TROMPA: Towards Richer Online Music Public-domain Archives

Deliverable 5.5

Annotation Tools

Grant Agreement nr	770376
Project runtime	May 2018 - April 2021
Document Reference	TR-D5.5-Annotation Tools v2
Work Package	WP5 - TROMPA Contributor Environment
Deliverable Type	R - Report
Dissemination Level	PU - Public
Document due date	28 Feb 2021
Date of submission	7 May 2021
Leader	UPF
Contact Person	Alastair Porter (alastair.porter@upf.edu)
Authors	Alastair Porter (UPF), David Weigl (MDW), Federico Zubani (GOLD)
Reviewers	Jordi Janer (VL)

Executive Summary

This deliverable provides functionality for Task 5.5 of the TROMPA project (Annotation tools). Annotations are important for TROMPA through a wide cross-section of participants and use-cases. For example, music enthusiasts give feedback on how they perceive music and how it makes them feel. Music scholars share notes and insights among themselves to improve their understanding of compositions and performances. Instrumental players annotate scores as personal aide-memoires during rehearsal and share them in pedagogical contexts. Annotations also help us to improve the functionality of automatic systems. Such systems may need annotations to use as ground truth for building automatic classifiers. This deliverable presents the requirements of an annotation tool that can be used in TROMPA to collect annotations from users for a number of the end-user pilots that are being developed in the project. It presents a data model for storing the collected annotations in a way that integrates with the existing tools that have already been built to support the research taking part in TROMPA, for example the Contributor Environment (CE). In this deliverable we focus on annotations of audio, however we have worked closely with other project partners who require the annotation of other modalities (for example, Task 3.4, analysis and annotation of music scores) in order to ensure that our work in the design of the annotation schema can also support annotations for these other types of data. Aspects of the presented tooling are integrated with the Digital Score Edition component (D5.2) to support cross-modal annotations anchored in music score encodings, and with the prototype for music enthusiasts (D6.7) for soliciting feedback from users and storing it as annotations, and using these annotations to train models to recommend new music for a user to listen to.

The software and processes produced as a part of this deliverable are designed to be used by the software demonstrators developed as part of TROMPA in WP 6. This software is designed to be reusable, both by the use cases for which we plan to develop software, and also as an open software library that can be used freely by anyone who has a need for similar annotation tasks. We have to ensure that what we produce will fulfil the requirements of the members who need to integrate it into their demonstrators. We provide functionality for the Performers, Singers, Music enthusiasts, and scholars pilots. We determined a set of annotation tasks needed by all pilots. Annotations should be able to be applied to different types of media (audio, images, music scores), and should reflect the kind of operations that people make on physical representations of the same items. The values of annotations could be markers or indicators, freely chosen text or text from a fixed vocabulary, or ratings on a scale such as quality of a music performance or similarity between two items. The tools that are generated as a part of this task integrate with the rest of the TROMPA project, including the Contributor Environment (CE). Annotation tools are built using standard Web technologies (Javascript and HTML). Components and functionality from the annotation tools can be used independently, there is no requirement to use all of the functionality at once. Annotations are stored in a format that allows for the maximum compatibility with other annotation tools and other software developed as part of the project, including the CE.

We identify two different annotation workflows that should be supported. The first, task-driven annotation tasks, represent situations where the creator of the task explicitly defines which annotations they want users to create. For example, the music enthusiasts pilot will ask users to listen to a single pre-defined music recording and ask them to describe a specific set of aspects of the recording, often using a controlled vocabulary. User-driven annotation tasks are used in pilots where the task is more free-form and the user has the ability to decide which items or item fragments to

annotate, and what the contents of the annotation should be. Within the TROMPA project some annotation tasks should result in annotations that are private and can only be read by the user who created them. Other annotations can be public, and some tasks also have a workflow where annotations can be initially created privately but then published to an open repository at a later stage.

Within the TROMPA consortium we have experience storing similar types of annotations using the Web Annotations Data Model (Web Annotations) standard (Weigl and Page 2017). This model defines a structured format for expressing annotations of connected resources. As the Contributor Environment also implements linked data models, using Web Annotations is a good choice for a standard that can be easily integrated into existing tools within the TROMPA project. The TROMPA Contributor Environment storage system uses neo4j, a graph database which is suited to storing information represented by Linked Data structures. We extended the data model of the CE, allowing users to store annotations following the Web Annotations schema. The Annotation type is the core type representing a single annotation. Annotation targets can refer to any resource identifiable by a URL, including a node in the CE, or a separate item on the Web. Targets which are nodes in the CE can represent the actual node (e.g., an Annotation), or the file which is identified by a field in a node in the CE (e.g., the URL at the contentUrl field in an AudioObject node). URLs can optionally include fragments identifying a sub-part of the content at the URL (e.g. bars in a music score, regions in an image, or parts of an audio recording). An annotation's body is either an external URL or a node in the CE. Bodies can be text or other nodes in the CE, including ratings or pre-defined terms from a fixed vocabulary, or even other Annotations. An annotation should always have a motivation set. The Web Annotations standard describes the 13 basic types of motivations. Users can create their own custom motivations which more specifically describe the reason for the annotation. These custom motivations should be specializations of one of the base types, so that other software tools that display annotations can fall-back to this base type when displaying the annotation if they don't know how to render a specific motivation.

A *fixed vocabulary* defines a constrained set of textual tags that can be used as the body of an annotation. For example, an annotation task may involve indicating if the mood of a song is *happy* or *sad*. We provide the ability to define fixed vocabularies. *Rating templates* can be created to indicate the scale of assessments made by users, i.e., the lower and upper bounds that a rating value can take. When a Rating is created as the body of an annotation, it should be linked to the template that was used with the prov:wasDerivedFrom relationship. Some annotation use-cases require developers to group together a number of related annotation markings into an annotation *palette* (Lewis, Weigl, & Page, 2019). For example in the context of music performance analysis for string players one might want to identify when in a score to use pizzicato or arco, or add other performance instructions. A toolkit is a collection of annotation motivations, fixed vocabularies, or Rating definitions. An *annotation session* is a container used to logically group together a collection of annotations of one or more documents. For example, a music scholar may start an annotation session in order to annotate the collected works of a composer. All annotations that they make are added to the session. An annotation session could contain only annotations by a single user or by many users, and could contain annotations targeting only a single object, or many different objects.

In addition to the implementation of the annotations model in the Contributor Environment, the trompace-client python library was updated to provide library functions for creating custom motivations, annotation palettes, annotation toolkits, rating templates, ratings, and Annotations.

We developed a software library, to help with the creation and storage of annotations. The library includes functions to create annotation objects in either the CE or a Solid Pod, functions to to convert annotation representations between Solid Pods and the CE, graphical components for loading audio files from the CE or external locations (for example a Solid Pod), a graphical interface to display a graphical representation of audio files and make annotations at points in time on this representation and React components to create annotation palettes and annotation toolkits and make annotations of different types. Developers can pick and choose which components they wish to use, all items are optional and standalone. We provide some examples of demo applications that join together multiple components.

The audio player component is a wrapper around the open source library wavesurfer. This component can load an audio file and display it graphically as a waveform. The component has an audio player that can be used to play back the audio file which is displayed. By clicking and dragging the mouse on the audio waveform, an annotation can be made that spans a time range. We provide React components for entering text annotations, ratings, and other types of annotations that can be stored using the data model. The palette and toolkit editor provides a graphical user interface for creating fixed term vocabularies, annotation palettes and annotation toolkits. When creating a vocabulary or palette, the user can specify a name, and add child items. The toolkit editor allows a user to create a list of annotation motivations, or add a specific vocabulary, palette, or rating template. The annotation session viewer component can show a list of all annotation sessions created by a user. When a session is selected, it lists all annotations that have been saved to that session. In its current state, the annotation session viewer can only display a session if it contains items made by one user and of one audio recording.

We implemented a search function for annotations that can be integrated into the Multimodal Component (D5.3). This allows users to use the multimedia component to search for annotations based on textual content of the annotation body, the value of tags, or the name of the object that is annotated. We provide the ability for annotations to be written to a private Solid Pod initially, and then *published* to the CE. When an annotation is published to the CE it can be linked to the original annotation to indicate the canonical version.

We developed a number of demo applications that show how each component works: Editors to create fixed-term vocabularies, annotation palettes, annotation toolkits, and rating templates, an example of basic (one file and one annotation type) and more complex (multiple files and types) annotation tasks; saving annotations to the CE, Solid Pods, and publishing annotations from Solid Pods to the CE.

The schema and software described in this deliverable has been integrated into the working prototypes for music enthusiasts, scholars, and instrument players.

