Music360

A 360 DEGREES PERSPECTIVE ON THE VALUE OF MUSIC

Deliverable 2.2

A distributed architecture for music data collection, representation, and distribution – version 1

Disclaimer
The Music360 project has received funding from the European Union's Horizon Europe research and innovation action under grant agreement number 101094872.

Version history

Ver.	Date	Comments/Changes	Author/Reviewer
0.1	05/10/2023	Initial version	Giovanni Giachetti
1.0	30/10/2023	Updated according to reviewers comments	Giovanni Giachetti
1.1	08/01/2024	Integrated various contributions	Jaap Gordijn
1.2	22/01/2023	Feedback of all partner processed	Jaap Gordijn
1.3	22/01/2023	Feedback of all partner processed	Jaap Gordijn

Project Acronym	Music360	
Project Title	360 DEGREES PERSPECTIVE ON THE VALUE OF MUSIC	
Project Number	101094872	
Instrument	Research and Innovation Action (RIA)	
Topic	HORIZON-CL2-2022-HERITAGE-01-05	
Project Start Date	01/03/2023	
Project Duration	36 months	
Work Package	WP2 - Standardized, trusted and unified collection of music metadata	
Task	T2.2a Designing and implementing a platform for data collection and representation	
Deliverable	D2.2. A distributed architecture for music data collection, representation, and distribution – version 1	
Due Date	31/10/2022	
Submission Date	12/10/2022	
Dissemination Level[1]	Public	
Deliverable Responsible	UPV	
Version	1.0	
Status	In Progress	
Author(s)	Giovanni Giachetti	UPV
	Jaap Gordijn	VU
	Oscar Pastor	UPV
	Roel Wieringa	TVE
	Ed Green	VU
	Yulu Wang	VU
	Gonçal Calvo	BMAT
	Jacobus Koen	BMAT
Reviewer(s)	Sander Teekens	SENA
	Bruno Gaminha	GDA
	Lisa NíChoisdealbha	IMRO

[1] PU= Public, CO=Confidential, only for members of the consortium (including the Commission Services), CL=Classified, as referred to in Commission Decision 2001/844/EC

Disclaimer/Acknowledgement:

“The project Music360 has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101094872.

The opinions expressed in this document reflect only the author’s view and in no way reflect the European Commission’s opinions. The European Commission is not responsible for any use that may be made of the information it contains.”

Contents

Table of figures 6
1 Introduction 7
2 Requirements 9
2.1 Multi-party 9
2.2 Data owner remains in control with respect to their own data 9
2.3 Authentication and authorization 10
2.4 Data isolation 10
2.5 Open to future data providers and users 11
2.6 Scalable 11
2.7 Summary 11
3 Overall architecture 13
3.1 The Music360 architecture 13
3.2 External services 14
3.3 Internal services 14
3.4 Message bus 15
4 Identifiers 16
4.1 Recordings and works 16
4.2 Neighbouring- and author right holders 16
4.3 Other actors 16
5 Content-based routing and aggregation 18
5.1 Identifier-based routing 18
5.2 Aggregation-based routing 18
6 Front ends 20
6.1 Dashboards 20
6.1.1 Dashboard Views 24
6.1.2 Authentication and User Management 26
6.2 Ecosystem modelling 27
7 Data providers 31
7.1 The distributed Music360 database 31
7.1.1 Characterization of the data 31
7.1.2 Music360 Ontology Alignment 31
7.1.3 Partitioned and replicated databases 32

7.2 The data provider database 33
7.3 Aggregation services 35
8 Security, authentication & authorization 36
8.1 Multi-source authentication 36
8.2 Access control 37
8.3 The Music360 user database 37
9 Implementation 39
9.1 Authentication and access control 39
9.1.1 Overview 39
9.1.2 Technology 39
9.2 Router 40
9.2.1 Overview 40
9.2.2 Technology 40
9.3 Music360 database 40
9.3.1 Overview 40
9.3.2 Technology 40
9.3.3 Data model for living labs version 1 40
9.3.4 Data model for the Music360 ontology 42
9.3.5 Creating the databases 44
10 Conclusion 45
11 Appendix A Music360 ontology implemented in the relation model 46
12 Appendix B Music datamodel phase 1 implemented in the relational model 69

Table of figures

Figure 1 Music360 high level components and data flow 7
Figure 2 Music360 architecture 13
Figure 3 Router model 18
Figure 4 Examples of front end elements used in other related music analytics platforms ..24
Figure 5 Azure Active Directory B2C integration 27
Figure 6 Overview of the EcoSphere architecture 28
Figure 7 EcoSphere login screen 29
Figure 8 EcoSphere dashboard 29
Figure 9 EcoSpere graphical editor 30
Figure 10 Architecture of the data provider 32
Figure 11 Architecture data provider 34
Figure 12 Architecture aggregation services 35
Figure 13 User model 38
Figure 14 Music360 metamodel - phase 1 41
Figure 15 Music360 ontology, cf Deliverable D2.1 43
Figure 16 Relational Database Model for the Music360 ontology 46

1 Introduction

This deliverable outlines the architecture, and its implementation, of the Music360 platform – version 1. It will be the foundation for the Living Labs. After evaluation of the first phase of the Living Labs, a second version of the Music360 platform will be designed and implemented.

To work on the architecture of the technical solution, we have previously defined all input data source types and output interfaces, and the high-level components that will process and manage such data, and we have represented them in a diagram that contains the data flow of the system:

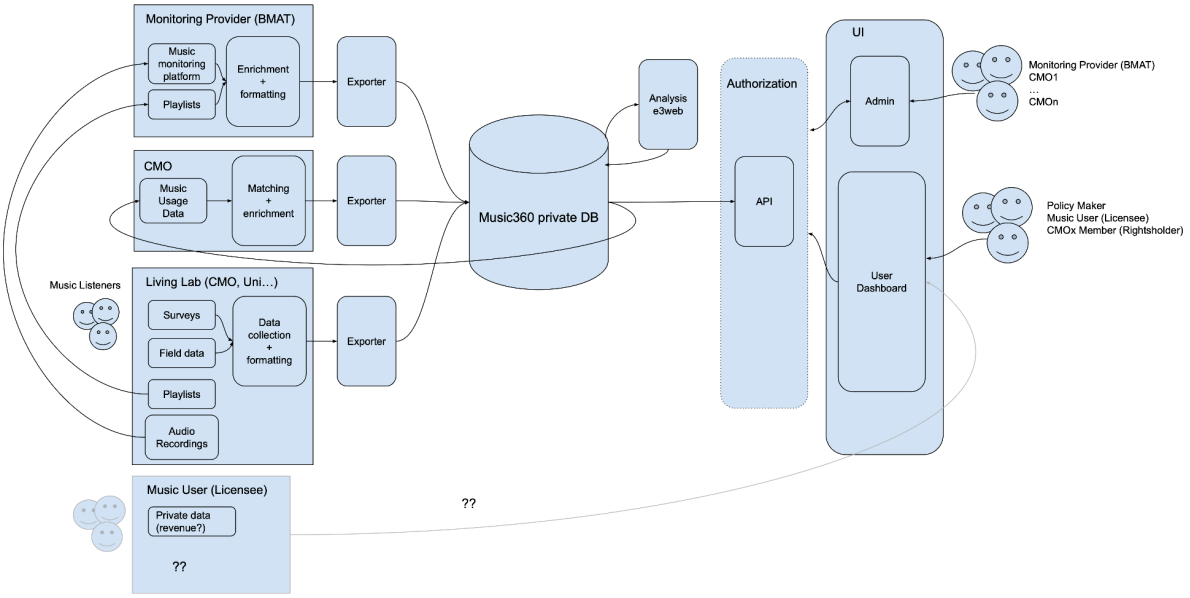


Figure 1 Music360 high level components and data flow

On the left side of the diagram, we have the modules that collect and provide data to the Music360 platform, such as music usage data provided by music technology companies like BMAT, enriched royalty distribution information provided by CMOs, information coming from the different Living Labs collected through surveys and field measurements, and optionally data provided directly by music users like venues. This information is made available to the platform using a common ontology and database schema and can then be analysed and served through an API to applications like dashboards or external services. The data is protected by an authentication and authorization layer to make sure that it is only accessed with the correct permissions, set by each data owner, depending on the final user role and group membership.

This deliverable is structured as follows. First, section 2 presents the requirements. As the stakeholder requirements are already discussed in D6.1 “Stakeholder needs for understanding the value of music - version 1”, this section will focus mainly on the non-functional requirements of the platform. The overall architecture of the platform is introduced in section 3. An important aspect of the data oriented Music360 platform is how to identify objects (e.g., recordings, works, right holders, venues) stored. This is discussed in section 4. The platform is largely

distributed, meaning that objects can be partitioned and replicated over several different data providers. This is primarily to allow data owners to keep in control of their own data, but this requires that they trust the data provider. For this purpose, we need to route data requests from the users to the data providers and back, which is explained in section 5. Section 6 briefly introduces the user-facing frontends of the platform. The backends, consisting of data providers, are discussed in section 7. Security is an important topic, as the platform stores privacy sensitive information (see section 8). Finally, implementation of the platform is outlined in section 9. Section 10 wraps up.

2 Requirements

This section presents several requirements. Stakeholders requirements are already discussed in D6.1 “Stakeholder needs for understanding the value of music - version 1”. The following paragraphs therefore mostly focus on the non-functional requirements that are relevant for the architecture of the Music360 platform.

2.1 Multi-party

The Music360 platform should support multiple parties of different stakeholder types:

- Creatives: Various kinds of music performers (featuring, session, studio, etc.), producers, publishers, lyricists and songwriters.
- Right societies: Right societies in terms of the right they collect for (author right and neighbouring right in Music360), their mandate (often associated with a region), their customers (called venues afterwards), and the (type of) right holders represented.
- Venues: Shops, restaurants, bars, factories, etc., effectively everyone who is using music to create additional value.
- Policy makers: On the world-wide-, EU-, national and local level, policy influencers, unions, etc.
- Commercial entities, e.g. streaming parties, such as Spotify and radio & television stations. They are not considered venues but may profit from the platform otherwise, e.g. by enriching their meta data, or delivering meta data as part of their contractual agreement. Note that commercial entities may also be required to pay license fees to CMOs, but are not considered for that matter in the Music360 project. The focus of the project is on background music, and the use of music in therapy.

This very broad range of stakeholders complicates things in many ways, resulting in the concerns discussed below.

An important topic, not discussed in this deliverable, is the governance of the Music360 platform. However, for a multi-party platform as Music360, this is highly relevant because there is no single authoritive decision making entity. Therefore, governance will be one of the topics dealt with in Deliverable 5.1 “Sustainable business model of Music360”. One of the possibilities we investigate is whether SCAPR and CISAC, which are umbrella organisations for neighbouring and author rights respectively, are interested in playing a key role in the governance. Because we pay attention to the governance concern in a separate deliverable, this deliverable has mainly a technical focus.

2.2 Data owner remains in control with respect to their own data

We expect that one of the critical success factors for the Music360 platform is that all parties, specifically those who own data, remain in control with respect to their data. Most parties have data that is to a certain extent private: creative entities are worried about data on their earnings and repertoire, CMOs have large clustered data sets about repertoire, line-ups, and usage of music, which would be very interesting for all kinds of machine learning applications of commercial entities, and venues have data concerning the effect of music on customer behaviour and ultimately sales, which therefore is competitive sensitive data. All this data should be protected and controlled by the respective data owner. Most data usage concerns the capability to read data. Creating, updating and deleting data is done by the data providers

on behalf of the data owners themselves, so they are in control by definition. We distinguish the following scenarios with respect to data accessibility, specifically reading data:

- Data is available to a user if the user has sufficient access rights. Data is stored in databases which are under control of the data owner (e.g., physically hosted at the premises of the data provider trusted to the owner, or in a data centre, still under control of the data owner). Access to data should be restricted at least at the instance level (e.g., a recording, work, right holder), and preferably at the property level (earnings of the recording, use of the recording). Access control is dealt with in detail in Deliverable D2.3 “Secure and trusted sharing of music data – version 1”.
- Data owners may delegate access control management to others. An example of this scenario is a creative entity which allows their representing CMO to make its data available under certain conditions. Often this data is clustered and enriched with additional metadata. In this scenario, the CMO is responsible for data storage and access control on behalf of the creative entity.
- Data owners can make data available in an inaccessible, containerized way (that is: someone who tries to access the data cannot read the data), but with a set of allowed predefined operations users can do on the data (e.g. arithmetic operations). The result of the operations can be inspectable or not, depending on the use case. It allows to make data available to untrusted parties, without disclosing the data, but allowing them to do some controlled operations on the data. This scenario is further developed in Deliverable 2.6 “Secure and trusted sharing of music data – version 2”.

2.3 Authentication and authorization

Authentication is about verifying whether someone is who s/he claims to be. Authorization concerns granting/preventing an authenticated user to do operations (e.g., create, read, update, delete) on a resource, for example data or a service.

In Music360, we assume that the data owner is in control of, responsible for, and executing the access control (hence authorization). However, for authentication, Music360 data owners rely on the identity service of Music360, or a delegated party operating an identity service. This means that there is a trust relationship between Music360 data owners and (external) identity providers. In other words, data providers assume that authentication providers correctly, and reliably identify users. Consequently, misbehaving identity providers are not considered as an attack vector.

In the second phase of the architecture, we may consider developing decentralized identity providers, e.g. each data provider then many become an identity provider.

2.4 Data isolation

Related to data ownership, data of different stakeholders should be properly isolated. This means that parties (specifically data providers), cannot see the data of the other Music360 participants. Depending on the data storage solution chosen, this requires specific measures. In others words, all data will not be stored in one database, where all the data owners have full access to.

2.5 Open to future data providers and users

Currently, the Music360 platform is restricted to the EU Music360 project partners, and external parties to which the project partners want to make the platform available (e.g., the participants in the Living Labs - WP6).

However, Music360 has the explicit ambition to remain in operation after the EU funding ends. For this reason, we consider Deliverable 5.1 “Sustainable business model of Music360” as very important.

Already some (large) CMOs have expressed their interest in the Music360 platform, and in its ambition to understand the value better. This also holds for one of the sister projects, OpenMuse, with which we agreed to harmonise quantitative studies on the value of music.

Because of this interest, and also because of the nature of the international music industry, the Music360 platform should be as open as possible. From a technical perspective, this implies the use of (de-facto) open standards:

- DDEX where applicable.
- Relevant ISO/IETF/de-facto standards on interoperability (e.g. REST, WS*, SMTP, HTTP) and security.
- Identifiers such IPN, IPI, ISWC, ISRC and ISNI.

2.6 Scalable

The world-wide music industry is large, both in terms of the number of stakeholders, and the amount of data it generates. Therefore, the Music360 platform should be scalable, with respect to the number of users and the amount of data.

The vast majority of data operations are READ operations, meaning that data enters the platform once, and is queried many times. This allows for replication and/or partitioning of the very large data set, hence distributing data over a large number of decentralised databases and hardware.

Replication of data refers to the idea that the same data is available at many places. Partitioning of data means that different data is stored at different places. A special case of data partitioning is if data of one object (e.g,. a recording) is fragmented over multiple data providers. This is one way to ensure that a data owner remains in control of its own data via a trusted data provider in terms who is allowed to access that data, namely by owning the relevant partition of the database.

Both replication and partitioning distribute data over multiple software- and hardware resources, which is a way to achieve scalability in terms of the amount of data stored. By replication of the same data over multiple machines, it is also possible to reach scalability in terms of the number of users. However, it is then required that replicated data is preferably not updated, which is a characteristic of the Music360 data: create data once and rarely update or delete. If data is updated, this happens only by a few data, so that changes can quickly be propagated to the replica’s.

2.7 Summary

Concern	MoSCoW prioritisation	Description
C1	MUST	Multi party

C2	MUST	Access control at instance level
C3	SHOULD	Access control at property level
C4	MUST	Access control delegation
C5	MUST	Data sharing to untrusted environments
C6	MUST	Trusted identity providers
C7	MUST	Platform parties can not access each other data without having the proper access controls
C8	MUST	Technical openness to other parties, support of open (de-facto) standards
C9	MUST	Scalable in terms of users
C10	MUST	Scalable in terms of create/read on data
C11	COULD	Scalable in terms of update/delete on data

Table 1. Music360 MoSCoW feature prioritisation

3 Overall architecture

3.1 The Music360 architecture

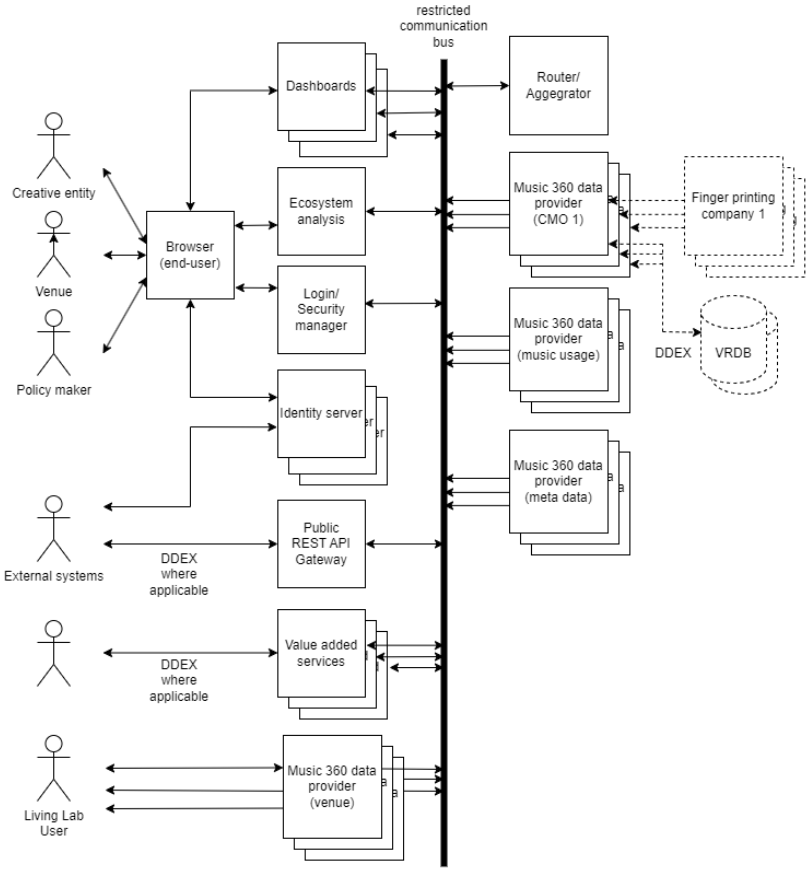


Figure 2 Music360 architecture

The overall architecture of Music360 is presented by Figure 2. Boxes visualise software- and/or hardware components; arrows represent communication between these components. The platform is mainly a data presentation service; there are a few exceptions, but in most cases, users only query data. This queried data is related to works, recordings, their use, right holders, and right users (dashboard), or relates to the EU music system as a whole (ecosystem).

