

Music360

A 360 DEGREES PERSPECTIVE ON THE VALUE OF MUSIC



Deliverable 2.5

A distributed architecture for music data collection, representation, and distribution – version 2

	<h3>Disclaimer</h3> <p>The Music360 project has received funding from the European Union’s Horizon Europe research and innovation action under grant agreement number 101094872.</p>		
<p>The opinions expressed in this document reflects 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.</p>			
<h3>Version history</h3>			
Ver.	Date	Comments/Changes	Author/Reviewer
1.0	01/04/2025	Initial version	Giovanni Giachetti
1.5	05/05/2025	Updated with last Architecture implementations	Jesús Carreño
1.9	26/05/2025	Improvements from VU	Yulu Wang
2.0	28/08/2025	Final Version 2.0 Compiled	Giovanni Giachetti

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	T2.5a Designing and implementing a platform for data collection and representation
Deliverable	D2.5. A distributed architecture for music data collection, representation, and distribution – version 2

Due Date	31/05/2025	
Submission Date	30/05/2025	
Dissemination Level ¹	Public	
Deliverable Responsible	UPV	
Version	2.0	
Status	In Progress	
Author(s)	Giovanni Giachetti	UPV
	Jaap Gordijn	VU
	Oscar Pastor	UPV
	Roel Wieringa	TVE
	Gonçal Calvo	BMAT
	Jacobus Koen	BMAT
	Jesús Carreño	UPV
	David Nogales	BMAT
Reviewer(s)		
	Yulu Wang	VU

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.”

Content

1. Introduction	6
1.2. Main Changes from Version 1 to Version 2	7
2. Requirements	8
2.1. Non-Functional Requirements	8
2.1.1. Multi-party	8
2.1.2. Data owners remain in control with respect to their own data	9
2.1.3. Authentication and authorization	10
2.1.4. Data isolation	11
2.1.5. Open to future data providers and users.....	11
2.1.6. Scalable.....	11
2.1.7. Summary	12
3. Architecture Design.....	12
3.1. End-Users Data Visualization and Analysis Platform.....	13
3.2. Security Layer.	14
3.3. Data Loading Layer.....	14
3.4. Data Providers.	15
3.5. Importer API.....	16
3.6. Music360 Database.	16
3.7. Request API	18
4. Architecture Implementation	19
4.1. Bitbucket (Version control)	20

4.2.	Data Transformation Tools.....	21
4.3.	Importer API	21
4.4.	Private DB.....	25
4.5.	Resource Server.....	25
4.5.1.	Endpoints	25
4.5.2.	Local aggregations	28
4.5.3.	Technology	28
4.6.	Request API	29
4.6.1.	Endpoints	29
4.6.2.	Technology.....	29
5.	Architecture Execution and Demonstrator	30
5.1.	Data	31
5.2.	Importer execution	31
5.3.	Resource Server.....	32
5.4.	Demonstrators	34
6.	Security.....	36
6.1.	Multi-source authentication.....	37
6.2.	Access control mechanism	38
6.3.	Privacy preserving computation	38
7.	Conclusions	39

1. Introduction

This deliverable presents the design and implementation of the second version of the architecture that supports the Music360 platform. This version has been refined and improved to consider different efficiency aspects, enabling it to manage large amounts of distributed data from various stakeholders in the music value chain. The architectural design has been refined based on technical insights and stakeholder feedback gathered during the initial project phase, particularly through the first Living Labs round. Consequently, the new architecture provides adequate support for enriched data evaluation and visualisation, facilitating the measurement of monetary and non-monetary aspects of the music value chain.

Music360 Architecture V2 addresses the challenge of managing large-scale distributed data, ensuring that data owners — such as creators, collective management organisations (CMOs), venues and policymakers — retain control over their data. It recognises the diversity of stakeholders involved, including commercial entities, rights holders and public institutions, each of which has specific concerns regarding privacy, control and access to music data. To support this multi-party ecosystem, the platform architecture is based on decentralisation. Each data provider maintains their own partition of the distributed database, which is aligned with a shared data model. This approach ensures consistency across the platform while enabling each stakeholder to independently manage their data contributions, thereby ensuring privacy and security.

The design addresses relevant security and access control concerns by implementing robust authentication and authorization mechanisms based on open standards such as OpenID Connect and OAuth2. This approach supports federated identity management and decentralized authorization servers. This ensures that only authorized users can access or interact with specific datasets in line with the permissions granted by the data owners.

Scalability is another key consideration. The platform is designed to handle large volumes of data and an increasing number of users. Techniques such as data partitioning and replication enable efficient access to data and help to maintain performance under increasing demand. The architecture also anticipates expansion beyond the immediate project partners, promoting openness by adopting standard identifiers (e.g. ISRC and ISWC) and APIs designed for interoperability.

Furthermore, a significant enhancement in this version is the incorporation of advanced technologies for backend services and data integration. Using tools such as NestJS and PostgreSQL alongside modern development practices and version control via Bitbucket

contributes to improved performance, maintainability and responsiveness. The architecture also includes a well-defined API layer, a secure message bus for internal communications and modular services, including a router, aggregator, importer and resource servers. These components work together to enable seamless data exchange and integration between the various parts of the system.

The second version of the Music360 architecture also provides improved support for the front end of the Music360 platform. This makes the dashboard interfaces more responsive, facilitating the visualization, analysis and interpretation of collected data. The new architecture enables the dashboard to adapt to the needs of different users, ranging from creatives and venues to policymakers and researchers.

1.2. Main Changes from Version 1 to Version 2

Version 2 of the Music360 architecture introduces several key improvements over its predecessor that are mainly related to the following aspects:

- **Decentralized Data Management:** Unlike centralized systems, the architecture allows each data provider (e.g., CMOs, venues, music usage data providers) to manage its own partition of the database. This ensures that data owners maintain control over their data, while still enabling seamless access and interaction across different providers.
- **Scalability:** The system is designed to handle large volumes of data and users. It employs techniques like data partitioning and replication, which distribute data across multiple servers to improve performance, reliability, and scalability.
- **Enhanced Security and Access Control:** The platform uses robust authentication and authorization mechanisms based on OpenID Connect and OAuth2 protocols. These ensure secure access to data and services, with fine-grained access control that allows data to remain protected while being shared securely with trusted parties.
- **Updated Technology Stack:** Version 2 moves from Spring Boot to NestJS for improved data processing speed and code quality. The backend services are now more modular and optimized for performance, which facilitates faster response times and better integration with the platform's components.
- **Interoperability:** The architecture includes an API layer that supports standard data exchange formats and makes the platform easily integrable with external systems. This openness helps ensure that the platform can grow and include new stakeholders, especially as the platform moves towards being operational after EU funding.

- **User-Friendly Interfaces:** The architecture includes dashboards and data visualization tools, making it easy for users to explore and analyze the data collected on the platform, whether they are music creators, venues, or policymakers.

The rest of the document is organized as follows: The Requirements section provides functional and non-functional requirements, as well as a stakeholder analysis and prioritization using the MoSCoW methodology. The 'Data Model' section provides a general description of the conceptual model underpinning the data representation based on the Music360 Ontology V2, as presented in Deliverable D2.4. The Architecture Design section provides a detailed explanation of the platform components and how they interact, including the API, router, message bus and data providers. The Architecture Implementation section explains the technologies used, the repository structure, and the development methodology. The Architecture Execution and Demonstrator section demonstrates the setup and use of the implemented Music360 platform based on the second version of the architecture, using real-world examples with data from several countries. The Security section provides an overview of the authentication, authorization, and access control mechanisms, which are described in more detail in Deliverable 2.6. The 'Dashboard' section describes the visual interfaces for data consumption and analysis that will comprise Deliverable 3.2. Finally, the Conclusion section summarizes the new implemented architecture, highlighting its benefits and alignment with the project goals.

2. Requirements

This section presents several requirements involved in the design and implementation of Music360 Architecture. 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 second version of the architecture used for the Music360 platform implementation.

2.1. Non-Functional Requirements

2.1.1. 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 neighboring 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.
- Research teams: These teams carry out studies in specific settings with the aim of identifying the impact of music on the people and venues involved. These studies focus on measuring the economic and non-economic value of music. The Music360 living labs are included in this subset.

To properly support the different parties involved in the Music360 ecosystem, which ultimately interact with the developed platform, there are various technical aspects to consider. In particular, the privacy of the managed information and the proper connection between the different facets of the music value, which can be associated with different levels of measurement and context, must be guaranteed. These elements have been discussed in order to implement the music conceptual model proposed in Work Package 1 (WP1) and the Music360 Ontology presented in Deliverable 2.4 (D2.4), both of which are supported by the version 2 of the Music360 Architecture.

One important topic that is not discussed in this deliverable is the governance of the Music360 platform. However, this is highly relevant for a multi-party platform such as Music360 because there is no single authoritative decision-making entity. As this deliverable (D2.5) relates to the technical aspects of the Music360 architecture, the conceptual governance concerns are addressed in a separate deliverable (D5.1), 'Sustainable business model of Music360'. In order to address the governance challenges, we collaborated with SCAPR and CISAC, which are umbrella organizations for neighboring and authors' rights respectively.

2.1.2. Data owners remain in control with respect to their own data

We expect one of the critical success factors for the Music360 platform to be that all parties, particularly data owners, retain control over their data. Most parties have data that is, to a certain extent, private. Creative entities are concerned about data relating to their earnings and repertoire. CMOs have large, clustered data sets concerning

repertoire, line-ups and music usage, which would be of great interest to commercial entities for machine learning applications. Venues have data relating to the effect of music on customer behavior and, ultimately, sales. This data is therefore competitive and sensitive. All this data should be protected and controlled by the respective data owner. Most data usage concerns the ability to read data. Data providers create, update and delete data on behalf of the data owners, so they are in control by definition. We distinguish the following scenarios with respect to data accessibility, specifically reading data:

- Data is only available to users who have sufficient access rights. Data is stored in databases under the control of the data owner (e.g. physically hosted on the data provider's premises, which are trusted by the owner, or in a data center under the owner's control). Access to data should be restricted at least at the instance level (e.g. a recording or work right holder), and preferably at the property level (e.g. earnings or use of a recording). Deliverable D2.6, 'Secure and trusted sharing of music data – version 2', deals with access control in detail.
- Data owners may delegate access control management to others. For example, a creative entity may allow its representing CMO to make its data available under certain conditions. This data is often clustered and enriched with additional metadata. In this case, the CMO is responsible for storing the data and controlling access on behalf of the creative entity.
- Data owners can make data available in an anonymized, aggregated or derived format, thus enabling users to perform a set of predefined operations on it (e.g. arithmetic operations) without those trying to access it being able to read the detailed and private data. This approach allows data to be shared with untrusted parties while preventing disclosure and enabling controlled operations. This scenario is further developed in Deliverable 2.6, 'Secure and Trusted Sharing of Music Data – Version 2'.

