TimeSide

Python audio processing framework and server made for the web

Guillaume Pellerin, Antoine Grandry, Martin Desrumaux

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

WASABI

ircam Centre Pompidou

POW

parisson.com

telemeta

cnrs

SORBONNE UNIVERSITÉ

ANR

MINISTÈRE DE LA CULTURE
Liberté
Égalité
Fraternité

Séminaire STMS - 14/10/2020 - IRCAM

Outline

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

TimeSide

Context

- digization > 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

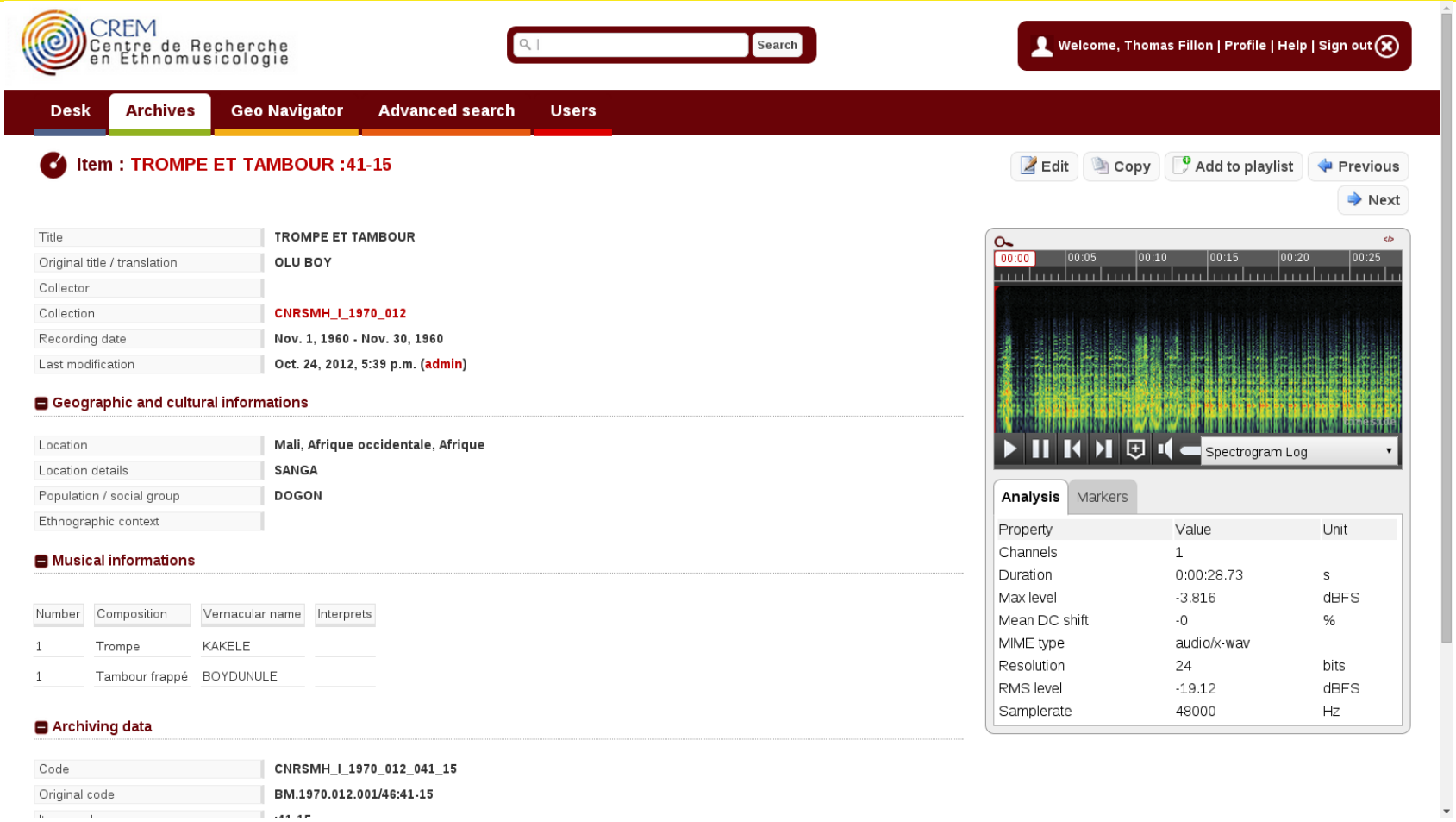
- continuous development, continuous integration
- reproducible research
- sustainable digital archiving
- copyrighted data
- open source and standards
- format evolution and abstraction
- duplicate and version everything
- access everywhere