Version Log		
#	Date	Description
v0.1	26 April 2019	Initial version submitted for internal review
v0.2	30 April 2019	Revised version after internal review
v1.0	30 April 2019	Final version submitted to EU
v1.1	30 April 2021	Initial version submitted for internal review
v1.2	7 May 2021	Revised version after internal review
v2.0	7 May 2021	Final version submitted to EU

Table of Contents

1. Introduction

This deliverable provides functionality for Task 5.5 of the TROMPA project (Annotation tools). In Task 5.5 we consider “annotation acquisition, feedback to the user, and crowd planning strategies to dynamically adapt the behaviour of the systems as response to the quantity and quality of completed tasks or to the availability and reliability of crowd contributors”. Annotations are important for TROMPA because they help us to enhance existing collections of music, adding feedback and insights from the different types of audiences who we target. For example, music enthusiasts give feedback on how they perceive music and how it makes them feel. Music scholars share notes and insights among themselves to improve their understanding of compositions and performances. Instrumental players annotate scores as personal aide-memoires during rehearsal, optionally sharing them, e.g., in pedagogical contexts. From a technical perspective, annotations help us to improve the functionality of automatic systems that we are developing in TROMPA to automatically annotate and analyse music works. Automatic systems may need annotations to use as *ground truth* for building automatic classifiers, requiring a number of examples of each type of annotation such as performance difficulty.

This deliverable, D5.5 is concerned with implementing the features described in Task 5.5. We will focus on three aspects concerned with the implementation of an annotations platform:

❖ The development of a schema and storage format for annotations
❖ The development of an interactive tool that runs in a web browser for annotating audio files
❖ The development of tools for creating and storing annotations about other types of media files

This deliverable presents the requirements of an annotation tool that can be used in TROMPA to collect annotations from users for a number of the end-user pilots that are being developed in the project. It presents a data model for storing the collected annotations in a way that integrates with the existing tools that have already been built to support the research taking part in TROMPA, for example the Contributor Environment (CE). In this deliverable we focus on tools for annotations of audio, however we have worked closely with other project partners who require the annotation of other modalities (for example, Task 3.4, analysis and annotation of music scores) in order to ensure that our work in the design of the annotation schema can also support annotations for these other types of data. Aspects of the presented tooling are integrated with the Digital Score Edition component (D5.2) to support cross-modal annotations anchored in music score encodings, and with the prototype for music enthusiasts (D6.7) for soliciting feedback from users and storing it as annotations, and using these annotations to train models to recommend new music for a user to listen to.

1.2 Document structure

The remainder of this document is as follows: Section 2 describes the requirements for annotations within the scope of the TROMPA project. Section 3 describes the web annotations schema and how we implemented it using tools available in TROMPA. Section 4 outlines the javascript tools that were developed to take user input and create annotations, and Section 5 describes how these tools and the developed schema are used in other use-cases within TROMPA.

2. Requirements

The software and processes produced as a part of this deliverable are designed to be used by the software demonstrators developed as part of TROMPA in WP 6. This software is designed to be reusable, both by the use cases for which we plan to develop software, and also as an open source software library that can be used freely by anyone who has a need to perform similar annotation tasks. As the result of this deliverable will be used by other members within the TROMPA consortium we ensured that what we produced fulfils the requirements of the members who need to integrate it into their demonstrators. We provide functionality for the following end user pilots:

- ❖ Performers
- ❖ Singers
- ❖ Music enthusiasts
- ❖ Scholars

In discussion with the task leaders of the relevant end user pilots we determined requirements for the following types of annotation tasks:

Performers (MDW)

- ❖ Annotations that serve solely to identify a particular part of a score to make it salient during rehearsal (analogous to circling notes in a paper score)
- ❖ Free-form text labels that can be applied to a point in time in a musical score or recording
- ❖ closed-vocabulary annotations around performative aspects (e.g., fingerings, dynamics)

Singers (UPF, VL)

- ❖ The rating of the difficulty to sing an entire musical score or recording (on a difficult scale)
- ❖ The performance rating of an aspect of audio segment (on a scale, or a binary yes/no annotation)
- ❖ The similarity between two sections of recordings

Music enthusiasts (UPF)

- ❖ Folksonomy tagging of audio recordings or segments
- ❖ Free-form text labels applied to audio recordings or segment
- ❖ The assessment of some aspect of an audio recording on a fixed scale

Scholars (GOLD)

- ❖ Storage of annotations made on music scores
- ❖ Linking of items in a music score to a part of a music recording

The tools that are generated as a part of this task integrate with the rest of the TROMPA project, including the Contributor Environment (CE). In order to integrate with other parts of the overall project we include these additional non-functional requirements:

- ❖ End-user pilots are developed as web applications, and so the annotation tools are built using standard Web technologies (Javascript and HTML).
- ❖ Not all tasks require all functionality. Some tasks need to display audio files and have interfaces for capturing user input, but others may only need to use the storage model in the CE, or send annotations created using custom interfaces with a library method.
- ❖ Annotations must be relatable to items that exist in the CE
- ❖ It should be known who performed an annotation (identifying people with some standardised user id system). Different users can annotate the same item

- ❖ The storage and retrieval of annotations should be in a format that allows for the maximum compatibility with other annotation tools and other software developed as part of the project.

Based on the requirements from the use cases, we identify two different annotation workflows that should be supported. The first, *task-driven annotation tasks*, represent situations where the creator explicitly defines which annotations they want users to create. For example, the music enthusiasts pilot asks users to listen to a single pre-defined music recording and asks them to describe a specific set of aspects of the recording, often using a controlled vocabulary. These tools only require the data model, functionality to submit annotations to the CE, and some graphical components for gathering user input. *User-driven annotation tasks* are used in pilots where the task is more free-form and the user has the ability to decide which items or item fragments to annotate, and what the contents of the annotation should be. For example, the music scholars pilot provides a free-form interface where the user can define the kinds of annotations that they want to make on a music score, and then let them create annotations according to that schema. Such an interface should provide the ability for users to create templates of the types of annotations that they want to make, and also allow annotations to be grouped together to form a logical annotation *session*, allowing the user to make some annotations, save them, and then return at some later stage to create more annotations or refine existing annotations.

Within the TROMPA project some annotation tasks should result in annotations that are private and can only be read by the user who created them. Other annotations can be public, and some tasks also have a workflow where annotations can be initially created privately but then published to an open repository at a later stage (Weigl et al., 2020). Application developers should be able to choose if they want to support saving annotations privately to a Solid pod, publically to the CE, or a hybrid approach allowing users to select previously created private annotations and publish them to the CE.

3. Annotation Schema

This section describes the schema that was developed to store annotations in the TROMPA project. It assumes a basic knowledge of the Contributor environment, GraphQL, and the underlying data structures present in the CE[1].

A number of different use-cases and pilots in the TROMPA project need to store annotations of various different types of multimedia. Within the TROMPA consortium we have experience storing similar types of annotations using the Web Annotations Data Model (Web Annotations) standard (Weigl and Page 2017). This model defines a structured format for expressing annotations of connected resources. As the Contributor Environment also implements linked data models, using Web Annotations is a good choice for a standard that can be easily integrated into existing tools within the TROMPA project.

Key concepts in Web Annotations can be linked to content that we are working with in TROMPA:

- ❖ An **ID** (unique identifier, externally addressable as a URL using the CE's JSON-LD wrapper) representing the annotation
- ❖ One or more **Targets** which refer to items being annotated, represented as URLs. These would represent items in the CE, which can already be identified by a URL. Parts of items

[1] For more information see D5.1

such as a time range of a recording can be represented by the media fragments model[2]. Optionally, targets can also refer to external URLs.

- ❖ Zero or more **Bodies** which contain information that the annotation associates with its target(s). For example, a tag, description, or rating. Optionally, bodies can also refer to external URLs.
- ❖ A **Motivation** describing the purpose for which the annotation was made. The Web Annotation specification defines a number of fixed motivations[3], but this list can be extended by use of the Simple Knowledge Organization System model (SKOS)[4]

3.1 Core annotation concepts

The TROMPA Contributor Environment storage system uses neo4j[5], a graph database which is suited to storing information represented by Linked Data structures, like Web Annotations. We extended the data model of the CE, allowing users to store annotations following the Web Annotations schema. This extension includes the implementation of some core Types for storing annotations, as well as the implementation of a number of additional types defined in schema.org that are useful for defining and storing annotations. Where possible, we maintain semantic links within the neo4j database between annotations and related items (for example existing nodes which are the target of an annotation). This section describes how the concepts in the Web Annotations model map to entities made available in the GraphQL API.

3.1.1 Annotations

The `Annotation` type is the core type representing a single annotation. Depending on the type of data being annotated, this type can contain fields that link to external URLs, or it can be related to other objects in the CE. The creator field should be set to an identifier of the agent who created the Annotation, for example a user's Solid vCard URL, or the url of a software agent.

