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

Deliverable D2.2a

RESILINK resource sharing digital platform – v1

Responsible Editor:	ORANGE
Contributors:	UPPA
Document Reference:	RESILINK D2.2a
Distribution:	Public
Version:	1.1
Date:	March 2024

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 19th, 2024	Public release
V1.0	March 25th, 2024	First draft version for internal approval
V0.1	March 4th, 2024	First release for review

Executive Summary

Deliverable D2.2a describes the intermediate platform, referred to as the RESILINK platform, that stands between the RESILINK mobile app and the ODEP platform to implement additional functionalities on top of ODEP. It also describes the open API approach to enable third-party applications to use the RESILINK platform to build their own exchange platforms.

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 connect fields and farms resources, automatize and add intelligence in the agri-food value chain to provide simple application interfaces adapted to smallholders.

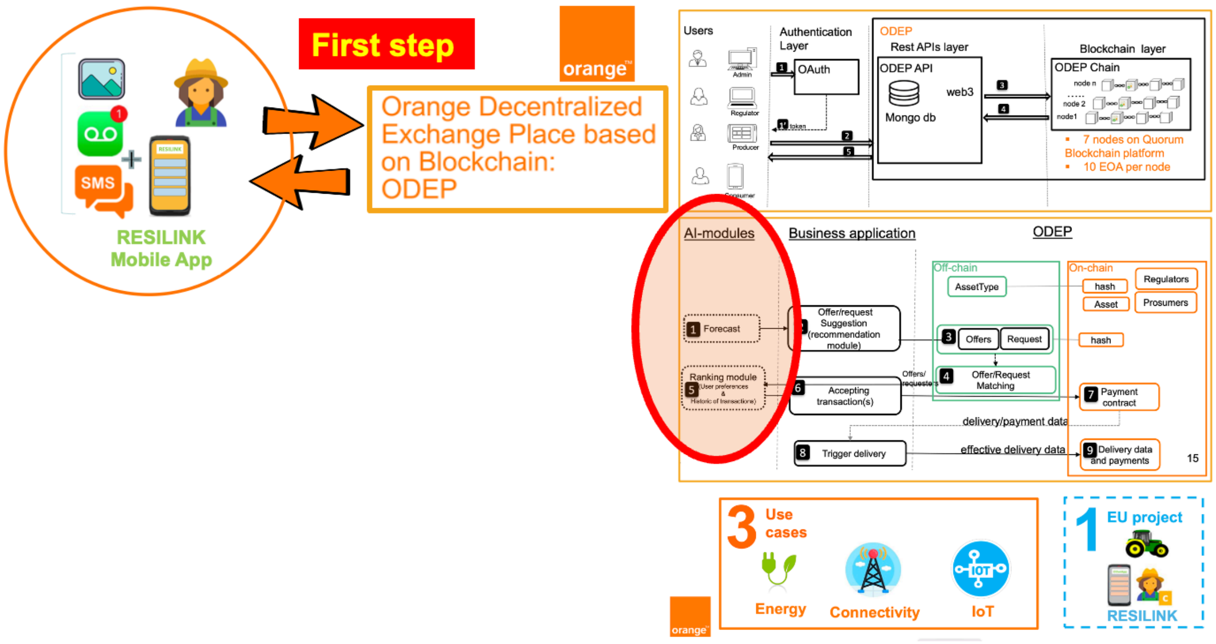
The lowest layer of the RESILINK digital resource management consists of the **Orange Decentralized Exchange Place (ODEP)** platform initially developed by Orange for energy and telecommunication ecosystems.

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 stand between the RESILINK mobile app and the ODEP platform to implement additional functionalities on top of ODEP.

