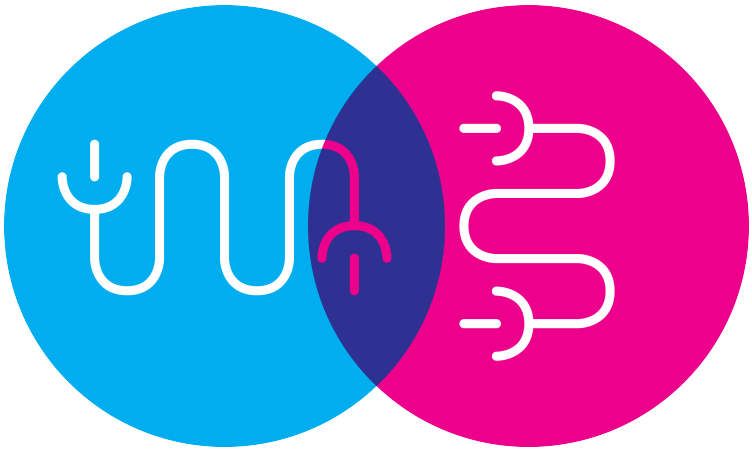


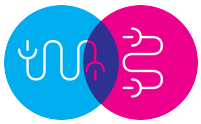
Project Acronym:	**MusicBricks**
Project Full Title:	**Musical Building Blocks for Digital Makers and Content Creators**
Grant Agreement:	**N°644871**
Project Duration:	**18 months (Jan. 2015 - Jul. 2016)**

D3.1 Functional analysis, technical requirements and API definition

Deliverable Status:	**Final**
File Name:	**MusicBricks_D3.1.pdf**
Due Date:	**31st March 2015 (M3)**
Submission Date:	**27th March 2015 (M3)**
Dissemination Level:	**Restricted**
Task Leader:	**Universitat Pompeu Fabra**

Authors: **Jordi Janer (UPF), Hanna Lukashevich (IDMT), Jakob Abeßer (IDMT), Sascha Grollmisch (IDMT), Thomas Lidy (TU WIEN), Alexander Schindler (TU WIEN), Frederic Font (UPF)**

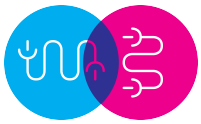
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



The MusicBricks project consortium is composed of:

SO	Sigma Orionis	France
STROMATOLITE	Stromatolite Ltd	United Kingdom
IRCAM	Institut de Recherche et de Coordination Acoustique Musique	France
UPF	Universitat Pompeu Fabra	Spain
Fraunhofer	Fraunhofer-Gesellschaft zur Foerderung der Angewandten Forschung E.V	Germany
TU WIEN	Technische Universitaet Wien	Austria

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



Revision Control

Version	Author	Date	Status
0.1	Jordi Janer (UPF)	Month 03, 2015	Initial Draft
0.2	Thomas Lidy (TUW)	March 27, 2015	Draft
0.3	Jordi Janer (UPF)	March 27, 2015	Draft
0.4	Jordi Janer (UPF)	March 30, 2015	Final Draft reviewed
0.5	Cyril Laurier (STROM)	March 30, 2015	Quality check
1.0	Marta Arniani	March 31, 2015	Quality check and submission to the EC

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

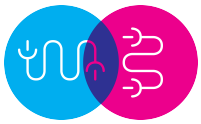
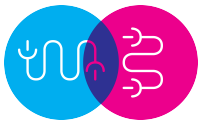


Table of Contents

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

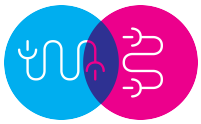


Executive summary

The present document is a deliverable of the #MusicBricks project, funded by the European Commission's Directorate-General for Communications Networks, Content & Technology (DG CONNECT), under its Horizon 2020 research and innovation programme.

