A new concept of public administration based on citizen co-created mobile urban services

Grant Agreement: 645845

D3.3 – GUIDELINES FOR THE INTEGRATION AND POPULATION OF THE WELIVE ENVIRONMENT V1

DOC. REFERENCE: WeLive-WP3-D33-REP-160930-v10
RESPONSIBLE: FBK
AUTHOR(S): TECNALIA, UDEUSTO, EUROHELP, FBK, ENG, LAUREA and DNET
DATE OF ISSUE: 30/09/16
STATUS: FINAL
DISSEMINATION LEVEL: PUBLIC

<table>
<thead>
<tr>
<th>VERSION</th>
<th>DATE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1</td>
<td>16/03/2016</td>
<td>First version of the document including the ToC and distribution of tasks</td>
</tr>
<tr>
<td>v0.2</td>
<td>17/06/2016</td>
<td>Added some contributions from TECNALIA and UDEUSTO</td>
</tr>
<tr>
<td>v0.3</td>
<td>28/06/2016</td>
<td>Added more contributions from EUROHELP and LAUREA</td>
</tr>
<tr>
<td>v0.4</td>
<td>30/06/2016</td>
<td>Added more contributions from DNET and TECNALIA</td>
</tr>
<tr>
<td>v0.5</td>
<td>18/07/2016</td>
<td>Last contributions; final internal review made by FBK</td>
</tr>
<tr>
<td>v0.6</td>
<td>18/07/2016</td>
<td>Minor updates before the external review process</td>
</tr>
<tr>
<td>v1.0</td>
<td>30/09/2016</td>
<td>Reviewers´ comments processed and accepted version submission.</td>
</tr>
</tbody>
</table>
INDEX

1. EXECUTIVE SUMMARY ................................................................. 4
2. INTRODUCTION .............................................................................. 6
3. DATASETS POPULATION .................................................................. 7
   3.1. TECHNICAL GUIDE................................................................. 7
   3.1.1. Creating a new dataset using interface provided by the Open Data Stack ...................................................... 7
   3.1.2. Harvesting datasets from existing Open Data portals ....................................................................................... 8
   3.1.3. Creating new datasets using the WeLive REST API ......................................................................................... 9
   3.2. METHODOLOGICAL GUIDELINES ....................................... 10
   3.3. BEST PRACTISES .................................................................... 11
4. BUILDING BLOCKS POPULATION ................................................... 12
   4.1. TECHNICAL GUIDE................................................................. 12
   4.2. METHODOLOGICAL GUIDELINES ....................................... 13
   4.3. BEST PRACTISES .................................................................... 14
5. PUBLIC SERVICES POPULATION .................................................... 15
   5.1. TECHNICAL GUIDE................................................................. 15
   5.2. METHODOLOGICAL GUIDELINES ....................................... 16
   5.3. BEST PRACTISES .................................................................... 16
6. HOSTING SOFTWARE ARTIFACTS .................................................. 17
   6.1. CHOOSING THE HOSTING ENVIRONMENT ............................. 17
   6.2. TECHNICAL GUIDE FOR CLOUDFOUNDRY HOSTING ENVIRONMENT ................................................................. 17
   6.2.1. Using CloudFoundry ................................................................................................................................. 17
   6.2.2. Deploying app to CloudFoundry .......................................... 18
   6.2.3. Using CloudFoundry services ........................................... 18
   6.2.4. Using CloudFoundry Eclipse plugin ................................. 20
   6.3. TECHNICAL GUIDE FOR CNS HOSTING ENVIRONMENT ....... 20
   6.3.1. Using Building Blocks hosted in the CNS Marketplace ....... 20
   6.3.2. Becoming a CNS Marketplace user or provider ............... 20
   6.3.3. Publishing existing data refining services as Building Blocks ........................................................................ 21
   6.3.3.1. Obtaining access to a data refining solution ................. 21
   6.3.3.2. Submitting a new Building Block ................................... 21
   6.3.3.3. Publishing a BB in the WeLive Marketplace ................. 22
   6.3.4. Implementing new data refining solutions ....................... 22
   6.3.4.1. Integration Principles .......................................................... 22
   6.3.4.2. Cloud’N’Sci Refinery Protocol .............................................. 22
   6.3.4.3. Refinery Toolkits ................................................................. 23
   6.3.4.4. Docker .............................................................................. 23
   6.3.5. Publishing data refining solutions .................................... 23
   6.3.5.1. Data Interfaces ................................................................. 23
   6.3.5.2. Function Interfaces .......................................................... 24
   6.3.5.3. Submitting Solutions ....................................................... 24
   6.3.5.4. Describing Algorithms .................................................... 24
   6.3.5.5. Building Images ............................................................... 24
   6.3.5.6. Making Solution Releases ............................................... 24
   6.3.5.7. Launching Sales Campaigns ........................................... 25
   6.3.6. Protected Resources .......................................................... 25
   6.4. METHODOLOGICAL GUIDELINES .................................... 26
   6.5. BEST PRACTISES .................................................................... 26
7. OPEN INNOVATION AREA POPULATION ........................................................................................................... 28

7.1. TECHNICAL GUIDE .................................................................................................................................................. 28

7.1.1. Creating an idea ..................................................................................................................................................... 28

7.1.2. Creating a need report .......................................................................................................................................... 29

7.1.3. Creating a challenge ............................................................................................................................................. 30

7.2. METHODOLOGICAL GUIDELINES ..................................................................................................................... 30

7.3. BEST PRACTICES .................................................................................................................................................... 31

8. DECISION ENGINE POPULATION .......................................................................................................................... 32

8.1. TECHNICAL GUIDE ................................................................................................................................................ 32

8.2. METHODOLOGICAL GUIDELINES ....................................................................................................................... 34

8.3. BEST PRACTICES .................................................................................................................................................. 34

9. CONCLUSIONS ......................................................................................................................................................... 35

10. ABBREVIATIONS .................................................................................................................................................. 36

11. REFERENCES ........................................................................................................................................................ 37

12. COMMENTS FROM EXTERNAL REVIEWERS ........................................................................................................ 38

12.1. UDEUSTO ............................................................................................................................................................ 38

12.1. EUROHELP ......................................................................................................................................................... 39

13. ANNEX I – ETHICAL COMPLIANCE CHECK ...................................................................................................... 40

ILLUSTRATIONS

Figure 1 – Button for starting the dataset creation process. .......................................................................................... 7
Figure 2 – Dataset creation form. .................................................................................................................................. 7
Figure 3 – Resource creation form. .................................................................................................................................. 8
Figure 4 – Harvester creation form. ............................................................................................................................... 9
Figure 5 – Terms of Use agreement when creating a new dataset. .................................................................................. 10
Figure 6 – Starting the building block publication wizard. ........................................................................................... 12
Figure 7 – Building Block publication wizard. ............................................................................................................ 13
Figure 8 – Publication of a Public Service Application as a new artefact. .................................................................... 15
Figure 9 – New Public Service Application form. ....................................................................................................... 15
Figure 10 – Adding a logo to a Public Service Application. .......................................................................................... 16
Figure 11 – Targetting the CloudFoundry endpoint. ................................................................................................... 17
Figure 12 – Cloudfoundry login. .................................................................................................................................. 18
Figure 13 – Packaging a mavenized Java project. ......................................................................................................... 18
Figure 14 – Pushing to CloudFoundry an artifact with a deployment descriptor. ......................................................... 18
Figure 15 – Pushing to CloudFoundry an artifact by referencing it. ............................................................................ 18
Figure 16 – List of services available on CloudFoundry marketplace. ........................................................................ 19
Figure 17 – Creation of service instance on CloudFoundry. ......................................................................................... 19
Figure 18 – Bind a service to an applicatin in CloudFoundry. ...................................................................................... 19
Figure 19 – OIA main page (for a Citizen user). .......................................................................................................... 28
Figure 20 – OIA idea creation form. ............................................................................................................................ 29
Figure 21 – OIA need report creation form. ................................................................................................................ 29
Figure 22 – OIA challenge creation form ................................................................................................................... 30

TABLES

No table of figures entries found.
1. EXECUTIVE SUMMARY

The 2010 edition of the EU eGovernment Benchmark Report states that currently public services are built following an administration-centric approach, driving to a low usage, rather than according to the citizens’ need (user-centric approach). Public administrations are facing key socioeconomic challenges such as demographic change, employment, mobility, security, environment and many others. Besides, citizens expectations, in terms of burden reduction, efficiency, and personalization, are growing and will make the take-up of traditional public e-services steadily harder in the following years. Citizens want to transit from being mere consumers of public services to providers of those services, i.e. prosumers of the open government ecosystem. Public-private partnerships and active contribution of citizens are two key instruments to transform the way currently cities and territories are being governed. To turn cities and territories into hubs of welfare, innovation and economic growth (i.e. to give place to Smarter Cities or Territories) not only they have to make a more efficient management of resources but they also have to be aware and reactive to the socio-economic needs and wants of their stakeholders, i.e. their citizens, local businesses and companies. ICT-enabled Open and Collaborative Government is the recipe to deliver "more from less". Indeed, governments cannot be any longer the single providers of public services. Empowerment of stakeholders is necessary by incentivizing them to take a more active role. Public-private partnerships have to be catalysed to give place to a more sustainable model of government which also behaves as a economy promotion dynamizer.