2.1.3. Authentication and authorization

Authorization involves granting or preventing an authenticated user from performing operations (e.g. creation, reading, updating or deleting) on a resource, such as data or a service. In Music360, we assume that the data owner controls, is responsible for and keeps the access control. However, Music360 data owners rely on Music360's identity service, or that of a delegated party, for authentication. This establishes 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. In the second phase of the architecture, we also consider decentralized identity providers, whereby each data provider becomes an identity provider.

2.1.4. Data isolation

In terms of data ownership, the data of different stakeholders should be properly isolated. This means that data providers cannot see the data of other Music360 participants. Depending on the chosen data storage solution, specific measures are required. In other words, not all data will be stored in one database to which all data owners have full access.

2.1.5. Open to future data providers and users

Currently, access to the Music360 platform is restricted to EU Music360 project partners and any other parties they wish to grant access to (e.g. Living Labs participants in WP6). However, Music360 explicitly aims to continue operating after EU funding ends. For this reason, we consider Deliverable 5.1, 'Sustainable business model of Music360', to be of the utmost importance. Some (large) CMOs have already expressed interest in the Music360 platform and its ambition to better understand value. The same applies to one of our sister projects, OpenMuse, with which we have agreed to harmonize quantitative studies on the value of music.

Due to this interest and 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 open (de facto) 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.

2.1.6. Scalable

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

The vast majority of data operations are read-only, meaning that data enters the platform once and can be queried many times. This enables the large data set to be replicated and/or partitioned, thereby distributing the data across a large number of decentralized databases and hardware..

Data replication refers to the idea that the same data is available in multiple locations. Data partitioning means that different data is stored in different places. One example of data partitioning is when the data of one object (e.g. a recording) is fragmented across

multiple data providers. This ensures that the data owner remains in control of their data via a trusted data provider, as they own the relevant database partition and therefore control who has access to the data.

Both replication and partitioning distribute data across multiple software and hardware resources. This approach can be used to increase the amount of data that can be stored. Replicating the same data over multiple machines also enables scalability in terms of the number of users. However, the replicated data should preferably not be updated, which is characteristic of Music360 data: create data once and rarely update or delete it. If data is updated, only a few records are updated, so changes can quickly be propagated to the replicas.

2.1.7. 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. Architecture Design

The architecture defined for the Music360 platform is mainly a data presentation service. Most of the end users are oriented to query data related to works, recordings, their use, right holders, and right users (dashboard), or relates to the EU music system as a whole (ecosystem). The second version of the architecture implemented enhances the

system's ability to manage large-scale, distributed music data while ensuring that data remains under the control of the respective stakeholders.

The architecture is decentralised and implemented in a distributed way, which is a prerequisite for acceptance by most actors. For instance, a recording or musical work data can be replicate in different data sources, but the specific royalty management and payment is controlled by a specific CMO. The new architecture version also facilitates small data owners who do not have the facilities to implement and host their own data. In these cases, the data can be hosted 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”.

The Music360 platform architecture is outlined in Figure 1, the key components and their interactions are described below.

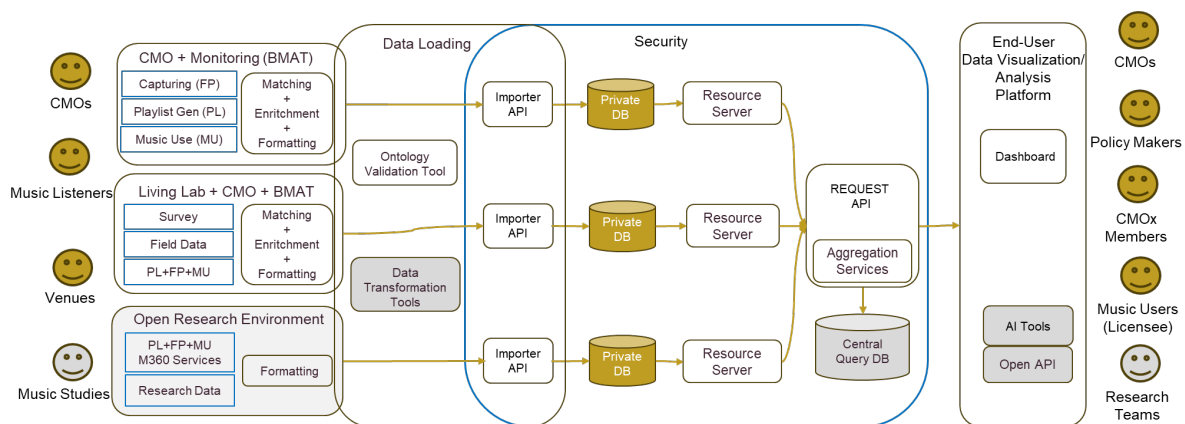


Figure 1 General Data Flow of Music360 Architecture Version 2 (in grey components under development)

3.1. End-Users Data Visualization and Analysis Platform.

The Music360 architecture provides a platform through which end users can visualise and analyse data. This corresponds to the interfaces through which stakeholders, such as music creators, venues, policymakers and other participants, interact with music-related data. The dashboard enables users to query and visualise data, offering insights into the value of music at various levels of aggregation (e.g. individual works, venues, and entire ecosystems). Depending on their user profile, users are granted differentiated access to and visualisation of data. The following user profiles are available: CMOs and their members (right holders); policymakers; music users; and research teams. To improve data management and access to data on the Music360 platform, the architecture incorporates two additional components. These are support for AI tools to facilitate the analysis of unstructured data from different studies (such as the Music360 living labs) and the implementation of an open API to extract raw information from the

Music360 platform for use in third-party analysis processes and tools. These last two components are being implemented as part of the Open Data Access Living Labs deliverable included in Work Package 6 (D6.7).

3.2. Security Layer.

The security layer is responsible for managing user authentication and authorization. This ensures that only authorized users can access the platform's resources. This layer is supported by a Security Manager and an Identity Server that use OpenID Connect and OAuth2 protocols, enabling secure login and the handling of roles such as rights holders, music users, and others.

Authentication will be decentralized to selected data providers in the Music360 project. Running different a authorization server depending on the data provider and end user. This would mean that other data providers should trust the data providers running also an authorization server. The authorization server will register the credentials of the users. Currently the access user roles that we manage are author right holder, neighbouring right holder and CMO.

After the user logged in, the authorization 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 APIs 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).

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).

3.3. Data Loading Layer.

This layer is the central dispatcher of requests from the front-end (user interface) to the correct data provider. It uses content-based routing, meaning it directs each request to

the appropriate data provider based on the type of data or resource being queried (e.g., a music track, artist, or venue). This ensures that the right data is retrieved from the right source.

This data loading layer supports the receiving of requests for data and dispatches these requests to the relevant data provider(s). It then receives the responses and forwards them to the front-ends via the request API. Requests are forwarded to the correct data provider(s) based on the content of the request, which can be any field, or an identifier such as IPI, IPN, ISRC or ISWC. This is similar to enterprise application integration and content-based routing specifically. For a more detailed example see: <https://www.enterpriseintegrationpatterns.com/patterns/messaging/ContentBasedRouter.html>.

3.4. Data Providers.

There are different kind of data providers: CMOs, music usage data providers (e.g., BMAT), providers of miscellaneous metadata (e.g., OpenMuse). Also music venues or music listeners (users of music) are relevant providers of data to evaluate 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). Some of these data providers are responsible for hosting, storing, and securing data.

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.

Each data provider is responsible for managing a partition of the Music360 data. To persist, load, and access data, a data provider consists of a private database, an importer, and a resource server. Usually, this partition corresponds to the data that data providers already have. Partitions may overlap, as the internal databases of data providers are not necessarily disjoint. It is important to understand that each partition has the same Music360 data model. Version 2 of Music360 will be based on Deliverable D2.4 (Music360 Ontology for the Value of Music – Version 2), Consequently, users of the data do not have to deal with differences in (semantics of) data models.

3.5. Importer API.

This component is responsible for loading data from external sources into the Music360 platform. Data providers use it to upload music-related data, including information on music tracks, usage, royalties and other metadata.

The importer API processes all input data from these providers and loads it into the Music360 platform. As the resource server, it publishes the required endpoints to perform import processes as a REST API. Access is restricted to the private network of the data provider.

Figure 2 shows the execution flow of an import job that can contain multiple import tasks. Each task corresponds to a play and creates the associated resources if they do not already exist. Despite a first validation of the input document being performed, the import can fail for any reason and must be monitored. To handle these situations, each task is wrapped in a transaction. If the import fails, the data associated with the task will not be imported, and an error will be logged. The data provider can then either try to import the affected rows again with corrections, skip the import job, or undo it.