In the scope of the MusicBricks project, Workpackage 3 (Application Programming Interfaces) is devoted to enabling the transfer of research results in the form of a number of APIs. Our final users are members of the creative communities that participate in prototyping (hacking) events. Participants of these events are mostly individuals with general technical and programming skills but not experts in audio or music technologies. Therefore the design of APIs shall focus on offering a clear and attractive functionality to target users, rather than present all possibilities of research algorithms.
During the first three months of the project, the partners involved in the Task 3.1 (Wrapping existing tools and technologies together) in WP3 (TU WIEN, Fraunhofer and UPF) worked on preparing existing technologies and defining the APIs to be provided.
Seven APIs will be included in WP3: Melody & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT), Realtime Monophonic & Polyphonic Pitch Detection API (by Fraunhofer IDMT), RP extract: Rhythmic and Spectral Audio Feature Extraction (by TU WIEN), "Search by Sound" Music Similarity Retrieval API (by TU WIEN/Spectralmind), Freesound 2.0 Web API (by MTG-UPF), Melody Extraction Library API (MTG-UPF), Concatenative Synthesis Library API (by MTG-UPF).
In this document, we first give an overview of the users and scenarios that the APIs shall consider. Then, we provide the functional analysis, the technical requirements, and issues related to the Intellectual Property Rights. Finally, we provide a list of the APIs with a detailed description about the available platforms and interfaces.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



1. Introduction

This document provides a description of the technologies that will contribute to the #MusicBricksAPI. Each research partner has identified how their technologies portfolio shall be adapted to be used in the creative context of MusicBricks events. We first situate the tasks of WP3 (API) in the context of the whole project, and then provide the functional analysis, technical requirements and finally, a detailed description of the technologies

1.1 Context

To define the characteristics of the APIs, we need to understand well the #MusicBricks ecosystem and the typologies and background of involved creative communities.

1.1.1 Target scenario: hackathon events

Our technologies shall be used in the context of *hackathon* events. These events have special characteristics: they are short-term (usually 24hour) activities, in which participants need to intensively and rapidly work on the development of a prototype.

Several factors are relevant when preparing technologies for these events:

- clear documentation available online before the event,
- tutorials or plug-n-play examples to modify and extend, and
- standard or widespread programming languages (e.g. Javascript, Python) to facilitate code reusability.

1.1.2 Target users: hacker communities

In the #MusicBricks project, we refer to the participants in the hackathon events as “creative communities”. This is a too broad definition that does not exactly reflect the actual profile of the participants. We identify several main characteristics to narrow down the expertise and background of the target participants:

- creative and exploratory mindset,
- technical background in programming (especially in web technologies),
- strong commitment and focus, and
- occasionally, non-technical participants (designers, musicians).

Participants find the motivation to get involved in hackathons because of their technically challenging aspects, also for the social/collaborative part of it (meeting new people), and simply because it is fun.

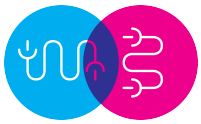
When preparing the #MusicBricks APIs, one important aspect to consider is to understand what the decisive motivation factors for hackers to choose a particular technology/API (Application Programming Interface) are. These factors can be directly related to the technology or simply external factors. The identified factors are listed in the following:

Technical

- Quality/ease of deployment: a clear documentation with examples makes a steep learning curve for using a technology/API more likely. Thus, in #MusicBricks we should make sure that we provide sufficient material.
- Programming language: a technology/API that can be used with a known programming language is more prone to be used than if it uses an obscure in-house scripting language.
- Novelty: this is a double-edged sword. It offers the possibility to be the first in building a prototype using this technology, but at the same time there is the risk that the technology/API has not been sufficiently tested.

Non-technical

The MusicBricks project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement n°644871



- Prize: sponsors that provide a technology/API usually give a prize to the best hack that uses their technology. This is a marketing strategy that can be considered as part of the #MusicBricks project.
- Fun factor: an API might enable to build a prototype that exploits the 'fun factor'.
- Challenge: From their technical mindset point of view, participants might regard the complexity of an API as a 'challenge'.
- Hall of fame: creative users participate also because of intrinsic motivation; the public visibility of the event is important for the exposure of the winner(s). In #MusicBricks we have the dissemination potential to be attractive to participants.

1.2 Objectives

Principally, MusicBricks is about bringing latest research outcomes in Music Technology to creative users. Two workpackages (WP3 and WP4) are devoted to the preparation and adaptation of technologies in a form that creative users (i.e. hackers and non-researchers) can experiment with them. WP3 targets specifically the development of APIs that allows building new prototypes with these technologies.