The WeLive project was born as a means to address the above challenges. WeLive aims at transforming the current e-government approach followed by most public administrations into we-government where all the stakeholders of public administration, namely citizens, local businesses and companies, are treated as peers (collaborators) and prosumers (providers) instead of the usual customer role associated to them. WeLive will enable also the so-called “t-Government” (Transformational Government) by providing stakeholders with the technology tools that enable them to create public value. In addition, WeLive is also thought to embrace l-Government (Lean Government), which aims to do more with less by involving other players, leaving the Government as an orchestrator around enabled platforms. Finally, WeLive fully adopts m-Government, i.e. an extension or evolution of e-government through utilization of mobile technologies for public service delivery. Consequently, WeLive proposes a new concept of e-Government, which provides the means, i.e. an environment or platform, analogously to the Web, and leaves others, all the stakeholders in a city or territory, to lead the innovation process and so turn public resource assets into artifacts to nurture economic growth and job creation.

An important stage of the WeLive process that follows the setup of the WeLive framework and precedes its usage in the trials of the four pilot cities/regions is the initial population of the environment. This activity consists of exposing onto the framework a number of artifacts that is sufficient to trigger the innovation process for generating higher value services and applications.

This deliverable will provide the details for such injection of artefacts into the WeLive framework components. It will consider both the technological and the methodological levels, providing precise instructions about how to upload different kind of objects into the WeLive tools – the WeLive Marketplace in particular, but not only –along with more high level considerations about how to perform this activity. Such considerations are a priori suggestions about how to operate with WeLive components that stem directly from their intended role and functionality. Taking advantage of the fact that the population activities have already started, this deliverables will also report about some preliminary post hoc considerations about the actual population with real objects. The collection of these best practises is only an initial step of the more thorough assessment that will be completed in the next project phases. We remind here that the actual population of the framework by the four trials task forces will be described in the deliverable D3.5.
Within this document, we concentrate first on the population of the WeLive framework with the so-called WeLive artifacts, that is to say, datasets, building blocks and public service applications. Not all of such objects are stored or hosted within the WeLive framework. Some of them can reside elsewhere. This document takes into account this occurrence describing the alternatives and how the objects, anyhow, are exposed onto the framework. Then, we also consider the filling of the WeLive components with other kind of objects, that is to say, needs, challenges, ideas, and recommendation rules, which complete the scenario about the population stage of the WeLive innovation process.
2. INTRODUCTION

The purpose of this deliverable is to present the outcomes of Task 3.2, namely of the elicitation of the guidelines for the population of the WeLive framework. In this first version of the deliverable, we will describe the results of the activities within this task related to the first integrated version of the WeLive framework that will be used during the first pilot phase. A second version of this deliverable will be produced before the start of the second pilot phase.

In the deliverable D3.1 – Methodologies and validation of the integration of the WeLive environment v1 [1], we described how to achieve a successful integration of the WeLive components into the WeLive framework and how to setup an executable instance of such framework that will serve the execution of the first phase trials. Following the plan for the realization of the innovation process proposed by the WeLive project, this deliverable represents the logical continuation of D3.1 since it described how to populate the frameworks instance described there. Conceptually, it is the trait d’union towards the deliverable D3.5 – Trento, Bilbao, Finish Region and Novi Sad environment v1 [2] that will describe the actual population of the frameworks. It represents also a fundamental milestone enabling the execution of WP4 activities.

The rationale that motivates the need for the guidelines described in this document consist of the simple observation that trying to engage the stakeholder of the WeLive process (citizens, public administrations businesses) using an empty framework would easily result into a failure. Providing a partially filled environment fulfills actually two aims. First, it supplies samples that helps clarifying the concepts on which the WeLive approach is based, namely need, challenge idea, dataset, building block, public service application. Then it also provide concrete bits that can used since the beginning of the pilot execution to build those added value services and applications that constitute the ultimate goals of the WeLive process. Thus, it is important to fill the framework instance with real artefacts. Furthermore, it is important, as well, to make explicit those guidelines that drive such filling both to clarify the usage of the framework itself and to document the replicability of the approach followed. In this sense, the deliverable addresses different aspects of the population activity. First, the technical ones that enables to use properly the available population tools. Then the methodological ones that explain the background motivations behind the choices to be taken during the population. Finally, the best practices that suggest what to do and what to do not, in order to achieve an effective and efficient filling of the framework, by avoiding errors and replicating success cases.

In this deliverable, we will assume that the concepts and artefacts (need, challenge, idea, dataset, building block, public service application), on which the WeLive approach is based, are already known to the user and we will also assume that the functional role and behaviour of the WeLive framework components and tools is clear as well. Therefore, we will concentrate on how those artefacts can be injected into the framework specifying one by one where we expect them to reside (mainly inside, but possibly also outside the framework) and how to make them available on the framework. In particular, this report is structured in the following sections:

- Section 3 deals with the population of datasets;
- Section 4 deals with the population of building blocks;
- Section 5 deals with the population of public service applications;
- Section 6 deals with the hosting of software artifacts that are exposed on the framework to be reused;
- Section 7 deals with the population of the OIA component with ideas, needs and challenges;
- Section 8 deals with the population of the DE component with those rules that support the recommendation functionalities of the framework;
- Finally, section 9 provides conclusions about task T3.2 and about this report.
3. DATASETS POPULATION

3.1. TECHNICAL GUIDE

For creating a dataset into the WeLive Framework, the Open Data Stack offers three different alternatives: manually, using its web interface; or programmatically using one of its harvesters or its RESTful API.

3.1.1. Creating a new dataset using interface provided by the Open Data Stack

The ODS provides an intuitive wizard for creating datasets and related resources. As shown in Figure 1, to start the process the user has to click on “Add dataset” button, under “Datasets” menu item.

![Figure 1 – Button for starting the dataset creation process.](image)

Next, the dataset creation form is shown (Figure 2). This form provides all the fields required for fulfilling properly all needed metadata for a dataset. In addition to the predefined fields, this form allows defining a custom set of key-value pairs. These fields are widely explained in subsection 3.2.

![Figure 2 – Dataset creation form.](image)
At the next step, user can create one or many resources related to the dataset, as can be seen in Figure 3.

![Resource creation form](image)

3.1.2. Harvesting datasets from existing Open Data portals

The ODS allows harvesting metadata of datasets and resources hosted by existing Open Data portals. For this purpose, a CKAN harvester and a DCAT harvester have been developed. The objective of these harvesters is, given the URL of a CKAN data portal or a DCAT description of a data collection, create the correspondent datasets and resources into the ODS. These harvesters only replicate the metadata, while the URL of resources remains linking to original ones. In Figure 4, the way of creating a new harvester is shown.
3.1.3. Creating new datasets using the WeLive REST API

The WeLive framework allows creating datasets and resources programmatically through its REST API. The following two methods are relevant for creating datasets and resources:

- **POST /ods/dataset**: this method allows creating datasets, receiving dataset metadata inside the body of the POST request as JSON data.

The harvesting process can be configured to be launched manually or periodically (daily, weekly or monthly). This process checks the original data source in order to import new datasets into the ODS. Different aspects of the harvester, like the extra attributes, the language of the datasets and so on, can be configured inserting the appropriate JSON data in “Configuration” field.
POST /ods/{datasetID}/resource: this method allows creating resources within a dataset identified by its ID. Alike the dataset creation method, resource metadata is inserted inside the body of the request as JSON data.

In order to execute these methods, the proper OAuth token provided by WeLive’s AAC is required.

### 3.2. METHODOLOGICAL GUIDELINES

Independently of the method selected to populate the Open Data Stack, all datasets are required to fill a minimum set of metadata. In the case of datasets, these are the most relevant fields:

- **Title (required):** a descriptive title for the dataset. The ODS uses this title to construct the URI which uniquely identifies the dataset.
- **Language (required):** the language of the resources published within the dataset. The Decision Engine uses this field mainly for performing artifact recommendations.
- **Description:** basic information about the data defined by the dataset.
- **Tags:** the usage of tags is highly recommended to allow the Decision Engine making dataset recommendations.
- **License (required)** under the datasets are published. The selected license has to be aligned with the Terms of Use of WeLive Framework. In case of selecting a license, which demands to provide the name of the creator and attribution parties, **Author** and **Source** fields must be filled.
- **Organization (required):** organization to which the dataset belongs. In the case of WeLive project, these organizations are City of Bilbao, City of Novi Sad, Helsinki-Uusimaa region and City of Trento.
- **Visibility (required):** the visibility of a dataset can be public or private. All users in the WeLive platform can access public datasets, while the creator and a set of users selected by him or her only can access private datasets.
- **Agreement of the terms of use:** before proceeding to the publication of the dataset, user has to agree with the terms of use from WeLive project, as Figure 5 shows.

