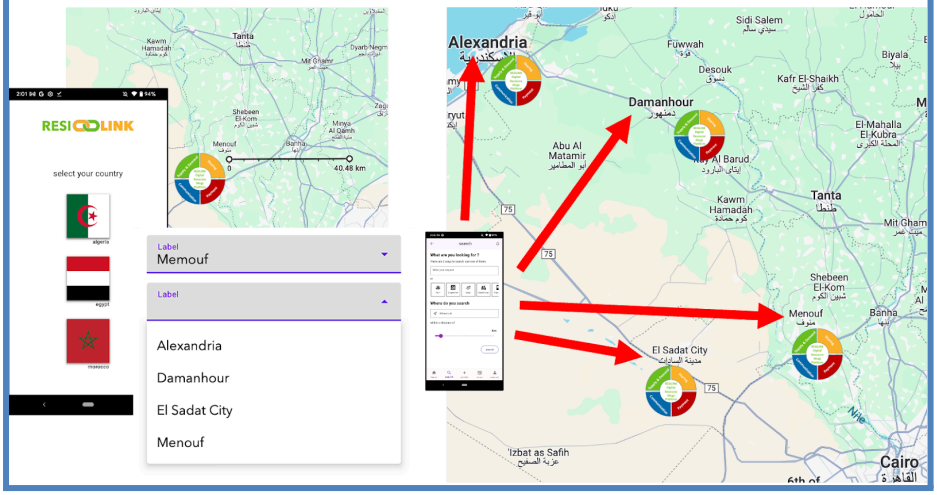
Light-weight servers & fast deployment

- The RESILINK digital platform server can run on a small server, even on a Raspberry Pi !



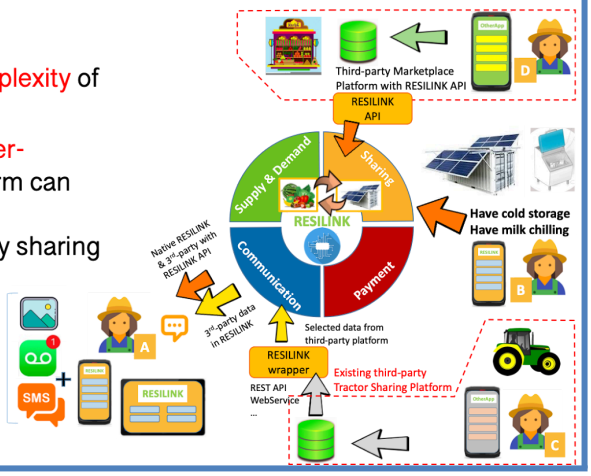
- A RESILINK server can be locally deployed by local city gouvernment agencies, agriculture cooperatives, agriculture services, … in 1 hour!

Incremental deployment



Stimulating local innovation?

- The open API will reduce development time & complexity of new platforms
- All platforms can fully inter-operate and a new platform can benefit from all the other platforms' communities by sharing users & contents
- RESILINK develops a consistent API to enable fast development and deployment platforms



