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

Deliverable D2.2c

RESILINK resource sharing digital platform – v3

Annex: Installation & Setup Manual

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

Contributors table

DOCUMENT SECTION	AUTHOR(S)
SECTION 1	C. Pham (UPPA)
SECTION 2	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

This document is an annex on the installation & setup of the RESILINK resource sharing digital platform – v3 (see D2.2c). It provides complete, step-by-step instructions for installing, configuring, and deploying the RESILINK platform, covering the 3 main components:

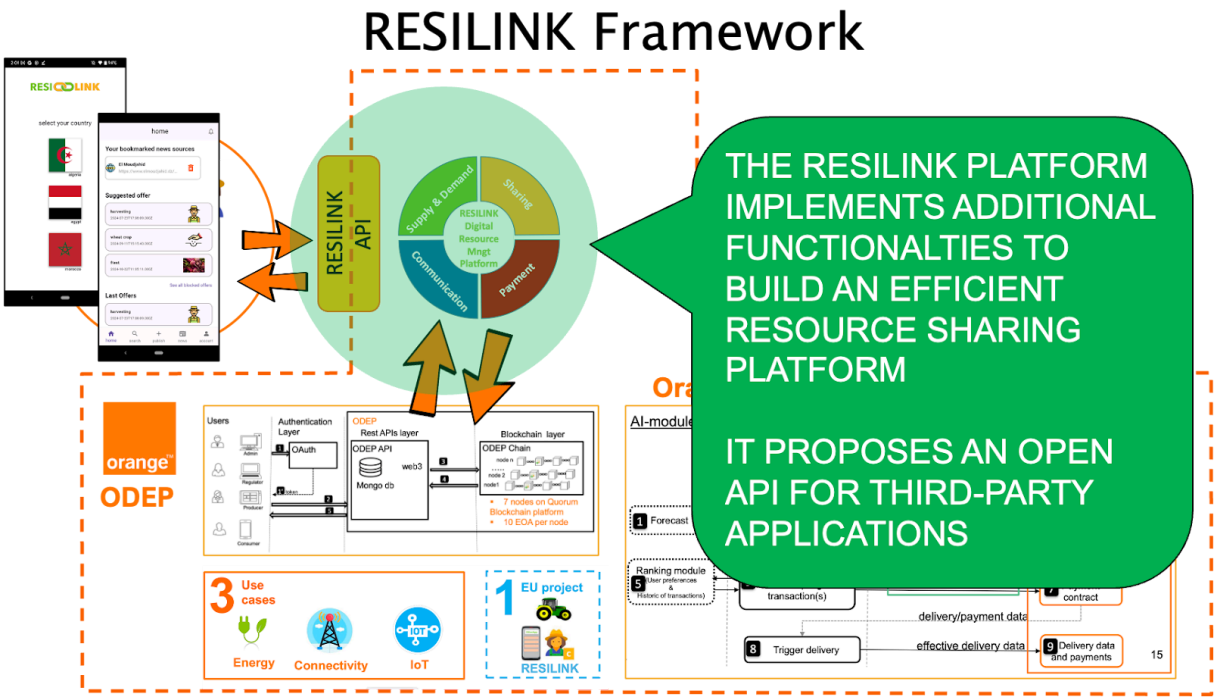
- The RESILINK Backend Server (Node.js + MongoDB)
- The RESILINK Mobile Application (Flutter/Android), including code configuration, APK/AAB build, signing, and release
- The mobile app public distribution via the Google Play Store

.

Table of Contents

1. Introduction

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 it as “RESILINK Back-end Server” in this D2.2c Annex document.



2. Back-end Server Installation

The RESILINK Back-end Server is a Node.js REST API backed by MongoDB. This section covers cloning the repository, configuring the environment, and starting the server on both Linux and Windows.

2.1. Repository Setup

The source code is hosted on GitHub. You can clone the repository or download a ZIP archive: https://github.com/ZiQuwi/RESILINK_BackEnd_Server

Option A — Clone (recommended)

To access all branches:

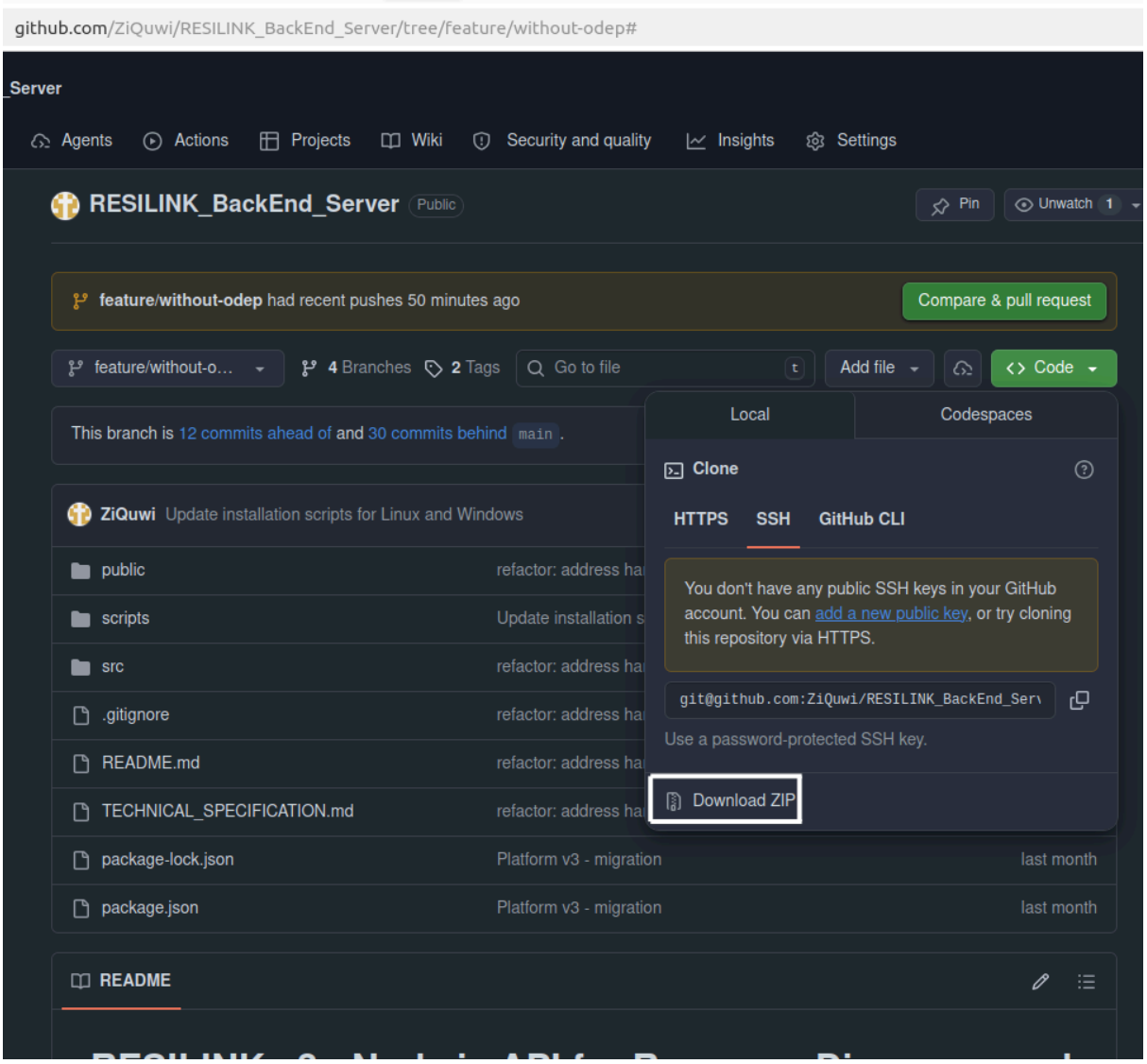
```
git clone https://github.com/ZiQuwi/RESILINK_BackEnd_Server.git
cd RESILINK_BackEnd_Server
git fetch
git checkout feature/without-odep
```

To clone only the target branch directly:

```
git clone -b feature/without-odep
https://github.com/ZiQuwi/RESILINK_BackEnd_Server.git
```

Option B — Download ZIP

Navigate to the GitHub repository, select the desired branch (feature/without-odep), then click Code → Download ZIP and extract the archive to your working directory.

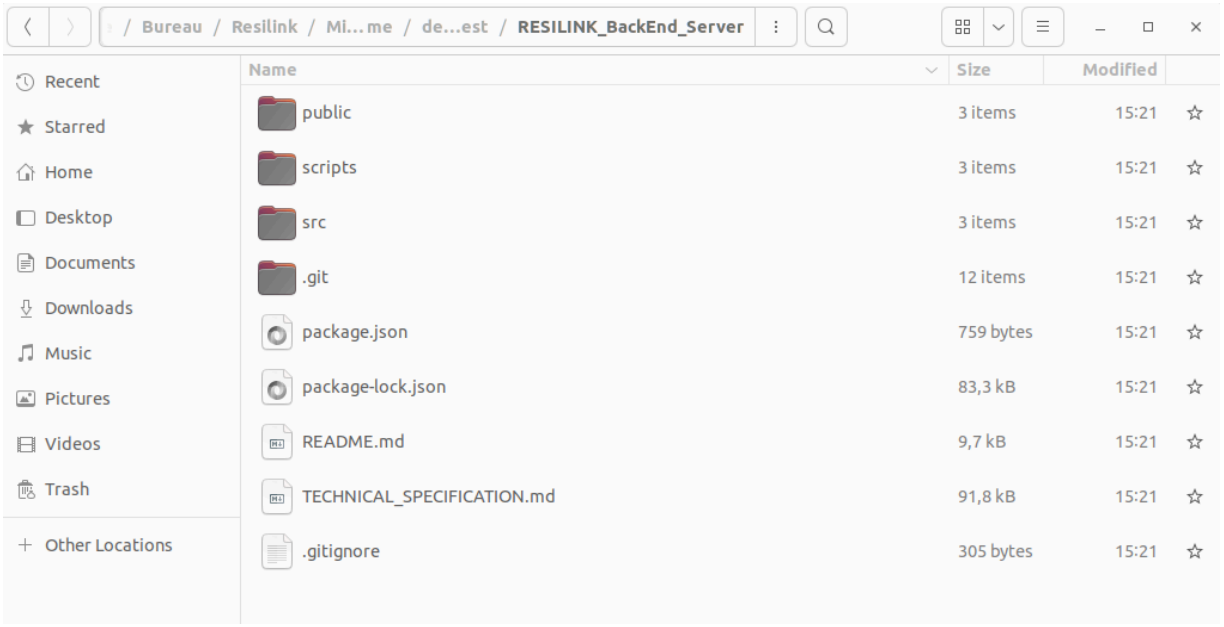


2.2. Project Structure

After cloning or extracting, verify that the project root contains the following files and folders:

Item	Description
public/	Static public assets
scripts/	Installation & helper scripts
src/	Application source code

package.json	Node.js project manifest
package-lock.json	Dependency lock file
README.md	Project readme
TECHNICAL_SPECIFICATION.md	Technical specification
.gitignore	Git ignore rules



2.3. Pre-Installation Configuration

Before running the installation script, **you must set a secure admin password. The default password in the script is 123456 and must be changed**.

