6

SDKs and WEB-RI

6.1 Approach

The SDK global approach for i3-MARKET is based on the provision of four main pillars: (a) SDK-generator, (b) SDK-core, (c) SDK reference implementation or SDK-RI, and, finally, (d) Web-RI.

The layered SDK approach defined here is the mechanism that allows to adapt and extend existing data marketplaces to interface with the i3-MARKET Backplane.

Specifically, the layers that are part of the proposed solution for the SDK and shown in Figure 6.1 are the following:

- **SDK-core:** This layer aims to simplify the i3-MARKET SDK building process by generating client stubs for any i3-MARKET backend endpoint/API, defined with the OpenAPI (formerly known as Swagger) specification. In this way, therefore, the development team can better focus on the implementation and adoption of these backend endpoints or APIs.
- **SDK-reference implementation (SDK-RI):** This layer aims to identify and provide a set of common services to be implemented for consuming available Backplane functionalities.
- **SDK-execution patterns (SDK-EP):** It is including the atomic functions that make use of Backplane API (via SDK) adding some business logic.
- **Web-RI:** It is supporting the front-end or GUI integrating the common services provided by the SDK-RI and that can be reused and customized as part of the pilot specification and implementation defined in the context of WP5.

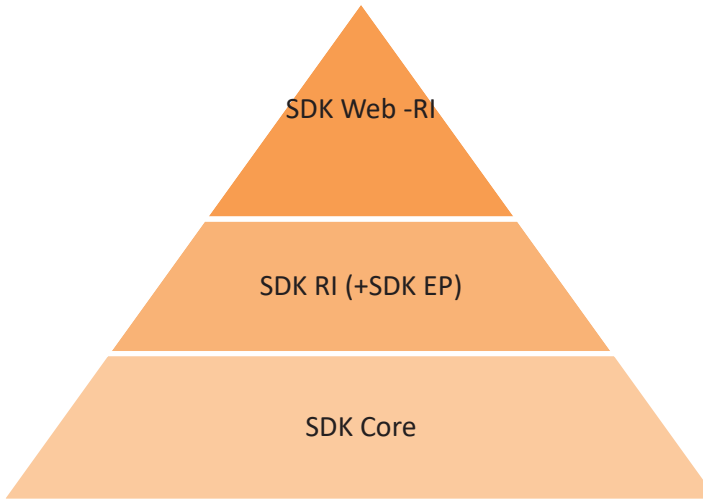


Figure 6.1 SDK layered approach.

6.2 SDK-Core Specification

General objectives:

The three main objectives identified are the following:

(a) Backplane API SDK
(b) Enhanced Backplane API SDK
(c) Automatically build Backplane API SDK

Considering the objectives, the following updates in terms of capabilities have been provided for the i3-MARKET FINAL release.

(a) Backplane API SDK. Addressing fully following modules:

- User-centric authentication SDK
- Cloud Wallet SDK module
- Data access SDK module
- Standard payments SDK module
- Tokenization SDK module
- Smart contracts SDK module
- Notifications SDK module
- Rating SDK module

(b) Enhanced Backplane API SDK
(c) Automatically build Backplane API SDK

Context:

The updated context in terms of interactions with other SW pieces in the i3-MARKET ecosystem is shown in Figure 6.2.

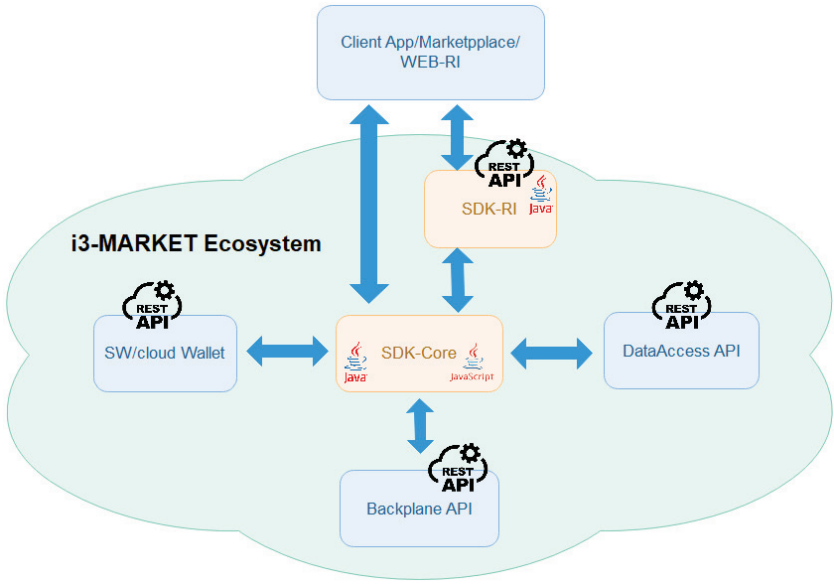


Figure 6.2 SDK-core interactions.

As a reminder, the i3-MARKET SDK-core interacts with:

i. Backplane API, allowing stakeholder's developers to create software (App Client) based on the (Backplane) API, in an easy and efficient way.
ii. Cloud Wallet to guarantee the security on the interactions between the stakeholders and i3-MARKET Backplane.
iii. App Client, allowing to be part of the i3-MARKET ecosystem.

Big picture:

The SDK-core is supported as a main pillar for the SDK-generator, which is one of the outcomes of i3-MARKET solutions.

The main updates on SDK-generator are the following:

(a) Update on the *openapi-generato*r client due to issues detected managing keywords *oneOf, anyOf*, and *allOf* in some of the OAS files supported by i3-MARKET backend services.

(b) Update on the *openapi-generator* setup. The concrete setup used in last version was: *openapi-generator-cli generate -g javascript –additional-properties=groupId={{ ARTIFACT_GROUP_ID }},artifactId={{ ARTIFACT_NAME }},artifactVersion={{ ARTIFACT_VERSION }}, modelPackage=com.i3m.model.data-access,apiPackage=com.i3m.api.data-acess, prependFormOrBodyParameters=true, hideGenerationTimestamp=true -o /tmp/oas/javascript -i http://xx.xx.x.xxx:yyyy/repository/i3m-raw/i3m-raw/files/dataaccessapi.json –generate-alias-as-model –skip-validate-spec"*

This is the same setup for SDK-core Java version but using "*java*" for the option "- g".

6.2.1 SDK-core implementation

As introduced, the SDK-core is built using SDK-generator REST API and an Ansible playbook in charge of generating all the client stub for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC, VC, and data access API encapsulated into the SDK-core Java/JavaScript library.

