Spotify Characterization as a Software Ecosystem

Vinicius J. Schettino
Universidade Federal de Juiz de Fora
Juiz de Fora, Minas Gerais
vinicius.schettino@ice.ufjf.br

Regina Braga
Universidade Federal de Juiz de Fora
Juiz de Fora, Minas Gerais
regina.braga@ufjf.edu.br

José Maria N. David
Universidade Federal de Juiz de Fora
Juiz de Fora, Minas Gerais
jose.david@ufjf.edu.br

Marco Antônio P. Araújo
Universidade Federal de Juiz de Fora
Juiz de Fora, Minas Gerais
marco.araujo@ufjf.edu.br

ABSTRACT

[Context] Software production methods are changing at high rate. Organizations and their former product lines are evolving to software ecosystems, seeking to take advantage of external resources that can aggregate value on their projects, such as seasoned engineers and open-source components. Meanwhile, external developers are looking to bond products with established market share, aiming for niches that may provide innovative business opportunity. [Objective] The objective of this work is to describe and evaluate technical, organizational and social aspects of Spotify, called software ecosystem, regarding previous works. [Methods] Besides the literature review and research about Spotify characteristics, we developed an application that extends Spotify native recommendation and presents information about the tracks listened, such as energy, danceability and popularity. These properties are gathered from the Spotify Web API, bringing forward deepen enlightenment about technical aspects of the platform. [Results] Through the knowledge acquired on this work, we present improvement suggestions for the ecosystem, regarding interoperability aspect.

CCS CONCEPTS

• **Software and its engineering → Software design engineering**; **Reusability**; **Software product lines**;

KEYWORDS

Software ecosystems, software reuse, software engineering

ACM Reference format:
Vinicius J. Schettino, Regina Braga, José Maria N. David, and Marco Antônio P. Araújo. 2017. Spotify Characterization as a Software Ecosystem. In *Proceedings of SBCARS 2017, Fortaleza, CE, Brazil, September 18–19, 2017,* 10 pages.
DOI: 10.1145/3132498.3133836

SBCARS 2017, Fortaleza, CE, Brazil

978-1-4503-5325-0/17/09…$15.00
DOI: 10.1145/3132498.3133836

1 INTRODUCTION

Software ecosystems (or SECOs) are growing as a research field [14]. Some definitions, encapsulating concepts of interaction and symbiosis (e.g. business and digital ecosystems) are proposed. One of the most recent, proposed by Manikas [14] says that a software ecosystem is a "software and actor interaction in relation to a common technological infrastructure, that results in a set of contributions and influences directly or indirectly the ecosystem.".

On software ecosystem, interactions between actors are motivated by the value creation. This value can be, for instance, monetary fee, skills, experience, or software development contributions. Companies can no longer develop an entire software product themselves and still fulfill all customers' demands [26], because that would require expertise in several different fields.

This model extends the product line concept, evolving it to take advantage of external resources and keep competitive traits, while each organization can focus on its own specialty. These resources can be software, people, companies, and technical specifications, as protocols or standards. However, it is important to highlight that SECO emerging from SPL is one approach. Another way is to evolve software products or services towards ecosystem platforms [12].

Spotify[1] is an online audio streaming service, containing more than 30 millions of tracks and 75 millions of active users [24]. Software as a service model (SaaS) ensures that its content is available through mobile, desktop and web apps in exchange for a periodic fee (for premium users) or being exposed to advertising (for free users). The SaaS approach allows users to switch providers without long term contracts or infrastructure bounding costs, as well as smaller costs on distribution, maintenance, and deployment of products [7]. These aspects nourish interaction, both in terms of competition and partnership, between several elements that compose a software ecosystem [3].

As stated by Manikas [14], it is important to provide more in-depth studies about real-world context SECOs, focusing on studying the different aspects of this type of platform. Detailing these studies would allow the identification of similar cases for theory confirmation, improvement, and repeatability. Based on this argument, we proposed to investigate Spotify as an Ecosystem platform. Spotify is the biggest audio streaming service. Its fast growth over the past few years engaged us to understand how they are tackling the SECO challenges, that can be similar to other software companies. We

[1] http://spotify.com/

choose to look for a privately-owned product, to study its scenario and specific challenges.

This work aims to investigate relations regarding the state-of-the-art software ecosystems descriptions and organizational, technical, and social issues described in the literature [6] about SECO platforms.

The disruptive nature of Spotify platform aggregates challenges from the organizational point of view. The streaming model breaks the content distribution paradigm that giants from the audiovisual industry have been keeping for decades. Furthermore, Spotify needs to deal with those big players to bargain authorial rights from famous artists.

Regardless engaging the users to pay for the service and encouraging the content producers to participate, the fees and royalties involved must find a balance. It gets even more challenging when the free users are taken into consideration, that contribute with advertising incomes. At the same time, they are important to the ecosystem, the advantages of the premium users, such as no advertising, offline availability and high audio quality must be attractive enough to boost new premium users. This balance is one of the social aspects briefly addressed in this work.

Lastly, some technical challenges are considered here. The interoperability level between the several agents is critical for the ecosystem success [3]. Furthermore, the rapid growth of the service, as the user base went from 10 to 100 million active users in five years [28].

Deepening into the technical point of view, it is presented a small application, named KnowYourBeat, that integrates with Spotify Web API and extends the platform features, adding value to the users on a specific niche.

We hypothesize that the use of KnowYourBeat does extend the Spotify native recommendation system in order to characterize Spotify as a software ecosystem platform. Based on this hypothesis, our research question is "How KnowYourBeat extends Spotify native recommendation system, in order to find new tracks to listen?"

Therefore, as contribution, this work present Spotify characterization as a software ecosystem, along with suggestions that may improve interoperability issues.

This paper is organized as follows: section 2 presents related works and fundamental concepts and theories supporting this research. Section 3 reports the tools and methods that supported this research. Section 4 discuss this concepts on Spotify scenario, structuring its business model in the light of software ecosystems. Section 5 is about the developed application in order to better understand Spotify's ecosystem. There are presented architectural, functional and conceptual decisions made, and how they are related with ecosystem's infrastructure. Whereas in section 6 we discuss the found the constraints and suggestions, in section 7 we provide a closure for this work, also indicating future paths of related research.

2 RELATED WORK

The ecosystem characterization is conducted mainly in adherence with the framework proposed by Santos [6], where three reflection dimensions of a software ecosystem are described: technical, organizational and social.