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



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

Telemeta - CREM

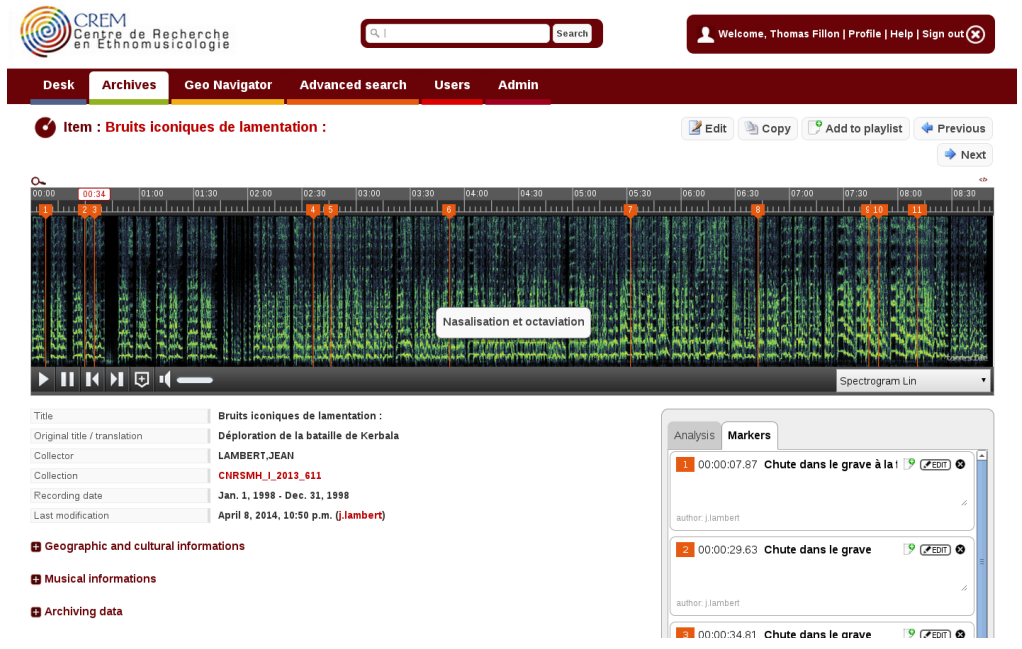
Telemeta / TimeSide integration

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

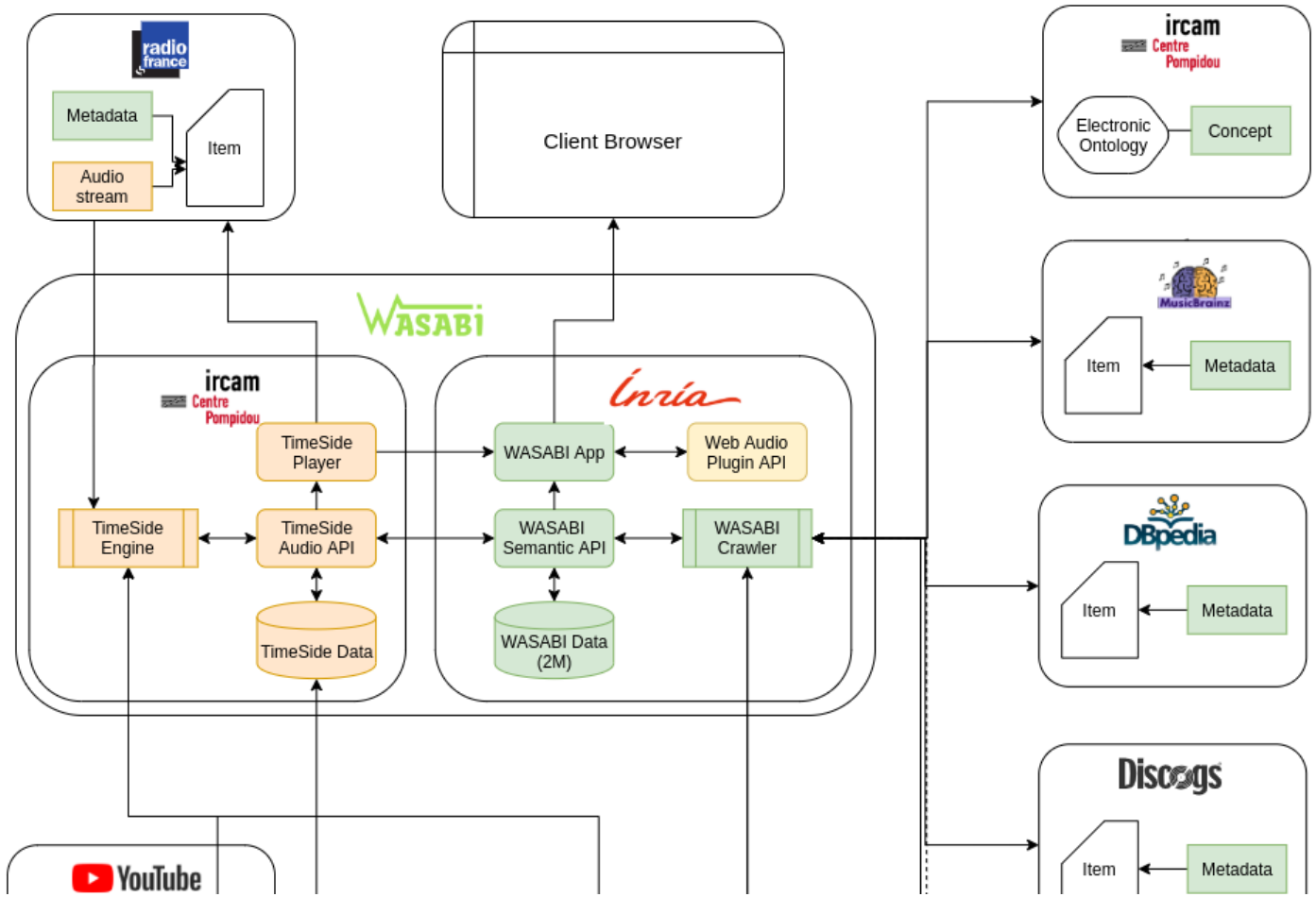