The architecture is decentralised and implemented in a distributed way, which allows for the situation that data remains under control of the data provider, for instance a CMO. Data (e.g., about a recording or musical work) is replicated (provided that the replica remains under control of the same data provider), and more often partitioned over various data providers, each offering their own perspective on the object at hand (e.g., a recording). In the current music ecosystem, this is the case too. We are convinced that a decentralised setup is a prerequisite for acceptance by most actors, CMOs. With this architecture, it can be guaranteed that owners of data remain in control. To facilitate small data owners who do not have the facilities to implement and host their own data provider, there is also the possibility host the data by a third party, e.g. the Music360 platform provider, or a CMO. This however introduces an additional trust relationship: the data owner must trust the platform provider. Overcoming this limitation will be studied by Deliverable D2.6 “Secure and trusted sharing of music data – version 2”.

There are different kind of data providers: CMOs, music usage data providers (e.g., BMAT), providers of miscellaneous metadata (e.g., OpenMuse) or venues (users of music) that provide data related to the impact of music usage (e.g., as a result of questionnaires held or measurements of the effect of music on the revenue of the venue).

Note that CMOs (and other parties) often use the services of music fingerprinting companies (e.g., BMAT) to get music usage data that help them in the royalties distribution processes. Fingerprinting companies are represented with dashed lines because they are already part of the current state of the art, and they are used by many CMOs. They are crucial for the internal business processes of CMOs, which we do not change. Data concerning the use of music may come available to the Music360 platform via the CMOs. However, we choose to make data about the usage of music, as provided by music fingerprinting services, directly available to the platform too. The reason for this is that CMOs often aggregate usage data, so sometimes fine-grained data about the usage of music disappears.

3.2 External services

Several external services exist.

There can be many different identity services. These services play an important role in the authentication of users. There will be at least one Music360 identity service, but there may be other, e.g. coming from data providers or CMOs. Identity services closely relate to the login/security manager. In essence, this service controls the access to all the data in the ecosystem.

The dashboards provide data regarding the use of music, and its value, in a detailed or aggregated way. There can be many different dashboards, such as per country or music type. Because presentation of data (via the dashboards) is decoupled from the data itself, dashboard focussing on a specific topic can be developed. Once an entity is part of the Music360 ecosystem, it can develop such dashboards.

There will also be a service to analyse the music ecosystem as a whole (ecosystem analysis). We expect this is relevant for policy makers.

Finally, data will be exposed via an API service. This allows parties to query the raw or aggregated data for further processing. We expect that companies such as Spotify will be interested in this service, to enrich their meta data. Again, because the data is query-only, this API gateway can be easily replicated to achieve scalability.

3.3 Internal services

There are several internal services.

First, there is the data routing/aggregation service. If a user wants to see the available data concerning, for example, a recording, the router/aggregator service consults all the data providers where data about this recording is stored and presents the result to the user. Note that data, about a recording or any other entity can be stored at various data providers.

Second, data providers expose data about music users, right holders, recordings and musical works to other services in the platform to the dashboards, ecosystem analysis service, value-added services, and API gateways. Most data providers rely on their internal databases to provide data to the Music360 platform. In case of the Living Lab experiments, data is obtained

using questionnaires and other data collection methods and will be available in an accessible way to the users of the platform.

Third, venues provide additional data to the platform, especially regarding the value effects of music played at the venue. Also, venues can access the platform for data, which can be used for further analysis by these venues. Venues provide the sales data they want to share, and get the data that they need. Data shared may be indexed, e.g. not disclose the sales numbers themselves. We expect this is crucial for the acceptance of the platform by venues.

Fourth, there are services to identify played recordings, a.k.a. as fingerprinting services.

3.4 Message bus

All internal services should be able to communicate with each other in a secure way. To this end, we will employ a virtual Music360 computer network, implemented as a Virtual Private Network (VPN) based on Wireguard (or similar technology), which is a high-performance virtual network technology over the Internet. This restricts access to the Music360 network to the eligible parties. The message bus will support (a)synchronous request/response type REST services, and an event driven topic-driven publish-subscribe facility.

4 Identifiers

Users will query for specific objects, such as right holders, right users, recordings and works, their use and impact. These objects need to be properly identified, and more importantly, related to each other. For identification, we assume that the available industry standards work.

4.1 Recordings and works

Recordings and works are identified by the International Standard Recording Code (ISRC) and International Standard Musical Work Code (ISWC) respectively. In the Music360 project, we assume that the ISRC and ISWC codes behave as proper identifiers, meaning that they uniquely identify recordings and works, and that a recording/work has precisely one ISRC/ISWC code. We are aware that this is not always the case (there are some flaws in the use of ISRC and ISWC codes in the industry), but solving these issues is outside the scope the Music360 project.

Relating recordings and works is not trivial. In the Music360 project, this will be addressed by the author right CMOs and the fingerprint companies (BMAT in the consortium).

4.2 Neighbouring- and author right holders

Neighbouring right holders (artists and producers) are identified by the International Performer Number (IPN). Authors are identified by the Interested Party Identifier (IPI). For all these identifiers, we assume that they identify an actor uniquely, and that a right holder has at most one IPI/IPN identifier (but may have one of each). Currently, there does not exist industry standard identifiers for producers. This will solved in the project by a Music360 identifier but it is up to the industry to propose a solution for the long term./

What however is possible, and actually happens quite frequently, is that an actor has multiple roles (e.g. artist and author). Unfortunately, the current state-of-the-art is that there is no explicit relationship between the IPN and IPI identifiers in the available databases. However, the ambition of Music360 is to show all data known for a particular actor, such as artist-related data and author-related data. To solve this, we implement the following:

The first time after successful authentication, and if the user claims to be a creative entity, that user will be asked to login at one neighbouring right society and/or one author right society to obtain his/her IPN and IPI respectively. To login, the user uses his/her already existing credentials for the relevant CMOs. If a person is both an artist/producer (neighbouring right) and an author (author right), this allows the Music360 to relate and store these roles, which is missing in the current state of the art of the metadata of music. This is one of the few cases where the Music360 database is updated, based on information provided by the user.

4.3 Other actors

There are a number of other actors relevant, namely agents of venues using music, who usually are employees of an organisation, and users of the external API.

There will be a facility to register an organisation. The identifier for an organisation depends on the country (there is no usable EU organisation identifier for this purpose yet). Therefore, we will use the national organisation code (e.g., assigned by the chamber of commerce), scoped with the country code. Organisations will have an administrative user, who can add and remove ordinary users for the organisation at hand. These users have an internal

Music360 identifier. Onboarding of the other users will be handled by the CMOs, who already have a relationship with most actors.

Organisations can create an access token, which can be used to authenticate for the external API.

5 Content-based routing and aggregation

The content router takes query-based requests from the front ends (portals and ecosystem analysis), and routes them to the relevant data sources. The router also receives responses from the data sources and streams the results to the front ends. The router addresses query-only functionality. Hence the router can perfectly be replicated to achieve high availability and high performance. As discussed below, there is some further intelligence in the router.

5.1 Identifier-based routing

In the case where queries refer to objects known to the Music360 platform (concretely: works identified by their ISWC, recordings identified by their ISRC, authors identified by their IPI, artists identified by their IPN, and venues by their company identifier), the router directs the query to appropriate data provider(s). It is important to understand that data about an object may be stored at multiple data providers. For example, metadata about a recording may be stored by a neighbouring right CMO, an author right CMO, and a party that provides information about the usage of the recording at hand.

To accomplish this, the router keeps a database of the different kinds of identifiers of Music360 objects, their type (e.g., ISRC, ISWC) plus a reference to the data provider(s) who keeps data about these objects. Data sources must register their objects at the router, once they are imported into the Music360 platform, so that efficient dispatching of queries can happen. Similarly, if an object is unloaded from the platform, the reference should be removed.

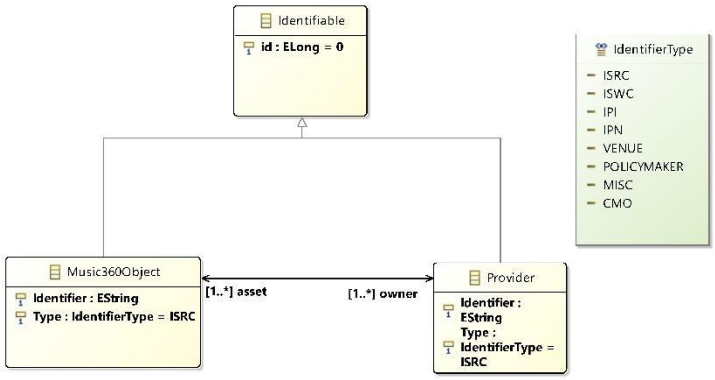


Figure 3 Router model

5.2 Aggregation-based routing

Music360 will support aggregation kind of queries. Providing a list of these queries will be a separate project task (mainly work done by the Living Labs in WP6), but non-exhaustive examples are:

1. Total economic effect of recording X.
2. Total minutes played of work Y.
3. Total minutes played of genre “popular”.

The first two queries refer to identifiable objects (namely recording X and work Y). The aggregation property aspect of the query is to calculate a number, namely the total economic effect and the total minutes played. The router will break down the query into subqueries,

namely one for each data provider who keeps information about the object. Each data provider will then execute the aggregation query and the results of each data provider will be aggregated by the router.

The third query does not refer to a particular identifiable object but aggregates over a certain property of an object, in this example the genre. The router will then fire the query to each data provider in the platform and will aggregate the results.

To facilitate aggregate queries, a predefined set of these queries will be established. Determining this set is largely dependent on the needs of the users, therefore finding these queries is part of WP6 (Living Labs).

In sum, the aggregation type of queries is handled at two levels:

1. The data source level: each data source must handle the pre-agreed aggregated queries.
2. The router level: in case an aggregate query spans multiple data sources, the router should dispatch the query to the relevant data providers, aggregate the responses, and stream the aggregated response to the front end.