Windows — Edit scripts/install_resilink_helper.ps1 at line ~133:

```
const hashedPassword = await bcrypt.hash("YOUR_SECURE_PASSWORD", 10);
```

```
      const cipher = crypto.createCipheriv("aes-256-cbc", encryptionKey, iv);
      let encrypted = cipher.update(value, "utf8", "hex");
      encrypted += cipher.final("hex");
      return iv.toString("hex") + ":" + encrypted;
    }

    const userCol = mainDb.collection("user");
    const existing = await userCol.findOne({ userName: "admin" });
    if (existing) {
      console.log("Admin user already exists. Skipping.");
    } else {
      const hashedPassword = await bcrypt.hash("123456", 10);
      const adminUser = {
        _id: new ObjectId().toString(),
        phoneNumber: "",
        userName: "admin",
        firstName: "admin",
        lastName: "admin",
        roleOfUser: "admin",
        email: encryptAES("admin@gmail.com"),
        password: hashedPassword,
        gps: "",
        createdAt: new Date().toISOString(),
```

Linux — Edit scripts/install_resilink.sh at line ~170:

```
const hashedPassword = await bcrypt.hash("YOUR_SECURE_PASSWORD", 10);
```

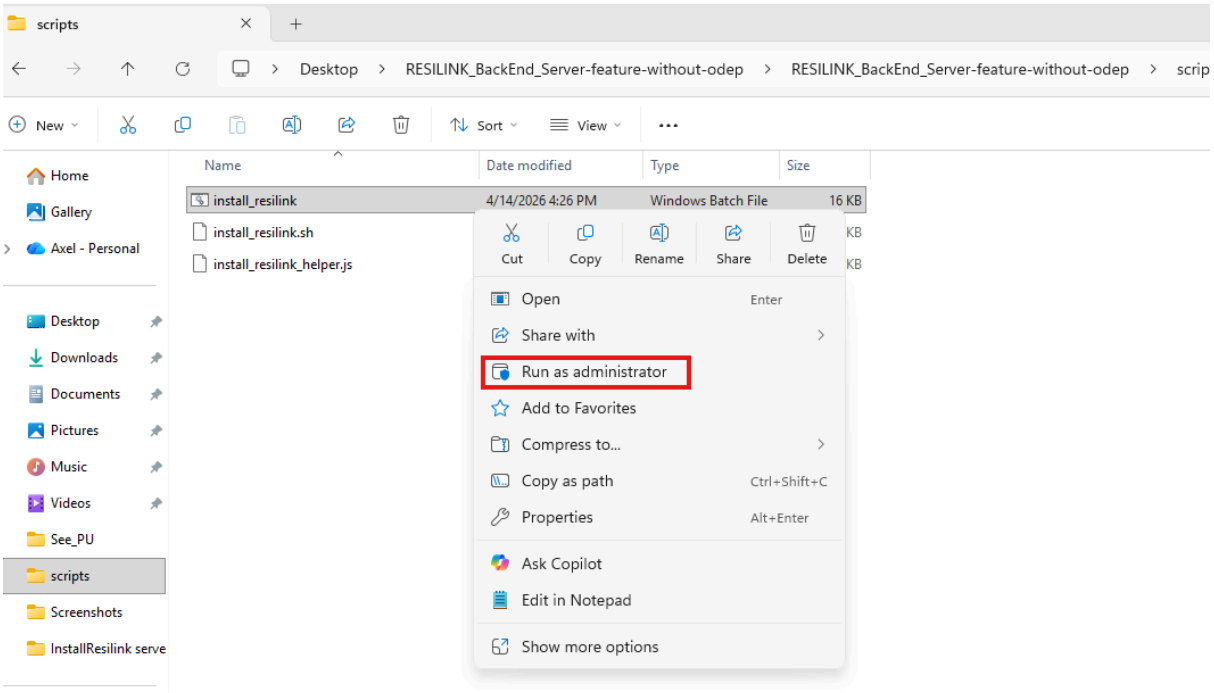
```
(async () => {
  const client = new MongoClient(uri);
  try {
    await client.connect();
    const db = client.db("ResilinkWithoutODEP");
    const userCol = db.collection("user");

    const existing = await userCol.findOne({ userName: "admin" });
    if (existing) {
      console.log("Admin user already exists. Skipping.");
      return;
    }

    const hashedPassword = await bcrypt.hash("123456", 10);

    const adminUser = {
      _id: new ObjectId().toString(),
      phoneNumber: "",
      userName: "admin",
      firstName: "admin",
      lastName: "admin",
      roleOfUser: "admin",
      email: encryptAES("admin@gmail.com"),
      password: hashedPassword,
      gps: "",
      createdAt: new Date().toISOString(),
```