A set of APIs will give access to different types of technologies and platforms, from web services to binary libraries. The objective of this workpackage is not to build a new comprehensive framework that encapsulates and unifies existing technologies. The aim of WP3 is rather to identify the relevant partner's technology and provide them with the necessary functionality in order to be practically used by creative communities. The APIs will hence give access to self-contained technologies that can be either used individually or in an interconnected manner.

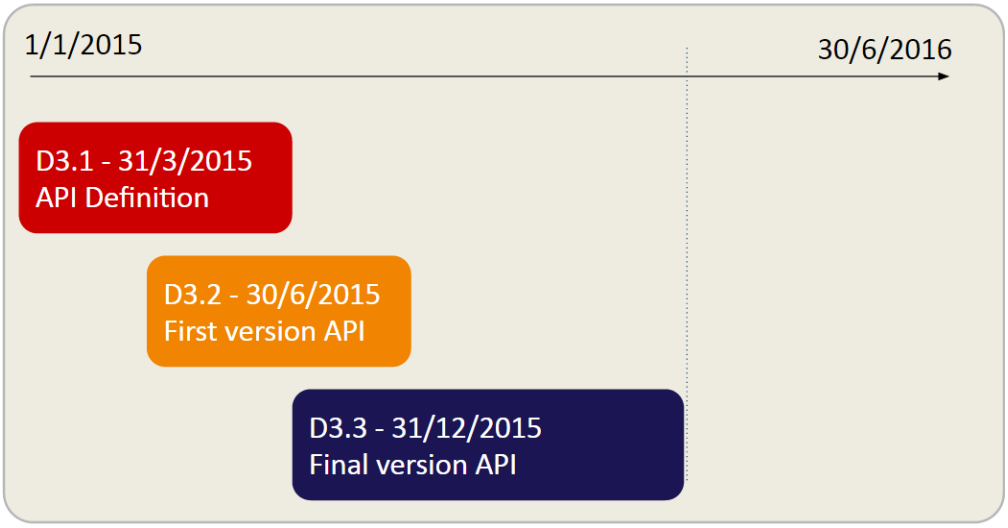
Within WP3, each partner is responsible to provide a fully working and fully documented technology and API. The outcomes of WP3 will be a catalogue of technologies and APIs under the umbrella of #MusicBricks. A common webpage with a consistent look&feel will be prepared in the context of the project, which shall serve as an entry point for creative communities.

This workpackage consists of three tasks:

- Task 3.1: Wrapping existing tools and technologies together (MTG-UPF + IDMT, TU WIEN)
  - Definition of Wrappers & Clients Requirements and Application examples
  - M1 → M12
- Task 3.2: Creating specific tool combinations for advanced features (TU WIEN+MTG-UPF)
  - Integration of Wrappers, Clients and Application examples
  - M4 → M12
- Task 3.3: Extending the frameworks beyond the project (IDMT + MTG-UPF, TU WIEN, IRCAM)
  - Guidelines for new Wrappers and Clients and Application examples
  - M7 → M12

