MusicoNet: a social network for musicians based on the Internet of Musical Things and People paradigm

Luca Turchet
Dept. Information Engineering and Computer Science
University of Trento
Trento, Italy
luca.turchet@unitn.it

Jacopo Tomelleri
Dept. Information Engineering and Computer Science
University of Trento
Trento, Italy
jacopo.tomelleri@studenti.unitn.it

Alessandro Ruffo
Dept. Information Engineering and Computer Science
University of Trento
Trento, Italy
alessandro.ruffo@studenti.unitn.it

2024 IEEE 5th International Symposium on the Internet of Sounds (IS2) | 979-8-3503-6652-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IS262782.2024.10704117

***Abstract*—To date, there are a number of online social networks dedicated to musicians. However, these do not truly leverage and capitalize on the musical diversity, i.e., the variability that exists across musicians, their instruments, musical activities and social relations. For instance, existing social networks are not designed to search and find for musicians with specific characteristics related to their profile and, above all, the actual particuliarities of their playing style. More importantly, their integration with the Internet of Musical Things (IoMusT) has been largely overlooked thus far. To bridge these gaps, in this paper we propose MusicoNet, a social network for musicians based on IoMusT technologies. The network leverages Semantic Web methods and is made accessible through an app for smartphones and tablet devices, which can wirelessly interact with smart musical instruments (or, through a laptop, with conventional instruments). MusicoNet was not conceived to exchange topic-oriented communication through textual and photographic posts as it occurs in popular social networks, but to support the search for and connectivity among musicians having given diversity factors. Such a search is not simply based on sole textual queries as it occurs in conventional social networks, but also on content-based queries which can be performed via musical instruments. We describe the technical implementation of MusicoNet, the IoMusT ecosystem it enables, and a preliminary technical validation. We then discuss the lessons learned and future avenues for the proposed technology, which represents the first instance of the recently proposed Internet of Musical Things and People paradigm.**

***Index Terms*—Online Social Networks, Semantic Web, Internet of Musical Things, Internet of Musical Things and People, Music Information Retrieval, Digital Ecosystems.**

I. Introduction

The *Internet of Musical Things* (IoMusT) relates to a vision that extends the Internet of Things (IoT) paradigm to the musical domain [1]. In such a vision, a set of physical objects dedicated to musical purposes (Musical Things) are interconnected via a networking infrastructure. Such objects can take the form of a smart device used to generate, control or track responses to musical content. These include smart musical instrument [2], which are self-contained instruments embedding the ability of processing music-related information and communicating wirelessly with external devices. Another example of Musical Things is represented by devices composing the hardware part of a Networked Music Performance (NMP) system. NMP systems aim at enabling geographically displaced musicians to play together over a wireless or wired network [3], [4].

Among the envisioned capabilities of Musical Things are those of being context-aware and proactive. These features are common in IoT devices, and usually are achieved by leveraging methods from the fields of Semantic Web (e.g., ontologies, automatic reasoning) and Recommender Systems. The capability of IoT devices (but also social networks) of being aware of environments or situations around their users enables the creation of networked services that can respond proactively based on such awareness. Machine learning methods are typically used to mine patterns from users' longitudinal data, which allows to make predictions about users' likely next steps and provide recommendations. Whereas the concepts of context-awareness and proactivity have been investigated extensively in computer science and IoT disciplines [5], they have been overlooked by music technology research, especially that related to music making. Thus far, the application of such concepts to the musical domain has been limited to the development of recommender systems for music listening [6], being this research fostered by big corporations dealing with music streaming (e.g., Spotify).

The interconnection of Musical Things may lead to the formation of IoMusT ecosystems that enable multidirectional communication between musical stakeholders as well as among them and musical resources or devices. In more detail, digital ecosystems form around some given technologies and communities of users utilizing them [7]–[9]. This holds true also for musical ecosystems [10]–[12]. An example is represented by online music repositories, such as Spotify or Deezer, which are based on music recommendation systems [13], [14]. However, communication in a streaming service is not bidirectional, as the user does not connect directly with artists or record labels. Another example of digital ecosystems is represented by online social networks. These enable the

formation of global connections that are related to a specific domain or based on a common need among the participants. To date, there are a number of social networks dedicated to musicians, such as Vampr[1], or services that allow musicians to share their productions and users to follow artists and commenting on posts (similar to social networks), such as SoundCloud[2]. However, these are not designed to search and find for musicians with specific (and deeply detailed) characteristics related to their profile and, above all, the actual particuliarities of their playing style (based on audio features). More importantly, their integration with the IoMusT paradigm has been largely overlooked thus far.

To bridge these gaps, in this paper we propose *MusicoNet*, a social network for musicians based on the Internet of Musical Things paradigm. The network leverages Semantic Web methods and is made accessible through an app for smartphones and tablet devices, which can wirelessly interact with smart musical instruments (or, through a laptop, with conventional instruments). MusicoNet was not conceived to exchange topic-oriented communication through textual and photographic posts as it occurs in popular social networks such as Facebook, but to support the search for and connectivity among musicians having given diversity factors, such as age, musical expertise, musical instrument(s) and genre(s) played. Such a search is not simply based on sole textual queries as it occurs in conventional social network, but also on content-based queries which can be performed via musical instruments (especially smart ones). Moreover, MusicoNet was devised to support musicians in finding venues where to conduct their musical activities as well as musical events to engage with.