The technical perspective focuses on an internal view of the ecosystem, mainly on its platform. Software architectures, evolution guidelines, code maintainability among other internal metrics can be discussed, regarding their influence over the ecosystem. Interoperability between the ecosystem components is a technical challenge, mainly for the platform. It has to provide a stable and flexible interface, while avoiding opportunities for defective or malicious external code to affect the whole system [3].

The ecosystem needs to grow in features, components and interactions. Old features tend to be less differentiating, so new features are necessary to keep advantage over other ecosystems. These architecture transactions need to be carefully designed to minimize the impact, especially if it involves interfaces with other components. New features and its interface changes need to be presented long before the actual release takes place [3].

The organizational (or transactional) perspective includes business level interaction challenges between ecosystem components. The ecosystem must offer advantages to new incomers, such as advanced infrastructure or competitive advantage among its competitors, or it will not be attractive. However, these needs must be equated with niches prospection and opportunities to innovative solutions, or it can be seen as a competitor that will not allow a mutual relationship of growth. In this point of view also are included challenges with external regulatory bodies and stability risks, that may hinder new incomers.

On the social point of view, challenges as requirements gathering, extensibility, feedback capture and reputation are discussed [13]. How people see the ecosystem can influence how willing they are to actively participate, through use, feedback and promotion. This participant's perspective goes beyond the product itself and can encompass branding, social actions and even environment positioning.

Although these dimensions presented above are used to categorize the discussed topics, the challenges are not exclusive of each point of view. In software ecosystems, as it happens in nature, the impacts of decisions are felt in other dimensions, being unfeasible to study them in an isolated way.

To support the terms used in this paper, two exploratory studies are resorted. Jansen and Cusumano [12] suggest, among other proposals, some classification parameters for platforms, taking into consideration projects such as Spotify itself. Through a systematic mapping, Manikas [14] suggests state-of-the-art definitions for the field. Concepts proposed by Christesen et al. [5] are considered on this systematic mapping and are especially useful for the characterization and description aimed at this work.

3 METHODS AND TOOLS

In order to characterize different aspects of Spotify's SECO, we used some different approaches, each one was focused on specific traits. Although we understand that using assessment models proposed in previous works [6, 26] would be more suitable, there are problems with this approach. Since Spotify is a proprietary platform, we do not have access to all data necessary, especially when it comes from the development process statistics and information.

We propose to use previous assessment models, limited to the data we could gather from public sources, such as official documentation and specialized media. These sources were useful to understand technical, social and organizational traits of the SECO, to characterize it towards the literature already presented in this paper.

To deepen our knowledge about technical aspects, we propose an application that extends features from Spotify native recommendation service, with its data and public API. For evaluating the application's capacity of adding value through participation in Spotify's SECO, we conducted a Case Study.

4 CHARACTERIZING SPOTIFY ECOSYSTEM

As a matter of organizational structure, there are traits discussed in the literature [14] that support SECOs characterization, inspired by IT governance research field. It investigates software as a mean to support business in a corporation [27]. More specific, Manikas et al. [15] discuss ecosystem orchestration, or who make the decisions and strategic leading of the platform:

Monarchy: All the orchestration is made by a single actor, typically the one with great influence over the main technology of the ecosystem.

Federal: There is a set of representative actors that conduct the orchestration together.

Collective: All actors can participate the decision process, through practices as polls.

Anarchy: Each actor acts on its own, and there is no general orchestration.

Spotify can be seen as a monarchy ecosystem since the main decisions are held by the platform itself. It can lead to an easier decision-making process, but leaves out opinions and visions that may be helpful for the ecosystem health. Also, the monocratic model can increase the social challenges discussed in section 2.

Over the perspective of business structure, the ecosystem may be analyzed by its means of creating value [14]. Thus, they can be characterized as:

Proprietary: The main value creation is made by proprietary contributions. The components may have a common interface for exchange of data and features, but they are typically protected by intellectual property (IP) management processes. In this scenario, the values exchanged often involve monetary fees.

Open Source: The contributions are generally open to the public. The main value types are knowledge, experience and satisfaction, whereas the monetary value decreases in participation.

Hybrid: TThere are cases where the ecosystem supports both proprietary and open source contributions, with the correspondent compensations associated.

Spotify can be seen as a Hybrid ecosystem, since the platform is proprietary but the partner services can be both proprietary or open source, as long as they implement a common interface. With future strategic decision, for instance, there can be fees for APIs access, open source modules or additional SDKs, taking the ecosystem to a different business structure.

4.1 Platform Classification

The platform is the base technology behind an ecosystem. In this work we will consider mainly the interaction between its components. Considering Jansen and Cusumano [12] classification, Spotify is considered a service platform, privately owned and coordinated.

The platform is the base technology behind an ecosystem. In this work, we will consider mainly the interaction between its components. Considering Jansen and Cusumano [12] classification, Spotify is considered a service platform, privately owned and coordinated. Extension Market is a component present in some ecosystems [12], and it aims to support niche players to publish and even sell their products. Some examples are Firefox[2] and Slack[3], where the extension market is an appeal for external developers to interact with the ecosystem. Spotify had this service, called "App Finder", but it was discontinued[4]. To publish in the App Finder and even get access to the APIs, the application would undergo an evaluation by Spotify. This service was removed when new SDKs and APIs were developed and made available to the general public.

An extension market can be seen by external developers as an attractive, since it supports the contact between several ecosystem's actors, such as content producers and user. In this same space, it is possible to showcase successful applications and to stress unexplored niches, increasing the ecosystem's advantages visibility.

The closer service to a extension market available now is the "Case of Success" section provided by Spotify, where selected applications are displayed for the developer community for inspirational and motivational purposes [22].

4.2 Niche Players

Niche players are actors inside an ecosystem that interacts with the platform and among themselves, influencing its evolution. The niche players aim to distinguish their services and products from competitors, that can be in the same ecosystem. They are classified in three major groups[26]:

Influencers. : They are responsible for interventions on the key organization and their strategies and main politics. Generally, are early adopters, developing its technology with a fundamental role played by other ecosystem components.

Hedgers. : They present solutions for interacting with several platforms and ecosystem, not being strongly attached to a single component.

Disciples. : They are compromised to only one platform and impute a fundamental role to it on their products and services. Usually, one does not have the same key role in the ecosystem strategies as seen on the influencers.