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

Timeline & Deliverables

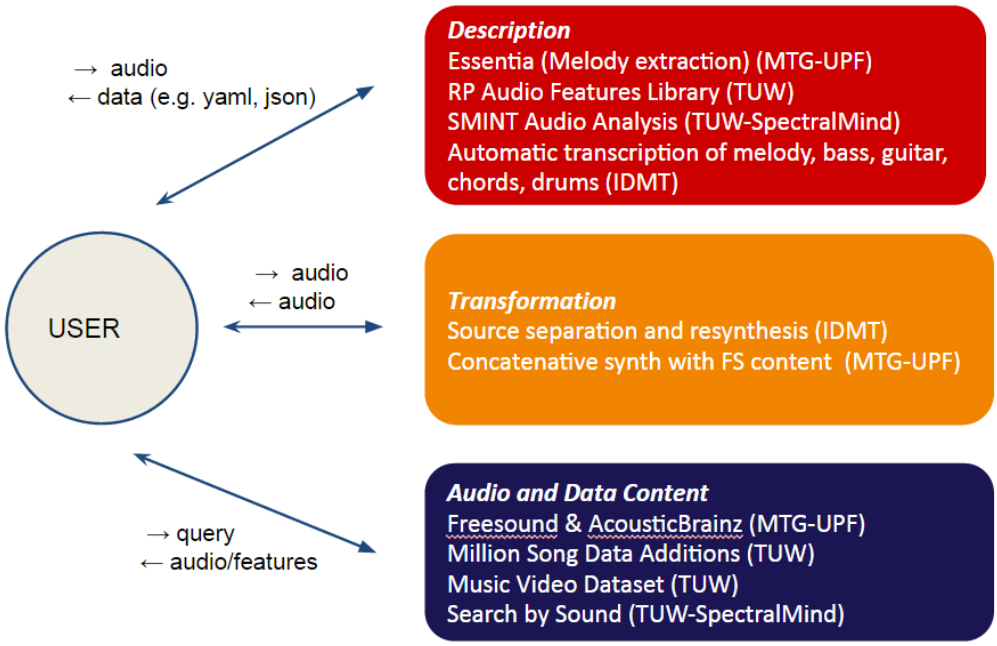


Throughout the first three months of the project, we worked on Task 3.1 (Wrapping existing tools and technologies together), in which each research partner has defined the technology, and the functional analysis and technical requirements to be provided in the API. This deliverable (D3.1 Functional analysis, technical requirements and API definition) reports our activities.

2. Functional analysis

MusicBricks proposes to validate a set of state-of-the-art tools developed as a result of academic and national research, through deployment by means of Application Programming Interfaces (APIs), Graphic User Interfaces (GUIs) and Tangible User Interfaces (TUIs), to a set of testbeds in the creative, industry and market settings. MusicBricks' ambition is to progress the way Creative SMEs approach the latest state-of-the-art music technologies. The proposed state-of-the-art tools available through MusicBricks can be grouped into: i) Transformation tools; ii) Description tools; and iii) Data and Content access and data. The next figure summarizes the technologies provided by each partner in these three categories:

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



3. Technical Requirements

Above all, the technical requirements of the APIs should be in line with user expectations in a hackathon context. It will of course depend on the type of technology, and how users can interact with it. We define three types of APIs:

1. Web API: web services running on partners' servers. Public API to run audio & music processes
2. Binary Library API: objective code libraries that can be used by hackers according to the agreed license terms
3. Source code Library API: open source code libraries that can be used by hackers. Source code can be extended or modified.

For these three types of APIs, we can outline several recommendations:

- All APIs should come with complete technical documentation.
- Working examples and demo applications along with the API will help its adoption by hackers.
- For Web API, we should implement clients in common languages for server-side or web applications (e.g. Javascript, Python).
- For Library API, we should make it available on as many platforms as possible. We recommend releasing it on several platforms, at least Linux and OSX, Windows also if possible.

All the developments for creating APIs in WP3 will be carried out individually by the research partners (UPF, IDMT, TU WIEN). Each partner is responsible for managing the source code base. In some cases, such as for MTG-UPF

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

technologies and the TU WIEN feature extractor, a public repository on Github will be made available. In other case, the technologies are provided as closed binary object code, and source code is only accessible to the partner. Only the API will be made public in these cases.

4. IPR and licensing conditions

The Intellectual Property Rights of all technologies used in the project is an important aspect. Each research partner is owner of all background IP that is provided in the scope of MusicBricks.

Different IPR models coexist, as it depends on each technology and the way partners aim at their exploitation. Therefore, together with the defined APIs, we will provide information about the IPR for each individual technology, and what the conditions are so that creative users (hackers) can employ the technology, on-site (during the hackathon events), and after the event, and how a potential commercial exploitation of the results of a new developed prototype (hack) can be managed.

5. First Definition of MusicBricks APIs

5.1 Introduction

In this section we describe the technologies and APIs provided by each research partner.

