

# TimeSide

---

Python audio processing framework and server made for the web

Guillaume Pellerin, Antoine Grandry, Martin Desrumaux

POW (Web Team), Innovation and Research Mean Dpt, IRCAM



Séminaire STMS - 14/10/2020 - IRCAM

# Outline

---

- Introduction
- History
- Core
- Server
- Player
- Perspectives

# TimeSide

---

## Context

- digitization > big data > platforms > machine learning > users & listeners
- more data sets, databases, formats, algorithm versions, open DSP libraries & communities
- difficulties to port and scale some algorithms on production platforms (DSP at scale)
- computer science need human data, digital humanities needs computers
- collaborative workflows, citizen sciences through the web

## Constraints

- continuous development, continuous integration
- reproducible research
- sustainable digital archiving
- copyrighted data
- open source and standards
- format evolution and abstraction
- duplicate and version everything
- access everywhere

# TimeSide

---

## History

- 2007 : [Telemeta](#) developed for the sound archives of the CNRS / Musée de l'Homme
- 2010 : TimeSide separation as a autonomous library and then a framework with a plugin oriented architecture
- 2011 : Telemeta v1.0 released and <http://archives.crem-cnrs.fr/>
- 2013 : DIADEMS project (ANR) : external plugins
- 2015 : TimeSide server and RESTFul API prototypes
- 2015 : KAMoulox (ANR), DaCaRyh (Labex) projects
- 2016 : WASABI (ANR), CREM-NYU-Parisson projects

# Telemeta - CREM

 Search

Welcome, Thomas Fillion | Profile | Help | Sign out

Desk Archives Geo Navigator Advanced search Users

## Item : TROMPE ET TAMBOUR :41-15

Title	TROMPE ET TAMBOUR
Original title / translation	OLU BOY
Collector	
Collection	CNRSMH_I_1970_012
Recording date	Nov. 1, 1960 - Nov. 30, 1960
Last modification	Oct. 24, 2012, 5:39 p.m. (admin)

### Geographic and cultural informations

Location	Mali, Afrique occidentale, Afrique
Location details	SANGA
Population / social group	DOGON
Ethnographic context	

### Musical informations

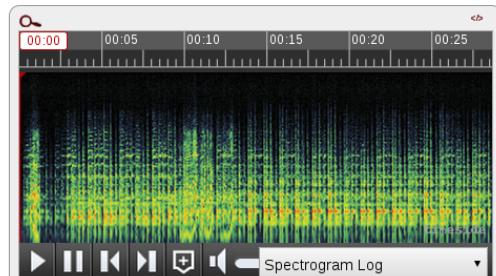
Number	Composition	Vernacular name	Interprets
1	Trompe	KAKELE	
1	Tambour frappé	BOYDUNULE	

### Archiving data

Code	CNRSMH_I_1970_012_041_15
Original code	BM.1970.012.001/46:41-15

Edit Copy Add to playlist Previous

Next



Analysis	Markers
Property	Value
Channels	1
Duration	0:00:28.73
Max level	-3.816
Mean DC shift	-0
MIME type	audio/x-wav
Resolution	24
RMS level	-19.12
Samplerate	48000
	Hz

<http://archives.crem-cnrs.fr/>

# Telemeta - CREM

Desk Archives **Geo Navigator** Advanced search Users

 Geographic Navigator

Map | List

Map Satellite



A world map with numerous red location pins placed across various countries and continents, indicating active users or data points. The map includes labels for major oceans (North Pacific, North Atlantic, South Pacific, South Atlantic, Indian) and countries like Canada, United States, Mexico, Venezuela, Colombia, Brazil, Peru, Bolivia, Chile, Argentina, Iceland, Norway, Sweden, Finland, United Kingdom, France, Spain, Italy, Germany, Poland, Russia, Mongolia, China, South Korea, Japan, India, Pakistan, Afghanistan, Iran, Turkey, Syria, Iraq, Jordan, Lebanon, Israel, Egypt, Sudan, Libya, Algeria, Morocco, Mauritania, Mali, Niger, Burkina Faso, Côte d'Ivoire, Ghana, Togo, Benin, Nigeria, Cameroon, Gabon, Congo, Republic of Congo, Angola, Zambia, Malawi, Mozambique, Zimbabwe, Namibia, Botswana, Lesotho, South Africa, Madagascar, Australia, New Zealand, Papua New Guinea, Indonesia, Thailand, Philippines, Vietnam, Laos, Cambodia, and Mongolia.

North Pacific Ocean

North Atlantic Ocean

South Pacific Ocean

South Atlantic Ocean

Indian Ocean

Google

Terms of Use Report a map error

6 / 38

# Telemeta / TimeSide integration

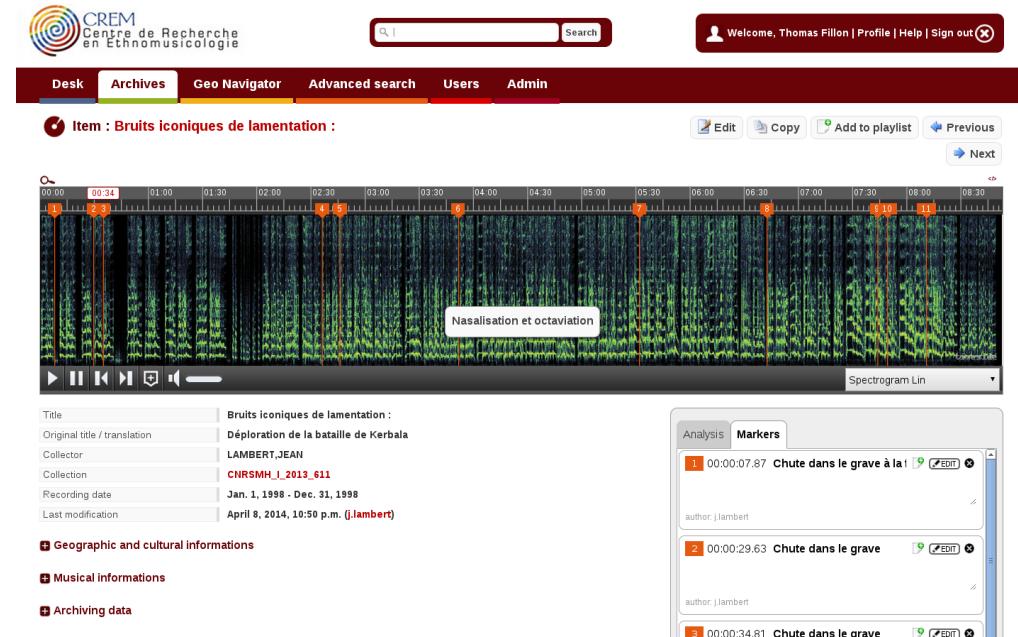


Collaborative multimedia asset management system

MIR + Musicology + Archiving = MIRchiving

## Ecosystem

- 20 public partners
- 15 historical developers (6000 commits)
- 500 users (CREM)
- and thousands of developers! (open source community)
- mutualized development model