To find and analyze Spotify's niche players, official sources [19, 20], and specialized media [16, 25] were addressed. The main niche players founded, excluding the already discussed platform users, are:

4.2.1 Record Labels. The record labels are the main audio providers to the platform. Thus, they can influence the platform's business

[2] https://addons.mozilla.org/pt-BR/firefox/
[3] https://slack.com/apps
[4] https://developer.spotify.com/technologies/apps/

model and strategic decisions. In exchange for the authorial content for streaming, they receive a financial compensation (royalties) based on the amount of streaming. In addition, the record labels diversify the logistics of distribution and disclose of its material. The origin and values diversity of these compensations are discussed in section 4.3.

The copyright protection has been a challenge in audio and video content in the internet era. The Spotify business proposal can be considered an alternative that, even possibly less lucrative, faces the charms of piracy, such as lower costs and distribution facilities.

4.2.2 Independent Artists. Independent Artists. Content providers without the support of major record labels are increasingly common, thanks to the accessibility of technologies. Today, it is possible to produce high-quality audio content with a fraction of what was needed some years ago. However, the traditional distribution logistics and publicity still supported by the major record labels.

In this scenario, the Spotify can be seen as an alternative platform of audio content host. Since the gains are bounded to the amount of streaming, small producers can finance undergoing projects and find a stable way of publicizing their work. They can be seen as disciples or hedgers, depending on how they use the platform.

4.2.3 Developers. Spotify holds more than 30 million of tracks and 75 million of active users [24], as well all the metadata associated with these records. This can add value in several forms for different services related to music. This data is available through APIs and SDKs maintained and documented by the platform. In counterpart, the eventual success of third-party's applications increases the Spotify market share and its brand value. They can be seen as disciples or hedgers, depending on how they use the platform.

Although the discontinuity of the central extension market, it is possible to find some third parties applications that claim to extend the Spotify services and features. For example, the Djay[5] and Waze [6][7]. On the unofficial application discover site AppCrawlr it is possible to list 18 Android[8] and 24 iOS[9] apps that claim some kind of integration with Spotify.

4.3 Monetization Model

Spotify holds two main ways of raising funds: Premium users' payment and advertisements circulation. The first is converted through a monthly fee related to a chosen subscription plan [10]. The former is accessible from the free users, that are subjected to advertisers between tracks. Depending on the origin, the income associated to the streaming is different [21].

Even though a record label is not mandatory for keeping tracks on the Spotify, independent authors need an intermediary for it [21]. There are some options available, and the bargain over the royalties' percentage must be discussed with the intermediary.

[5]https://www.algoriddim.com/spotify
[6]https://support.google.com/waze/answer/7341361?hl=pt-BR
[7]http://tecnologia.ig.com.br/2017-03-14/waze-spotify.html
[8]http://appcrawlr.com/app/search?q=spotify+integrationdevice=android
[9]http://appcrawlr.com/app/search?q=spotify+integrationdevice=iphone%2Cipad
[10]https://www.spotify.com/br/subscriptions/

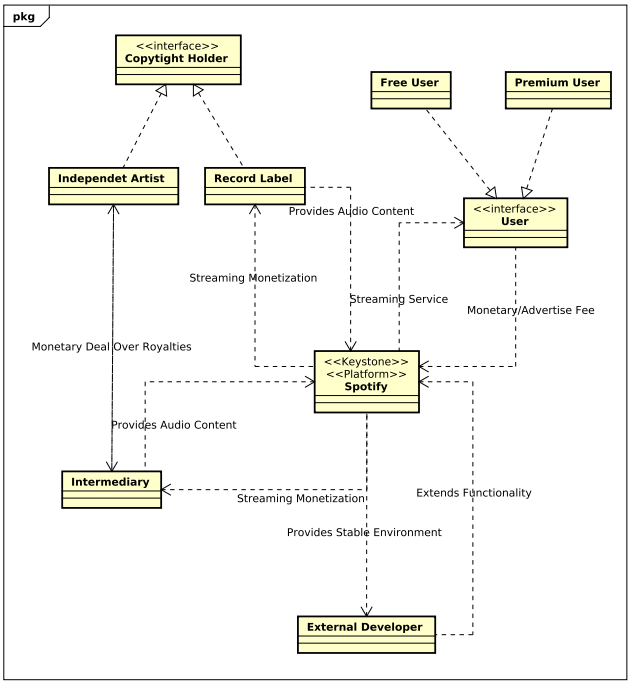

Figure 1: Interaction Between Spotify's ecosystem components

4.4 Ecosystem Interaction

Figure 1 describes the interaction between Spotify1s main agents. The communication occurs primarily over the platform. The content providers can be either Independent Artists or Record Labels, but the first needs an intermediary to publish their content They receive, from Spotify Platform, royalties based on the amount of streaming. There are free and paid users. The last pay a monthly fee for the service, and the former pays through advertising fees.

The developers, through the available APIs, offer extensions and niche appliances of the Spotify services, adding value to its products, improving the Spotify brand value and market share and increasing the amount of data available.

4.5 APIs and Niche Players Interface

The communication between the applications and the ecosystem are mainly conducted over the APIs available by the Spotify[11]. There are SDKs for mobile (Android and iOS) and web applications, the Play Button (meant to be incorporated in blogs and social pages) and integration tools for Internet of Things (IoT) devices.

The web API endpoints are mostly compliant with the REST approach. According to Fielding and Taylor [8], REST is a "coordinated set of architectural constraints that attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations". Web Services that implement these approaches are called popularly as RESTful services.

[11]https://developer.spotify.com/

When a resource is fetched, the Spotify API also include hypertext links to related resources. For instance, on a track, there are links to the album and the artist. However, this feature does not adhere with the "Hypermedia as the Engine of Application State" (HATEOAS) proposed by the REST approach [8]. This standard advertises that a RESTful API needs to be machine navigable. Thus, it must be possible to discover other actions and resources that may be reachable from the current resource and its current status. Furthermore, accordingly Richardson Maturity Model for APIs, HATEOAS is necessary to reach high levels of discoverability and interaction [9, 17].

To access this APIs, it is mandatory to have a previous authorization protocol [23]. Endpoints that do not provide personal data from the users only asks for the third parties applications credentials, that must be registered on the Spotify Developer's area[11].

However, to access user data, such as streaming history, personal information and playlists, it is necessary to follow the OAuth 2.0 protocol, described in RFC-6749 [11]. Through this flow, presented in Figure 2, it is possible to be granted with access credentials limited to a well-defined scope. For instance, the user can allow access to his name and email, but not to its birth date. The authorization must be refresh under defined time boxes, but it does not require user direct interaction.