Lists of APIs:

- Melody & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT),
- Realtime Monophonic & Polyphonic Pitch Detection API (by Fraunhofer IDMT),
- RP extract: Rhythmic and Spectral Audio Feature Extraction (by TU WIEN),
- “Search by Sound” Music Similarity Retrieval API (by TU WIEN/Spectralmind),
- Freesound 2.0 Web API (MTG-UPF),
- Melody Extraction Library API (MTG-UPF),
- Concatenative Synthesis Library API (MTG-UPF)

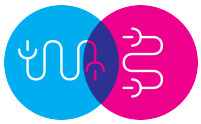
5.2 Melody & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT)

5.2.1 Description

The first binary executable allows performing different music analysis tasks for a given WAV or MP3 audio file of 15 minutes length maximum:

- **Bass Transcription**
  - This algorithm transcribes the bass line of a song as a sequence of note events
- **Melody Transcription**
  - This algorithm transcribes the predominant melody of a song as a sequence of note events
- **Key**
  - This algorithm estimates the most likely major or minor key of a given song

The MusicBricks project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement n°644871



- **Beat Grid**
  - This algorithm estimates the metric structure of a given song and returns the beat times and beat numbers
- **Tempo**
  - This algorithm estimates the average tempo of a song in beats per minute (bpm)

5.2.2 Platforms

- Windows
- Mac OSX
- Linux

5.2.3 Interface Description

The executable allows to select the **analysis tasks** to be performed using **commandline parameters**.
All analysis results are exported as XML file.
Optionally, the analysis results (transcription, beat grid, etc.) can be exported as

- **MIDI** (melody & bass transcription results),
- **MusicXML** (melody & bass transcription result), and
- **CSV** (both the transcription and beat-grid results can be exported in such way that they can be easily imported to SonicVisualizer as note layer or time instant layer)

for further use in other programs.

5.3 Realtime Monophonic & Polyphonic Pitch Detection API (by Fraunhofer IDMT)

5.3.1 Description

The realtime melody transcription technology will be provided as dynamic libraries (DLL/SO files) to be included into the users programs.

5.3.2 Platforms

- Windows, Mac OSX, Linux
  - Sample-Code in C++ will be provided
- iOS, Android
  - A sample application for both operating systems will be provided as demo

5.3.3 Interface Description

The interface will provide methods to

- switch between the pitch detection version (monophonic, polyphonic),
- create/delete a pitch detection object,
- set reference frequencies for polyphonic pitch detection (to limit the search range), and
- get one/multiple detected fundamental frequencies for a given sample buffer.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

5.4 RP extract: Rhythmic and Spectral Audio Feature Extraction (by TU WIEN)

5.4.1 Description

A library that processes PCM (WAV or decoded MP3) data as input and analyzes the rhythmic and spectral information in the audio to create different audio descriptors (a.k.a. features):

- RP: Rhythm Patterns: a spectral representation of rhythmic patterns (audible frequency vs. repetition frequency)
- SSD: Statistical Spectrum Descriptor: describing timbral aspects of the Sonogram
- RH: Rhythm Histogram: simplified rhythm descriptor (roughly showing rhythmic strength for different bpm values)

These audio descriptors can be used to find similar sounding songs, create playlists, make music recommendations etc. Depending on the needs, a rhythmic descriptor can be used (RP for high dimensions, RH for low dimensions) or a timbral one, or a combination of both.

The output is a corresponding list of feature vectors for the supplied input data, in the following formats:

- Matlab: SOMlib file format (an extended CSV file format with headers)
- Java: SOMlib file format or Weka ARFF
- Python: Python dictionary with the feature abbreviations as dictionary keys and the feature vectors as values.

More information about the features is available at http://ifs.tuwien.ac.at/mir/audiofeatureextraction.html . [1]

5.4.2 Platforms

Implementations:

- Python
- Matlab
- Java

All platforms running Python, Matlab or Java are supported:

- Windows, Mac OS X, Linux

Matlab and Java versions include decoding functionality for compressed audio in MP3 format. The Python version operates only on decoded audio data.

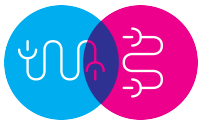
We plan to provide the Python library also as a Web service API for direct and easy integration without local deployment.