Objectives

- 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

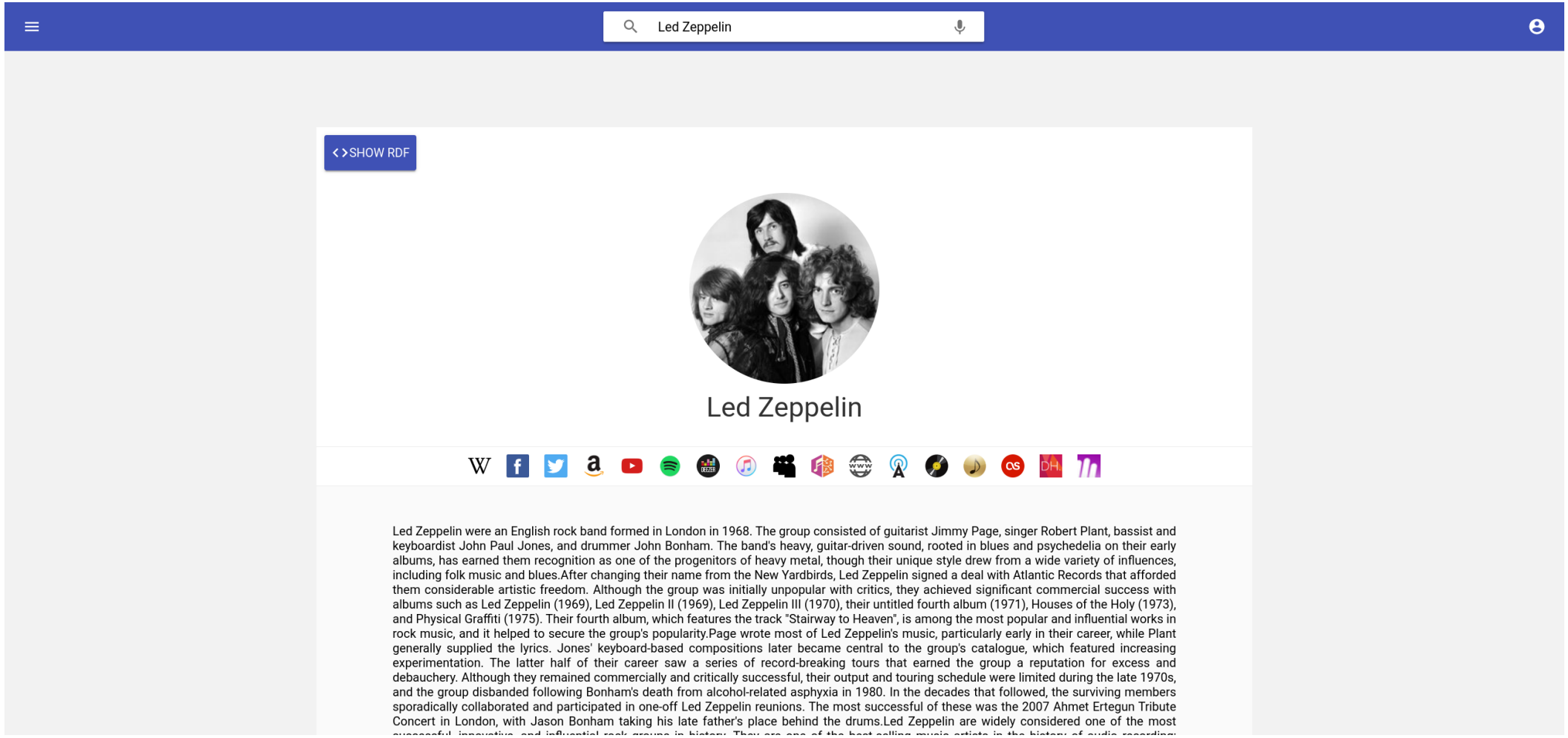
Use cases

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

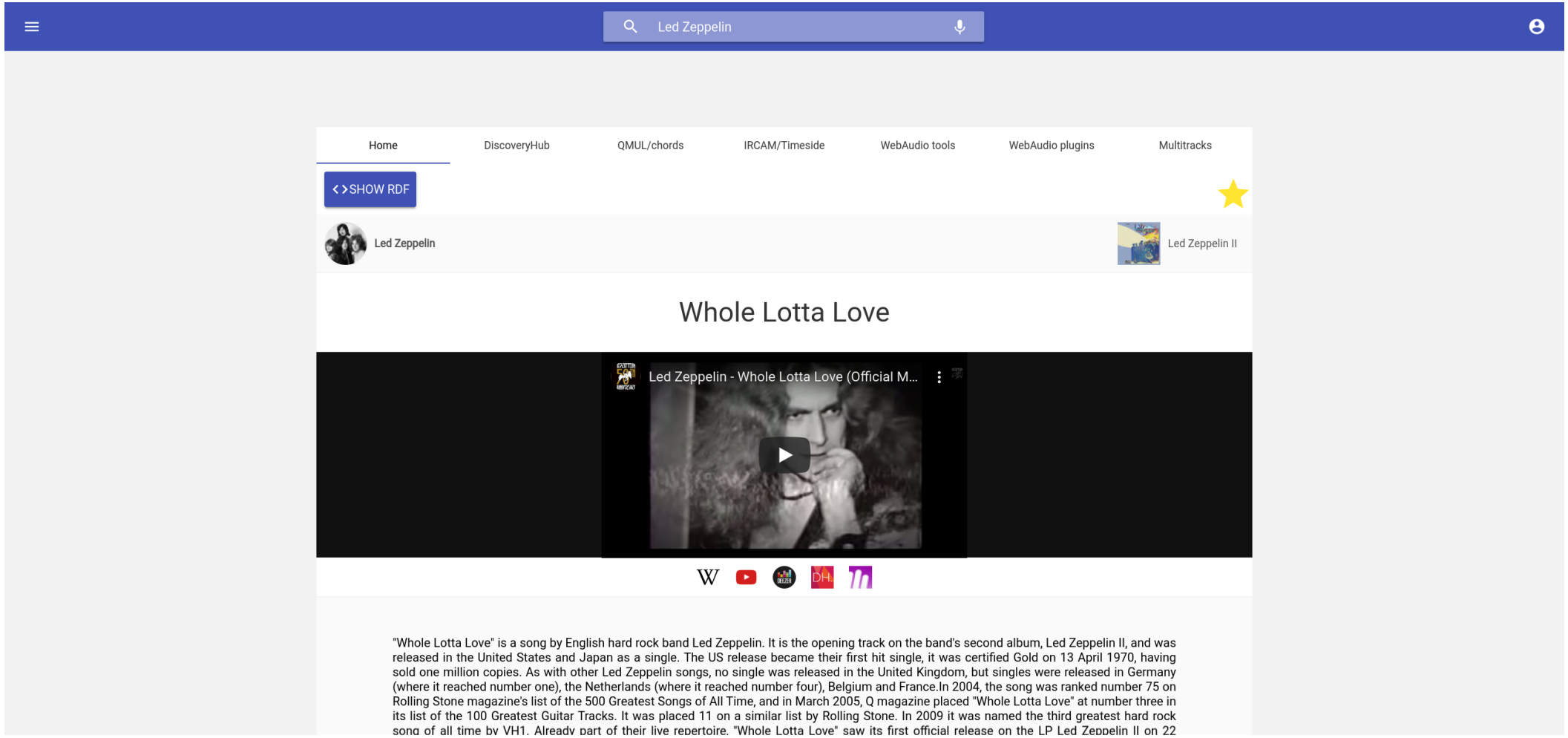
WASABI platform architecture



WASABI platform (link)



WASABI platform (link)