4.6 Discussions about Spotify's Characterization

To be characterized as a SECO, a set of interacting components of Spotify must overcome challenges from different points of view, in order to create benefits for those involved. A successful ecosystem provides an adequate environment that provides competitive advantages to all participants, especially in comparison with outsiders [4]. In this scenario, Spotify presents some strategies and interaction models that characterize it as an ecosystem, and they will be discussed further in this section.

4.6.1 Niche Creation: Spotify does not offer all predictable value inside the platform. It focuses on main features related to audio content streaming and discovery for mainstream public, but it let specific domains such as DJs, musicians and audiophiles need to be handled by external developers.

4.6.2 Platform Value Addition. a SECO that does not provide enough value on its keystone will not be seen as attractive as other ecosystems. This may commoditize the ecosystem and cause disruption of the platform and the keystone firm over time [4].

4.6.3 Functionality Evolution. Functionality Evolution. Spotify does add new features in order to avoid commoditized functionalities only on the platform. In the API changelog[12] it is possible to see that new features are added, some of them based on community requests [18]. In an ecosystem, this innovation is necessary in order to keep competitive advantages over external agents and increase SECO's attractiveness [3].

4.6.4 Public APIs. The public APIs provided by Spotify follow designs and standards in order to increase interoperability, as discussed in subsection 4.5. It also allows client inputs that tailor the outcomes, like pagination, sorting and filtering. This kind of functionality enables new client features without high amounts of computational resources, like network capacity or processing power, thus offering new niches and value addition.

4.6.5 Security Concerns. As discussed by Bosch [3], the platform is the main responsible to avoid malicious code to infect all the ecosystem and its data, since usually it is the main communication channel on a SECO. The Spotify Web API shows concerns on security and privacy, for instance through HTTPS connections and OAuth 2.0 protocol, as discussed in subsection 4.5.

4.6.6 External Developers Feedback Channel. Feedback from ecosystem's components is a valuable input that the keystone organization must gather and consider for decision making process [13]. Spotify uses Github as a public issue tracker[13], encouraging enhancement and feature requests from external developers.

Another way of analyze the SECO is to understand its keystone company and how balanced its strategy is with the challenges faced on a ecosystem business model. Bosch [4] says that there are five levels on how ecosystem engaged is an organization. We will briefly present each of these stages.

In the beginning, the company is i*internally focused*. It only participates on an ecosystem because it needs to, e.g. they need services or products that internal production is unfeasible. Then, it becomes an *ad hoc ecosystem engagement*, where it finds out SECO's advantages, primarily from cost perspective. It suspiciously starts to seek for external components, trying to forge partnerships for specific tasks, incapable to link demands between different relations as e ecosystem. In this level, the company avoids high-level dependency over external developers and lack governance techniques to manage the ecosystem.

When the SECO's advantages become more widespread, the company reaches the *Tactical ecosystem engagement* level. More contextual areas start to considering outsourcing due to a change on the organization's culture. Then it becomes a *Strategic single ecosystem management*, where being on a ecosystem starts to being part of strategic decisions. It sees the ecosystem as a whole, broad seeking advantages of its structure.

The last level an organization can reach consists to thinking itself as part of several ecosystems, as it usually happens to big players [4]. This stage is called *Strategic multi-ecosystem management*. From this point, the company tries to leverage competitiveness by interacting in all its ecosystems at the same time with win/win perspective rather than a traditional win/lose lens.

In our analysis, Spotify still figures as an *ad hoc ecosystem engagement*. It already takes responsibilities as a platform and ecosystem keystone by facing the challenges discussed above. On the other hand, the presented aspects show that there are more than monetary contributions, but it still lacks ecosystem governance tactics to increase fidelity and nourish long term relationships.

Some of the discussed topics need an insider perspective to be explained, especially those related to technical traits. Furthermore, a deepened point of view helps us to find challenges and weak spots against the theoretical content we faced. For these reasons, we developed an application inserted on the present SECO. By this

[12] https://developer.spotify.com/web-api/change-log/

[13] https://github.com/spotify/web-api

approach, we expect to enhance our knowledge about the ecosystem and observe possible improvements that could be made. Bosch[4] says that traditional way of thinking puts a company on only one ecosystem.

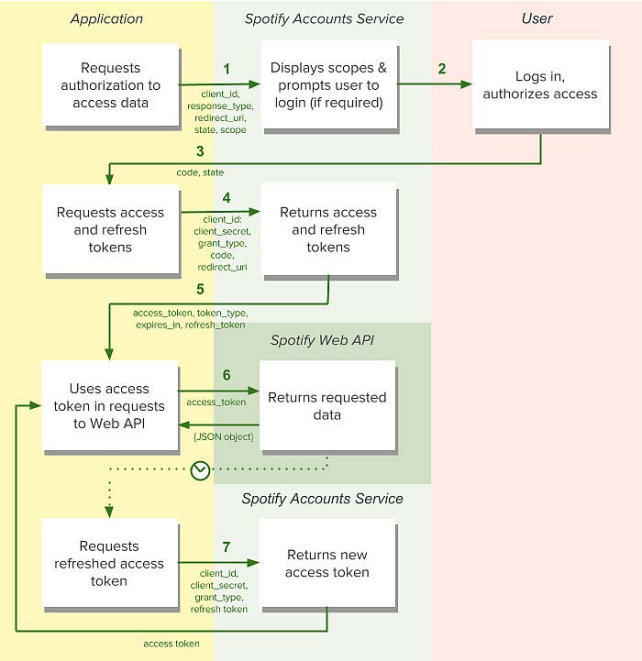

Figure 2: Authorization Code Flow [23]