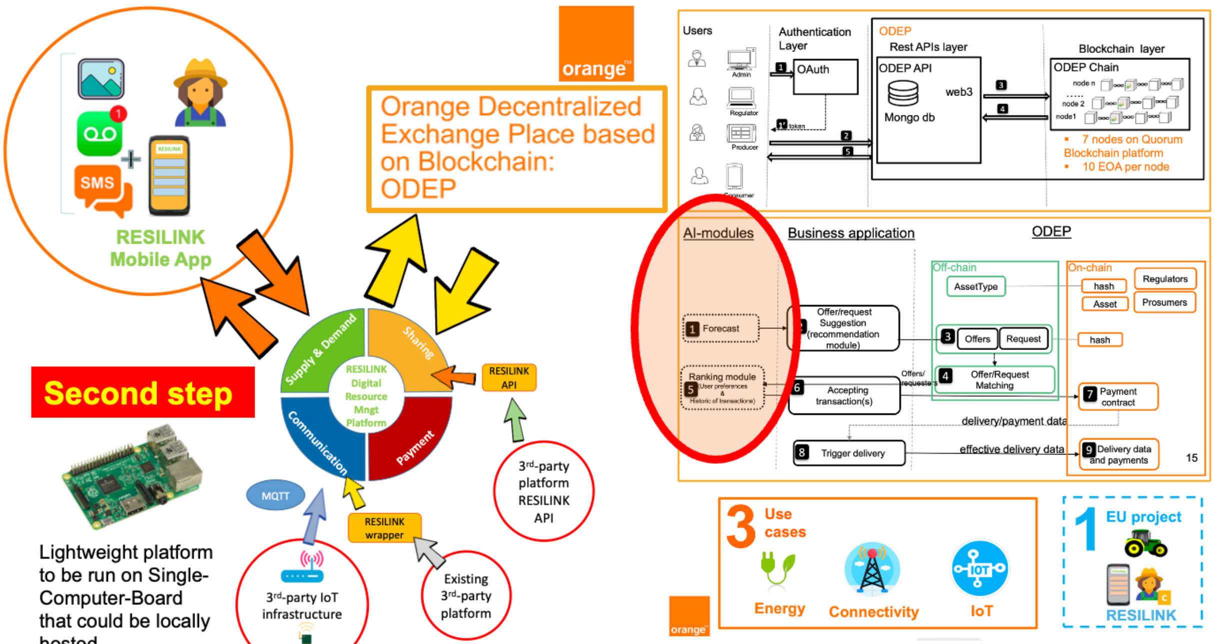
This deliverable complements D5.2b "Evaluation technical framework – v2" and describes in more detail the RESILINK platform.

2. RESILINK PLATFORM: MOTIVATIONS

RESILINK implements a dedicated intermediate RESILINK platform to serve as a front-end to the ODEP platform. The illustration below depicts this concept.



From Orange ODEP presentation. Salma Bradai, Sofrecom



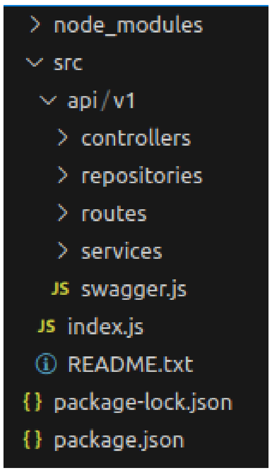
From Orange ODEP presentation. Salma Bradai, Sofrecom

The RESILINK platform will function as an API intermediary between the ODEP API and the RESILINK mobile app. Currently, the RESILINK platform is not deployed for open access but is exclusively available for developers.

The RESILINK platform will also leverage Swagger technology. This approach provides automatic generation of documentation, streamlining testing, and facilitating seamless interchange between different versions of the RESILINK platform in the future.

2.1. Architecture

Since the project RESILINK is still in the developmental phase, a simple server architecture has been chosen for the RESILINK platform and employed a backup file as a temporary database. It is important to note that a genuine database will be required for scaling in the future. The Proof-of-Concept (PoC) server is structured as follows – it is the result of a first PoC carried out by Manuel Munier from UPPA with the help of Marc Despland from Orange.


```
> node_modules
v src
  v api/v1
    > controllers
    > repositories
    > routes
    > services
    JS swagger.js
  JS index.js
  ⓘ README.txt
{} package-lock.json
{} package.json
```

In the root directory, the server configuration files with packages and versions, along with various modules essential for server operation, will be stored.

The "src" child folder will contain the server code, encompassing the "index.js" file and the "api" folder. The "index.js" file manages incoming calls (GET, POST, etc.) and defines the paths to the files/folders to be utilised, specifying the server version.

Within the version-specific folder (e.g., v1), there are the "swagger.js" file, along with the "controllers", "repositories", "routes" and "services" folders.

The Swagger file facilitates the definition of how URLs for accessing documentation are formed and allows the addition of extra functionalities.

Files within the "controllers" folder serve as controllers responsible for managing HTTP requests, validating data, and interacting with the appropriate services.

The "services" folder contains files with the application's business logic, handling data manipulation, communication with models, and execution of domain-specific tasks.

In the "routes" folder, the files define API entry points and determine which controller methods should be called in response to HTTP requests.

Finally, in the "repositories" folder, the files are responsible for accessing data from databases, file systems, or other sources. They encapsulate data access logic and enable the rest of the application to work with data in an abstract manner.

2.2. Proof of concept

Orange and UPPA investigated the technology building blocks: NodeJS, Javascript, and conducted tests on HTTP REST API.