![Figure 5 – Terms of Use agreement when creating a new dataset.](image)

In the same way, there are some guidelines to follow when creating a new resource within a dataset:

- **File (required):** every resource requires a data file. This file can be uploaded from the local storage of the user or provided through a direct link to the original location of the resource.
- **Title (required):** a descriptive title for the resource.
- **Description:** description of the data contained by the resource. We strongly recommend including clarifications about the format of the data, units, scope and other characteristics of the data in this field.
Format: indicating the format of the resource is not mandatory, but it is recommending for integrating it into the Query Mapper.

Validation schema: this schema allows validating resources against CSV, JSON, XML and RDF schemas. Recommended if the dataset is going to be opened to collaborations from third users.

Mapping: this mapping allows the query mapper to query the resource using SQL queries. When it is leaved blank, the ODS generates a default mapping automatically.

Permissions: in this field, creator of the resource can define the read and write permissions for this resource.

3.3. BEST PRACTISES

For the first pilot phase, the ODS has been populated using both manual population (Novi Sad and Helsinki-Uusimaa) and harvesters (Bilbao and Trento). The usage of harvesters has demonstrated their usefulness, despite of some issues that have forced to develop some patches to fix them. One example of these issues is that the DCAT harvester was not expected to support multilingual DCAT catalogues, while the catalogue from Bilbao describes the datasets in both Basque and Spanish languages. This issue was fixed by performing an extra development that was not planned. Another detected issue was the behavior of the default CKAN harvester. This harvester detects as change all the extra attributes included by Open Data Stack and not present in the original datasets, so it used to delete them. A fork of this harvester has been created for fixing this issue and others.
4. BUILDING BLOCKS POPULATION

4.1. TECHNICAL GUIDE

A Building Block (BB) is a kind of software artefact. In particular, it is a web service (SOAP or RESTful based) that exposes APIs to provide functionalities to the developers.

In order to make a BB available to the other participants of the WeLive ecosystem, an authorized user (as a diffusor) should describe the BB in compliance with the BB specification metamodel and publish it onto the WeLive Marketplace.

The WeLive Marketplace component provides two ways to support the BB publication:

- By using wizard on line tool;
- By using the authenticated APIs.

Both the ways provides validation rules useful to verify the compliancy with the BB specification metamodel.

To proceed with the BB publication through the APIs, the authorized user (as a diffusor) has to invoke programmatically:

1. the “validation” API method, to verify the compliancy of the descriptor schema file.
2. the “create-service” API method, indicating into the request header the basic authentication credentials and as inputs the BB required metadata.

To proceed with the BB publication through the API wizard tool, the authorized user (as a diffusor) has to:

1. perform the login onto the WeLive Platform;
2. select the Marketplace on the platform header;
3. click on “New Artefact” button in the Marketplace catalogue;
4. and then choose the “Building Blocks”, as shown in Figure 6.

![Figure 6 – Starting the building block publication wizard.](image)

In order to support the user during the building block publication, the wizard applies several validation rules to the provided values and checks the compliancy of the web service descriptor.
As shown in Figure 7 an alternative to the manual description, the user can upload a RDF file that describes properly the building block to be published. Furthermore, the wizard allow the user to upload some images to be associated to the new building block and to be shown into the BB details page.

If at least all required fields are provided then the user can complete the procedure and the Marketplace publish the BB onto the platform.

4.2. METHODOLOGICAL GUIDELINES

During the artefact publication onto the Welive Platform, it is necessary to provide the required metadata in compliance with the Welive BB specification metamodel.

In the case of web services, the BB metamodel consider the following metadata:

**Important note: the fields marked with an asterisk are required.**

- Title (*): the title of the artefact;
- Description (*): a short description of the artefact;
- Type (*): the artefact category;
- REST or SOAP web services for Building Blocks;
- WSDL/WADL descriptor URL (*): indicates one of the following;
- WADL URL for REST Web Services;
- WSDL URL for SOAP Web Services;
- Documentation Page: source code, documentation or Swagger page of the artefact;
- Tags: list of terms that characterize the artefact separated by comma.

The BB descriptor schema file URL is a required field, thus the deployment onto the preferred hosting environment must be performed before the publication onto the Marketplace.
Building block population inherently support the WeLive innovation model and must be performed following the principles of the WeLive Code of Conduct [17]. In particular, attention should be paid to ethical decision in the provisioning of building blocks and to tackle the needs of vulnerable people. The guidelines suggested here are compatible with all the prescriptions of the WeLive Code of Conduct.

4.3. BEST PRACTISES

In order to avoid redundancies and to maintain clear and fast the Welive environment at whole, before the publication of the initial set of Building Blocks (BBs) onto the Welive Platform, it is suggested to follow the following guideline:

- To collect all the web services provided by the referred pilot Municipalities;
- For each collected web service:
  - To collect required information: referred pilot, name, description, datasets used, URL;
  - To collect other useful data: images, user and installation manuals, etc...
  - To check the compliance with the WeLive BB requirements by invoking the API exposed by the Marketplace to validate the descriptor schema file.
  - To identify the hosting environment where it has been (or must be) deployed.
- Group the BBs by the referred pilot.

Therefore, each Authority can take in charge the publication of the web service related to its pilot.
5. PUBLIC SERVICES POPULATION

In this section we describe how the WeLive platform is populated with the public service applications (PSA), providing both the technical and the methodological guidelines for this activity.

5.1. TECHNICAL GUIDE

Public Service Applications represent mobile applications (currently only Android apps are supported) and responsive Web applications (based on HTML5) that are built on top of the WeLive building blocks and potentially exploit the WeLive platform cross-cutting components (i.e., AAC and logging).

Currently, the only way to populate the platform with PSAs is through the marketplace interface, in particular using the interactive artefact publication wizard.

The publication starts from the marketplace interface by pressing the “New Artefact” button. The user has to choose “Public Service Application” option in order to proceed with PSA population.

![Figure 8 – Publication of a Public Service Application as a new artefact.](image)

In the following step, the user has to provide the relevant artefact attributes of the published PSA:

![Figure 9 – New Public Service Application form.](image)
Next, the user can provide an image or a logo associated to the application:

![Image of logo](image.png)

**Figure 10 – Adding a logo to a Public Service Application.**

Additionally, in the advanced mode, the user may provide the Linked-USDL specification of the artefact if already present uploading the corresponding RDF file.

Once the relevant information is provided, the artefact may be published by pressing the “Publish” button.

### 5.2. METHODOLOGICAL GUIDELINES

During the publication of the PSA artefact, it is necessary to adhere to the corresponding requirements. Specifically, it is necessary to

1. Provide the human-readable application title (required field);
2. Provide a description of the application (required field);
3. Define the appropriate type of the application (e.g., Android or Web app);
4. Provide a link to the PSA page. In case of Android application, this should be a link to the Google Play Store application page. In case of the Web application, this should be a URL of the starting page of the application.

Provide a list of tags/keywords associated to the application. The choice of the keywords should be taken with care as this information may affect the user recommendations.

### 5.3. BEST PRACTISES

Manual publication is the only currently supported way for the PSA publication. Given the nature of the kind of the applications considered so far (e.g., Android application), this is not a strong limitation of the platform as such artefact are produced not in a massive way. However, more automation may be eventually provided in order to facilitate the process of publication. For example, in case of PSAs produced as an outcome of the Visual Composer, the publication of such apps may be realized directly by the Visual Composer tool using the Marketplace API. Another example of such an automation may be a tool to publish Android apps given only their reference Google Play identifier: the rest of the information may be obtained from the Google Play directly.
6. **HOSTING SOFTWARE ARTIFACTS**

WeLive expects that all submitted artifacts be hosted in some servers accessible via the Internet. In this section, we provide technical and methodological guidelines about development and hosting of artifacts for the WeLive environment.

6.1. **CHOOSING THE HOSTING ENVIRONMENT**

Artifact developers can choose to host their own artifacts by themselves or utilize one of the two hosting environments offered by WeLive. Developers who are experienced in hosting web applications and maintaining servers may find it easiest to host their own servers. They can freely choose their favorite virtual server providers or use their own physical servers if available. In this case, they will be responsible for related costs and maintenance tasks as well.

Developers who already have implemented a building block or web application and just need a place to run it are recommended to consider using the CloudFoundry Hosting Environment provided by WeLive. This option is the best also for software artifacts that need access to various other services such as databases. Technical guide for using the CloudFoundry Hosting Environment is given in Section 6.2.

The CNS Hosting Environment is best suited for building blocks that require a lot of computing power, make frequent sub-requests to other building blocks, need special access control or involve payments. In order to use this environment, developers must follow special technical instructions to make their software modules compatible with the CNS environment. Technical guide for using the CloudFoundry Hosting Environment is given in Section 6.3.

6.2. **TECHNICAL GUIDE FOR CLOUDFOUNDRY HOSTING ENVIRONMENT**

CloudFoundry is a Platform-as-a-Service Cloud infrastructure used to host and execute Web applications developed using a set of supported technologies and frameworks.