6.2.2 Core technology

The SDK-core implementation is based on the usage of SDK-generator, and it is described in detail in the following subsections.

The SDK-core is supported by means of (a) the SDK-generator REST API and (b) an Ansible playbook in charge of generating:

1) an SDK-core Java artifact that contains client stub for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC (OpenID Connect), VC (Verifiable Credentials), and data access API;
2) an SDK-core JavaScript artifact that contains client stub for Backplane API (semantic engine, notification manager, and smart contract manager), OIDC, VC, and data access API.

SDK-generator:

The SDK-generator is the main pillar of the SDK-core. The SDK-generator is based on SDK as a service approach. SDK-generator aims to automatically generate the client stubs needed to interact and consume all the functionalities

exposed in a REST API. The SDK as a service approach is shown in Figure 6.3.

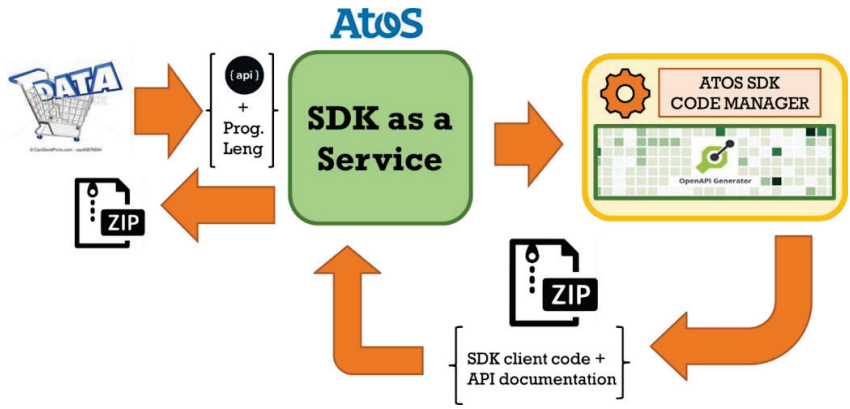


Figure 6.3 SDK-generator approach.

The workflow behind SDK-generator is based on the provision of a programming language specification next to an OAS file and making use of the OpenAPI generator[1] server, which is able to produce as output SDK client stubs next to associated documentation about how to use it.

The languages supported by the SDK-generator are shown in Figure 6.4.

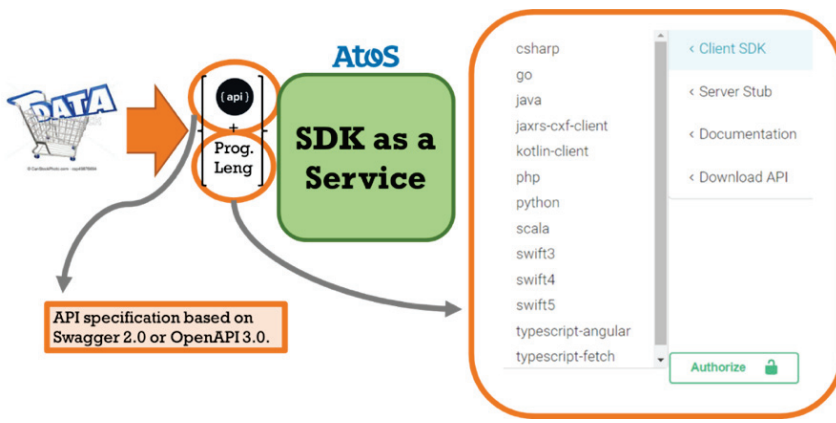


Figure 6.4 SDK-generator supported programming languages.

[1] OpenAPI generator: https://github.com/OpenAPITools/openapi-generator

Continuous integration and delivery:

The SDK-core artifact is automatically provided by means of a CI/CD pipeline based on Ansible AWX. A conceptual view of SDK-core pipeline is shown in Figure 6.5.

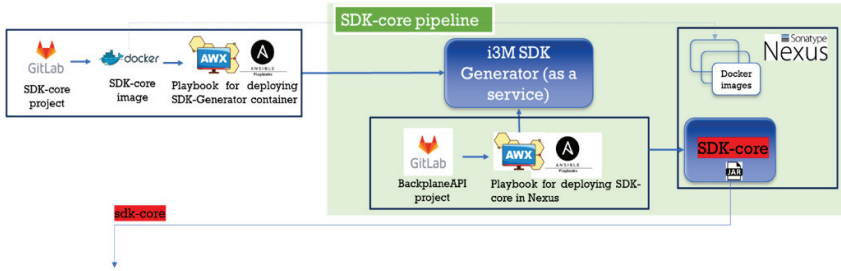


Figure 6.5 SDK-core CI/CD pipeline.

As initial step in the pipeline, the SDK-core artifact is triggering the compilation and deployment of a new version of the SDK-generator once a commit into master branch of SDK-generator project happens. As a second step (represented as a green area in Figure 6.6), the generation and publishing of a new version of the SDK-core artifact is triggering each time a new version of the Backplane API is deployed. The CI/CD behind Backplane API includes a triggering to SDK-core pipeline. In this way, SDK-core covers a set of tasks mainly in charge of generating SDK-core artifacts for Java and JavaScript

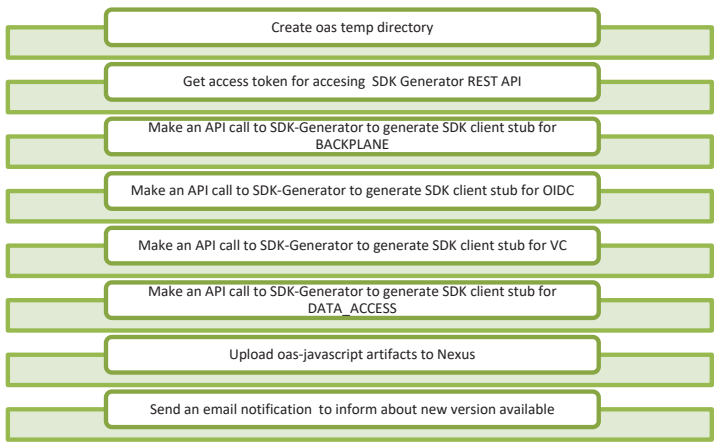


Figure 6.6 SDK-core playbook internal workflow.

versions taking a set of relevant OAS files associated with the following artifacts:

- Backplane API (including semantic engine, notification manager, and smart contract manager)
- OIDC API
- Verifiable Credentials API
- Data access API

Concretely, the Ansible playbook is used to automatize the process of generation of the SDK-core client stub.