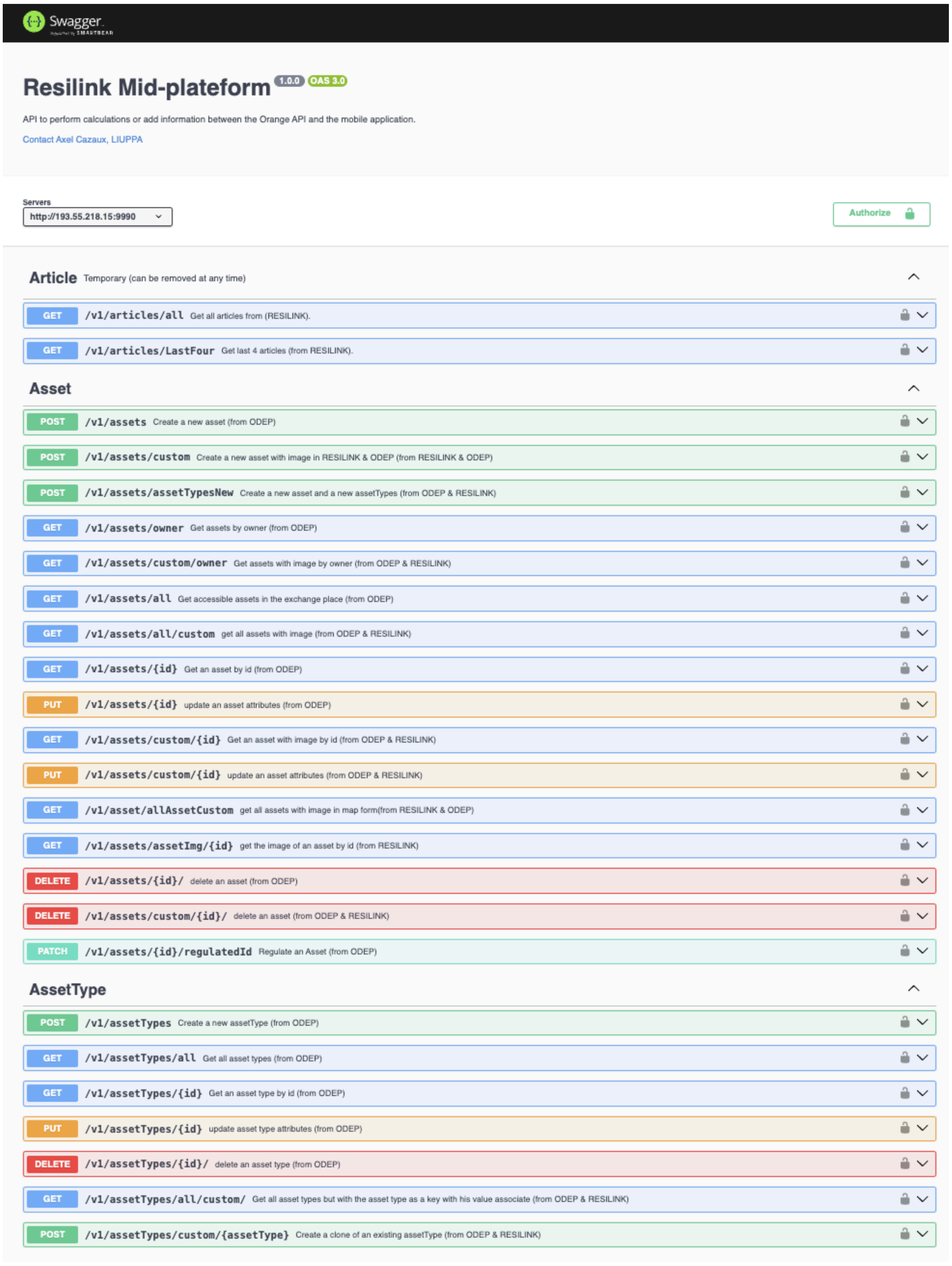
The PoC developed by Manuel Munier consists of two servers. The initial API intended for RESILINK, designed to act as an intermediary platform for data filtering, is coded in Node.js (Javascript) using Express and Swagger technology to simplify server construction and documentation.

In addition to this server, a second Java server, also using Swagger, serves as the ODEP API. Its role is focused on data retrieval and communication.

As of now, the PoC is operational, with both servers communicating effectively with each other.

2.3. Deployed RESILINK platform

The RESILINK platform is currently hosted on a free server provided by the hosting service Render, and one can access the Swagger documentation for the RESILINK platform: https://RESILINK-api.onrender.com/v1/api-docs/.



Swagger

Resilink Mid-plateform 1.0.0 OAS 3.0

API to perform calculations or add information between the Orange API and the mobile application.

