WEKIT D5.6
Repositories interfacing with the Legacy - Interactive Content Repository

Editors:
Roland Klemke (OUNL)
**Revision History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Contributor(s)</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17-07-2018</td>
<td>Daniele Di Mitri</td>
<td>Initial Outline</td>
</tr>
<tr>
<td>2</td>
<td>05-11-2018</td>
<td>Daniele Di Mitri</td>
<td>Additional changes, updated the format</td>
</tr>
<tr>
<td>3</td>
<td>07-11-2018</td>
<td>Daniele Di Mitri</td>
<td>Added content</td>
</tr>
<tr>
<td>4</td>
<td>20-11-2018</td>
<td>Roland Klemke</td>
<td>Provided feedback</td>
</tr>
<tr>
<td>5</td>
<td>21-11-2018</td>
<td>Daniele Di Mitri</td>
<td>Implemented feedback</td>
</tr>
<tr>
<td>5</td>
<td>30-11-2018</td>
<td>Daniele Di Mitri</td>
<td>Added comments from Will Guest and Paul Lefrere</td>
</tr>
<tr>
<td>6</td>
<td>03-12-2018</td>
<td>Mikhail Fominykh</td>
<td>Quality review, style and formatting</td>
</tr>
</tbody>
</table>

*Disclaimer: All information included in this document is subject to change without notice. The Members of the WEKIT Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the WEKIT Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.*
Repositories interfacing with the Legacy - Interactive Content Repository

WP 5 | D5.6

Editor:
Roland Klemke (OUNL)

Authors:
Daniele Di Mitri (OUNL)
Bibeg Limbu (OUNL)

Reviewers:
Chris Brox (LT)
Paul Lefrere (CCA)
Will Guest (OBU)
Mikhail Fominykh (EP)

<table>
<thead>
<tr>
<th>Deliverable number</th>
<th>D5.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination level</td>
<td>Public</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Date</td>
<td>30.11.2018</td>
</tr>
<tr>
<td>Due date</td>
<td>M36</td>
</tr>
</tbody>
</table>
# Table of Contents

REVISION HISTORY .......................................................................................................................... 2

EXECUTIVE SUMMARY .................................................................................................................. 5

1. INTRODUCTION .......................................................................................................................... 6

2. ARCHITECTURE ............................................................................................................................ 6

3. IMPLEMENTATION ......................................................................................................................... 7
   3.1. SYSTEM ENTITY ....................................................................................................................... 7
   3.2. USER AUTHENTICATION ......................................................................................................... 8
   3.3. WEKIT.ONE CLOUD API ....................................................................................................... 10
       3.3.1. Uploading a session .......................................................................................................... 10
       3.3.2. Generating list of sessions ............................................................................................... 11
       3.3.3. Downloading a session .................................................................................................... 11
       3.3.4. Fetching a session using the filename ............................................................................. 11
   3.4. FRONT END GUI ..................................................................................................................... 12
       3.4.1. List of ARLEM sessions .................................................................................................... 12
       3.4.2. List of LearningHub sessions ......................................................................................... 12
   3.5. INTERACTION WITH WEKIT.ONE RECORDER AND PLAYER .................................................. 13

4. FUTURE WORK .............................................................................................................................. 13
   4.1. EDITING SESSION METADATA .............................................................................................. 14
   4.2. ARLEM EDITOR ....................................................................................................................... 14
   4.3. LEARNING/HUB SESSION VISUALIZATION .......................................................................... 15
   4.4. SCALING-UP THE INTERACTIVE REPOSITORY .................................................................... 15

REFERENCES ..................................................................................................................................... 16
Executive summary

Deliverable D5.6, Repositories interfacing with the Legacy – Interactive Content Repository, describes the cloud-based content repository used as backend content storage and retrieval solution as part of the WEKIT.One platform. The deliverable introduces the role of the repository in section 1. It describes its architecture in section 2, and its implementation details in section 3. The deliverable concludes with an outlook on future work of the repository in the light of usability (editing features), data analysis (visualisation), and overall scalability.
1. Introduction

The WEKIT Interactive Content Repository, also defined in the WEKIT ecosystem as WEKIT.One Cloud, is the online storage of the content created with the WEKIT.One Recorder application and downloadable/usable for the WEKIT.One Player component. The version of the WEKIT.One Cloud has been documented in the deliverable D3.6 Software Prototype with Sensor Fusion API Specification and Usage Description in section 3.3. In this deliverable, the WEKIT.One Cloud is placed within the general WEKIT architecture ecosystem.