2.4. Install the server

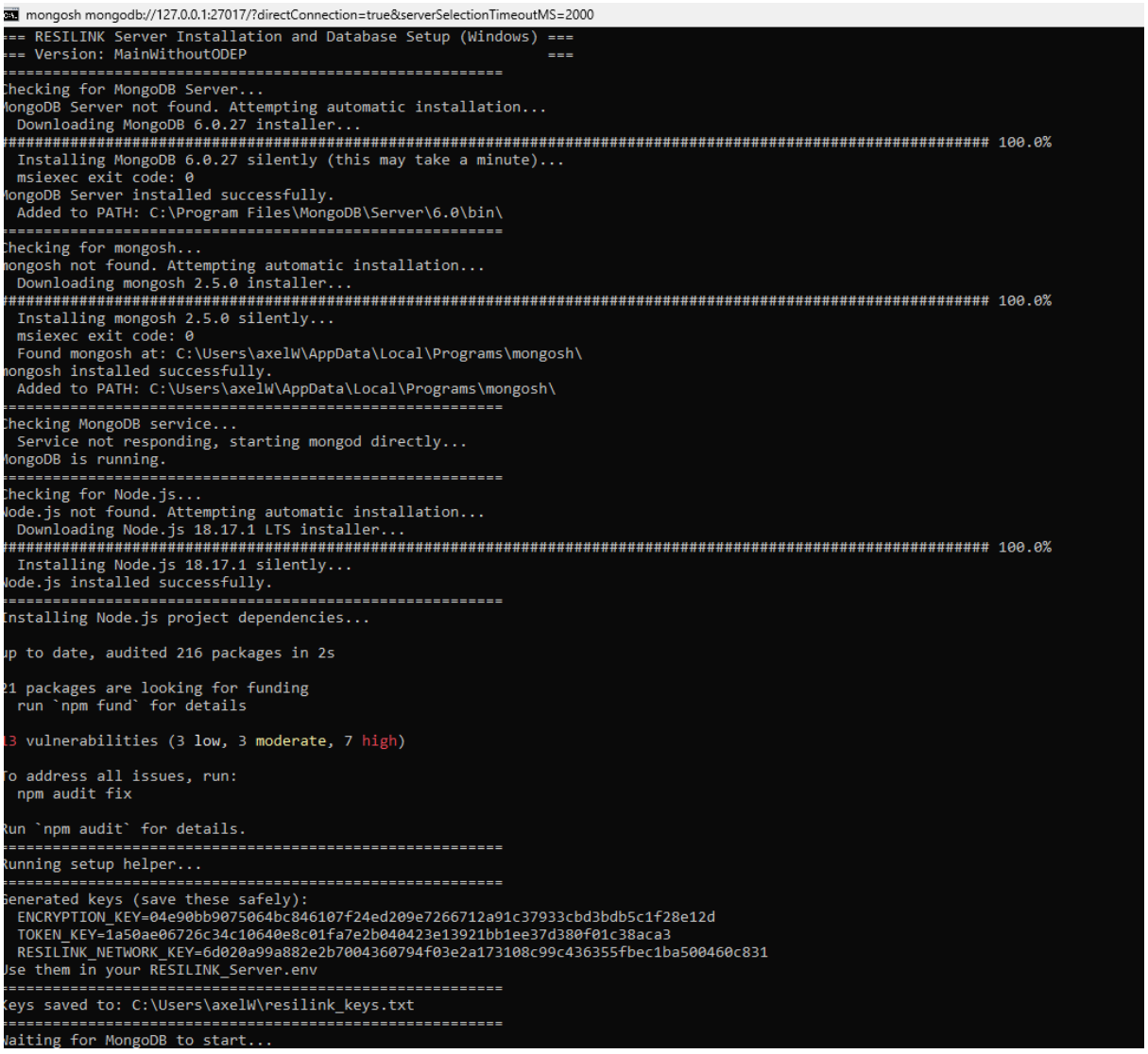
2.4.1. Run the script (Recommended)

The installation script automatically installs MongoDB, Mongosh, Node.js, and all project dependencies, then creates the database and the admin user.

Windows — Right-click scripts/install_resilink.bat and select Run as administrator or run the file from the PowerShell.



Result of running the script.


mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
=== RESILINK Server Installation and Database Setup (Windows) ===
=== Version: MainWithoutODEP ===
Checking for MongoDB Server...
MongoDB Server not found. Attempting automatic installation...
Downloading MongoDB 6.0.27 installer...
100.0%
Installing MongoDB 6.0.27 silently (this may take a minute)...
msiexec exit code: 0
MongoDB Server installed successfully.
Added to PATH: C:\Program Files\MongoDB\Server\6.0\bin\
Checking for mongosh...
mongosh not found. Attempting automatic installation...
Downloading mongosh 2.5.0 installer...
100.0%
Installing mongosh 2.5.0 silently...
msiexec exit code: 0
Found mongosh at: C:\Users\axelW\AppData\Local\Programs\mongosh\
mongosh installed successfully.
Added to PATH: C:\Users\axelW\AppData\Local\Programs\mongosh\
Checking MongoDB service...
Service not responding, starting mongod directly...
MongoDB is running.
Checking for Node.js...
Node.js not found. Attempting automatic installation...
Downloading Node.js 18.17.1 LTS installer...
100.0%
Installing Node.js 18.17.1 silently...
Node.js installed successfully.
Installing Node.js project dependencies...
up to date, audited 216 packages in 2s
21 packages are looking for funding
run `npm fund` for details
13 vulnerabilities (3 low, 3 moderate, 7 high)
To address all issues, run:
npm audit fix
Run `npm audit` for details.
Running setup helper...
Generated keys (save these safely):
ENCRYPTION_KEY=04e90bb9075064bc846107f24ed209e7266712a91c37933cbd3bdb5c1f28e12d
TOKEN_KEY=1a50ae06726c34c10640e8c01fa7e2b040423e13921bb1ee37d380f01c38aca3
RESILINK_NETWORK_KEY=6d020a99a882e2b7004360794f03e2a173108c99c436355fbec1ba500460c831
Use them in your RESILINK_Server.env
Keys saved to: C:\Users\axelW\resilink_keys.txt
Waiting for MongoDB to start...

```
eys saved to: C:\Users\axelW\resilink_keys.txt
============================================================
aiting for MongoDB to start...
ongoDB is up!
============================================================
reating databases and collections...
ogs database collections created.
esilinkWithoutODEP database collections and indexes created.
============================================================
reating admin user...
dmin user created successfully (password: bcrypt, email: AES-encrypted).
============================================================
pdating RESILINK_Server.env with generated keys...
eys updated in RESILINK_Server.env
============================================================
============================================================

== Setup completed successfully ===

ext steps:
 1. Review and configure RESILINK_Server.env:
    - IP_ADDRESS, PORT, SWAGGER_URL
    - SERVER_NAME (display name for federated server discovery)
    - TOKEN_REQUIRED (true/false - controls GET endpoint authentication)
    - CENTRAL_SERVER_URL (URL of the central federation server)
    - DB_URL (MongoDB connection string for ResilinkWithoutODEP)
    - DB_LOGS_URL (MongoDB connection string for Logs database)
 2. Start the server: node src\index.js
 3. Default admin credentials: admin / admin123

ppuyez sur une touche pour continuer...
```