Currently, the CloudFoundry platform made available within the WeLive project supports the following languages and frameworks (referred to as Buildpacks):

- Java (executable spring apps or web apps packed as war files);
- Ruby
- NodeJS
- Go
- Python
- PHP

6.2.1. **Using CloudFoundry**

To start using the CloudFoundry, it is necessary to install the command-line interface. The installation instructions may be found at [3].

Once installed, it is necessary to target the CloudFoundry endpoint:

```
> cf api https://api.cloudfoundry.welive.eu
```

Figure 11 – Targetting the CloudFoundry endpoint.
The endpoint may also be associated during the login phase. To perform the login, it is necessary to use the credentials obtained from the provider:

```
> cf login -a https://api.cloudfoundry.welive.eu -u username -p password
```

Figure 12 – CloudFoundry login.

Once entered, the user is automatically associated to the organization (WeLive) and the space (corresponds to the project organization name). The users have the developer permissions and are allowed to deploy the applications within the space quota.

### 6.2.2. Deploying app to CloudFoundry

To deploy an application to the cloud, the following steps are required:

1. Prepare the application artefacts. This may be done, e.g., by packaging the app (Java war or jar file) or by providing the information about source code, deployment manifest, etc. More details may be found at [4].

   For example, to prepare the artefact for the mavenized Java project, it is enough to build a jar or war file:

   ```
   > mvn clean package
   ```

   Figure 13 – Packaging a mavenized Java project.

   2. Push the application to CloudFoundry. If the artefact folder contains the deployment descriptor (yml file), the following command may be used:

   ```
   > cf push APPNAME
   ```

   Figure 14 – Pushing to CloudFoundry an artifact with a deployment descriptor.

   Here APPNAME is the name of the application as it will be deployed and exposed in the CloudFoundry.

   Alternatively, to push the packaged artefact it is necessary to pass the reference to it:

   ```
   > cf push APPNAME -p target/welive.quickstart.war
   ```

   Figure 15 – Pushing to CloudFoundry an artifact by referencing it.

   Once deployed, the application will be exposed on the internet at the following address:

   `https://APPNAME.cloudfoundry.welive.eu`

   Please note that by default the endpoint certificate may be not recognized as valid signed SSL certificate.

### 6.2.3. Using CloudFoundry services

CloudFoundry allows the application to exploit additional infrastructural components, such as message queues, databases, etc. These components, referred to as CloudFoundry services, should be instantiated and

---

1 Currenty the WeLive SSO integration is not supported.
bind to the application during the application. More details about CloudFoundry services and their usage for the applications may be found at [5].

Currently, the following services are enabled for the developers:

- PostgreSQL
- MongoDB

To see the list of services available, it is necessary to access the CloudFoundry service marketplace:

```bash
> cf marketplace
Getting services from marketplace in org WeLive / space tecnalia as admin...
OK
<table>
<thead>
<tr>
<th>service</th>
<th>plans</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB</td>
<td>Default</td>
<td>A simple MongoDB service broker implementation</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Basic</td>
<td>PostgreSQL on shared instance</td>
</tr>
</tbody>
</table>
```

![Figure 16 – List of services available on CloudFoundry marketplace.](image)

The command returns the list of available service types and associated plans.

To create an instance of a service, the following command is used:

```bash
> cf create-service SERVICE PLAN SERVICE_INSTANCE_NAME
```

![Figure 17 – Creation of service instance on CloudFoundry.](image)

Where SERVICE is the name of the service, Plan is the associated plan (use "" around the plan name) and SERVICE_INSTANCE_NAME is an alias to be used for the service instance.

To bind a service to an application, the following command is used:

```bash
> cf bind-service APP_NAME SERVICE_INSTANCE
```

![Figure 18 – Bind a service to an application in CloudFoundry.](image)

This allows the application to use the service instance within the application.

Note that the same service may be bound to many applications if necessary.

Once you have a service instance created and bound to your application, you need to configure the application to dynamically fetch the credentials for your service instance. The VCAP_SERVICES environment variable contains credentials and additional metadata for all bound service instances. There are two methods developers can leverage to have their applications consume binding credentials.

- Parse the JSON yourself: See the documentation for VCAP_SERVICES. Helper libraries are available for some frameworks.
- Auto-configuration: Some buildpacks create a service connection for you by creating additional environment variables, updating config files, or passing system parameters to the JVM.

For details on consuming credentials specific to your development framework, refer to the Service Binding section in the documentation for your framework’s buildpack (for Java Spring apps see [6]).
6.2.4. Using CloudFoundry Eclipse plugin

CloudFoundry provides an Eclipse IDE extension for Java developers. This allows configuring, deploying, and running the applications directly from the Eclipse IDE. More details may be found at [7].

6.3. TECHNICAL GUIDE FOR CNS HOSTING ENVIRONMENT

The CNS Hosting Environment (CNS-HE) is based on the Cloud’N’Sce.fi marketplace (CNS Marketplace), available at CNS Site ([9]), which has been extended with functionality to create and host WeLive Building Blocks. The CNS Marketplace is a commercial platform developed and operated by CNS. It implements the Algorithms-as-a-Service concept and offers an easy way to create, utilize and commercialize algorithmic data refining solutions as web services.

Online help pages of the CNS Marketplace can be viewed at [8]. They contain detailed descriptions about CNS marketplace features and guide through common use cases. Help pages will be referred in this document for further details and include the most up-to-date information. Page specific help content can be opened by simply clicking any of the title texts shown on pages. For example, clicking the ‘CATALOG’ title word at the Application Catalog page will open the related help page.

6.3.1. Using Building Blocks hosted in the CNS Marketplace

Building blocks can be used via their automatically generated HTTP REST API. In practice, a CNS Marketplace component called ‘Building Block Proxy’ (BBP) handles all BB requests and takes care of user authentication and BB usage monitoring. From the CNS marketplace’s perspective, all BBs are applications that access data refining services via the CNS marketplace API (CNS-API) and BBP is a general-purpose implementation of such BB applications. Therefore, Building Blocks are included to the Application Catalog of the CNS Marketplace.

All building blocks hosted in the CNS Marketplace require a valid access token to be transmitted with every BB request. Both WeLive and CNS access tokens are accepted. In order to create a new access token, the application accessing the BB should require user authentication with either WeLive or CNS login.

Clicking the BB entry START button on the Application Catalog page opens a Swagger UI that shows detailed description of the building block REST API. The UI also demonstrates making API requests by allowing user to enter input parameters submit requests and see the responses. Submitting BB requests requires user authentication with either WeLive or CNS login. Clicking the authentication button in the BB Swagger UI initiates the authentication process using the WeLive AAC BB.

Some building blocks, which access protected resources such as databases, may deny access from users or applications unauthorized to access the resource. The CNS Marketplace has a special support for managing access control for protected resources. See Section 6.3.6 for more information about protected resources.

The CNS marketplace creates customer entries automatically for all WeLive users that access building blocks hosted by CNS either directly or via some public service. In other words, the end user does not need to be aware of the fact that the BB is hosted in the CNS Marketplace. Later on, if the same user also signs up to the CNS Marketplace, the automatically generated CNS customer entry can be associated to the CNS user account.

In case the user is already logged into the CNS Marketplace, a new BB session and access token will be generated automatically for the user whenever necessary.

6.3.2. Becoming a CNS Marketplace user or provider

In order to enable the CNS Login option for BBs or to use/create data refining services at the CNS marketplace, the user must first register as a CNS marketplace user and accept usage terms. A new user
account can be created by entering the CNS Site, clicking “Login” from the upper-right corner, and selecting “Sign up” from the popup window. After submitting the form and clicking the confirmation link in the sent confirmation email, the account is ready for use.

Each user is associated to at least one customer profile, which defines the legal entity presented by the user that can currently be either a consumer or a company (please contact CNS in case neither of them fits for your organization). During registration, a consumer profile, which defines the individual properties, is created for each user. User may create one or more company profiles as needed. Users may be asked to fill missing details of customer profiles before making payments or entering agreements to ensure that the created invoices, receipts and legal documents are valid.

Users who would like to submit new content to the CNS marketplace must also register one or more provider accounts and accept related usage terms. The provider account can represent a person, a team, a company or an organization and works as a marketing brand for published content. However, provider account is always associated to a real-world customer profile who is responsible for provider activity. Registered providers and content published by them can be browsed in the CNS marketplace and each provider has a forum page for user support and marketing purposes. You can create a provider account in your My Business page by clicking “ADD PROVIDER” and filling the form with basic information such as name, description, logo and associated customer profile.

6.3.3. Publishing existing data refining services as Building Blocks

Any data refining service available in the CNS marketplace can be published as a WeLive compatible building block. Any registered CNS User can become a “BB Publisher” by obtaining access to an existing data refining service in the CNS marketplace and allowing others to use it via an automatically generated REST API. In other words, the one who publishes a building block and pays for possible related costs does not need to be the original developer – it can be, for example, a city that wants to make a nice BB available to citizens free.