The work reported in this paper falls in the remits of the *Internet of Musical Things and People* (IoMusTP) paradigm recently proposed in [15], and represents one of the first concrete technical steps towards the realization of such a vision. The move from the IoMusT to the IoMusTP is a move from a network of musical devices to a network of musical stakeholders, whose interactions with musical resources as well as other stakeholders are empowered by devices. In the IoMusTP, technology is not only aware of the users and their surrounding context, but is also compliant to ethical and sustainable principles that will make it possible more inclusive, personalized, and socially acceptable experiences for musical stakeholders. With MusicoNet we aim at introducing a technology that can empower richer and more inclusive social interactions through IoMusT ecosystems capitalizing on *musical diversity*. By musical diversity we refer to the variability that exists across musicians, their instruments, musical activities and social relations. Ultimately, MusicoNet focuses on connecting musicians for addressing their needs, such as finding the right musician(s) for a given task.

In the following sections, we describe the technical implementation of MusicoNet, the IoMusT ecosystem it enables, and a preliminary technical validation. We then discuss the lessons learned and future avenues for the proposed technology. The actual evaluation with users is a matter that will be addressed in a future research endeavor involving longitudinal studies.

II. Design

MusicoNet was designed to enable users to search for musicians via a smartphone application based on a classic client-server architecture. The app was conceived to support two types of queries: *i)* textual-based, and *ii)* content-based. In the former the user can perform a search by selecting some options from the app's interface, in the latter the user records a small excerpt from which audio features are extracted and then matched with those associated to music loaded in the profile of each user in the database. The app was thus designed to be capable of interacting with musical instruments, smart or conventional. Smart musical instruments can directly record an audio file and transmit it to the cloud where the audio file analysis is performed. Alternatively, a musician can use conventional instruments, but in tandem with a laptop and audio equipment to record and transmit the file.

Once a musician with desired characteristics is found, the user can interact with him/her (if online) by two styles of interaction: *i)* via a chat, *ii)* via a NMP system.

A schematic representation of the resulting IoMusT ecosystem is illustrated in Figure 1. Section III provides technical details about the implementation of MusicoNet.

III. Implementation

MusicoNet has been implemented following best practices in the fields of Semantic Web, programming and web design. Hereinafter we detail the main technological components underlying the proposed social network in its first version (v. 1.0.0), which are illustrated in Fig. 2. The code is made available open source[3].

A. The Musician's Context Ontology

At the heart of MusicoNet is the "Musician's Context Ontology" (MUSICO). The ontology and its documentation is publically available[4]. MUSICO was devised to represent the knowledge related to musicians and musical activities. Such an ontology was created to facilitate the development of context-aware musical applications. It enables the creation of profiles of musicians based on a large variety of diversity factors (demographics, role, level of musical expertise, instrument(s) and genre(s) played, etc.), as well as on the representation of their social relationships and participation to events.

MUSICO leverages various ontologies previously developed, including the Music Ontology [16], the Friend of a Friend Ontology [17], the Time and Timeline Ontologies [18], the Emotions Ontology [19], the Internet of Musical Things Ontology [20], the Smart Musical Instruments Ontology [21], and the Studio Ontology [22].

[1]https://vampr.me/
[2]https://soundcloud.com/
[3]https://github.com/CIMIL/MusicoNet
[4]https://w3id.org/musico#

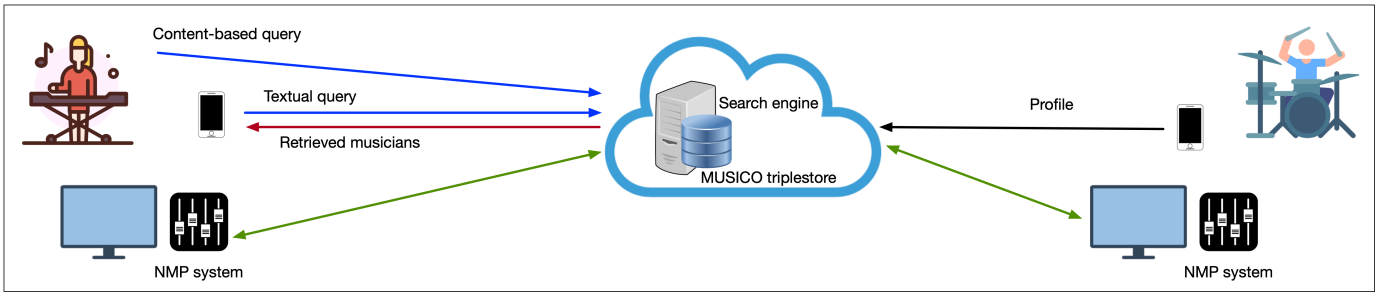

Fig. 1. A schematic representation of the developed IoMusT ecosystem.