Linux —

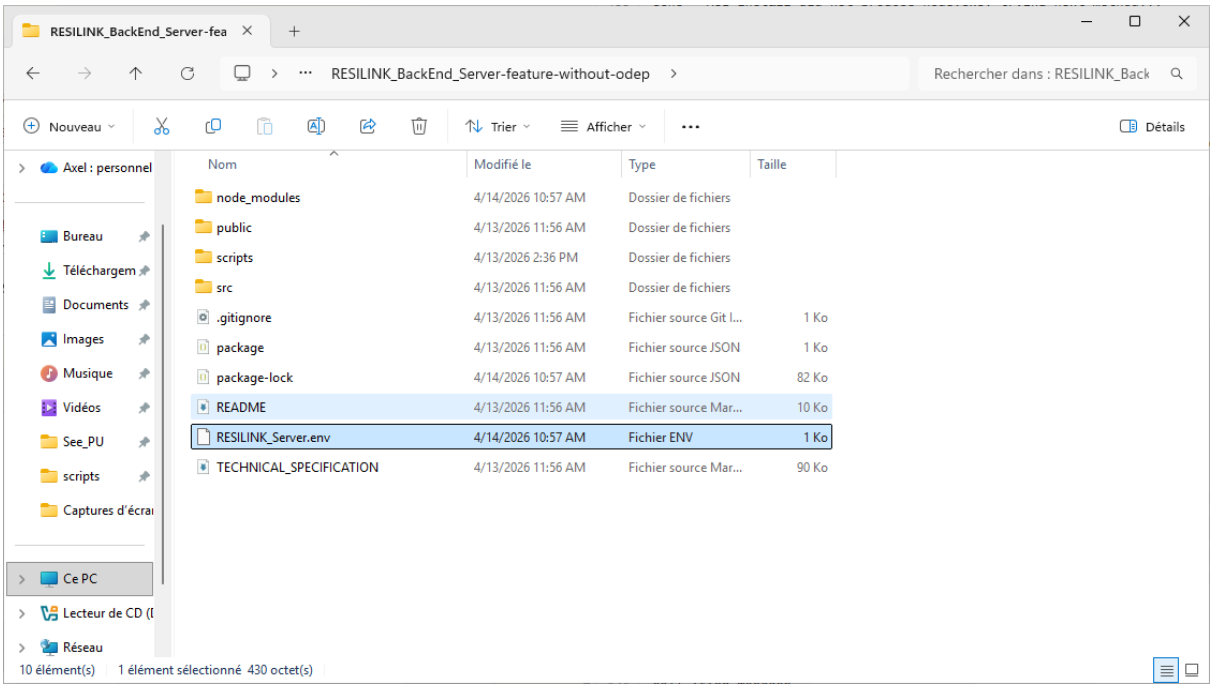
```
cd scripts
./install_resilink.sh
```

The script will prompt for your sudo password. It produces the same result as on Windows.

```
acazaux007@SCINFR038:~/Bureau/Resilink/Mid_Plateforme/deployment_test/RESILINK_BackEnd_Server/scripts$ ./install_resilink.sh
=== RESILINK Server Installation and Database Setup (Linux) ===
=== Version: MainWithoutODEP                                 ===
============================================================
[sudo] password for acazaux007:
```

Once the script completes successfully, the following artifacts are created:

- A .env file in the project root, pre-populated with generated keys
- A backup key file containing all generated secrets:
    - Linux: ~/resilink_keys.txt
    - Windows: C:\Users\<User>\resilink_keys.txt



2.4.2. Manual Installation (For advanced users)

If you prefer not to use the automated script, or if the script fails on your system, you can perform each installation step manually. This section mirrors exactly what the script does, in the same order.

2.4.2.1. Install MongoDB

Option A — Via terminal (Ubuntu 22.04+)

Import the MongoDB public GPG key and add the official repository:

```
curl -fsSL https://pgp.mongodb.com/server-6.0.asc | \
    sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor


echo "deb [ arch=amd64,arm64
signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg ] \
https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse" \
    | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list


sudo apt update
sudo apt install -y mongodb-org
```

Enable and start the MongoDB service:

```
sudo systemctl enable mongod
sudo systemctl start mongod
```

Option B — Via installer (Windows or manual download)

Go to the official MongoDB download page:

https://www.mongodb.com/try/download/community

Select version **6.0**, your platform, and download the installer. Run it and follow the setup wizard. Make sure to check the option **Install MongoDB as a Service** so it starts automatically.

2.4.2.2. Install Node.js

Option A — Via terminal (Linux)

```
sudo apt install -y nodejs npm
```

Verify the installation:

```
node -v
npm -v
```

Option B — Via installer (Windows or manual download)

Go to the official Node.js download page: **https://nodejs.org/en/download**

Download the **LTS** version for your platform and run the installer. The installer includes npm automatically.

2.4.2.3. Install Project Dependencies

From the project root directory, run:

```
npm install
```

This installs all packages listed in package.json, including bcrypt, mongoose, and all other dependencies required by the API.

2.4.2.4. Generate Security Keys

The server requires three cryptographic keys: an AES encryption key, a JWT token key, and a RESILINK network key. Generate them with the following commands:

Linux:

```
ENCRYPTION_KEY=$(openssl rand -hex 32)
TOKEN_KEY=$(openssl rand -hex 32)
RESILINK_NETWORK_KEY=$(openssl rand -hex 32)
```

```
echo "ENCRYPTION_KEY=$ENCRYPTION_KEY"
echo "TOKEN_KEY=$TOKEN_KEY"
echo "RESILINK_NETWORK_KEY=$RESILINK_NETWORK_KEY"
```

Windows (PowerShell):

```
$ENCRYPTION_KEY = -join ((1..32) | ForEach-Object { '{0:x2}' -f (Get-Random
-Max 256) })
$TOKEN_KEY = -join ((1..32) | ForEach-Object { '{0:x2}' -f (Get-Random -Max
256) })
$RESILINK_NETWORK_KEY = -join ((1..32) | ForEach-Object { '{0:x2}' -f
(Get-Random -Max 256) })

Write-Host "ENCRYPTION_KEY=$ENCRYPTION_KEY"
Write-Host "TOKEN_KEY=$TOKEN_KEY"
Write-Host "RESILINK_NETWORK_KEY=$RESILINK_NETWORK_KEY"
```

Save these values, you will need them in the next step.