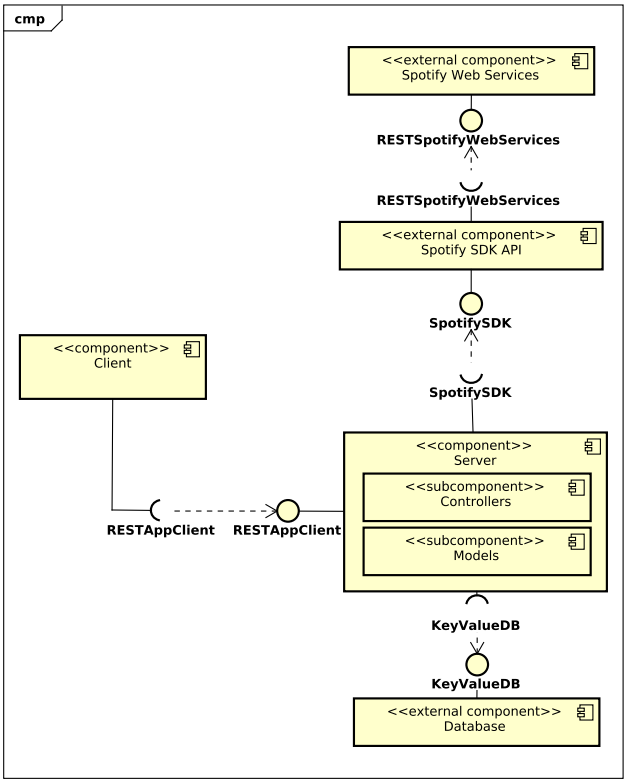

Figure 3: KnowYourBeat Component diagram

5 KNOWYOURBEAT

To better understand the interactions and relationships between niche players and the platform, KnowYourBeat was developed. It helps user to understand his musical taste and find songs he may like. It extends native recommendation subsystem on Spotify, letting the user seek audio properties such as energy, valence and popularity. This data is gathered through Spotify API and made accessible directly for the user, feature which is not currently available the platform. This track metadata come from the recent streaming history of the user, by a new endpoint [18]. The main features are (i) display recent tracks and aggregated statistics such as popularity, energy and danceability, (ii) display top artists from the current user and their average popularity, (iii) display the most popular artist between the current user's top artists, (iv) display recommendations based on user's top tracks and artists and the mean audio features found over those tracks.

Although the Spotify native recommendation system is straightforward to use, since it requires no direct input, this feature can also be considered a weakness. Users have no way to understand the parameters that will be used, therefore cannot narrow the recommendation to only new data, or to be based on specific aspects as energy or danceability. The importance of user input in recommendation system context has been already explored in the literature [2]. Thus, we improved Spotify's recommendation system with the new (user input) feature.

This scenario can be seen as a niche for KnowYourBeat, since it fulfills a peculiar group of individuals needs who want tailored recommendations, even it demands more interaction and user manual input.

The Spotify API allows the client application to defined tailored seeds to be used on the recommendation system. It can be any combination of tracks, artists and genres, as long as it summarizes a set of five elements. It is also possible to set minimal and maximal intervals of audio features, as well as target values for these fields. In this last case, tracks with closest values will be returned.

We propose a recommendation model where the user's favorite artists and recently played tracks are used for seeds, as well as for target values. This might give a reasonable balance between adequate results (because of the top artists) and a tailored suggestion based on the present user's mood.

5.1 Architecture

KnowYourBeat have two main components. The first is a server application that consumes the Spotify Web API, transforms and filters it, keeping the authentication and authorization data. The second is a client application that interacts with the user, rendering data and collect user inputs. Figure 3 shows the relation between these components and the Spotify services.

The server application is a RESTful service mostly written in PHP7, on the top of Yii2 Framework[14]. The communication with the API is supported by a PHP SDK for Spotify[15]. The choice for this technology, among others same purpose libraries is justified by the recent development activity (making available new endpoints integration) and higher PHP dependencies (indicative of taking advantage of new language features). The server code is available on Github[16].

To store the authorization data, Redis 3.2.9 is used. It is a key-value data store with optional persistence, that can be reached in different forms. It is provided by a Docker Container[17], abstracting some installing and management details of the tool.

The client is mostly written in Javascript, on the top of React[18], a library for build user interfaces. React is a component based library, where the developer is encouraged to implement small and reusable components, from himself and from open source projects. For instance, all the interface components (tables, menus, drawers) were extended from the Material UI set of react components. They are based on Google's Material Design, a guideline for interface design that are behind google products and serves as a foundation for many other design standards over the industry [10].

This architecture was design to allow changing the components when needed. It is possible, for instance, to change the client or the database for other components that offer the same interface. The REST approach also enables that each component implements its own languages or paradigms internally, but still communicating whereas the standard interfaces are provided.

6 EVALUATION

This section presents an initial evaluation of KnowYourBeat and its capability of extend native recommendation subsystem on Spotify. It presents the method used to evaluate the hypothesis, and finally the results of this evaluation. KnowYourBeat was evaluated through a case study. This study was specified to demonstrate the technical feasibility of the KnowYourBeat service, and also, concepts and technologies involved in the solution. Therefore, it was designed to be a testbed to reveal improvements that could be done and flaws that must be addressed.

6.1 Study Definition

We aimed to perform an initial evaluation of KnowYourBeat and its capability of extending native recommendation subsystem on Spotify platform. Based on the Goal/Question/Metric (GQM) [1] approach, the objective was defined as follows: "**Analyze** the use of KnowYourBeat **for the purpose of** verifying its capacity of extending native recommendation subsystem on Spotify **from the point of view** of a user **in the context** of finding new tracks to listen."

From the scope definition, the research question is: How Know Your Beat extends Spotify native recommendation system, in order to find new tracks to listen?

[14]https://github.com/yiisoft/yii2/
[15]https://github.com/jwilsson/spotify-web-api-php/
[16]https://github.com/facebookincubator/create-react-app
[17]https://hub.docker.com/_/redis/
[18]https://facebook.github.io/react/

In order to collect evidences about the research question, direct observations were used as data sources.

6.1.1 Study Planning.

6.1.2 Context Selection. This study was carried out, considering a real-world context were a Spotify user wants to find new songs he may like.

6.1.3 Hypothesis Formulation. The evaluations were conducted bearing in mind the following hypothesis. (i) H0 (Null hypothesis): The use of KnowYourBeat does not extend the Spotify native recommendation system. (ii) H1 (Alternative hypothesis): The use of KnowYourBeat does extend the Spotify native recommendation system.

6.1.4 Goals. This case study aims to evaluate if KnowYourBeat helps Spotify to recommend new songs to listen, based on his current mood. This mood can be roughly translated at the audio features of his recent streaming history, such energy, positiveness, or danceability.

6.1.5 Subject Characterization. The subject is a male, 23 years old, student, Spotify premium user that declares to use the platform, on average, between 2 and 3 hours every business day. He will be identified as A, from now on. He was selected based on his knowledge and familiarity with Spotify. The central criterion for participant selection was his knowledge on Spotify functioning and execution.

6.1.6 Scenario and Results. This subject A wants recommendations of new songs to listen, based on his current mood. This mood can be roughly translated at the audio features of his recent streaming history, such energy, positiveness or danceability.

In order to use KnowYourBeat, subject A was required to log in with his Spotify account. This is achieved with the OAuth 2.0 protocol (as explained in subsection 5.1), so the client application does not have access to his access credentials. Instead, the request describes which data is needed, and it is granted by Spotify upon user confirmation.