Contact Axel Cazaux, LIUPPA

Servers
http://193.55.218.15:9990

Authorize

Article
Temporary (can be removed at any time)

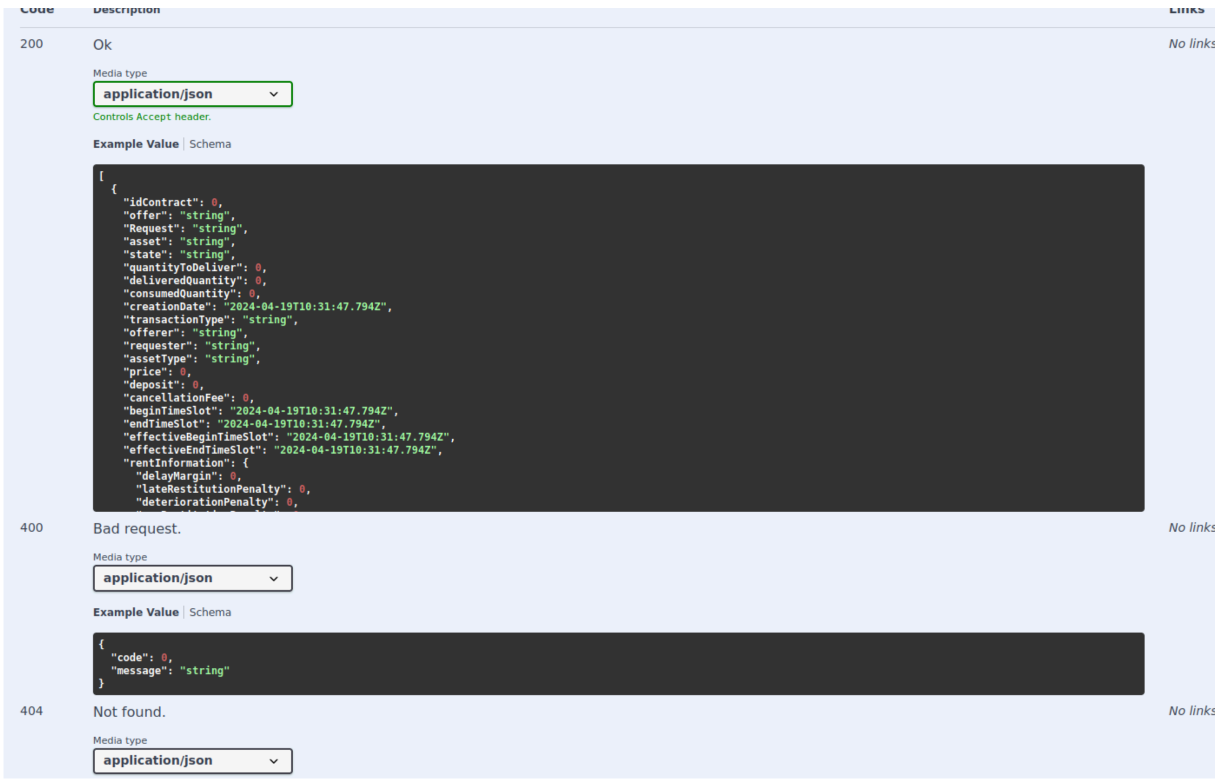
GET `/v1/articles/all` Get all articles from (RESILINK).
GET `/v1/articles/LastFour` Get last 4 articles (from RESILINK).

Asset

POST `/v1/assets` Create a new asset (from ODEP)
POST `/v1/assets/custom` Create a new asset with image in RESILINK & ODEP (from RESILINK & ODEP)
POST `/v1/assets/assetTypesNew` Create a new asset and a new assetTypes (from ODEP & RESILINK)
GET `/v1/assets/owner` Get assets by owner (from ODEP)
GET `/v1/assets/custom/owner` Get assets with image by owner (from ODEP & RESILINK)
GET `/v1/assets/all` Get accessible assets in the exchange place (from ODEP)
GET `/v1/assets/all/custom` get all assets with image (from ODEP & RESILINK)
GET `/v1/assets/{id}` Get an asset by id (from ODEP)
PUT `/v1/assets/{id}` update an asset attributes (from ODEP)
GET `/v1/assets/custom/{id}` Get an asset with image by id (from ODEP & RESILINK)
PUT `/v1/assets/custom/{id}` update an asset attributes (from ODEP & RESILINK)
GET `/v1/asset/allAssetCustom` get all assets with image in map form(from RESILINK & ODEP)
GET `/v1/assets/assetImg/{id}` get the image of an asset by id (from RESILINK)
DELETE `/v1/assets/{id}/` delete an asset (from ODEP)
DELETE `/v1/assets/custom/{id}/` delete an asset (from ODEP & RESILINK)
PATCH `/v1/assets/{id}/regulatedId` Regulate an Asset (from ODEP)

AssetType

POST `/v1/assetTypes` Create a new assetType (from ODEP)
GET `/v1/assetTypes/all` Get all asset types (from ODEP)
GET `/v1/assetTypes/{id}` Get an asset type by id (from ODEP)
PUT `/v1/assetTypes/{id}` update asset type attributes (from ODEP)
DELETE `/v1/assetTypes/{id}/` delete an asset type (from ODEP)
GET `/v1/assetTypes/all/custom/` Get all asset types but with the asset type as a key with his value associate (from ODEP & RESILINK)
POST `/v1/assetTypes/custom/{assetType}` Create a clone of an existing assetType (from ODEP & RESILINK)



Code Description Links

200 Ok — *No links*

Media type
application/json
Controls Accept header.