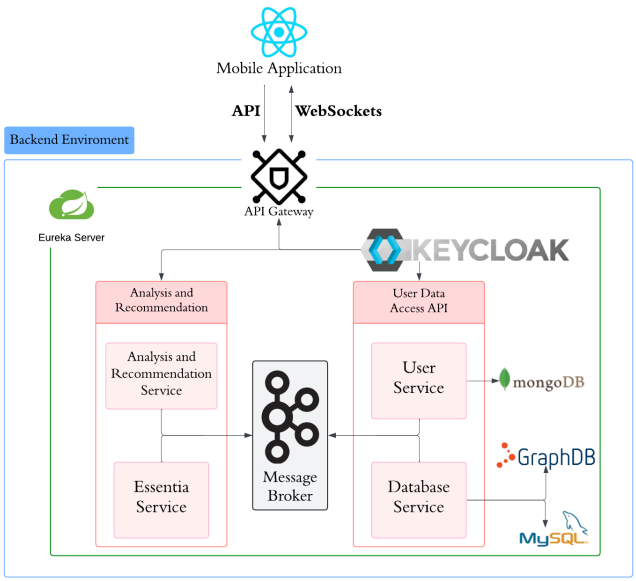

Fig. 2. A diagram of the MusicoNet's components.

B. Backend

The MusicoNet's backend structure is divided into three main macro components:

- **Identity and Access Manager:** third-party software used to manage user authentication and authorisation;
- **Databases:** three separate databases that manage all the data in the system;
- **Application:** the main structure of the system, developed according to a microservices architecture.

Identity and Access Manager

An Identity and Access Manager (IAM) is an essential tool for controlling user access to a system and managing their identities and privileges. Depending on the implementation, an IAM allows different types of users to access specific parts of the system. An IAM also manages the login process, providing services such as Single Sign-On and Multi-Factor Authentication.

For the implementation of MusicoNet, Keycloak[5] was chosen as the IAM. Keycloak is an open source project developed by the Cloud Native Computing Foundation that provides a simple way to manage and configure login methods and user permissions in the backend microservices. Users register and then log into the system via Keycloak, with the ability to use external social network accounts, giving them access to all components of the system for which they are authorized.

Communication between the user and Keycloak is mediated by the Gateway API (see Section III-B), which filters each request it receives and redirects it to the appropriate component. The Gateway API is then be responsible for receiving any call external to the application and, in the case of unauthorized users, redirects the call to Keycloak for login.

To ensure a secure login system, the OAuth 2.0 with PKCE (Proof Key for Code Exchange) authentication flow [23] has been implemented using Keycloak. This authentication flow adds an extra layer of security to the OAuth 2.0 standard, which is particularly effective for mobile or client-side applications. PKCE prevents the interception of credentials when exchanging messages with the backend, ensuring that only the application that generated the credential can use it, thus reducing the risk of man-in-the-middle and replay attacks.

Databases

Three different databases were utilized: *i)* a triple-store database built around the MUSICO ontology (see Section III-A), *ii)* a relational database, and *iii)* a NoSQL database. This triple database system was created to reduce the workload on the triple-store database as much as possible.

The triple store database plays a central role in the whole system. It stores all the relationships describing an artist's musical context, details of musical events, information about musical genres and other relevant data. Ad-hoc written inference rules are also added to the database to automatically infer data such as users who might know each other or events that might be of interest to them.

The relational database maintains the data related to the social aspect of the application, i.e., messages, account settings, user groups and friend request management. In addition, the relational database also contains information related to the user's profile, which is not relevant to the definition of the user's musical context. In some cases, the relational database is also used to more quickly obtain the IDs of the entities to be searched for in the graph database.

[5]https://www.keycloak.org/

The NoSQL database plays a dual role, storing the media uploaded by users and assisting the triple-store database in storing concepts. The media uploaded by users is broken down into smaller chunks and then reconstructed as needed, all of which is handled automatically by the database chosen for implementation. The support offered to the triple-store database is to contain complete data on unused concepts. A graph that is as light as possible is essential to ensure speed in the execution of queries and inferences. To achieve this, it was decided to load the data of concepts useful for describing the musical context of an artist only when they are actually used, taking them from the non-relational database.

Ontotext GraphDB, MySQL, MongoDB were chosen for the triple-store, relational, and NoSQL databases respectively. Ontotext GraphDB was chosen as the triple-store database because of its performance, full compatibility with RDF4J and, most importantly, the ability to add custom inference rules. The main reason for choosing MySQL and MongoDB was the ease of integration with the Spring framework[6], which was used to implement the microservice architecture. For these databases, Spring provides quick configuration for connection and various tools for development. In addition, MongoDB with the GridFS specification[7] also allows for storing media files.

Application

The microservice architecture represents an approach to application development based on the division of functionality into independent services communicating with each other via APIs. This development method allows for the creation of heterogeneous components with different languages or frameworks, thus enabling the use of the most appropriate technologies to create specific functionalities.