The internal workflow covered by the SDK-core playbook is shown in Figure 6.6.

Finally, the pipeline includes a couple of tasks in charge of publishing the generated Java and JavaScript versions of SDK-core into i3-MARKET Nexus repository.

SDK-core installation:

SDK-core is a Java/JavaScript library that is installed by simply importing from i3-MARKET Nexus repository.

6.3 SDK Reference Implementation (SDK-RI)

The SDK-RI implementation is based on Java and Swagger framework, and the following subsections are focusing on the SDK-RI specifications. SDK-RI is a web app deployed within Jetty and encapsulated in a Docker container.

The SDK-RI has been updated in terms of common services as per the following (see Figure 6.7):

i) Notification manager common services: The functionalities related with notification services and queues are listed in Figure 6.7.

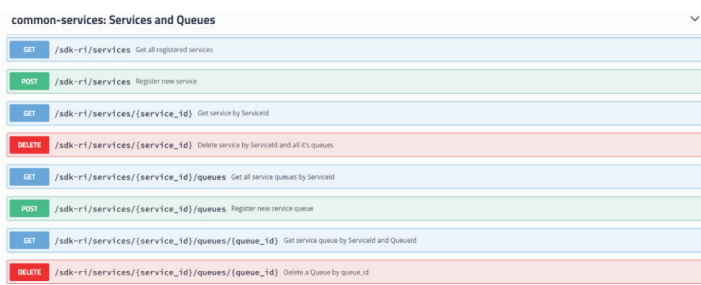


Figure 6.7 Services and queues common services.

ii) Alerts common services: The functionalities related with alerts are listed in Figure 6.8.

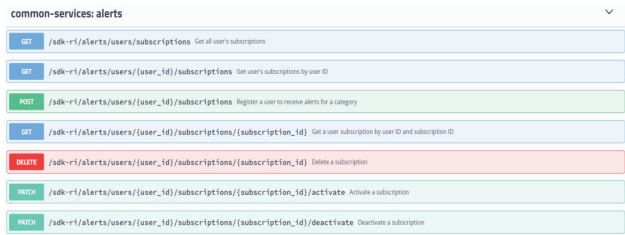


Figure 6.8 Alerts common services.

iii) Conflict resolution common services: This is listed in Figure 6.9.

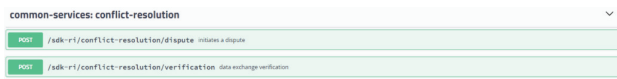


Figure 6.9 Conflict resolution common services.

iv) Contracts common services: The functionalities related with smart contracts management are listed in Figure 6.10.

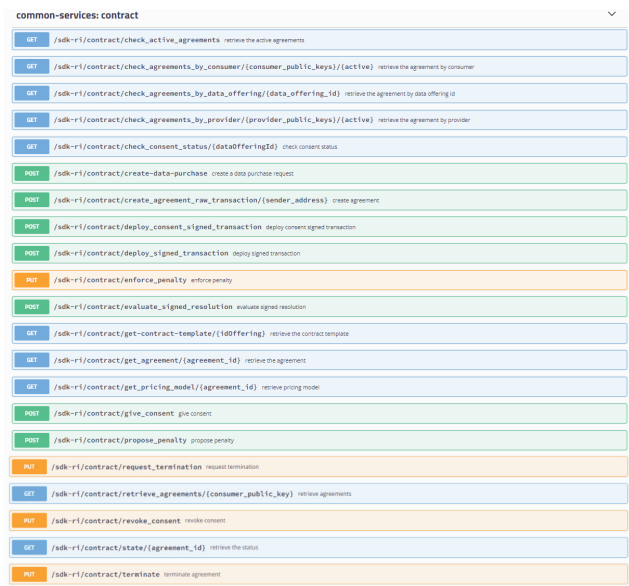


Figure 6.10 Contracts common services.

v) Credential common services: The functionalities related with authentication, identities, and credentials are listed in Figure 6.11.

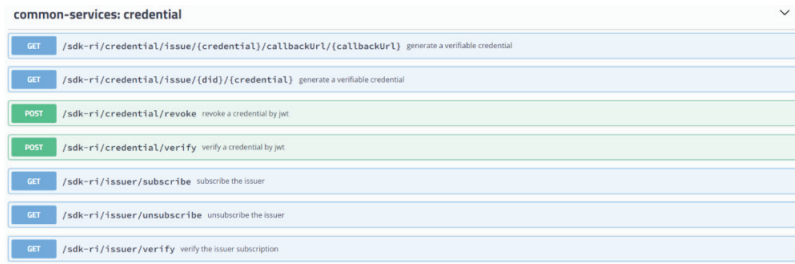


Figure 6.11 Credentials common services.

vi) Exchange common services: The functionalities related with data exchange are listed in Figure 6.12.

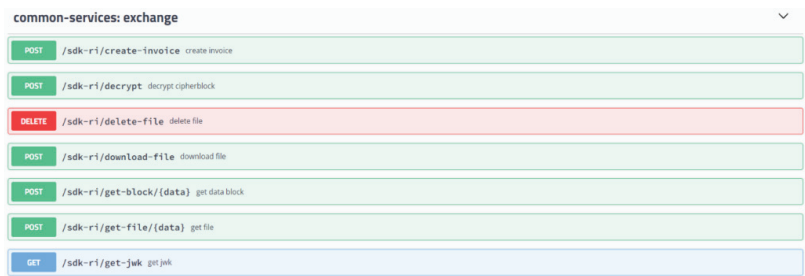


Figure 6.12 Exchange common services.

vii) Notification common services: The functionalities related with notifications are listed in Figure 6.13.

Figure 6.13 Notification common services.

viii) Offering management common services: The functionalities related with data offering management are listed in Figure 6.14.

Figure 6.14 Offering common services.

ix) Pricing managing common services: The functionalities related with pricing managing are listed in Figure 6.15.