Example Value | Schema

```
[
  {
    "idContract": 0,
    "offer": "string",
    "Request": "string",
    "asset": "string",
    "state": "string",
    "quantityToDeliver": 0,
    "deliveredQuantity": 0,
    "consumedQuantity": 0,
    "creationDate": "2024-04-19T10:31:47.794Z",
    "transactionType": "string",
    "offerer": "string",
    "requester": "string",
    "assetType": "string",
    "price": 0,
    "deposit": 0,
    "cancellationFee": 0,
    "beginTimeSlot": "2024-04-19T10:31:47.794Z",
    "endTimeSlot": "2024-04-19T10:31:47.794Z",
    "effectiveBeginTimeSlot": "2024-04-19T10:31:47.794Z",
    "effectiveEndTimeSlot": "2024-04-19T10:31:47.794Z",
    "rentInformation": {
      "delayMargin": 0,
      "lateRestitutionPenalty": 0,
      "deteriorationPenalty": 0,
```

400 Bad request. — *No links*

Media type
application/json

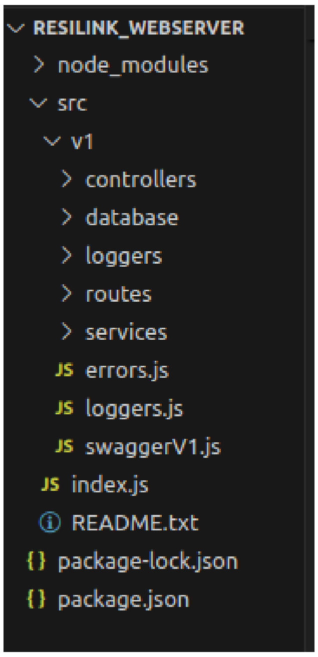
Example Value | Schema

```
{
  "code": 0,
  "message": "string"
}
```

404 Not found. — *No links*

Media type
application/json

The RESILINK platform is a server platform constructed with Node.js, hence using Javascript. The use of Javascript enables Swagger, a documentation library that is also applicable to a Java server, and Express, a framework enhancing the robustness of server-side applications. Additionally, a MongoDB (NoSQL) database is integrated with the server. This is particularly beneficial for storing data in memory that ODEP doesn't consider (e.g., a user's telephone number).



```
RESILINK_WEBSERVER
  node_modules
  src
    v1
      controllers
      database
      loggers
      routes
      services
      errors.js
      loggers.js
      swaggerV1.js
    index.js
    README.txt
  package-lock.json
  package.json
```

The server architecture mirrors that of the PoC, with the "repositories" folder now labelled as "database" and 2 new files named "error.js" and "loggers.js", which will respectively contain the various Error classes for differentiating exceptions and additions to log entries, associated with a name and a way of writing the log, as well as the destination file, in this case the "loggers" folder, which has been added alongside the "services", "controllers", "routes" and "database" folders.

The server's purpose is to redirect and transform data transmitted to and received from ODEP. All requests initially directed to ODEP have been replicated with the same header (path after the domain URL). The distinction lies in the domain name, and requests are now directed to an HTTPS address rather than HTTP, making it compatible with iPhones. All data transformation and calculation functions are now transferred from the mobile application to the server to enhance the telephone's performance.

Furthermore, it is possible to store additional data and associate it with other data in the database. If users choose to provide them, the RESILINK platform will save them in the database, associating it with their ODEP ID for future retrieval. To avoid complications with data setup, a need to exclude them from the data has been intended for transmission to ODEP.

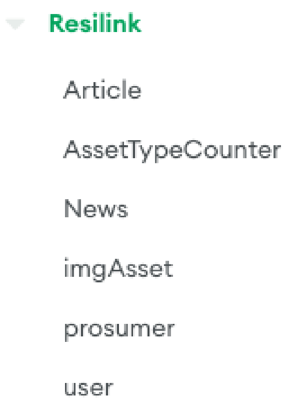
3. RESILINK PLATFORM: FEATURES FOR RESILINK

ODEP handles its own data but to use the RESILINK mobile application, there’s a need for additional data such as a user's phone number, or data that is not used by ODEP, such as items. In addition, the RESILINK mobile application needs functions that are more focused on “usage”, such as creating a user profile, which will need creating a "user" account and a "prosumer" account, without having to wait for the results of the 2 ODEP functions.

This is the reason a separated database along with functions to manage additional data are necessary for RESILINK and have been implemented in the RESILINK platform. The possibility of adding more data requirements and altering the data sent by ODEP can simplify the mobile application in the future or implement solutions to overcome ODEP's constraints.

3.1. RESILINK database

The MongoDB database currently contains 6 collections that are being used or will be used in the near future. They are illustrated on the right.



3.1.1. Article collection

The "Article" collection will be used to store data linked to articles concerning RESILINK. In the current vision, the list of articles is likely to be replaced by the list of news, which represents the institutes that will divulge news due to problems of data access and retrieval in the most widely used information-sharing software such as Facebook, Instagram, etc.