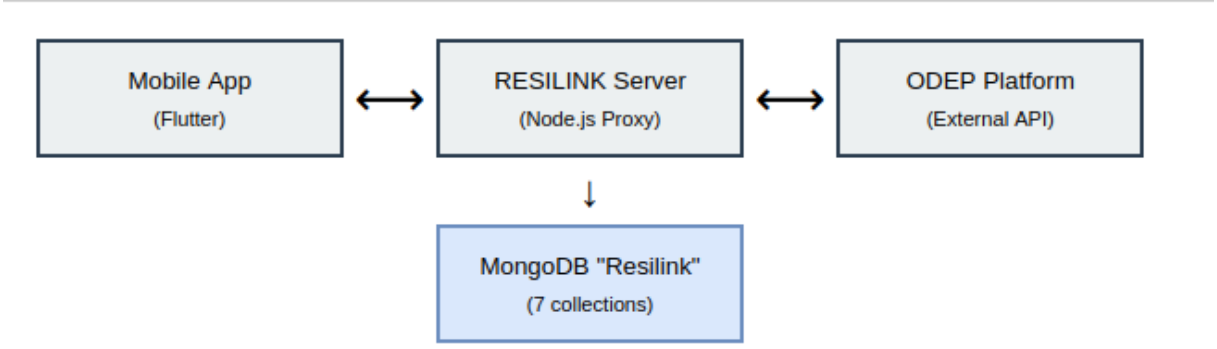
4. RESILINK PLATFORM – TOWARDS V3

4.1. Architecture

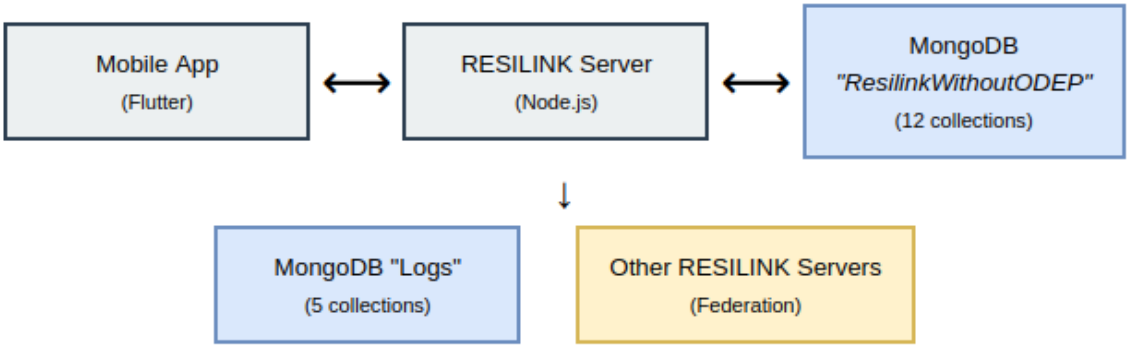
Since v2, the RESILINK platform has been available in two variants. The first variant (main branch on the GitHub) acts as a proxy to the ODEP platform: incoming requests from the mobile application are forwarded to the corresponding ODEP endpoints for processing, and responses are relayed back after optional enrichment. The second variant (feature/without-odep branch on the GitHub) was designed to replicate the core functionalities of ODEP locally, without relying on the external infrastructure and without supporting contract, request, or regulator features.

The following diagram illustrates the ODEP-integrated variant:

ODEP-Integrated Variant (main branch)



In v3, the autonomous variant has been substantially enriched. While it already handled data operations locally in v2, the v3 version **adds federation, recommendation, advanced security, and other features** described in this section:



Architecture v3 autonome

The server continues to follow the MVC+S (Model-View-Controller + Service) pattern already established in v2, with Routes, Controllers, Services, and Database modules for each

functional domain. The key difference is that the Database layer now communicates directly with the MongoDB database instead of forwarding requests to ODEP.

Although the ODEP dependency has been removed, the v3 platform remains interface-compatible. The Contract and Request endpoint groups are preserved as stubs, ensuring that the API surface stays consistent across both variants. Therefore, should ODEP integration be needed in the future, the main branch of the repository continues to support it.

4.2. RESILINK platform v3

The RESILINK platform v3 represents a major evolution from the previous version. Since v2, the RESILINK platform has existed in two variants: one that integrates with the ODEP infrastructure for blockchain-backed transaction management, and one that operates autonomously without ODEP. The autonomous variant was designed to replicate the core functionalities of ODEP (asset and offer management, user profiles) while excluding transaction-related features such as contracts, requests, and regulator endpoints. In v3, this autonomous variant has been significantly enhanced with new features -- federation, recommendation engine, advanced security -- making it a fully self-contained middleware that can be deployed independently on lightweight hardware.