In our context, the decision was made to use this design pattern to mainly guarantee three characteristics considered essential for a social network:

- **Scalability**: The system can adapt to changes in the number of users and the resulting workload. This enables the system to maintain high performance and availability by replicating its services;
- **Maintainability**: The assignment of each backend task to a specific microservice facilitates the identification and resolution of any issues that may arise. This segregation of responsibilities enables the debugging, updating and management of the system;
- **Modularity**: The microservice architecture permits the incorporation of new functionalities through the creation of new microservices without disrupting the existing ones.

Furthermore, the implementation of a microservices architecture facilitates the continuous deployment and continuous integration processes, enhancing the efficiency of the development cycle and the quality of the released software.

The microservice architecture was implemented via the Spring framework. In particular, Spring Boot was used to create microservices, complete with the necessary dependencies. Each service that exposes APIs will have Spring Security[8] and Spring OAuth2 Resource Server as dependencies, so that the privileges of the user making the request can be controlled.

At the time of this writing, the system consists of six microservices. Communication between these takes place by exchanging messages via Apache Kafka[9], a distributed streaming platform that enables the publication, subscription, storage and processing of streams of records in real-time. Due to its ability to handle large volumes of data with low latency, Kafka guarantees reliable and efficient communication between microservices, allowing the system to be scalable and responsive.

Microservices can be divided into three categories: *i)* Interface management (i.e., Api Gateway and Eureka service), *ii)* Analysis and Recommendation (i.e., Essentia service and Analysis and Recommendation service), and *iii)* User data access API (i.e., User service and Database service). These are detailed hereinafter.

Interface Management. This pair of microservices has two fundamental roles: first, to identify microservices within the network via the Eureka Service; second, to manage communication with the external world via the Gateway API. The Eureka Service is responsible for monitoring and maintaining a record of the IP addresses of other microservices. To fulfil this task, Eureka creates a server to which the microservices connect. Consequently, Eureka is able to store and communicate the addresses of other services to the API Gateway, thereby facilitating the discovery and communication between microservices. In contrast, the API Gateway acts as a filter for requests from outside sources. Upon receipt of a request from a non-authenticated user, the PKCE authentication process is initiated, resulting in the exchange of messages with Keycloak. In the event that the user is already authenticated, the API Gateway, via the Eureka Service, forwards the received requests to the appropriate microservice.

Analysis and Recommendation. These microservices are responsible for analysing audio files and returning the results to the user. First, the Essentia service exposes an API through which users can upload audio files. The name of the service is derived from the Essentia library [24] used to create the service, an open source project dedicated to audio-based music information retrieval. Unlike the other microservices, since Essentia is a library available for Python, the microservice was also written in that language. Using this library, the service extracts details of musical characteristics (such as key, tempo, mood, or danceability) from the music piece in input. The results are then passed to both the Analysis and Recommendation Service and the Database Service.

The Analysis and Recommendation Service connects to the client via websocket and sends messages to it using the STOMP protocol[10]. Two communication channels, called

[6]https://spring.io/

[7]ttps://www.mongodb.com/docs/manual/core/gridfs/

[8]https://spring.io/projects/spring-security

[9]https://kafka.apache.org/

[10]https://stomp.github.io/

"queues", are established with the user. As results are received, they are returned to the user in the assigned queue.

In the database service, the results of the audio analysis are used to query the triple-store database. Since each user can store tracks that represent them in their profile, the service searches the database for users whose tracks match the search results. The resulting profiles are passed to the Analysis and Recommendation Service and returned to the user in a second dedicated queue.

In more detail, this service implements the user recommendation system. The recommendation is based on both the analysis of the audio file used for the query-by-content and on the contextual information provided by textual input. Concerning the recommendation based on the audio file analysis, this was inspired by our previous study reported in [25] which used a set of similarity metrics. Specifically, we used these metrics:

- BPM: the tracks whose BPM falls within 5% of the requested BPM are withheld, with preference given to the ones that are closest to the requested value;
- Mood: tracks with exactly the same mood as the requested one are selected;
- Key: tracks with exactly the same key as the requested one are selected;
- Danceability: the tracks whose danceability falls within 5% of the requested danceability are withheld, with preference given to the ones that are closest to the requested value;
- Tuning: tracks with exactly the same tuning (e.g., A = 432 Hz or A = 440 Hz) as the requested one are selected;

The final result then consists of the intersection of the these criteria, when these are selected by the user in his/her search preference. The system can return up to 10 tracks that meet the requested search criteria, provided enough tracks are available. These tracks are listed in descending order based on how well they match the criteria.

User data access API. The User service exposes the main APIs for retrieving, modifying or searching for data within the system. All the standard operations of searching for users or events, creating or modifying one's profile are provided by this service. This component also takes care of storing and retrieving the user's media from the NoSQL database. The User service relies on the Database service to store the data in the remaining databases. In general, a service dedicated solely to database communication is not a best practice. However, since in our case is necessary to query two different databases at the same time and reconstruct the data to merge them into a single object, creating a dedicated service proved to be the best choice.