See the CNS help page ([10]) for more information.

6.3.3.1. Obtaining access to a data refining solution

CNS marketplace users can find available data refining solutions and active campaigns from the Solution Catalog ([11]). You can obtain access to an interesting solution by choosing the most suitable campaign and accepting its terms by signing a contract called Solution Agreement. Contracts are effective for a limited period and may involve usage limitations such as maximum request count.

You may also want to create a new project for the building block costs. Enter the My Projects page to create a new project and manage its budget. Use this project whenever purchasing access to solutions that you plan to make available as building blocks.

Some solutions may require access to protected resources not included in the campaign. In case you already have access to the required resource, it will be automatically associated to the contract and you are ready to make requests. In case the resource is missing, you may try to borrow it from its owner and associate it to the contract later from the My Contracts page. See Section 6.3.6 for more information about protected resources.

6.3.3.2. Submitting a new Building Block

To create a new building block, go to your My Business page and click the number under “BBs” in your Provider Accounts section. Then click the “ADD BUILDING BLOCK” button. This opens a simple form where you can give a name and a description for the building block and add one or more BB features by associating a function name to an existing solution contract.
Optionally, you can allow the BB to use some of your protected resources (see Section 6.3.6). Be careful when enabling resources that grant write access to common data repositories or external services involving usage cost, because anyone can make any number of requests to public BBs. In case the BB requires resources, which should not be publicly accessible, a better option may be to grant resources to each application that uses the BB.

After submitting BB information, the building block is available for use at the CNS marketplace via an automatically generated REST API, which bases on the function and data interfaces of the solutions associated to the BB. The given function names are included in the BB entry point URLs and identify the contract to be applied for incoming BB requests.

Note that submitted building blocks are public by default. In other words, any WeLive or CNS user can access the BB and make requests to associated solution contracts at your cost. In case you wish to limit access to CNS users that are associated to your provider team, select option ‘Private’. When you are ready, click the ‘Submit’ button to create and publish the building block in the CNS Marketplace.

### 6.3.3.3. Publishing a BB in the WeLive Marketplace

Created building blocks can be submitted to the WeLive Marketplace with a couple of clicks. Once you have verified the functionality and descriptions of your BB, go to the My Business page, open the My Building Blocks view and click the name of the BB you want to publish at WeLive. In case your CNS user account is already associated to a WeLive account, a PUBLISH button will be shown (if you do not see it, please contact CNS support to enable it). Clicking the button opens a window showing the technical specification of your BB (Linked-USDL) and a ccUserId field identifying the WeLive user who will become the owner of the BB. Please check that everything is OK and click the PUBLISH button to submit the specification to the WeLive platform. If everything succeeds, your BB should appear in the WeLive Marketplace.

### 6.3.4. Implementing new data refining solutions

This subsection describes the main principles for implementing new data refining solutions that can be hosted in the CNS Marketplace and published as WeLive Building Blocks.

#### 6.3.4.1. Integration Principles

Data refining services available in the CNS marketplace are based on software modules that implement one or more specified function interfaces. Software modules can be implemented in practically any programming language and may utilize third party libraries. This allows developers to use their favourite technologies the most suitable for the purpose and significantly lowers the overhead of integrating an existing solution to the marketplace.

Technical flexibility is achieved by CNS’s decision to do software integration on process and file system level instead of requiring developers to implement their software based on some specific software framework. Software modules are launched as stand-alone processes in a secure sandbox and communicate with the marketplace system using standard input/output/error streams and via file system. The protocol used to communicate with algorithm processes is described in the following subsection.

#### 6.3.4.2. Cloud’N’Sci Refinery Protocol

When a new request to a specific data refining service arrives, the CNS Marketplace first checks whether the process executing the software module implementing the service is already running and starts a new process if necessary. Then it transfers all input data items to the server running the process and extracts them into local file system folders accessible by the process (read/write allowed by the sandbox). After this the system
writes a command line to the process’s standard input stream, including a unique request id, absolute file system paths to the request input data folders and a path for folder to which request output should be saved. After completing request processing (either ready, failed or aborted) the process should indicate this by writing a line to standard output stream including the completed request id and paths to output folders.

This protocol used to communicate with data refining processes details is called the Cloud’N’Sci Refinery Protocol (CNS-RP). Detailed documentation of the CNS-RP protocol can be found in GitHub repository at [12]. All software modules to be hosted in the CNS marketplace are required to implement this protocol. CNS will provide example implementations of the protocol in most popular programming languages as well as refinery toolkits to simplify refinery development and testing.

6.3.4.3. Refinery Toolkits

Refinery toolkits implement the CNS-RP protocol and simplify refinery implementation and testing. Currently, refinery toolkits for Java and Python are available for beta testing. Please contact CNS in case you wish to get access to a GitHub repository containing the toolkit source code. Later on, toolkits will be made publicly available as open source software.

See the CNS help page ([13]) for the most up-to-date information about available refinery toolkits.

6.3.4.4. Docker

Docker technology is used to define the exact execution environment and dependencies required by the software module. Unless some specific refinery toolkit is used (which take care of Docker related things), the repository should have a file named Dockerfile in the root directory including instructions for setting up the virtual machine to run the software. Usually, defining a new Dockerfile is easy because it can be based on one of the existing base images, which install required system tools and libraries. For example, in case the software is written in Python, base image continuumio/anaconda3 will set up a full python execution environment. After that, only module specific python source files needs to be added to the image in order to run the software.

6.3.5. Publishing data refining solutions

Solutions are high-level descriptions of data refining services. They can be published in the marketplace even before actual implementation to introduce the idea to the target audience and get valuable feedback. When a solution is implemented, released and made available by launching a sales campaign it becomes a concrete data refining service that can be purchased in the marketplace and accessed under the terms specified by the campaign. The following subsections describe the main steps of the solution submission processes.

See the CNS help page ([15]) for more information about publishing solutions.

6.3.5.1. Data Interfaces

Data Interfaces specify the detailed structure and semantics of input/output/error data items. Developers can write their own data interface documentation, refer to existing standards, give valid data examples or give an exact schema definition, e.g. as a JSON Schema. Published data interfaces can be browsed and viewed on the Data Interfaces page. Developers are highly recommended to re-use an existing data interface whenever possible instead of writing a duplicate.

If you cannot find a data interface that suits your needs, you can create a new one in the CNS marketplace Data Interfaces page. Clicking “ADD DATA INTERFACE” opens a form, where you can enter a name, a provider and a description for the data interface. You can also choose MIME Type for the data and/or select to use JSON and insert a JSON Schema.
6.3.5.2. Function Interfaces

Function Interfaces specify the input, output and error data for a valid request to a data refining service and describe high-level service functionality, i.e. how output depends on the given input. Each data refining service implements exactly one function interface. Each function interface defines 1-N input data items, 1-M output data items and 1-K error data items. Each data item has a unique name (in the scope of the function interface) and refers to one of the data interfaces.

Function interfaces can be created on the Function Interfaces page. Clicking “ADD FUNCTION INTERFACE” opens a form, where you can enter a name, a provider and a description for the function interface. You also need to specify input, output and error data of the function and associate them to existing data interfaces.

6.3.5.3. Submitting Solutions

You can create a new solution in your My Business page by clicking “ADD SOLUTION”. Solution information must include a name, short description and a link to a function interface specifying usage instructions. Providers are highly recommended to re-use existing function interfaces whenever possible instead of writing their own. Optionally, solutions can also have a logo and links for further information or demos.

6.3.5.4. Describing Algorithms

Solution releases can be associated to one or more algorithms implemented by the solution. Please refer to an existing algorithm whenever possible so that people can find your solution easier by visiting the Algorithm Summary page. You can also submit new algorithms in your My Business page by selecting “Algorithms” under your Provider Accounts section and clicking “ADD ALGORITHM”. This opens a simple form, where you can give a name and a description for your algorithm and add further information links and a logo.

6.3.5.5. Building Images

Images are snapshots of software modules deployed into the marketplace. An image can comprise implementations of multiple solutions and required resource files. Developers can build and deploy images by first pushing all necessary files into a Git repository (support for other version control system may be added later) that is accessible from Internet.

To build an image, go to your My Business page, select “Images” in Provider Accounts section and click “ADD IMAGE” button. This opens a form, where you can enter a name, a type and a description for your image. You will also need a Git repository url and a repository branch. In case of private repositories, you must allow access to it with the SSH key generated by the marketplace. The key is generated when you click “Continue”. In GitHub, you can allow access by adding the generated public SSH key to the list of user keys or as a repository deployment key. If your repository is public, SSH key is not needed.

6.3.5.6. Making Solution Releases

Releases bind solutions and images together. Developer can launch a new version of a solution by submitting a new release, which is associated to an image and defines the exact launch command that will start the refinery process in the Docker container based on the image. The launch command can include any number of arguments to configure solution instances for specific purposes. For example, an optimization algorithm can take the maximum number of search iterations as a command line argument affecting both running time and result quality. In other words, the developer can create different releases of the same solution targeted to different user groups and needs simply by modifying the launch command.