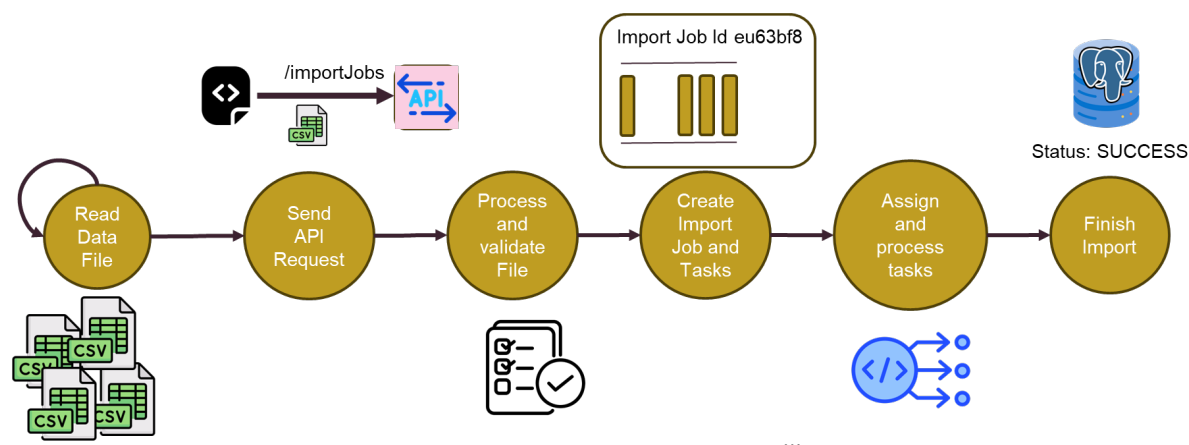


Figure 2 Importer API - Execution Flow for an Import Job

3.6. Music360 Database.

Music360 data is loaded from distributed, heterogeneous sources related to different data providers using the Importer API. Data owners and providers can remain in control of their data by acting as the administrative and technical responsible entity for it. The Music360 private database uses a consistent structure to store music-related data, based on the Music360 Ontology V2 reference (Deliverable D2.4). The database schema derived will be replicated across all Music360 database instances. Therefore, each data provider partitions and/or replicates the Music360 database. This means that the

information can be managed for analysis purposes without any structural or interoperability issues. This is implemented using PostgreSQL (see www.postgresql.org), a reliable, tried-and-tested open-source database management system.

The Music360 platform considers the different user roles and their interactions with each other and with the platform according to the following elements: Participation in generating relevant information to evaluate the monetary and non-monetary use of music creations; enriching this information with metadata to identify the applicability of music works in specific contexts and evaluate their positive or negative impact according to specific venues and uses; using the platform to manage royalties and authors' rights; and analysing the generated data to monitor proper royalty distribution and ensure transparent and adequate benefit assignment. In this context, Music360 user 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 those of an authorized identity provider, including CMOs. One person may have multiple roles, especially if they are both an author and an artist/producer. Distinguishing between these roles is essential for accurately handling metadata about music usage.
- Policymakers: These users access the platform to understand how music benefits are managed so they can make decisions about how to regulate their distribution according to the real value of music.
- Other users/systems: These users interact with the Music360 platform programmatically via a published REST API. These users can represent various entities, such as CMOs, venues, and fingerprinting companies.
- 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 3 presents the Music360 user data model that consider the following elements:

- A user can be a creative entity (everyone with an IPI and/or IPN), a policy maker, or music users, among others. For example, authoring right holders and neighbouring rightsholders.
- 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.

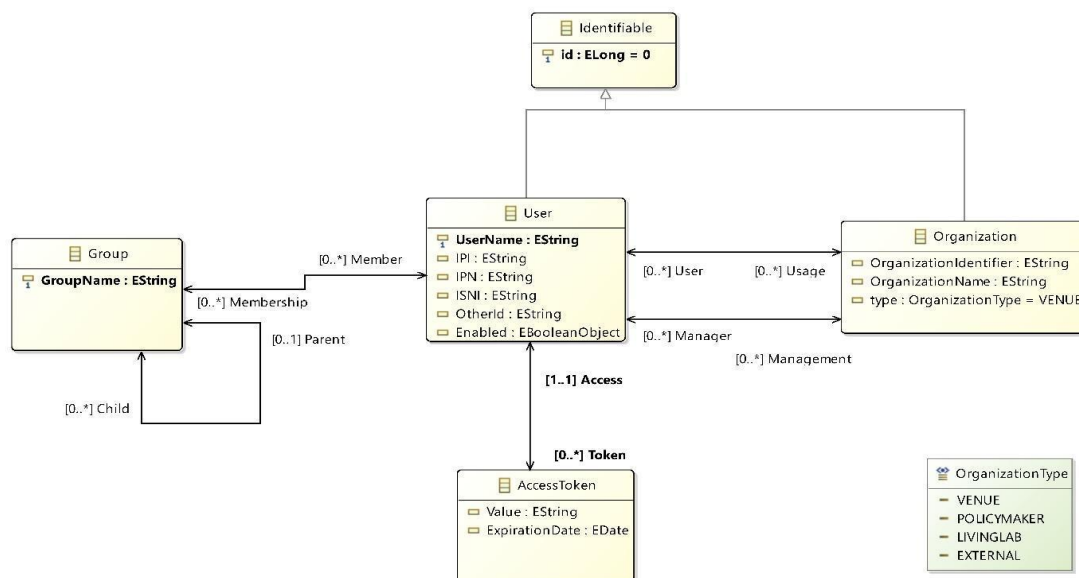


Figure 3 Music360 User Model

In terms of security, the Music360 database acts as a resource provider. This means that the database receives the token generated during user login and uses it to check access rights for specific data. Delegating access control to the database creates a uniform access control mechanism that executes precisely the same way for all dashboards.

To enhance management of large queries, a dedicated query database has been proposed as part of the open API facilities (D6.7). This database would collect and summarise information from various private Music360 databases provided by data providers. The query database would also support specific AI tools designed to improve the efficiency with which large amounts of data from different providers and countries can be managed.

3.7. Request API

The REQUEST API is the single entry point to the Music360 data that external users interact with to obtain insights. It publishes the endpoints required to perform both types of query: retrieving a list of records for a concept or an aggregation. Aggregation services combine data from multiple sources (data providers). When data from different providers needs to be combined into a single view — for example, aggregated statistics or metrics across multiple music rights organisations — the aggregator collects and consolidates

this information to provide comprehensive insights. Also, different filters and aggregation parameters are supported in the new architecture

Examples of API endpoints are the following (non exhaustive):

- **Get play information:** Obtain details about music usage, including played date, time, duration, and playlist associated, including earnings, claims, creation, venue involved. It can be filtered by any attribute available.
- **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 revenue:** The endpoint aggregates revenue by the parameters provided. Revenue refers to the total earnings generated from the claims.
- **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.

4. Architecture Implementation

This section outlines the implementation details of the Music360 architecture. Its components have been developed using up-to-date technologies such as NestJS and

PostgreSQL, in accordance with standards such as OpenAPI. These technologies and standards have produced excellent performance results (see Section 5).

The following components have been implemented to support the Music360 Data Flow from the data providers to the end-users:

- Importer API (Data provider)
- Music360 Private DB (Data provider)
- Resource Server (Data provider)
- Aggregation Services
- Request API

Dashboard implementation will be further explained in Deliverable 3.2.

4.1. Bitbucket (Version control)

The various project partners were responsible for developing component solutions based on a common architecture and specifications. It was imperative to merge the generated code into the same code repository in a controlled manner that could be reviewed, accepted, or rolled back as required. Therefore, the Bitbucket is used as the Git version control tool, which integrates with other tools, such as Jira, and facilitates collaboration between partners. For the Music360 Bitbucket project, we created five repositories, one for each architectural component. (see Figure 4):

- music360-api: It corresponds to the Music360 API. It consists of the aggregator / router.
- music360-auth-b and music360-auth-f: It implements the authorization server, frontend and backend respectively.
- music360-db: The code in this repository implements the resource server, the importer and the music database.
- music360-frontend: It corresponds to the code of the Dashboard, developed in Vue (see <https://vuejs.org/>).

BMAT Music Innovation / Projects / Music360

Repositories













Name	Size	Last updated	Builds
 music360-api	517.9 MB	2025-03-12	 
 music360-auth-b	739.1 KB	2025-01-08	
 music360-auth-f	779.3 KB	2024-12-04	
 music360-db	33.9 MB	6 days ago	 
 music360-frontend	10 MB	2025-01-29	

Figure 4 Music360 Code repositories

4.2. Data Transformation Tools

Background music playlists are harmonised by BMAT to make them compatible with music usage data. This is the first step in the data processing pipeline, after which the workflow remains the same. This pipeline is commonly known as ETL (extract, transform, load). The music monitoring service used is BMAT’s Vericast. Data from this music monitoring service is manually extracted from its API in the form of a CSV or Excel file. Once extracted, the raw data is matched against different metadata databases to add genre, language and BPM (beats per minute).

To obtain economic and music industry data, enriched reports are sent to the CMOs for further processing, matching and enrichment with their own data. Currently, reports for the tool are extracted and received manually. These reports include international codes for music recordings and works, right holder information, and corresponding revenues. Once this process is complete, the enriched reports are loaded to the Importer API (Section 4.3).

4.3. Importer API

The importer is a REST API that publishes endpoints, allowing the data provider to automatically and transparently upload their music data to its private database. It is imperative that the user, i.e. the data provider, has complete control over the data loading process. This means:

- **Check the status of the process.** Monitor whether the import is in progress, complete or failed in real time or on demand.
- **Error feedback:** Access clear, actionable error messages, including details of the cause and location of the issue in the data.

- **Roll back imported data if necessary.** Undo previously imported data to maintain consistency, especially after errors or misconfigurations.
- **Validate data format and consistency:** Data is automatically checked to ensure it meets the required format, schema and business rules before import.
- **Reimport partial data:** Select and re-upload only specific segments of the dataset, such as failed records or updated entries.

The implementation of this system has been carried out using the NestJS framework and the Sequelize ORM, with the purpose of synchronizing logical models and database tables. The response format is JSON. The importer defines the necessary resources to achieve these goals using the following concepts:

- **Import Job:** Its job is to load data from the input file to the database. Its properties are a unique identifier, the start time, and an error log. Its status can be one of the following: CREATED, ABORTED, PROCESSING_FILE, TASK_CREATED, SUCCESS or FINISHED.
- **ImportTask:** It is a task related to an import job. It is associated with a play. Its properties are an identifier, the row index, status, the total number of claims included, the import job ID and the ID of the imported play. Its status can be: PENDING, STARTED, SUCCESS or ERROR.

When the importer API receives an input file, an instance of the import job is created. First, the importer checks the data format. If it is valid, a task is created for each entry in the file. The atomic level of data loading is at task level, including earnings, claims and rightsholders. Each task is wrapped in a transaction, so if an error occurs, no private data will be imported. If the import process is successful, all tasks will finish in a 'SUCCESS' state, as will the import job. Any errors will be reflected in the import task and import job status. The log property in the import job will help users recover from errors. The data provider can then import a new file containing only valid data for the plays that were not imported.