An element of the "Article" collection is formed as follows:

```
{
  _id: ObjectId() // Given by MongoDB to create a unique element
  title: String // Article title
  body: String // Article content
  link: String // Link to article
  image: String // Image converted to String containing base64 code
}
```

3.1.2. AssetTypeCounter collection

The “AssetTypeCounter” collection was created to overcome the problem of a single assetType for an asset in ODEP. Thanks to this collection, a so-called "child" assetTypes of a generic assetType can be created, so as to be able to reuse it. To do this, a counter is associated with an assetType and attach this counter to the assetType name to create a unique assetType identical to its parent when an user wants to create an assetType.

An element of the "AssetTypeCounter" collection is formed as follows:

```
{
  _id: ObjectId() // Given by MongoDB to create a unique element
  assetType: String // Father assetType
  count: Int32 // Number of children of an assetType created
}
```

3.1.3. News collection

The "News" collection, like the "Article" collection which stores papers or posts from news accounts, is designed to store only the name of an institute, newspaper or personality and one of its networks. Unlike the "Article" collection, this approach does not entail any danger of data appropriation or web scrapping problems.

An element of the "News" collection is formed as follows:

```
{
  _id: String // id of a news source
  url: String // account url
  Country: String // country of origin of the account/newspaper
  Institute: String // account/journal name
}
```

3.1.4. imgAsset collection

The "imgAsset" collection will enable a user to have a photo for his asset and his offer. Previously, the RESILINK mobile application used the imgur platform database, but this constrained users to a limit of stored images.

An element of the "imgAsset" collection is formed as follows:

```
{
  _id: ObjectId() // Given by MongoDB to create a unique element
  id: Int32 // asset id
  owner: String // user creating the asset
  img: String // base64 image set to string
}
```

3.1.5. prosumer collection

The "prosumer" collection will store all the data you want to add to a prosumer profile that ODEP doesn't take into account. For the moment, a prosumer only needs 1 additional piece of information: a list of "news" accounts registered by the user.

An element of the "prosumer" collection is formed as follows:

```
{
  _id: String // user name
  bookMarked: List<String> // list of news ids the prosumer wishes to save
}
```

3.1.5. user collection

The "prosumer" collection will store all the data you want to add to a prosumer profile that ODEP doesn't take into account. For the moment, a prosumer only needs 2 additional pieces of information: his telephone number and his address.

An element of the "prosumer" collection is formed as follows:

```
{
  _id: String // user id given by ODEP
  username: String // user name
  phoneNumber: String // user phone number
  password: String // user password
}
```

3.2. Additional RESILINK function

Numerous functions have been added and will undoubtedly be increased in the future to meet user or the RESILINK mobile app need.

All added functions accessible by request are on the Swagger page of the web server, but additions in relation to ODEP are not displayed directly on it. Here are the details of what the RESILINK platform does in addition.

3.1.1. Article

The Article entity represents a newspaper article. With the RESILINK platform, all or only the last 4 articles entered in the database can be retrieved. However, updating this collection automatically is not feasible, as web scraping would be required to retrieve data from several sites at once. The web scraping solution can be complex from a legal point of view, given that some data may be the property of someone else or some other entity.

3.1.2. Asset

Add the "img" key to the default "asset" entity. This is not a function added in addition to the ODEP function, which means that it is processed in every function that has the same query path as the ODEP queries. This key will contain the asset image if the user has set one when creating an asset. In addition to managing an asset image, there is a function for creating an asset and an associated assetType, a function for retrieving an asset map with the asset id and associated value as key, and the asset itself, as well as a function for retrieving an asset image via its id.

3.1.3. Contracts

For the "Contracts" entity, the ability to retrieve only the user's current contracts has been added.

3.1.4. News

News accounts can be retrieved by country or with a list of ids.

3.1.5. Offer

For offers, many additions have been made:

- Recovery of offers that are valid "for sale". Offers that can be displayed in the list of offers for sale.
- Retrieval of offers via a more flexible filter than ODEP's, in which you can specify
    - assetType
    - asset
    - price

    - whether the price should be higher, lower or equal
    - the date
    - if the date must be less than or greater than
    - existing properties such as gps location.
- Retrieve offers from the owner.
- Creation of an offer with its asset and assetType. This is a query that will contain the body of an offer and an asset (repeat of the previous function to create an asset and its assetType).

3.1.6. Prosumer

In the prosumer entity, similar to the "asset" entity, a key named "bookmarked" has been introduced, representing the list of news items saved by the user. Additionally, users are able to edit their "bookmarked" list and create a user profile and prosumer status simultaneously.

Furthermore, a distinction is made here between retrieving prosumers solely with ODEP data and those with additional data from the RESILINK database.