5.4.3 Interface Description

Function name: rp_extract

- **data**: pcm signal data
- **samplerate**: signal sampling rate
- **extract_rp:** extract Rhythm Patterns features
- **extract_ssd:** extract Statistical Spectrum Descriptor
- **extract_sh:** extract Statistical Histograms
- **extract_tssd:** extract temporal Statistical Spectrum Descriptor

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



- **extract_rh:** extract Rhythm Histogram features
- **extract_trh:** extract temporal Rhythm Histogram features
- **extract_mvd:** extract modulation variance descriptor
- **resample:** do resampling before processing: 0 = no, > 0 = resampling frequency in Hz
- **skip_leadin_fadeout:** how many sample windows to skip at the beginning and the end
- **step_width:** >=1 each step_width'th sample window is analyzed
- **n_bark_bands:** 15 or 20 or 24 (for 11, 22 and 44 kHz audio respectively)
- **mod_ampl_limit:**
- **spectral_masking** use Spectral Masking
- **transform_db:** use transformation into DeciBel
- **transform_phon:** use transformation into Phon
- **transform_sone:** use transformation into Sone (requires transform_phon enabled)
- **fluctuation_strength_weighting:** apply Fluctuation Strength weighting curve

5.5 "Search by Sound" Music Similarity Retrieval API (by TU WIEN/Spectralmind)

5.5.1 Description

The "Search by Sound" system can analyze music tracks and store their fingerprints (features) in a database. Via an API, one can query for similar sounding songs, using a combination of audio-feature based and meta-data query (e.g. "give me rhythmically similar songs from the genre 'Electronic'").

The system can be set up with a custom (or empty) music library where additional songs can be added via the API. We also provide a pre-analyzed library with 50,000 songs from http://freemusicarchive.org

"Search by Sound" consists of:

- SMAFE Audio Feature Extractor, a server side application that analyzes rhythmic and spectral content of MP3 files and stores them in a database,
- SMINT Audio Similarity System which computes similarities between the different feature vectors of all songs in the database,
- SMINT API: A REST Web API to query for similar songs, as well as adding and removing songs (see Interface Description below), and
- Search by Sound Web Frontend: A web frontend to interactively search for songs in the database, retrieve and listen to similar songs and perform segment searches based on a waveform segment of a song.

5.5.2 Platforms

- SMAFE and SMINT run as binary libraries on Linux systems (optionally Mac OS)
- SMINT API runs on Apache Web Server using PHP 5.3

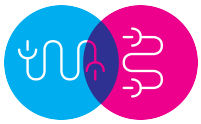
All this is ready hosted at: http://musicbricks.ifs.tuwien.ac.at [2]

- SMINT API can be queried by any application capable of accessing **REST APIs**
- Search by Sound Web Frontend is provided for testing and evaluation purposes (can be also customized/extended on request)

5.5.3 Interface Description

The SMINT API is a REST API which can be queried by using HTTP requests.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



The API is accessible from server http://musicbricks.ifs.tuwien.ac.at

The API methods are:

- **track/add ...** Add a Track by sending a URL where the track a) can be downloaded or b) providing a local file location (on the same server as the API is running).
- **track/delete/:smint_track_id …** Removes a track from the system.
- **track/:smint_track_id** … Returns a list of tracks that are similar to the given trackID.
- **track_external_key/:external_key** … Returns a list of tracks that are similar to the given external key.
- **version** … Returns version information on the API.

The API returns a proper HTTP status code and an XML document.
A separate API description document will be provided.

5.6 Freesound 2.0 Web API (MTG-UPF)

5.6.1 Description

Freesound (www.freesound.org) is a state-of-the-art online collaborative audio database, built utilising MTG-UPFs technologies that contains over 200.000 sounds uploaded and annotated by registered web users. Freesound is the only EU academic database with an API used by commercial organizations as a resource of Creative Commons licensed sound samples. Freesound is continuously growing at a rate of about 120 new sounds and 1000 new registered users per day. The contents of Freesound are very heterogeneous, ranging from environmental recordings to instrument samples, including voice, foley, music loops and sound effects. All these sounds are annotated with user-provided free-form tags and textual descriptions that enable text-based retrieval. Content-based audio features are also extracted from sound samples to provide sound similarity search.