<https://github.com/Parisson/Telemeta>

# WASABI project

---

## Web Audio and SemAntics in the Browser for Indexation

- 42 months from 2016 Q4 to april 2020 Q4
- 750 k€ project granted by the french Research National Agency

## Consortium

- INRIA (I3S)
- IRCAM (APM, AnaSyn, POW)
- Parisson
- Deezer R&D
- Radio France

# WASABI project

---

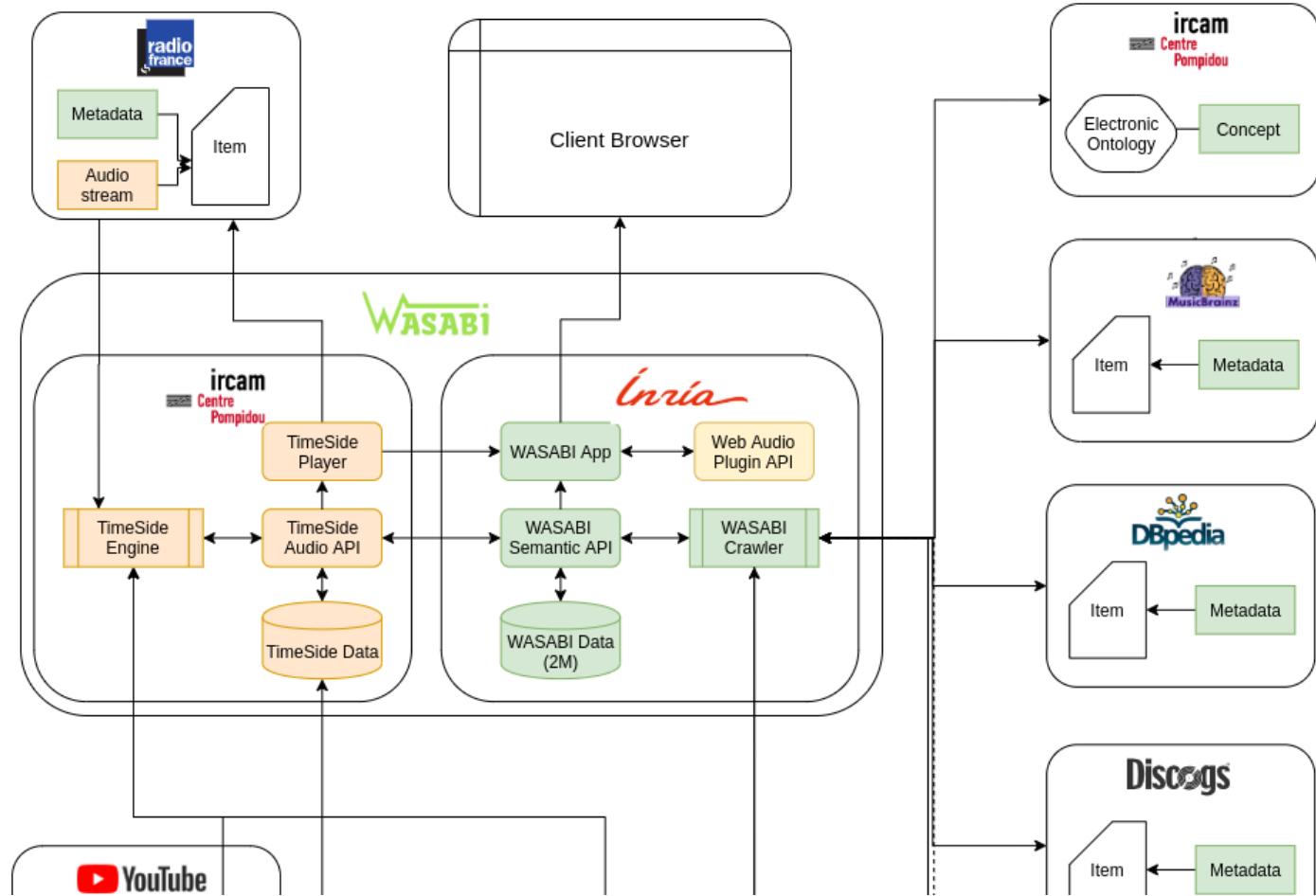
## Objectives

- Propose some new methodologies to index music in the web and audio contexts
- Link semantics (linked metadata) + acoustics (MIR data)
- Develop and publish some open source web services through original APIs

## Use cases

- enhanced web music browsing
- computational musicology
- data journalism
- music learning through modular web interfaces (schools)
- lyrics synchronization

# WASABI platform architecture



# WASABI platform ([link](#))

≡ 🔍 Led Zeppelin ⌂

SHOW RDF



Led Zeppelin

W f t a y s p m n D E Z E P P I N G W W W R i c o n s D H T h

Led Zeppelin were an English rock band formed in London in 1968. The group consisted of guitarist Jimmy Page, singer Robert Plant, bassist and keyboardist John Paul Jones, and drummer John Bonham. The band's heavy, guitar-driven sound, rooted in blues and psychedelia on their early albums, has earned them recognition as one of the progenitors of heavy metal, though their unique style drew from a wide variety of influences, including folk music and blues. After changing their name from the New Yardbirds, Led Zeppelin signed a deal with Atlantic Records that afforded them considerable artistic freedom. Although the group was initially unpopular with critics, they achieved significant commercial success with albums such as Led Zeppelin (1969), Led Zeppelin II (1969), Led Zeppelin III (1970), their untitled fourth album (1971), Houses of the Holy (1973), and Physical Graffiti (1975). Their fourth album, which features the track "Stairway to Heaven", is among the most popular and influential works in rock music, and it helped to secure the group's popularity. Page wrote most of Led Zeppelin's music, particularly early in their career, while Plant generally supplied the lyrics. Jones' keyboard-based compositions later became central to the group's catalogue, which featured increasing experimentation. The latter half of their career saw a series of record-breaking tours that earned the group a reputation for excess and debauchery. Although they remained commercially and critically successful, their output and touring schedule were limited during the late 1970s, and the group disbanded following Bonham's death from alcohol-related asphyxia in 1980. In the decades that followed, the surviving members sporadically collaborated and participated in one-off Led Zeppelin reunions. The most successful of these was the 2007 Ahmet Ertegun Tribute Concert in London, with Jason Bonham taking his late father's place behind the drums. Led Zeppelin are widely considered one of the most successful, innovative, and influential rock groups in history. They are one of the best-selling music artists in the history of audio recordings.