WEKIT.One Cloud is an online content repository where the sessions recorded are stored, added with appropriate metadata and retrieved for later being replayed in the WEKIT.One Player application. WEKIT.One Cloud implements two main components a Graphic User Interface (GUI) and an interactive Application Programming Interface (API). The contents stored in the interactive WEKIT.One Cloud are of two kinds:

- WEKIT session files, serialized into ARLEM file format compressed with their dependencies into a .zip folder.
- Sensor combined recordings, coming from the WEKIT.One Hub, which is the sensor integration component of WEKIT.One.

The ARLEM data format is described in D2.2 and D2.3, while the data formats of the WEKIT.One Hub is detailed in D3.6 (section in 3.2).

2. Architecture

The architecture of WEKIT.One Cloud is represented in Figure 1. WEKIT.One Cloud is composed by the 1) Front-end GUI and the 2) Back-end application.

The Front-end GUI is generated at client-side and it is meant to facilitate the user-interaction. The purpose of the GUI is mainly to allow the experts using the recorder to have a desktop and user-friendly interface to check if the content saved in the repository is meeting the expectations. As the content stored in the repository are of two types, the GUI comes with two interfaces, 1) ARLEM session list and the 2) Hub recordings.

The Back-end application is the server-side component of WEKIT.One Cloud. The back-end application is designed as layered architecture composed with different services: 1) the authentication service, 2) the API which, and the 3) Cloud Storage API.

The Authentication service works using an OAuth server, the main purpose is to provide secure access to the repository for authorised users, while preventing unauthorised access.

The WEKIT.One Cloud API is the custom designed API which implements all the connection endpoints of the WEKIT.One Cloud and allows the WEKIT Recorder and Player to make use of the functionalities of the repository.

The Cloud Storage API is the proprietary API which handles the storage of files internally into the Cloud system to store the two different kinds of files as mentioned before: the Sensor recordings and the ARLEM sessions.
Figure 1. Architecture diagram of WEKIT.One Cloud

3. Implementation

WEKIT.One Cloud is implemented through a Java RESTful web service, which uses servlets running on a Google App Engine instance (standard environment). This application uses Objectify and GSon, libraries which facilitate in Java web projects the coupling between the entities in the online database (Google Datastore) and the classes.

The advantage of using a Platform as a Service as Google Application Engine allows the application to use a scalable computing infrastructure running in the cloud. In the standard environment, the AppEngine provides moreover this cloud environment for free.

3.1. System entities

The entities of WEKIT.One Cloud are: Session, AccessToken and AccessDownload. We first describe the entity session as the reminders are discussed in the next session concerning the User Authentication.

The entity **Session** is based on the upload of the main file (compressed folder) of the session plus relevant metadata. The Session has the following attributes:

- **Id** - unique identifier of the session in the Datastore
- **Author** - the text field containing a reference to the author that uploaded the session
- **UploadingDevice** - is the device which uploaded the session
In the Listing 1 and Listing 2, two different examples are provided of instantiations of Sessions. As noticeable, the first one example as an additional attribute “category” specified as ARLEM.

```
{
   "id": 5767216922361856,
   "author": "User #1",
   "uploadingDevice": "Hololens",
   "description": "itmo",
   "category": "ARLEM",
   "size": 4398767,
   "filename": "session-2018-11-06_13-57-30.zip",
   "filetype": "application/x-zip-compressed",
   "key": "/gs/wekitproject.appspot.com/L2FwcGhvc3Rp...",
   "uploadingDate": "Nov 6, 2018 1:59:05 PM"
}
```

Listing 1 - Example of an instance of an ARLEM session.

```
{
   "id": 5755553334689792,
   "author": "DESKTOP-0F322KN",
   "uploadingDevice": "LearningHub 02150308C2FE",
   "description": "KinectReader_",
   "size": 533807,
   "filename": "2018-11-7-12H1M48S36.zip",
   "filetype": "application/x-zip-compressed",
   "key": "/gs/wekitproject.appspot.com/L2FwcGhvc3Rp...",
   "uploadingDate": "Nov 7, 2018 11:03:08 AM"
}
```

Listing 2 - Example of an instance of a Sensor session

### 3.2. User authentication

To implement secure access to the repository we use the LearningLayer OpenId Connect server of the WEKIT Community platform. OpenID Connect (OIDC) is an authentication layer on top of OAuth 2.0, an authorization framework.
The flow of authentication is shown in Figure 2 and consists of five steps:

Step 1) When accessing the repository homepage (https://wekitproject.appspot.com/), the user generates an Authentication and Authorization Request to login into the system provided by LearningLayers OIDC. The links connect to the Authorization Endpoint. This request consists of a response type, the scope of the request, the client id, and the redirect URI.