The autonomous variant, located in the "feature/without-odep" branch of the repository, is the version currently deployed for the RESILINK living-labs. It retains the same REST API design principles as the ODEP-integrated variant but handles all data operations internally using MongoDB. This section describes the key changes and additions that characterize version 3.

> ℹ **NOTE** The version with ODEP does not include the following new features: recommendation systems and communication within server networks, due to the use of ODEP.

4.2.1. Data Model Evolution

The v3 data model has been significantly expanded compared to v2. The main MongoDB database has been renamed from "Resilink" to "ResilinkWithoutODEP" when using the server without ODEP to prevent any conflict between the two variants when both are installed on the same machine. The number of collections has grown from seven to twelve, reflecting the addition of new features.

The five new collections introduced in v3 are:

- RecommendationStats: records per-user consultation patterns, tracking which types of offers each user views most frequently. This data feeds the recommendation engine described in section 4.3.6.
- GlobalRecommendationStats: aggregates consultation statistics across all users into a single global view, recalculated daily by a scheduled CRON task.

- RegisteredServers: maintains a directory of known RESILINK servers within the federated network, storing each server's name and access URL.
- FavoriteServers: stores each user's preferred list of external servers, enabling personalized cross-server offer aggregation.
- Counters: provides auto-increment sequences for offer, asset, and news identifiers, replacing the identifier generation previously handled by ODEP.

The existing collections (user, prosumer, Asset, AssetType, Offer, News) have also been enhanced. The prosumer collection now includes a "blockedOffers" field structured as a map keyed by server name, enabling multi-server offer blocking.

Sensitive fields such as email addresses and phone numbers are encrypted at rest using AES-256-CBC. User passwords are now hashed using bcrypt. More broadly, existing functions across all modules have been adjusted or rewritten to match the updated data structures, ensuring consistency between the API layer and the new database schema.

4.2.2. Extended API Surface

The v3 API has been substantially expanded, growing from approximately 30 endpoints across 7 route groups to 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

The new route groups (RecommendationStats, RegisteredServers, FavoriteServers) correspond directly to the newly introduced collections. In addition, several existing endpoint groups have been extended: the Offer endpoints now include federated retrieval routes, the Prosumer endpoints support blocked offer management, and Asset management now supports image upload and local storage.

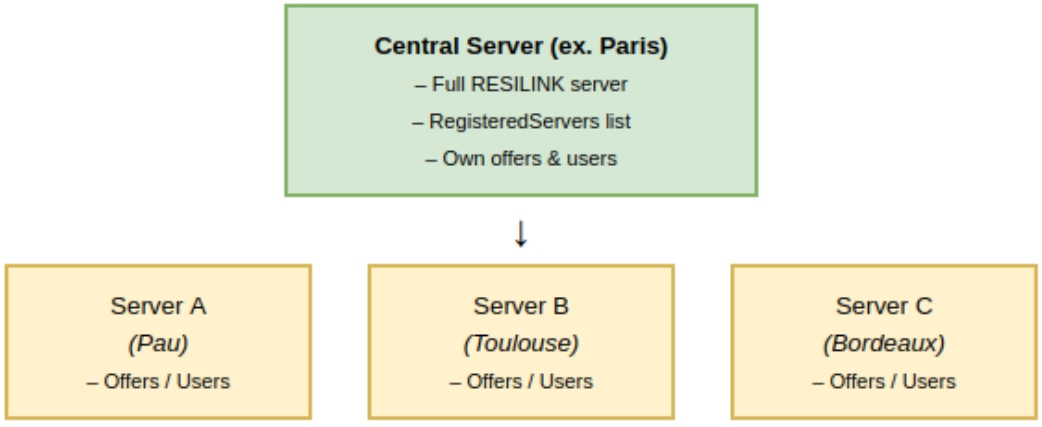
As part of this expansion, deprecated and redundant functionalities inherited from v2 have been removed to streamline the codebase. For instance, the automatic creation of AssetType entries upon asset creation has been deactivated; asset types must now be defined explicitly before being referenced by an asset, giving administrators full control over the available categories.