Once properly authenticated and conscious of his grant over his data, subject A navigates through visualizations regarding his musical taste. In Figure 4, his recent streaming record is partially shown, along with average audio features of this history. This information reveals peculiar patterns such as relatively low popularity rate and energy of his recently played tracks.

Artists have tender relations with genres and moods. So, we choose to display user top 20 artists, not based only in the recent history, as partially shown in Figure 5. This data is also gathered on Spotify Web API, and the most popular artist and the average popularity of the artists is found and displayed.

Subject A can also have access to a list of 20 recommendations, based on the parameters explained at subsection 5.1. The partial list is represented on Figure 6.

He declares that his recommendations are tailored for his actual mood, but they are still attached to known artists. This conclusion indicates that the provided application explores a niche created by Spotify SECO, where users can personalize recommendation with recent history input.

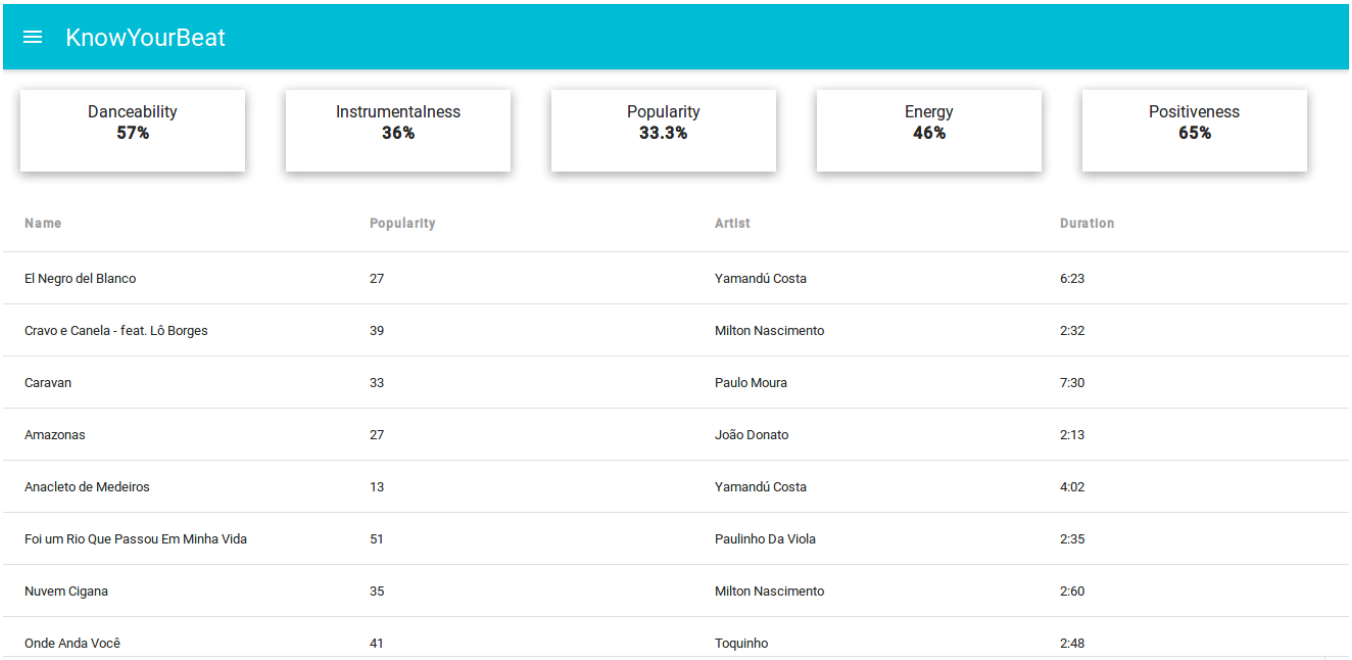

Figure 4: User A's recent streaming page

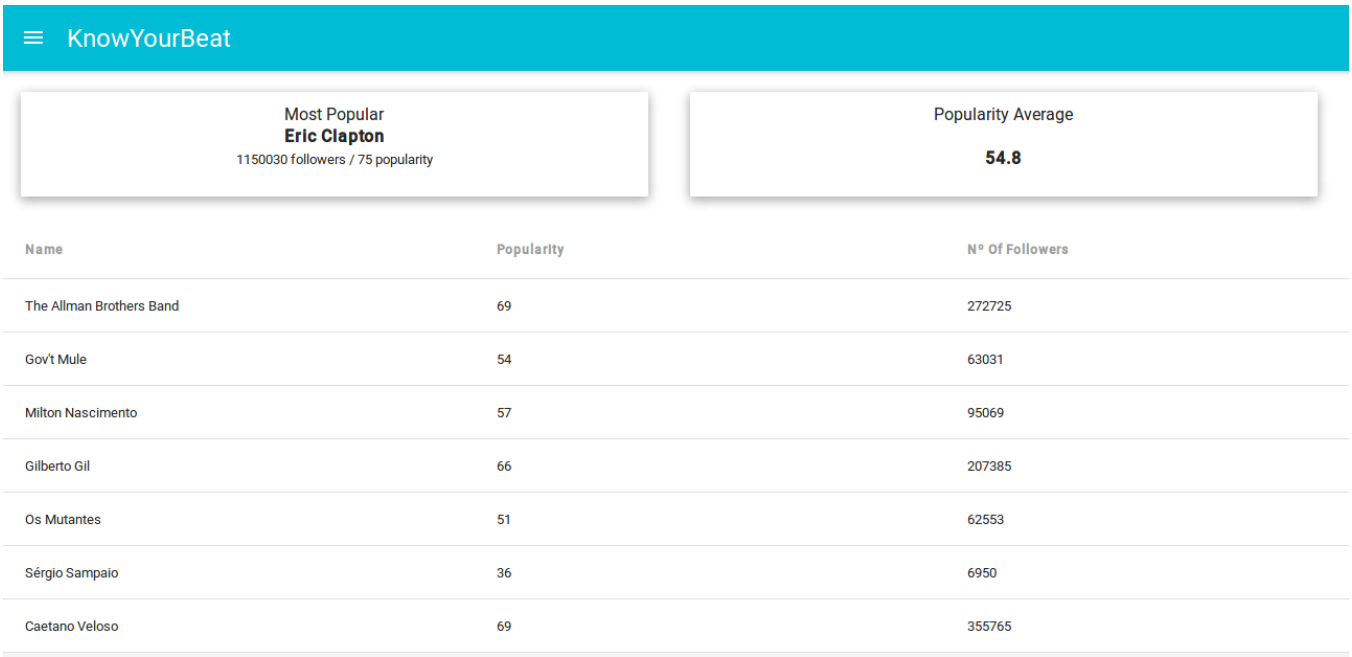

Figure 5: User A's top 20 artists page

OAuth 2.0 protocol allowed proper authorization without client access to the user credentials. It decreases the withdrawal factor that involves placing user's password in an unknown application, as well as drops security levels that client would need to have for manipulating and storing this kind of data.