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

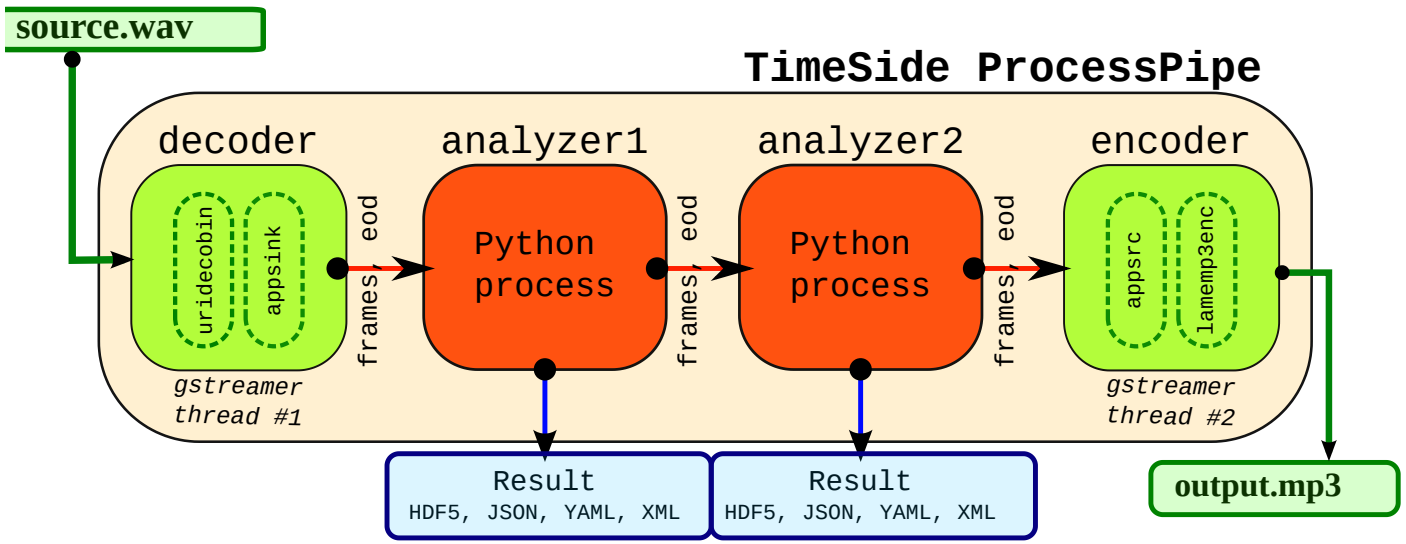
- Scaled audio processing (filtering, transcoding, machine learning, etc...)
- Audio process prototyping
- Audio dynamic visualization
- Automatic segmentation and labelling synchronized with audio events
- Collaborative annotation
- Audio web services

License: AGPL v2

timeside.core

API & architecture

- streaming oriented core engine
- data persistence




```
pipe = (decoder | analyzer1 | analyzer2 | encoder)
```