The endpoints to performing all this actions are the following:

- **/importJobs (POST):** Create an import job. Input: data file and CMO id. If the request is successful, it returns the id of the import job created.
- **/importJobs (GET):** Get the information about a specific import job.
- **/importJobs/{importJobId}/tasks (GET):** Get all the tasks of an import job and their information.

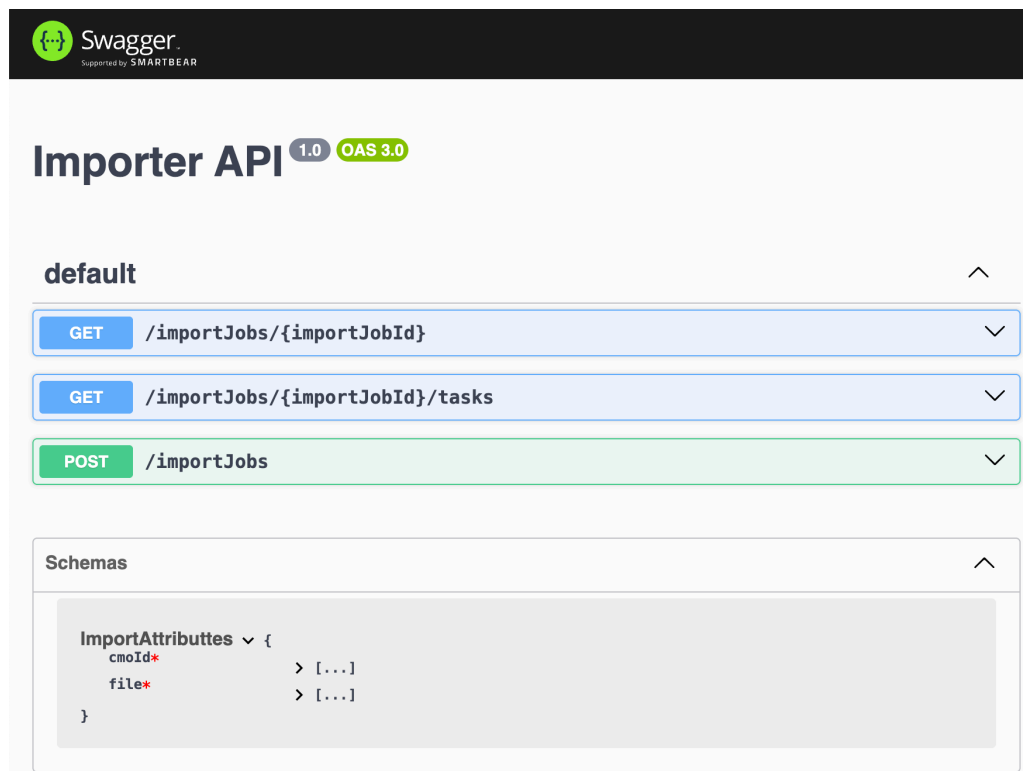


Figure 5 Importer API specification

It takes the enriched and formatted file as input, which was generated in the previous step of the data flow (Section 4.2). This file contains play-level data, including earnings claims for each involved right holder. The format and columns of each data row are shown Figure 6.

Some considerations about the input file are the following:

- **File format.** It must be a tab-separated value (TSV) file containing all columns, even if some are empty. All columns must be present and it must be tab-separated, even if some columns are empty.
- **Flexibility.** Some columns may be left empty to allow for cases in which the data provider is unable to enrich all the columns.

Order	Column	Format	Can be empty
0	date	yyyy-mm-dd hh:mm:ss yyyy-mm-dd	No
1	time	hh:mm:ss	No
2	duration	-	Yes
3	start_time	yyyy-mm-dd hh:mm:ss	No
4	stop_time	yyyy-mm-dd hh:mm:ss	No
5	channel	TEXT	Yes
6	city	TEXT	Yes
7	country	TEXT (within the accepted values, otherwise it will be mapped to "OTHER")	Yes
8	track	TEXT	Yes
9	artist	TEXT	Yes
10	label	TEXT	Yes
11	isrc	TEXT	No
12	bmatid	TEXT	Yes
13	iswc	TEXT	No (If Type of RH is "AUTHOR").
14	album	TEXT	Yes
15	language	TEXT (within the accepted values, otherwise it will be mapped to "OTHER")	Yes
16	genre	TEXT (within the accepted values, otherwise it will be mapped to "OTHER")	Yes
17	bpm	NUMBER	No
18	Main Artist	-	Yes
19	Title Track	-	Yes
20	Version	-	Yes
21	ISRC	-	Yes
22	VRDBID	TEXT	Yes
23	Title	-	Yes
24	ISWC	-	Yes
25	International Identifier of RH	TEXT	No
26	Right Holder Name	TEXT	Yes
27	Visibility to CMO	-	Yes
28	Type or RH	TEXT (within the accepted values, otherwise it will be mapped to "PERFORMER")	No
29	Value	NUMBER	No

Figure 6 Music360 Importer input file format

music360-db / data_model / importer / src / importJobs / taskConsumer

Name	Size	Last commit	Message
⬆️ ..			
📄 ClaimTaskHandler.ts	595 B	2024-12-02	New REST APIs: Importer & Resource server
📄 EarningTaskHandler.ts	752 B	2024-12-02	New REST APIs: Importer & Resource server
📄 PlayTaskHandler.ts	914 B	2024-12-02	New REST APIs: Importer & Resource server
📄 RecordingTaskHandler.ts	3.52 KB	2024-12-02	Bugs fixed
📄 RightHolderTaskHandler.ts	3.96 KB	2024-12-02	Importer bug fixed
📄 TaskConsumer.ts	9.05 KB	2024-12-02	Importer bug fixed
📄 VenueTaskHandler.ts	2.23 KB	2024-12-02	New REST APIs: Importer & Resource server

Figure 7 Task Consumer files

4.4. Private DB

The private database is a PostgreSQL instance that uses a relational data model. (

Figure 8) derived from the Music360 Ontology (See Deliverable 2.4. Music360 Ontology for the value of Music – Version 2). The imported data to data provider private database is accessible by the resource server. The server defines RLS policies that return only authorized data (More details in Section 6). See Music360 DB relational model in

Figure 8.

4.5. Resource Server

The Resource Server is one of the components of a Data Provider instance. It serves as the local API for accessing private data. It has been implemented to address the specified requirements and meet the needs of the users who will access the Music360 data. Technical choices of version 2 such as the new endpoints, response structure, chosen technologies and local aggregations enable integration into a scalable architecture and ensure fast responses.

4.5.1. Endpoints

The importer consists of a REST API which publishes endpoints that allow the end users to query the music360 data.

- **/plays:** It returns complete plays data, including earnings, claims, creation, playlist and venue. It can be filtered by any attribute available.
- **/revenue:** The endpoint aggregates revenue by the parameters provided. Revenue refers to the total earnings generated from the claims.
- **/stats:** The endpoint aggregates statistic data filtered by query parameters. It returns total artists involved, total cities, total countries, total genres, total languages, total plays, total revenue and total venues.
- **/tracks:** This endpoint aggregates plays data per venue and recording. For each group, it returns total plays, total revenue, recording metadata (title, artist, album, genre, bpm, language, ISRC, ISWC) and venue information (country, city, name, type), stakeholder name and identifiers (IPI, IPN).