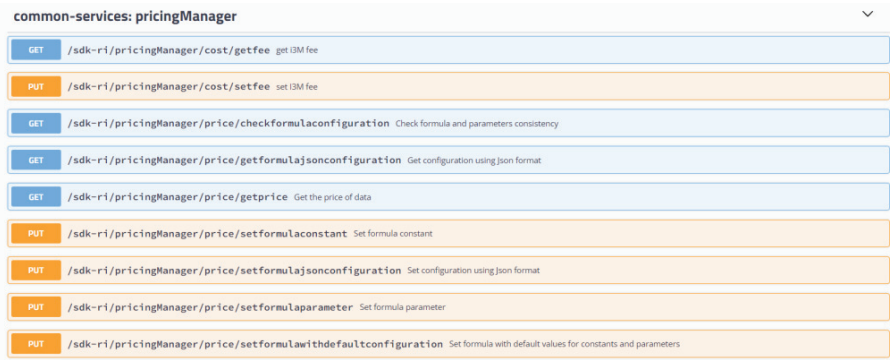


Figure 6.15 Pricing common services.

x) Token managing common services: The functionalities related with token management are listed in Figure 6.16.

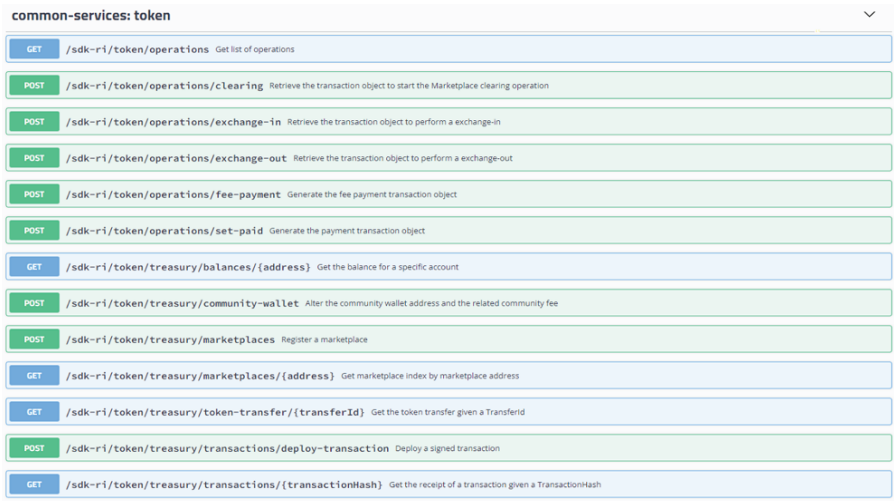


Figure 6.16 Token common services.

As an initial stage, the SDK-RI imports the last version of the SDK-core published in i3-MARKET Nexus maven repository as a library. It is precisely in this part where the way to generate the Java version of the imported SDK-core library has been slightly updated. As a second stage, once a commit is done into master branch of SDK-RI Git project, a compilation and deployment of a new version is automatically launched.

6.4 WEB-RI

The Web-RI is a GUI web interface that allows the users to interact with the functionalities provided by i3-MARKET Backplane solutions on top of the SDK-RI. It can be reused and customized as part of each pilot specification and deployment integration as a reference implementation of the backbone data marketplace to facilitate stakeholder needs that want to reuse i3-MARKET artifacts and functionalities.

6.4.1 Purpose

The WEB-RI proposes itself as a reference for the implementation of a user interface to allow human users to use and interact with the functionalities provided by i3-MARKET. The WEB-RI has three main objectives, which are:

- As a management tool, to allow i3-MARKET developers to test their functionalities in the context of a user usage.
- As a marketing team, allowing the promotion and demonstration of i3-MARKET functionalities using a generic approach and language that can be easily translated to the available data marketplaces used by different domains.
- As a reference implementation, providing functional examples of how the i3-MARKET SDKs can be used to implement/integrate i3-MARKET functionalities into a data marketplace. As a reference implementation, WEB-RI is also a useful tool to help i3-MARKET pilots on the implementation of their use-case scenarios and on testing of Backplane technologies by providing specifications and code that can be used.

In Figure 6.17, the architecture of WEB-RI is represented.

A consumer or a provider can access WEB-RI[2] via internet browser and proceed with the authentication for which the wallet[3] must be installed and running on his personal computer. The authentication process is executed on WEB-RI frontend by calling the OIDC service, which will call the wallet to perform the authentication itself.

The WEB-RI frontend is connected to a backend, which has two main functions: manage user sessions and have a way to interact with the functionalities provided by i3-MARKET.

[2] https://gitlab.com/i3-MARKET-V3-public-repository/i3-MARKET-web-ri
[3] https://gitlab.com/i3-MARKET-V3-public-repository/sp3-scgbssw-i3mwalletmonorepo

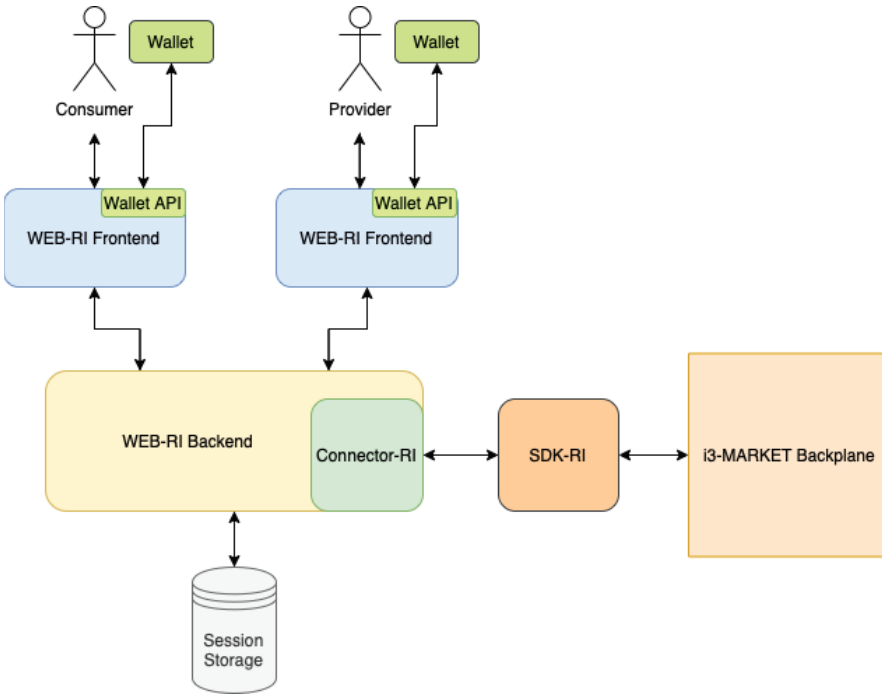


Figure 6.17 WEB-RI architecture.

To manage the user sessions, the WEB-RI backend saves the user session in a session storage called connect-mongo[4].