To release your solution, go to your My Business page, select “Solutions” in Provider Accounts section and click your solution version. This opens a popup window, where you can use older releases as a template by
clicking their version number. You can also create a new release from empty template by clicking “New Release” without selecting a version.

Version number has to be different from any of the earlier releases for the same solution and may contain text. Optionally, you can associate the release to one or more algorithms utilized in the solution. Image refers to the Docker image that contains your solution implementation. Launch command has to be the exact command that starts the refinery process in the Docker container based on the image. You can also add release notes about this version. Maintenance & Support period is the time you agree to maintain and support this release.

In case your solution uses other solutions as a part of its data refining process, you need to list them as dependencies. First, you need to obtain access to the required solution by signing a contract for a suitable campaign. Then add a new release dependency row and select the contract from the drop-down list. Finally, specify the exact function name called by the release when making sub-requests. This maps the abstract function name to a concrete data refining service contract. In other words, whenever the release makes a sub-request using the function name, it will trigger a request to the data refining service specified by the contract. Note that normal contract terms apply also to sub-requests made by your releases. For example, in case the contract defines a transaction price, it will be charged for each sub-request from the project chosen when signing the contract.

By default, data refining solutions are executed inside a secure sandbox, which denies access to the Internet. This is sufficient for pure algorithmic solutions that just refine the given input data to generate the desired output, possibly using some static resources included in the image. In case the solution requires access to some specific external resource such as a database or web service, it must be specified as a required resource when submitting the solution release. See the CNS help page ([14]) for more information about required resources.

6.3.5.7. Launching Sales Campaigns

Campaigns make solution releases available to marketplace users as purchasable data refining services. You can design and launch sales campaigns by using a campaign plan wizard. To start the wizard, go to your My Business page, select “Solutions” in Provider Accounts section and select “Campaigns” in the solutions row. Opened popup window shows current campaigns. Click “NEW CAMPAIGN PLAN” to open the campaign plan wizard.

In the opened popup window, you can select the solution version offered by the campaign. You can freely choose a transaction base price to be charged from users per successful request under the basic campaign terms. You can also choose to offer lowered transactions prices for users that make an initial purchase, i.e. pay for a certain number of transactions in advance. You can set this with “Volume discount limit”. Developer may also decide to offer few free trial transactions per user so that customers can try the solution before making a purchase decision. Regardless of the transaction price charged from end-users, developers have to pay a fixed hosting fee for each successfully executed transaction.

Optionally, you can include access to one or more protected resources to a campaign. If you choose to do so, the resource will be associated to all contracts based on the campaign so that it can be utilized in solution requests. The customer only sees the name of the resource option, but not the associated value.

6.3.6. Protected Resources

Protected Resources allow controlling access to external resources such as databases. Each resource has a unique name and can comprise multiple “features”. For example, a database resource can be named “MyDB”
and include features “login”, “password”. Each resource feature can have multiple available options that include a public name and secret value. Every feature has one default option named N/A indicating that the feature is not available.

New resources can be created from the My Resources page. Resource creator owns the resource and can “borrow” it to other marketplace users without exposing the secret values associated to feature options. For example, the owner of MyDB resource can grant other users access to the database by borrowing MyDB.login and MyDB.password features without exposing the actual password. The borrowed resources can be associated to solution contracts or application sessions to enable accessing the database when processing data refining requests.

Protected resources are especially useful with building blocks. Building blocks are public by default, which may involve risks if too powerful functionality can be requested by anybody without limitations. For example, publishing a building block that inserts data to a shared database could endanger database quality - it would be trivial to do a malicious attack filling the database with unacceptable data such as advertisements or adult content. A better solution is to publish the BB without the necessary resource, and grant database write access only to applications by trusted providers. The CNS Marketplace detects which application is calling the BB and can complete the set of required resources per BB request.

See the CNS help page ([14]) for more information about protected resources and tips for using them.

6.4. METHODOLOGICAL GUIDELINES

Artifact developers should consider the following aspects when selecting the hosting solution:

- **Compatibility**: All artifacts must satisfy the given requirements for WeLive compatible artifacts.
- **Availability**: All artifacts submitted to the WeLive Marketplace must be hosted somewhere before they can be published. Availability of artifacts should be as high as possible without long unexpected breaks.
- **Sustainability**: Developers should be prepared to maintain their artifacts and hosting arrangements for a reasonable long period (preferably years rather than weeks or months). In case a published artifact becomes unavailable, is relocated or is modified significantly, other artifacts using it may become unusable.
- **Costs**: Hosting an artifact usually involves some costs. Developers should be prepared to handle artifact maintenance costs even if it becomes very popular.
- **Scalability**: Some artifacts may become very popular and cause high server load. Developers should be prepared to scale up the underlying computing resources to meet the increasing demand.
- **Version Control**: Usually artifacts need to be updated after the initial release, e.g. due to bug fixes or new features. Developers should be prepared to update their artifacts without disturbing other artifacts depending on the previous version. This may require that multiple versions of the same artifacts are kept running concurrently.
- **Commercial Potential**: Some artifacts may have commercial potential. Developers should consider business aspects already when implementing and publishing the first version of the artifact to keep future business possible. Some bad decisions made in the early phase may make it hard to commercialize the artifact later.

6.5. BEST PRACTISES

When designing and developing the applications for the Cloud-based hosting, it is necessary to carefully take into account the specific nature of the Cloud infrastructure. There are several principles that facilitate the maintenance and migration of the app in the Cloud, they indeed apply for the building blocks deployed to the
CloudFoundry and CNS hosting environments. Below we highlight these best practices for the Cloud apps. Discussion that is more detailed may be found also in relation to the “Twelve-Factor App” ([16]).

**Avoid writing to the local file system.** In many Cloud-based platforms, the hosting containers provide only short-lived local system to facilitate the instance migration and recovery. Moreover, it is often the case that different instances of the same app do not share the local system and, therefore, additional synchronization effort may be required. It is recommended to use the shared data services, such as database, for the purposes of dynamic data storage.

**Use logging service provided by the WeLive platform.** In the Cloud environment local file logging may be unavailable or inappropriate solution for the reasons described above: different instances may have separate files, the file system may be destroyed and recreated by the infrastructure, etc. Using the centralized logging service ensure the relevant information is kept in one place and is well-formed.

**Run multiple instances to increase availability.** To avoid the risk the building block becomes unavailable, it is recommended to scale the application by running several instances simultaneously. This, however, poses additional constraints on the application implementation, such as data synchronization.

**Adhere to stateless, share-nothing implementation.** To ensure seamless scalability and failover of the application, it is recommended to avoid the assumptions of data and state cache. Even the user session caching should be avoided as it poses additional constraints on the request routing: the request from the same users should be routed to the same instances. On the other hand, the stateless services that rely on an external data storage can be shut down and restarted without affecting the application functionality and logic.

**Scale out via process model.** Assign different types of activities to different isolated types of services. Avoid using a model, where all the tasks are handled by a single app. This allows running different types of services independently from each other, allowing for higher flexibility and efficiency.

**Minimize the startup/shutdown time.** This principle allows for faster application evolution and maintenance operations, making the restart of the instances almost invisible for the service consumers.

**Cookie accessibility.** The hosting environments like CloudFoundry may allow for cookies to be shared across different domains corresponding to different applications. This aspect should be taken into account when the applications rely on cookies for secure operations.
7. OPEN INNOVATION AREA POPULATION

In this section, aspects related to the Open Innovation Area will be described, by covering both technical and methodological aspects.

7.1. TECHNICAL GUIDE

In order to populate Open Innovation Area (OIA), it is necessary to create ideas, needs and challenges. In this section will be described the creation of OIA main concepts, that are: idea, need and challenge.

The OIA main page contains three main sections that describe this concepts and that provide the creation buttons to access the intuitive forms that allow the Authorities to create a new challenge and the registered users (as a Citizen, Business or Academy) to begin the Author of a new idea or a new need report.

The following figure depicts the OIA main page (for a Citizen user).

![Figure 19 – OIA main page (for a Citizen user).]

7.1.1. Creating an idea

To start the idea creation process, the registered user (as a Citizen, Business or Academy) has to click on “Create” button provided by the section named “Ideas”, contained into the OIA main page.

As shown in Figure 20, the idea creation form supports the Author to relate the new idea to an active challenge or to an Authority and provides all fields required for fulfilling properly required information (i.e. title, description, communicative image, categories) and optionally information (i.e. tags, Point Of Interest on the map, attached documents). Moreover, the Author can select other users to be involved as collaborators into the idea co-definition activities.
7.1.2. Creating a need report

To start the need report creation process, the registered user (as a Citizen, Business or Academy) has to click on “Create” button provided by the section named “Needs”, contained into the OIA main page.

As shown in Figure 21, the need report creation form supports the Author to send the new report to the pilot referred Authority and provides all fields for required information (i.e. title, description, communicative image, categories) and optionally information (i.e. tags, Point Of Interest on the map, attached documents).