timeside.core

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')
```

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

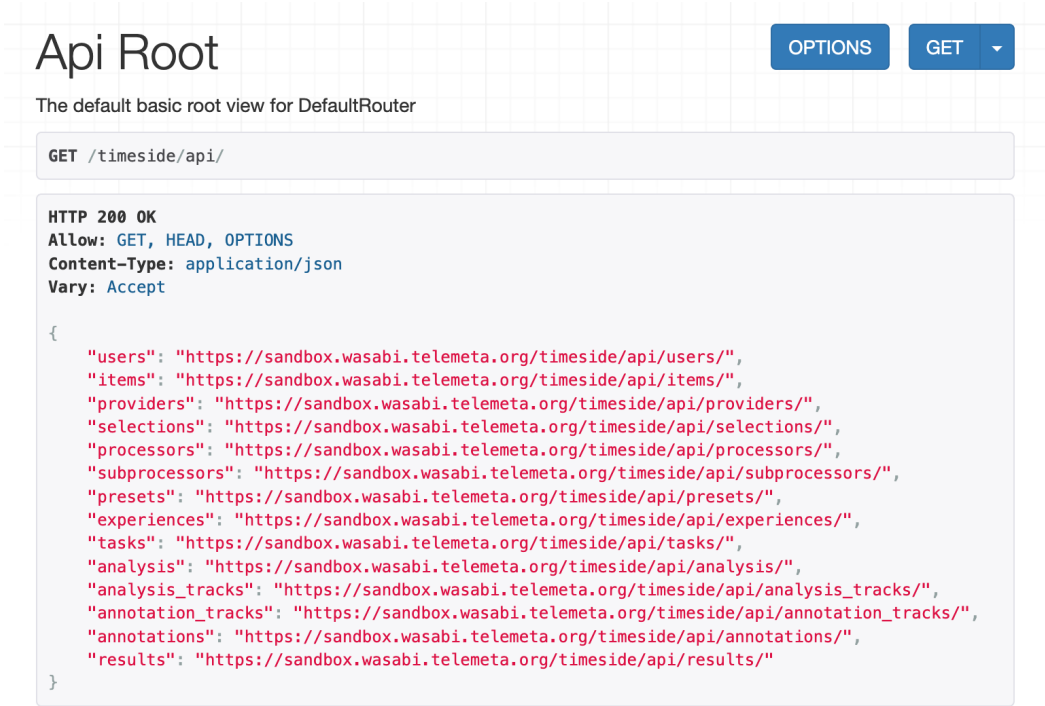
timeside.server

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



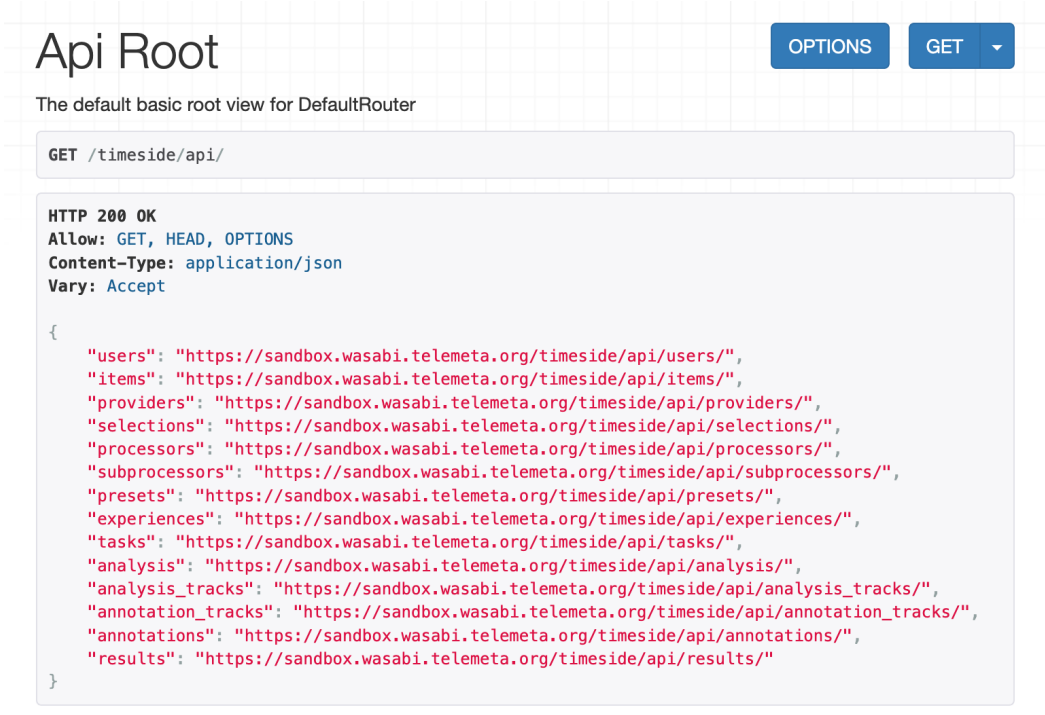
timeside.server

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



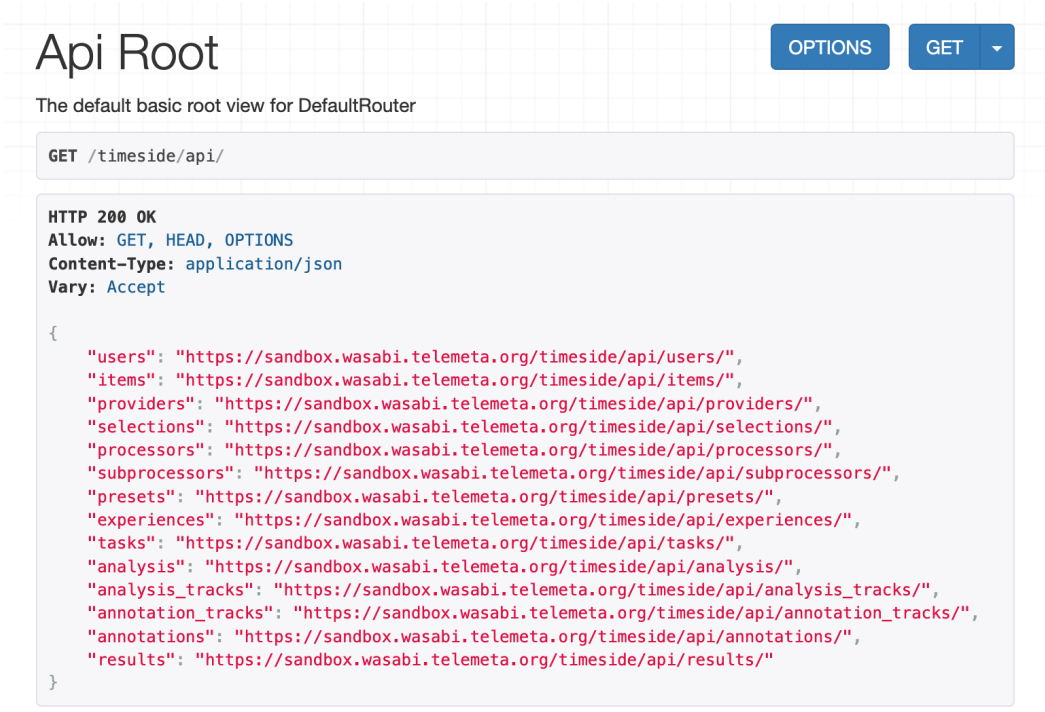
timeside.server

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



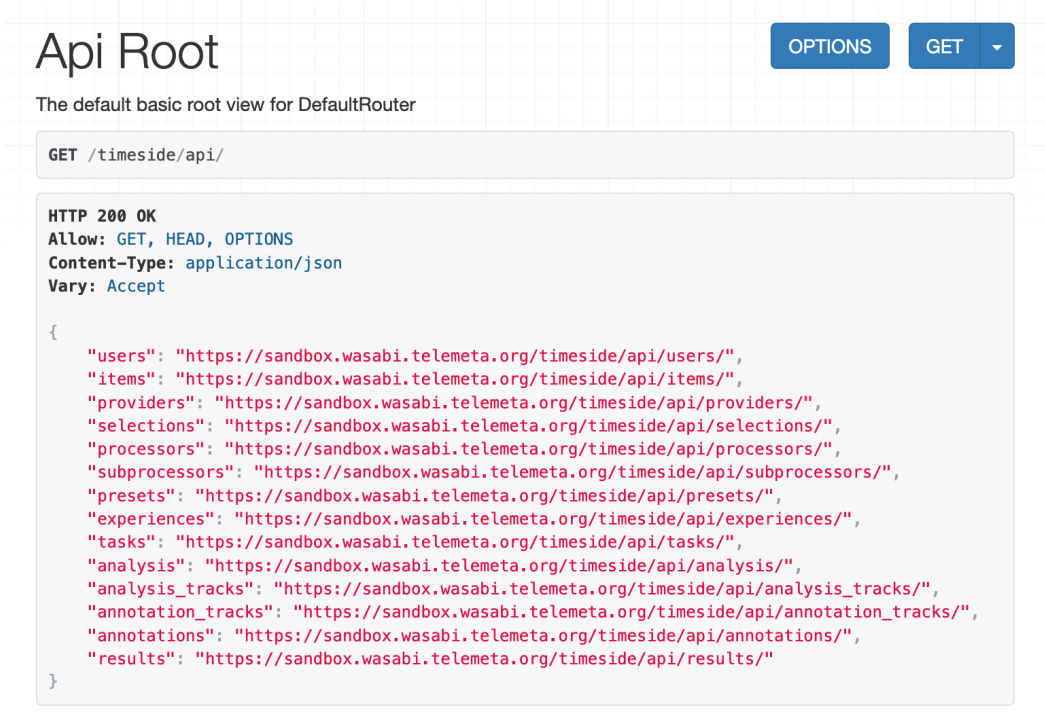