4.2.3. Multi-Server Federation

One of the most innovative features of v3 is its support for multi-server federation. This mechanism allows independent RESILINK deployments to form networks, enabling users on one server to discover offers published on other servers. This feature is the cornerstone of our “platform-of-platforms” approach.

The federation operates with a central server that acts as a directory for the network. Each server wishing to join registers itself with this central server. The central server is itself a full RESILINK instance with its own offers and users; it additionally maintains the list of all registered servers.

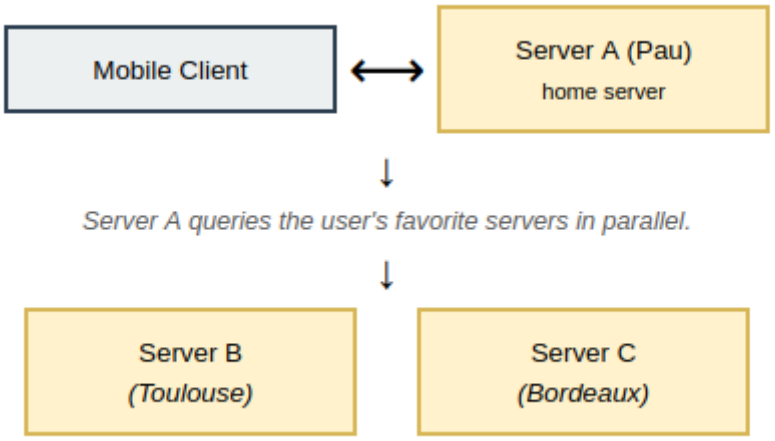
Federation Network Topology



Inter-server communication is authenticated using a shared secret (the RESILINK network key), passed as a dedicated HTTP header, ensuring that only servers belonging to the same trusted network can exchange data.

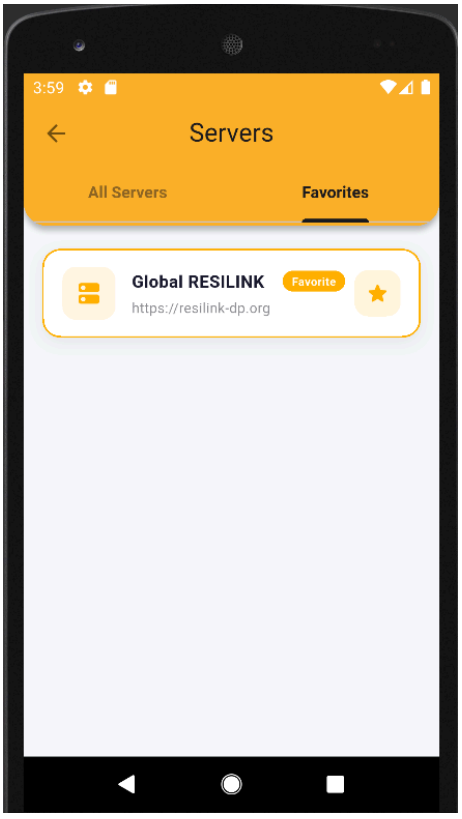
The federation supports flexible topologies: a single central registry with multiple leaf servers, multiple independent networks with separate keys, geographic partitioning by region, or domain specialization where different servers handle different resource categories.