3.1.2 Annotation targets

Annotation targets can refer to any resource identifiable by a URL. This means that the item being annotated could be a node in the CE, or it could be a separate item on the Web. To store both types of targets, an Annotation object has two target fields, `targetUrl`, and `targetNode`. If the target is an external resource, any number of URLs can be added to the `targetUrl` field when creating the Annotation object. If the target of the annotation is a node that exists in the CE, a link should be made using the `AddAnnotationTargetNode` mutation to an `AnnotationCETarget` node.

The `AnnotationCETarget` type allows for a flexible approach in setting a node in the CE as the target of an annotation. For example, given an `AudioObject` node whose contentUrl field contains a URL to a music recording, an annotation can target this URL by creating an `AnnotationCETarget` node with the `target` field linked this AudioObject node (via the `AddAnnotationCETargetTarget` mutation), the `field` field set to "contentUrl", and if necessary, the `fragment` field set to a valid media fragment if the annotation is to target only a portion of the given music recording (e.g. t=60,120 to refer to the part only between 1 and 2 minutes). In the case that the target of the annotation is a node in the CE itself, the `field` field can be omitted. For

[2] https://www.w3.org/TR/media-frags/
[3] https://www.w3.org/TR/annotation-vocab/#motivation
[4] https://www.w3.org/2009/08/skos-reference/skos.html#broader
[5] https://neo4j.com/

example, a user may want to make an annotation about an Annotation object that exists in the CE. In this case, the target would be an annotation object. This data model allows us to maintain the semantic relationships between items in the CE, making it easy to identify, for example, all annotations that target a given audio recording or music score.

3.1.3 Annotation Bodies

Like targets, an annotation's body can also refer to either an external URL or a node in the CE. In the case of an external URL, set the `bodyUrl` field. If the body should be an existing node (for example, a tag from a fixed vocabulary as described in Section 3.2.1), the `AddAnnotationBodyNode` mutation can be used. An annotation's body can be text provided by the user. In this case, an AnnotationTextualBody can be created. This corresponds to the TextualBody definition in the Web Annotations standard. An `AnnotationTextualBody` has a value field, and optional fields for `format` (i.e., mimetype) and `language`. In the case of a plain text body, the format field should be set to text/plain or omitted. A body of HTML markup, for example would have the format set to text/html. A textual body can be joined to an Annotation with the `AddAnnotationBodyText` mutation. If an annotation assesses some quality of a target, e.g., the accuracy of the tonality of a performance, or how much someone likes the recording on a scale of 1-10, a `Rating` object can be used as the body of the annotation. The Rating contains fields for the `ratingValue`, `worstValue`, and `bestValue`. It can also contain an optional `ratingExplanation` explaining why the value was given.

3.1.4 Annotation Motivations

An annotation should always have a motivation set. The `AnnotationMotivation` enumeration describes the 13 basic types of motivations present in the Web Annotations standard. Users can create their own custom motivations which more specifically describe the reason for the annotation. These custom motivations should be specializations of one of the base types, so that other software tools that display annotations can fall-back to this base type when displaying the annotation if they don't know how to render a specific motivation. The relationship between a custom motivation and the base motivation is made using the skos:Broader relationship. The CE Annotations model supports two kinds of custom motivations. One is the AnnotationCEMotivation type, which allows a new motivation to be created in the CE and referenced in annotations. This can be joined to an annotation using the `AddAnnotationMotivationNode` mutation. The second is an annotation palette, described in Section 3.2.3, which can be joined to an annotation using the `AddAnnotationMotivationDefinedTerm` mutation.

3.2 Other models

3.2.1 Fixed Vocabularies

A fixed vocabulary defines a constrained set of textual tags that can be used as the body of an annotation. For example, an annotation task may involve indicating if the mood of a song is *happy* or *sad*. In order to know that the body of two different annotations refers to the same concept and don't just happen to share the same value, we provide the ability to define fixed vocabularies. A fixed vocabulary is created using the `DefinedTermSet` type[6]. A vocabulary is represented by a

[6] https://schema.org/DefinedTermSet and https://schema.org/DefinedTerm

DefinedTermSet with a custom additionalType set[7] in order to define its purpose, while each term in the vocabulary is a DefinedTerm with a custom additionalType[8]. The value of the item in the vocabulary is set with the termCode field. When creating an annotation which uses one of these values as the body, the relevant DefinedTerm should be set using the AddAnnotationBodyNode mutation.

3.2.2 Rating templates

In order for an annotation interface to present an input to a user to enter a rating, it needs to know what the scale of the rating is, i.e., what the minimum and maximum values are. Additionally, different people should be able to perform ratings which refer to the same concept. To represent this, a *rating template* can be created. This is a Rating object with a custom additionalType[9]. This object has the worstValue and bestValue fields set, but no ratingValue. When a Rating is created as the body of an annotation, it should be linked to the template that was used with the prov:wasDerivedFrom relationship, using the AddRatingWasDerivedFrom mutation.

3.2.3 Annotation palettes

Some annotation use-cases require developers to group together a number of related annotation markings into an annotation *palette* (Lewis, Weigl, & Page, 2019). For example in the context of music performance analysis for string players one might want to identify when in a score to use pizzicato or arco, or add other performance instructions. A DefinedTermSet can be created with a custom additionalType in order to define their role within the CE[10]. DefinedTerms within this DefinedTermSet also have an additionalType to define their CE role[11] and must also have an additionalType of oa:Motivation set. By requiring such DefinedTerms to also be represented as annotation Motivations, the object can be used directly in the motivation field of an annotation. In this case, the annotation motivation of which this DefinedTerm is a more specific version should be specified in the broader field. DefinedTerms can set termCode (text representation) and/or image (URL to an external image for a graphical representation) depending on how the user wants them to be rendered.

3.2.3 Annotation toolkits

An annotation task may require a user to perform many different types of annotations on a single item. In order to group together these tasks, an annotation toolkit can be created. A toolkit is a collection of annotation motivations, DefinedTermSets, or Rating definitions. An annotation toolkit is stored as an ordered or unordered ItemList[12], with an additionalType[13]. Each ListItem in the ItemList has an item field pointing to a DefinedTermSet or Rating (in the case that the annotation has a schema), or an AnnotationCEMotivation if the item in the toolkit is a

[7] https://vocab.trompamusic.eu/vocab#TagCollection
[8] https://vocab.trompamusic.eu/vocab#TagCollectionElement

[9] https://vocab.trompamusic.eu/vocab#RatingDefinition

[10] https://vocab.trompamusic.eu/vocab#AnnotationMotivationCollection
[11] https://vocab.trompamusic.eu/vocab#AnnotationMotivationCollectionElement

[12] https://schema.org/ItemList
[13] https://vocab.trompamusic.eu/vocab#AnnotationToolkit

Motivation. The software tools that we developed can read an annotation toolkit definition to determine which input components to show to a user to allow them to create their annotations.

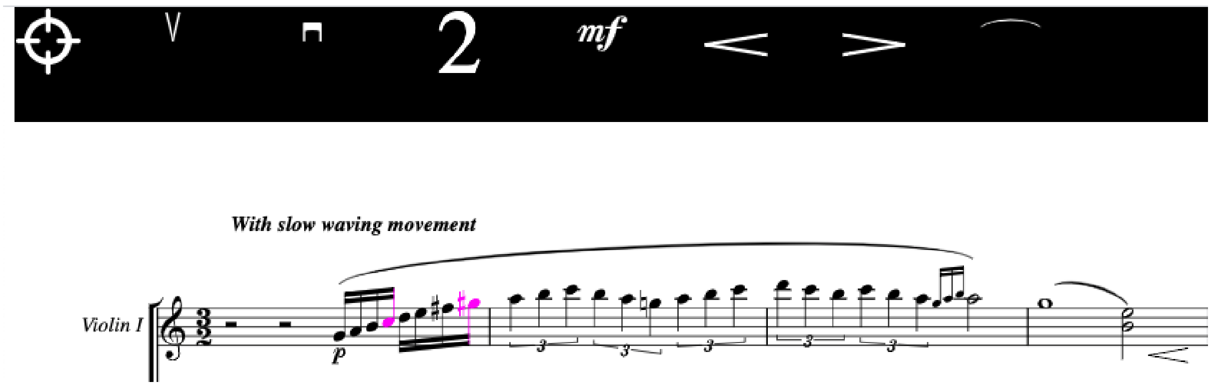


Figure 3.1: An example of an annotation toolkit used when annotating a musical score. The toolkit allows users to mark bowing direction, fingering, or dynamics directions. Clicking on *mf* opens a further annotation palette containing a range of dynamic markings[14]

3.2.4 Annotation sessions

An *annotation session* is a container used to logically group together a collection of annotations of one or more documents. For example, a music scholar may start an annotation session in order to annotate the collected works of a composer. All annotations that they create are added to the session. An annotation session allows a user to make some annotations and then pause and come back to the task at a later time. By loading a past annotation session they can see all of the annotations made up to that point and continue where they left off. There is no restriction to the type of annotations that can be added to a session. It could contain only annotations by a single user or by many users, and could contain annotations targeting only a single object, or many different objects. An annotation session could be used when one person makes a collection of annotations, and others respond to these annotations, critiquing them or adding additional comments.