To interact with the functionalities provided by i3-MARKET, a library was implemented, called Connector-RI[5]. This connector has all the methods needed to call the respective APIs from the SDK-RI, which have the functionalities to interact with the i3-MARKET Backplane. This allows to have a clean and simple WEB-RI backend where it is only needed to call the respective methods from the connector.

Sitemap:

In Figure 6.18, the sitemap of WEB-RI is represented.

WEB-RI is composed of several pages, which are Authentication, Homepage, Offerings, Search, and Notifications.

[4] https://github.com/jdesboeufs/connect-mongo
[5] https://gitlab.com/i3-MARKET-V3-public-repository/i3-MARKET-connector-ri

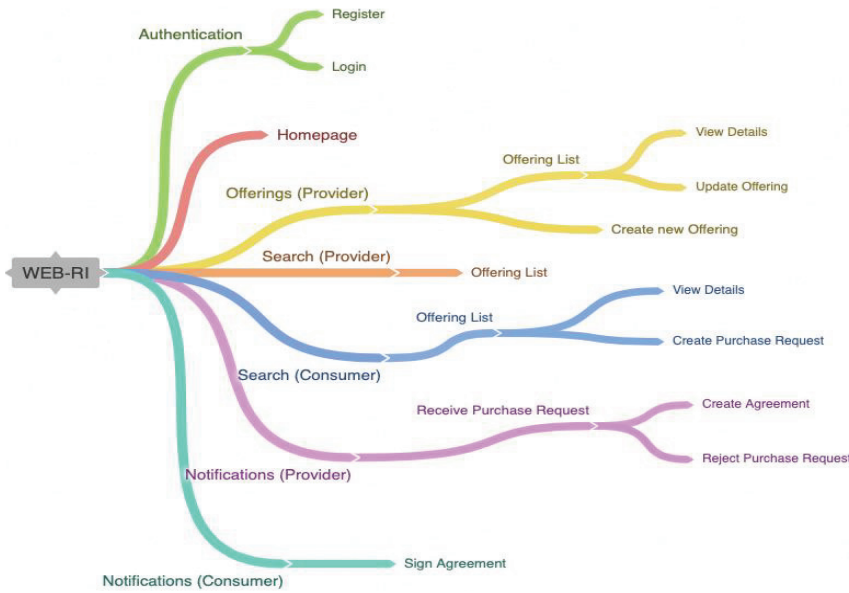


Figure 6.18 WEB-RI sitemap.

In the Authentication page, the user has the possibility to register a new provider or consumer and log in with some existing user registered in WEB-RI.

The Homepage is the main page of WEB-RI, which has a navigation bar that allows the user to navigate to the other available pages. Also, there are statistics related with the number of offerings and providers.

The Offerings page is only visible to a provider, where he can manage the offerings registered by him and register new ones.

The Search page is visible either to a provider or a consumer. The only difference is that a consumer has the possibility to create a purchase request for the offering he searched.

In the Notifications page, a provider can receive a purchase request for some of its offerings and he can accept (and create the agreement) or reject it. A consumer can sign the agreement if it was accepted before by the provider.

6.5 IMPLEMENTATION

In the following subsections, some screenshots of each page are presented, and an explanation of its content is given.

Register:

Figure 6.19 shows the WEB-RI register page.

i3-Market has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 871754

Figure 6.19 WEB-RI registration page.

Before the WEB-RI page is opened for the first time, the user must have the wallet running on his personal computer. When the user opens the WEB-RI initial page, he will see the page for registering a new user. He must select the desired role (consumer or provider) and username – Figure 6.20.

After that, the user must confirm the addition of the new user in the wallet; see Figure 6.20.

Login:

Figure 6.21 shows the WEB-RI login page.

With a user is registered in the wallet, it is possible to authenticate in WEB-RI. The user must select the role (consumer or provider) he wants to use to login in the system. After having selected the role in the login page, the user must confirm the authentication in the wallet; see Figure 6.22.

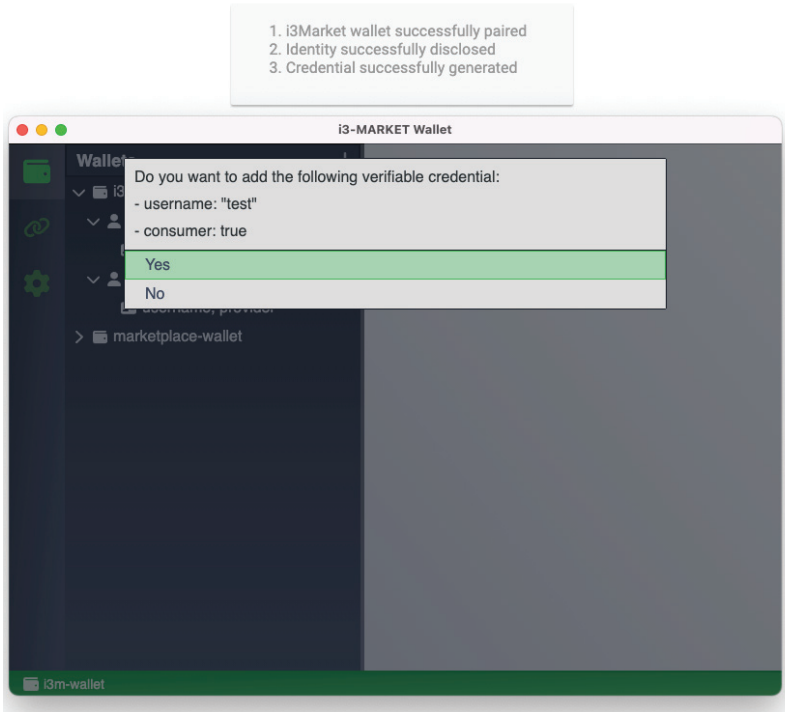
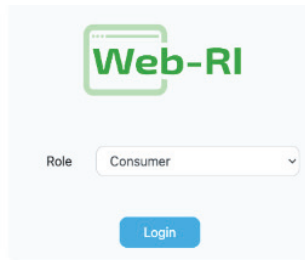


Figure 6.20 WEB-RI register with wallet.



i3-Market has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 871754

Figure 6.21 WEB-RI login page.