# WASABI platform ([link](#))

The screenshot shows the WASABI platform's user interface. At the top, there is a blue header bar with a search bar containing the text "Led Zeppelin". Below the header is a navigation menu with links to "Home", "DiscoveryHub", "QMUL/chords", "IRCAM/Timeside", "WebAudio tools", "WebAudio plugins", and "Multitracks". A prominent blue button labeled "SHOW RDF" is visible. The main content area displays a search result for "Led Zeppelin". On the left, there is a thumbnail image of the band and the text "Led Zeppelin". On the right, there is another thumbnail image of an album cover and the text "Led Zeppelin II". A yellow star icon is positioned above the "Led Zeppelin II" section. The title "Whole Lotta Love" is centered below the thumbnails. Below the title is a video player showing a thumbnail for "Led Zeppelin - Whole Lotta Love (Official M...)" with a play button in the center. At the bottom of the main content area, there are several small icons: a white "W", a red YouTube icon, a black "DH" icon, and a purple "T" icon. A detailed description of the song follows:

"Whole Lotta Love" is a song by English hard rock band Led Zeppelin. It is the opening track on the band's second album, *Led Zeppelin II*, and was released in the United States and Japan as a single. The US release became their first hit single; it was certified Gold on 13 April 1970, having sold one million copies. As with other Led Zeppelin songs, no single was released in the United Kingdom, but singles were released in Germany (where it reached number one), the Netherlands (where it reached number four), Belgium and France. In 2004, the song was ranked number 75 on Rolling Stone magazine's list of the 500 Greatest Songs of All Time, and in March 2005, Q magazine placed "Whole Lotta Love" at number three in its list of the 100 Greatest Guitar Tracks. It was placed 11 on a similar list by Rolling Stone. In 2009 it was named the third greatest hard rock song of all time by VH1. Already part of their live repertoire, "Whole Lotta Love" saw its first official release on the LP *Led Zeppelin II* on 22

# timeside.core

---

# TimeSide

---

Python audio processing framework and server made for the web

<https://github.com/Parisson/TimeSide>

## Goals

- **Process** audio fast and asynchronous with **Python**,
- **Decode** audio frames from *any* audio or video media format into **Numpy arrays**,
- **Analyze** audio content with some **state-of-the-art** audio feature extraction libraries like **Aubio**, **Essentia**, **Librosa**, **Yaafe**, **VAMP** and pure python processors
- **Visualize** audio data with various fancy waveforms, spectrograms and other cool graphers,
- **Transcode** audio data in various media formats and stream them through web apps,
- **Serialize** feature analysis data through various portable formats (XML, JSON, HDF5)
- **Playback and interact on demand** through a smart high-level **HTML5 extensible player**,
- **Index, tag and annotate** audio archives with **cultural and semantic metadata**,
- **Deploy and scale** your own audio processing engine flawlessly through any infrastructure with **Docker**

# TimeSide

---

Python audio processing framework and server made for the web

<https://github.com/Parisson/TimeSide>

## Use cases

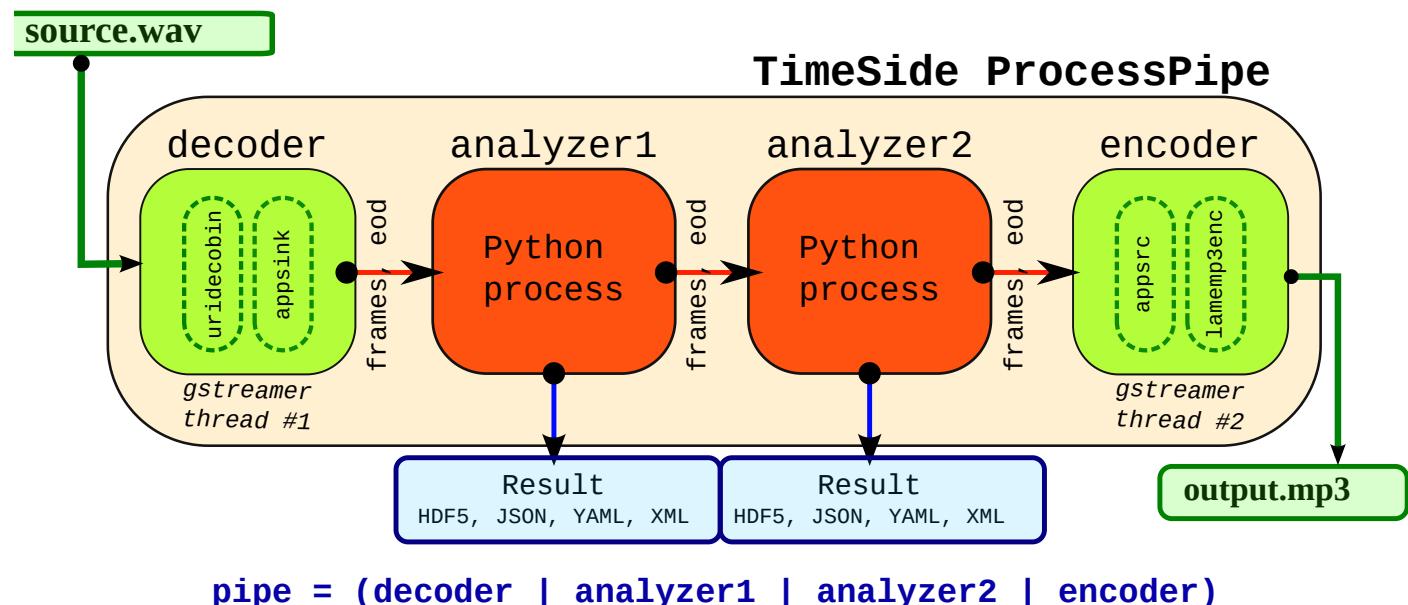
- Scaled audio processing (filtering, transcoding, machine learning, etc...)
- Audio process prototyping
- Audio dynamic visualization
- Automatic segmentation and labelling synchronized with audio events
- Collaborative annotation
- Audio web services