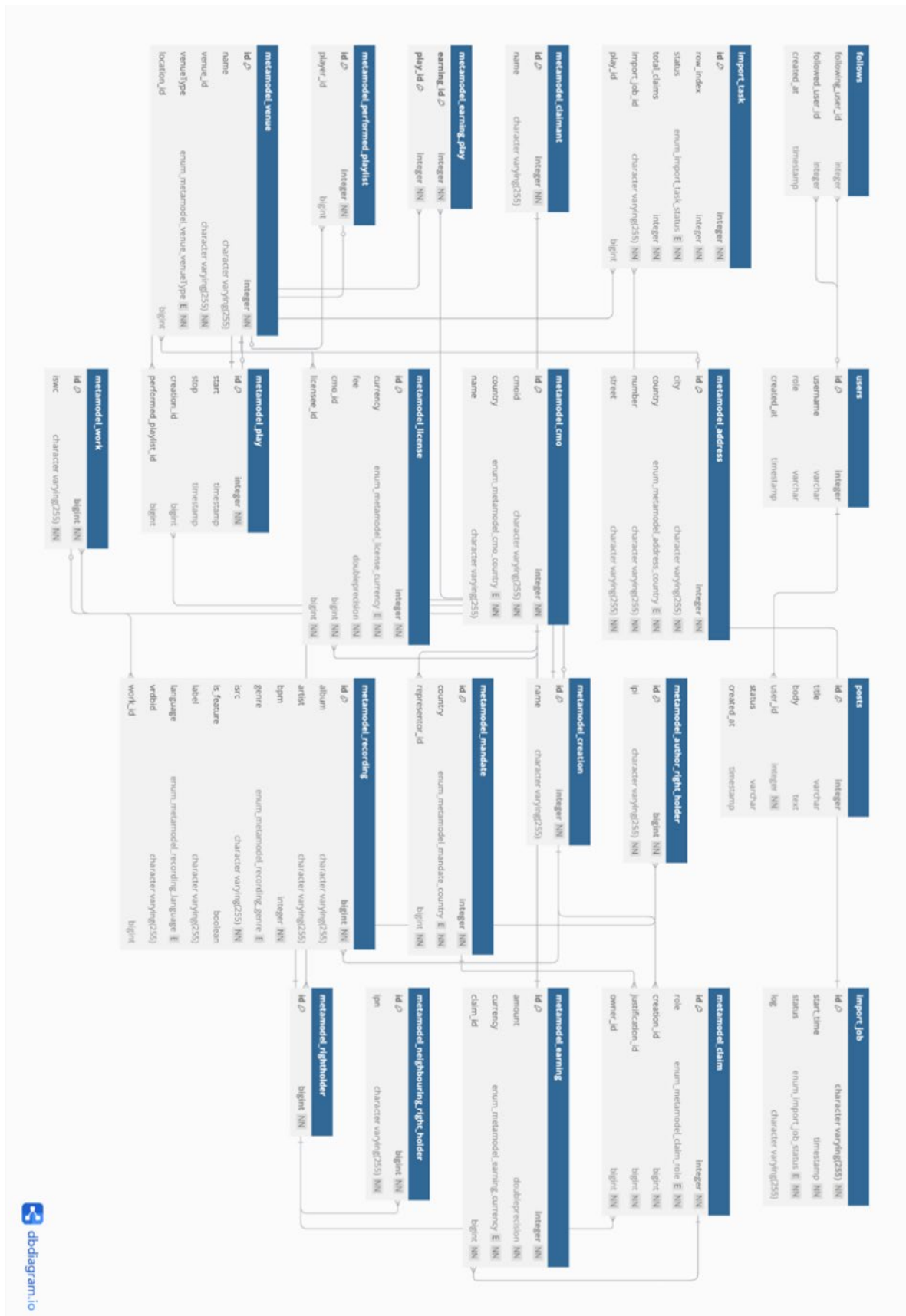


Figure 8 Music360 Relational Data Model

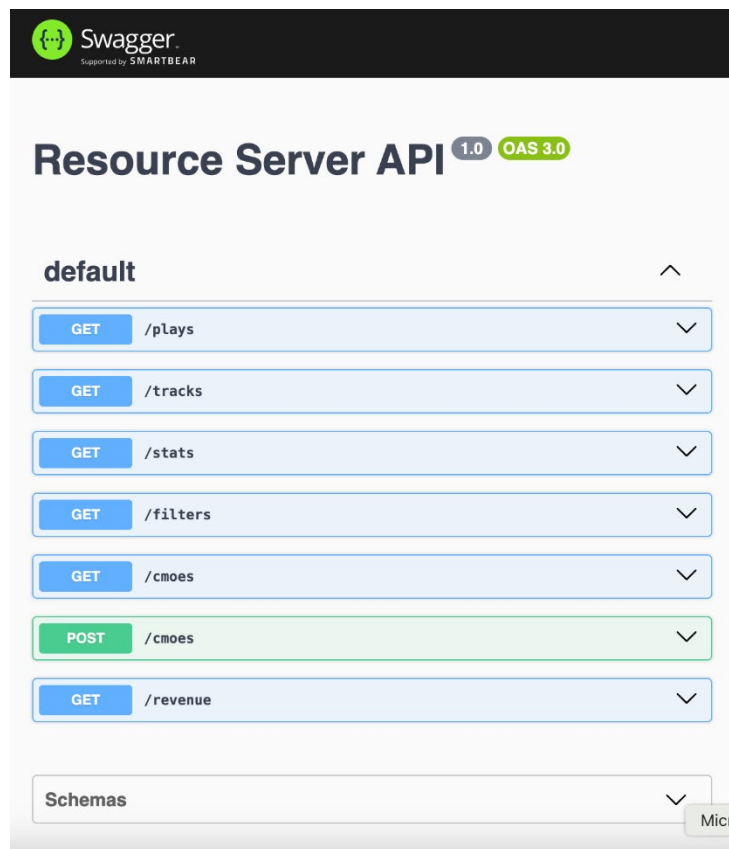


Figure 9 Resource Server API specification

These levels of aggregation are passed as query parameters in the URL along with the filters selected by the user (date range, country, type of venue, etc.).

Filter parameters include *title, iswc, isrc, artist, performer_ipn, performer_ipi, album, country, city, venue, venue_type, genre, label, is_feature, stakeholder_name*.

The client can aggregate data with a maximum of 2 properties. It includes *artist, album, genre, language, label, month, year, country, city, venue, venue_type*.

Regarding authorization, the resource server is in charge of decoding the JWT token passed by the first Music360 API. It extracts the information (e.g. Figure 10) and sets the RLS variable accordingly for each connection to the database, so the client can only access its own subset of data.

```

{
  "IPN": "61319773",
  "aud": "M360-Dashboard",
  "cmo_id": "CMO_SENA",
  "email": "burning@wolves.co.uk",
  "exp": "2026-03-18T11:37:20.875343171Z",
  "iat": "2025-03-18T11:37:20.875343171Z",
  "iss": "auth.SENA.nl",
  "nbf": "2025-03-18T11:37:20.875343171Z",
  "role": "rightholder_performer",
  "sub": "David Moreno"
}

```

Figure 10 JWT decoded token

4.5.2. Local aggregations

Data aggregation has been moved to the database level. This makes the calculations much more efficient and reduces response sizes significantly. Now, the aggregator is only responsible for meta-aggregating local aggregations. Thus, the version 2 includes one aggregation level more than version 2 (see Figure 11). These aggregations are defined in the resource server using SequelizeORM syntax.

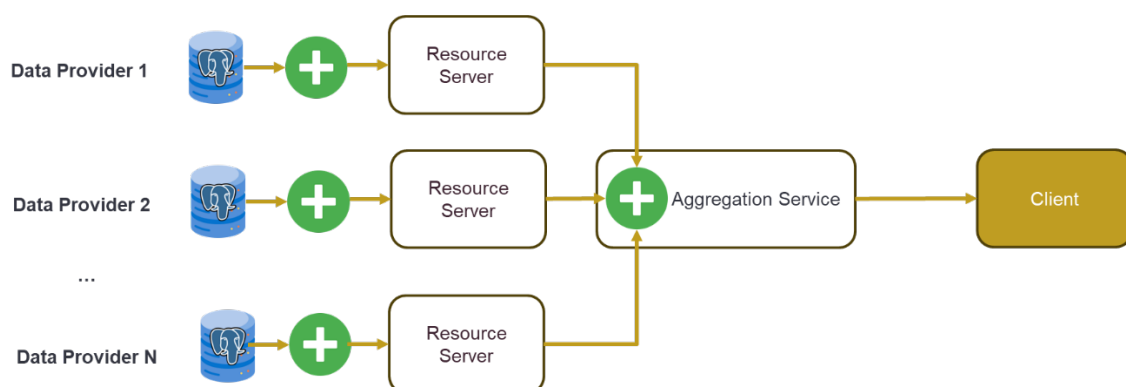


Figure 11 Music360 V2 aggregations levels

4.5.3. Technology

The main technology change from version 1 is the migrating from Java Spring to **NestJS**. This technology provides greater flexibility for performing complex queries on the

database, improved response times, better scalability in this distributed architecture, and easier deployment. Results are shown in Section 5.

NestJS includes Object Relational Mappers (ORM) that handles the communication and synchronization with the database tables. We have used **SequelizeORM**, which defines models and methods to operate with the registries in a developer friendly way. In addition, it adds another layer of field validation. A node instance of a resource server can be configured and replicated as many times as needed for scaling.

To facilitate collaboration and interoperability, the resource server supports JSON. 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

4.6. Request API

The Request API is the primary entry point to the Music360 API. It is central and independent of any data provider. The published endpoints are those that users can publicly target to query the data. It has been implemented with a focus on aggregation services and uses technologies capable of processing large volumes of data, performing transformations, and combining data from different sources.

4.6.1. Endpoints

Semantically, it implements the same endpoints and parameters as Resource Server. It meta-aggregates all local aggregations.

1. It receives a request with client-selected filters in the URL as query parameters and additional aggregation parameters.
2. The aggregator checks its cache, if available it returns the cached response. Otherwise, it forwards the request to the appropriate resource servers, including the user's JWT token and the query parameters that describe filtering and aggregation criteria.
3. The aggregator caches the response and proceeds to aggregate the data as per the aggregation parameters or endpoint type.
4. The aggregator sends a JSON response with aggregated data in a predefined format back to the client.

4.6.2. Technology

- **FastAPI:** Web framework for serving the API
- **Polars:** To aggregate the results obtained from the resource servers
- **Pydantic:** To validate the requests

5. Architecture Execution and Demonstrator

A complete architecture for validation with different stakeholders have been implemented in a demonstration environment. The Music360 demo has been mounted on **Docker** containers, which enable solutions to be run in an isolated and platform-independent way. Figure 12 shows all the containers running on the same host. The containers are the following:

- **data_model**: it is the container related to the data provider, which runs the resource server, the importer and the database.
- **music360-frontend**: it runs the dashboard of Music360.
- **sena**: it runs the SENA authorization server
- **buma**: it runs the BUMA authorization server
- **music360-api**: it runs the Music360 request API

To deploy Music360 Version 2 on production, docker run on **EC2** instances of Amazon Web Services (AWS). This version was delivered to the partner and has been used for the demonstrations. In total, 2 demonstrations have been performed in Portugal during a CMOs workshop organized by SCAPR, and in the EuroSonics conference in Netherlands.

The following sections show the execution architecture of each performer, following the data flow presented in this deliverable. Finally, we present the demonstration results.

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)
<input type="checkbox"/> ▾	data_model	-	-	-	2.95%
<input type="checkbox"/>	music360_importer	d8ffe811b374	data_model	3000:3000 ↗	0.63%
<input type="checkbox"/>	music360_resource_server	4f45177e06fe	data_model	8081:8080 ↗	0%
<input type="checkbox"/>	music360_db	be84f3646b85	postgres	5432:5432 ↗	1.76%
<input type="checkbox"/>	music360_redis	b3f9c62ad35c	redis	6379:6379 ↗	0.56%
<input type="checkbox"/> ▾	music360-api	-	-	-	0.89%
<input type="checkbox"/>	music360-aggregator-cache-service	fd55f2ebc9ce	redis:7.4.0	6390:6379 ↗	0.55%
<input type="checkbox"/>	music360-aggregator-server	f251be4231d8	music360-api	8080:80 ↗	0.34%
<input type="checkbox"/> >	music360-frontend	-	-	-	0.58%
<input type="checkbox"/> >	sena	-	-	-	0.26%
<input type="checkbox"/> >	buma	-	-	-	3.57%

Figure 12 Music360 running on Docker containers

5.1. Data

Because of confidentiality concerns, the data was anonymized. The total data generated ready to import resulted in four files:

- **MUSIC360_SCAPR2024_DEMO_ED SHEERAN_NL_anonymised.csv:** It includes SCAPR data from Netherlands
- **MUSIC360_SCAPR2024_DEMO_ED SHEERAN_IE_anonymised.csv:** It contains SCAPR data from Ireland.
- **in_GDA_PT_anonymised.csv:** It contains GDA (Gestao dos Direitos dos Artistas) data from Portugal.
- **in_GTM_FI_anonymised.csv:** It includes GTM (GT Musiikkiluvat) data from Finland.