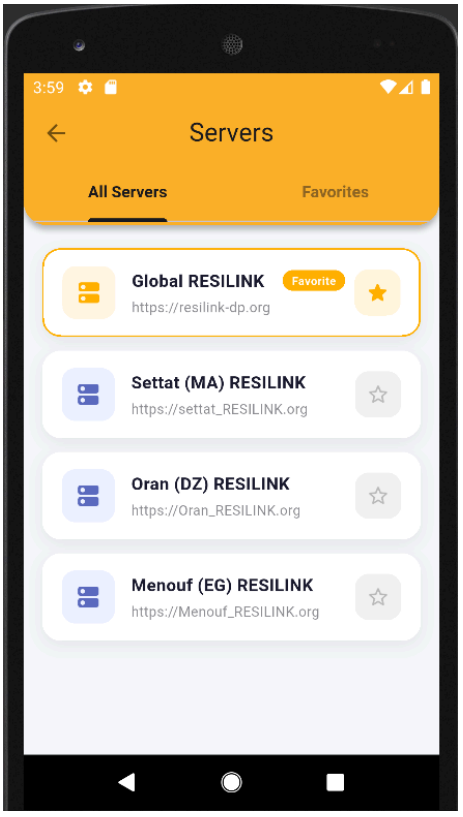
User interaction



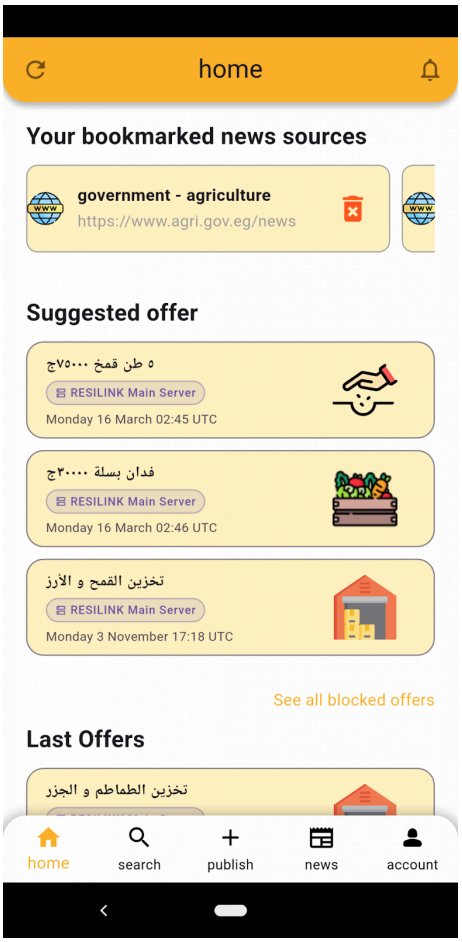
The mobile application always communicates with a single server (the user's home server). When a user requests federated offers, it is the home server that queries the user's favorite external servers in parallel, filters out blocked offers, merges the results with its own local offers, and returns a consolidated response to the client.

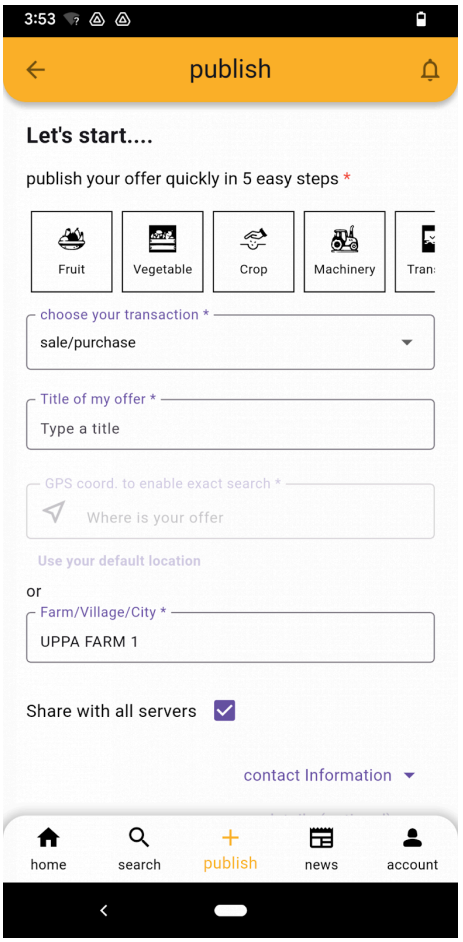
Although the RESILINK Mobile Application will be discussed in more detail in D2.4b “Final report on test and validation of the full RESILINK framework”, below are the Multi-Server Federation functionality integrated into the mobile application.