To work with both databases, a system was created that could represent an entity present in both databases in a single class. When the service is started, the MUSICO ontology and the ontologies imported by it are loaded into memory. The RDF4J library[11] is used to parse the ontology files and create a model containing all of MUSICO's relationships and concepts. The model is made statically available to all classes in the microservice.

The Spring Data JPA library[12] was used to create classes annotated with the *Entity* tag, so that Spring would recognize them as tables in the relational database. Next, the *OntEntity* interface and *OntEntityField* annotation were created. The interface is implemented by the classes that represent entities in both the relational and triple-store databases, such as users. Similarly, the *OntEntityField* annotation is used to mark table fields present in both databases, specifying the predicate and relationship type in the graph.

Thanks to the combination of these two libraries and the Java Reflection feature[13], it was possible to create an automatic query generation system based on the instantiated object. By calling the method of the interface, it is possible to obtain the body of a query containing the data of the instantiated object or variables if the data is not available. The resulting query is then enriched with the operation to be performed (SELECT, INSERT, etc.) and sent via the endpoint to execute SPARQL queries against the triple-store database. The results are then combined with the data in the relational database, if necessary, and returned to the user service.

C. Frontend

For the development of the front-end we chose Expo, an open-source platform for making universal native apps for Android, iOS, and the web, for the following reasons:

- Cross-platform development;
- Streamlined workflows thanks to Expo toolset;
- Large active community and ecosystem;
- Native performance.

Key Libraries

Hereinafter, we list some of the key libraries that were used to enhance functionality and reduce development time:

- NativeWind: A utility-first CSS framework that uses Tailwind CSS as scripting language to create a universal style system for React Native also improving Developer UX and code maintainability;
- Expo Router: A file-system-based routing solution that allows to manage navigation between screens in the app, allowing users to move seamlessly between different parts of the app's user interface (UI);
- Expo Auth Session: Enables web browser-based authentication, like OAuth 2.0 in your app by utilizing WebBrowser and Crypto;
- Expo Document Picker: Allows the user access to the system's UI for selecting documents from the available providers on device;
- React Native Async Storage: Provides an asynchronous, unencrypted, persistent, key-value storage system for React Native;

[11] ttps://github.com/eclipse-rdf4j/rdf4j

[12] https://spring.io/projects/spring-data-jpa

[13] https://www.oracle.com/technical-resources/articles/java/javareflection.html

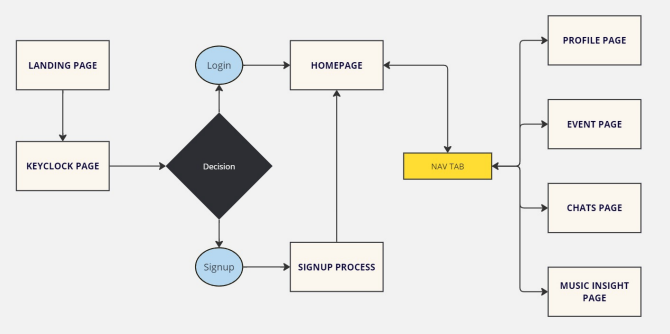

Fig. 3. User flow chart.

- React Native Elements: Cross-platform UI Toolkit for React Native providing pre-built, ready-to-use components that are fully configurable and strictly follow manufacturers' guidelines;
- Expo OSC: Allows the app to exchange messages and data with smart musical instruments through the Open Sound Control (OSC) protocol.

All these libraries are fundamental in forming the core of our application's front end, each of which was selected according to the unique needs of a musician-focused social network. Their combination makes it possible to build a solid, fast, and functional platform based on the users' requests.

User Flow

Figure 3 depicts a user flow chart that shows the general functionality of the app. The chart explains that the user begins on the landing page, which then redirects them to the Keyclock page loaded inside the in-app browser. Here, the user can choose to either login (see Fig. 4) or sign up (see Fig. 5). Signing up leads to the signup process, which then redirects the user to the homepage. Conversely, the login option immediately redirects the user to the homepage.

The homepage displays a list of suggested musicians (see Fig. 6). From there, the user is able to navigate to other sections of the application, including events, chats, profile, and music insights. In the music insights page, the user can either communicate to smart instruments through OSC messages or directly upload an audio file. The results of the audio analysis is displayed on the same page. Notably, each user profile may include the username identifying himself/herself on a NMP platform, so that the connection among users of such platforms is facilitated.

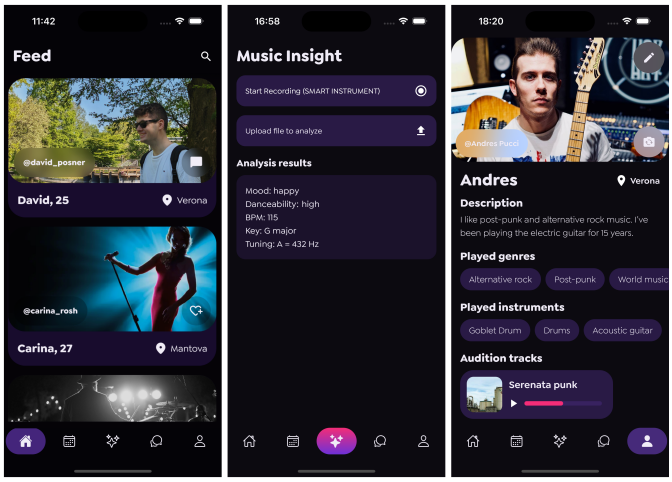

Fig. 6. App navigation.