timeside.server

RESTful API built on TimeSide

☞ https://sandbox.wasabi.telemeta.org/timeside/api/

Models

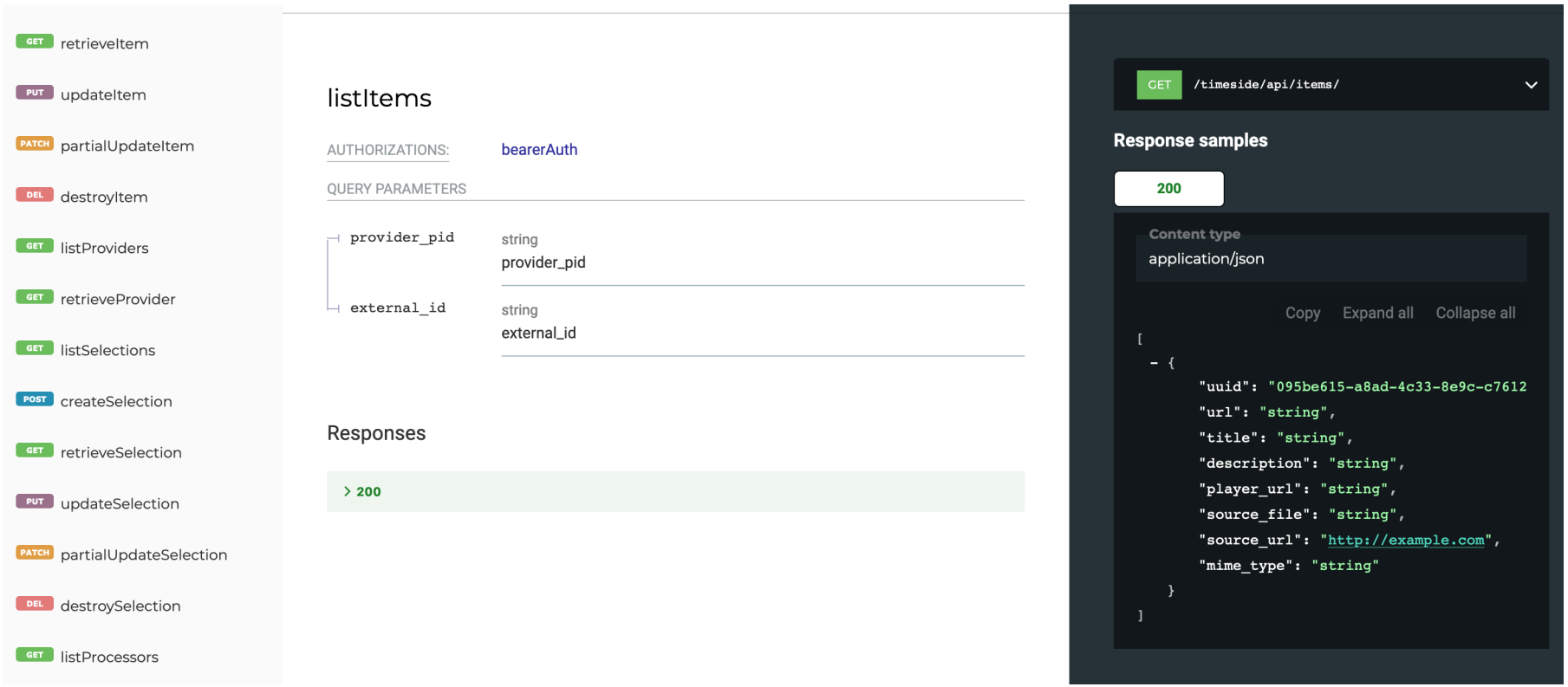
- `Item`: audio content and metadata (external id)
- `Provider`: provide audio from a given plateform
- `Selection`: list of items (corpus)
- `Processor`: plugins with version and default parameters
- `Preset`: processor and a set of parameters
- `Experience`: list of presets forming a pipe (reproductible)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