Step 2) If the request comes from a session unknown to the LearningLayers OIDC server, the user will be asked to log in as shown in Figure 3. If the user does not have an account in the WEKIT Community plugin, he/she needs to create one. Hence, the user can fill in the credentials for the login. The Authentication Authorization is handled internally in the LearningLayers OIDC server.

Step 3) After authorization, the LearningLayers OIDC server redirects to the redirect URI provided in step 1.

Step 4) The client receives the Authorization code and compose a Token Request to the Token endpoint, this time providing the grant type, the code received in step 3, the redirect URI and the client secret code.

Step 5) The server responds with the token and expiration date of this token. Having obtained the Access token and the expiration date, we then store the access information in the Datastore WEKIT.One Cloud. We have created two entities AccessToken and AccessDownload as shown in Figure 4. In AccessToken, we store along with the Token received from the OIDC server also the
timestamp, the user email and the username. In AccessDownload, we log instead of the download of the sessions. This functionality allows to track the files downloaded by the users.

Figure 4. The Datastore Entities to log the user activities.

3.3. WEKIT.One Cloud API

The WEKIT.One Cloud API is designed to connect the WEKIT application with the storage backend. It allows communicating recording sessions between application and cloud storage. A session comprises all data recorded, annotated, and edited. The following subsections detail the implemented functions for the /storage REST API of WEKIT.One Cloud, which comprises uploading, listing, editing, and downloading sessions.

<table>
<thead>
<tr>
<th>Method description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Uploading a new session to the repository</td>
<td>/storage/requestupload</td>
</tr>
<tr>
<td>2) Generating a list of sessions</td>
<td>/storage/sessions</td>
</tr>
<tr>
<td>3) Downloading a session (using blob key)</td>
<td>/storage/serve/&lt;blobKey&gt;</td>
</tr>
<tr>
<td>4) Fetching a session (using filename)</td>
<td>/storage/fetch/&lt;filename&gt;</td>
</tr>
</tbody>
</table>

3.3.1. Uploading a session

A session can be uploaded by an authorised client in two steps.

Step 1) request to Google Blobstore a temporary token for the /storage.

- **URL**: /storage/requestupload
- **Method**: POST/form-data
- **URL Params:**
  - **Success response:**
    - Code: 200
    - Content: <the_upload_URL>

Step 2) Upload the session compressed folder (e.g., in Zip file) to the Google Blobstore using <the_upload_URL> generated in step 1:

- **URL**: <the_upload_URL>
- **Method**: POST/form-data
3.3.2. **Generating list of sessions**

The list of session can be retrieved the following method. A single session can be retrieved with:

- **URL**: /storage/sessions
- **Method**: GET
- **URL Params (optional)**:
  - category = [string]
  - id = [integer]
- **Success response**:
  - Content: <JSON_list>

3.3.3. **Downloading a session**

To download a session the Blob key is necessary. That can be found in each item of the `<JSON_list>` with the attribute "key". The Blob key has the following format

/gs/wekitproject.appspot.com/<long_Alphanumeric_Token>.

- **URL**: /storage/serve/<blobKey>
- **Method**: GET
- **Success response**:
  - Content: <binary_file>

3.3.4. **Fetching a session using the filename**

To download a session the Blob key is necessary. That can be found in each item of the `<JSON_list>` with the attribute "key". The Blob key has the following format

/gs/wekitproject.appspot.com/<long_Alphanumeric_Token>.

- **URL**: /storage/fetch?filename=
- **Method**: GET
- **URL Params (required)**:
  - filename = [string]
- **Success response**:
  - Code: 200
  - Content: <binary_file>
3.4. Front end GUI

Besides the API, which can generate content into JSON file, we implemented two Graphic User Interfaces, which allow consulting the sessions uploaded in the repository in rich HTML format. The first one is specific for the ARLEM sessions, while the second one – for the LearningHub recordings.

Both the GUIs were developed using Javascript, Jquery and the DataTable library. This library allows to take as input JSON list and render the information in rich HTML format. With the help of JQuery and Javascript, it is also possible to sort each attribute of the table by clicking on the header, search for a specific term and navigate with the buttons on the bottom-right. Datatable handles also the pagination (the division of the results into pages) of the results client-side.