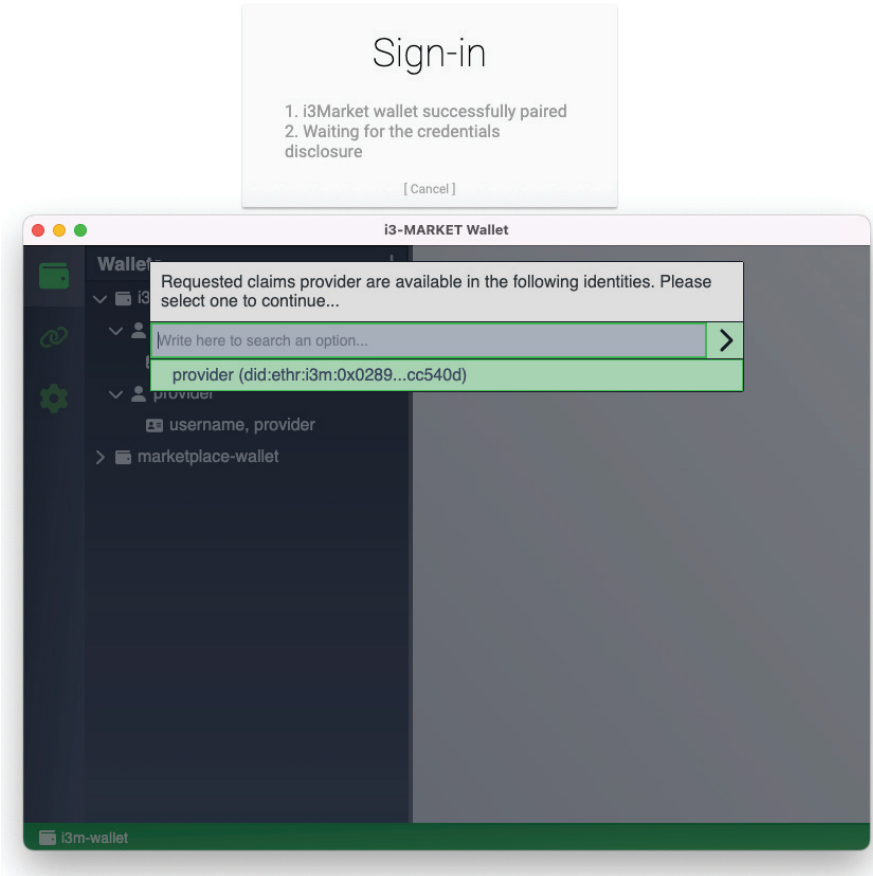


Figure 6.22 WEB-RI login with wallet.

6.6 Navigation:

With successful login, the user accesses the WEB-RI homepage. This page has a navigation bar, which is different to each role. The provider has access to offerings, search, and notifications pages and account options; instead, the consumer has access to same pages but not to the offerings page.

In Figure 6.23, the navigation bar for a provider is presented.

Figure 6.23 WEB-RI navigation (provider).

Figure 6.24 presents the navigation bar for a consumer.

Figure 6.24 WEB-RI navigation (consumer).

Homepage:

In Figure 6.25, the WEB-RI home page is presented.

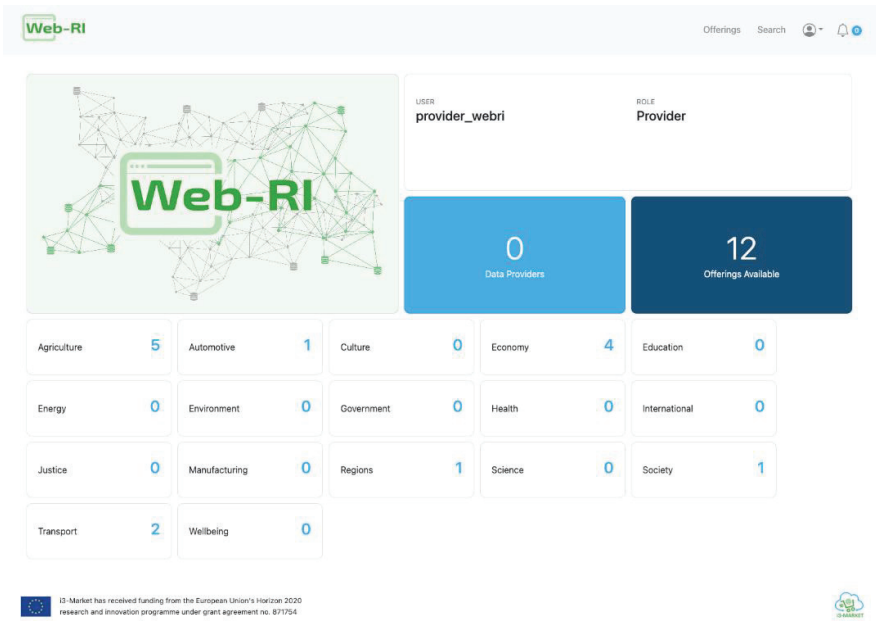


Figure 6.25 WEB-RI home page.

Besides the navigation bar, the WEB-RI home page has also the information about the logo and details about the user logged-in (username and role).

As main information, WEB-RI also shows the total number of providers and active offerings available in the whole marketplace ecosystem. Also, it is possible to see the total number of active offerings filtered by each category.

Offerings:

As mentioned before, the provider has access to the offerings page. The next subsections will describe each page related to the offerings.

Offering list:

Figure 6.26 shows the page with the list of offerings of a provider.

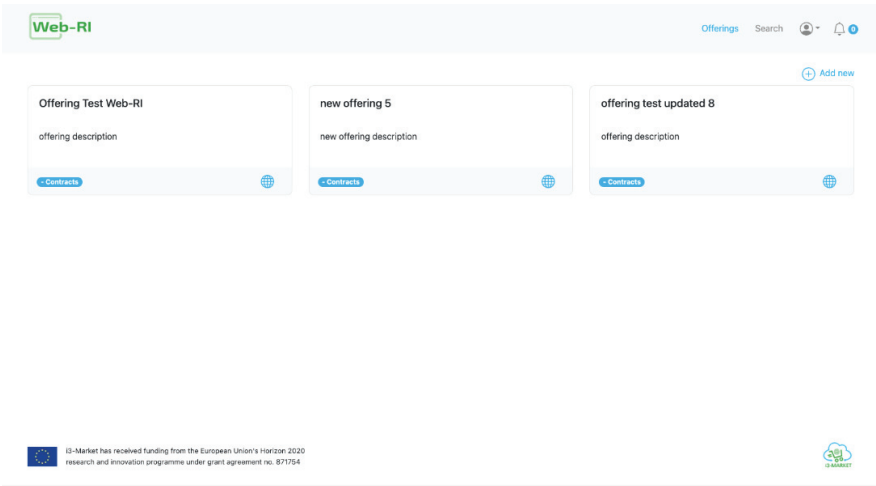


Figure 6.26 WEB-RI offerings page.