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.





4.2.4. Security Enhancements

The v3 platform also introduces several security improvements over v2:

- **Password Hashing**: user passwords are now hashed using bcrypt, replacing the previous storage approach. A migration script is provided to upgrade existing databases.
- **Data Encryption**: sensitive user fields (email addresses, phone numbers) are encrypted at rest using AES-256-CBC with random initialization vectors. A migration script is provided to encrypt existing plaintext data.
- **JWT Authentication**: JWT-based authentication already existed in v2, but v3 introduces two dedicated middlewares: optionalAuth.js (which handles both mandatory and optional token validation depending on the endpoint) and serverAuth.js (which validates inter-server network keys). The system now supports multi-device sessions, and a daily scheduled job cleans expired tokens.

- **Configurable Access Control**: the TOKEN_REQUIRED environment variable controls whether read-only (GET) endpoints require authentication. When set to false, public browsing is allowed without login.
- **Structured Logging**: all API operations are logged to a dedicated Logs database with five collections (ConnectionLogs, GetLogs, PutLogs, PatchLogs, DeleteLogs), supporting audit trails and troubleshooting. Log entries now retrieve the username from the last active token, improving traceability by associating each operation with the authenticated user. In addition, unnecessary or sensitive data (such as raw passwords or internal stack traces) has been cleansed from log payloads to reduce noise and prevent accidental exposure of confidential information.

4.2.5. Recommendation System

The v3 platform includes a built-in recommendation engine designed to improve user experience as the marketplace grows. The system operates in two stages:

- **Real-time tracking**: every time a user views an offer, the system records which asset type was consulted, incrementing a per-user counter in the RecommendationStats collection.
- **Daily aggregation**: a scheduled CRON job aggregates individual statistics into the GlobalRecommendationStats collection, producing a platform-wide view of the most-consulted categories and most active offer creators.

The mobile application collect these statistics to display personalized "recommended for you" sections based on the user's browsing history, and for example, a "trending categories" sections based on platform-wide activity can be implemented.

4.2.6. Offer Lifecycle Management

The management of offers has been enhanced in v3 to reflect the autonomous nature of the platform:

- **Auto-Increment Identifiers**: offer and asset identifiers are now generated using a dedicated Counters collection, providing unique sequential IDs without relying on a synchronous queue.
- **Offer Blocking**: prosumers can block specific offers on a per-server basis. Blocked offers are automatically excluded from federated query results.
- **Validity Management**: offers include a validity limit date, beyond which they are automatically filtered out of query results. Offers with a depleted remaining quantity are also excluded.
- **Sharing Control**: the acceptSharing flag determines whether an offer is visible to federated queries from other servers. A migration script is provided for backward compatibility with existing data.

4.2.7. Note on Platform Variants

Two variants of the RESILINK platform coexist within the same repository. The main branch (v2) integrates with ODEP and supports blockchain-backed transaction management. The feature/without-odep branch (v3) operates autonomously, focusing on offer discovery and contact exchange.

Platform Variant Comparison

Feature	main (ODEP)	feature/without-odep (v3)
Data storage	ODEP + local MongoDB	Local MongoDB only
Database name	Resilink	ResilinkWithoutODEP
Collections	7	12
Endpoints	~30	~80+
Contract management	Active (blockchain)	Stub (interface only)
Request management	Active	Stub (interface only)
Federation	Not available	Full support
Recommendation engine	Not available	Full support
Offer blocking	Supported (not multi-server)	Full support (multi-server)
Image storage	Local storage (public/images/)	Local storage (public/images/)
Deployment dependency	ODEP infrastructure	MongoDB only
Hardware requirement	Standard server	SBC (Raspberry Pi) compatible

The autonomous variant is the version currently deployed for the RESILINK Living-Labs. It can operate independently on lightweight hardware, including single-board computers such a Raspberry Pi. The Contract and Request endpoints are preserved as interface stubs to maintain API compatibility, but transactions are handled outside the platform through direct prosumer-to-prosumer contact.