3.4.1. List of ARLEM sessions

At the URL /storage/list/ it is possible to access the list of ARLEM sessions. A screenshot of the interface is shown in Figure 5 below. The table has six columns: Uploading date, Author, Device, Description, Filesize and Download link.

![Screenshot of the Cloud Repository session list](image)

**Figure 5.** Screenshot of the Cloud Repository session list

3.4.2. List of LearningHub sessions

At the URL /learninghub/list/ it is possible to access the user-friendly list of ARLEM files. The table has eight columns: Idfile, Author, Device, Description, Filesize, Filetype Download link and Uploading date (Fig. 6).
### 3.5. Interaction with WEKIT.One Recorder and Player

An important functionality of the WEKIT.One Cloud Repository is to work as a bridge between the WEKIT.One Recorder and the WEKIT.One Player, respectively responsible for 1) the recording of the wearable enhanced learning experiences from the expert and 2) replaying the recorded experience to the learner. This linkage is possible using the WEKIT.One Cloud API described in section 3.3 in this deliverable.

After having recorded one expert session, the WEKIT.One Recorder implements the API call Upload of the Session and it adds all the relevant metadata. When a learner wants to replay the session recorded by the expert, starts the WEKIT.One Player, which through a dedicated GUI displays a user-friendly list of all the available recordings. This list is created by the API call List of sessions which displays the relevant metadata to the user and adds the link for directly downloading the session via the API. Once the selected session is clicked, the session is downloaded into the WEKIT.One Player and is ready to be replayed.

### 4. Future work

The current version of the WEKIT.One Cloud repository can be expanded with further functionalities which we consider future work for the project.
4.1. Editing session metadata

Above all, an interesting addition would be to allow the user to better edit the uploaded sections. One possibility is to allow the user to click on the section in the graphic user interface and change the metadata as shown in Figure 7.

Another possibility to implement in the future is to allow the user to upload a session through a dedicated GUI, capability which is now only possible through the API.

![Figure 7. Possible interface for editing a session uploaded in the Repository](image)

4.2. ARLEM editor

As part of a master thesis at RWTH Aachen, a prototype for an ARLEM editor has been developed, that is capable of visualising an existing ARLEM file, allowing users to modify it in a safe way. The prototype of the ARLEM editor has been implemented as a separate tool so far, that lacks full ARLEM support and that misses integration into the WEKIT.One platform yet. Figure 8 below displays a view of the ARLEM editor for editing ARLEM workplace files.

![Figure 8. ARLEM Editor prototype](image)
4.3. LearningHub session visualization

As a future component for the Interactive Repository, we envision the development of a Visual Inspection Tool. This tool would allow to load, visualize and annotate different sessions of the LearningHub. Such component should be able to visualize data coming from multiple sensor applications, triangulate these data with the video recording of the session of and select different time-intervals on these data. Hence, add custom attribute-value annotations to this intervals. This manual annotation procedure is thought of as an ex-post activity for the expert to reason about the collected data and bridge sensor performance of the learner with the task execution. In figure 9, we show an initial prototype of the Visual Inspection Tool.

![Figure 9. Snapshot of the first version of the Visual Inspection Tool.](image)

4.4. Scaling-up the Interactive Repository

Another aspect which needs additional investigation is adapting the Interactive Repository to handle hundreds or thousands (or more) of recordings. Surely the selected cloud infrastructure allows to scale the number of the sessions to a very large number. However, the architecture of Wekit.One Cloud has to be reconsidered. In particular, filtering mechanisms have to be set in place in order to avoid loading all the sessions each time the user accesses the list. This requires smart indexing of the sessions similarly to the one YouTube is using. To make it more efficient, metadata have to be redesigned to be more descriptive of the session and give the user better chance to search and find the desired one.

Additional functionalities of the Interactive Repository can implement assessment of the recorded sessions. Suppose the repository contains multiple instances of expert-level demonstrations of a common manufacturing process. When a human looks at those instances, it is clear that parts of the demos are identical, and parts are different. Some differences can be at surface level and may be unimportant (e.g., are recorded from a different position), some demos are substantially similar at a deeper level. In this case, the human viewer can immediately spot that both involve monitoring the same set of key performance indicators (KPI), but the two lists have the KPIs in different orders and maybe use different tags. These issues are to be explored in future research into the repository.
References

WEKIT Deliverable D2.2 Learning Experience Content Model

WEKIT Deliverable D2.3 Final Architecture and Learning Experience Content Model

WEKIT Deliverable D3.6 Software Prototype with Sensor Fusion API Specification and Usage Description