In total, 9318 claims generated from more than 20 creations and 690 plays in 23 venues in 4 counties. It included 125 claimants, mostly performers. Figure 13 shows a preview of the file *MUSIC360_SCAPR2024_DEMO_ED SHEERAN_NL_anonymised.csv*.

	date	time	duration	start_time	stop_time	channel	city	country	track	artist	label	isrc	bmatid	iswc	album	la
1																
2	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
3	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
4	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
5	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
6	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
7	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
8	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
9	2024-03-14	00:00:00	14:49:22	00:02:58	2024-03-14 14:49:22	2024-03-14	14:49:22	2024-03-14	14:52:20	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
10	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
11	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
12	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
13	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
14	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
15	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
16	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
17	2024-03-13	00:00:00	08:53:18	00:02:58	2024-03-13 08:53:18	2024-03-13	08:53:18	2024-03-13	08:56:16	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
18	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
19	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
20	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
21	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
22	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			
23	2024-03-12	00:00:00	07:18:25	00:02:58	2024-03-12 07:18:25	2024-03-12	07:18:25	2024-03-12	07:21:23	Lonely Song Pavilion	Rotterdam	NL	Storm unde			

Figure 13 Data Provider example file

5.2. Importer execution

After BMAT and CMOs enriched the data, it was loaded into the data provider database using the importer. A script was developed and executed to automatically read all the files and send them to the importer API (Figure 14). The import process for four import jobs, corresponding to 690 imports and 9,318 claims, was successfully loaded into the Music360 DB tables in less than five seconds using a single CPU. This performance was made possible in part thanks to the direct integration between the importer and the database via Sequelize ORM.


```

curl -X POST -H 'Content-Type: application/json' -d '{"country":"PT","name":"GDA","cmoid":"GDA"}' http://localhost:8081/cmoe
curl -X POST -H 'Content-Type: application/json' -d '{"country":"FI","name":"GTM","cmoid":"GTM"}' http://localhost:8081/cmoe
( \
    cd ./reports_out; \
    curl -X 'POST' 'http://localhost:3000/importJobs' -H 'accept: */*' -H 'Content-Type: multipart/form-data' -F 'cmoid=GDA'
-F 'file=@in_GDA_PT_anonymised.csv;type=text/plain'; \
    curl -X 'POST' 'http://localhost:3000/importJobs' -H 'accept: */*' -H 'Content-Type: multipart/form-data' -F 'cmoid=GTM'
-F 'file=@in_GTM_FI_anonymised.csv;type=text/plain'; \
    curl -X 'POST' 'http://localhost:3000/importJobs' -H 'accept: */*' -H 'Content-Type: multipart/form-data' -F 'cmoid=GTM'
-F 'file=@MUSIC360_SCAPR2024_DEMO_ED_SHEERAN_IE_anonymised.csv;type=text/plain'; \
    curl -X 'POST' 'http://localhost:3000/importJobs' -H 'accept: */*' -H 'Content-Type: multipart/form-data' -F 'cmoid=GTM'
-F 'file=@MUSIC360_SCAPR2024_DEMO_ED_SHEERAN_NL_anonymised.csv;type=text/plain'; \
)
{"message":"Import job created","importJobId":"ca7f586af6596ab154c57cff8b7d60c8"}{"message":"Import job created","importJobId":"9523dd68
ebe5690c54d960ce58258b9f"}{"message":"Import job created","importJobId":"ed0109bf0680221d85041ae551ed0ec9"}{"message":"Import job create
d","importJobId":"a19f9e0f440dba82ea43c86f20a1e7c2"}

```

Figure 14 Importer API response after creating an import job

5.3. Resource Server

After populating the database with the CMOs data, it could be accessed through the resource server API. Figure 15, Figure 16, Figure 17 and Figure 18 show the API responses for the `/tracks`, `/stats`, `/revenue`, and `/plays` endpoints, respectively. The tokens used corresponded to multiple users with different roles, such as performer, composer, and CMO representative.

Thanks to database-level aggregations and a compact, information-efficient object structure, the responses were fast. Compared to Version 1, there has been significant improvement. Note that in Version 1, which was developed using Java Spring, response sizes were on the order of megabytes (see Figure 18).

The screenshot shows a web browser interface for the Music360 API. The URL bar shows `http://localhost:8081/tracks` with a `GET` method. The response is a JSON object with the following fields:

```

{
  "plays": "108",
  "revenue": "110.55662897499998",
  "title": "Dream beneath Star",
  "artist": "Burning Wolves",
  "album": null,
  "genre": "POP",
  "bpm": 100,
  "language": "EN",
  "isrc": "MHIYN7168473",
  "iswc": "T2477119367",
  "label": "Sacred Winds Music Co.",
  "country": "PT",
  "city": "Lisbon (Lisboa)",
  "venue": "Hidden Flame Lounge",
  "venue_type": "CATERING",
  "stakeholder_name": "DAVID MORENO",
  "performer": "61319773",
  "composer": null
}

```

The response status is `200 OK`, with a time of `337 ms` and a size of `26.88 KB`.

Figure 15 Tracks endpoint response (with time and size details)

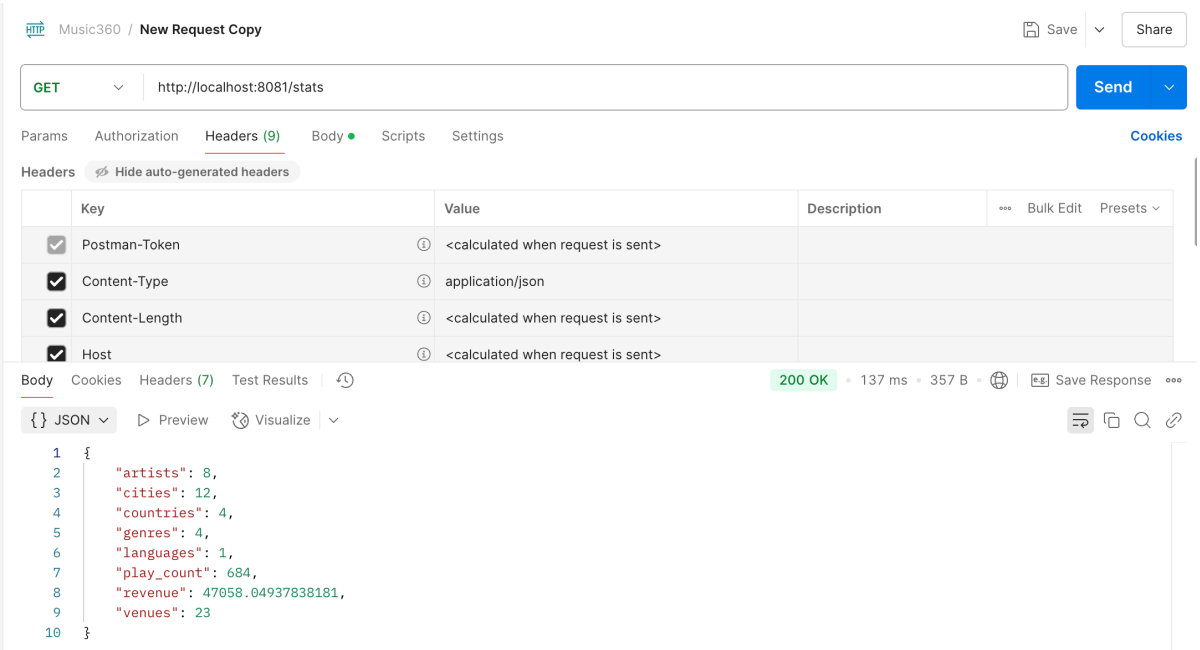


Figure 16 Stats endpoint response (with time and size details)

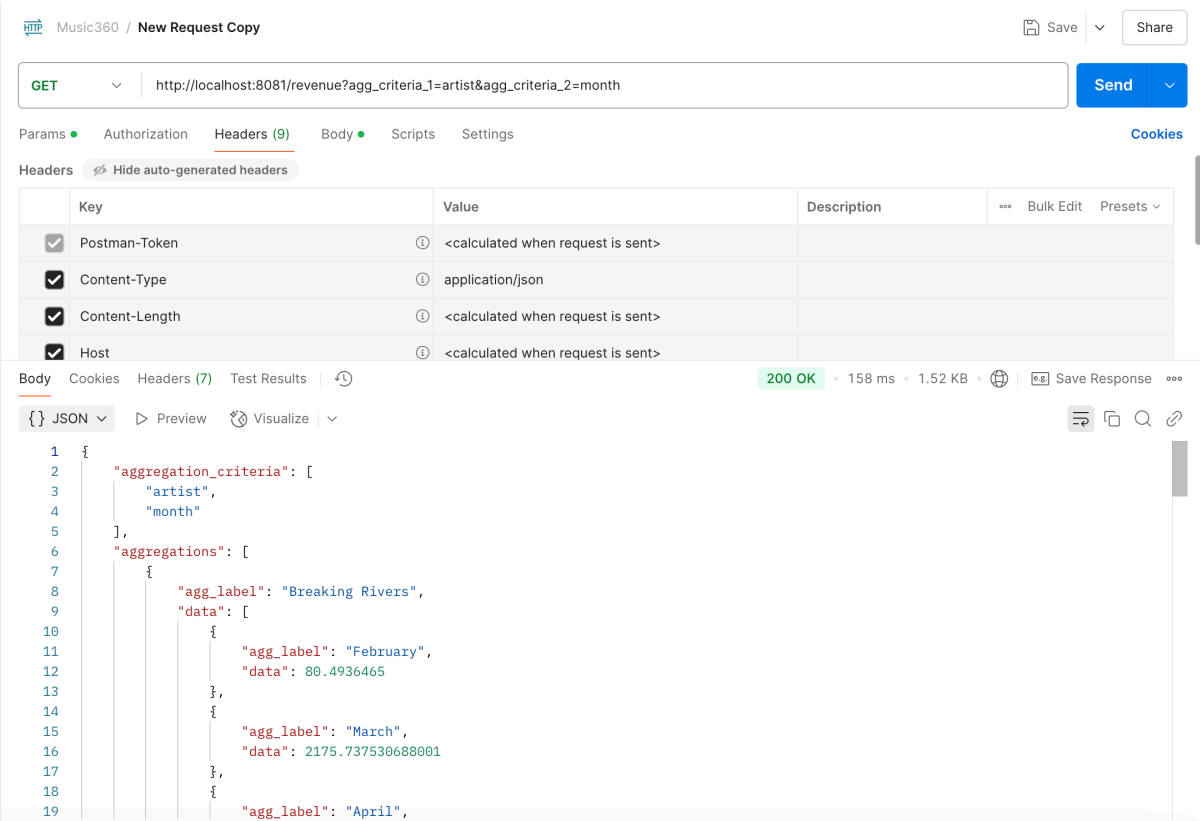


Figure 17 Revenue endpoint response (with time and size details)