License: AGPL v2

# timeside.core

## API & architecture

- streaming oriented core engine
- data persistence



## API & architecture

- streaming oriented core engine
- data persistence
- processing API
- plugin architecture
- namespace

```
class DummyAnalyzer(Analyzer):
    """A dummy analyzer returning random samples from audio frames"""
    implements(IAnalyzer)

    @interfacedoc
    def setup(self, channels=None, samplerate=None, blocksize=None, totalfram
        super(DummyAnalyzer, self).setup(channels, samplerate, blocksize, tot
        self.values = numpy.array([0])

    @staticmethod
    @interfacedoc
    def id():
        return "dummy"

    @staticmethod
    @interfacedoc
    def name():
        return "Dummy analyzer"

    @staticmethod
    @interfacedoc
    def unit():
        return "None"

    def process(self, frames, eod=False):
        size = frames.size
        if size:
            index = numpy.random.randint(0, size, 1)
            self.values = numpy.append(self.values, frames[index])
        return frames, eod

    def post_process(self):
        result = self.new_result(data_mode='value', time_mode='global')
        result.data_object.value = self.values
```

# timeside.core

## API & architecture

- streaming oriented core engine
- data persistence
- processing API
- plugin architecture
- namespace
- ~500 unit tests
- docker packaged (Linux, OSX, Windows)

```
import timeside.core
from timeside.core import get_processor
from timeside.core.tools.test_samples import samples

wavfile = samples['sweep.wav']
decoder = get_processor('file_decoder')(wavfile)
grapher = get_processor('spectrogram')()
analyzer = get_processor('level')()
encoder = get_processor('vorbis_encoder')('sweep.ogg')

pipe = (decoder | grapher | analyzer | encoder)
pipe.run()

grapher.render(output='spectrogram.png')
print('Level:', analyzer.results)
Level: {'level.max': AnalyzerResult(...), 'level.rms': AnalyzerResult(...)}
```

```
$ git clone --recursive https://github.com/Parisson/TimeSide.git
$ docker-compose run app python3 manage.py shell
$ docker-compose up
```

<https://timeside.readthedocs.io/en/latest/index.html>

# timeside.core

---

## Plugin examples

- FileDecoder, ArrayDecoder, LiveDecoder, AubioDecoder
- VorbisEncoder, WavEncoder, Mp3Encoder, FlacEncoder, OpusEncoder, etc.
- WaveformAnalyzer, SpectrogramAnalyzer
- AubioTemporal, AubioPitch, etc.
- Yaafe wrapper (graph oriented)
- librosa (function oriented)
- VampPyHost
- Essentia bridge
- ...
- Speech detection
- Music detection
- Singing voice detection
- Monophony / Polyphony
- Dissonance
- Timbre Toolbox

<https://github.com/DIADEMS/timeside-diadems>

# timeside.core

---

## What's new?

0.9.4 > 1.0.0a (released yesterday!  )

- 674 commits, 7 contributors
- Python 2.7 to 3.7
- Drop GStreamer for Aubio as default decoder and encoder
- Regroup all dependencies on pip requirements, drop conda
- Add `Provider` object
- Add Provider plugins `deezer-preview`, `deezer-complete` and `youtube`
- Add loggers
- Improve Docker packaging and image building

timeside.server

---

# timeside.server

## RESTful API built on TimeSide

👉 <https://sandbox.wasabi.telemeta.org/timeside/api/>

### Use cases

- **Upload** audio tracks
- **Retrieve** audio tracks from remote providers
- **Stream** original or transcoded sources
- **Run** on-demand analysis
- **Customize** processors parameters
- **Collect** track's annotation to build audio corpora
- **Deliver** and share several types of results:
  - transcoded audio
  - serialized analysis as JSON or image
- **Export/import** an audio analysis dataset

### Api Root

The default basic root view for DefaultRouter

GET /timeside/api/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{  
    "users": "https://sandbox.wasabi.telemeta.org/timeside/api/users/",  
    "items": "https://sandbox.wasabi.telemeta.org/timeside/api/items/",  
    "providers": "https://sandbox.wasabi.telemeta.org/timeside/api/providers/",  
    "selections": "https://sandbox.wasabi.telemeta.org/timeside/api/selections/",  
    "processors": "https://sandbox.wasabi.telemeta.org/timeside/api/processors/",  
    "subprocessors": "https://sandbox.wasabi.telemeta.org/timeside/api/subprocessors/",  
    "presets": "https://sandbox.wasabi.telemeta.org/timeside/api/presets/",  
    "experiences": "https://sandbox.wasabi.telemeta.org/timeside/api/experiences/",  
    "tasks": "https://sandbox.wasabi.telemeta.org/timeside/api/tasks/",  
    "analysis": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis/",  
    "analysis_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis_tracks/",  
    "annotation_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/annotation_tracks/",  
    "annotations": "https://sandbox.wasabi.telemeta.org/timeside/api/annotations/",  
    "results": "https://sandbox.wasabi.telemeta.org/timeside/api/results/"  
}
```

OPTIONS

GET



# timeside.server

## RESTful API built on TimeSide

👉 <https://sandbox.wasabi.telemeta.org/timeside/api/>

### Server design

- Based on Django REST Framework (DRF)
- **Interoperability** between other servers or frontends and TimeSide instance and its data
- Object-relational **database** in order to store music tracks and processing results
- **Models**: define essential fields and behaviors of stored data
- **queue-worker architecture** enables to run tasks asynchronously

### Api Root

The default basic root view for DefaultRouter

GET /timeside/api/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{  
    "users": "https://sandbox.wasabi.telemeta.org/timeside/api/users/",  
    "items": "https://sandbox.wasabi.telemeta.org/timeside/api/items/",  
    "providers": "https://sandbox.wasabi.telemeta.org/timeside/api/providers/",  
    "selections": "https://sandbox.wasabi.telemeta.org/timeside/api/selections/",  
    "processors": "https://sandbox.wasabi.telemeta.org/timeside/api/processors/",  
    "subprocessors": "https://sandbox.wasabi.telemeta.org/timeside/api/subprocessors/",  
    "presets": "https://sandbox.wasabi.telemeta.org/timeside/api/presets/",  
    "experiences": "https://sandbox.wasabi.telemeta.org/timeside/api/experiences/",  
    "tasks": "https://sandbox.wasabi.telemeta.org/timeside/api/tasks/",  
    "analysis": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis/",  
    "analysis_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis_tracks/",  
    "annotation_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/annotation_tracks/",  
    "annotations": "https://sandbox.wasabi.telemeta.org/timeside/api/annotations/",  
    "results": "https://sandbox.wasabi.telemeta.org/timeside/api/results/"  
}
```