IV. Ecosystem Description

Thanks to MusicoNet it is possible to create an IoMusT ecosystem composed of musicians and devices. We developed an ecosystem that leverage smart musical instruments as devices and preliminary tested its functioning. The ecosystem's components are detailed hereinafter.

Smart musical instruments. We used two prototypes of the smart instruments: the smart guitar described in [25] and the smart mandolin reported in [26]. Both of them are based on the Bela embedded system enhanced with a Wi-Fi dongle. The instrument enables to record a .wav audio file upon the wireless reception of start/stop messages delivered via the OSC protocol, and to submit it directly to the cloud server hosting the backend of MusicoNet. We opted to perform the analysis of the audio recording on the cloud rather than the embedded

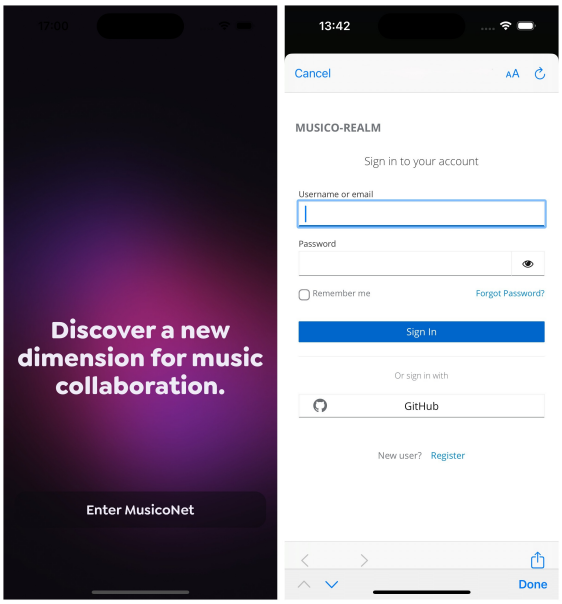

Fig. 4. Landing and login page.

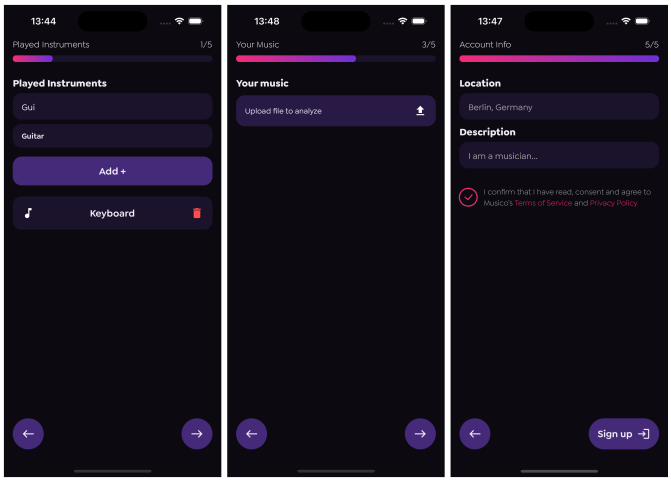

Fig. 5. Sign up process.

system because our previous measurements reported in [25] showed that this was much faster.

Smartphones. The MusicoNet app runs over a smartphone, and enables the user to wireless interact with the smart musical instrument to start/stop the recording, as well as to refine the search via textual queries and ultimately to connect to the desired musician (both via chat and via the NMP system).

NMP system. We used the Elk LIVE NMP system by Elk Audio [27]. Among the various NMP systems available as experimental prototype or commercial product (e.g., [28], [29], [30], [31], [32]), we selected Elk Live because its graphical user interfaces enables users to easily establish connections among themselves by simply adding a username.

V. Technical Validation

The app has undergone extensive technical validation through numerous tests, demonstrating its ability to perform searches within a triplestore containing 1.000 (fake) user profiles. Specifically, the technical validation phase of the system was carried out in three separate iterations.

The first iteration, conducted in parallel with the development phase, used Postman [14] to test each API as it was created. This software was used to generate requests to reproduce the conditions necessary to obtain all the implemented error codes. The tested conditions included: authentication and permissions, method and format of the request to the endpoint, connection verification and message exchange via websockets and file upload limits.

In the second iteration, automated tests were implemented for each microservice. These tests verified that the components of the service worked correctly, independently of the other parts of the backend. For instance, in the case of the Database service, the database connections and their data were simulated to ensure that the internal functions manipulated the data correctly before returning it to the user. Subsequently, the integration of the microservices was verified by checking the connection to the Eureka server, the message exchange with Apache Kafka, and the interfacing with Keycloak and the databases. Lastly, the front-end of the application was tested to ensure that the mobile application could correctly handle any API request errors and adapt to different scenarios.