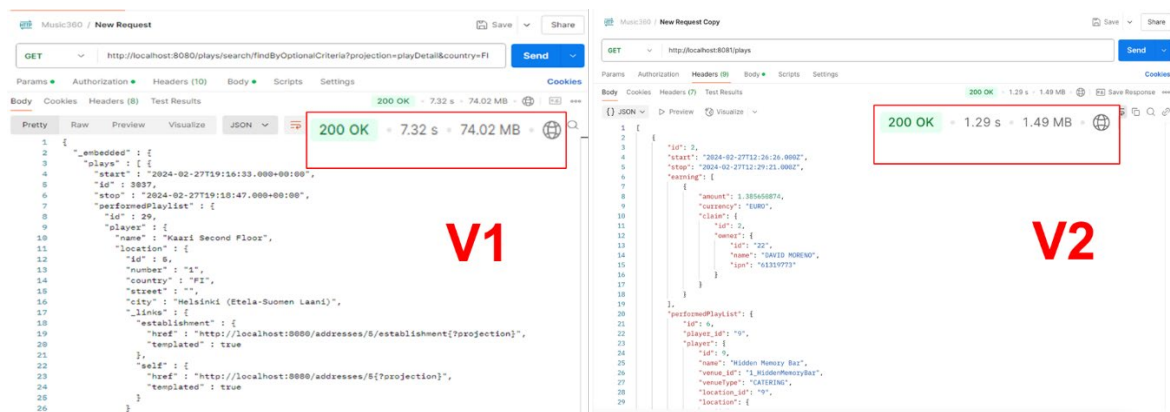


Figure 18 Plays endpoint response comparison Architecture V1 vs V2 (with time and size details)

5.4. Demonstrators

The Music360 demo has been tested in three different scenarios as a Performer User. (Figure 19 and Figure 20), a Composer user (Figure 21) and a CMO user (Figure 22 and Figure 23). In these demo scenarios, the dashboard successfully loaded all the required data and provided aggregated information according to the requested parameters. For example, it showed revenue by artist and city.

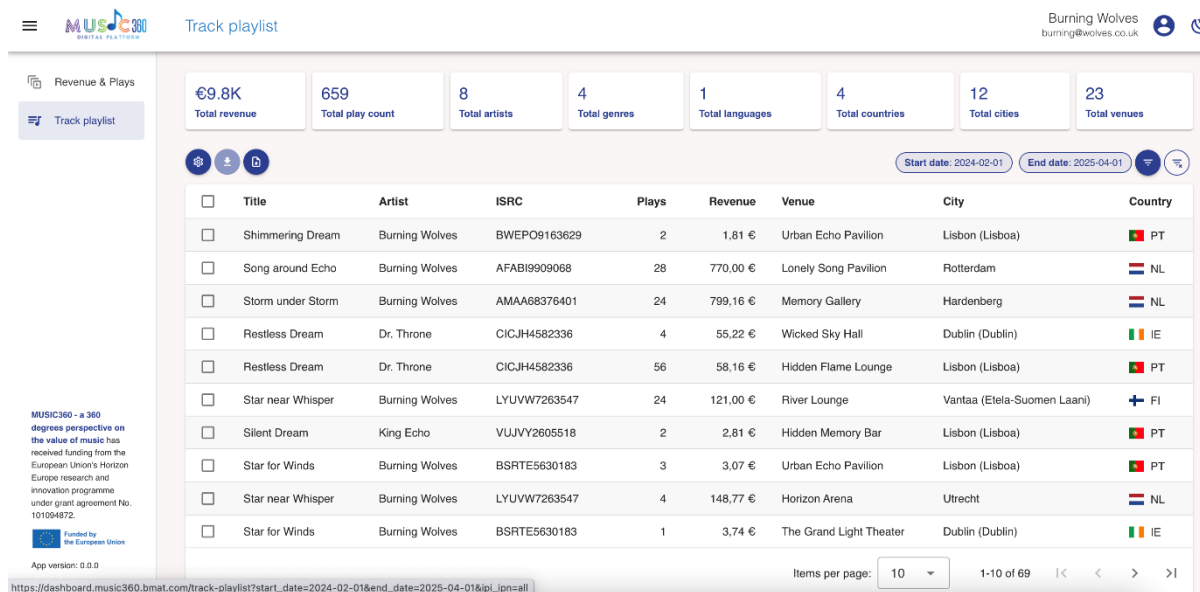


Figure 19 Revenue and plays view of Music360 Dashboard (Performer user)

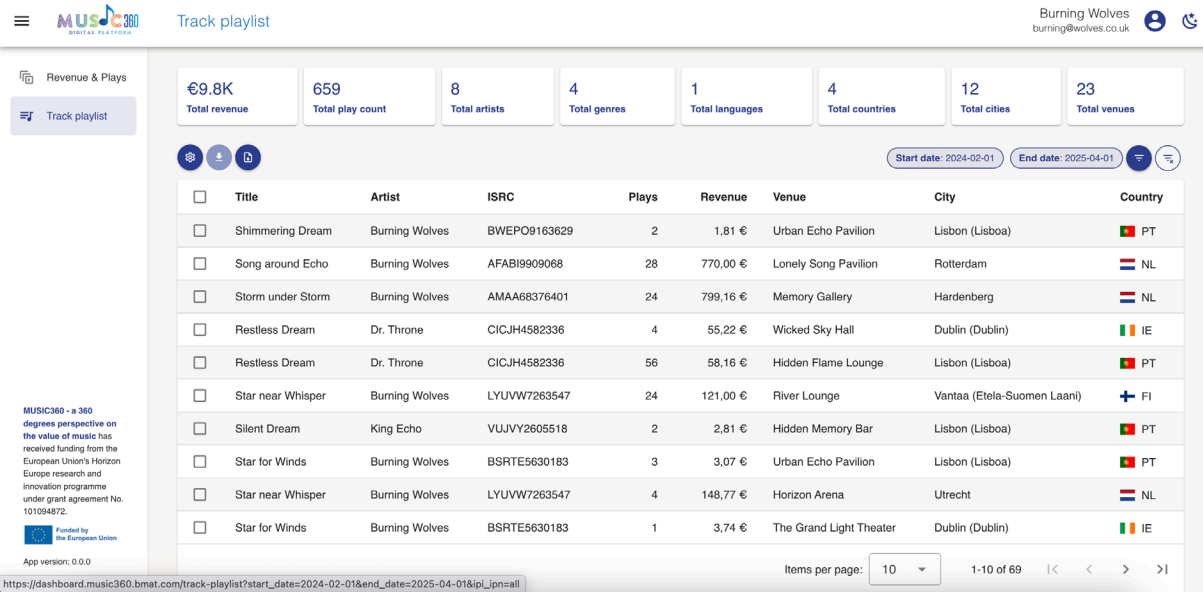


Figure 20 Track playlist view of Music360 Dashboard (Performer user)

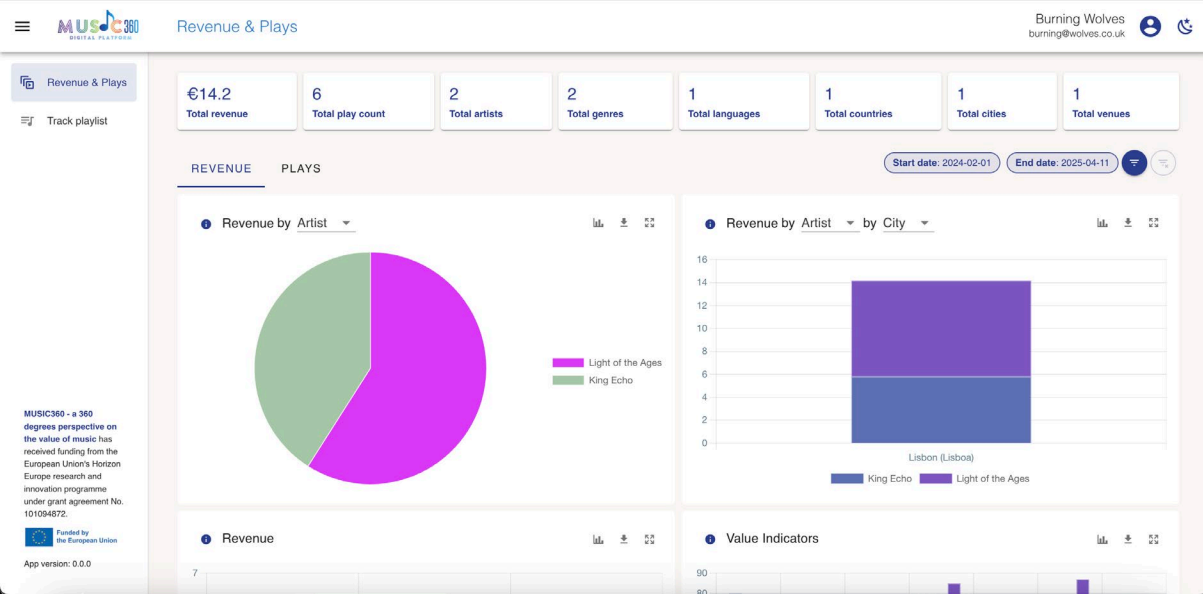


Figure 21 Revenue and plays view of Music360 Dashboard (Composer user)