2.4.2.5. Create and Configure the .env File

Create a file named **RESILINK_Server.env** in the project root and fill in KEY values:

```
ENCRYPTION_KEY=YOUR_GENERATED_ENCRYPTION_KEY
TOKEN_KEY=YOUR_GENERATED_TOKEN_KEY
RESILINK_NETWORK_KEY=YOUR_GENERATED_NETWORK_KEY
IP_ADDRESS=
PORT=
SWAGGER_URL=
SERVER_NAME=
TOKEN_REQUIRED=
CENTRAL_SERVER_URL=
DB_MODE=local
DB_URL=mongodb://127.0.0.1:27017/ResilinkWithoutODEP
DB_LOGS_URL=mongodb://127.0.0.1:27017/Logs
```

2.4.2.6. Create the MongoDB Databases and Collections

Wait for MongoDB to be ready, then execute:

```
mongosh --eval "
db = db.getSiblingDB('Logs');
db.createCollection('ConnectionLogs');
db.createCollection('DeleteLogs');
db.createCollection('GetLogs');
db.createCollection('PatchLogs');
db.createCollection('PutLogs');
db.createCollection('SecurityLogs');

db = db.getSiblingDB('ResilinkWithoutODEP');
db.createCollection('Asset');
db.createCollection('AssetType');
```

```
db.createCollection('Counters');
db.createCollection('FavoriteServers');
db.createCollection('GlobalRecommendationStats');
db.createCollection('News');
db.createCollection('Offer');
db.createCollection('Rating');
db.createCollection('RecommendationStats');
db.createCollection('RegisteredServers');
db.createCollection('prosumer');
db.createCollection('user');

db.RegisteredServers.createIndex({ serverName: 1 }, { unique: true });
db.RegisteredServers.createIndex({ serverUrl: 1 }, { unique: true });
db.Offer.createIndex({ id: 1 }, { unique: true });
db.Asset.createIndex({ id: 1 }, { unique: true });
db.AssetType.createIndex({ name: 1 }, { unique: true });
db.FavoriteServers.createIndex({ id: 1 }, { unique: true });
db.Rating.createIndex({ userId: 1 }, { unique: true });
db.RecommendationStats.createIndex({ name: 1 }, { unique: true });
db.prosumer.createIndex({ id: 1 }, { unique: true });
db.user.createIndex({ userName: 1 }, { unique: true });
db.user.createIndex({ email: 1 }, { unique: true });
"
```

Type **exit** to leave the mongosh shell.

2.4.2.7. Create the Admin User

Create a file named **createAdmin.js** in the project root with the following content, replacing **YOUR_ENCRYPTION_KEY** with the key generated in step 2.4.2.4. The default admin credentials are **admin / 123456**. Change the admin password directly in the file:

```
const bcrypt = require("bcrypt");
const crypto = require("crypto");
const { MongoClient, ObjectId } = require("mongodb");

const uri = "mongodb://127.0.0.1:27017";
const encryptionKey = Buffer.from("YOUR_ENCRYPTION_KEY", "hex");

function encryptAES(value) {
  const iv = crypto.randomBytes(16);
  const cipher = crypto.createCipheriv("aes-256-cbc", encryptionKey, iv);
  let encrypted = cipher.update(value, "utf8", "hex");
  encrypted += cipher.final("hex");
  return iv.toString("hex") + ":" + encrypted;
}

(async () => {
  const client = new MongoClient(uri);
  try {
    await client.connect();
    const db = client.db("ResilinkWithoutODEP");
    const userCol = db.collection("user");

    const existing = await userCol.findOne({ userName: "admin" });
    if (existing) {
      console.log("Admin user already exists. Skipping.");
```

```
      return;
    }

    const hashedPassword = await bcrypt.hash("123456", 10);
    const adminUser = {
      _id: new ObjectId().toString(),
      phoneNumber: "",
      userName: "admin",
      firstName: "admin",
      lastName: "admin",
      roleOfUser: "admin",
      email: encryptAES("admin@gmail.com"),
      password: hashedPassword,
      gps: "",
      createdAt: new Date().toISOString(),
      updatedAt: new Date().toISOString(),
      accessToken: ""
    };

    await userCol.insertOne(adminUser);
    console.log("Admin user created successfully.");
  } catch (err) {
    console.error("Error:", err);
  } finally {
    await client.close();
  }
})();
```

Run the script from the project root:

```
node createAdmin.js
```

Then delete the file once the admin user is created:

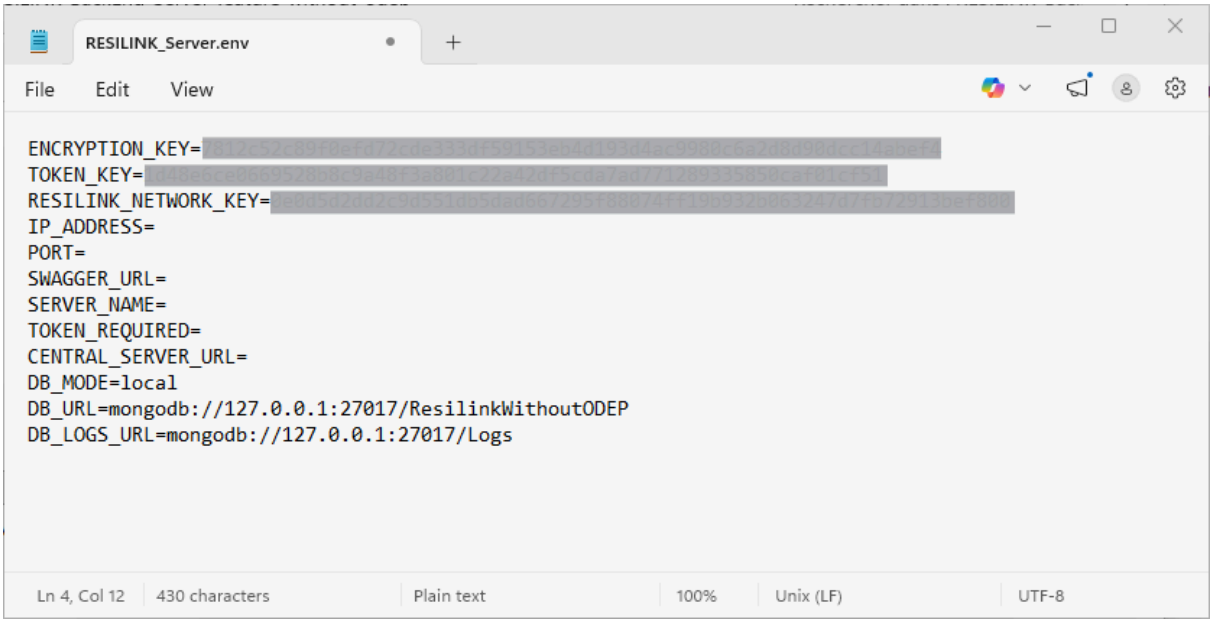
```
rm createAdmin.js
```

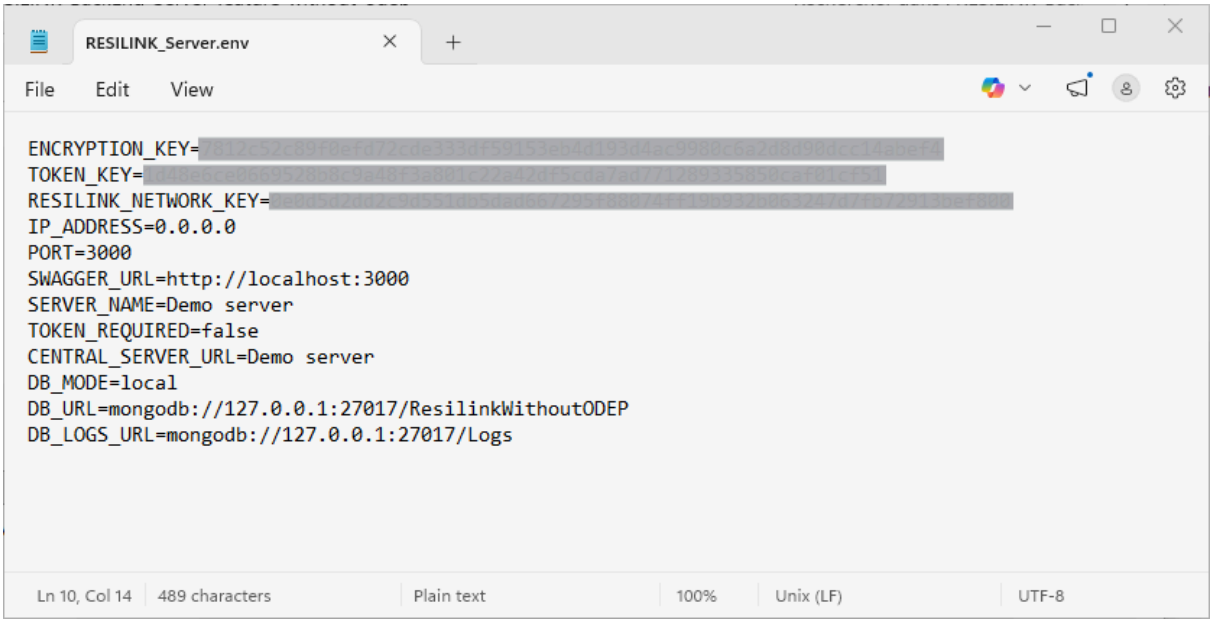
2.5. Configure the Environment

Edit the .env file in the project root to match your deployment setup:

Variable	Description
IP_ADDRESS	Server IP address (e.g. 0.0.0.0 to listen on all interfaces)
PORT	HTTP/S port the API listens on (e.g. 3000)
SWAGGER_URL	Base URL for Swagger UI (e.g. http://your-domain:3000)
SERVER_NAME	Display name for this server instance
TOKEN_REQUIRED	Set to true to enforce JWT authentication on GET endpoints
CENTRAL_SERVER_URL	URL of the central federation server (if federated)

DB_MODE	Database mode: local
DB_URL	MongoDB connection string for the main database
DB_LOGS_URL	MongoDB connection string for the logs database
ENCRYPTION_KEY	Auto-generated AES encryption key (do not modify)
TOKEN_KEY	Auto-generated JWT secret key (do not modify)
RESILINK_NETWORK_KEY	Auto-generated network key (do not modify)