OPTIONS

GET ▾

# timeside.server

## RESTful API built on TimeSide

👉 <https://sandbox.wasabi.telemeta.org/timeside/api/>

### What's new on server?

- Add audio providers (Deezer, Youtube)
- Switch from MySQL to PostgreSQL
- Add a JWT authentication
- Make the API follow the OpenAPI specification
- Build a TypeScript SDK on the REST API
- Add several tools, views, models and serializers
- Improve server unit testing
- Fix few bugs
- Python, Django, DRF and Celery upgrades

### Api Root

The default basic root view for DefaultRouter

GET /timeside/api/

HTTP 200 OK  
Allow: GET, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{  
    "users": "https://sandbox.wasabi.telemeta.org/timeside/api/users/",  
    "items": "https://sandbox.wasabi.telemeta.org/timeside/api/items/",  
    "providers": "https://sandbox.wasabi.telemeta.org/timeside/api/providers/",  
    "selections": "https://sandbox.wasabi.telemeta.org/timeside/api/selections/",  
    "processors": "https://sandbox.wasabi.telemeta.org/timeside/api/processors/",  
    "subprocessors": "https://sandbox.wasabi.telemeta.org/timeside/api/subprocessors/",  
    "presets": "https://sandbox.wasabi.telemeta.org/timeside/api/presets/",  
    "experiences": "https://sandbox.wasabi.telemeta.org/timeside/api/experiences/",  
    "tasks": "https://sandbox.wasabi.telemeta.org/timeside/api/tasks/",  
    "analysis": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis/",  
    "analysis_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis_tracks/",  
    "annotation_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/annotation_tracks/",  
    "annotations": "https://sandbox.wasabi.telemeta.org/timeside/api/annotations/",  
    "results": "https://sandbox.wasabi.telemeta.org/timeside/api/results/"  
}
```

OPTIONS    GET ▾

# timeside.server

## RESTful API built on TimeSide

👉 <https://sandbox.wasabi.telemeta.org/timeside/api/>

### Models

- Item: audio content and metadata (external id)
- Provider: provide audio from a given platform
- Selection: list of items (corpus)
- Processor: plugins with version and default parameters
- Preset: processor and a set of parameters
- Experience: list of presets forming a pipe (reproducible)
- Task: an experience and a selection
- Result: transcoded audio or numerical outputs (hdf5 file)
- Annotation: label audio file on a given time or segment

### Api Root

The default basic root view for DefaultRouter

GET /timeside/api/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{  
    "users": "https://sandbox.wasabi.telemeta.org/timeside/api/users/",  
    "items": "https://sandbox.wasabi.telemeta.org/timeside/api/items/",  
    "providers": "https://sandbox.wasabi.telemeta.org/timeside/api/providers/",  
    "selections": "https://sandbox.wasabi.telemeta.org/timeside/api/selections/",  
    "processors": "https://sandbox.wasabi.telemeta.org/timeside/api/processors/",  
    "subprocessors": "https://sandbox.wasabi.telemeta.org/timeside/api/subprocessors/",  
    "presets": "https://sandbox.wasabi.telemeta.org/timeside/api/presets/",  
    "experiences": "https://sandbox.wasabi.telemeta.org/timeside/api/experiences/",  
    "tasks": "https://sandbox.wasabi.telemeta.org/timeside/api/tasks/",  
    "analysis": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis/",  
    "analysis_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/analysis_tracks/",  
    "annotation_tracks": "https://sandbox.wasabi.telemeta.org/timeside/api/annotation_tracks/",  
    "annotations": "https://sandbox.wasabi.telemeta.org/timeside/api/annotations/",  
    "results": "https://sandbox.wasabi.telemeta.org/timeside/api/results/"  
}
```

OPTIONS    GET ▾

# timeside.server

## RESTful API documentation

auto-generated thanks to ReDoc on top of OpenAPI ↗ <https://sandbox.wasabi.telemeta.org/timeside/api/docs/>

The screenshot shows the API documentation for the `listItems` endpoint. On the left, a sidebar lists various API endpoints with their HTTP methods and descriptions:

- `GET retrieveItem`
- `PUT updateItem`
- `PATCH partialUpdateItem`
- `DEL destroyItem`
- `GET listProviders`
- `GET retrieveProvider`
- `GET listSelections`
- `POST createSelection`
- `GET retrieveSelection`
- `PUT updateSelection`
- `PATCH partialUpdateSelection`
- `DEL destroySelection`
- `GET listProcessors`

The main content area for `listItems` includes:

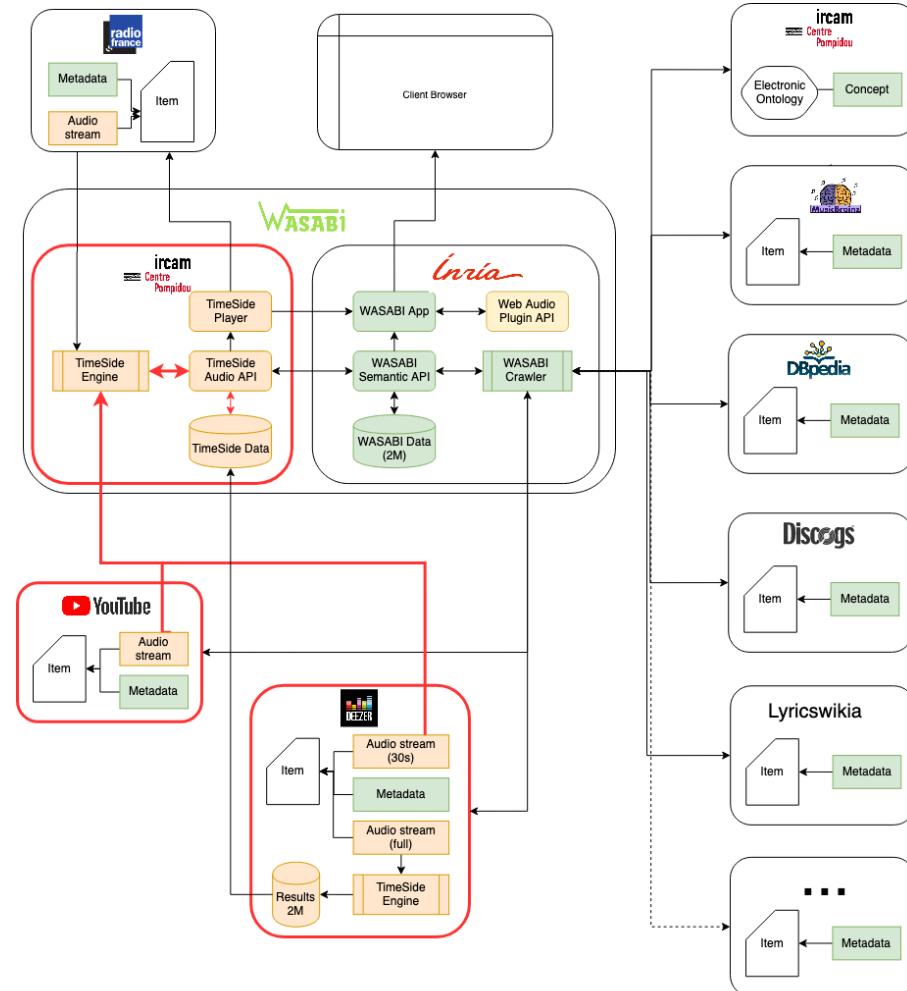
- AUTHORIZATIONS:** `bearerAuth`
- QUERY PARAMETERS:**
  - `provider_pid`: string
  - `external_id`: string
- Responses:**
  - > 200**:
    - Content type**: application/json
    - Copy Expand all Collapse all
    - [
      - {
        - `"uuid": "095be615-a8ad-4c33-8e9c-c7612`
        - `"url": "string"`
        - `"title": "string"`
        - `"description": "string"`
        - `"player_url": "string"`
        - `"source_file": "string"`
        - `"source_url": "http://example.com"`
        - `"mime_type": "string"`