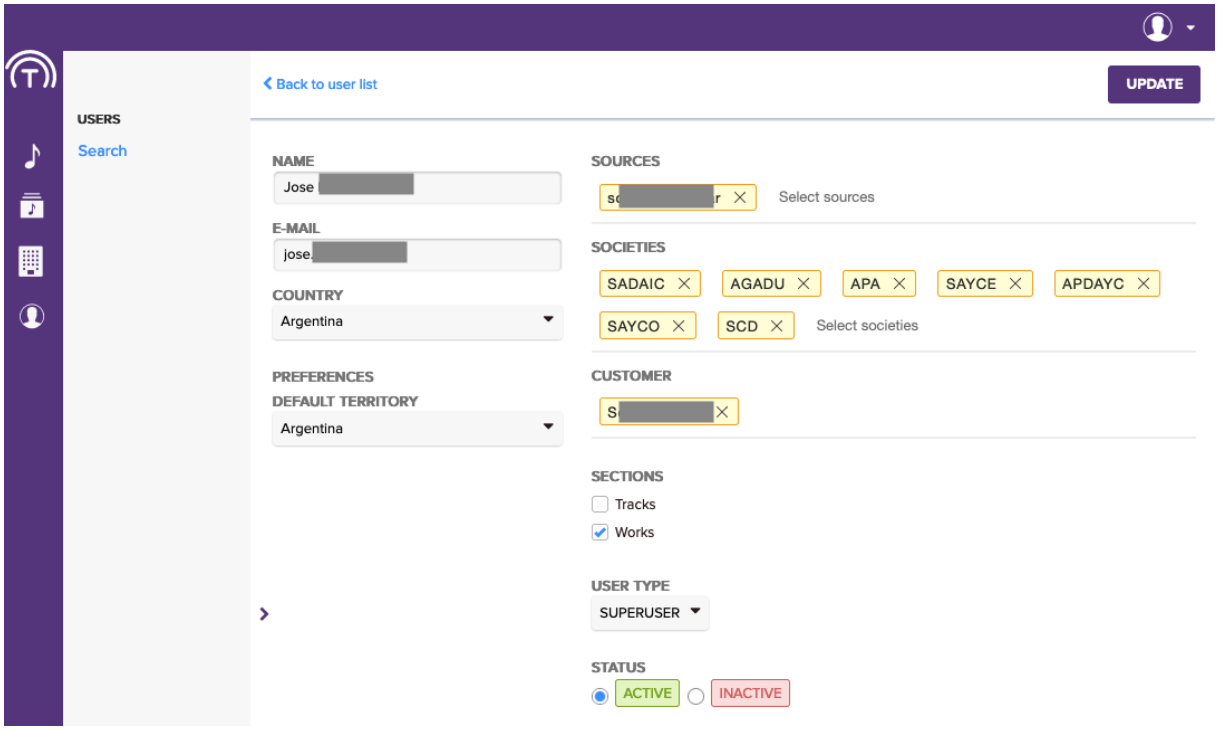
6 Front ends

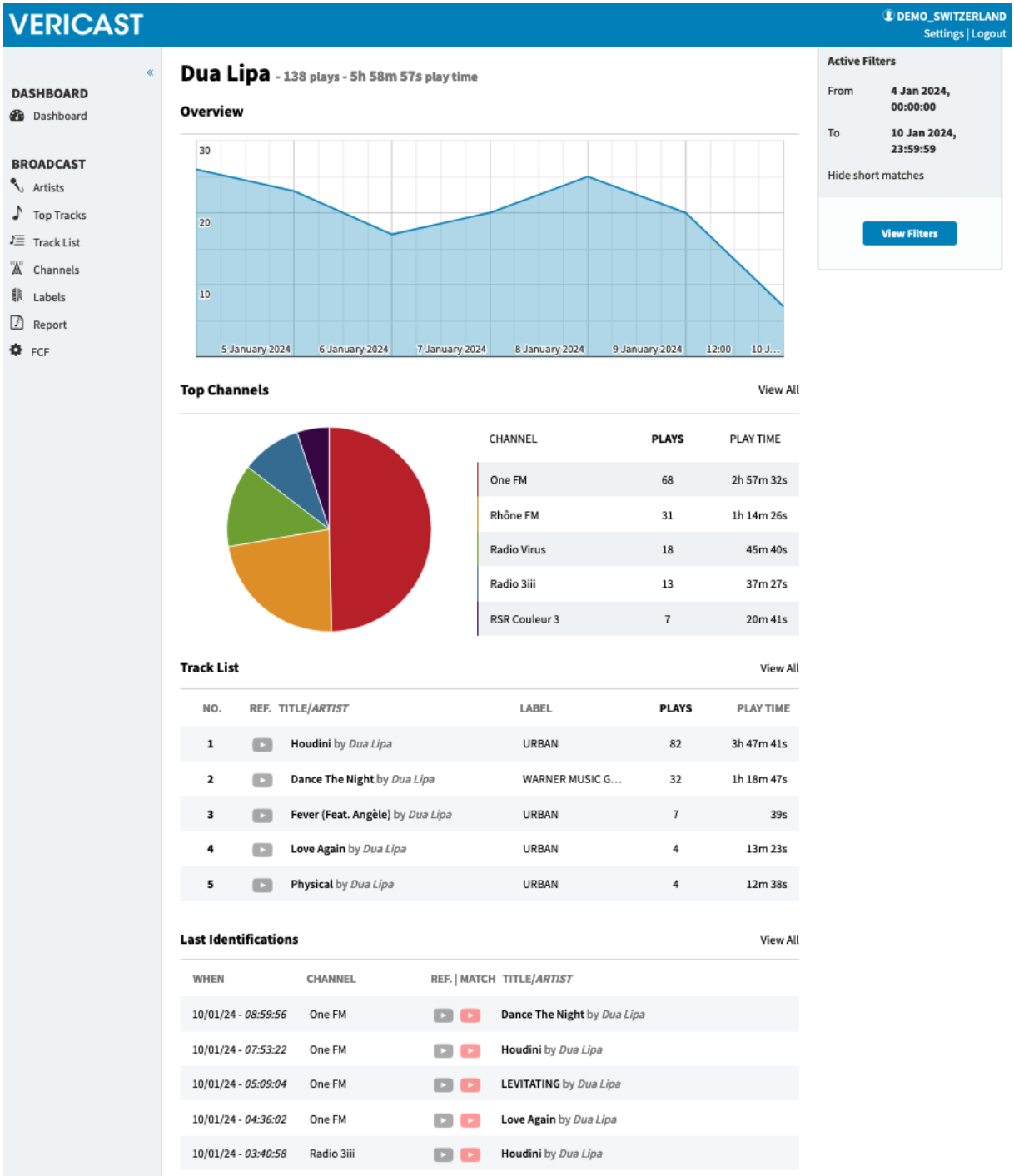
6.1 Dashboards

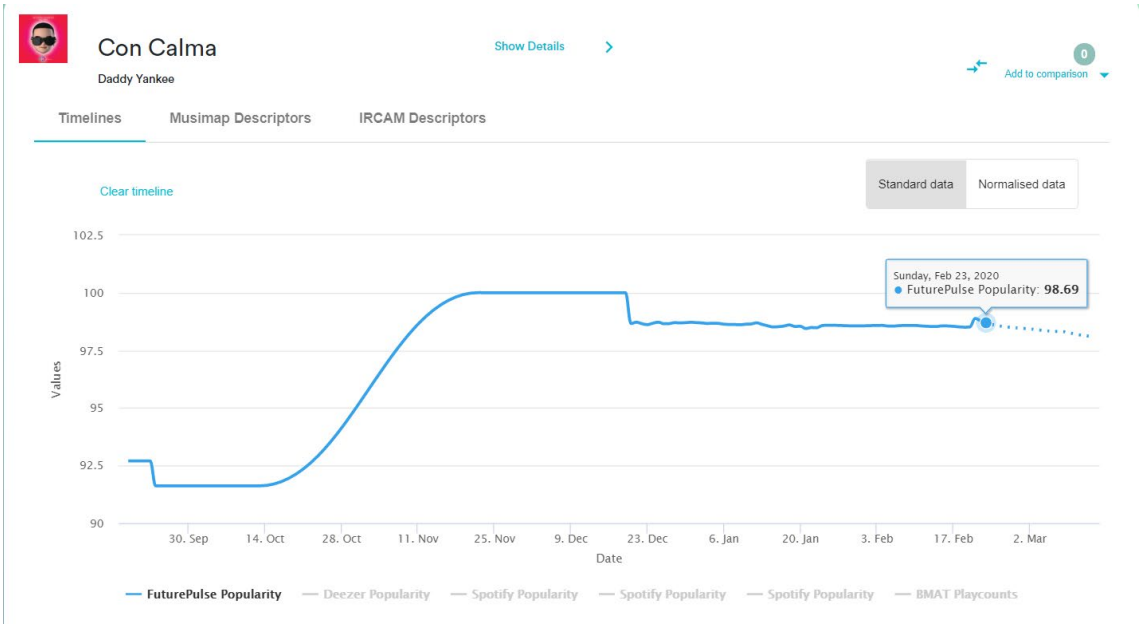
The public face of the project results are the user interfaces planned to be designed and developed in WP3 “Stakeholder-level reporting and analysis of the value of music”. Besides programmatic access to the data collected and generated by Music360, the platform will feature a web application that allows the user to query and explore it in a visual way, in the form of data dashboards. Specifically, WP3 encompasses the development of such interfaces, a reusable dashboard for stakeholder-level analysis of the collected data, which can be used and adapted in the Living Labs to the national situations, and later to the international Living Lab. This will be a front end for the stakeholder modelling and analysis tool and will also have a direct interface to the distributed database.

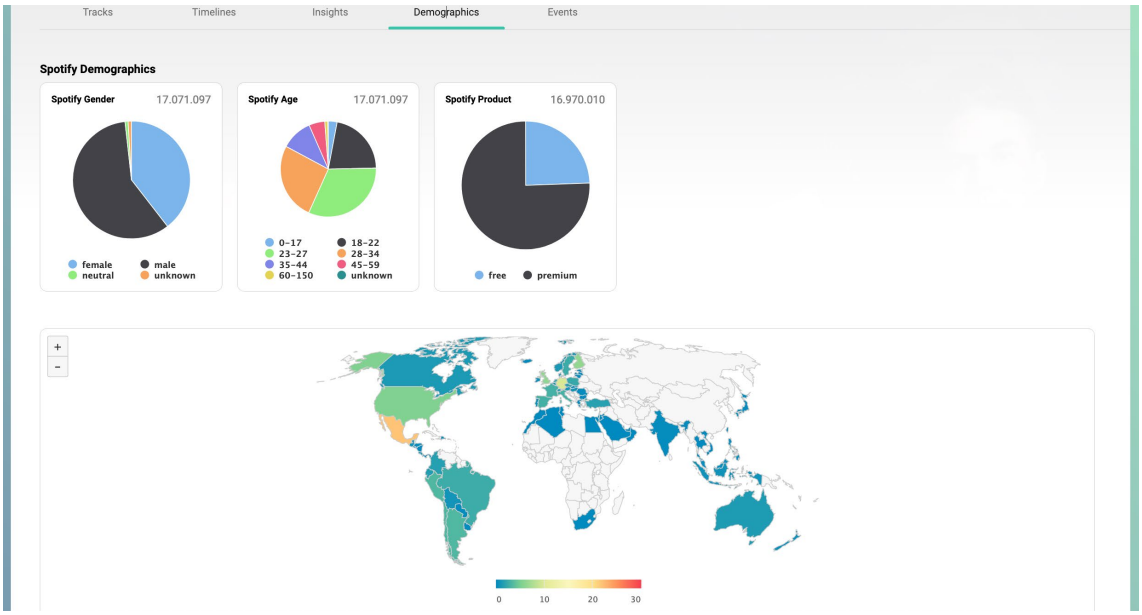
Consequently, all the work on the first iteration of the dashboard will be reported and detailed in the future deliverable D3.1 “Reusable dashboard to present and analyse the value of music - version 1”, describing a web front end for stakeholder-oriented modelling and value analysis. A second version of the dashboard, detailed in D3.2 “Reusable dashboards to present and analyse the value of music – version 2” will include the lessons learned in the Living Labs.

In the present section of this document, we introduce some elements that help understand the modules that will be part of the user interface in the overall platform architecture: the user dashboards and the authentication and user management. The figures below illustrate existing dashboards, developed prior to this project, for other products and initiatives led by members of the consortium. They depict several visualisation formats and user experiences that will be presented to end-users in order to validate the design of the Music360 Graphical User Interface (GUI). A similar design of some of the components might be used within this project, although a new dashboard, with a design and implementation based on these formats, will be developed in order to be compatible with the complete Music360 back end infrastructure and front end technologies used. The technology stack that we plan to use for the front end within the project is described in section 9.4 below.


Back to user list
UPDATE
USERS
Search
NAME
Jose
E-MAIL
jose
COUNTRY
Argentina
PREFERENCES
DEFAULT TERRITORY
Argentina
SOURCES
Select sources
SOCIETIES
SADAIC
AGADU
APA
SAYCE
APDAYC
SAYCO
SCD
Select societies
CUSTOMER
SECTIONS
Tracks
Works
USER TYPE
SUPERUSER
STATUS
ACTIVE
INACTIVE


VERICAST
DEMO_SWITZERLAND
Settings | Logout
DASHBOARD
Dashboard
BROADCAST
Artists
Top Tracks
Track List
Channels
Labels
Report
FCF
Dua Lipa - 138 plays - 5h 58m 57s play time
Overview
30
20
10
5 January 2024
6 January 2024
7 January 2024
8 January 2024
9 January 2024
12:00
Active Filters
From 4 Jan 2024, 00:00:00
To 10 Jan 2024, 23:59:59
Hide short matches
View Filters
Top Channels
View All
CHANNEL PLAYS PLAY TIME
One FM 68 2h 57m 32s
Rhône FM 31 1h 14m 26s
Radio Virus 18 45m 40s
Radio 3iii 13 37m 27s
RSR Couleur 3 7 20m 41s
Track List
View All
NO. REF. TITLE/ARTIST LABEL PLAYS PLAY TIME
1 Houdini by Dua Lipa URBAN 82 3h 47m 41s
2 Dance The Night by Dua Lipa WARNER MUSIC G... 32 1h 18m 47s
3 Fever (Feat. Angèle) by Dua Lipa URBAN 7 39s
4 Love Again by Dua Lipa URBAN 4 13m 23s
5 Physical by Dua Lipa URBAN 4 12m 38s
Last Identifications
View All
WHEN CHANNEL REF. | MATCH TITLE/ARTIST
10/01/24 - 08:59:56 One FM Dance The Night by Dua Lipa
10/01/24 - 07:53:22 One FM Houdini by Dua Lipa
10/01/24 - 05:09:04 One FM LEVITATING by Dua Lipa
10/01/24 - 04:36:02 One FM Love Again by Dua Lipa
10/01/24 - 03:40:58 Radio 3iii Houdini by Dua Lipa


Con Calma
Daddy Yankee
Show Details
Add to comparison
Timelines
Musimap Descriptors
IRCAM Descriptors
Clear timeline
Standard data
Normalised data
Sunday, Feb 23, 2020
FuturePulse Popularity: 98.69
Values
Date
FuturePulse Popularity
Deezer Popularity
Spotify Popularity
Spotify Popularity
Spotify Popularity
BMAT Playcounts


Tracks
Timelines
Insights
Demographics
Events
Spotify Demographics
Spotify Gender
17.071.097
female
male
neutral
unknown
Spotify Age
17.071.097
0-17
18-22
23-27
28-34
35-44
45-59
60-150
unknown
Spotify Product
16.970.010
free
premium
0
10
20
30

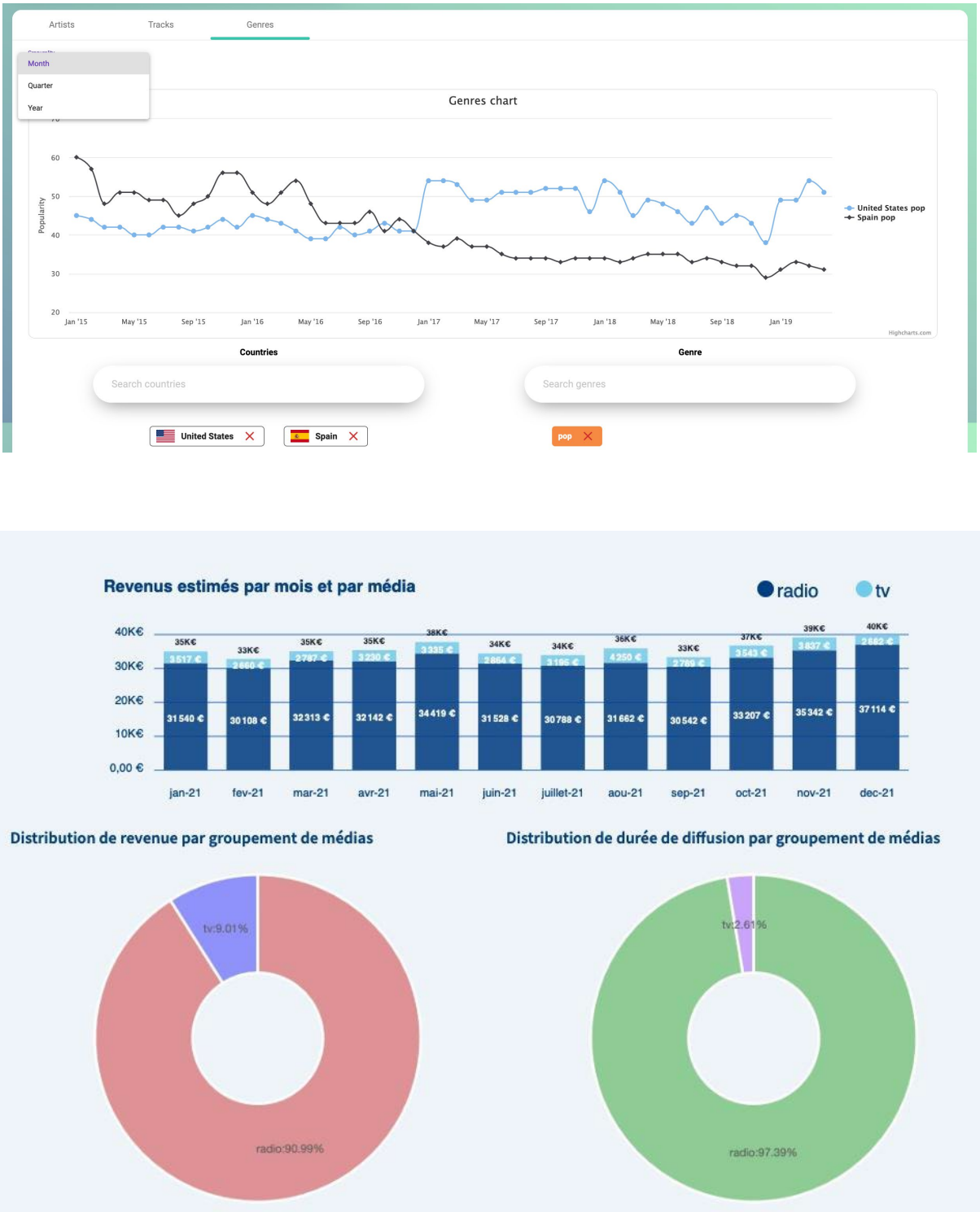


Figure 4 Examples of front end elements used in other related music analytics platforms

6.1.1 Dashboard Views

The main question that the user interface needs to address is how the Music360 platform will support the presentation and analysis of the value of music with software tools.

To answer that need, the Music360 platform will feature optimised dashboards for the stakeholders that will be made available through a secured web application. Specific

components will be added, and the resulting integrated platform will be tested with artificial data and with stakeholders, as a proof-of-concept. A generic view will be created, to provide an overview of the whole music ecosystem, but due to the complexity of the sector, a general view can be overwhelming, so several views will be created that will allow for clear visualisation of specific aspects of the music industry (specific Living Lab, use of music, monetary value of music, consumer sentiment analysis, ...).

Some information and details, such as aggregated music usage, will be made publicly available to all system users. Nevertheless, the system must be flexible enough to support different levels of personal data exposure depending on the business needs. We will use access control and homomorphic encryption to satisfy confidentiality requirements. Solutions based on access tokens will be made available to each Living Lab and stakeholder Admin to check all kinds of requests and ensure the correct use of the system. Also, an identifier will be assigned to every actor so as not to share personal data to all system users, for increased safety and privacy.

Therefore, the application will consist of two separate modules: an administration module – only accessible by administrative users–, and a data view or general application module – which will house all the data visualisation elements, restricted by the authorisation permissions of the user–. Here is a summary of the main features for each of these two modules of the Music360 application:

- Administration module:
  This module allows the platform administrators to manage users, roles and permissions in a clear, transparent, granular and secure way. There will be different user roles that will grant access to different levels of administration powers to manage users, including their organisation (e.g., a CMO, a Venue, a policy maker), their user role (e.g., admin, super user, plain user), and their permissions (e.g., permission to access to a certain data, granular or aggregated, on an asset-based policy). In summary, the different sections of the Administration module will allow:
    - User Management
      - Create, Read, Update, Delete users
      - Assign organisations to users
      - Assign roles to users
      - Assign permissions to users
    - Organisation Management
      - Create, Read, Update, Delete organisations
      - Assign permissions to organisations
    - Roles Management
      - Create, Read, Update, Delete roles
    - Permissions Management
      - Create, Read, Update, Delete roles

- Data View module:
  This module allows the platform users explore the data they have access to in an elegant, easy-to-understand and flexible way. The Data View visualisation tools will showcase the Music360 data streams with specific views that will be determined in the

UI design phase of WP3 to answer the needs of the different user roles, with the project Living Labs as a reference to get clear specifications and iterative improvements. In general, the dashboards will show data in tabular and chart format, with the ability to filter and sort by different criteria (e.g., time, region, music parameters), export it in different formats such as Excel or CSV, and compare different data sets.

The different views can be customised according to user types, based on roles and permissions, to make sure that it adapts to the needs of each stakeholder of the Music360 platform, with special attention to authorisation policies, to make sure that data is displayed following the access criteria defined by the data owner and administrator of the platform. In summary, the Data View module will allow the user to explore, always according to their permissions, either detailed reports of music usage data or aggregate data with enriched information that helps understand the value of music depending on each use case and scenario.

6.1.2 Authentication and User Management

The information accessible and displayed in the front end of the platform is made available by the back-end, but its accessibility is controlled by authentication and authorisation services, in order to restrict information concerning the value of music per stakeholder. There are cloud services that operate with standards-based authentication protocols including OpenID Connect, OAuth 2.0, and Security Assertion Markup Language (SAML) that can be used in the the implementation of such module, for instance the Microsoft's Azure Active Directory B2C authentication[2]. These types of services provide business-to-customer identity as a service, where the customers can use their preferred social, enterprise, or local account identities to get single sign-on access to applications and APIs. They offer interesting off-the-shelf functionalities that our platform needs and that will avoid having to develop them from scratch. Some of the main features are:

- B2C authentication
- Single sign on access with support for Social, enterprise, or local account identity providers
- Password recovery and management
- Create, Edit, Delete users
- Assign permissions to users and applications for access to specific content streams
- Assign roles to enable specific features (e.g., Admin, User Manager, Viewer).
- Logs of authentication actions (success and failures)

[2] https://learn.microsoft.com/en-us/azure/active-directory-b2c/overview

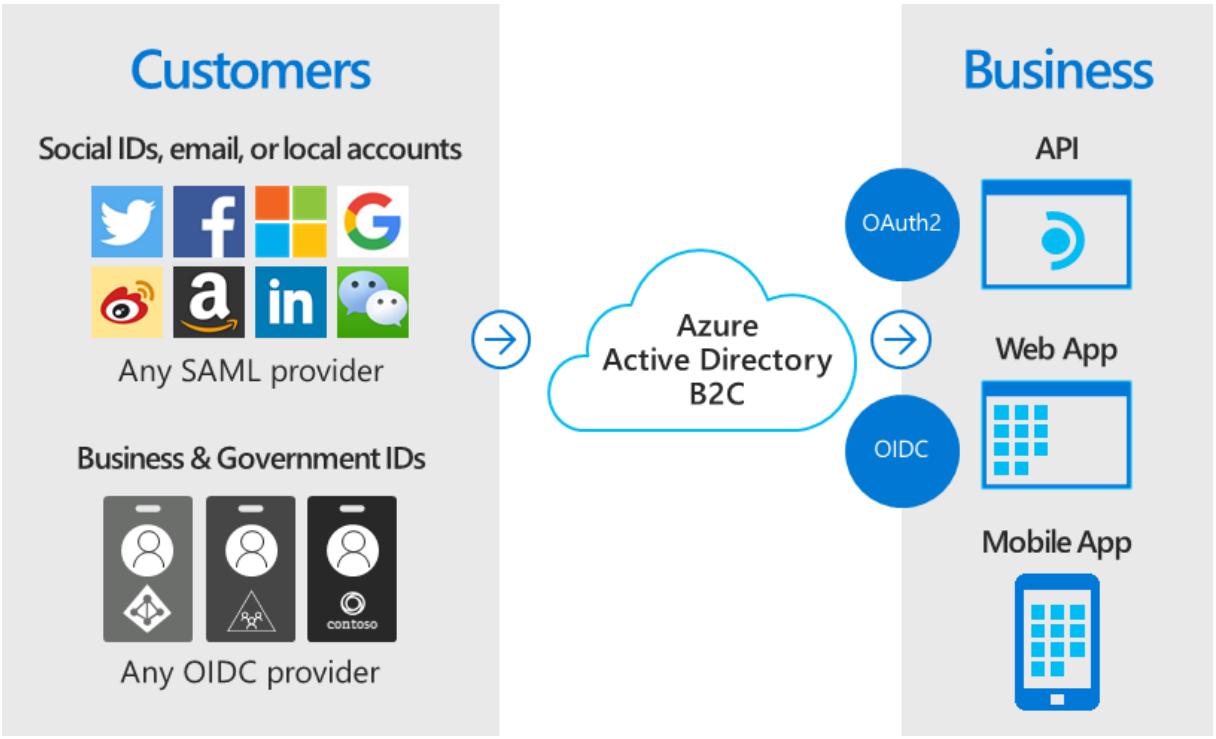


Figure 5 Azure Active Directory B2C integration

6.2 Ecosystem modelling

The dashboard provides insights for a single stakeholder at various level of abstraction. The ecosystem modelling tooling allows to view and analyse the ecosystem of background music. As such, it is inclusive in terms of actors considered. The ecosystem modelling tool is considered in detail in Deliverable D4.1 “Dashboard for ecosystem modelling”, but in this document, we share already some preliminary results. Also, we explain how the ecosystem modelling fits into the overall architect of Music360.