2.6. Start the Server

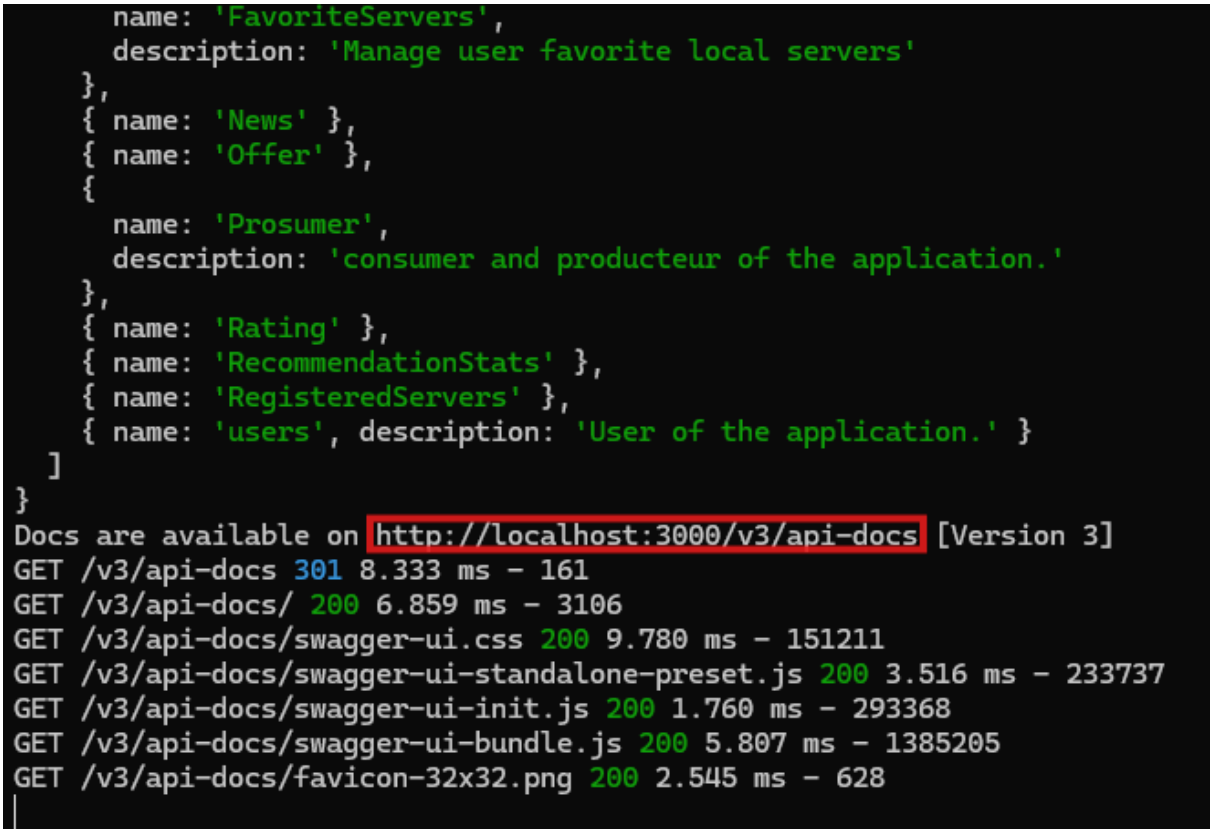
From the project root, run:

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

The server will start and print the Swagger documentation URL in the terminal.

2.7. Access the API

Use CTRL + click on the Swagger URL displayed in the terminal to open the interactive API documentation in your browser.



2.8. Deployment to a cloud platform

Deploying the RESILINK backend server to a cloud platform makes the API accessible from the public internet with few configurations. This section covers all prerequisites and the step-by-step configuration needed to expose the API securely over HTTPS using a valid domain name, Nginx as a reverse proxy, and a free SSL certificate from Let's Encrypt.

2.8.1. Prerequisites

Before starting, ensure the following are in place:

- **A cloud VPS** — Any Linux-based VPS will work (e.g. OVH, AWS EC2, DigitalOcean). The server must run Ubuntu 22.04 or later. The RESILINK backend installation procedure described in section 1 applies without modification.
- **A registered domain name** — A domain name purchased from any registrar (e.g. OVH Domains, Namecheap, Gandi). You will need access to the DNS zone management panel to create an A record pointing to your server's IP address.
- **An SSH client** — To connect and manage the remote server from your local machine.

2.8.2. Connect to the Server

From your local terminal, connect to the VPS via SSH:

```
ssh ubuntu@YOUR_VPS_IP
```

Replace **`ubuntu`** with the default user of your distribution and **`YOUR_VPS_IP`** with the public IP address assigned by your cloud provider.

2.8.3. DNS Configuration

In your domain registrar's DNS zone management panel, create an **A record** pointing a subdomain to your VPS IP address:

Type	Name	Value	TTL
A	api	YOUR_VPS_IP	300

This makes **`api.your-domain.com`** resolve to your server. DNS propagation can take a few minutes up to a few hours depending on the registrar.

2.8.4. Open the Required Ports

Access to the server must be allowed at two levels.

- **On the cloud provider dashboard** — In your provider's control panel (e.g. OVH Manager), navigate to your VPS network firewall settings and add inbound rules to allow TCP traffic on ports 22, 80, and 443.
- **On the server itself** — Run the following commands to configure UFW:

```
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
```

Port 3000 or the PORT selected by the developer does not need to be exposed publicly — all external traffic will go through Nginx on ports 80 and 443.

2.8.5. Install RESILINK Server

To deploy the server, please use the Linux script starting at section 2.1

2.8.6. Install Nginx

Nginx acts as a reverse proxy, receiving HTTPS requests on port 443 and forwarding them to the Node.js API running on port 3000:

```
sudo apt update
sudo apt install nginx
```

Create the Nginx configuration file for your domain:

```
sudo nano /etc/nginx/sites-available/resilink
```

Paste the following configuration, replacing **`api.your-domain.com`** with your actual subdomain and 3000 with your PORT:

```
server {
    listen 80;
    server_name api.your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Enable the configuration and reload Nginx:

```
sudo ln -s /etc/nginx/sites-available/resilink /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

2.8.7. Obtain an SSL Certificate

Install Certbot and the Nginx plugin:

```
sudo apt install certbot python3-certbot-nginx
```

Run Certbot for your subdomain. Certbot will automatically update your Nginx configuration to handle HTTPS:

```
sudo certbot --nginx -d api.your-domain.com
```

Follow the prompts. When asked about HTTP-to-HTTPS redirection, select option 2 (Redirect) to automatically redirect all HTTP traffic to HTTPS. The certificate is free and renews automatically every 90 days.

Verify that your .env file is properly configured with the settings for your cloud platform and reverse proxy:

```
IP_ADDRESS=0.0.0.0
PORT=YOUR_PORT
SWAGGER_URL=https://api.your-domain.com
```

2.8.8. Keep the Server Running with PM2

Without a process manager, the Node.js server stops as soon as the SSH session is closed. Install PM2 to keep the API running persistently:

```
npm install -g pm2
pm2 start src/index.js --name resilink
pm2 startup
pm2 save
```

The pm2 startup command generates a system command to run — execute it as instructed in the terminal output. From this point, the RESILINK API will restart automatically after any server reboot.

Acronyms List

Acronym	Explanation
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Android Package Kit
HTTP/HTTPS	Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure
JWT	JSON Web Token
SSL	Secure Sockets Layer

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.