# timeside.server

## Workflow examples in WASABI Project

### With providers

- Youtube
  - based on `youtube-dl`
  - must be adaptable to YouTube's changes
- Deezer 30 seconds long preview
  - consuming Deezer's API
  - find another solution to full contents



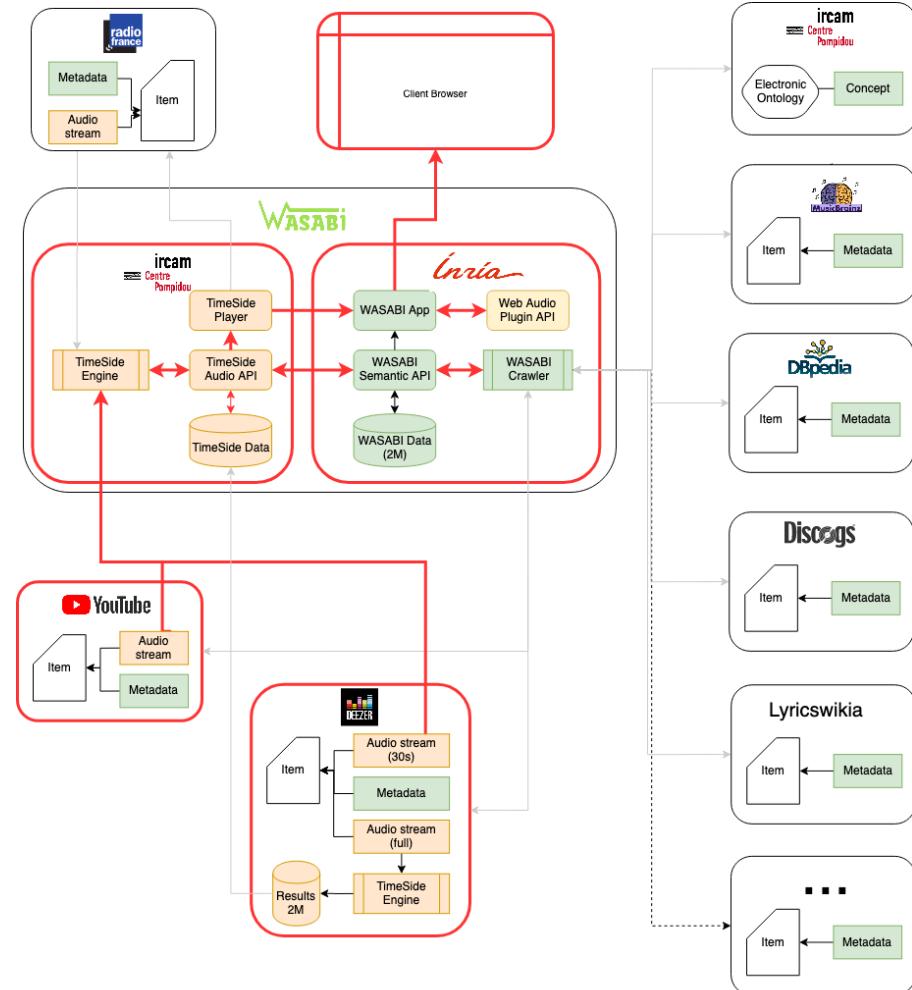
# timeside.server

## Workflow examples in WASABI Project

With server

POC of a webservice

- delivers audio analysis to another remote server
- enhance its musical metadata with results