The ecosystem modelling environment is called EcoSphere and runs in the browser, with a backend to store and retrieve ecosystem models, and to handle the registration process.

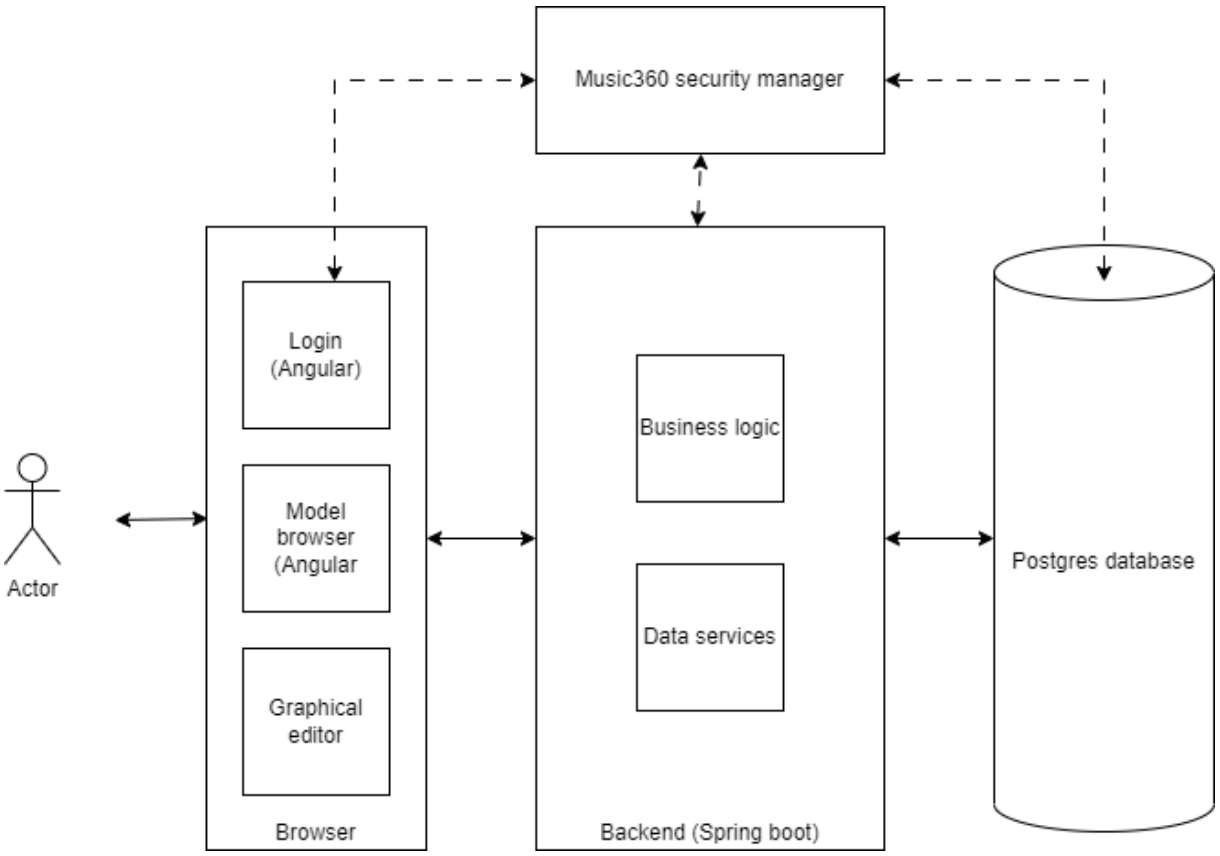


Figure 6 Overview of the EcoSphere architecture

Figure 6 presents the overall architecture of the EcoSphere business ecosystem modelling platform. From the view of the end-user, the platform is fully accessible via the Browser. This means that the end-user does not need to install any software. The frontend is divided into three parts: the login component (which also handles session management), a browser for ecosystem models stored, and the graphical editor to draw ecosystem models. The first two components use Angular as front-end framework, the editor is built using mxGraph, which is a framework to develop graphical editors that run in the browser. The frontend in the browser communicates with the backend via REST requests and responses. Currently, the backend is used to store and retrieve the models, to handle the registration process, to authenticate users, and to export graphical models to various formats. All data is stored in the Postgres database.

User authentication is handled by the EcoSphere frontend and backend. We want to offer facilities to be authenticated by the Music360 security manager too. Apart from a Single Sign On (SSO) for the full Music360 platform, this would also allow the EcoSphere platform to extract data from the Music360 data providers, which is the long term goal.

When the user hits the homepage of EcoSphere, first a login screen is presented (Figure 7). The dialog allows for registration of the users, which can be handled by the backend fully automatically, or may require human intervention to approve the registration.

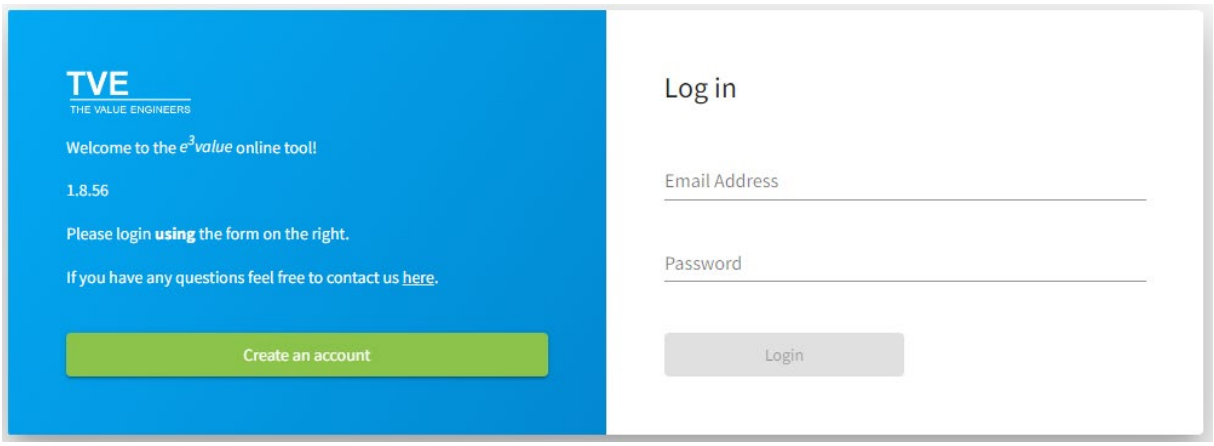


Figure 7 EcoSphere login screen

Once the user is logged in, a dashboard is shown (Figure 8).

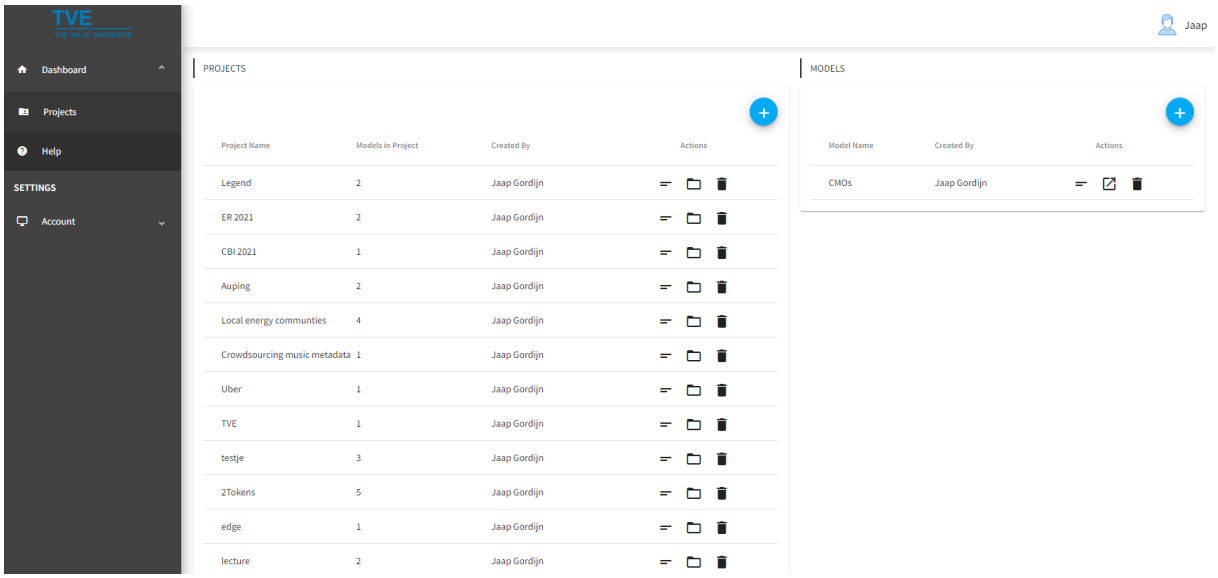


Figure 8 EcoSphere dashboard

Currently, the user can create and access projects (left), which contain one or ecosystem models (right). If the user selects a model, the graphical editor opens (Figure 9). Using drag-and-drop, the user can create ecosystem models based on the *e*^{3}*value* graphical modelling language.

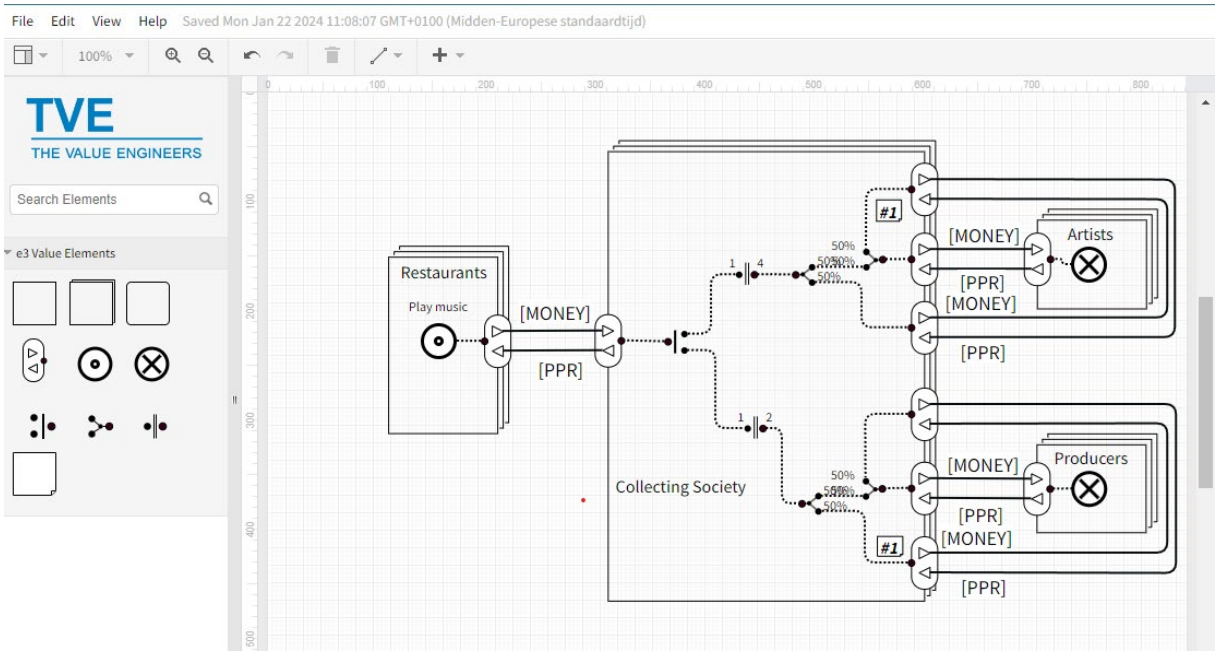


Figure 9 EcoSpere graphical editor

7 Data providers

7.1 The distributed Music360 database

7.1.1 Characterization of the data

The Music360 database contains information about the following:

- **Economic and non-economic value of music**: The database encapsulates information about creative entities, venues, and policy makers, the primary stakeholders in the music ecosystem. Thus, the database encapsulates user's credentials, personal details, and both economic and non-economic value metrics. Thus, Music360 ontology (see D2.1 "An ontology of the value of music – version 1") is derived in a set of tables to ensure that artists, producers, authors, and other creative entities are seamlessly integrated into the Music360 platform. The venues related data include details about professional users of music, storing venue-specific information alongside economic and non-economic value metrics, policy-related details, thus emphasising the platform's commitment to providing insights for informed decision-making related to the music value.
- **Music creation and use:** The Music360 database captures the essence of music itself. It includes details such as titles, genres, release dates, and audience metrics. Crucially, it establishes relationships with creative entities, ensuring that the creators are linked to their musical works, forming a comprehensive network within the database.
- **Music rights and licence management:** The database orchestrates the music rights and licence management through the Rights Societies and License Management information related to the different music works. The database stores the integration with neighbouring right and author right societies, fostering an ecosystem that respects and acknowledges the rights of creators, which includes licensing of musical works, and usage terms to facilitate legal and transparent interactions.
- **Music stakeholders and external software users:** The Music360 database considers the different stakeholders that interact with the Music360 platform and the creation and use of the music works. For those stakeholders that interact with the Music360 platform, specific REST services will be implemented to promote data interoperability and seamless integration with external systems. Living Labs users are also considered as providers of enriched data regarding the value of music through questionnaires and different artefacts to capture the economic and non-economic value of music in different application scenarios, thus further enriching the Music360 database.

7.1.2 Music360 Ontology Alignment

The Music360 ontology (see D2.1 "An ontology of the value of music – version 1") forms the guiding framework for the database, with concepts mapped meticulously to tables and fields. Economic and non-economic value metrics, the core of the Music360 ontology for measuring the value of music, are integrated seamlessly, ensuring a harmonious relationship between conceptual understanding and practical implementation.

In essence, the database for the Music360 platform emerges as a dynamic and secure repository, where every table and relationship is crafted with precision to harmonise the diverse data elements that are necessary to meet the Music360 project objectives. It not only captures the essence of music but also fosters collaboration, transparency, and innovation within the music ecosystem.

For the first phase of the project, we will use the data model as outlined in Deliverable D2.1 – An ontology of the value of music – version 1. This model is derived from the Music360 ontology and allows for a few simplifications.

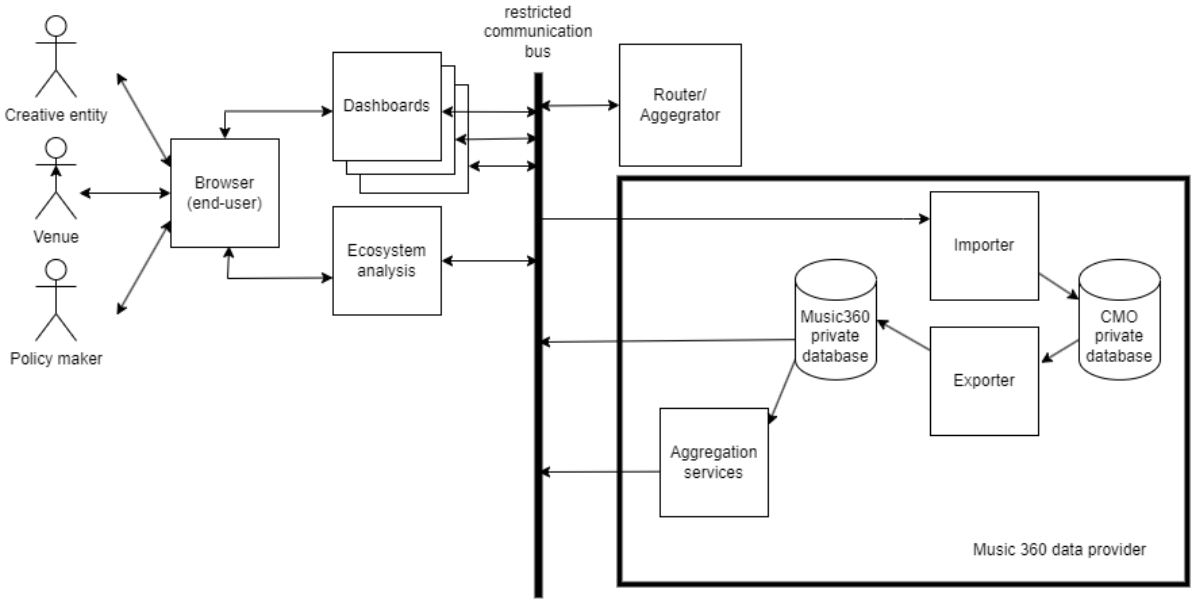


Figure 10 Architecture of the data provider

7.1.3 Partitioned and replicated databases

Conceptually, there is one database, namely the Music360 database, with one agreed data model, which stores the data that can be used by the front ends, such as the dashboards and ecosystem analysis tooling. However, this database will be partitioned, and if necessary, replicated.