4.3. Installing & deploying RESILINK platform v3

To support interaction between the RESILINK mobile application and the ODEP infrastructure, the RESILINK Platform v3 acts as an intermediate software layer. It enhances data processing, manages user interactions, and integrates additional business logic. The platform source code is publicly available on GitHub at the following URL:

https://github.com/ZiQuwi/RESILINK_Render_Server/tree/main

The project can be downloaded directly or cloned using Git. Be sure to select the "main" branch when cloning. An alternative branch that is autonomous (without the use of ODEP) exists and is located in the “feature/without-odep” branch. A complete README is available in the repository which details dependencies, setup and server usage. The essential configuration steps are summarized below.

4.4.1. Database Setup

The platform uses two MongoDB databases:

- **ResilinkWithoutODEP** (or **Resilink** for the main branch): stores all application data including users, prosumers, offers, assets, and federation information.
- **Logs**: stores structured API interaction logs for monitoring and auditing.

Two installation approaches are supported.

4.4.4.1. Local MongoDB Installation

The project includes automated setup scripts located in the "scripts" directory (for Linux and Windows). These scripts:

- Install MongoDB and MongoShell locally
- Create both the Logs and ResilinkWithoutODEP databases
- Create the required collections
- Install Node.js and all project dependencies
- Generate three encryption keys required in the environment configuration:
  - ENCRYPTION_KEY
  - TOKEN_KEY
  - RESILINK_NETWORK_KEY
- Initialize the administrator account
- Create the .env file to be filled in, but with the keys already populated by the one created by the script

The RESILINK_NETWORK_KEY key generated should only be used if the server is to be the central server; otherwise, the key should be ignored and the key on the central server should be used. The other two keys must be copied into the environment configuration file

4.4.4.2. MongoDB Atlas (Cloud Deployment)

If a managed cluster is used, the following collections must be created inside the Resilink database:

- Asset
- AssetType
- News
- Offer

- Rating
- prosumer
- user
- RecommendationStats
- GlobalRecommendationStats
- RegisteredServers
- FavoriteServers
- Counters

Additionally, the Logs database must include collections for logging, such as:

- ConnectionLogs
- DeleteLogs
- GetLogs
- PatchLogs
- PutLogs

For example, the following script illustrates the required setup:

```
use Logs
db.createCollection("ConnectionLogs")
db.createCollection("DeleteLogs")
db.createCollection("GetLogs")
db.createCollection("PatchLogs")
db.createCollection("PutLogs")

use ResilinkWithoutODEP
db.createCollection("Asset")
db.createCollection("AssetType")
db.createCollection("News")
db.createCollection("Offer")
db.createCollection("Rating")
db.createCollection("prosumer")
db.createCollection("user")
db.createCollection("RecommendationStats")
db.createCollection("GlobalRecommendationStats")
db.createCollection("RegisteredServers")
db.createCollection("FavoriteServers")
db.createCollection("Counters")
```

4.4.2. Environment Configuration

A configuration file named RESILINK_Server.env must be created (or already created with the script) at the root of the project. This file controls network settings, security keys, database access, and federation parameters. The following table describes each variable:

Variable	Description
IP_ADDRESS	IP address or domain name on which the server listens
PORT	Port number for the HTTP server
SWAGGER_URL	Base URL for Swagger API documentation
SERVER_NAME	Human-readable name displayed in federated responses
ENCRYPTION_KEY	256-bit hex key for AES-256-CBC encryption
TOKEN_KEY	256-bit hex key for signing and verifying JWT tokens
TOKEN_REQUIRED	Controls whether GET endpoints require authentication (true/false)
CENTRAL_SERVER_URL	URL of the central RESILINK server for federation
RESILINK_NETWORK_KEY	Shared secret for inter-server authentication
DB_MODE	Boolean: “atlas” or “local”. ‘atlas’ to use a remote database / “local” for a local database
DB_URL	MongoDB connection string – main database
DB_LOGS_URL	MongoDB connection string – Logs database