The third iteration involved conducting a stress test on the system. To achieve this, 1000 profiles of fictitious users were inserted into the databases. The system's behavior, with particular focus on the triple-store database, was then observed during simulated user interactions with the application.

Additionally, a preliminary user study was conducted with two professional musicians using smart guitar and smart mandolin prototypes. For this evaluation, we created 50 (real) profiles of musicians who had participated in previous studies related to collecting music excerpts for an individual instrument dataset (see e.g., [33]). The results clearly showed that the musicians greatly appreciated the novelty and concept of the proposed system and were satisfied with the service provided while using it.

A more extensive evaluation campaign is currently underway, along with the development of a companion website to make the social network accessible to musicians using conventional musical instruments. However, additional equipment, such as a soundcard and a laptop, will be required in this case.

VI. Discussion and Conclusions

This paper described the technical implementation of MusicoNet, a novel social network based on the IoMusT paradigm, which was conceived to support musicians in finding other musicians with given characteristics. In part, our work parallels other previous efforts, which focused on the integration of social networks and the IoT [34], [35].

A novelty of the MusicoNet lies in its ability of supporting not just textual queries, but also content-based queries that can be performed by means of smart musical instruments. The proposed system represents a novelty in the landscape of social networks and platforms for musical activities, as it focuses on recommending musicians rather than music, thus shifting the focus from the listener to the player. Such a shift from music retrieval to musicians retrieval represents one of the first steps towards the accomplishment of the recent IoMusTP vision proposed in [15].

As with all other social networks, also MusicoNet users are exposed to a set of risks [36], such as privacy violations, identity theft, or unfair recommendations. Ethical concerns such as these are matter of debate in various music technology communities (e.g., NIME [37], Music Recommender Systems [38], [39]) and have recently made their inroads in the IoMusT community as well (see e.g., [40]). It is the precise goal of the authors to address satisfactorily any potential ethical concerns, so that MusicoNet is properly designed in the full interest of its beneficiaries.

Several avenues are possible for future work. The authors plan to continue the development of MusicoNet, with a long-term goal of releasing the app for public use. Before this is possible, MusicoNet needs some core functionality added

[14]https://www.postman.com/

to it and the interface needs to be modified to be more accessible to musicians. First of all, before widespread release, the application needs some more formal privacy protection in place. These and other aspects will be fine-tuned by directly involving the end-users in the design process, leveraging participatory design methodologies [41], [42]. Furthermore, we plan to conduct an extensive evaluation campaign with several musicians in order to assess the musicians' experiences in interacting with it. The authors believe that MusicoNet has strong potential to be an empowering, musician-friendly social network, and we look forward to continuing with future releases.

REFERENCES

[1] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of Musical Things: Vision and Challenges," *IEEE Access*, vol. 6, pp. 61 994–62 017, 2018.

[2] L. Turchet, "Smart Musical Instruments: vision, design principles, and future directions," *IEEE Access*, vol. 7, pp. 8944–8963, 2019.

[3] L. Gabrielli and S. Squartini, *Wireless Networked Music Performance*. Springer, 2016.

[4] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[6] P. Knees, M. Schedl, B. Ferwerda, and A. Laplante, "User awareness in music recommender systems," *Personalized Human-Computer Interaction*, p. 223, 2019.

[7] H. Boley and E. Chang, "Digital ecosystems: Principles and semantics," in *Inaugural IEEE International Conference on Digital Ecosystems and Technologies*, 2007, pp. 398–403.

[8] G. Briscoe, S. Sadedin, and P. De Wilde, "Digital ecosystems: Ecosystem-oriented architectures," *Natural Computing*, vol. 10, pp. 1143–1194, 2011.

[9] O. Mazhelis, E. Luoma, and H. Warma, "Defining an internet-of-things ecosystem," in *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2012, pp. 1–14.

[10] A. Di Scipio, "'sound is the interface': from interactive to ecosystemic signal processing," *Organised Sound*, vol. 8, no. 3, pp. 269–277, 2003.

[11] S. Waters, "Performance ecosystems: Ecological approaches to musical interaction," *EMS: Electroacoustic Music Studies Network*, pp. 1–20, 2007.

[12] V. Lazzarini, D. Keller, and M. S. Pimenta, "Prototyping of ubiquitous music ecosystems," *Journal of Cases on Information Technology*, vol. 17, no. 4, pp. 73–85, 2015.

[13] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 457883, 2019.

[14] M. Velankar and P. Kulkarni, "Music recommendation systems: overview and challenges," *Advances in Speech and Music Technology: Computational Aspects and Applications*, pp. 51–69, 2022.

[15] L. Turchet, "Entangled Internet of Musical Things and People: A more-than-human design framework for networked musical ecosystems," *IEEE Transactions on Technology and Society*, 2024.

[16] Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson, "The music ontology," in *Proceedings of International Society for Music Information Retrieval Conference*, 2007.

[17] L. Ding, L. Zhou, T. Finin, and A. Joshi, "How the semantic web is being used: An analysis of FOAF documents," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, 2005, pp. 1–10.