Each data provider is responsible for operating a partition of the data. Usually, this partition corresponds to the data the data providers already have. Partitions may overlap, as the internal databases of data providers are not necessarily disjunct.

This setup satisfies concern C2 and possibly C3, namely that data owners stay in control with respect to access of their data. It is up to data provider to host their partition of the Music360 data. Because we have agreed about the identifiers, it is possible to relate data in different partitions well (e.g., relating a musical work managed by one data owner to a recording managed by another data owner).

It is important to understand that each partition has the same Music360 data model. Consequently, users of the data (dashboards, ecosystem analysis, router) do not have to deal with differences in (semantics of) data models.

For performance reasons it is allowed to have replicas of partitions. The following should then be realised:

1. The data owner keeps control over its data, that is, all replicas.
2. The data provider balances the load over the different replicas, which means that replicas are transparent to the rest of the Music360 platform.
3. The data provider ensures that the data in the replicas remain consistent, specially in the case of updates. It is sufficient for the use case of the platform if data is eventually consistent, meaning after some time.

To facilitate collaboration and interoperability, the database supports standardised data exchange formats like JSON and XML. Well-defined API endpoints enable external software users to seamlessly query the Music360 platform programmatically. Comprehensive documentation assists developers in understanding data formats, authentication processes, and REST API usage for data integration. Examples of such API endpoints are (non exhaustive):

- **Get user information:** Retrieve detailed information about a user, including creative entities, venues, and policy makers. The response includes user credentials, personal details, and associated economic and non-economic value metrics.
- **Get music creation details:** Fetch comprehensive details about a specific musical creation, including title, genre, release date, and audience metrics. This service establishes a connection with the creative entities responsible for the music.
- **Get venue information:** Retrieve information about a specific venue, including its name, location, and business details. This service also provides economic and non-economic value metrics associated with the venue.
- **Get rights society information:** Obtain details about a rights society, including its name, type (neighbouring right or author right), and other relevant details. This service supports transparency in rights management.
- **Get licence details:** Retrieve information about a specific licence, including the associated music, society, licence type, expiry date, and usage terms. This service aids in understanding the legal aspects of music usage.
- **Search music creations by criteria:** Search for music creations based on criteria such as genre, release date, or audience metrics. This service provides flexibility for users to explore the platform's musical content.
- **Get Living Lab data:** Obtain enriched data contributed by Living Labs, including responses from questionnaires. This service facilitates the extraction of subjective assessments related to the value of music.

7.2 The data provider database

Figure 2 also shows the private, already existing, database(s) of the data provider, here the database of a CMO. Databases of data providers are very different in terms of technology (operating system, type and brand of database management system), but also with respect to the data model used. For example, although CMOs are quite similar in terms of operations, the data model of their databases differs, and also the technology used is not the same.

The one and only assumption that we make with respect to the internal data provider databases is that, somehow, the database can be exported. This can be achieved via an API

(preferred solution), a database dump, or some other way, depending on the database management system used.

The private provider databases correspond to the databases of the different actors whose information must be stored in the Music360 platform.

We assume at least the following databases:

- CMO databases: Details about creative entities, including artists, producers, and authors. This may also include personal information, artistic aliases, and roles (artist/producer, author). This data establishes the roles of artists/producers and authors in the music plays. This data must be integrated with external software providers (such as music fingerprinting companies) that facilitate the music usage identification and the related metadata that is necessary for rights management.
- Venue databases: Details about venues, which are professional users of music. This may include venue names, locations, and other relevant business information, such as their revenue over periods of time, relatable to music played during that period.
- Music usage databases: Data about the use of music, e.g., provided by streaming providers or music finger printing companies. Often, this data is enriched with additional meta data, such as genre, tempo, and language.
- Living Labs databases: Enriched data contributed by Living Labs users, potentially gathered through questionnaires or other means. This data may include subjective assessments of the value of music.

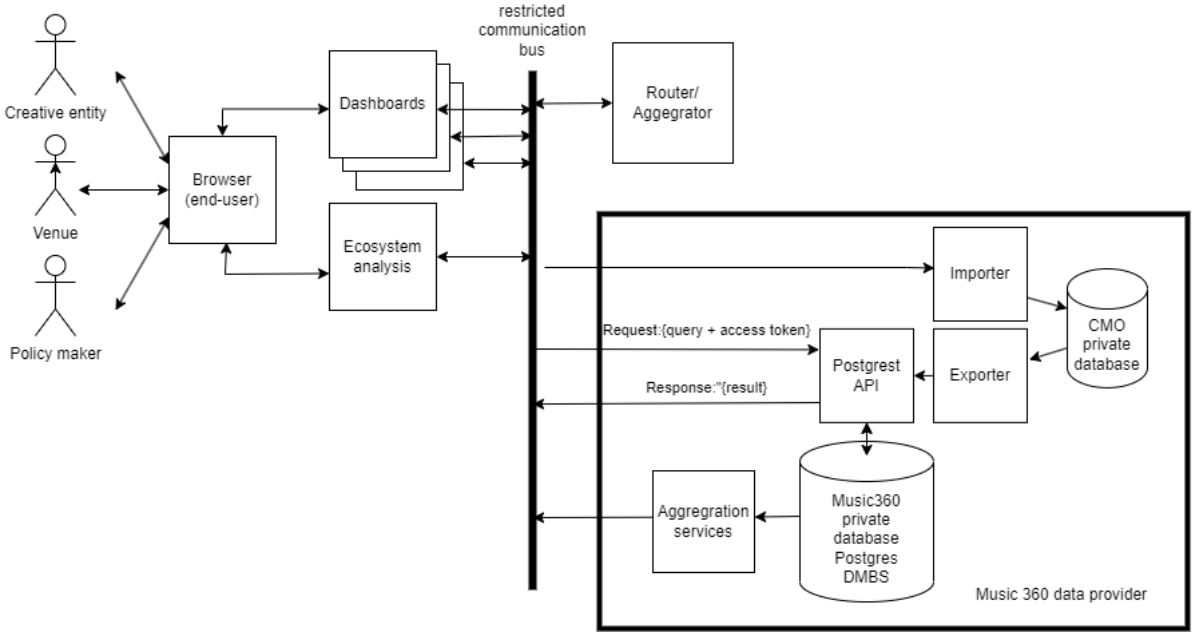


Figure 11 Architecture data provider

The Exporter component in Figure 7 retrieves data from the private database of the data provider (here a CMO) and loads the extracted data into the Music360 database partition of the data provider. To do so, the Exporter uses the published REST interface for the Music360 data model. Data can be loaded incrementally, or as a daily full snapshot of part of the data provider's database. Implementing the exporters is part of the Living Labs (WP6).

It is also possible that the data provider wants to load data from the Music360 platform into its own private database. This can be achieved by an Importer, which behaves just like any other

party who wants to query the Music360 data. This means that the Importer should have sufficient access rights to fetch the relevant data.

7.3 Aggregation services

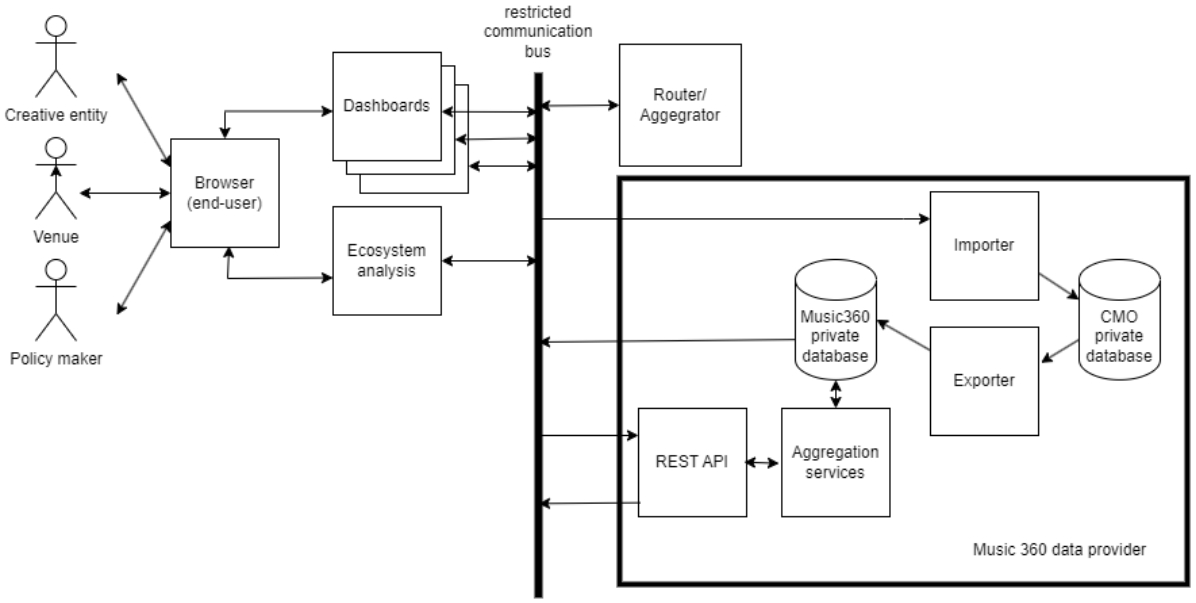


Figure 12 Architecture aggregation services

As explained before, two kinds of queries are distinguished:

1. Queries that explicitly refer to an identifiable Music360 object: recording, work, right owner, or right user
2. Aggregate queries, usually with the intent to summarise, totalize, group, etc.

Aggregate queries will be exposed by a REST API. Definition of this API largely depends on the needs of the Living Labs (WP6) and is currently under study. Aggregation queries come with their own security regime: A possible scenario is that all users cannot access the earnings of one specific and identifiable artist but can see the total earnings of all artists in a specific country. Hence, access control for aggregate queries is different from access control for queries referring to identifiable objects.

8 Security, authentication & authorization

Classic security concerns two topics: authentication (how do we know that the user is the one s/he claims to be) and authorization (controlling access to resources by the user). Deliverable D2.3 “Secure and trusted sharing of music data – version 1” deals with these security considerations in detail.

Additionally, there is a need to share data, without giving users access to that directly, but instead allow them to do certain controlled operations of that data, where the result may be visible to the users. Deliverable D2.6 “Secure and trusted sharing of music data – version 2” deals with providing data to untrusted parties.

What follows now is a high-level explanation of how Music360 will execute authentication and authorization.

8.1 Multi-source authentication

A multi-source authentication mechanism, often referred to as Federated Identity Management, supports multiple identity providers to enable users to access a system seamlessly using credentials from various sources.

The multi-source authentication mechanism for the Music360 platform is designed to offer users a seamless and secure authentication experience by supporting authentication from multiple identity providers (IPs). Users can log in using credentials from diverse sources, including Music360 and other relevant providers such as EU collaborative projects (such as OpenMuse). The Music360 multi-source authentication mechanism supports a variety of identity providers, including but not limited to Music360 Credential, EU Related Projects, and other custom or industry-standard providers (OpenID Connect, OAuth 2.0, etc.).

The authentication flow is as follows: The user initiates the login process by selecting the preferred identity provider on the Music360 login page via its browser (the Client) and is redirected to the corresponding Identity Provider (IP) for the user authentication.

Upon successful authentication, the identity provider generates an authentication token and returns it to the Music360 platform. The platform behaves as a resource provider (RP), which validates the received token with the corresponding identity provider to ensure its authenticity. This token is then used to get access to data and services, depending on the authenticated users. Data and services are provided by Resource Providers (RPs). All this is handled by the login and security manager.

If it is the user's first time logging in, the Music360 platform creates a new user account, based on the user details provided. Note that it is only necessary to store the username and not the password, as storing the password is the responsibility of the IP. Otherwise, it updates the existing account with any new information from the identity provider.

Also, if the user logs in for the first time, he is asked to login to a neighbouring right CMO, and/or an author right CMO. This is used to connect the neighbouring right role of a user to his author right role. This information is not available in the current state-of-the-art of music metadata. Harvesting this information is an important contribution of the Music360 platform.

8.2 Access control

Once the user is properly authenticated, the user can use the platform via one of the dashboards, or the ecosystem analysis tool. All these components (dashboard, ecosystem analysis) act as a resource provider; based on the token provided, representing the authenticated user, services and/or data are provided, and access rights are checked. This checking of access rights will be delegated as much as possible to the private Music360 databases of the data providers. This ensures that data providers will remain in control with respect to their data. The only assumption is that data owners and providers trust the identity providers, that is assume that they reliably and correctly authenticate users.

8.3 The Music360 user database

The Music360 platform considers different users roles that have different interactions with the platform and each other according to the following elements: The participation in the generation of relevant information to evaluate the monetary and non-monetary use of the music creations; the enrichment of this information with meta data that is valuable to identify the applicability of certain music works in specific context to evaluate the positive or negative impact that can have according to specific venues and use; the use of the platform information to the management of royalties and authors rights, and the analysis of the data generated to monitoring the proper royalties distribution and assure the transparent and adequate benefits assignment. In this context, the Music360 user's roles can be defined as follows:

- Creative entities: These are individuals or entities that provide music. They can log in to the Music360 platform using their Music360 credentials or other authorized identity providers (including CMOs). A single person may play multiple roles, particularly in the context of being both an author and an artist/producer. The distinction between these roles is essential for handling metadata about music usage accurately.
- Venues: These are professional users who use music in their venues. They also log in to the Music360 platform with various credentials. For each company, there is an administrative user who can add/delete users of the venue.
- Policymakers: These users access the platform to understand the management of music benefits to take decisions about how to regulate the distribution of these benefits according to the real value of music.
- Other users/systems: These interact with the Music360 platform programmatically via a published REST API. These users can represent various actors such as CMOs (Collective Management Organizations), venues, finger printing companies, etc.
- Living Labs users: Users who enter data about the value of music from different venues into the platform. This data may come from enriched questionnaires or other sources.

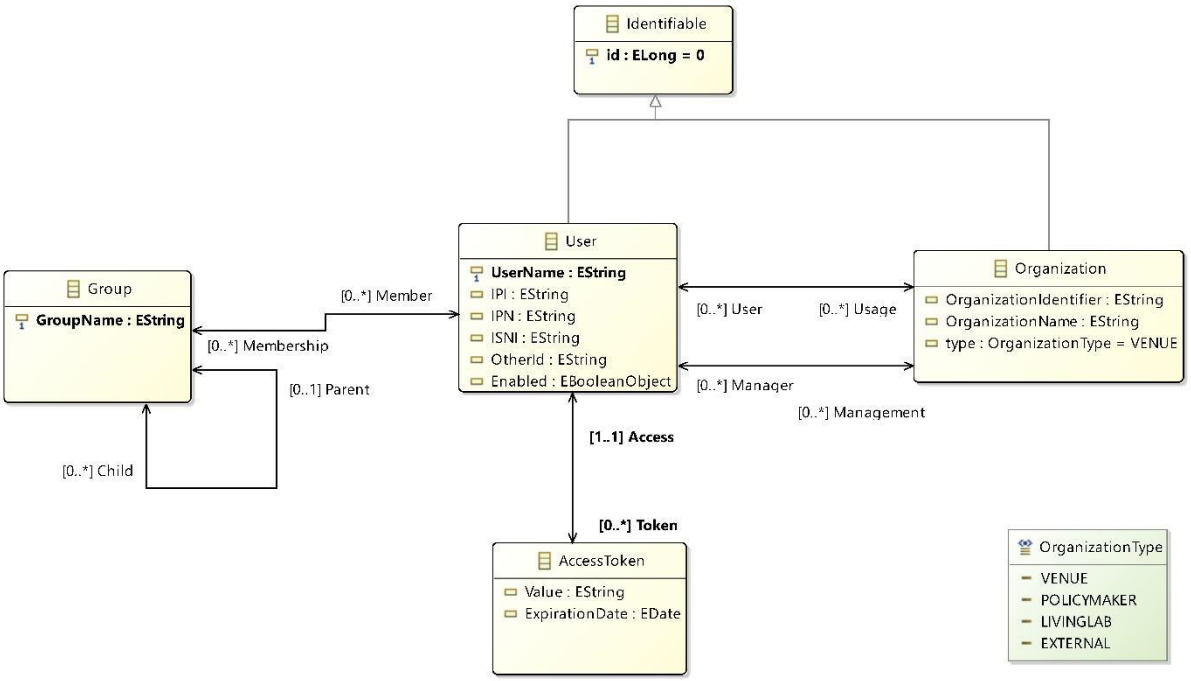


Figure 13 User model

Figure 9 presents the Music360 user data model:

- A user can be a creative entity (everyone with an IPI and/or IPN), a policy maker, or music users, among others.
- Users may be associated with an organisation, which can be a venue, policymaker, Living Lab or an external organisation. Organisations have one or more administrative users who can grant access to other users of that organisation, based on the granted rights to that organisation.
- Users may be members of a group. Groups may have children, who are also members. Rights can also be granted to groups.
- A user may have access tokens. These access tokens are used by external services to access the platform.