Sorting and pagination features provided by the web API were used considering user interaction with minimal effort by the client application. Subject A asked about genre information on tracks,

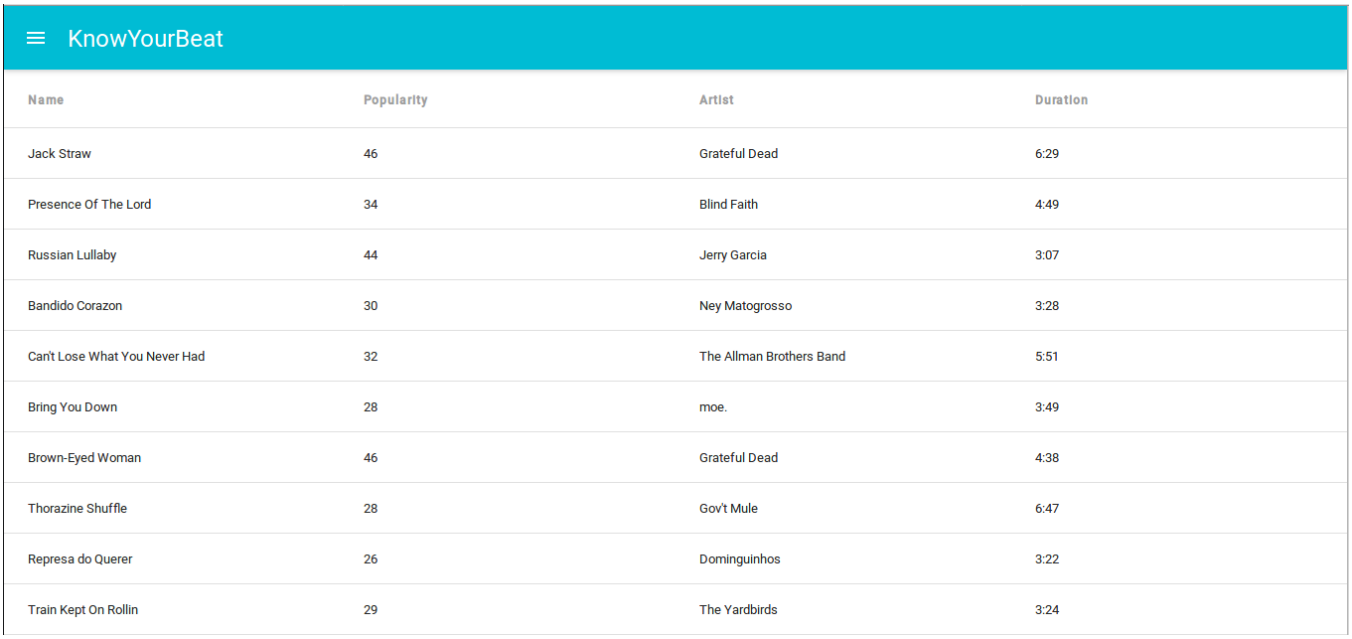

Figure 6: User A's recommendations page

both on recent history and recommendations page. This information is not provided by Spotify API, thus cannot be displayed for the end user.

Subject A asked about how he could find similar applications, especially when visualizing relations between artists, tracks, and genres In the lack of a extension market, according to the Google Play model for instance, we presented the "Case of Success" section [22] for him.

6.2 Analysis of Results

Considering this case study, we analyzed the results from the user's point of view. This research evidenced that Spotify can be considered a software ecosystem, regarding definitions and aspects presented and discussed in the related work [3–5, 12, 14]. As a SECO, the presented challenges must be addressed in constant evolution.

Through this evaluation, we have shown how KnowYourBeat can find new tracks to listen, raising evidences to answer the research question: How KnowYourBeat extends Spotify native recommendation system, in order to find new tracks to listen? However, they cannot be generalized through a single case study. Rather, we identified situations in which similar results could be obtained. Thus, considering our hypothesis, there is evidence that KnowYourBeat is feasible and its use can improve Spotify SECO through the recommendation service. Therefore, there is evidence that the null hypothesis (H0) could be rejected and the alternative hypothesis (H1) could be accepted.

In addition, we can draw suggestions to enhance Spotify platform, considering SECO dimensions, which is our main goal. The first suggestion is about the lack of extension market, discussed at subsection 4.1. This kind of structure is attractive to newcomers, since it supports the contact between the product and the users. Also, this space can serve as showcase for successful applications and unexplored niches, that may increase the ecosystem visibility and attractiveness. We recommend that the provided "Case of Success" section [22] grow to a wild access extension market.

Second suggestion regards the lack of HATEOAS [8] standard that allows automated feature and data discover, as discussed on subsection 4.5. For instance, with suitable linkage and description between a track representation resource and the action of adding the said track on a playlist, a client application could discover this feature and present it on a user interface automaticaly. This is a pattern found in high-level maturity APIs [9, 17], and especially useful regards the interoperabilty aspect. That said, we suggest the adition of a HATEOAS standard over the provided endpoints of the Spotify Web API.

Finally, there are additional audio content streaming services, such as TIDAL[19] and Apple Music[20] that also have public APIs for external developers. These APIs use data and services to generate value for their ecosystems with applications such as KnowYourBeat. However, there is no common standard over these APIs. This scenario is going against the interoperability issues and aggregated value, since it demands client application to implement different integration solutions to attend to different users. In this scenario, Spotify could leverage its ecosystem strategic engagement level, as discussed in subsection 4.6, looking for interaction support between competitors as a business opportunity.

6.3 Threats to Validity

This evaluation is only valid within the specified context. Additional evaluations are necessary, considering other SECOs contexts.

[19] http://tidal.com
[20] https://www.apple.com/br/music/

Therefore, the results cannot be generalized nor other problems faced in Spotify domain. It is necessary to prepare and conduct additional experimental studies to extend the validity of the hypotheses previously stated.

Another drawback is the number of case studies. Further studies could collect additional evidence that were not observed in the presented case study. Additional experimental studies could also reveal certain aspects that were not explored such as health, robustness and attractiveness.