An annotation session is stored as an unordered `ItemList`, with an `additionalType`[15]. Each `ListItem` in the `ItemList` has an `item` field pointing to an annotation. Annotation sessions can also be stored in a Solid Pod as an LDP container[16], which fulfil an analogous role.

3.2.5 JSON-LD output

In order to integrate with other tools within the TROMPA project, Annotations are also available via the CE's REST interface in JSON-LD format. As the underlying data model of the annotations stored in the CE does not map directly to the output format of the Web Annotations standard, a custom JSON-LD transformer converts the data model into the correct format.

[14] Image from https://delius-annotation.linkedmusic.org, developed as part of Lewis, Weigl, & Page (2019)
[15] https://vocab.trompamusic.eu/vocab#AnnotationSession
[16] https://www.w3.org/ns/ldp

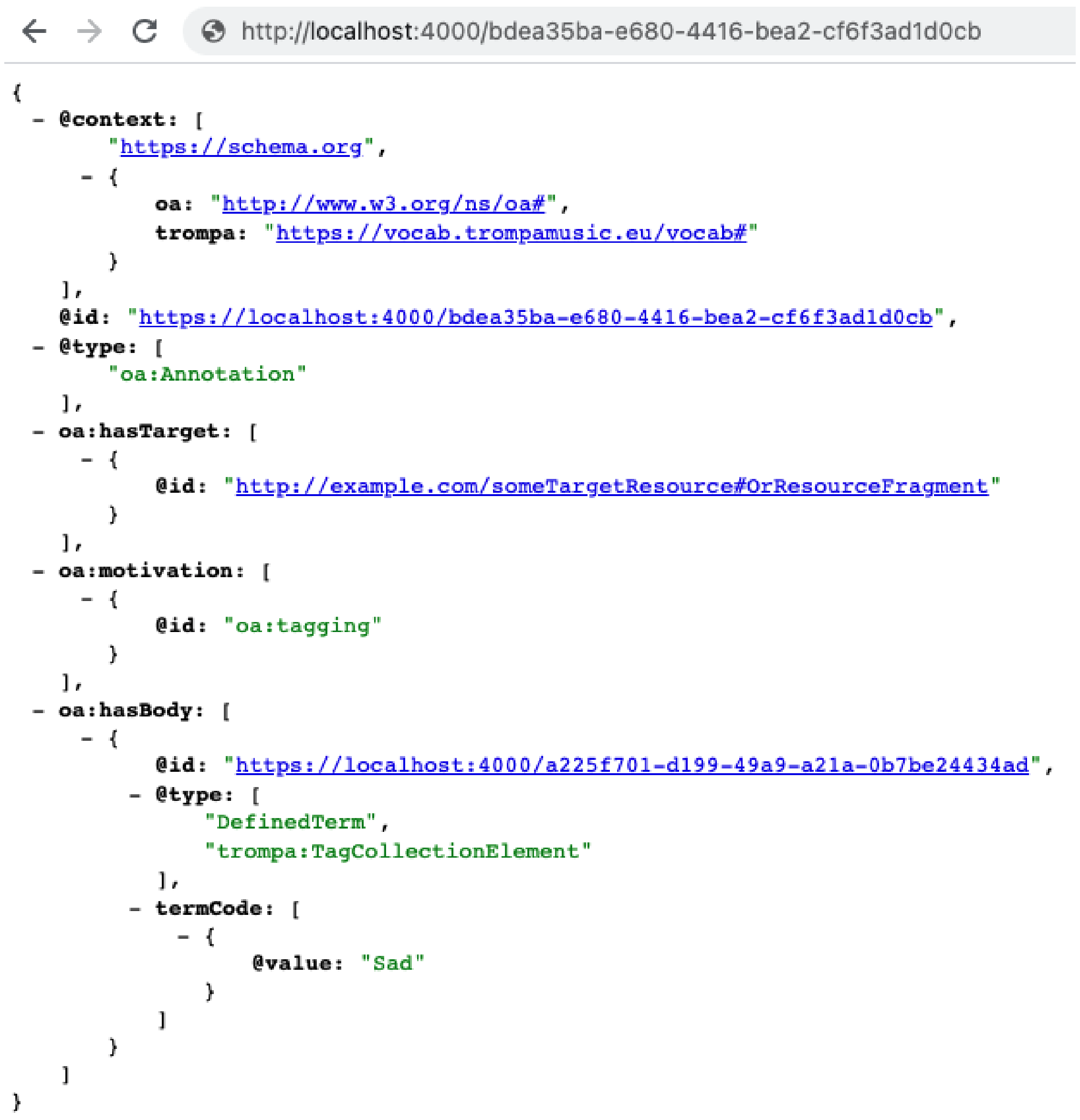

```
http://localhost:4000/bdea35ba-e680-4416-bea2-cf6f3ad1d0cb
{
  - @context: [
        "https://schema.org",
      - {
            oa: "http://www.w3.org/ns/oa#",
            trompa: "https://vocab.trompamusic.eu/vocab#"
        }
    ],
    @id: "https://localhost:4000/bdea35ba-e680-4416-bea2-cf6f3ad1d0cb",
  - @type: [
        "oa:Annotation"
    ],
  - oa:hasTarget: [
      - {
            @id: "http://example.com/someTargetResource#OrResourceFragment"
        }
    ],
  - oa:motivation: [
      - {
            @id: "oa:tagging"
        }
    ],
  - oa:hasBody: [
      - {
            @id: "https://localhost:4000/a225f701-d199-49a9-a21a-0b7be24434ad",
          - @type: [
                "DefinedTerm",
                "trompa:TagCollectionElement"
            ],
          - termCode: [
              - {
                    @value: "Sad"
                }
            ]
        }
    ]
}
```

Figure 3.2: Example of JSON-LD output of an Annotation from the CE

In addition to the implementation of the annotations model in the Contributor Environment, the `trompace-client` python library[17] was updated to provide library functions for creating DefinedTermSets and DefinedTerms, Ratings, rating templates, and Annotations.

3.3 Annotation examples

In order to orient readers with specific details about the Web Annotations model, we provide example annotations which cover the requirements of various partners in TROMPA. These examples are written in Turtle[18], a standard description language for Linked Data. While these examples are valid Turtle, they only contain fields that are relevant to demonstrate the examples, and don't include some fields which are required in the Contributor Environment (e.g., `identifier`, which is a

[17] https://github.com/trompamusic/trompace-client
[18] https://www.w3.org/TR/turtle/

unique identifier used to refer to the item in the CE, and `creator`, which is a URL specifying the vCard of the user who created the object in the CE.

3.3.1 A fixed vocabulary

A fixed vocabulary is a DefinedTermSet containing DefinedTerms, which are fixed tags that a user can select from when annotating an item. This example includes two tags, “happy” and “sad”, which a user can select as the response elicited when they listen to a specific music recording. This example also contains the RDF prefix definitions used in the remainder of the examples.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix schema: <http://schema.org/> .
@prefix trompa: <https://vocab.trompamusic.eu/vocab#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix ex: <http://example.org/example/> .

ex:DTS_Mood a schema:DefinedTermSet, trompa:TagCollection ;
 schema:name "Mood" ;
 schema:hasDefinedTerm ex:DT_Happy, ex:DT_Sad .

ex:DT_Happy a schema:DefinedTerm, trompa:TagCollectionElement ;
 schema:termCode "Happy" .

ex:DT_Sad a schema:DefinedTerm, trompa:TagCollectionElement ;
 schema:termCode "Sad" .
```

Listing 3.1: A fixed vocabulary

3.3.2 Annotation Palette

An annotation palette groups together related terms or markings. This example is part of a palette used to mark up a music score performed by a stringed instrument, indicating when the player should use an upbow or a downbow.