Security-wise, the Music360 database behaves as a resource provider. This means that the database receives the token generated during user login to check the right to access specific data. By delegating access control to the database as much as possible, a uniform access control mechanism is created that executes access control precisely the same for all dashboards.

9 Implementation

9.1 Authentication and access control

9.1.1 Overview

Authentication will be implemented using OpenID Connect, which is based on OAuth2. For this purpose, a reference Music360 identity server will be made available, based on open source software. In phase of the technical deployment, we will decentralize the identity server to selected data providers in the Music360 project. They will run their own identity server. This would mean that other data providers should trust the data providers running also an identity server.

The identity server will register the credentials of the users, and the claims they have. Currently these claims are:

- Author right holder
- Neighbouring right holder
- Music usage data provider
- CMO
- Venue
- Living lab
- External user

After the user logged in, the identity server will provide to the client (effectively the browser) a JWT token, which contains a header, payload data (username + claims) and a signature. Specific actions are taken to ensure secure storage in the browser

The token will be used by the user to get access to data. To this end, the REST backends of the data providers and their databases act as resource providers. In short, this means that the backend and database checks whether the token is valid (this can be done by checking the validity of the token by means of the enclosed signature).

The database stores per row in the database tables an access control list, consisting of principals (user) which may have access rights (e.g. read/write/create/delete) for that row. By implementing access control directly in the database, the system will be less vulnerable to access at the backend. This implements requirement C2. Access control that allows securing at the property level of the database will be part of Deliverable D2.6 “Secure and trusted sharing of music data – version 2).

9.1.2 Technology

For the identity server, the open-source server Keycloak (see https://www.keycloak.org/) will be used. The database that will be used is Postgres, since Postgres has excellent support for Row Level Security (RLS). To open up the database, we use the open source package PostgREST (https://postgrest.org/) that exposes the full database as REST services.

9.2 Router

9.2.1 Overview

The task of the router is to receive requests for data from the front-ends, to dispatch these requests to the appropriate data provider(s), to receive the responses and the forward them to the front-ends.

In particular, requests are forwarded to the right data provider(s) based on the content in the request, which can be any field in general, and an identifier (IPI, IPN, ISRC, ISWC) in particular. This is actually close to enterprise application integration, and content-based routing specifically (see e.g. https://www.enterpriseintegrationpatterns.com/patterns/messaging/ContentBasedRouter.html for a more detailed explanation).

Routing should be very fast, e.g. limit the additional latency used because of the routing process.

9.2.2 Technology

We will use a well known enterprise integration toolkit, which is also open source, namely Apache Camel (see https://camel.apache.org/). Data providers register their database sources based on the identifier(s) of the Music360 objects, and also remove registrations once the database is removed from their respective Music360 database partition. Routes to destinations (see Figure 3 for an introduction) are stored in a fast database, most likely an in-memory database. We consider amongst others Ignite (https://ignite.apache.org/), Ehcache (https://www.ehcache.org/)m, and Hazelcat (https://hazelcast.com/).

9.3 Music360 database

9.3.1 Overview

The Music360 database is a distributed, replicated, and partitioned database. Every data owner, or the data provider representing that data owner can stay in control of their own data, by being the administrative and technical responsible entity for their own data. Below we discuss the technology used as well as the data schema for the database. This data schema will be replicated over all instances of the Music360 database.

9.3.2 Technology

The Music360 database (which is partitioned and/or replicated by each data provider will be implemented using PostgreSQL (see https://www.postgresql.org/). This is a reliable, battle-tested and open source database management system. The data will be made available by using PostgREST, which exposes a full database as a collection of REST services (see https://postgrest.org/).

9.3.3 Data model for living labs version 1

For the sake of completeness, the slightly updated data model that we will use for the first phase of the living labs is shown in Figure 14. For further information, see D2.1 “An ontology of the value of music – version 1”.

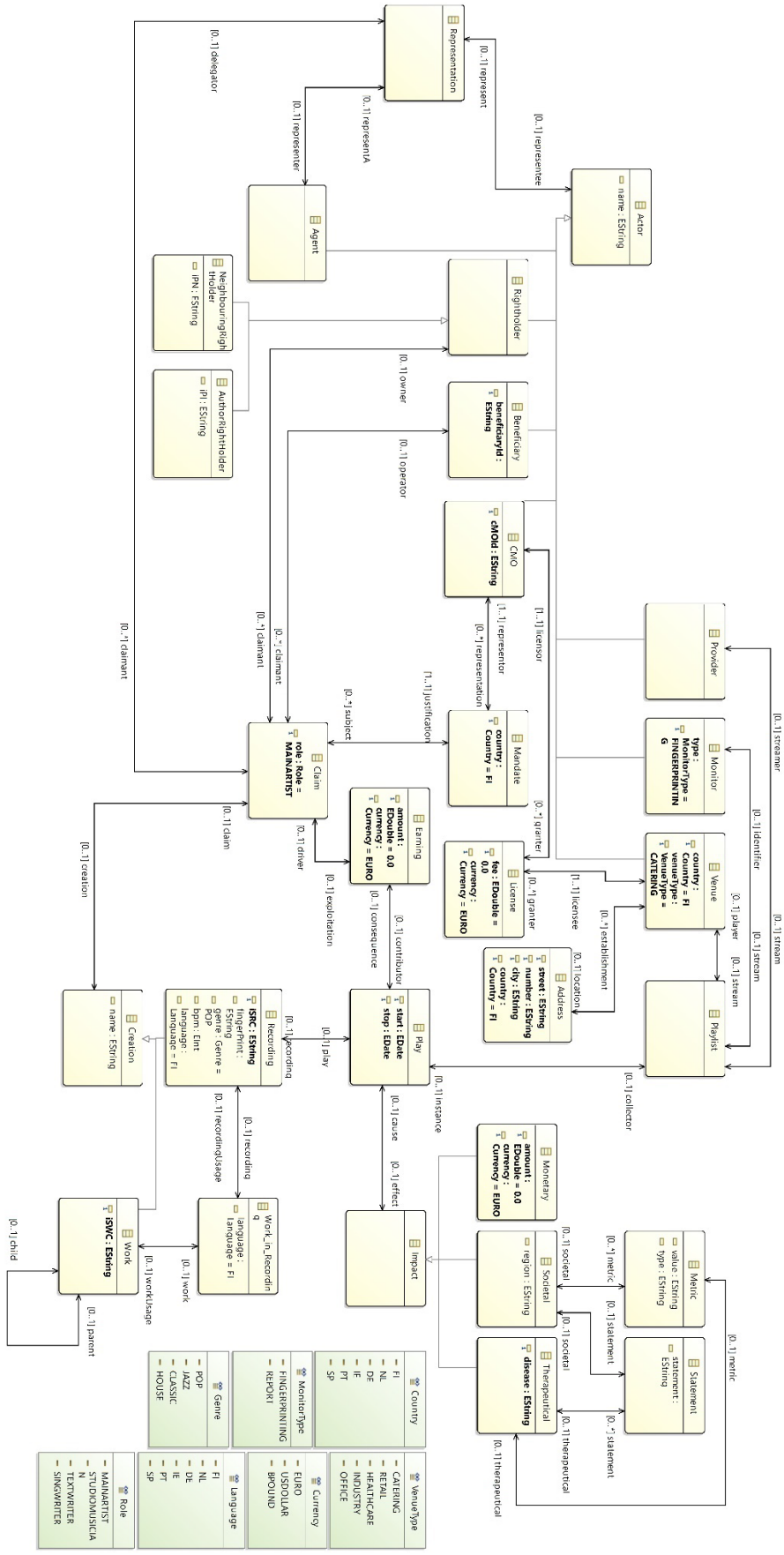


Figure 14 Music360 metamodel - phase 1

9.3.4 Data model for the Music360 ontology

For the sake of completeness, the data model for the Music360 ontology is shown in Figure 15. For further information, see D2.1 “An ontology of the value of music – version 1”.

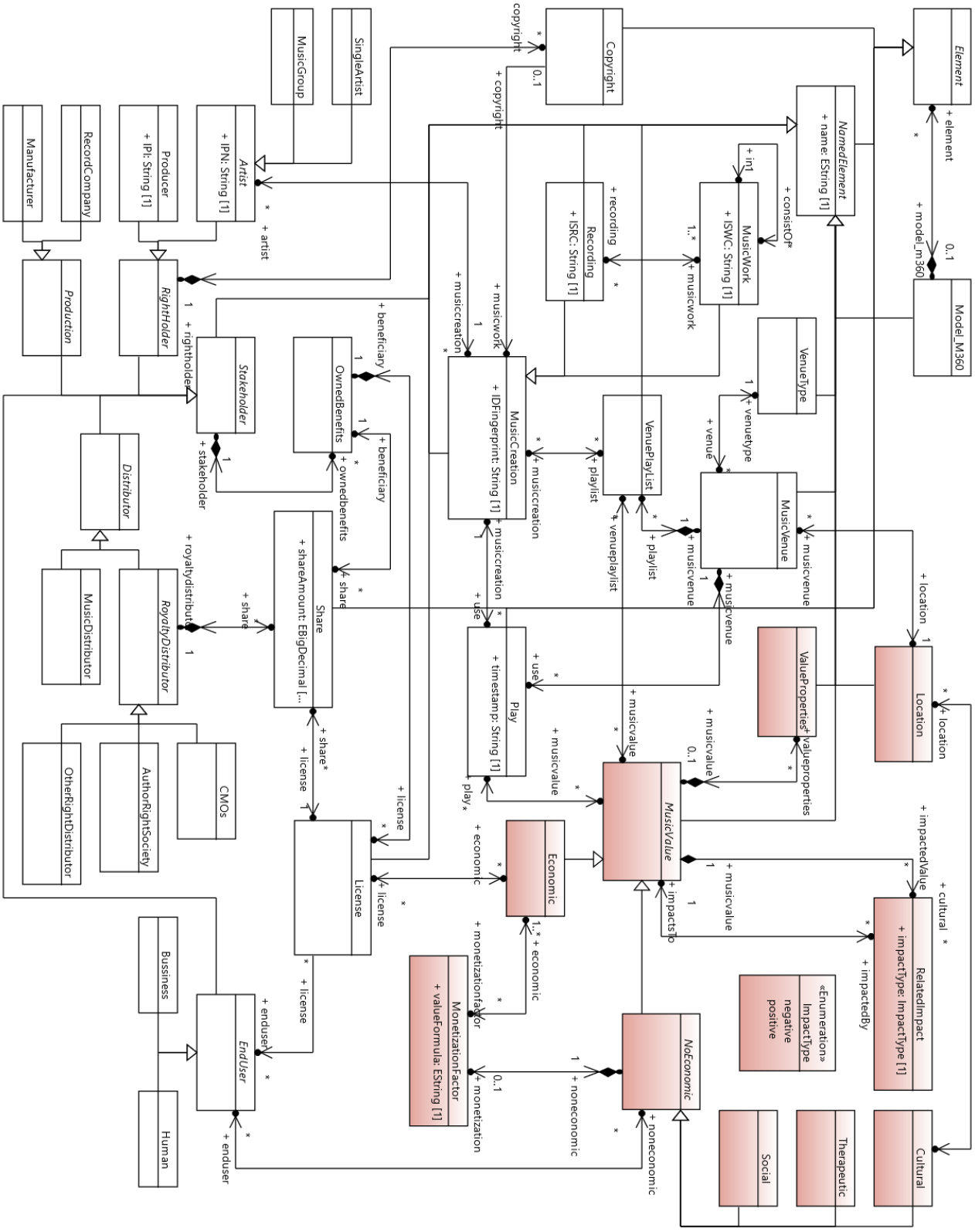


Figure 15 Music360 ontology, cf Deliverable D2.1

9.3.5 Creating the databases

Translating a data model represented by a UML class diagram into a relational model involves mapping classes, attributes, associations, generalization/specialization relationships, and handling many-to-many relationships. In the context of the Music360 platform, we are using the following rules to obtain the relation model for the Music360 database from the reference ontology:

- **Many-to-many relationships**: Many-to-many relationships in the Music360 data model are translated into a relational database model using an associative (or junction) table. This table acts as a bridge between the two entities involved in the many-to-many relationship and has a composite primary keys referencing the participating entities.
- **Concrete and abstract metaclasses:** For the generation of the Music360 database most of the abstract and concrete metaclasses are represented as tables in the Music360 database model. This rule considers the following exceptions:
    - The Model_M360, Element, and NamedElement metaclasses are not represented in the database model, and the corresponding attributes are defined directly in the child classes. These metaclasses are omitted because they are relevant for instantiating the ontological model, but this structure is not appropriate for a relational model implementation because it generates multiple relationships across the Element and NamedElement classes.
    - Child classes that do not provide new attributes or relationships are not represented in the database model. Instead, the attribute "type" is defined in the table representing the immediate parent class. This is considered to simplify the database model, but can be changed if the child classes introduce new properties in future versions of the ontology.
- **Generalization/Specialization:** To translate the generalization/specialization relationship defined in the Music360 data model, the parent and child classes are implemented as different tables in the Music360 database model, maintaining a unique identifier as a common connector to ensure consistency with the inheritance hierarchy. Attributes are defined in the appropriate classes. This implies that to obtain a complete child class representation, it is necessary to perform a join with all the tables representing the parent classes in the hierarchy. This approach is feasible in the context of the Music360 database because most of the generalizations/specializations defined are disjoins, thus reducing the amount of overlapping information between different tables.

10 Conclusion

This deliverable outlined the overall Music360 architecture. The architecture is highly distributed to cater for implementation of secure data ownership and scalability.

The main components of the architecture are:

- The browser of the user; everything end-user facing runs in the browser. Consequently, the user does not need to install specific software, and the platform is available via all major browsers.
- Two frontends: dashboards for understanding the value of music at various aggregation level (e.g. individual recording, work, performer, sing & song writer, publisher, producer, venue, recording and work), and understanding of the music industry at the ecosystem level (EcoSphere).
- The login/security manager; deals with login/logout, registration, and (in phase 2 of the project) data sharing to untrusted parties.
- The identity server, responsible to handle authentication conform OpenID connect / OAuth, and provisioning of authorization rules.
- The router, handling requests from the frontends, dispatching them to the appropriate data provider(s), receiving the results, and sending them back to the frontends.
- Data providers, which each can host their own partition of the Music360 database.
- Import- and export services to connect to the local databases of the data providers.
- An API gateway, to make data available to third parties.
- A secure network communication that allows for communication between all components.

Where appropriate, we have indicated which deliverables deal with which architectural component(s). The shared infrastructure consists of the secure network, the router and the replicated and partitioned Music360 database. In Appendix A and B we have explained the implementation of this database in terms of the relational model, and associated constraints.