Figure 21 – OIA need report creation form.
7.1.3. Creating a challenge

To start the challenge creation process, the Authority has to click on “Create” button provided by the section named “Challenge” contained into the OIA main page.

As shown in Figure 22, the challenge creation form provides to the Authority all fields for required information (i.e. title, description, communicative image, categories) and optionally information (i.e. tags, Point Of Interest on the map, reward description and attached documents). Moreover the Authority can select a need report to be associated to the challenge.

Figure 22 – OIA challenge creation form.

7.2. METHODOLOGICAL GUIDELINES

In order to populate properly the OIA and to ensure the achievement of their goals, the PAs should be aware of the following principles:

- **Transparency**: PAs share the information about their activities in order to be able to demonstrate the commitment and to enable the citizen to become an integral part of the organization as the controller.
- **Collaboration**: the PAs have the neutrality role of promoting dialogue, the cooperation and interaction between the local stakeholders.
- **Accountability**: the PAs can create public value and can be accountable to the citizens in a transparent way.
- **Innovation**: PAs can take innovative solutions that address not only the technological dimension, but also above all the paradigms of interaction with citizens and how their needs should be managed.

In fact, from the PAs point of view, the main goal should be to understand the real needs of the territory and meet them effectively. In this scenario, the citizens’ involvement is crucial, because their personal experience in the real life context get them capable to detect and report the requirements and to propose the desired solution.
Therefore, according to the Open Social Innovation and to the Service Co-creation paradigms, the Open Innovation Area (OIA), the Authorities should take into account the well-known needs and all the requirements provided by the local actors. People or groups of people with a certain level of territory knowledge should be identified and involved to collect, group, order and categorize all the well-known social and territorial issues. Next, they should plan a (simple) strategy to prioritize the issues and then to schedule the challenges to be launched.

After the publication of the initial challenges into the OIA, the Authorities must analyze all the contributions received (in form of ideas and/or needs) to understand if the scheduled challenges fit the real citizen needs. Moreover, in order to motivate the citizen involvement the Authorities should demonstrate the commitment and the interest on the social and territorial needs providing detailed information about the idea selection criterions and the achieved results by the implemented solutions.

7.3. BEST PRACTICES

In order to drive OIA population with a clear and defined strategy, before the initial OIA population, the Authorities could collect well-known needs by organizing several thematic meetings, brainstorming, focus groups, workshops, and gamification based activities.

The collected needs will inspire the initial set of challenges to be published into the OIA.

So, in compliance with the transparency principle, thanks to the OIA challenges explorer page, all the stakeholders will be able to see the pilot scheduling and to promote their ideas or (in case) to create a new needs report. These circumstances will trigger the citizens’ spontaneous contribution.

In order to be correctly evaluated and appreciated, all challenges/ideas/needs reports created into the OIA should respect the following simple guidelines:

- To adopt a simple language to get the theme/problem/solution understandable by all.
- To add a gallery of images clearly representing the exposed theme/problem/solution.
- To add tags that help other users to understand the referred topics.
- To add good and well-known collaborators for the possible solution implementation.

Instead, to achieve the objectives and to involve other users into the idea lifecycle will be useful the following best practice:

- The PA representatives should adopt the direct communication channels provided by the OIA, and by the Welive Platform at whole, in form of feedbacks, comments, chats.
- All the users should use the sharing functionalities to share on the social network links to the OIA contents.

In the end, the Authorities should motivate the stakeholders’ involvement into the idea lifecycle by:

- responding to the requests (comments, chats, emails, etc.);
- providing details about the challenges;
- evaluating the ideas and promoting the best one;
- supporting the idea definition and implementation phases providing useful requirements and feedbacks about the task outcomes.
8. DECISION ENGINE POPULATION

In this section, both technical and methodological guidelines for populating the Decision Engine are given. In addition, we describe some best practices learnt during the first phase of the pilots.

8.1. TECHNICAL GUIDE

For managing artifacts (datasets, building blocks or apps), ideas and collaborators, the Decision Engine provides a set of methods within WeLive’s REST API. Through these methods, different components from WeLive framework can add, update or delete artifacts, ideas or collaborators. Regarding to artefacts, the Decision Engine provides the following methods:

- **PUT /de/{artifactType}/{artifactID}:** this method allows creating or modifying artifacts, receiving artifact metadata inside the body of the PUT requests as JSON data. This method receives two URL parameters:
  - `artifactType`: the type of artifact that is going to be created: “dataset”, “building-block” or “app”.
  - `artifactID`: the unique ID for the proper identification of the artifact. The component that publishes artifacts is the one responsible of maintaining the uniqueness of this ID. In the case of WeLive, this is suitable because the Marketplace is the only component that publishes artifacts into the Decision Engine, so it can use its internal IDs when publishing artefacts.

  Regarding to the content of the body of these requests, the required JSON expects the following fields:
  - `lang`: the language of the given artefact. Available values are “spanish”, “serbian”, “english”, “italian” or “finnish”.
  - `tags`: tags used to describe the artifact.

  In the case of applications, two extra fields have to be inserted into the JSON:
  - `scope`: the scope (city or region) to which the app is associated.
  - `minimum_age`: the minimum recommended age for using the application.

- **DELETE /de/{artifactType}/{artifactID}:** this method allows deleting artifacts from the Decision Engine. The parameters from the URL are the same than the parameters from the PUT method.

Same as artifacts, ideas are managed through their PUT and DELETE methods:

- **PUT /de/idea/{ideaID}:** this method allows creating or modifying ideas into the Decision Engine. Similar to the creation of artefacts, the `ideaID` is a unique ID that has to be managed by the publisher of the idea. In the case of WeLive, this is suitable because the Open Innovation Area is the only component that publishes ideas into the Decision Engine, so it can use its internal IDs. Regarding to the fields of the JSON given in the body of the requests, only two fields are required:
  - `lang`: the language of the idea.
  - `text`: the text of the idea.

- **DELETE /de/idea/{ideaID}:** this method allows deleting ideas from the Decision Engine. The parameters from the URL are the same than the parameters from the PUT method.
In order to suggest WeLive users that may collaborate in the consecution of an idea regarding their skills, the Decision Engine provide the following method:

- **POST /de/idea/recommend/users_by_skills**

This method invokes the /cdv/getusersskills API (exposed by the Welive Citizen Data Vault), to obtain the list of all users with their known languages, skills and respective endorsement counter, and returns a list of users whose skills match the tags and/or categories specified in the response. These tags and categories are related to an idea to be implemented.

Although logically a GET operation, it has been designed as a POST operation due to the complexity of the format of the request parameters. The request accepts application/json Content-Types according to the format of the following example:

```json
{
    "lang": "english",
    "tag_list": [
        "java",
        "php"
    ],
    "category_list": [
        "security"
    ]
}
```

- **lang**: The language of the tags and categories specified. When different from english, tags and categories are translated into English via Microsoft Translator API before any matching. Languages currently supported are danish, dutch, english, finnish, french, german, hungarian, italian, norwegian, porter, portuguese, romanian, russian, spanish and swedish.

- **tag_list**: A list of tags the users’ skills must match. The tags are supposed to match the specified language.

- **category_list**: A list of idea categories the users’ skills must match. The categories are supposed to match the specified language.

The returned list of recommended collaborators includes those whose skills cover at least 50% of the tags and categories specified. The matching between the tags/categories and skills is performed by approximation. The response is formatted as a JSON list (Content-Type: application/json) as can be observed in the following example:

```json
[
    {
        "user_id": 123,
        "matching_rate": 0.67,
        "matching_skill_list": [
            "java",
            "php"
        ]
    }
]
```

- **user_id**: Unique user ID (also known as CCUserID) of the matching collaborator.

- **matching_rate**: Number of user’s skills that matched the tags and categories as a fraction of unity.
This call requires HTTP Basic Authentication.

On error, any HTTP response code different from 200 is returned.

8.2. METHODOLOGICAL GUIDELINES

Two different components from WeLive framework populate the Decision Engine: the WeLive Marketplace and the Open Innovation Area. The WeLive Marketplace is in charge of creating artifacts into the Decision Engine. When a user, app or other component from the framework publish a dataset, a building block or an app in the Marketplace, it automatically creates the proper artifact into the Decision Engine. On the other hand, the Open Innovation Area is in charge of publishing ideas into the Decision Engine. When a user creates an idea, the proper notification is done for creating the correspondent idea into the Decision Engine. As mentioned in subsection 8.1, the Decision Engine is not responsible of the unique IDs given to artifacts and ideas; the component that publishes artifacts and ideas has to take care of their uniqueness. This can be achieved using a centralized ID generation or prefixing the IDs with the name of the component (“componentID_artifactID”). In the case of WeLive, both Open Innovation Area and WeLive Marketplace maintain the uniqueness of the IDs from the artifacts and ideas published by them.

8.3. BEST PRACTISES

Given that the artifact recommendations are calculated based on tags from artifacts, it is mandatory that if we want to include an artifact in Decision Engine’s recommendations, and recommend other artifact related to the given one, this artifact has to be tagged. These tags, as more descriptive and less generic are, the Decision Engine could perform better recommendations. The specification of the language of the artifact allows applying different Natural Language Processing operations over these tags, so it is important to define it.