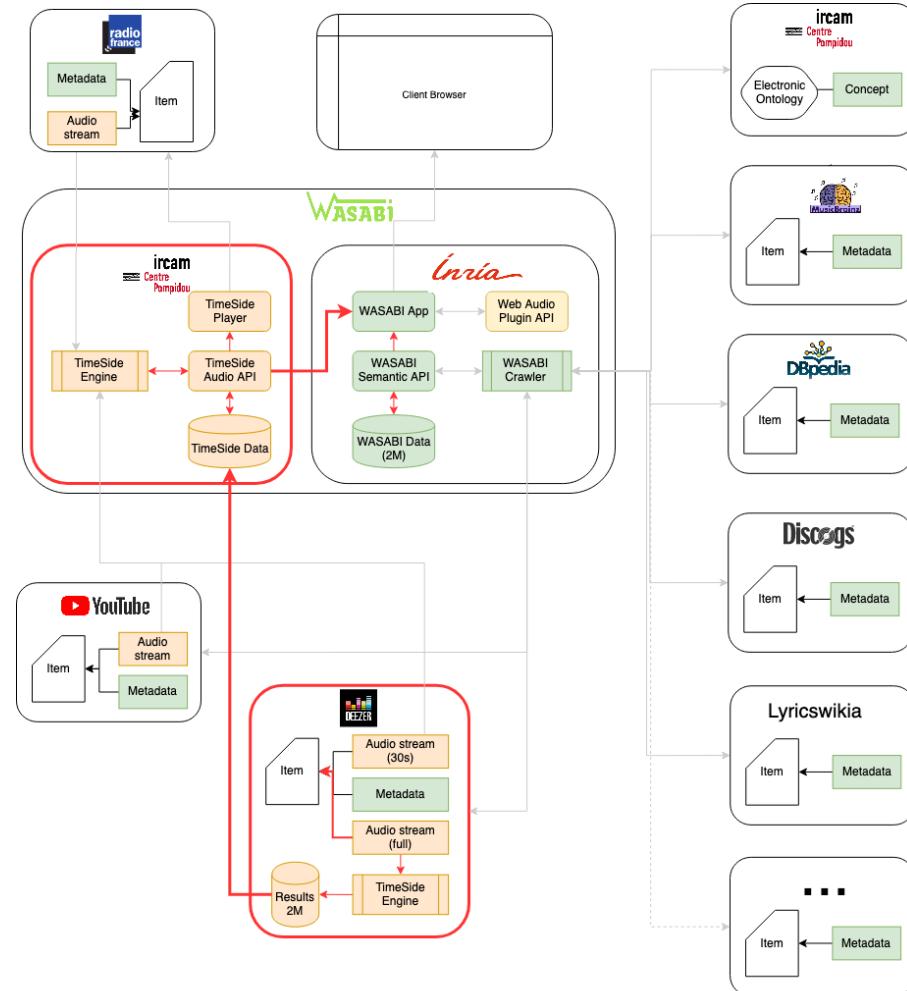
# timeside.server

## Workflow examples in WASABI Project

Import/export of a run on Deezer's infrastructure

POC of a sharing system of an audio analysis datasets

- easy deployment thanks to docker
- audio does not have to be shared



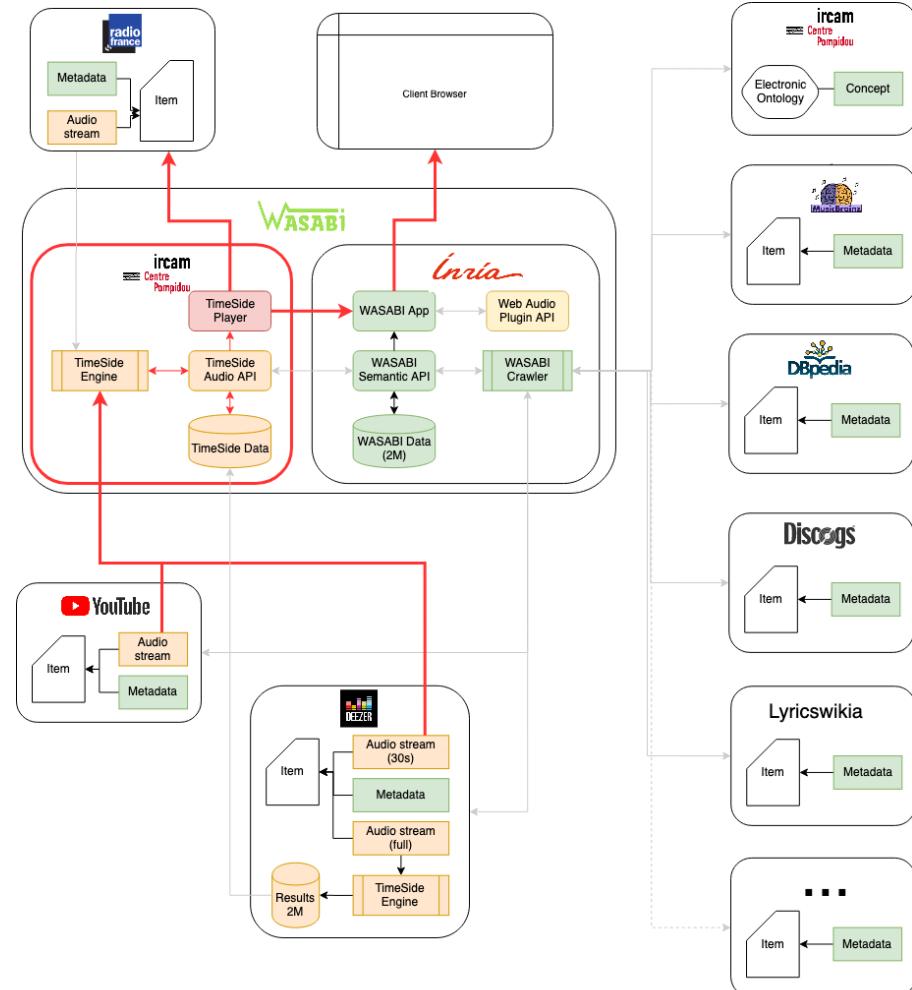
# timeside.server

## Workflow examples in WASABI Project

With a frontend player

POC of a JavaScript app that consume the Rest API

- serialize data as JSON or image
- deliver analysis track (spectrogram, waveform)
- deliver or get annotations on audio track