11 Appendix A Music360 ontology implemented in the relation model

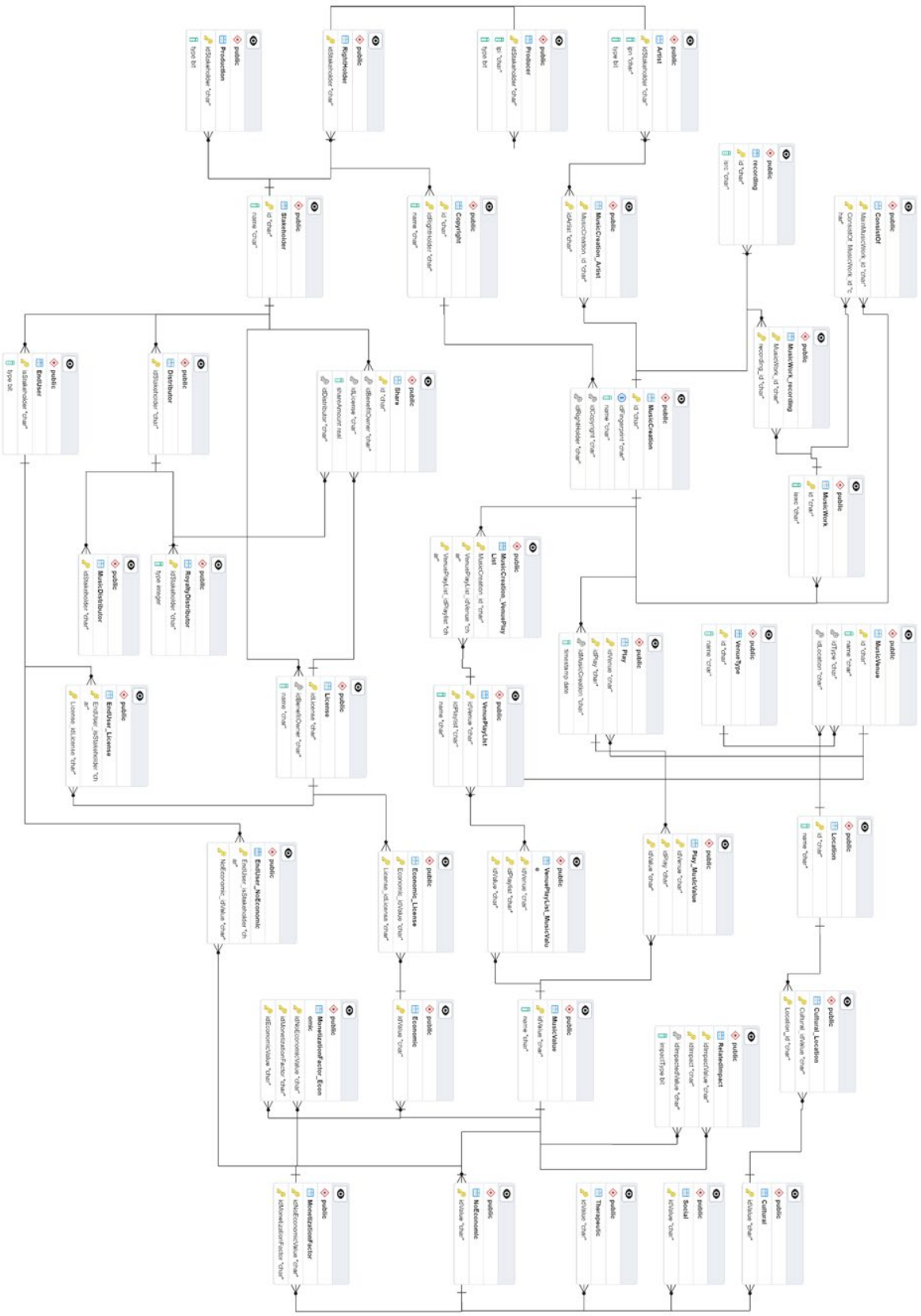


Figure 16 Relational Database Model for the Music360 ontology

CREATE TABLE IF NOT EXISTS public."MusicWork"

```
(
        id "char" NOT NULL,
        iswc "char" NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.recording
(
        id "char" NOT NULL,
        isrc "char" NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public."Artist"
(
        "idStakeholder" "char" NOT NULL,
        ipn "char" NOT NULL,
        type bit NOT NULL,
        PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."Copyright"
(
        id "char" NOT NULL,
        "idRightHolder" "char" NOT NULL,
        name "char" NOT NULL,
        PRIMARY KEY (id, "idRightHolder")
);

CREATE TABLE IF NOT EXISTS public."MusicCreation"
(
        id "char" NOT NULL,
    "idFingerprint" "char" NOT NULL,
```

```
        name "char",
        "idCopyright" "char",
        "idRightHolder" "char",
        PRIMARY KEY (id),
        UNIQUE NULLS NOT DISTINCT ("idFingerprint"),
        UNIQUE ("idCopyright", "idRightHolder")
);

CREATE TABLE IF NOT EXISTS public."MusicCreation_Artist"
(
        "MusicCreation_id" "char" NOT NULL,
        "idArtist" "char" NOT NULL,
        PRIMARY KEY ("MusicCreation_id", "idArtist")
);

CREATE TABLE IF NOT EXISTS public."MusicWork_recording"
(
        "MusicWork_id" "char" NOT NULL,
        recording_id "char" NOT NULL,
        PRIMARY KEY ("MusicWork_id", recording_id)
);

CREATE TABLE IF NOT EXISTS public."VenuePlayList"
(
        "idVenue" "char" NOT NULL,
        "idPlaylist" "char" NOT NULL,
        name "char" NOT NULL,
        PRIMARY KEY ("idVenue", "idPlaylist")
);

CREATE TABLE IF NOT EXISTS public."MusicVenue"
(
        id "char" NOT NULL,
```

```
        name "char" NOT NULL,
        "idType" "char",
        "idLocation" "char",
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public."VenueType"
(
        id "char",
        name "char",
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public."Play"
(
        "idVenue" "char" NOT NULL,
        "idPlay" "char" NOT NULL,
        "idMusicCreation" "char" NOT NULL,
        "timestamp" date,
        PRIMARY KEY ("idPlay", "idVenue")
);

CREATE TABLE IF NOT EXISTS public."MusicCreation_VenuePlayList"
(
        "MusicCreation_id" "char" NOT NULL,
        "VenuePlayList_idVenue" "char" NOT NULL,
        "VenuePlayList_idPlaylist" "char" NOT NULL,
        PRIMARY KEY ("VenuePlayList_idPlaylist", "VenuePlayList_idVenue",
"MusicCreation_id")
);

CREATE TABLE IF NOT EXISTS public."ConsistOf"
(
        "MainMusicWork_id" "char" NOT NULL,
```

```
        "ConsistOf_MusicWork_id" "char" NOT NULL,
        PRIMARY KEY ("MainMusicWork_id", "ConsistOf_MusicWork_id")
);

CREATE TABLE IF NOT EXISTS public."License"
(
        "idLicense" "char" NOT NULL,
        "idBenefitOwner" "char",
        name "char" NOT NULL,
        PRIMARY KEY ("idLicense")
);

CREATE TABLE IF NOT EXISTS public."Share"
(
        id "char" NOT NULL,
        "idBenefitOwner" "char" NOT NULL,
        "idLicense" "char" NOT NULL,
        "shareAmount" real,
        "idDistributor" "char" NOT NULL,
        PRIMARY KEY ("idBenefitOwner")
);

CREATE TABLE IF NOT EXISTS public."Stakeholder"
(
        id "char" NOT NULL,
        name "char" NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public."MusicValue"
(
        "idValue" "char" NOT NULL,
        name "char" NOT NULL,
```

```
        PRIMARY KEY ("idValue")
);

CREATE TABLE IF NOT EXISTS public."Economic"
(
        "idValue" "char" NOT NULL,
        PRIMARY KEY ("idValue")
);

CREATE TABLE IF NOT EXISTS public."Economic_License"
(
        "Economic_idValue" "char" NOT NULL,
        "License_idLicense" "char" NOT NULL,
        PRIMARY KEY ("Economic_idValue", "License_idLicense")
);

CREATE TABLE IF NOT EXISTS public."Play_MusicValue"
(
        "idVenue" "char" NOT NULL,
        "idPlay" "char" NOT NULL,
        "idValue" "char" NOT NULL,
        PRIMARY KEY ("idVenue", "idPlay", "idValue")
);

CREATE TABLE IF NOT EXISTS public."Location"
(
        id "char" NOT NULL,
        name "char" NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public."VenuePlayList_MusicValue"
(
```

```
        "idVenue" "char" NOT NULL,
        "idPlaylist" "char" NOT NULL,
        "idValue" "char" NOT NULL,
        PRIMARY KEY ("idPlaylist", "idVenue", "idValue")
);

CREATE TABLE IF NOT EXISTS public."RightHolder"
(
        "idStakeholder" "char" NOT NULL,
        PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."Production"
(
        "idStakeholder" "char" NOT NULL,
        type bit NOT NULL,
        PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."Producer"
(
        "idStakeholder" "char" NOT NULL,
        ipi "char" NOT NULL,
        type bit NOT NULL,
        PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."Distributor"
(
        "idStakeholder" "char" NOT NULL,
        PRIMARY KEY ("idStakeholder")
);
```

```
CREATE TABLE IF NOT EXISTS public."EndUser"
(
	"isStakeholder" "char" NOT NULL,
	type bit NOT NULL,
	PRIMARY KEY ("isStakeholder")
);

CREATE TABLE IF NOT EXISTS public."RoyaltyDistributor"
(
	"idStakeholder" "char" NOT NULL,
	type integer NOT NULL,
	PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."MusicDistributor"
(
	"idStakeholder" "char" NOT NULL,
	PRIMARY KEY ("idStakeholder")
);

CREATE TABLE IF NOT EXISTS public."EndUser_License"
(
	"EndUser_isStakeholder" "char" NOT NULL,
	"License_idLicense" "char" NOT NULL,
	PRIMARY KEY ("EndUser_isStakeholder", "License_idLicense")
);

CREATE TABLE IF NOT EXISTS public."NoEconomic"
(
	"idValue" "char" NOT NULL,
	PRIMARY KEY ("idValue")
);
```

```
CREATE TABLE IF NOT EXISTS public."MonetizationFactor"
(
        "idNoEconomicValue" "char" NOT NULL,
        "idMonetizationFactor" "char" NOT NULL,
        PRIMARY KEY ("idNoEconomicValue", "idMonetizationFactor")
);

CREATE TABLE IF NOT EXISTS public."MonetizationFactor_Economic"
(
        "idNoEconomicValue" "char" NOT NULL,
        "idMonetizationFactor" "char" NOT NULL,
        "idEconomicValue" "char" NOT NULL,
        PRIMARY KEY ("idMonetizationFactor", "idNoEconomicValue", "idEconomicValue")
);

CREATE TABLE IF NOT EXISTS public."RelatedImpact"
(
        "idImpactValue" "char" NOT NULL,
        "idImpact" "char" NOT NULL,
        "idImpactedValue" "char" NOT NULL,
        "impactType" bit,
        PRIMARY KEY ("idImpactValue")
);

CREATE TABLE IF NOT EXISTS public."Cultural"
(
        "idValue" "char" NOT NULL,
        PRIMARY KEY ("idValue")
);

CREATE TABLE IF NOT EXISTS public."Therapeutic"
(
        "idValue" "char" NOT NULL,
```

```
        PRIMARY KEY ("idValue")
);

CREATE TABLE IF NOT EXISTS public."Social"
(
        "idValue" "char" NOT NULL,
        PRIMARY KEY ("idValue")
);

CREATE TABLE IF NOT EXISTS public."Cultural_Location"
(
        "Cultural_idValue" "char" NOT NULL,
        "Location_id" "char" NOT NULL,
        PRIMARY KEY ("Cultural_idValue", "Location_id")
);

CREATE TABLE IF NOT EXISTS public."EndUser_NoEconomic"
(
        "EndUser_isStakeholder" "char" NOT NULL,
        "NoEconomic_idValue" "char" NOT NULL,
        PRIMARY KEY ("EndUser_isStakeholder", "NoEconomic_idValue")
);

ALTER TABLE IF EXISTS public."MusicWork"
        ADD FOREIGN KEY (id)
        REFERENCES public."MusicCreation" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID;


ALTER TABLE IF EXISTS public.recording
        ADD FOREIGN KEY (id)
```

```
        REFERENCES public."MusicCreation" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Artist"
        ADD FOREIGN KEY ("idStakeholder")
        REFERENCES public."RightHolder" ("idStakeholder") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID;


ALTER TABLE IF EXISTS public."Copyright"
        ADD FOREIGN KEY ("idRightHolder")
        REFERENCES public."RightHolder" ("idStakeholder") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID;


ALTER TABLE IF EXISTS public."MusicCreation"
        ADD FOREIGN KEY ("idCopyright", "idRightHolder")
        REFERENCES public."Copyright" (id, "idRightHolder") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID;


ALTER TABLE IF EXISTS public."MusicCreation_Artist"
        ADD FOREIGN KEY ("MusicCreation_id")
        REFERENCES public."MusicCreation" (id) MATCH SIMPLE
```

```
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicCreation_Artist"
    ADD FOREIGN KEY ("idArtist")
    REFERENCES public."Artist" ("idStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicWork_recording"
    ADD FOREIGN KEY ("MusicWork_id")
    REFERENCES public."MusicWork" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicWork_recording"
    ADD FOREIGN KEY (recording_id)
    REFERENCES public.recording (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."VenuePlayList"
    ADD FOREIGN KEY ("idVenue")
    REFERENCES public."MusicVenue" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
```

```
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicVenue"
    ADD FOREIGN KEY ("idType")
    REFERENCES public."VenueType" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicVenue"
    ADD FOREIGN KEY ("idLocation")
    REFERENCES public."Location" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Play"
    ADD FOREIGN KEY ("idVenue")
    REFERENCES public."MusicVenue" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Play"
    ADD FOREIGN KEY ("idMusicCreation")
    REFERENCES public."MusicCreation" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
```

```
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicCreation_VenuePlayList"
    ADD FOREIGN KEY ("MusicCreation_id")
    REFERENCES public."MusicCreation" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicCreation_VenuePlayList"
    ADD FOREIGN KEY ("VenuePlayList_idVenue", "VenuePlayList_idPlaylist")
    REFERENCES public."VenuePlayList" ("idVenue", "idPlaylist") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."ConsistOf"
    ADD FOREIGN KEY ("MainMusicWork_id")
    REFERENCES public."MusicWork" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."ConsistOf"
    ADD FOREIGN KEY ("MainMusicWork_id")
    REFERENCES public."MusicWork" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public."License"
    ADD FOREIGN KEY ("idBenefitOwner")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Share"
    ADD FOREIGN KEY ("idBenefitOwner")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Share"
    ADD FOREIGN KEY ("idLicense")
    REFERENCES public."License" ("idLicense") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Share"
    ADD FOREIGN KEY ("idDistributor")
    REFERENCES public."RoyaltyDistributor" ("idStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public."Economic"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Economic_License"
    ADD FOREIGN KEY ("Economic_idValue")
    REFERENCES public."Economic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Economic_License"
    ADD FOREIGN KEY ("License_idLicense")
    REFERENCES public."License" ("idLicense") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Play_MusicValue"
    ADD FOREIGN KEY ("idPlay", "idVenue")
    REFERENCES public."Play" ("idPlay", "idVenue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public."Play_MusicValue"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."VenuePlayList_MusicValue"
    ADD FOREIGN KEY ("idVenue", "idPlaylist")
    REFERENCES public."VenuePlayList" ("idVenue", "idPlaylist") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."VenuePlayList_MusicValue"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."RightHolder"
    ADD FOREIGN KEY ("idStakeholder")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Production"
```

```
    ADD FOREIGN KEY ("idStakeholder")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Producer"
    ADD FOREIGN KEY ("idStakeholder")
    REFERENCES public."RightHolder" ("idStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Distributor"
    ADD FOREIGN KEY ("idStakeholder")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."EndUser"
    ADD FOREIGN KEY ("isStakeholder")
    REFERENCES public."Stakeholder" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."RoyaltyDistributor"
    ADD FOREIGN KEY ("idStakeholder")
```

```
    REFERENCES public."Distributor" ("idStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MusicDistributor"
    ADD FOREIGN KEY ("idStakeholder")
    REFERENCES public."Distributor" ("idStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."EndUser_License"
    ADD FOREIGN KEY ("EndUser_isStakeholder")
    REFERENCES public."EndUser" ("isStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."EndUser_License"
    ADD FOREIGN KEY ("License_idLicense")
    REFERENCES public."License" ("idLicense") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."NoEconomic"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
```

```
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MonetizationFactor"
    ADD FOREIGN KEY ("idNoEconomicValue")
    REFERENCES public."NoEconomic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MonetizationFactor_Economic"
    ADD FOREIGN KEY ("idNoEconomicValue", "idMonetizationFactor")
    REFERENCES public."MonetizationFactor" ("idNoEconomicValue",
"idMonetizationFactor") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."MonetizationFactor_Economic"
    ADD FOREIGN KEY ("idEconomicValue")
    REFERENCES public."Economic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."RelatedImpact"
    ADD FOREIGN KEY ("idImpactValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
```

```
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."RelatedImpact"
    ADD FOREIGN KEY ("idImpactedValue")
    REFERENCES public."MusicValue" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Cultural"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."NoEconomic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Therapeutic"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."NoEconomic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Social"
    ADD FOREIGN KEY ("idValue")
    REFERENCES public."NoEconomic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
```

```
    NOT VALID;


ALTER TABLE IF EXISTS public."Cultural_Location"
    ADD FOREIGN KEY ("Cultural_idValue")
    REFERENCES public."Cultural" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."Cultural_Location"
    ADD FOREIGN KEY ("Location_id")
    REFERENCES public."Location" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."EndUser_NoEconomic"
    ADD FOREIGN KEY ("EndUser_isStakeholder")
  REFERENCES public."EndUser" ("isStakeholder") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."EndUser_NoEconomic"
    ADD FOREIGN KEY ("NoEconomic_idValue")
    REFERENCES public."NoEconomic" ("idValue") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

12 Appendix B Music datamodel phase 1 implemented in the relational model

An explanation of the data model be used by the Living Labs phase 1 can be found in Deliveravle D2.1 “An ontology of the value of music – version 1”. A revised version of that model can be found below:

There is Java Persistence API available for developers to retrieve, update, and create instanced following the data model Figure 11. This library is available at https://www.dise-lab.nl/wp-content/uploads/2024/01/music360-datamodels-1.zip.

Below, the required SQL expressions are given to create the database scheme in a relational database.