```
ex:PerformanceInstructionsPalette a schema:DefinedTermSet,
trompa:AnnotationMotivationCollection ;
 schema:name "Performance Instructions" ;
 schema:image
<https://alastair.trompa-solid.upf.edu/anno-images/conductor-baton.png> ;
 schema:hasDefinedTerm ex:DT_Upbow, ex:DT_Downbow, ex:DT_Arco .

ex:Upbow a schema:DefinedTerm, oa:Motivation, trompa:TagCollectionElement ;
 skos:broader oa:commenting ;
 schema:image <https://alastair.trompa-solid.upf.edu/anno-images/upbow.png> ;
 schema:termCode "Upbow" .

ex:Downbow a schema:DefinedTerm, oa:Motivation, trompa:TagCollectionElement ;
 skos:broader oa:commenting ;
 schema:image <https://alastair.trompa-solid.upf.edu/anno-images/downbow.png> ;
 schema:termCode "Downbow" .
```

Listing 3.2: Annotation Palette

3.3.3 Rating definition

A rating definition describes the lower and upper bounds that a user's rating value can contain when making an annotation on a range.

```
ex:RatingDefinition a schema:Rating, trompa:RatingDefinition ;
 schema:name "Intonation" ;
 schema:bestRating 100 ;
 schema:worstRating 1 .
```

Listing 3.3: Rating definition

3.3.4 Annotation toolkit

An annotation toolkit is a collection of types of annotations that a user could make. In this case, a user is able to choose to choose a mood elicited by a recording, rate its intonation on a scale of 1-100, or make a generic comment about it.

```
ex:InstructionsToolkit a schema:ItemList, trompa:AnnotationToolkit ;
 schema:name "Annotation Tasks" ;
 schema:itemListOrder schema:ItemListUnordered ;
 schema:description "Listen to the recording and perform each annotation task" ;
 schema:itemListElement ex:MoodItem, ex:RatingItem, ex:MotivationItem .

ex:MoodItem a schema:ListItem, trompa:AnnotationToolkitItem ;
 schema:item ex:DTS_Mood .

ex:RatingItem a schema:ListItem, trompa:AnnotationToolkitItem ;
 schema:item ex:RatingDefinition .

trompa:commenting a skos:Concept, schema:Thing ;
 skos:broader oa:commenting ;
 skos:closeMatch oa:commenting .

ex:MotivationItem a schema:ListItem, trompa:AnnotationToolkitItem ;
 schema:item trompa:commenting .
```

Listing 3.4: Annotation toolkit

3.3.5 Basic textual annotations

These examples show how an annotation can be made where the body is text, either plain unformatted text in the form of a folksonomy tag, or a more detailed description formatted using HTML.

```
ex:TagAnnotation a oa:Annotation ;
 oa:bodyValue "rock" ;
 oa:hasTarget <http://example.com/foo.mp3> ;
 oa:motivation oa:tagging .

ex:FullDescriptionAnnotation a oa:Annotation ;
 oa:hasBody ex:FullDescriptionAnnotationBody ;
 oa:hasTarget <http://example.com/someTargetResource#OrResourceFragment> ;
 oa:motivation oa:describing .

ex:FullDescriptionAnnotationBody a oa:TextualBody ;
 schema:value "detailed textual <b>description</b> of something with html!" ;
 dc:format "text/html" ;
 dc:language "en" .
```

Listing 3.5: Textual annotations

3.3.6 Annotations that link to other objects

Fields of an annotation can refer to items that have previously been created. In these examples, we set the Body of an annotation to a Defined Term which represents the “Sad” tag, the Motivation of an annotation to a Defined Term which which represents the Upbow musical direction, or the Target of an annotation to a previously created annotation.

```
ex:DefinedTermAnnotation a oa:Annotation ;
 oa:hasTarget <http://example.com/someTargetResource#OrResourceFragment> ;
 oa:motivation oa:tagging ;
 oa:hasBody ex:DT_Sad .

ex:ArcoAnnotation a oa:Annotation ;
 oa:hasTarget <http://example.com/someTargetResource#OrResourceFragment> ;
 oa:motivation ex:Upbow .

ex:ReplyingAnnotation a oa:Annotation ;
 oa:hasTarget ex:DefinedTermAnnotation ;
 oa:motivation oa:replying ;
 oa:hasBody "I disagree!" .
```

Listing 3.6: Annotations linking to other objects

3.3.7 Rating annotations

An annotation that rates an object contains a Rating object linked to a rating definition, and then uses this object as the body of an annotation.

```
ex:Rating a schema:Rating ;
 schema:ratingValue 90 ;
 schema:bestRating 100;
 schema:worstRating 1;
 prov:wasDerivedFrom ex:RatingDefinition .

ex:RatingAnnotation a oa:Annotation ;
 oa:hasTarget <http://example.com/someTargetResource#OrResourceFragment> ;
 oa:hasBody ex:Rating ;
 oa:motivation oa:assessing .
```

Listing 3.7: Rating annotations

3.3.8 Grouping together multiple annotations

If multiple Annotations should be considered as a single logical unit, they can be linked together. We create a specific Motivation which represents the linking of annotations for use in the music enthusiasts pilot.

```
ex:GroupingMotivation a oa:Motivation ;
 skos:broader oa:linking ;
 schema:name "Music enthusiasts grouping" .

oa:LinkingAnnotation a oa:Annotation ;
 oa:motivation ex:GroupingMotivation;
 oa:hasTarget <http://example.com/someTargetResource#OrResourceFragment> ;
 oa:hasBody ex:DefinedTermAnnotation, ex:RatingAnnotation .
```

Listing 3.8: Annotation groupings

3.3.9 Annotation Sessions

An annotation session lists many different but related annotations.

```
ex:AnnotationSession a schema:ItemList, trompa:AnnotationSession ;
 schema:name "Mahler 4 detailed analysis" ;
 schema:itemListOrder schema:ItemListUnordered ;
 schema:description "Some interesting academic discourse about this work" ;
 schema:itemListElement ex:ListItem1 .