4. RESILINK PLATFORM: SECURITY AND INTEGRATION

The RESILINK platform has evolved to become more complete and accessible. The primary objective is to ensure accessibility and usability, particularly the API, for all users.

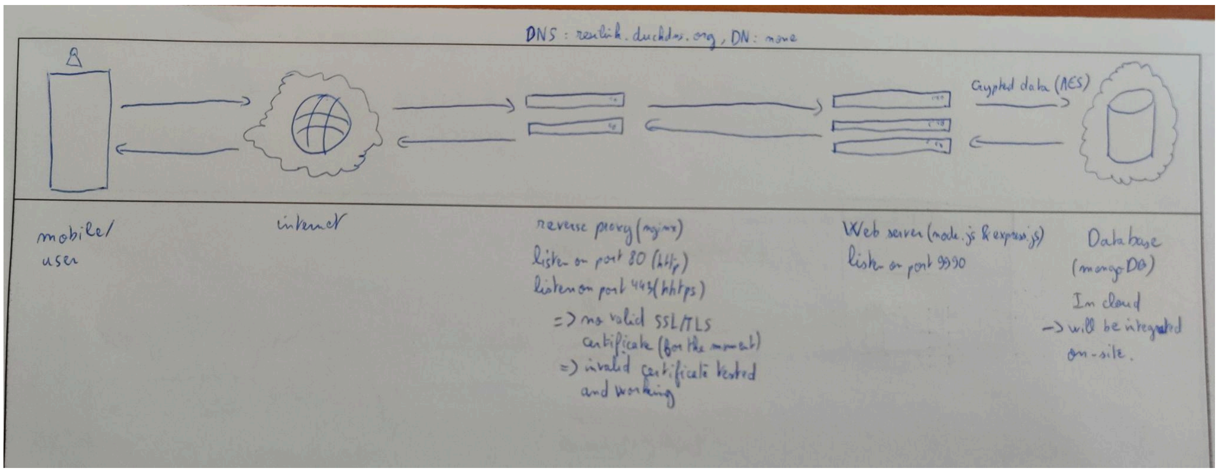
The server in its current state has been redesigned to be supported by servers at the University of Pau. The servers hosting the RESILINK platform are not suitable for real use of RESILINK.

Since hosting is free, Render has a lot of constraints when it comes to the speed of processing requests received and sent, a viable alternative where total control over the server is needed.

Therefore, hosting the RESILINK platform at the University of Pau is one of the solutions, but to be entitled to do so, several security questions had to be addressed to comply with the RGPD security standard.

To meet these requirements, several points have been added to the RESILINK platform and mobile application

- A reverse proxy to ensure request certification
- web server logs to trace all communications with a client and ODEP or the RESILINK database
- logs in the mobile application when a request is sent and received.


DNS : resilink.duckdns.org , DN : none
Crypted data (AES)
mobile/ user
internet
reverse proxy (nginx)
listen on port 80 (http)
listen on port 443 (https)
=> no valid SSL/TLS certificate (for the moment)
=> invalid certificate tested and working
Web server (node.js & express.js)
listen on port 9990
Database (mongoDB)
In cloud
-> will be integrated on-site.

4.1. Reverse proxy and certification

The server operates as a web server with the express framework used in node.js. Therefore, the implementation of a reverse proxy was necessary to ensure protection and compliance with an SSL or TLS (security protocol) agreement for the server. The NGINX software was selected for the reverse proxy role, as it offers simpler reverse proxy setup, easier configuration and better handling of so-called static requests (requests containing images or files, for example) than its competitors.

Furthermore, the SSL/TLS contract can be set up directly in the web server, but having a reverse proxy provides an additional layer of security to mask the backend server or redirect via DNS.

But to be able to create a valid SSL or TLS contract, a domain name and the associated DNS that comply with A and/or AAAA standards, i.e. IPv4 and/or IPv6, is needed. For the moment, a discussion to obtain a domain name via the Université de Pau et des Pays de l'Adour is in progress, and a valid associated contract has already been created.

Tests with DuckDNS (free DNS generator) and let's encrypt (certification authority for SSL/TLS contracts) have been carried out, but without success for the creation of a contract with no domain name and just a DNS.

However, tests have been carried out with a self-signed certificate and show that it is possible to make requests with a key to the server.

To further secure sensitive data, the telephone number (if the user has one) and password are encrypted using an AES encryption algorithm.

This encryption uses a secret key to decipher the stored encrypted data, as opposed to hashing (for passwords) or masking (for phone numbers), where it will be very complex to recover the initial value for masking.

4.2. Logs

A request history has been set up by recording the requests sent to the client (smartphone) and to the server. The key identifying the user making the request is the token of the "User" account currently using ODEP.

For logging purposes, this takes the form of a string containing data relating to the use of the RESILINK platform in the form of a json and the name of the class as well as the timestamp on which the log was produced. The format of the log on the client and server sides will be similar, and if data is in the form of a json, it will be easier to sort them if this is necessary.