```
CREATE TABLE metamodel_actor (
  dtype VARCHAR(31) NOT NULL,
  id BIGINT NOT NULL,
  name VARCHAR(255),
  ipi VARCHAR(255),
  beneficiary_id VARCHAR(255),
  cmoid VARCHAR(255),
  type VARCHAR(255),
  ipn VARCHAR(255),
  country VARCHAR(255),
  venue_type VARCHAR(255),
  stream_id BIGINT,
  location_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_actor_claimant (
  metamodel_beneficiary_id BIGINT NOT NULL,
  claimant_id BIGINT NOT NULL,
  metamodel_rightholder_id BIGINT NOT NULL
);
CREATE TABLE metamodel_actor_granter (
```

```
  metamodel_venue_id BIGINT NOT NULL,
  granter_id BIGINT NOT NULL,
  metamodel_cmo_id BIGINT NOT NULL
);
CREATE TABLE metamodel_actor_representation (
  metamodel_cmo_id BIGINT NOT NULL,
  representation_id BIGINT NOT NULL
);
CREATE TABLE metamodel_address (
  id BIGINT NOT NULL,
  city VARCHAR(255) NOT NULL,
  country VARCHAR(255) NOT NULL,
  number VARCHAR(255) NOT NULL,
  street VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_address_establishment (
  metamodel_address_id BIGINT NOT NULL,
  establishment_id BIGINT NOT NULL
);
CREATE TABLE metamodel_claim (
  id BIGINT NOT NULL,
  role VARCHAR(255) NOT NULL,
  delegator_id BIGINT,
  justification_id BIGINT NOT NULL,
  operator_id BIGINT,
  owner_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_creation (
  dtype VARCHAR(31) NOT NULL,
  id BIGINT NOT NULL,
  name VARCHAR(255),
```

```
  bpm INTEGER,
  finger_print VARCHAR(255),
  genre VARCHAR(255),
  isrc VARCHAR(255),
  language VARCHAR(255),
  iswc VARCHAR(255),
  claim_id BIGINT,
  play_id BIGINT,
  parent_id BIGINT,
  work_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_earning (
  id BIGINT NOT NULL,
  amount FLOAT NOT NULL,
  currency VARCHAR(255) NOT NULL,
  driver_id BIGINT,
  primary key (id)
);
CREATE TABLE metamodel_impact (
  dtype VARCHAR(31) NOT NULL,
  id BIGINT NOT NULL,
  amount FLOAT,
  currency VARCHAR(255),
  region VARCHAR(255),
  disease VARCHAR(255),
  metric_id BIGINT,
  primary key (id)
 );
CREATE TABLE metamodel_impact_metric (
  metamodel_societal_id BIGINT NOT NULL,
  metric_id BIGINT NOT NULL
);
```

```
CREATE TABLE metamodel_impact_statement (
  metamodel_therapeutical_id BIGINT NOT NULL,
  statement_id BIGINT NOT NULL
);
CREATE TABLE metamodel_license (id BIGINT NOT NULL,
  currency VARCHAR(255) NOT NULL,
  fee FLOAT NOT NULL,
  licensee_id BIGINT NOT NULL,
  licensor_id BIGINT NOT NULL,
   primary key (id)
);
CREATE TABLE metamodel_mandate (
  id BIGINT NOT NULL,
  country VARCHAR(255) NOT NULL,
  representor_id BIGINT NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_mandate_subject (
  metamodel_mandate_id BIGINT NOT NULL,
  subject_id BIGINT NOT NULL
);
CREATE TABLE metamodel_metric (
  id BIGINT NOT NULL,
  type VARCHAR(255),
  value VARCHAR(255),
  societal_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_play (
  id BIGINT NOT NULL,
  start DATE NOT NULL,
  stop DATE NOT NULL,
  consequence_id BIGINT,
```

```
  effect_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_playlist (
  id BIGINT NOT NULL,
  identifier_id BIGINT,
  instance_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_representation (
  id BIGINT NOT NULL,
  representee_id BIGINT,
  representer_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_representation_claimant (
  metamodel_representation_id BIGINT NOT NULL,
  claimant_id BIGINT NOT NULL
);
CREATE TABLE metamodel_statement (
  id BIGINT NOT NULL,
  statement VARCHAR(255),
  societal_id BIGINT,
  therapeutical_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE metamodel_work_in_recording (
  id BIGINT NOT NULL,
  language VARCHAR(255),
  recording_usage_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_router_music360object (
```

```
  id BIGINT NOT NULL,
  identifier VARCHAR(255) NOT NULL,
  type VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_router_music360object_owner (
  org_music360_router_music360object_id BIGINT NOT NULL,
  owner_id BIGINT NOT NULL
);
CREATE TABLE org_music360_router_provider (
  id BIGINT NOT NULL,
  identifier VARCHAR(255) NOT NULL,
  type VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_router_provider_asset (
  org_music360_router_provider_id BIGINT NOT NULL,
  asset_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_access_token (
  id BIGINT NOT NULL,
  expiration_date DATE,
  value VARCHAR(255),
  access_id BIGINT NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_users_group (
  id BIGINT NOT NULL,
  group_name VARCHAR(255) NOT NULL,
  parent_id BIGINT,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_users_group_child (
```

```
  org_music360_users_group_id BIGINT NOT NULL,
  child_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_group_member (
  org_music360_users_group_id BIGINT NOT NULL,
  member_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_organization (
  id BIGINT NOT NULL,
  organization_identifier VARCHAR(255),
  organization_name VARCHAR(255),
  type VARCHAR(255),
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_users_organization_manager (
  org_music360_users_organization_id BIGINT NOT NULL,
  manager_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_organization_user (
  org_music360_users_organization_id BIGINT NOT NULL,
  user_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_user (
  id BIGINT NOT NULL,
  enabled boolean,
  ipi VARCHAR(255),
  ipn VARCHAR(255),
  isni VARCHAR(255),
  other_id VARCHAR(255),
  user_name VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE org_music360_users_user_management (
```

```
  org_music360_users_user_id BIGINT NOT NULL,
  management_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_user_membership (
  org_music360_users_user_id BIGINT NOT NULL,
  membership_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_user_token (
  org_music360_users_user_id BIGINT NOT NULL,
  token_id BIGINT NOT NULL
);
CREATE TABLE org_music360_users_user_usage (
  org_music360_users_user_id BIGINT NOT NULL,
  usage_id BIGINT NOT NULL
);



ALTER TABLE metamodel_actor_claimant ADD CONSTRAINT
UK_q5dibsc322jon9vnpj4nr8kai UNIQUE (claimant_id);

ALTER TABLE metamodel_actor_granter ADD CONSTRAINT
UK_7yivmam4ta1dtck0knu5f4qwq UNIQUE (granter_id);

ALTER TABLE metamodel_actor_representation ADD CONSTRAINT
UK_ssqhi2qfisgc8oalsmiumi6cx UNIQUE (representation_id);

ALTER TABLE metamodel_address_establishment ADD CONSTRAINT
UK_jw2eu0cojtkp7n1jmv9p52os8 UNIQUE (establishment_id);

ALTER TABLE metamodel_impact_metric ADD CONSTRAINT
UK_javib27ed5grq15w7ws5uv0k5 UNIQUE (metric_id);

ALTER TABLE metamodel_impact_statement ADD CONSTRAINT
UK_sxule07kygsn3eubq9pob617h UNIQUE (statement_id);

ALTER TABLE metamodel_mandate_subject ADD CONSTRAINT
UK_eespbmsdg08dq8kgnhw2xoln3 UNIQUE (subject_id);

ALTER TABLE metamodel_representation_claimant ADD CONSTRAINT
UK_qukvpsw316htwowdi6494yxeu UNIQUE (claimant_id);

ALTER TABLE metamodel_actor ADD CONSTRAINT FK59s0g1dla8pt93x1e9lh0ww4n
FOREIGN KEY (stream_id) REFERENCES metamodel_playlist;
```

ALTER TABLE metamodel_actor ADD CONSTRAINT FKk4r3c6s66voakbsjwyn6dka48 FOREIGN KEY (location_id) REFERENCES metamodel_address;

ALTER TABLE metamodel_actor_claimant ADD CONSTRAINT FK94h3uykb5xotvf2jnkug4ed2f FOREIGN KEY (claimant_id) REFERENCES metamodel_claim;

ALTER TABLE metamodel_actor_claimant ADD CONSTRAINT FKq2kuwy12khgs5q5jkc4vuspcl FOREIGN KEY (metamodel_beneficiary_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_actor_claimant ADD CONSTRAINT FKn6sm039eml81rkduyu00qhu44 FOREIGN KEY (metamodel_rightholder_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_actor_granter ADD CONSTRAINT FKc72magm5ydvsdo32c5ahak6qj FOREIGN KEY (granter_id) REFERENCES metamodel_license;

ALTER TABLE metamodel_actor_granter ADD CONSTRAINT FKqme6wmnbnxe8xyhcoi543ij7b FOREIGN KEY (metamodel_venue_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_actor_granter ADD CONSTRAINT FKl6i6qhavj7sgebcd5o56e6ghb FOREIGN KEY (metamodel_cmo_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_actor_representation ADD CONSTRAINT FKh3tv6fp39j82hfutytqnjaivr FOREIGN KEY (representation_id) REFERENCES metamodel_mandate;

ALTER TABLE metamodel_actor_representation ADD CONSTRAINT FKe9jscqrix44xplw6l7d3wowgk FOREIGN KEY (metamodel_cmo_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_address_establishment ADD CONSTRAINT FKkn28rt43k0g0d4hnk7vp5arqx FOREIGN KEY (establishment_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_address_establishment ADD CONSTRAINT FKr453ptcein2b8sbdetqy11mex FOREIGN KEY (metamodel_address_id) REFERENCES metamodel_address;

ALTER TABLE metamodel_claim ADD CONSTRAINT FKglqfrhfl0q7qd3chbsv4mfyeq FOREIGN KEY (delegator_id) REFERENCES metamodel_representation;

ALTER TABLE metamodel_claim ADD CONSTRAINT FKshh4bwf8c8drp0shkcrvfal3c FOREIGN KEY (justification_id) REFERENCES metamodel_mandate;

ALTER TABLE metamodel_claim ADD CONSTRAINT FKfye284wo5joqg8h7wuawjas2m FOREIGN KEY (operator_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_claim ADD CONSTRAINT FK2w6sid05cj61pjmgydssxx4xx FOREIGN KEY (owner_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_creation ADD CONSTRAINT FKa9pofjjm7r492ghiqkyb949x1 FOREIGN KEY (claim_id) REFERENCES metamodel_claim;

ALTER TABLE metamodel_creation ADD CONSTRAINT FKeecxljb5d8ni2qpsyhmfbewr8 FOREIGN KEY (play_id) REFERENCES metamodel_play;

ALTER TABLE metamodel_creation ADD CONSTRAINT FKs3dhmuy37efag7bpsoc8u1x9m FOREIGN KEY (parent_id) REFERENCES metamodel_creation;

ALTER TABLE metamodel_creation ADD CONSTRAINT FK1291f1bwnlekw1icod3a534bj FOREIGN KEY (work_id) REFERENCES metamodel_work_in_recording;

ALTER TABLE metamodel_earning ADD CONSTRAINT FK6ep6t70g317of39gc1rsakt8n FOREIGN KEY (driver_id) REFERENCES metamodel_claim;

ALTER TABLE metamodel_impact ADD CONSTRAINT FK5flxpigo4we36v0c2kxog8pws FOREIGN KEY (metric_id) REFERENCES metamodel_metric;

ALTER TABLE metamodel_impact_metric ADD CONSTRAINT FKqhojynb6pwx8yodwigptnp1r9 FOREIGN KEY (metric_id) REFERENCES metamodel_metric;

ALTER TABLE metamodel_impact_metric ADD CONSTRAINT FKd053gamyxsx0fmwuna2wlwyou FOREIGN KEY (metamodel_societal_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_impact_statement ADD CONSTRAINT FKc9tv174q4q60rxs0fuo6wl5v4 FOREIGN KEY (statement_id) REFERENCES metamodel_statement;

ALTER TABLE metamodel_impact_statement ADD CONSTRAINT FK8i699t68l2pax4av55liaav8e FOREIGN KEY (metamodel_therapeutical_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_license ADD CONSTRAINT FKoh3ts8bna0p05a8qnw5tik9wa FOREIGN KEY (licensee_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_license ADD CONSTRAINT FKcwhq7g0yv8xrqp06ckua7hogj FOREIGN KEY (licensor_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_mandate ADD CONSTRAINT FK63d7s9vhfev2xc9ic12i6wxwt FOREIGN KEY (representor_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_mandate_subject ADD CONSTRAINT FKkudb16xf8494g6pds74f7fa0d FOREIGN KEY (subject_id) REFERENCES metamodel_claim;

ALTER TABLE metamodel_mandate_subject ADD CONSTRAINT FK3yp7ln2y8x0nlc0d9j8sxrlnq FOREIGN KEY (metamodel_mandate_id) REFERENCES metamodel_mandate;

ALTER TABLE metamodel_metric ADD CONSTRAINT FKat7xkbuqjukov2wjten3dhywf FOREIGN KEY (societal_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_play ADD CONSTRAINT FKlsc48tftu52mt1mgog6rwr2kt FOREIGN KEY (consequence_id) REFERENCES metamodel_earning;

ALTER TABLE metamodel_play ADD CONSTRAINT FKr4po20h89wrubl7ptp1kf46a0 FOREIGN KEY (effect_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_playlist ADD CONSTRAINT FKe2t5ix0w6f2mfo5reevj0w0ql FOREIGN KEY (identifier_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_playlist ADD CONSTRAINT FKpfg00hdrunbd6fjq9cmjsi2fh FOREIGN KEY (instance_id) REFERENCES metamodel_play;

ALTER TABLE metamodel_representation ADD CONSTRAINT FKhbc8lrwsra8jjaweixlunfc4v FOREIGN KEY (representee_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_representation ADD CONSTRAINT FKbvr4pps5wgxd37s4c81ccr2pi FOREIGN KEY (representer_id) REFERENCES metamodel_actor;

ALTER TABLE metamodel_representation_claimant ADD CONSTRAINT FK2i0r5ti4lte3w5cwbxn5srmpy FOREIGN KEY (claimant_id) REFERENCES metamodel_claim;

ALTER TABLE metamodel_representation_claimant ADD CONSTRAINT FKp5yvv8wfh1vxvwg0v232xmorp FOREIGN KEY (metamodel_representation_id) REFERENCES metamodel_representation;

ALTER TABLE metamodel_statement ADD CONSTRAINT FK502jykagnj3xlew7m00bo8ve0 FOREIGN KEY (societal_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_statement ADD CONSTRAINT FKpgkvmw15s9uey4qw6sq8o46g FOREIGN KEY (therapeutical_id) REFERENCES metamodel_impact;

ALTER TABLE metamodel_work_in_recording ADD CONSTRAINT FKc08df2xc2qombhue5b34f8e2h FOREIGN KEY (recording_usage_id) REFERENCES metamodel_creation;

ALTER TABLE org_music360_router_music360object_owner ADD CONSTRAINT FK9rvew12mj6y9fpwu3nuvnf0qd FOREIGN KEY (owner_id) REFERENCES org_music360_router_provider;

ALTER TABLE org_music360_router_music360object_owner ADD CONSTRAINT FKjuypkixm16oqqvhct2ls8l2s7 FOREIGN KEY (org_music360_router_music360object_id) REFERENCES org_music360_router_music360object;

ALTER TABLE org_music360_router_provider_asset ADD CONSTRAINT FKti481mht0yt7pkoa6u57l1b6y FOREIGN KEY (asset_id) REFERENCES org_music360_router_music360object;

ALTER TABLE org_music360_router_provider_asset ADD CONSTRAINT FKqejjtedtd2dx31hrbcb0l51wt FOREIGN KEY (org_music360_router_provider_id) REFERENCES org_music360_router_provider;

ALTER TABLE org_music360_users_access_token ADD CONSTRAINT FKru8chh8fxepcy5i6lmigk112a FOREIGN KEY (access_id) REFERENCES org_music360_users_user;

```
ALTER TABLE org_music360_users_group ADD CONSTRAINT
FK360h2f0e5vu0ugteuokp280jg FOREIGN KEY (parent_id) REFERENCES
org_music360_users_group;

ALTER TABLE org_music360_users_group_child ADD CONSTRAINT
FKjvf05dkkfo865i43pggq78n61 FOREIGN KEY (child_id) REFERENCES
org_music360_users_group;

ALTER TABLE org_music360_users_group_child ADD CONSTRAINT
FKek26qp3hqvvay7v7idh26edue FOREIGN KEY (org_music360_users_group_id)
REFERENCES org_music360_users_group;

ALTER TABLE org_music360_users_group_member ADD CONSTRAINT
FKmgvw70w54coyox3c42e4pns9 FOREIGN KEY (member_id) REFERENCES
org_music360_users_user;

ALTER TABLE org_music360_users_group_member ADD CONSTRAINT
FKjofyy657q8c9vwlj9xxbg4oj5 FOREIGN KEY (org_music360_users_group_id)
REFERENCES org_music360_users_group;

ALTER TABLE org_music360_users_organization_manager ADD CONSTRAINT
FK1ae75qrpjuryirike84ekxwr3 FOREIGN KEY (manager_id) REFERENCES
org_music360_users_user;

ALTER TABLE org_music360_users_organization_manager ADD CONSTRAINT
FKpce19c0g86wke0t60yqbw6hp FOREIGN KEY (org_music360_users_organization_id)
REFERENCES org_music360_users_organization;

ALTER TABLE org_music360_users_organization_user ADD CONSTRAINT
FK6x9111lapasqcbnckpmx0jkvt FOREIGN KEY (user_id) REFERENCES
org_music360_users_user;

ALTER TABLE org_music360_users_organization_user ADD CONSTRAINT
FK5s1fo9vvpavid587lxxjb608h FOREIGN KEY (org_music360_users_organization_id)
REFERENCES org_music360_users_organization;

ALTER TABLE org_music360_users_user_management ADD CONSTRAINT
FKq1gdhqxffrjc04jbl0be7po2p FOREIGN KEY (management_id) REFERENCES
org_music360_users_organization;

ALTER TABLE org_music360_users_user_management ADD CONSTRAINT
FKnpc6cdn3canafia493oer5ibo FOREIGN KEY (org_music360_users_user_id)
REFERENCES org_music360_users_user;

ALTER TABLE org_music360_users_user_membership ADD CONSTRAINT
FKho6ogs9tmbfi708wipl0mapw2 FOREIGN KEY (membership_id) REFERENCES
org_music360_users_group;

ALTER TABLE org_music360_users_user_membership ADD CONSTRAINT
FK6uyf7d1u54bdbtt41g5j3vxdg FOREIGN KEY (org_music360_users_user_id)
REFERENCES org_music360_users_user;

ALTER TABLE org_music360_users_user_token ADD CONSTRAINT
FKgxu6foe7dv71ntiyc73p9ji87 FOREIGN KEY (token_id) REFERENCES
org_music360_users_access_token;
```

```
ALTER TABLE org_music360_users_user_token ADD CONSTRAINT
FKha4pw0d00g99g4n89qt2nye7s FOREIGN KEY (org_music360_users_user_id)
REFERENCES org_music360_users_user;

ALTER TABLE org_music360_users_user_usage ADD CONSTRAINT
FKtqvyhgk1ltkk2p2uxi8kr2fym FOREIGN KEY (usage_id) REFERENCES
org_music360_users_organization;

ALTER TABLE org_music360_users_user_usage ADD CONSTRAINT
FKiy8qxq4l13ocjrmobtokmukl4 FOREIGN KEY (org_music360_users_user_id)
REFERENCES org_music360_users_user;

ALTER TABLE org_music360_users_group_child ADD CONSTRAINT
UK_ibi2ym9o34qpenpkqfekw1rs3 UNIQUE (child_id);

ALTER TABLE org_music360_users_user_token ADD CONSTRAINT
UK_638v3bn12apbn4ehqcx6swfhh UNIQUE (token_id);
```