ex:ListItem1 a schema:ListItem, trompa:AnnotationSessionItem ;
 schema:item ex:RatingAnnotation .
```

Listing 3.9: Annotation sessions

4. Tools for annotating

We developed a software library, written in typescript, to help with the creation and storage of annotations[19]. This includes:

- ❖ library functions to create the annotation objects described in the previous section in either the CE or a Solid Pod

[19] https://github.com/trompamusic/trompa-annotation-component, can also be installed via npm from https://www.npmjs.com/package/trompa-annotation-component

- Library functions to convert annotations between Solid Pods and the CE, allowing users to publish private annotations
- graphical components for loading audio files from the CE or external locations (for example a Solid Pod)
- providing a graphical interface to display a graphical representation of audio files and make annotations at points in time on this representation
- React components to create annotations of different types
- An administration interface to create annotation palettes and annotation toolkits, storing the result in the CE or a Solid Pod

The graphical components of the annotation library are written in React, and can be integrated into any JavaScript application that also uses React[20]. Developers can pick and choose which components they wish to use, all items are optional and standalone. We provide some examples of demo applications that join together multiple components. Library functions to create annotations in external repositories and read them are available as JavaScript classes. This means that applications which do not use React, or which have other graphical components for reading user input can still use this functionality.

4.1 React components

4.1.1 Audio player

The audio player component is a wrapper around the open source library wavesurfer[21]. This component can load an audio file and display it graphically as a waveform. The component has an audio player that can be used to play back the audio file which is displayed. The wavesurfer regions plugin allows users to click on a point in the audio waveform to indicate that an annotation should be made at that point in time. By clicking and dragging the mouse on the audio waveform, an annotation can be made that spans a time range.

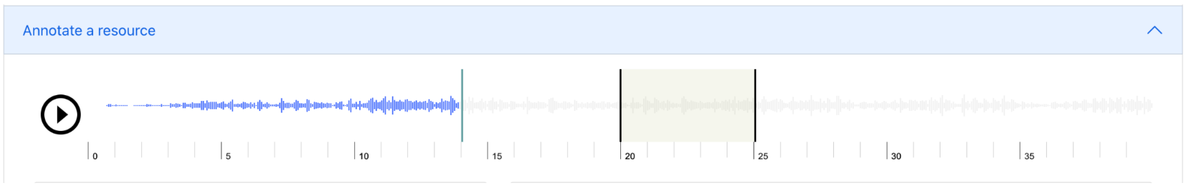


Figure 4.1: Audio player with a point annotation and a range annotation

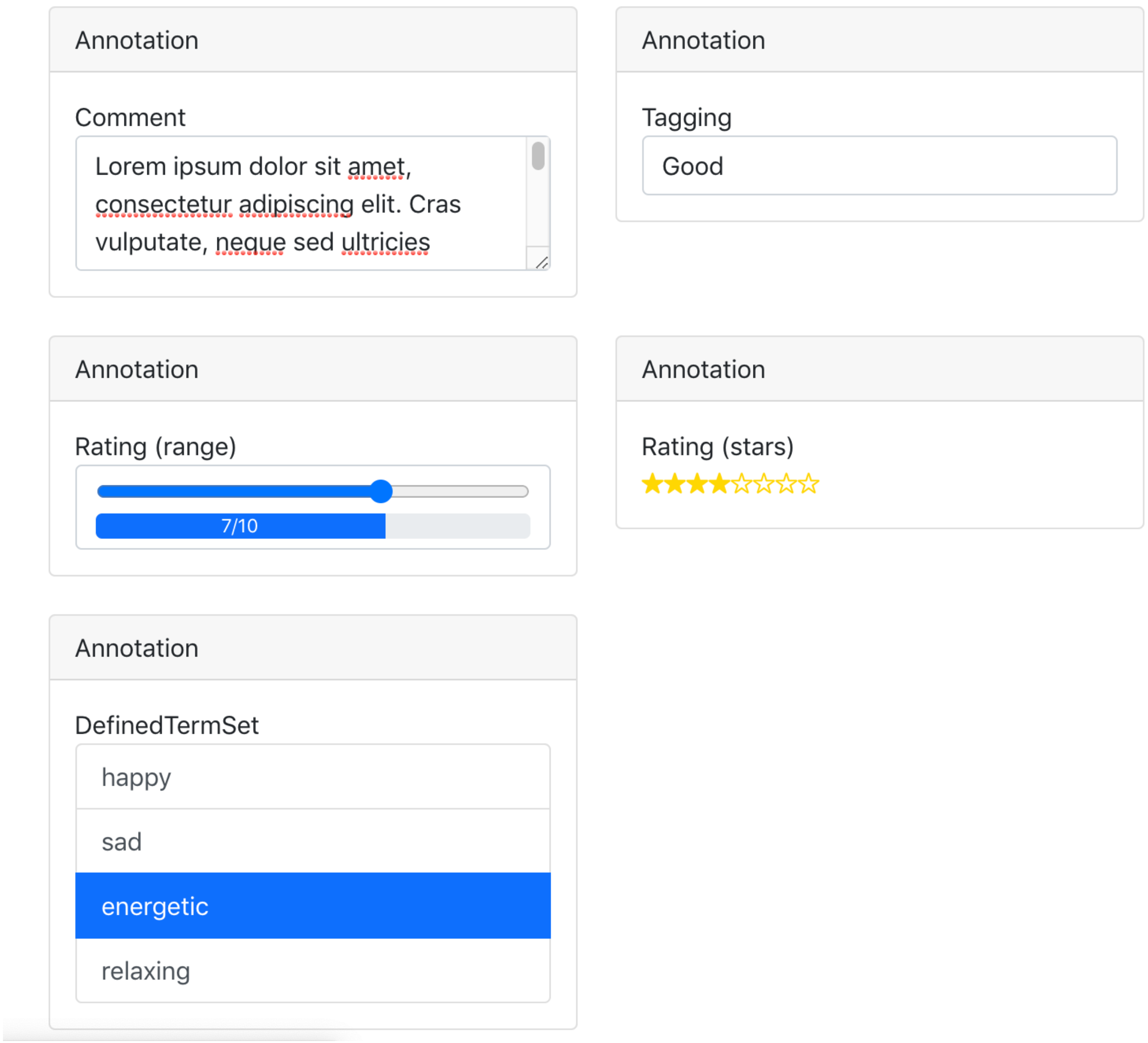
4.1.2 Annotation input components

We provide React components for entering text annotations, and for the other types of motivations and bodies described in section 3.2. If the application developer has identified that users should only create a motivation for a specific type of motivation, the developer can directly show a component for that motivation. If the tool is more general and allows a user to apply a number of different motivations, they can choose the desired motivation from a list of possible motivations (defined in an annotation toolkit), and the motivation selection component will show an input suitable for that motivation. For the `classifying` motivation, two components are available. One which provides a slider that can be dragged between the lowest and highest possible rating values, and a text box that allows a user to type in a rating. The value is verified to be within the range of possible values.

[20] We use the term *component* here to refer to the concept of a React component, an independent, reusable part of an application's user interface: https://reactjs.org/docs/components-and-props.html

[21] https://wavesurfer-js.org

Validation between a lower and upper value is done only if a Rating definition is available (section 3.2.2). For the `tagging` motivation, two components are available. For a free-form tagging operation, a text box lets a user enter a desired tag. For a fixed vocabulary, all of the items in the vocabulary are listed and a user can select one of them. For motivations defined by an annotation palette (section 3.2.3), all items in the annotation palette are displayed as in the fixed vocabulary and the user can select one of them. For other types of motivation, a text box is shown, allowing the user to type in the content of annotation.



Annotation

Comment

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vulputate, neque sed ultricies

Annotation

Tagging

Good

Annotation

Rating (range)

7/10

Annotation

Rating (stars)

★★★★☆☆☆☆

Annotation

DefinedTermSet

happy

sad

energetic

relaxing

Figure 4.2: Annotation widgets for entering freeform text, tags, ratings, and fixed choices

4.1.3 Annotation template and toolkit editors

The palette and toolkit editor provides a graphical user interface for creating fixed term vocabularies, annotation palettes and annotation toolkits. When creating a vocabulary or palette, the user can specify a name, and add child items. Child items can be identified by a textual term, or in the case of palettes, external images. The rating template editor allows the user to create a rating template, adding a name and a best value and worst value.

The toolkit editor allows a user to create a list of annotation motivations, or add a specific vocabulary, palette, or rating template.

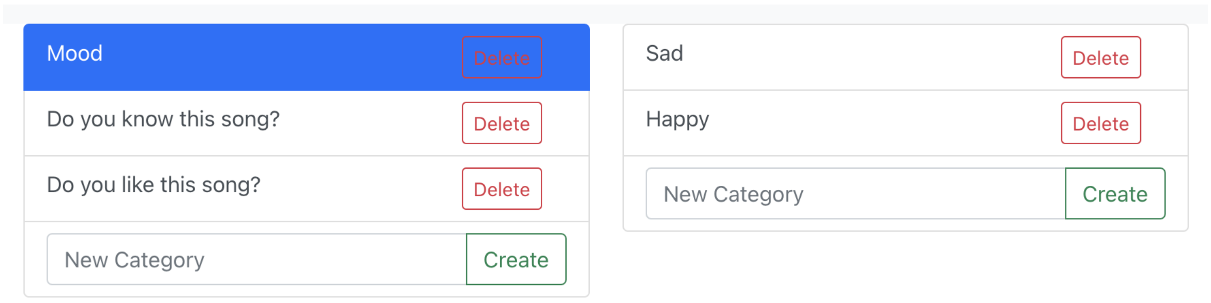
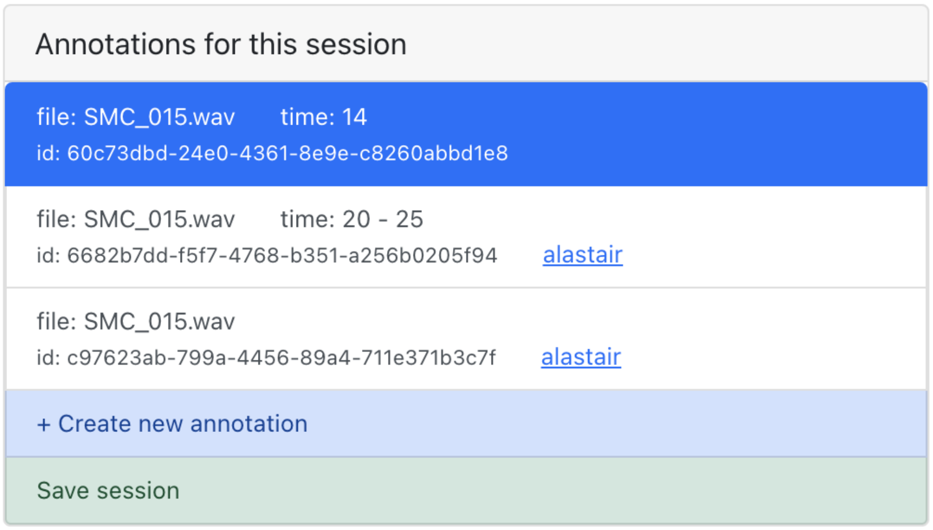


Figure 4.3: Editor which allows users to create fixed vocabularies, showing a list of vocabularies on the left and the terms in the selected vocabulary on the right. The same editor can also be used to create annotation palettes, allowing users to select an annotation motivation and an optional image

4.1.4 Annotation session viewer