It is also important to emphasize that these conclusions are drawn from preliminary evidence of the benefits of KnowYourBeat. Thus, it can be noted that the validity of the conclusions is related to the replication of the study in other contexts to ensure the feasibility of the approach. However, it was possible to identify situations in which similar results could be obtained.

7 CONCLUSION

Software ecosystem is a new way of creating software, born from the need of responsiveness over the emergent requirements software products face everyday. Companies start to see that they no longer can develop entire software products, so they reach out of their walls for resources, like expertise and open source components. At the same time they offer stable market share and features where external developers may find a niche where to extend their competitive advantages.

Brought into the light of recent SECO literature, we discussed it structure, such as governance strategies and niche players flavours, and its maturity as platform, keystone organization and infrastructure provider.

Regarding interoperability aspect, we analysed the Spotify ecosystem under the aegis of state-of-the-art literature, to understand which solutions they are proposing for the known challenges this kind of structure brings. Furthermore, the KnowYourBeat development process and constraints gave us a internal view of the ecosystem whereas the faced challenges contributed to this work. This research allows us to suggest evolutions that may extend its robustness and health.

In short, this paper brings a software ecosystem assessment, towards organizational, social and technical aspects, focusing on interoperability. Through the knowledge acquired on this work, especially on KnowYourBeat development, we present improvement suggestions for the ecosystem, regarding interoperability aspect

As future works, there are some aspects that can be deepened. For instance, we pretend to analyze other aspects as reputation, fidelty and niche creation regards Spotify ecosystem. Also we aim for collect empirical data to draw conclusions about this SECO's maturity and robustness. We seek to expand KnowYourBeat to explore further the ecosystem infraestructure, beyond the interoperability aspect.

ACKNOWLEDGMENTS

We would like to thank CNPq, CAPES, FAPEMIG, Jonathan Wilsson[21], creator and maintainer of Spotify Web API SDK for PHP, for his unconscious help to this work. We also thank the reviewers for their valuable contributions.

[21] https://github.com/jwilsson

REFERENCES

[1] V. R Basili and D. M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE Trans. Softw. Eng* SE-10, no. 6 (1984), 728–738.
[2] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.
[3] Jan Bosch. 2010. Architecture challenges for software ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 93–95.
[4] Jan Bosch. 2017. *Speed, Data, and Ecosystems: Excelling in a Software-Driven World*. CRC Press.
[5] Henrik Bærbak Christensen, Klaus Marius Hansen, Morten Kyng, and Konstantinos Manikas. 2014. Analysis and design of software ecosystem architectures–Towards the 4S telemedicine ecosystem. *Information and Software Technology* 56, 11 (2014), 1476–1492.
[6] Rodrigo Pereira dos Santos. 2016. *Managing and monitoring software ecosystem to support demand and solution analysis*. Ph.D. Dissertation. Universidade Federal do Rio de Janeiro.
[7] Abhijit Dubey and Dilip Wagle. 2007. Delivering software as a service. *The McKinsey Quarterly* 6, 2007 (2007), 2007.
[8] Roy T Fielding and Richard N Taylor. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology (TOIT)* 2, 2 (2002), 115–150.
[9] Martin Fowler. 2010. Richardson Maturity Model: steps toward the glory of REST. *Online at http://martinfowler.com/articles/richardsonMaturityModel. html* (2010).
[10] Google. 2017. Material design - Introduction. https://material.io/guidelines/introduction-goals. Accessed 2017-05-24.
[11] Dick Hardt. 2012. The OAuth 2.0 authorization framework. (2012).
[12] Slinger Jansen and Michael A Cusumano. 2013. Defining software ecosystems: a survey of software platforms and business network governance. *Software ecosystems: analyzing and managing business networks in the software industry* 13 (2013).
[13] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. 2009. A sense of community: A research agenda for software ecosystems. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*. IEEE, 187–190.
[14] Konstantinos Manikas. 2016. Revisiting software ecosystems research: A longitudinal literature study. *Journal of Systems and Software* 117 (2016), 84–103.
[15] Konstantinos Manikas, Krysztof Wnuk, and Arisa Shollo. 2015. Defining decision making strategies in software ecosystem governance. *Department of Computer Science, University of Copenhagen* (2015).
[16] Mic. 2016. How Does Spotify Make Money? Here's the Business Model Behind the Streaming Service. https://mic.com/articles/137400/how-does-spotify-make-money-here-s-the-business-model-behind-the-streaming-service. Accessed 2017-05-24.
[17] Leonard Richardson. 2008. Justice will take us millions of intricate moves. In *International Software Development Conference (QCon)*.
[18] Spotify. 2017. New Endpoint: Recently Played Tracks. https://developer.spotify.com/news-stories/2017/03/01/new-endpoint-recently-played-tracks/. Accessed 2017-06-02.
[19] Spotify. 2017. Spotify Artists Blog. https://artists.spotify.com/blog. Accessed 2017-05-24.
[20] Spotify. 2017. Spotify Artists FAQ. https://artists.spotify.com/faq/. Accessed 2017-05-24.
[21] Spotify. 2017. Spotify Artists Guide. https://artists.spotify.com/guide. Accessed 2017-06-02.
[22] Spotify. 2017. Spotify Developers Showcase. https://developer.spotify.com/showcase/. Accessed 2017-06-02.
[23] Spotify. 2017. Understanding the Spotify Web API. https://labs.spotify.com/2015/03/09/understanding-spotify-web-api/. Accessed 2017-06-02.
[24] The Telegraph. 2015. Apple Music vs Spotify: How do the two streaming services compare? http://www.telegraph.co.uk/technology/2016/03/17/apple-music-vs-spotify-how-do-the-two-streaming-services-compare/. Accessed 2017-05-01.
[25] Unicornomy. 2016. Spotify Business Model and How does Spotify Make Money. https://unicornomy.com/spotify-business-model-how-does-spotify-make-money/. Accessed 2017-05-24.
[26] Ivo van den Berk, Slinger Jansen, and Lútzen Luinenburg. 2010. Software ecosystems: a software ecosystem strategy assessment model. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 127–134.
[27] Peter Weill and Jeanne Ross. 2005. A matrixed approach to designing IT governance. *MIT Sloan Management Review* 46, 2 (2005), 26.
[28] Music Business Worldwide. 2016. Spotify is converting more people into paying subscribers than ever before. https://www.musicbusinessworldwide.com/spotify-is-converting-more-people-into-paying-customers-than-ever-before/. Accessed 2017-05-01.