Example configuration template (placeholder values):

```
IP_ADDRESS=resilink-dp.org
PORT=9990
SWAGGER_URL=https://resilink-dp.org
SERVER_NAME=RESILINK Server
ENCRYPTION_KEY=<your-256-bit-hex-key>
TOKEN_KEY=<your-256-bit-hex-key>
TOKEN_REQUIRED=false
CENTRAL_SERVER_URL=https://resilink-dp.org
RESILINK_NETWORK_KEY=<your-shared-network-key>
DB_MODE=atlas
DB_URL=mongodb+srv://<username>:<password>@<cluster-url>/ResilinkWithoutODEP
DB_LOGS_URL=mongodb+srv://<username>:<password>@<cluster-url>/Logs
```

After completing the database setup and environment configuration, install the project dependencies and start the server. If the installation was not performed using the script, run “**npm install**” first, then:

```
> node src/index.js
```

The platform will be available at the defined IP address and port, with Swagger documentation accessible at the URL specified by SWAGGER_URL followed by /api-docs. This interactive documentation allows developers and administrators to explore and test all endpoints directly from a web browser.

4.4. Use & Validation of RESILINK v3 for the Living-Labs

The RESILINK platform v3 is currently deployed on the same professional web hosting provider used for the previous version, ensuring continuous availability and stable performance. In the current stage of deployment, the platform operates in an autonomous mode — without reliance on the ODEP infrastructure — allowing the system to function reliably during real-world usage and testing phases. This configuration was adopted following extensive testing conducted at the end of the previous year, which demonstrated that running the platform independently from ODEP provided a more stable environment for user evaluation.

While the platform is presently used in autonomous mode, RESILINK v3 remains fully compatible with ODEP, and the system can be switched to operate in ODEP-integrated mode when needed. Additionally, the backend server offers flexible deployment options. A complete installation procedure is available for both Linux and Windows environments, including the automatic installation of MongoDB, initialization of the required databases, and configuration of encryption and authentication keys. Because of this flexibility, the platform can also be deployed on lightweight devices such as Raspberry Pi boards, for which a tailored deployment script is provided in a dedicated repository.

RESILINK v3 has now replaced previous versions for use with the mobile application, as earlier versions introduced major architectural differences that would significantly impact core functionalities and user experience. **While running both v2 and v3 simultaneously is technically possible, it is not recommended due to differences in data handling and feature behavior, making the systems difficult to align in practice**.

The deployed v3 platform is currently being used in a closed testing phase through the Google Play Store. Feedback collected throughout this period has indicated stable behavior and an absence of major technical issues. The system has demonstrated consistent performance during extended and varied usage conditions, validating the reliability of the platform for Living-Lab deployments.

The dedicated deliverable D2.4b “Final report on test and validation of the full RESILINK framework” describes all the testing phases required for the Google Play Store.

Furthermore, v3 provides improved deployment simplicity compared to v2. In particular, the ability to use a fully local MongoDB database, without requiring a cloud-based cluster, significantly lowers the barrier for installation in distributed or resource-constrained environments. This flexibility supports the long-term goal of enabling decentralized deployments across multiple community-level infrastructures.

Acronyms List

Acronym	Explanation
AES-256-CBC	Advanced Encryption Standard / Cipher Block Chaining
API	Application Programming Interface
APK	Android Package Kit
CRON	Command Run ON
HTTP	Hyper Text Transport Protocol
JWT	JSON Web Token
ODEP	Orange Decentralized Exchange Place
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.