As described in Section 3.2.4, an annotation session contains a collection of annotations, potentially by more than one user and of one or more audio recordings. The annotation session viewer component can show a list of all annotation sessions created by a user. When a session is selected, it lists all annotations that have been saved to that session. In its current state, the annotation session viewer can only display a session if it contains items made by one user and of one audio recording. The component is extensible and in the future it is hoped that we can add the functionality to display annotations by different users and of different recordings. Any annotation in the list can be selected to see information about the annotation or to show its position on the audio viewer.


Annotations for this session

file: SMC_015.wav time: 14
id: 60c73dbd-24e0-4361-8e9e-c8260abbd1e8

file: SMC_015.wav time: 20 - 25
id: 6682b7dd-f5f7-4768-b351-a256b0205f94 alastair

file: SMC_015.wav
id: c97623ab-799a-4456-89a4-711e371b3c7f alastair

Figure 4.4: Annotation session viewer which allows users to see previous annotations and load them again

4.1.5 Multimodal Component items for Annotations

We implemented a search function for annotations that can be integrated into the Multimodal Component (D5.3). This allows users to use the multimedia component to search for annotations based on textual content of the annotation body, the value of tags, or the name of the object that is annotated. This interface is only available for annotations which are stored in the CE. Annotations with a target or body outside of the CE cannot be searched.

4.2 Integration between Trompa CE and Solid

Within the TROMPA project we have varying needs for annotations where some annotation tasks should be private, some can be public, and some can be initially created in private but then published to an open repository (Weigl et al., 2020). In order to support these workflows, the annotation library contains functionality to store annotations in both the CE (using GraphQL mutations) and in Solid pods, storing the annotations in JSON-LD format.

4.2.1 Publishing annotations

A common use case which we encountered in TROMPA was the need to store annotations privately. For example, music scholars may wish to perform an annotation task and only publish the full result of the process once they have finished. If we published annotations directly in the CE as they were created, anyone may be able to read these annotations before the creator had a chance to complete and review them. We provide the ability for annotations to be written to a private Solid Pod initially, and then *published* to the CE. There are two ways of publishing an annotation. One is to create a new annotation object in the CE containing all of the data of the annotation (its target, motivation, and body). The other is to create an annotation object in the CE that links back to the annotation stored in the user's Solid Pod. We use the oa:canonical relation to indicate the canonical version of an annotation that is in the CE[22]. The annotations library provides functionality to create both kinds of annotations, and allows users to change the permissions of annotations in their Solid Pod, allowing them to be read by specified users, or opened to the public.

4.3 Demo application

We include in the annotation components source code repository a number of demo applications that show how each of the components that we described function.

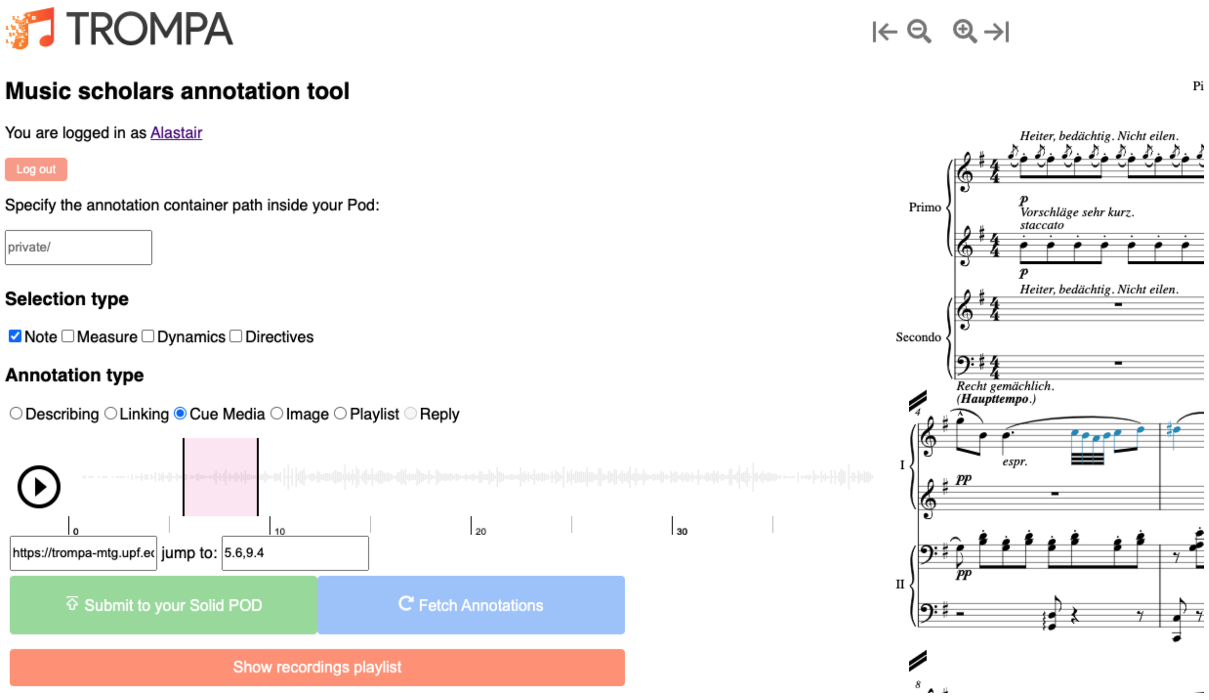
- ❖ Editors which allow users to create fixed-term vocabularies, annotation palettes, annotation toolkits, and rating templates. These editors save the contents of the related object to the CE, and provide functions for users of the software to load these items.
- ❖ An example of a basic annotation task that involves making a single type of annotation on a single target file.
- ❖ A more complex annotation task that involves creating an annotation session, selecting a file, selecting a toolkit, and making annotations on the file using items defined in the toolkit.
- ❖ An example of loading existing annotations from the CE and showing them on a waveform view of the target audio.
- ❖ Logging in to a Solid Pod and saving annotations to the Pod.
- ❖ Publishing annotations from the Solid Pod to the CE
- ❖ Integrates multimedia component and annotation components into a system that lets you select a file and make an annotation

5. Integration with other deliverables and software

Components and functionality from this deliverable have been integrated into other deliverables and projects in the scope of TROMPA. The scholars pilot (D6.3) via the selectable-score component (D5.2)

[22] https://www.w3.org/TR/annotation-model/#other-identities

uses the text input components to allow users to input annotations. It also uses the audio player to allow users to link parts of a music score with a point in time in an audio recording (Figure 5.1). In addition, the pilot uses the Javascript libraries developed in this deliverable to save annotations, both to users' Solid Pods and to the CE, and to publish annotations from user's Solid Pods to the CE.


TROMPA

Music scholars annotation tool

You are logged in as Alastair

Log out

Specify the annotation container path inside your Pod:

private/

Selection type

[✓ Note] Measure Dynamics Directives

Annotation type

Describing Linking [✓ Cue Media] Image Playlist Reply

https://trompa-mtg.upf.ec jump to: 5.6,9.4

Submit to your Solid POD

Fetch Annotations

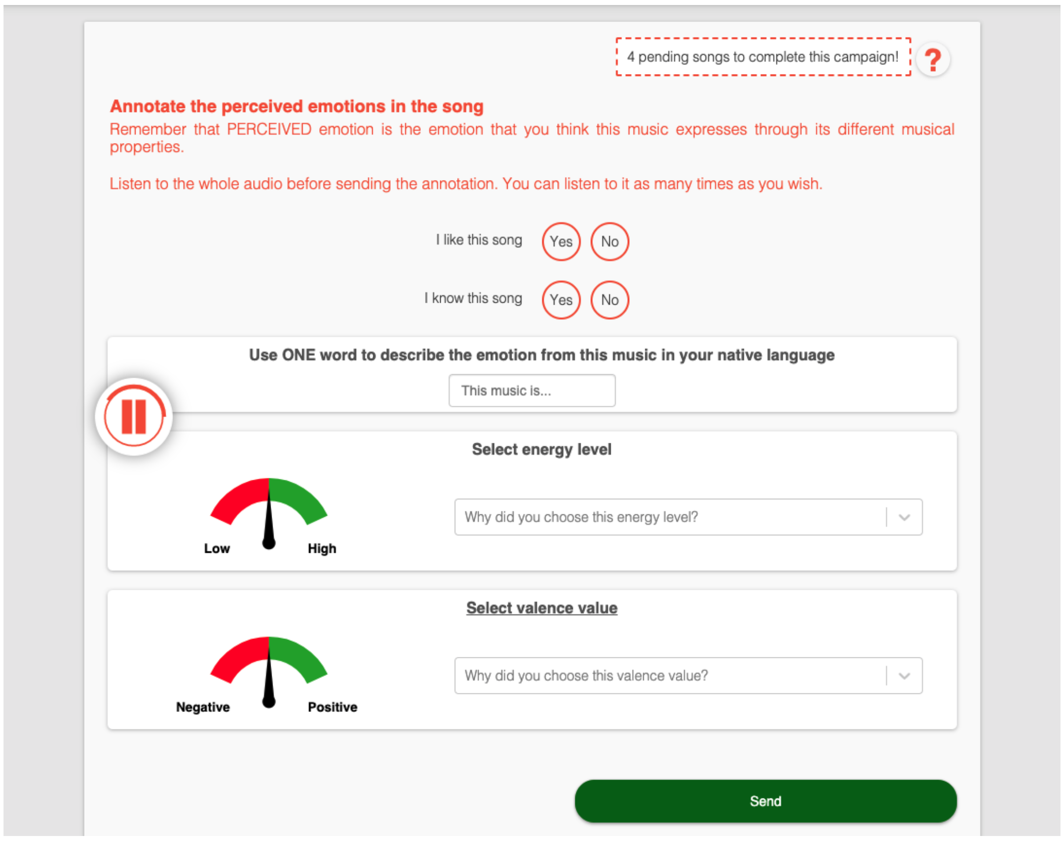
Show recordings playlist

0
10
20
30
Primo
Secondo
Heiter, bedächtig. Nicht eilen.
Vorschläge sehr kurz. staccato
Recht gemächlich. (Haupttempo.)
espr.
pp
I
II

Figure 5.1: Integration of the audio player in the scholars prototype