timeside.server

RESTful API documentation

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

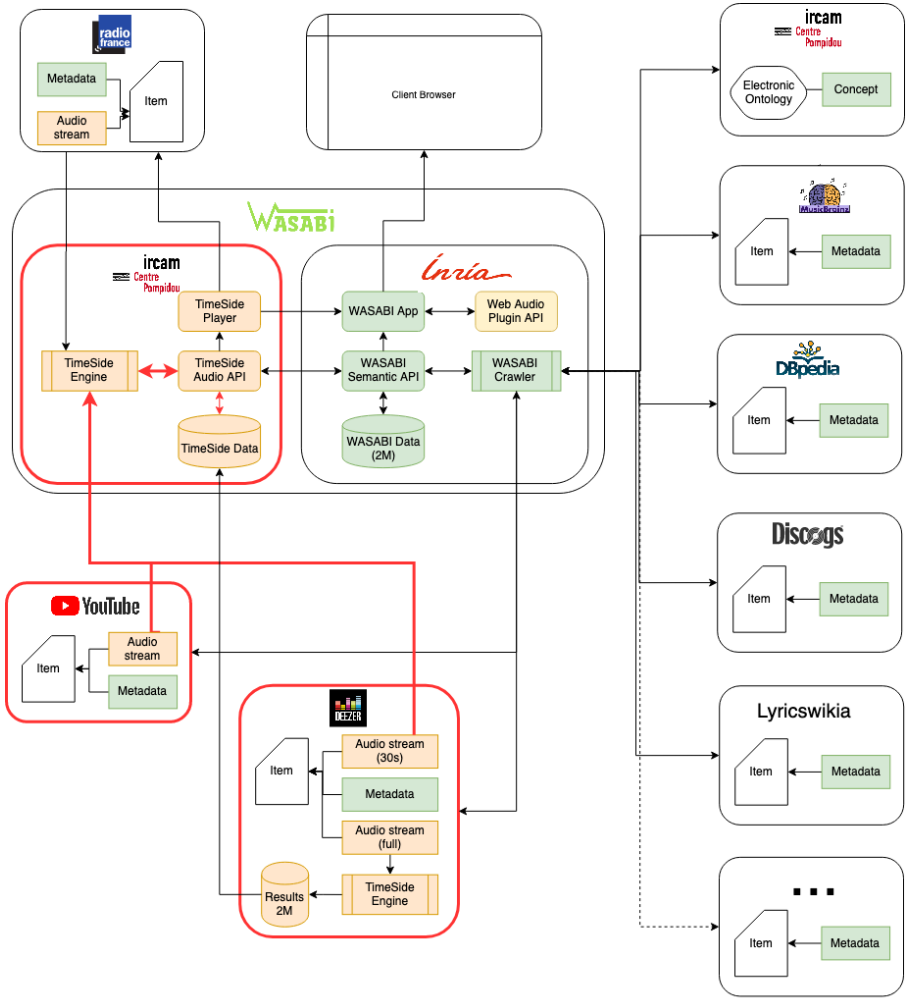


timeside.server

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



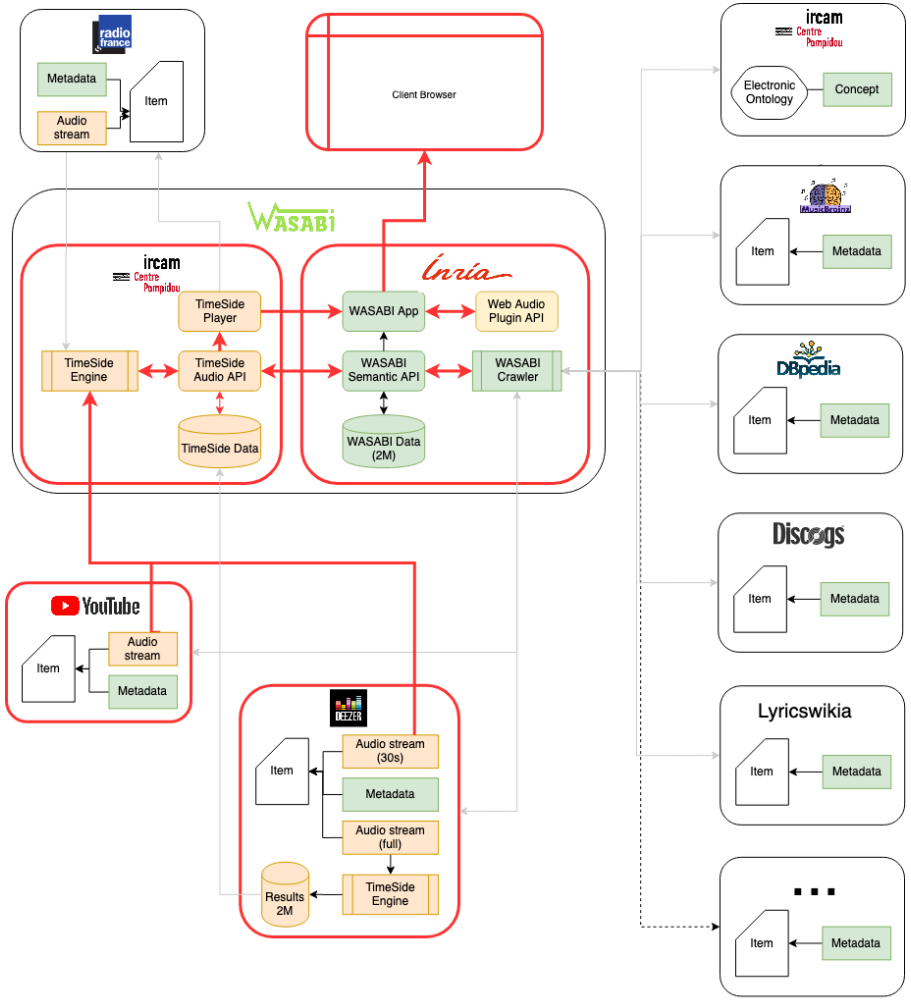
timeside.server

Workflow examples in WASABI Project

With server

POC of a webservice

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



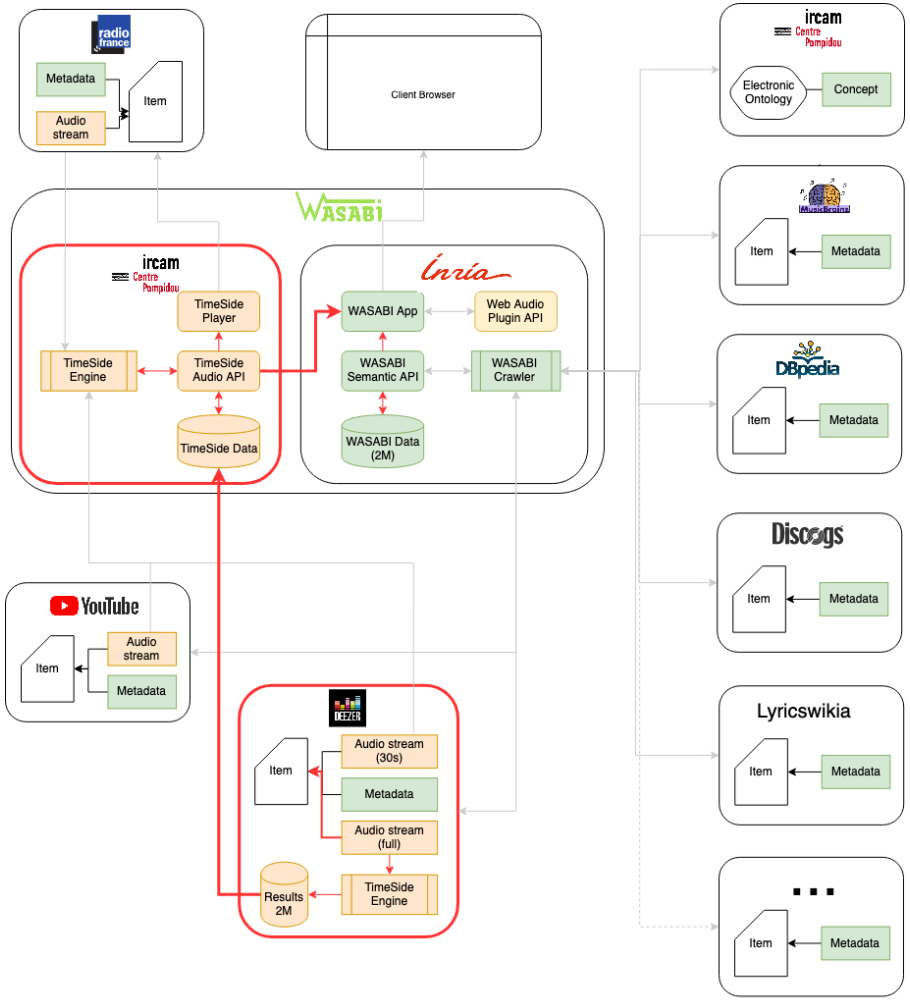
timeside.server