[18] J. Hobbs and F. Pan, "An ontology of time for the semantic web," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 1, pp. 66–85, 2004.

[19] J. Hastings, W. Ceusters, B. Smith, and K. Mulligan, "The emotion ontology: enabling interdisciplinary research in the affective sciences," in *International and Interdisciplinary Conference on Modeling and Using Context*. Springer, 2011, pp. 119–123.

[20] L. Turchet, F. Antoniazzi, F. Viola, F. Giunchiglia, and G. Fazekas, "The internet of musical things ontology," *Journal of Web Semantics*, vol. 60, p. 100548, 2020.

[21] L. Turchet, P. Bouquet, A. Molinari, and G. Fazekas, "The smart musical instruments ontology," *Journal of Web Semantics*, p. 100687, 2022.

[22] G. Fazekas and M. Sandler, "The Studio Ontology Framework," in *Proceedings of the International Society for Music Information Retrieval conference*, 2011, pp. 24–28.

[23] N. Sakimura, J. Bradley, and N. Agarwal, "Proof key for code exchange by oauth public clients," Tech. Rep., 2015, (No. rfc7636).

[24] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, P. Herrera Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. Zapata González, and X. Serra, "Essentia: An audio analysis library for music information retrieval," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2013, pp. 493–498.

[25] L. Turchet, J. Pauwels, C. Fischione, and G. Fazekas, "Cloud-smart musical instrument interactions: Querying a large music collection with a smart guitar," *ACM Transactions on the Internet of Things*, vol. 1, no. 3, pp. 1–29, 2020.

[26] L. Turchet, "Smart Mandolin: autobiographical design, implementation, use cases, and lessons learned," in *Proceedings of Audio Mostly Conference*, 2018, pp. 13:1–13:7.

[27] L. Turchet and C. Fischione, "Elk Audio OS: an open source operating system for the Internet of Musical Things," *ACM Transactions on the Internet of Things*, vol. 2, no. 2, pp. 1–18, 2021.

[28] K. Tsioutas, G. Xylomenos, and I. Doumanis, "Aretousa: A competitive audio streaming software for network music performance," in *Audio Engineering Society Convention 146*. Audio Engineering Society, 2019.

[29] C. Werner and R. Kraneis, "UNISON: A Novel System for Ultra-Low Latency Audio Streaming Over the Internet," in *2021 IEEE 18th Annual Consumer Communications Networking Conference*, 2021, pp. 1–4.

[30] A. Carôt, C. Hoene, H. Busse, and C. Kuhr, "Results of the fast-music project—five contributions to the domain of distributed music," *IEEE Access*, vol. 8, pp. 47 925–47 951, 2020.

[31] J. Cáceres and C. Chafe, "Jacktrip: Under the hood of an engine for network audio," *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, 2010.

[32] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via lola: Low latency high quality a/v streaming system," in *International Conference on Information Technologies for Performing Arts, Media Access, and Entertainment*. Springer, 2013, pp. 240–250.

[33] L. Turchet and J. Pauwels, "Music emotion recognition: intention of composers-performers versus perception of musicians, non-musicians, and listening machines," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 305–316, 2022.

[34] M. Kranz, L. Roalter, and F. Michahelles, "Things that twitter: social networks and the internet of things," in *What can the Internet of Things do for the Citizen (CIoT) workshop at the eighth international conference on pervasive computing (Pervasive 2010)*, 2010, pp. 1–10.

[35] J. I. R. Molano, J. M. C. Lovelle, C. E. Montenegro, J. J. R. Granados, and R. G. Crespo, "Metamodel for integration of internet of things, social networks, the cloud and industry 4.0," *Journal of ambient intelligence and humanized computing*, vol. 9, pp. 709–723, 2018.

[36] M. Fire, R. Goldschmidt, and Y. Elovici, "Online social networks: threats and solutions," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2019–2036, 2014.

[37] F. Morreale, S. A. Bin, A. P. McPherson, P. Stapleton, and M. Wanderley, "A NIME of the times: developing an outward-looking political agenda for this community," in *International Conference on New Interfaces for Musical Expression 2020*, 2020, pp. 191–197.

[38] K. Dinnissen and C. Bauer, "Amplifying artists' voices: Item provider perspectives on influence and fairness of music streaming platforms," in *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, 2023, pp. 238–249.

[39] L. Porcaro, E. Gómez, and C. Castillo, "Assessing the impact of music recommendation diversity on listeners: A longitudinal study," *ACM Transactions on Recommender Systems*, vol. 2, no. 1, pp. 1–47, 2024.

[40] J. Brusseau and L. Turchet, "Ethics Framework for the Internet Musical Things," *IEEE Transactions on Technology and Society*, 2024.

[41] L. J. Bannon and P. Ehn, "Design: design matters in participatory design," in *Routledge international handbook of participatory design*. Routledge, 2012, pp. 37–63.

[42] J. Sullivan, M. M. Wanderley, and C. Guastavino, "From fiction to function: Imagining new instruments through design workshops," *Computer Music Journal*, vol. 46, no. 3, pp. 26–47, 2022.