Regarding to the idea recommendation, the algorithm used to perform these recommendations (Latent Dirichlet Allocation) requires a wide corpus to train. This implies that as larger the texts of these ideas are, the Decision Engine will be able to perform better recommendations. For training the idea recommender, the training corpus is going to be generated during the first pilot phase. After this training, the idea recommender is going to be tested during the second pilot phase.

At conclusion, in order to allow useful recommendations, it is very important to invite the platform users to provide a detailed and sincere description about their skills, known languages, and preferences. Furthermore, it is very important to encourage the users that compose a co-working team to leave reciprocally an honest endorsement, in order to confirm the self-declared expertise and the competences of each collaborator involved into the idea implementation activities.
9. CONCLUSIONS

At the moment in which this deliverable is being written, the initial population of the WeLive framework instance by the partner of the project is running and is about to be completed. In this document, we have reported about different aspects of such population activity.

First, we have provided the technical instructions to inject the necessary artefacts and objects into the tools that are part of the framework and make the visible and accessible to users in order to trigger the innovation process proposed by WeLive. We have also addressed those cases in which the hosting of such artifacts does not occur strictly inside the WeLive framework explaining how they are made available onto the framework.

Then, we have provided the methodological guidelines that are driving the population that is taking place. The latter stem from the nature of the tools and artifacts involved and from the goals that are inherent to the WeLive project and the WeLive approach.

Finally, taking advantage of the already ongoing population activities we have devised some best practices, extracting the success and failure cases from the recent experiences. The detailed description of the populated environment will take place in the deliverable D3.5 – Trento, Bilbao, Finish Region and Novi Sad environment v1 [2].
## 10. ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC</td>
<td>Authentication and Authorization Component</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BB</td>
<td>Building Block</td>
</tr>
<tr>
<td>CF</td>
<td>CloudFoundry</td>
</tr>
<tr>
<td>CKAN</td>
<td>Comprehensive Knowledge Archive Network</td>
</tr>
<tr>
<td>CNS</td>
<td>Cloud'N'Sci</td>
</tr>
<tr>
<td>DB</td>
<td>Data Base</td>
</tr>
<tr>
<td>DE</td>
<td>Decision Engine</td>
</tr>
<tr>
<td>DCAT</td>
<td>Data Catalog Vocabulary</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>ODS</td>
<td>Open Data Stack</td>
</tr>
<tr>
<td>OIA</td>
<td>Open Innovation Area</td>
</tr>
<tr>
<td>PA</td>
<td>Public Administration</td>
</tr>
<tr>
<td>PSA</td>
<td>Public Service Application</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign On</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>USDL</td>
<td>Unified Service Description Language</td>
</tr>
</tbody>
</table>
11. REFERENCES

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>CNS resource protection documentation, <a href="https://cloudnsci.fi/info/protected-resources">https://cloudnsci.fi/info/protected-resources</a></td>
</tr>
</tbody>
</table>
### 12. COMMENTS FROM EXTERNAL REVIEWERS

#### 12.1. UDEUSTO

July 26\(^{th}\), 2016

<table>
<thead>
<tr>
<th>Issue</th>
<th>Yes</th>
<th>No</th>
<th>Score (1=low to 5=high)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is the architecture of the document correct?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Does the architecture of the document meet the objectives of the work done?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Does the index of the document collect precisely the tasks and issues that need to be reported?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Is the content of the document clear and well described?</td>
<td>X</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Does the content of each section describe the advance done during the task development?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Does the content have sufficient technical description to make clear the research and development performed?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Are all the figures and tables numerated and described?</td>
<td>X</td>
<td></td>
<td>3</td>
<td>Some figures do not have a caption. Please, add it.</td>
</tr>
<tr>
<td>Are the indexes correct?</td>
<td>X</td>
<td></td>
<td>4</td>
<td>Update them after adding missing captions.</td>
</tr>
<tr>
<td>Is the written English correct?</td>
<td>X</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Main technical terms are correctly referenced?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Glossary present in the document?</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Unai Aguilera  
unai.aguilera@deusto.es  
University of Deusto
### 12.1. EUROHELP

July 29th, 2016

<table>
<thead>
<tr>
<th>Issue</th>
<th>Yes</th>
<th>No</th>
<th>Score (1=low to 5=high)</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Is the architecture of the document correct?                         | X   |    | 5                        | X 5  
| Does the architecture of the document meet the objectives of the work done? | X   |    | 5                        | X 5  
| Does the index of the document collect precisely the tasks and issues that need to be reported? | X   |    | 5                        | X 5  
| Is the content of the document clear and well described?             | X   |    | 4                        | Some sections provide figures about the population process and others do not. |
| Does the content of each section describe the advance done during the task development? | X   |    | 3                        | The nature of the document is to explain the population process, not the advance done in the task. |
| Does the content have sufficient technical description to make clear the research and development performed? | X   |    | 4                        | X 4  
| Are all the figures and tables numerated and described?             | X   |    | 5                        | X 5  
| Are the indexes correct?                                             | X   |    | 5                        | X 5  
| Is the written English correct?                                      | X   |    | 4                        | X 4  
| Main technical terms are correctly referenced?                       | X   |    | 5                        | X 5  
| Glossary present in the document?                                    | X   |    | 5                        | X 5  

Aritz Rabadan

arabadan@euro-help.es

Eurohelp
13. ANNEX I – ETHICAL COMPLIANCE CHECK

This document focuses on the guidelines for the population of the WeLive framework that is preliminary to the execution of the first pilot. These guidelines mainly address the partners of the consortium and possible bodies that might adopt the framework in the future and have little to do with the final users of the framework. For this reason, this deliverable does not rise ethical concerns about Consent Form, Data Protection, and Terms of Use. Rather it involves the conduct that partners should follow when populating the platform and when providing guidelines for this task.

In particular the populating activities must be legitimate with respect to the WeLive Code of Conduct and the relevant guidelines must be compliant with it. The enforcement of such compliancy must be supported by the tools involved in the population. Where required this topic has been tackled within the deliverable (see for instance, Section 4.2). If not otherwise stated it is assumed that the guidelines are compliant with the WeLive Code of Conduct.

Detailed answers to the questions raised here are the following:

- Following point 2 and 4 of the WeLive Code of Conduct the selection of provisioned services chosen for populating the framework must be open and transparent and performed following ethical criteria.
- The guidelines for the population of the WeLive framework definitely support the WeLive Innovation Model in two ways. On the one hand, they represent a means for exemplifying the type of artifacts that are subject of the innovation process. On the other hand, they provide a first concrete element to trigger the innovation process itself.

2) What requirements does the new Data Protection Act sets for our work? Consent forms? Access to data and right to be forgotten? Transfer to third countries? Privacy by Design? The use of data for public purposes? The governance model and responsibilities? Hackering issues? (> see WeLive Data protection document, the New Data Protection Act and D8.6)

- The first population of the WeLive framework takes place in 2016 and, therefore, it is not subject to the prescription of the new Data Protection Act.
<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>3) How should we take into account WeLive Terms of Use in our development work? Should they be developed based on what we will do? (&gt; see WeLive Terms of Use and D8.6)</td>
<td>This question does not apply since the work described in this document has no link with the WeLive Terms of Use.</td>
</tr>
<tr>
<td>4) How should we take into account Consent Forms, data protection and authorizations in our research? Is it necessary to collect personal information? How is our data management? (&gt; see the D5.3, current templates for the Consent forms, D8.1 and D8.6)</td>
<td>These questions do not apply since the work described in this document does not focus on citizens’ engagement.</td>
</tr>
<tr>
<td>5) Is accuracy and precision of WeLive personal/other data an issue to be taken into our development work? (&gt; see the D5.3)</td>
<td>This question does not apply since the work described in this document does not focus on citizens’ engagement.</td>
</tr>
<tr>
<td>6) How should we make it possible for vulnerable people to also take part into development work? How are the Consent forms? Are the participating methods suitable? How about marketing material? (&gt; see the D5.3)</td>
<td>These questions do not apply since the work described in this document does not focus on citizens’ engagement.</td>
</tr>
<tr>
<td>7) Does the local data protection set requirements for our work? Does our work deal with data transfer to third countries? Do we need authorizations for the use of external data? (&gt; see local data protection act (and after 2018 the new Data Protection Act), D8.6 and D8.4)</td>
<td>Local data protection regulations impose constraints that must be fulfilled when exposing datasets onto the WeLive framework. The guidelines presented here suggest that the data management be updated when a dataset is added for populating the framework. When populating the platform with external data, particular care must be taken in providing the right acknowledgement and attribution to the owner or distributor of the data.</td>
</tr>
<tr>
<td>8) Is there any other issues which are relevant from the viewpoint of our work? If yes, discuss the situation with the Ethics Board before starting the work.</td>
<td>No other issue.</td>
</tr>
</tbody>
</table>