The music enthusiasts pilot uses the annotation data model to store user annotations in the CE. Annotations are made using a custom interface developed for the pilot, but can be stored in the CE. A custom backend was developed in PHP to save annotations[23]. This shows that the annotations schema can easily be used from a number of different programming languages. The pilot stores annotations using the fixed vocabulary model, ratings, single-word tags, and free-form textual descriptions. Each user input is stored as an individual annotation, and all annotations are joined together with a custom oa:linking motivation to indicate that these annotations should be considered as a single logical entity. A workflow process integrated with the TPL (D5.3) reads annotations from the CE in order to process them and develop custom recommendation models for users.

[23] https://github.com/trompamusic/music-enthusiast-front-end/tree/master/backend-tools



4 pending songs to complete this campaign!

Annotate the perceived emotions in the song

Remember that PERCEIVED emotion is the emotion that you think this music expresses through its different musical properties.

Listen to the whole audio before sending the annotation. You can listen to it as many times as you wish.

I like this song Yes No

I know this song Yes No

Use ONE word to describe the emotion from this music in your native language

This music is...

Select energy level

Why did you choose this energy level?

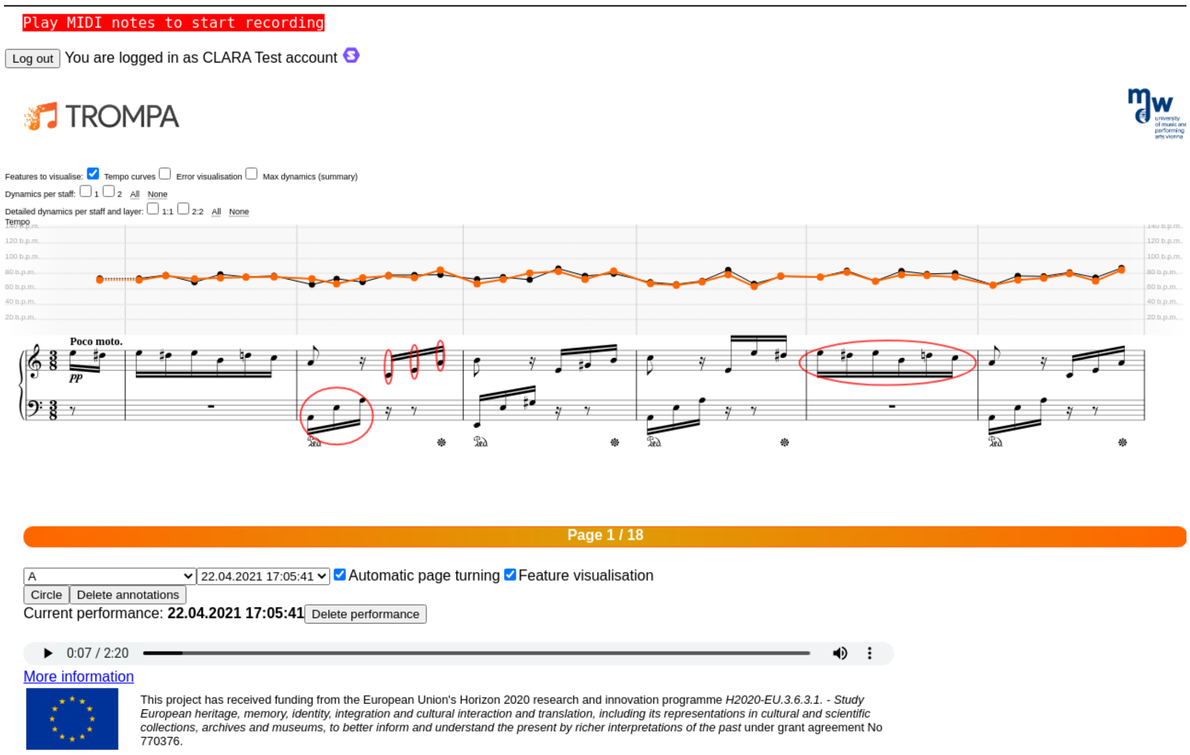
Select valence value

Why did you choose this valence value?

Figure 5.2: User input interface for music enthusiasts pilot. Inputs for each prompt were developed independently in this pilot, but the results of the annotations are stored in the CE

The instrument players pilot uses the javascript interface to store annotations created by users on music scores. In annotation selection mode users can click and drag along the rendered score to make a selection (selected score elements are coloured light blue). A oa:highlighting annotation is generated, which is rendered on score in the form of an ellipse enclosing all selected elements. Once an annotation is created, it can be stored to a user's Solid Pod or the CE using the functionality provided by the annotation component library. Annotations can also be published to the CE from existing annotations stored in a Solid Pod.

The integration of the library developed in this deliverable into other pilots shows that it is possible to use the data model over different modalities (music scores, and audio), and also integrate to varying degrees of detail. Some pilots used the input interfaces that we developed, while others developed their own to fit the design of their own software more easily. All pilots used the data model, but some saved annotations using their existing backend tooling while others used the javascript libraries that were developed as part of this deliverable. This range of different integrations to create different types of annotations shows that the requirements that we described in Section 2 to be able to represent a wide range of types of annotations and a software tool that could be integrated to varying degrees.


Play MIDI notes to start recording

Log out You are logged in as CLARA Test account

TROMPA

Features to visualise: [✓] Tempo curves [ ] Error visualisation [ ] Max dynamics (summary)

Dynamics per staff: [ ] 1 [ ] 2 All None

Detailed dynamics per staff and layer: [ ] 1:1 [ ] 2:2 All None

Page 1 / 18

A | 22.04.2021 17:05:41 | [✓] Automatic page turning [✓] Feature visualisation

Circle | Delete annotations

Current performance: **22.04.2021 17:05:41** Delete performance

0:07 / 2:20

More information

This project has received funding from the European Union's Horizon 2020 research and innovation programme *H2020-EU.3.6.3.1. - Study European heritage, memory, identity, integration and cultural interaction and translation, including its representations in cultural and scientific collections, archives and museums, to better inform and understand the present by richer interpretations of the past* under grant agreement No 770376.

Figure 5.3: Instrument players prototype, including annotation of music scores and storage of annotations to a Solid Pod

6. References

6.1 Written references

M. Cartwright, A. Seals, J. Salamon, A. Williams, S. Mikloska, D. MacConnell, E. Law, J. Bello, and O. Nov. "Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations." In Proceedings of the ACM on Human-Computer Interaction, 1(1), 2017.

D. Lewis, D. M. Weigl, and K. R. Page. “Musicological observations during rehearsal and performance: a Linked Data digital library for annotations. In Proceedings of the 6th International Conference on Digital Libraries for Musicology. Association for Computing Machinery, New York, NY, USA, 1–8, 2019. DOI:https://doi.org/10.1145/3358664.3358669

D. M. Weigl, and K. R. Page. “A framework for distributed semantic annotation of musical score: `Take it to the bridge!’”. In Proceedings of the 18th International Society for Music Information Retrieval Conference, 2017.

D. M. Weigl, W. Goebl, A. Hofmann, T. Crawford, F. Zubani, C. C. S. Liem, and A. Porter. 2020. Read/Write Digital Libraries for Musicology. In Proceedings of the 7th International Conference on Digital Libraries for Musicology, 2020. DOI:https://doi.org/10.1145/3424911.3425519

6.2 List of abbreviations

Abbreviation	Description
CE	Trompa Contributor Environment
JSON-LD	JSON-Linked Data, an extension to the JSON structured data format to allow semantic-Web Annotations to be embedded in the data
GOLD	Goldsmiths, University of London, Trompa partner
MDW	Universität für Musik und darstellende Kunst Wien, Trompa Partner
SKOS	Simple Knowledge Organization System
UPF	University Pompeu Fabra, Trompa Partner
VL	Voctro Labs, Trompa Partner