On the server side, the json takes the form :

```
{
  service: "log type",
  from: "function name" ,
  error: "error that occurred", // appears only if there is an error
  tokenUsed: "user token",
  level: "degree of severity", // can be “info”, “warning” or “error”
  message: "custom message for more information",
  timestamp: "timestamp in UTC format 'YYYY--MM-DDTHH:mm:ss.SSSZ'",
}
```

For the server, the log types will differentiate access between ODEP and the RESILINK database, and so the log types will be as follows:

- getDataODEPService (used when retrieving data from ODEP)
- getDataRESILINKService (used if no data is retrieved from ODEP)
- updateDataODEPService
- updateDataRESILINKService
- deleteDataODEPService
- deleteDataRESILINKService
- patchDataODEPService
- connectDBRESILINKService

On the client side, the log looks like this:

“degree of severity - timestamp - function name - custom message - {data: ‘data to send/receive’}”

In the event of an error, the "data" key will be replaced with "error," and the error message retrieved should mirror that of the server.

In each scenario, the logging process follows this logic: upon receiving a log, if no error is detected, it will be logged at the info level. However, for server-related logs, what is recorded is the content of the data received from ODEP or the RESILINK database. However, for mobile-related logs, what is logged is the received status code (e.g., 200 or 401).

Additionally, when initiating data transmission for modification, creation, or deletion to the RESILINK platform, ODEP, or the RESILINK platform's database, a log with a warning level will be generated. This log includes the data to be transmitted before its actual transmission.The data can be a whole body (a JSON) or just an id for a deletion, or both.

The logs are saved locally on the server and on the mobile, and on the server they will be separated into several files to differentiate between logs for ODEP and logs concerning RESILINK, while on the mobile they will be on one file, and logs that are not of "error" level will be deleted automatically if the file exceeds 100mo in size.

These files are a temporary measure that will be replaced by a backup in collections. For testing purposes, it is easier to have the logs in a file to directly see where the errors are.

4.3. Difficulties

One of the most important outcomes is the deployment of the server in the cloud for access by all. A request was made to the UPPA to remedy this problem, but to register the RESILINK platform there, tests were carried out and they were either impossible or complicated to validate, although some were simpler.

To be able to create a certification contract and secure communications, the option is to either have to pay for the contract, or create one using an authority authorized to create certificates. Under these circumstances, buying a certificate is not necessary, since UPPA will provide us with one after confirming that minimum security is in place, so creating one is probably simpler. However, even with an authority like "Let's Encrypt", obtaining a DNS is still necessary. Precaution is needed to avoid phishing, and a paid domain name is required for validation, as ownership of the domain name is necessary.

After numerous experimentation and iteration, creating a certificate without a domain name is not possible in the knowledge that UPPA will be able to provide us with one too.

Some issues have been addressed (solutions may change):

- Create a reverse proxy to filter data before it reaches the server.
- The purpose of the reverse proxy is to obtain the SSL/TLS certification contract.
- Set up a logging system to trace requests on both server and client sides.
- Problem of creating several assets under one assetType solved by creating a child assetType for each asset.
- Server set-up at a host with a free option (temporary, as it is very inconvenient).
    - accessible via iphone as the host is https
- Setting up functions for RESILINK and its mobile application
    - quick creation of an offer, asset and assetType.
    - sorting implemented on the server is much less restrictive (requires much less information).
- Image management is no longer handled by the imgur platform, but by our own RESILINK platform.

Some problems still need to be solved:

- Obtaining a certificate and testing it

    - deploy the RESILINK platform on an in-house or pay-as-you-go host.
- Recovering data from newspaper articles or instagram profiles etc (abandoned due to the disadvantages of web scraping).
- Distinction between two local servers data with shared data.
    - e.g. Local server B cannot access assetType created from local server A.

4.4. Plan for the future

The RESILINK platform in its current state allows the mobile application to function properly with its new functionalities, but it must be able to respond to more needs.

The RESILINK platform will enable other services to use it, which is an intermediary to ODEP. A discussion on this subject has already taken place to put forward ideas, and they are as follows:

- Distinguish entities by their origin (RESILINK and other services) by putting the service that created the assetType in the "description" key, or by asking ODEP to set up a corresponding key.
- Provide frameworks to create a mobile application quickly that can use RESILINK.
- To solve the problem of non-access to data created on a local server by another server, a request may be made to ODEP to create a field to handle such cases.

Acronyms List

Acronym	Explanation
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Android Package Kit
DNS	Domain Name Server
HTTP	Hyper Text Transfer Protocol
IPv4/IPv6	Internet Protocol v4 / v6
JSON	JavaScript Object Notation
ODEP	Orange Decentralized Exchange Place
PoC	Proof-of-Concept
SSL/TLS	Secure Socket Layer/Transport Layer Security
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.