In this page, the provider sees the list of the offerings that were registered by him. Each offering is displayed in a react-bootstrap card[6] with some information like title, description, number of contracts, and state (active, inactive, to be deleted, or deleted).

Also, the provider has the option to register a new offering, which will be described in the following sections.

Offering details:

Figure 6.27 represents the page with the details of an offering.

[6] https://react-bootstrap.github.io/components/cards/

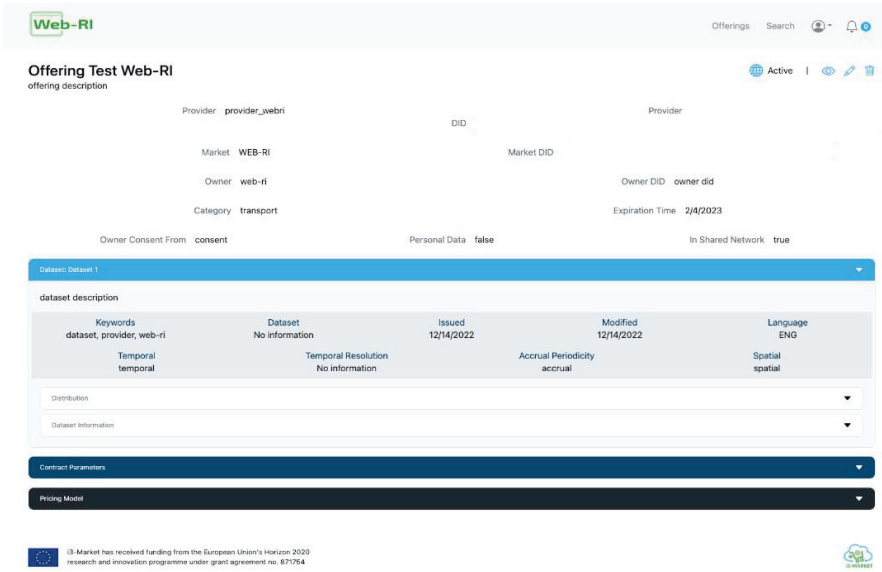


Figure 6.27 WEB-RI offering details page.

When a specific offering card is selected, it will open a new page with the details of the offering. Here, a user can see all the information related with that offering.

Since there is too much information to be displayed in a single page, a react-bootstrap accordion[7] was used to display information like dataset, contract parameters, and pricing model. This information is collapsed by default but can be expanded as well.

This page can be seen by a provider (through offerings page) or consumer (with search). If the user is a provider, he has options to activate, update, or delete the offering (in the top right corner of the site, next to the offering state). Instead, if he is a consumer, he has a button called "Buy Offering", which allows to initiate the process of creating a data purchase request.

Offering registration:

Figure 6.28 represents the page to register a new offering or update an existing one.

[7] https://react-bootstrap.github.io/components/accordion/

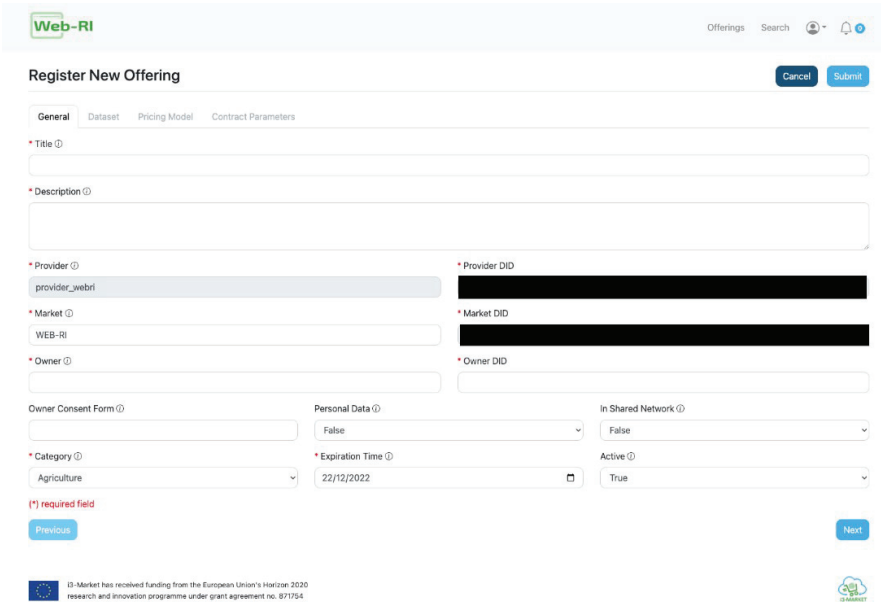


Figure 6.28 WEB-RI offering registration page.

The provider can register a new offering or update an existing one (but only the offerings registered by him). This page shown in Figure 6.29 is used for both purposes; the only difference is, when updating an offering, all the fields are already filled.

Since there is a lot of information associated with an offering, a react-bootstrap tab[8] was used on this page. With the help of the tabs, all fields were grouped by categories, which are general, dataset, pricing model, and contract parameters.

Also, inside each tab, some accordions were used to better display all the input fields to the user.

Offering purchase request:

Figure 6.29 represents the page where a consumer can initiate the process of buying a new offering.