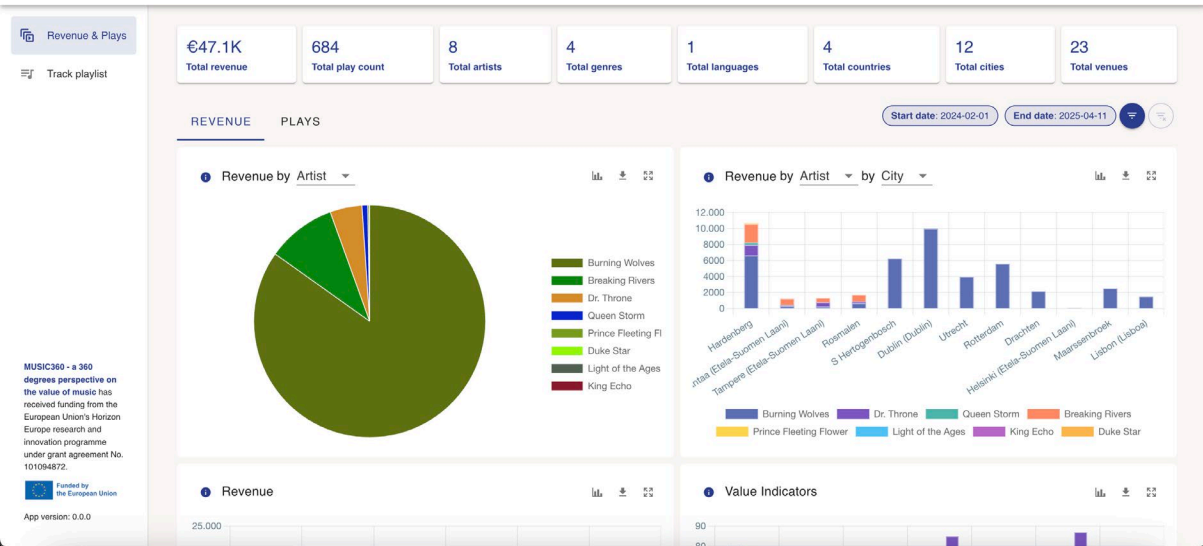


Figure 22 Revenue and plays view of Music360 Dashboard (CMO user)

Revenue & Plays									
Track playlist									
€47.1K Total revenue 684 Total play count 8 Total artists 4 Total genres 1 Total languages 4 Total countries 12 Total cities 23 Total venues									
Start date: 2024-02-01 End date: 2025-04-14									
<input type="checkbox"/>	Title	Artist	Composer (IPI)	Revenue	Plays	Performer (IPN)	Album	Genre	BPM
<input type="checkbox"/>	Song around Echo	Burning Wolves	-	641,12 €	20	61319773	Wave with Star	POP	100
<input type="checkbox"/>	Shimmering Dream	Burning Wolves	-	61,53 €	3	32917299	-	POP	100
<input type="checkbox"/>	Sacred Wave	Prince Fleeting Flower	-	6,10 €	2	81222382	Unplugged Radiant Horizon	-	100
<input type="checkbox"/>	Shimmering Dream	Burning Wolves	-	2,80 €	14	05013562	-	POP	100
<input type="checkbox"/>	Song around Echo	Burning Wolves	-	25,36 €	1	81885071	Wave with Star	POP	100
<input type="checkbox"/>	Shimmering Dream	Burning Wolves	-	41,88 €	3	58393912	-	POP	100
<input type="checkbox"/>	Shimmering Dream	Burning Wolves	-	2,75 €	14	39347812	-	POP	100
<input type="checkbox"/>	Haunting Heart	Breaking Rivers	-	24,45 €	125	79505558	-	POP	100
<input type="checkbox"/>	Restless Dream	Dr. Throne	-	16,60 €	1	47494628	Drifting for Flower	POP	100
<input type="checkbox"/>	Shimmering Dream	Burning Wolves	-	27,41 €	1	70209906	-	POP	100
				Items per page: 10 1-10 of 677					

Figure 23 Track playlist view of Music360 Dashboard (CMO user)

6. Security

Proper implementation of the Music360 architecture requires sharing data without giving users direct access to it. Rather, users should be permitted to perform specific, controlled operations on the data, the results of which may be visible to them. This section briefly describes the main security elements necessary for a complete overview of the architecture implementation. However, Deliverable D2.6, "Secure and Trusted Sharing of Music Data — Version 2," provides details about the Music360 security model and its implementation.

6.1. Multi-source authentication

A multi-source authentication mechanism, often referred to as federated identity management, enables users to access a system seamlessly using credentials from various sources by supporting multiple identity providers. The authentication flow is as follows: The user initiates the login process by selecting the preferred authentication provider on the Music360 login page via their browser. They are then redirected to the corresponding identity provider for user authentication (see Figure 24).

After successful authentication, the identity provider generates an authentication token and sends it back to the Music360 platform. The platform behaves as a resource provider (RP) that validates the received token with the corresponding identity provider to ensure its authenticity. Depending on the authenticated user, this token is then used to access data and services. These data and services are provided by resource providers (RPs). The login and security manager handles all of this. If it is the user's first time logging in, the Music360 platform creates a new user account based on the provided user details. Note that only the username needs to be stored, not the password, as the IP is responsible for storing the password. Otherwise, the platform updates the existing account with any new information from the identity provider.

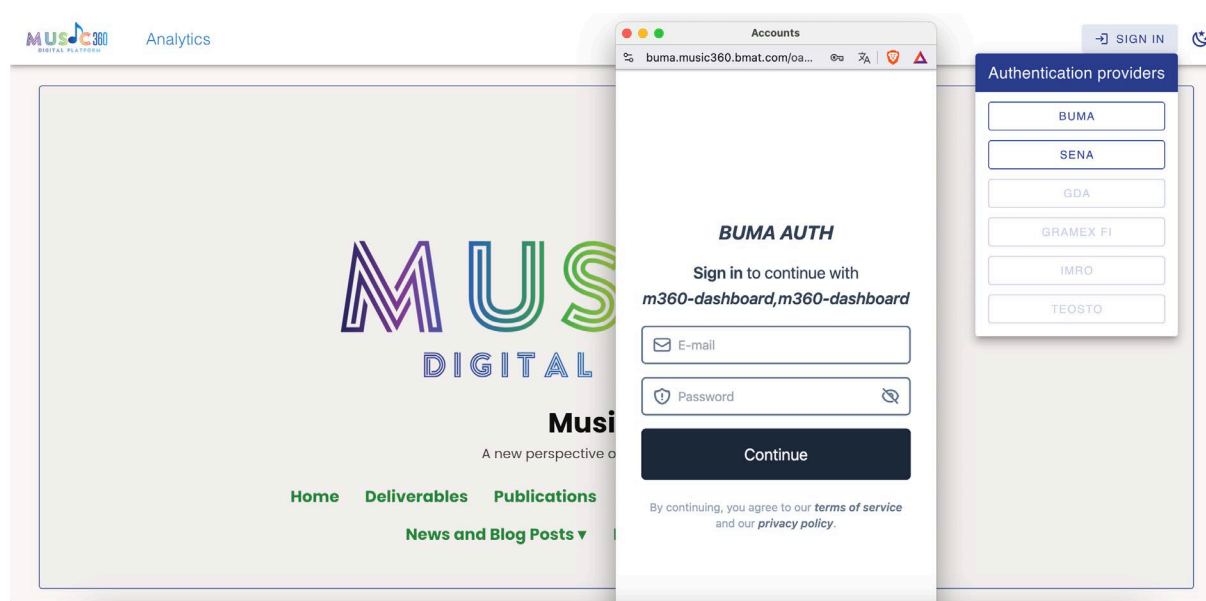


Figure 24 Music360 Platform Authentication Screen

Also, when a user logs in for the first time, they are prompted to log in to a neighbouring rights CMO and/or an author rights CMO. This connects the user's neighbouring right role to their author right role. This information is not currently available in music metadata. The Music360 platform's ability to harvest this information is an important contribution.

6.2. Access control mechanism

After proper authentication, users can access the platform via one of the dashboards. These components (dashboard, ecosystem analysis, etc.) act as resource providers. Based on the provided token representing the authenticated user, services and/or data are provided and access rights are checked. Access rights checking will be delegated as much as possible to the Music360 databases of the data providers. This ensures that data providers remain in control of their data. The only assumption is that data owners and providers trust the identity providers to reliably and correctly authenticate users.

Access control mechanisms in MUSIC360 are designed and implemented at the database level to address practical data security requirements, focusing on both Access Control Lists (ACLs) and Row-Level Security (RLS) in PostgreSQL. The ACL-based database security model establishes fine-grained permissions at the object level, governing read/write/delete operations through role-based assignments and permission inheritance mechanisms, thereby streamlining user privilege management. Concurrently, RLS enforces context-aware data filtering at the row level by leveraging session-specific variables (e.g., user IDs) to dynamically restrict query results, ensuring that users can only access data they are authorized to view. Implementation details include PostgreSQL-specific configurations (e.g., role/group definitions), and policy-driven RLS activation on targeted tables.

6.3 Privacy preserving computation

An important principle for Music360 platform data is that the owner of the data always maintains ultimate ownership. However, data contains important insights about the value of music that benefit stakeholders may wish to receive, and these data assets are often only accessible after some privacy data aggregation and computation. We apply the privacy preserving computation technology, more specifically, the secure multi-party computation to conduct some proof-of-concept experiments in specific security scenarios from MUSIC360 platform. This kind of innovative technology can act as a solution to achieve the propose that keeping the individual confidential data (from multi-party) as secrets but still provide overall insights about music value.

7. Conclusions

This deliverable presented the second, refined version of the Music360 architecture. Designed to efficiently manage distributed music data, it ensures data ownership, security, and scalability. The updated architecture incorporates decentralization, which allows data providers to maintain control over their own partitions within a shared model. This fosters trust and privacy in the multi-stakeholder ecosystem.

Key improvements include enhanced authentication and authorization mechanisms that utilize OpenID Connect and OAuth2 to ensure secure access and fine-grained data control. Scalability has been prioritized through data partitioning and replication techniques, enabling the platform to support a growing user base and increasing data volumes. Integrating advanced technologies, such as NestJS and PostgreSQL, optimizes system performance. An enriched API layer promotes interoperability with external systems.

Furthermore, the architecture provides enhanced dashboard interfaces that allow various stakeholders, including creators, venues, and policymakers, to effectively analyse music data. These interfaces facilitate better-informed decisions regarding the monetary and non-monetary aspects of the music value chain.

Looking ahead, Music360 is committed to evolving beyond the scope of the immediate project, encouraging broader adoption and integration with future stakeholders. This version of the architecture has a sustainable and scalable foundation that strengthens the platform's ability to provide meaningful insights and value to the music industry.