The Freesound API will provide access to:

- Collaborative repository
- CC licensed sounds +200k
- RESTful API
- API Clients:
    - Python, Javascript, Objective-C

With the Freesound API users can browse, search, and retrieve information about Freesound users, packs, and the sounds themselves, of course. Users can find similar sounds to a given target (based on content analysis) and retrieve automatically extracted features from audio files, as well as perform advanced queries combining content analysis features and other metadata (tags, etc. …). With the Freesound API, you can also upload, comment, rate and bookmark sounds.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

5.6.2 Platforms

Freesound API can be queried by any application capable of accessing **REST APIs.**

5.6.3 Interface Description

The following functionalities will be supported by the defined interface:

- **Authentication**: Supports standard token (api key) authentication or OAuth2 for post requests that need to be linked to a Freesound user account.
- **Searching**: Search based on textual terms, audio features, geo-tagging information, or a combination of these.
- **Downloading sounds**: Download original quality sounds or lossy previews with different qualities in a unified format (either mp3 or ogg).
- **Uploading sounds**: Use Oauth2 authentication to link API requests to a Freesound user account and upload sounds to Freesound from a third party application using the API.

Additionally, a set of Client Libraries for Freesound API will be included:

- Python
- Javascript
- Objective-C (iOS)

By using, for example, the Python client for Freesound API, one could make a query and retrieve a list of files using the example code below:

```
import freesound, sys,os
c = freesound.FreesoundClient()
c.set_token("<your_api_key>","token")
results = c.text_search(query="dubstep",fields="id,name,previews")
for sound in results:
    sound.retrieve_preview(".",sound.name+".mp3")
```


The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871

```
print(sound.name)
```

By the end of the tasks in WP3 (M12), we shall be able to provide accompanying example applications such as:

1. Freesound Drum Machine
    a. Web app
    b. Assign FS samples to an online step sequencer

2. Freesound SampleBank Creation
    a. Web app
    b. Specify tags and build a SampleBank archive
    c. Compatibility with common sampler formats (e.g. soundfonts)

5.7 Melody Extraction Library API (MTG-UPF)

5.7.1 Description

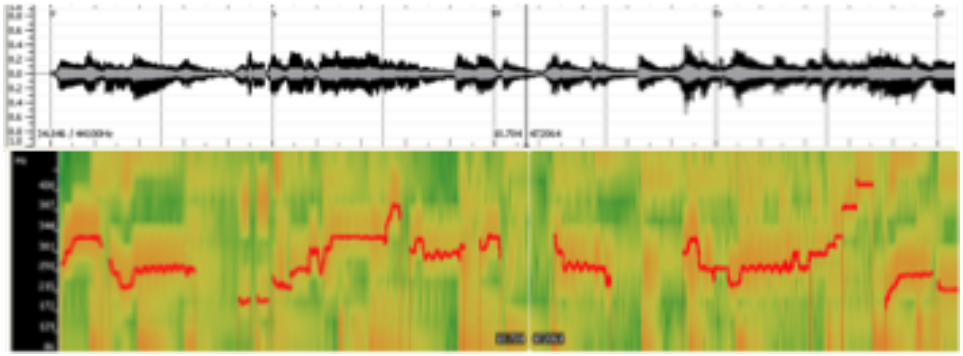
The Melody Extraction API is based on the Essentia Library (http://essentia.upf.edu) [3,4], developed at the Music Technology Group of Universitat Pompeu Fabra in the last years.

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval released under the Affero GPLv3 license (http://www.gnu.org/licenses/agpl.html). It contains an extensive collection of reusable algorithms which implement audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors.

For #MusicBricks, we will provide a specific library for melody extraction, both from solo or a cappella recordings, as well as for predominant melody extraction from polyphonic audio.

Melody is the most salient and identifiable element in a musical piece. It had a prominent role in the origins of music, which relied on oral tradition or the usage of rudimentary instruments (e.g. prehistoric flutes). Still today, melodies are at the musical forefront, their application ranging from an educational context to the latest commercial song hit. In the scope of MusicBricks, we aim at promoting music creation and therefore we consider the use and reuse of melodies as a fundamental step. Two existing and complementary technologies for melody extraction will be integrated in MusicBricks and accessed through an API. The first technology component inputs a cappella recordings by users and the second technology component processes polyphonic mixtures with predominant vocals. As part of the Ideas Incubation stages of MusicBricks (WP6), these melodies could then control external instrument synthesizers and remixed with other musical accompaniment.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



5.7.2 Platforms

The MelodyExtraction library is developed in C++. Python bindings are also provided in order to be able to use Essentia in an interactive development environment, and they fit naturally with the IPython/NumPy/Matplotlib environment (similar to Matlab).