[8] https://react-bootstrap.github.io/components/tabs/

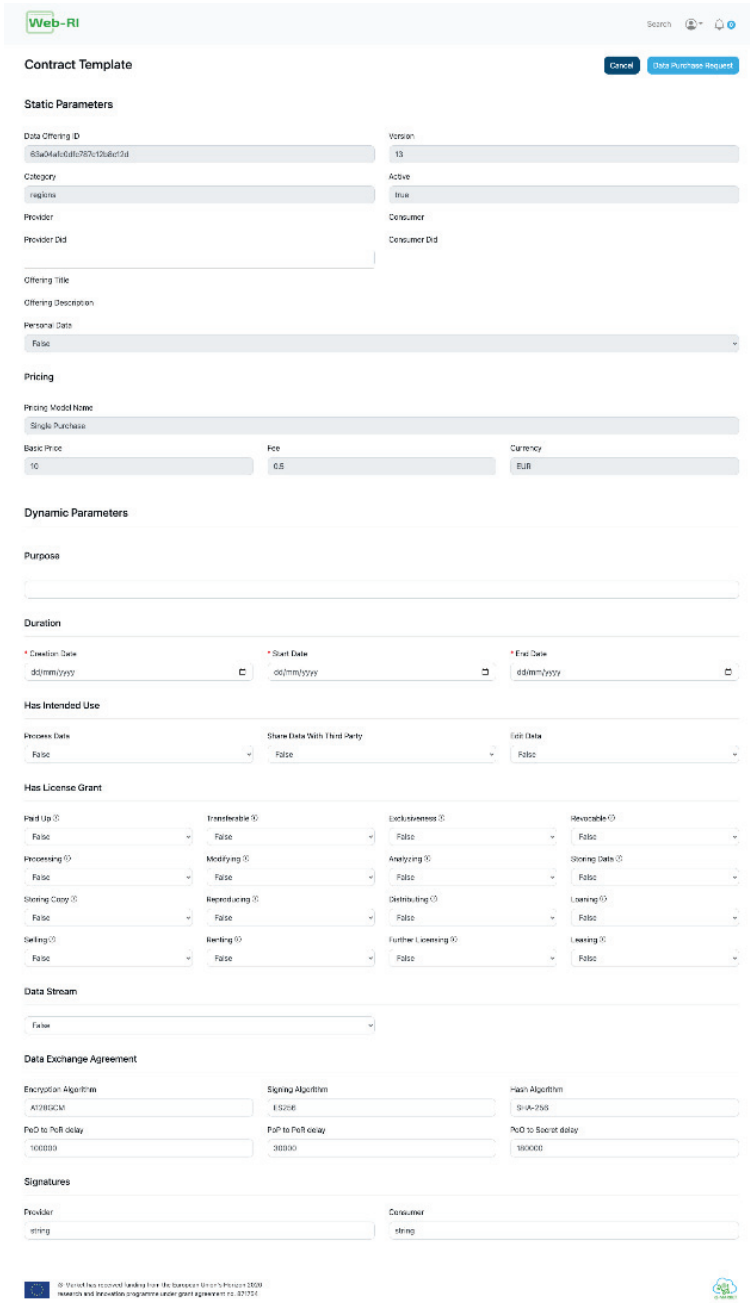


Figure 6.29 WEB-RI offering purchase page.

After the consumer selects the "Buy Offering" button in offering details page, a new page will be displayed with the contract template for that offering. In this page, the consumer must fill in the dynamic parameters of the template and then click on the "Data Purchase Request" button to proceed with the process of buying an offering.

Search:

Figure 6.30 represents the page where a user (provider or consumer) can search for offerings.

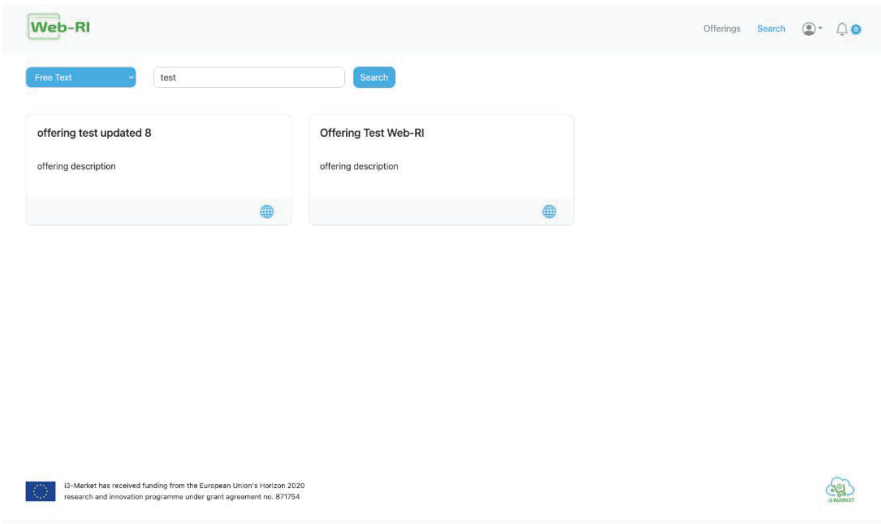


Figure 6.30 WEB-RI search page.

In the search page, the user (consumer or provider) can search for active offerings available in the whole marketplace ecosystem. He can search offerings by category, provider, or free text. As mentioned in the image above, the search is executed by entering a free text and returns the offerings that match the search criteria.

Notifications:

Figure 6.31 represents the page where a user can see his notifications.

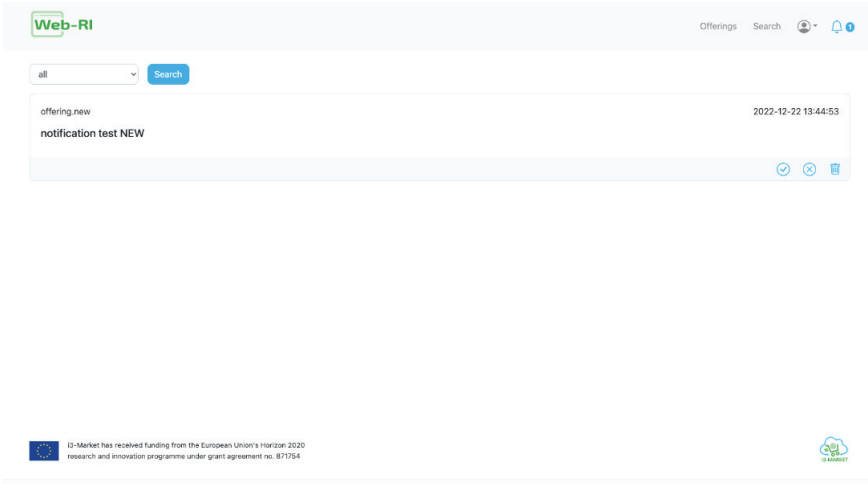


Figure 6.31 WEB-RI notifications page.

This page has all notifications associated with the user who is logged-in in WEB-RI.

If the provider is logged-in, he can receive notifications about a purchase request regarding some of his offerings. In this case, if he accepts the proposal, a new page will be displayed where the provider can create a new agreement. But he also can reject the proposal by sending some comments justifying the rejection of the proposal (this will be sent as a notification to the respective consumer).

If the consumer is logged-in, he can receive notifications about data purchase requests that were rejected by the provider or about proposals that were accepted and then he must sign the agreement.

Account:

This option, represented by a person icon in navigation bar, shows some options in a dropdown. One of those options allows the user to log off from WEB-RI.