timeside.player

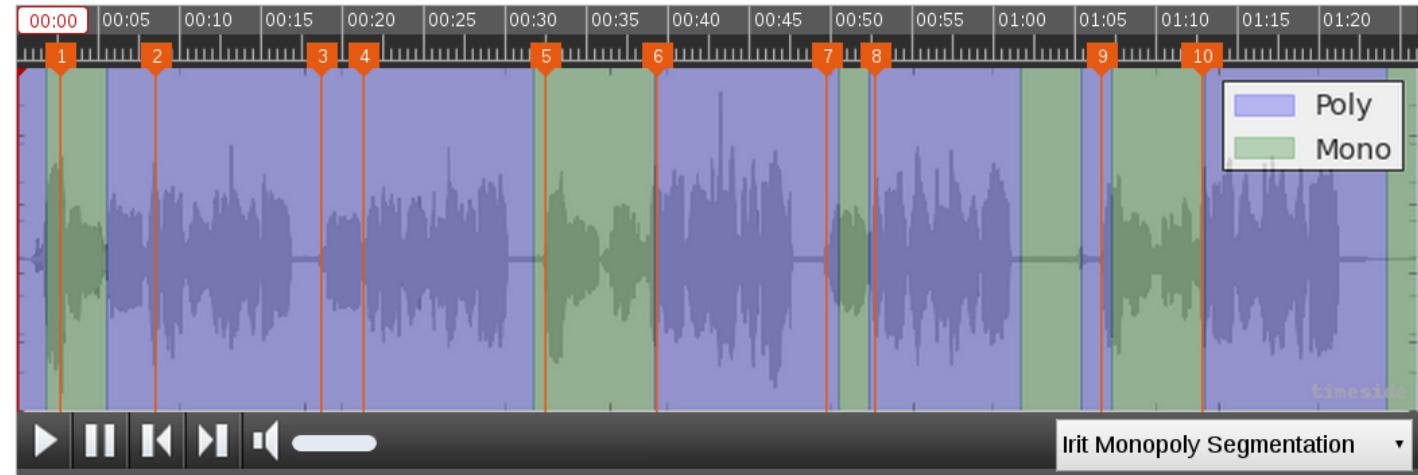
---

# timeside.player

---

## v1 (SoundManager2)

- on demand processing
- simple marker annotation
- bitmap image cache only

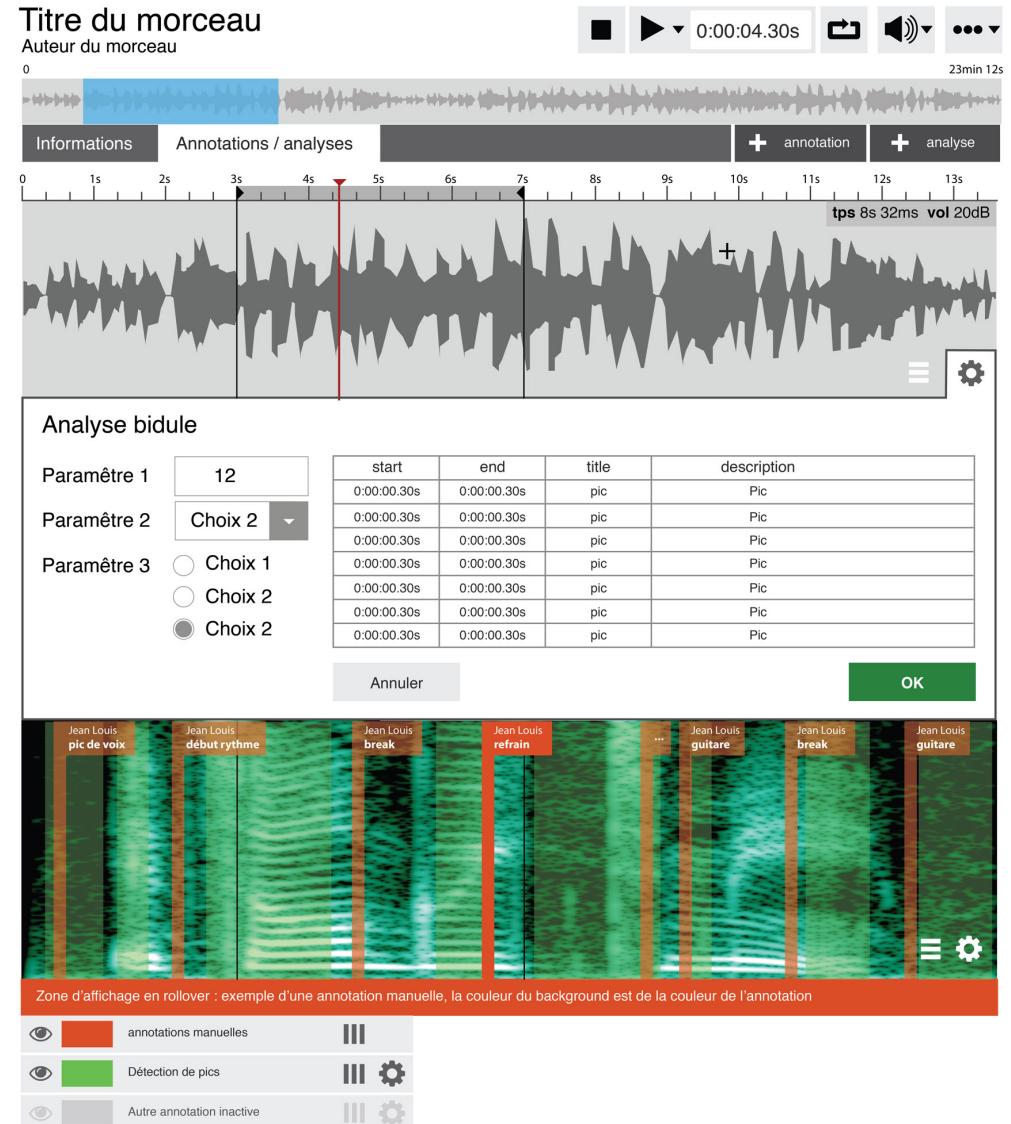


# timeside.player

v2

## Constraints

- Handling multiple hours audio files
- Multiple user annotations and analysis tracks
- Analysis rendered on the frontend
- Zooming



# timeside.player

---

## API SDK (client library)

- Timeside API: 75 routes
- openapi-generator
  - Typescript
  - Fetch
  - OpenAPI v3 Schemas
- Improve schema support on DRF (PR)
  - Components
  - Customize default names
- Glue code
  - Authentication
  - Initialization on Browser / Node
- Documentation
- SDK: <https://github.com/Ircam-Web/timeside-sdk-js>
- Node: <https://github.com/Ircam-Web/timeside-scripts>

```
/timeside/api/analysis/:  
  get:  
    operationId: listAnalysis  
    parameters: []  
    responses:  
      '200':  
        content:  
          application/json:  
            schema:  
              type: array  
              items:  
                $ref: '#/components/schemas/Analysis'
```

Opportunity: `openapi-generator` also supports Python, C/C++, Ruby, Go, Rust etc...

# timeside.player

---

## Player

### Technologies

- Vue (composition-api): DOM Manipulation, Data reactivity
- D3 (SVG): Render waveform / Analysis
- HTML5 Audio
- Web Animations API
- Resize Observer
- Github Action for continuous test & deployment (npm, gh-page)
- Github: <https://github.com/ircam-web/timeside-player>

### Usage

- Standalone app
- Web library
  - React
  - Vue
  - HTML



Tested on Firefox & Chrome

# Demo time!

---

# Perspectives

---

## Audio processing web service (SaaS)

### TODO

- Clustering and orchestration (Kubernetes, prototype ready)
- Implementing Websocket, ServerEvent or Webhook to avoid task status polling
- Parameters
- Scales
- Split repositories : core / server
- Test some JS MIR librairies
- More documentation, notebooks and tests

### Use cases

- MIRchiving (Telemeta 2, CMS embedding, MNHN, UNAM, UNESCO)
- Digitization and media packaging services (VectraCom, VDM, Gecko)
- Metadata enhanced streaming services (Spotify, Deezer, SoundCloud, Netflix)

### Dual licencing

- open source community release of the core framework (AGPL)
- proprietary (entreprise) release (SATT Lutech / Parisson / IRCAM Amplify)

# Thank you and kudos to all contributors!

---



<http://wasabihome.i3s.unice.fr/>

guillaume.pellerin@ircam.fr / @yomguy