The library is cross-platform and currently supports:

- Linux
- OSX

5.7.3 Interface Description

- Inputs:
  - audio filename (WAV, 44.1kHz, 16bit)
- Output:
  - YAML file with pitch values in Hz and pitch confidence

5.8 Concatenative Synthesis Library API (MTG-UPF)

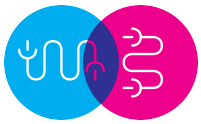
5.8.1 Description

The Concatenative Synthesis API is also based on the Essentia Library (http://essentia.upf.edu), developed at the Music Technology Group of Universitat Pompeu Fabra in the last years. A number of analysis/synthesis algorithms will be made available to transform and concatenate sounds. The main functionality of this API is not to compete with commercial state-of-the-art musical instrument synthesizers, but to provide audio tools to easily manipulate content from Freesound.

We plan to release the Concatenative Synthesis API as part of the Final Version of the API in the second milestone at the end of the first year of the project (M12). The library will be able to transform sounds in two ways: timing (time-scaling) and pitch (pitch-shifting).

With the functionality offered by this API, we will be able to modify sound in a musical way. In music signals, we find a confluence of other musical dimensions such as melody, harmony, rhythm, and timbre. From an acoustic perspective, we can observe a superposition of acoustic layers, usually a mixture of instrumental sound sources. Often, a melody stands out on top of an accompaniment or background. Hundreds of software tools are at disposal of musicians to create instrumental accompaniment by means of common synthesis techniques (from electronic modular synthesizers to remixing tools based on existing loops and instrumentals). MusicBricks brings innovation in the freedom given to Digital Makers for creating accompaniments from a large, heterogeneous and unstructured repository of sounds. To this end, we use an exploratory synthesis technique called concatenative synthesis that accesses sound material from

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



Freesound, the online collaborative audio database. A corpus of sound material is first segmented into short snippets that are automatically analyzed by a set of feature extractors. The synthesized sound is the result of a selection and combination of snippets according to some input data sequence. A possible example is the reconstruction of a 'target' music track of an artist A by snippets of tracks of an artist B according to its acoustic similarity. An outcome of special interest is how non-musical material in FreeSound (e.g. field-recordings) can acquire a musical character when properly segmented, transformed and remixed.

5.8.2 Platforms

The Concatenative Synthesis library is developed in C++. Python bindings are also provided in order to be able to use Essentia in an interactive development environment, and they fit naturally with the IPython/NumPy/Matplotlib environment (similar to Matlab).

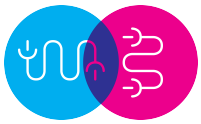
The library is cross-platform and currently supports:

- Linux
- OSX

5.8.3 Interface Description

- Inputs:
  - audio filename (WAV, 44.1kHz, 16bit)
  - Transformation parameters
    - time-scaling envelope: <timestamp, time scale> value pairs
    - pitch shifting envelope: <timestamp, pitch shift> value pairs
- Output:
  - out audio filename (WAV, 44.1kHz, 16bit)

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



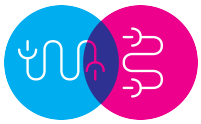
Conclusions

This document D3.1 outlines the API requirements and definitions. During the first three months, research partners have worked on identifying how their technologies can be presented and used in a hackathon context.

As a result, principally we provide here the list of technologies by research partners that will be integrated as APIs in the scope of MusicBricks.

In the following months, the tasks in WP3 will involve the actual implementation of the first version of the APIs. At the same time, these first versions will already be presented and tested in the planned events in MusicTechFest Scandi (Umea, May 29-31 2015) and MusicHackDay Sonar+D (Barcelona, June 17 2015). These activities will be reported in the deliverable D3.2.

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871



References

[1] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, pages 34--41, 2005. [pdf]

[2] "Search by Sound" Web frontend, SMINT Web API, available from http://musicbricks.ifs.tuwien.ac.at

[3] Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. International Society for Music Information Retrieval Conference (ISMIR'13). 493-498.

[4] Essentia Library, MTG-UPF, http://essentia.upf.edu, http://github.com/MTG/essentia

The MusicBricks project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°644871