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



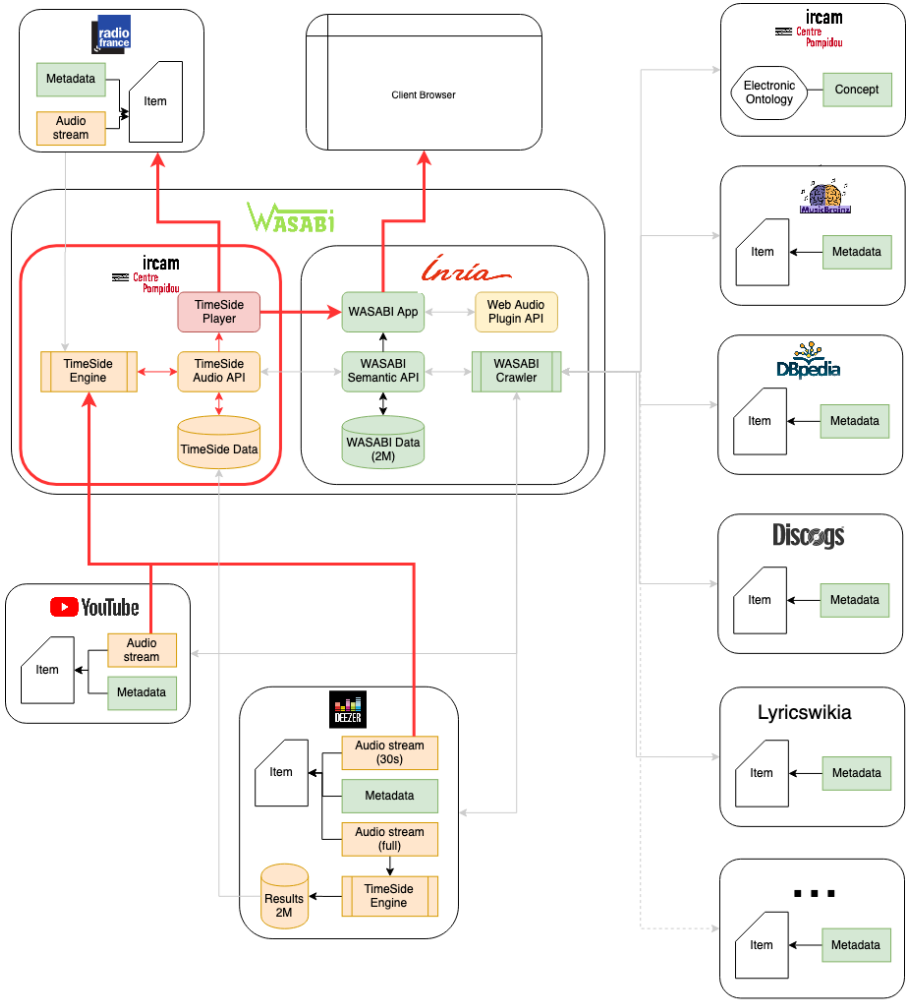
timeside.server

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

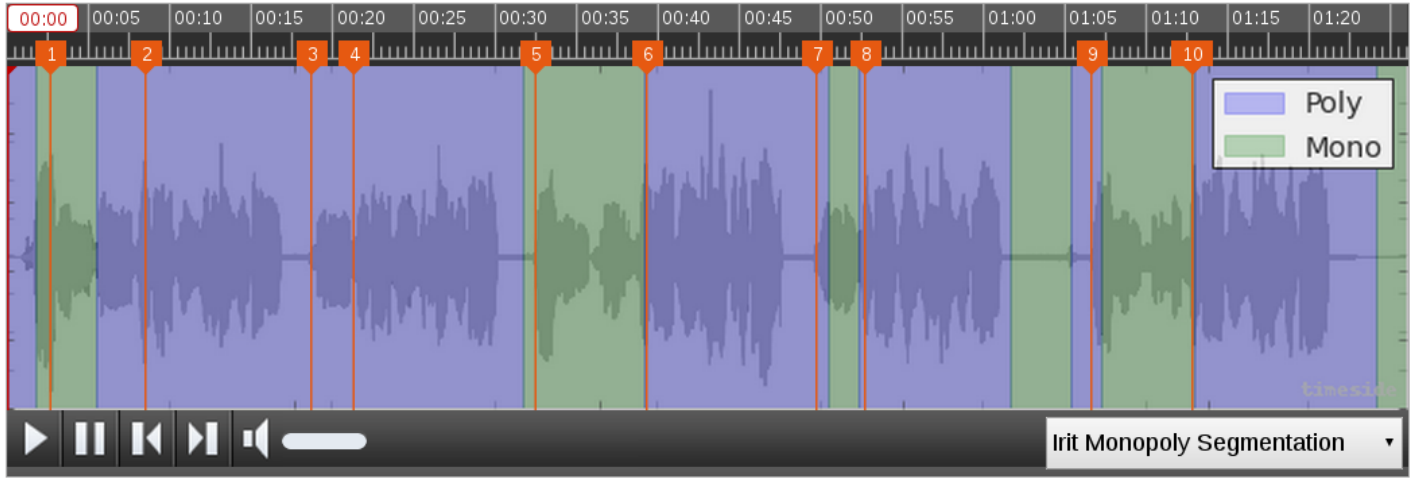


timeside.player

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

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

- SDK: https://github.com/Ircam-Web/timeside-sdk-js
- Node: https://github.com/Ircam-Web/timeside-scripts

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